TrustRAG: Enhancing Robustness and Trustworthiness in Retrieval-Augmented Generation

Huichi Zhou[1*], Kin-Hei Lee[2*], Zhonghao Zhan[2*], Zhenhao Li[2], Yue Chen[3], Huaxiu Yao[4], Hamed Haddadi[2], Emine Yilmaz[1]

[1]University College London [2]Imperial College London
[3]Peking University
[4]University of North Carolina at Chapel Hill

Abstract

Retrieval-Augmented Generation (RAG) enhances large language models (LLMs) by integrating external knowledge sources, enabling more accurate and contextually relevant responses tailored to user queries. These systems, however, remain susceptible to various RAG attacks, which can severely impair the performance of LLMs. To address this challenge, we propose TrustRAG, a robust framework that systematically filters malicious and irrelevant content and relies on knowledge resolution to dynamically balance internal and external knowledge, ensuring trustworthy and reliable generation. Our approach employs a two-stage defense mechanism. The first stage implements a cluster filtering strategy to efficiently remove surfaced attack patterns that are similar. The second stage in addition employs a self-assessment process that utilizes the reasoning capabilities of LLMs to resolve both intra-documents fact conflicts and internal-external knowledge inconsistencies, dynamically choosing the source where the final answer is grounded. TrustRAG provides a plug-and-play, training-free module that integrates seamlessly with any open- or closed-source language model. Extensive experiments show that TrustRAG significantly improves efficiency and robustness against diverse RAG attacks, including real-world situations.

Introduction

Large language models (LLMs) in recent years have shown remarkable in-context learning capabilities, and retrieval-augmented generation (RAG) (Lewis et al. 2020; Gao et al. 2023; Chen et al. 2024) amplifies this strength by providing external new knowledge to LLM for more accurate and up-to-date responses. This approach has been widely adopted in real-world applications including ChatGPT (Achiam et al. 2023), Microsoft Bing Chat (Microsoft 2024), Perplexity AI (Perplexity 2024), and Google Search AI (Google 2024). However, despite the success with RAG systems, they are still sensitive to the quality of retrieved documents, thus can suffer from corpus poisoning attacks, where malicious docs are injected to corrupt the knowledge base. Imagine asking an advanced LLM how to manage anxiety and receiving a confident but dangerously incorrect response such as "potassium cyanide". While such misinformation is a serious concern, it reflects a broader and systemic vulnerability in modern AI systems.

Recent work has shown that malicious instructions injected into retrieved documents can override user instructions and mislead LLMs to generate targeted information (Greshake et al. 2023), and query-specific adversarial prompts (e.g., adversarial prefix or suffix) can manipulate both retrievers and LLMs (Tan et al. 2024). PoisonedRAG (Zou et al. 2024) demonstrates how injected malicious documents can induce incorrect responses. Real-world incidents highlight these vulnerabilities in RAG systems, such as the "glue on pizza" fiasco in Google Search AI (BBC 2024). Another instance involved a corpus poisoning attack where ChatGPT retrieved compromised GitHub code, resulting in a $2.5k financial loss (rocky 2024). These incidents underscore the need to address RAG poisoning threats and improving safety and robustness of RAG systems.

Prior work has proposed advanced RAG frameworks, including ASTUTE RAG (Xiang et al. 2024), InstructRAG (Wei, Chen, and Meng 2024) and RobustRAG (Wang et al. 2024), that mitigate noisy information by employing majority-vote mechanisms in retrieved documents and carefully engineered prompts. However, these approaches become ineffective when malicious documents outnumber clean ones (Zou et al. 2024). Even in scenarios with less aggressive poisoning, these advanced RAG systems suffer from noisy or irrelevant content, which significantly hinders them to generate reliable answers.

In this work, as shown in Figure 1, we propose TrustRAG, which consists of two components: Clean Retrieval and Conflict Resolution. We reveal that the optimization setup used by most attackers (Tan et al. 2024; Zou et al. 2024) caused the generated malicious documents to be tightly clustered in the embedding space. Inspired by this finding, in clean retrieval stage we adopt a clustering method based on K-means to help identify the surfaced attacks patterns where malicious docs are grouped together. Meanwhile, to avoid over-filtering clean documents, we combine with ROUGE-L score (Lin 2004) as an additional similarity metric for precise filtering.

After filtering out the clustered malicious docs, we ad-

*These authors contributed equally.

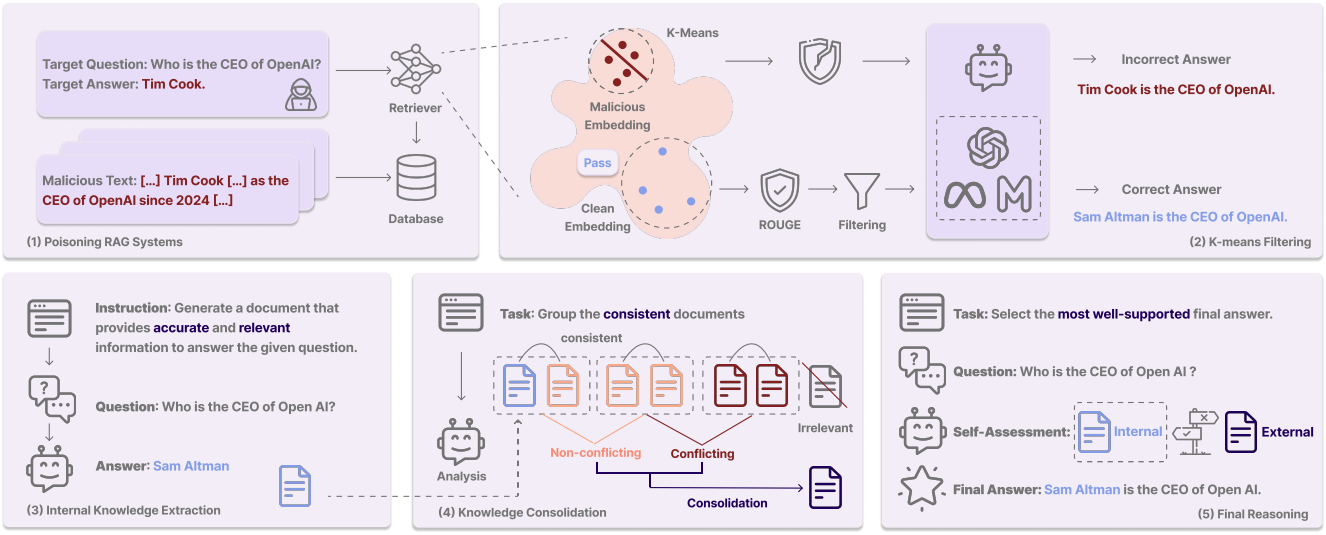

Figure 1: The TrustRAG framework protects RAG systems from corpus poisoning attacks using a two-stage process. In Stage 1 (Clean Retrieval), it (1) identifies malicious documents via K-means clustering and (2) filters malicious content based on embedding distributions. In Stage 2 (Conflict Resolution), 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.

dress the challenge of knowledge conflicts in the second stage, including inconsistency between retrieved documents, and the conflict of facts between internal and external knowledge. Wang et al. (2024) claim that roughly 70% retrieved documents do not contain true answers, causing the impeded performance of LLM with RAG systems. It could even be aggravated in corpus poisoning attacks, since the attacker may induce malicious documents to provide incorrect answers for a target query. Inspired by the findings of Sun et al. (2022), Yu et al. (2022), and Wang et al. (2024) that the internal knowledge of LLM can benefit external knowledge retrieval, we perform the Conflict Resolution stage by leveraging LLM itself to reconcile consistent information, detect conflicts, and discard malicious or irrelevant content. Finally, TrustRAG employs the LLM to choose either its internal knowledge or the externally consolidated information—but not both—to generate the final response, ensuring a clear and trustworthy source of truth.

The key contributions are as follows: (1) TrustRAG is the first framework designed to effectively address **both light and heavy** RAG attacks. (2)TrustRAG reduces attack success rates by up to 80% on average across all tasks, while also improving response accuracy by up to 30%. (3) TrustRAG is evaluated against multiple attack methods and defense baselines, demonstrating its superiority in RAG, and is further validated on two real-world datasets.

Related Work

Retrieval-Augmented Generation. RAG improves the trustworthiness and facticity of LLMs by retrieving relevant information specific to user query from an external knowledge database and grounding LLMs on the retrieved knowledge for conditional generations (Zhou et al. 2024). The common workflow of RAG involves two steps: retrieval and generation (Lewis et al. 2020; Guu et al. 2020; Izacard et al. 2023). With the emergence of LLMs, various techniques have been proposed to improve the ability of RAG, such as query rewriter (Zheng et al. 2023; Dai et al. 2022), retrieval reranking (Glass et al. 2022) and document summarization (Chen et al. 2023; Kim et al. 2024), etc.

RAG Attacker. Recent studies have shown that RAG systems are vulnerable to various attacks. We categorize these attacks into the following three classes. **Corpus Poisoning Attack** (Zou et al. 2024; Zhang et al. 2025; Xue et al. 2024; Tan et al. 2024; Zhang, Zhang, and Shmatikov 2025): This kind of attacks crafts malicious documents by appending deceptive text generated by LLMs. By strategically producing malicious documents and injecting them into the database, the attack ensures these poisoned documents are preferentially retrieved, thereby misleading the language model's outputs. **Prompt Injection Attack** (PIA) (Zhong et al. 2023; Greshake et al. 2023): PIA proposes an attack, in which an attacker perturbs only a few discrete tokens to craft queries that closely mimic legitimate training queries, thereby coercing RAG systems into producing attacker-desired outputs. **Jamming Attack**: Shafran, Schuster, and Shmatikov (2024) adds a single "blocker" document to the database that will be retrieved in response to a specific query and result in the RAG system not answering this query.

RAG Defender. To defend against these attacking methods, recent research has relied on consensus (Xiang et al. 2024; Wang et al. 2024), denoising (Wei, Chen, and Meng 2024; Li et al. 2025), and model's internal knowledge (Sun et al. 2022; Yu et al. 2022; Wang et al. 2024). For example, RobustRAG (Xiang et al. 2024) employs an isolate-

then-aggregate strategy by majority voting retrieved documents to produce the final output. InstructRAG (Wei, Chen, and Meng 2024) enables LLMs to explicitly learn the denoising process through self-synthesized rationales. AstuteRAG (Wang et al. 2024) adaptively elicits essential information from LLMs' internal knowledge, iteratively consolidates it with retrieved knowledge, and finalizes all information to form the final answer. In this work, we propose TrustRAG, which is based on clustering-based document filtering and selective comparison. Different from AstuteRAG, which relies on multi-round refinement, TrustRAG performs single-step internal knowledge extraction fused with external summaries, followed by a binary trust decision. This streamlined approach, coupled with surfaced attack pattern filtering, efficiently eliminates malicious documents and enables faster convergence during knowledge resolution, while also reducing the risk of hallucinations common in multi-step methods like AstuteRAG. Unlike RobustRAG, which relies on majority voting over noisy retrieved documents, TrustRAG proactively filters suspicious sources before generation, focusing on trust assessment rather than aggregation.

Problem Formulation

Threat Model. An attacker selects an arbitrary set of M questions, denoted as $\mathcal{Q} = [q_1, q_2, ..., q_M]$. For each question q_i, the attacker defines a desired (incorrect) answer r_i. For example, the attacker may assign q_i = *"Who is the CEO of OpenAI?"* and r_i = *"Tim Cook"*.

To realize this objective, the attacker injects N malicious documents into the knowledge base for each question q_i. We denote these documents as p_i^j, where $i = 1, 2, ..., M$ and $j = 1, 2, ..., N$. The full set of malicious documents is $\Gamma = \{p_i^j \mid i = 1, \ldots, M; j = 1, \ldots, N\}$.

The attack is successful if the LLM generates r_i in response to q_i when retrieving from the poisoned corpus $\mathcal{D} \cup \Gamma$. This objective can be defined as:

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

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

where $\mathbb{I}(\cdot)$ is the indicator function that outputs 1 if the condition holds and 0 otherwise. The function $\mathcal{E}$ returns the top-k retrieved documents based on a similarity function (e.g., cosine similarity) using encoders f_q for the query and f_t for the documents.

Defense Objective. The goal of a robust defense mechanism is to mitigate the influence of injected malicious content in the RAG pipeline without sacrificing retrieval effectiveness or generation quality.

Given a potentially poisoned corpus $\mathcal{D} \cup \Gamma$, we define a filtering function $\mathcal{F}$ such that the filtered retrieval set $\tilde{\mathcal{E}}(q_i) = \mathcal{F}(\mathcal{E}(q_i; \mathcal{D} \cup \Gamma))$ enables the LLM to produce accurate outputs. The defense objective is to minimize the attack success rate (ASR) while preserving high response accuracy:

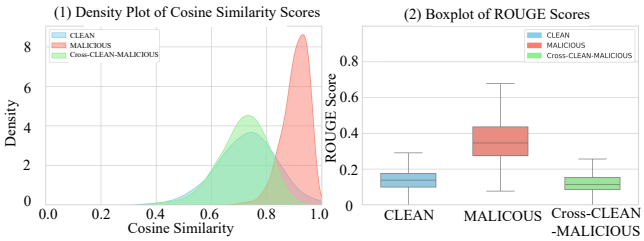

Figure 2: (1) The density plot of cosine similarity between three different groups. (2) The box plot of ROUGE Score between three different groups.

$$\min_{\mathcal{F}} \quad \text{ASR}(\text{LLM}(q_i; \tilde{\mathcal{E}}(q_i)))$$
$$\text{s.t.} \quad \text{ACC}(\text{LLM}(q_i; \tilde{\mathcal{E}}(q_i))) \geq \theta, \quad (3)$$

where θ is a minimum accuracy threshold. Maintaining ACC above θ is vital for the LLM's reliability with the filtered retrieval set $\tilde{\mathcal{E}}(q_i)$. While an LLM could persistently reject queries to evade attacks from the poisoned corpus $\mathcal{D} \cup \Gamma$, this over-rejection leads to excessive refusals, undermining its utility. It is essential to minimize the ASR while ensuring high response accuracy, balancing robust defense with effective query handling.

Methodology

TrustRAG is a defense framework against malicious attacks in RAG systems. It combines K-means clustering with both the LLM's internal knowledge and retrieved external documents to produce trustworthy responses. As shown in Figure 1, after attackers inject malicious documents targeting a specific question and answer, the retriever collects relevant sources. K-means filters out suspicious documents, and any remaining noise is further mitigated by comparing internal and external knowledge to resolve conflicts. The final answer can therefore be generated from the most reliable information.

Clean Retrieval – Stage 1

For each query, the attacker may generate either a single malicious document or multiple malicious documents to compromise the entire RAG pipeline. Under the heavy attack setting where attackers can inject multiple or even overwhelming number of malicious documents, this approach can effectively filter out malicious documents to ensure the integrity of the retrieved content. Conversely, in single-injection scenarios, most clean documents will be preserved and the only malicious document will be handled in the second stage of TrustRAG where knowledge conflicts will be resolved. For example, methods such as PoisonedRAG (Zou et al. 2024) utilize LLMs to generate the initial text I, employing varied temperature settings to produce multiple malicious documents for a query to perform multiple injection attacks. However, these documents inherently possess high similarity and form even tighter clusters in the embedding space compared to clean documents. Leveraging this

property, TrustRAG employs K-means clustering ($k = 2$) to distinguish between clean and potentially malicious documents. Unlike prior work, which primarily addresses the light single injection attack scenario, TrustRAG is designed to mitigate both single and multiple injection attacks.

K-means Clustering. In the first step, we apply the K-means clustering algorithm to analyze the distribution of text embeddings [1] generated by f_t and identify suspicious high-density clusters that may indicate the presence of malicious documents. We set k = 2 in our paper, because k > 2 would only split normal documents into artificial sub-clusters and require extra rules to merge them back, re-introducing hyperparameters the framework is designed to avoid. Moreover, malicious documents form a tight, high-density cluster while genuine passages are more dispersed, naturally supporting a binary partition.

N-gram Preservation. However, the drawback with vanilla K-means filtering is that it may also filter out clean documents especially under single attack scenario, thus losing useful information. Therefore, we propose using the ROUGE-L score (Lin 2004) to compare intra-cluster similarity, aiming to preserve the majority of clean documents. From Figure 2, we observed significant differences in ROUGE-L scores across three comparison types: clean-to-clean, malicious-to-malicious, and clean-to-malicious document pairs. By leveraging this property, we can avoid filtering groups that contain just one malicious document alongside clean documents. For example, when a malicious document is paired with a clean one, we refrain from filtering it. ROUGE-L can assist in this process, as pairs of clean and malicious documents will yield lower scores compared to groups of entirely malicious documents, thus minimizing information loss.

Conflict Resolution – Stage 2

In the Conflict Resolution 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.

Internal Knowledge Extraction. 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 LLMs to generate internal knowledge, following the work of Bai et al. (2022), which emphasizes the importance of reliability and trustworthiness in generated documents. In previous works (Sun et al. 2022; Yu et al. 2022; Wang et al. 2024), multiple diverse documents are generated using different temperature settings. However, in our preliminary experiments, we find that it may lead to hallucination or incorrectness, thus we only perform a single LLM inference with temperature of 0.

[1] We use the simcse-roberta-base as the text embedding model

Knowledge Consolidation. We employ the LLMs to explicitly consolidate information from both documents generated from its internal knowledge and documents retrieved from external sources. Initially, we combine documents 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 to guide the LLM in identifying consistent information across various documents while detecting and filtering out malicious content. This process helps reorganize the input documents by refining and consolidating unreliable knowledge into more coherent documents.

Self-Assessment of Retrieval Correctness. Finally, TrustRAG instructs the LLMs for self-assessment by comparing its internal knowledge with retrieved external documents. As shown in Figure 1, in this stage, both internal and external knowledge are provided to the LLM. We prompt the LLM to generate the best possible answer based on one of the two sources. This procedure pinpoints the most reliable sources, guaranteeing an accurate and reliable final answer, resulting in a sustained high accuracy.

Experiment Setup

Datasets. We use three question-answering datasets in this paper: Natural Questions (NQ) (Kwiatkowski et al. 2019), HotpotQA (Yang et al. 2018), and MS-MARCO (Bajaj et al. 2018) and two real-world datasets: RedditQA (Huang et al. 2024) and RAMDoc (Wang et al. 2025).

Attackers. We use four kinds of RAG attacks in this paper. (1) Corpus Poisoning Attack: PoisonedRAG (Zou et al. 2024). (2) Prompt Injection Attack: PIA (Zhong et al. 2023; Greshake et al. 2023). (3) Adversarial Decoding: AD (Zhang, Zhang, and Shmatikov 2025). (4) Denial-Of-Service Attack: Jamming Attack (Shafran, Schuster, and Shmatikov 2024).

Defenders. Considering the various types of attacks on the RAG process, several defense frameworks have been proposed. Among these, we introduce three widely recognized frameworks: RobustRAG (Xiang et al. 2024), InstructRAG (Wei, Chen, and Meng 2024), and AstuteRAG (Wang et al. 2024) to compare with our proposed TrustRAG model.

Poison Rates. We define the poison rate as the fraction of adversarial texts injected into the corpus, relative to the number of clean documents the retriever is expected to return. For example, at 20%, a target set of 10 clean documents is augmented with 2 poisoned ones before retrieval. The retriever then returns the top-k items (which may include poisons) to the LLM. Unless stated otherwise, we set $k = 5$ in all experiments.

Evaluation Metrics. Following prior work, we adopt two key metrics to evaluate the performance of all defense methods for Retrieval-Augmented Generation (RAG) systems: (1) Accuracy ($\mathbf{ACC}$): This measures the proportion of correct responses generated by the RAG system under normal conditions, reflecting its retrieval and generation reliability.

Models	Defense	HotpotQA (Yang et al. 2018)				NQ (Kwiatkowski et al. 2019)				MS-MARCO (Bajaj et al. 2018)			
		PIA ACC ↑ / ASR ↓	PoisonedRAG ACC ↑ / ASR ↓	AD ACC ↑ / ASR ↓	Clean ACC ↑	PIA ACC ↑ / ASR ↓	PoisonedRAG ACC ↑ / ASR ↓	AD ACC ↑ / ASR ↓	Clean ACC ↑	PIA ACC ↑ / ASR ↓	PoisonedRAG ACC ↑ / ASR ↓	AD ACC ↑ / ASR ↓	Clean ACC ↑
$\text{Mistral}_{\text{Nemo-12B}}$	No RAG	-	-	-	57.0	-	-	-	70.0	-	-	-	70.0
	Vanilla RAG	43.0/49.0	28.0/68.0	24.0/72.0	**78.0**	45.0/50.0	42.0/51.0	41.0/49.0	69.0	47.0/49.0	52.0/43.0	53.0/39.0	82.0
	$\text{RobustRAG}_{\text{Keyword}}$	55.0/25.0	51.0/27.0	50.0/32.0	54.0	55.0/**4.0**	54.0/**9.0**	54.0/8.0	57.0	75.0/**6.0**	72.0/**9.0**	72.0/9.0	72.0
	$\text{InstructRAG}_{\text{ICL}}$	31.0/64.0	36.0/59.0	37.0/59.0	73.0	53.0/41.0	48.0/40.0	52.0/42.0	65.0	57.0/37.0	57.0/36.0	63.0/28.0	83.0
	ASTUTE RAG	59.0/28.0	60.0/24.0	62.0/24.0	76.0	62.0/19.0	**69.0**/10.0	66.0/**6.0**	**72.0**	72.0/24.0	77.0/16.0	84.0/**5.0**	**84.0**
	$\text{TrustRAG}_{\text{stage 1}}$	37.0/51.0	38.0/54.0	22.0/72.0	74.0	45.0/43.0	46.0/40.0	42.0/47.0	66.0	42.0/54.0	50.0/44.0	48.0/47.0	79.0
	$\text{TrustRAG}_{\text{stage 1\&2}}$	**77.0**/**9.0**	**74.0**/**13.0**	**77.0**/**12.0**	**78.0**	**66.0**/8.0	67.0/11.0	**67.0**/**6.0**	69.0	**81.0**/9.0	**84.0**/12.0	**85.0**/10.0	82.0
$\text{Llama}_{\text{3.1-8B}}$	No RAG	-	-	-	38.0	-	-	-	70.0	-	-	-	75.0
	Vanilla RAG	3.0/95.0	27.0/81.0	37.0/59.0	71.0	4.0/93.0	26.0/73.0	41.0/56.0	71.0	2.0/98.0	28.0/70.0	52.0/44.0	79.0
	$\text{RobustRAG}_{\text{Keyword}}$	55.0/4.0	40.0/50.0	49.0/49.0	54.0	44.0/11.0	51.0/27.0	61.0/24.0	61.0	69.0/15.0	67.0/19.0	71.0/15.0	72.0
	$\text{InstructRAG}_{\text{ICL}}$	64.0/27.0	54.0/41.0	46.0/51.0	**83.0**	55.0/19.0	58.0/37.0	61.0/34.0	68.0	57.0/19.0	63.0/33.0	70.0/26.0	**89.0**
	ASTUTE RAG	51.0/28.0	65.0/**16.0**	67.0/20.0	65.0	70.0/14.0	77.0/11.0	81.0/**4.0**	75.0	71.0/25.0	54.0/41.0	85.0/8.0	83.0
	$\text{TrustRAG}_{\text{stage 1}}$	28.0/61.0	43.0/47.0	36.0/59.0	70.0	40.0/52.0	43.0/50.0	45.0/52.0	65.0	31.0/67.0	45.0/47.0	43.0/52.0	81.0
	$\text{TrustRAG}_{\text{stage 1\&2}}$	**73.0**/**3.0**	**66.0**/18.0	**70.0**/**18.0**	74.0	**83.0**/**2.0**	**82.0**/**9.0**	**82.0**/**4.0**	**82.0**	**86.0**/**7.0**	**83.0**/**11.0**	**86.0**/**7.0**	85.0
GPT_{4o}	No RAG	-	-	-	64.0	-	-	-	76.0	-	-	-	80.0
	Vanilla RAG	60.0/37.0	52.0/48.0	56.0/40.0	82.0	52.0/41.0	56.0/39.0	66.0/23.0	76.0	67.0/28.0	67.0/24.0	72.0/14.0	81.0
	$\text{RobustRAG}_{\text{Keyword}}$	60.0/8.0	51.0/27.0	41.0/40.0	54.0	40.0/38.0	39.0/28.0	37.0/33.0	45.0	48.0/29.0	50.0/22.0	50.0/19.0	56.0
	$\text{InstructRAG}_{\text{ICL}}$	58.0/41.0	33.0/63.0	55.0/40.0	**86.0**	63.0/34.0	53.0/39.0	67.0/24.0	79.0	69.0/28.0	59.0/35.0	71.0/18.0	81.0
	ASTUTE RAG	74.0/16.0	78.0/22.0	80.0/**4.0**	80.0	81.0/4.0	82.0/6.0	77.0/**2.0**	81.0	86.0/11.0	77.0/13.0	**86.0**/**2.0**	85.0
	$\text{TrustRAG}_{\text{stage 1}}$	56.0/37.0	54.0/46.0	52.0/44.0	76.0	49.0/41.0	57.0/35.0	60.0/28.0	76.0	63.0/35.0	62.0/24.0	72.0/19.0	77.0
	$\text{TrustRAG}_{\text{stage 1\&2}}$	**83.0**/**3.0**	**84.0**/**6.0**	**85.0**/6.0	84.0	**83.0**/**1.0**	**83.0**/**4.0**	**85.0**/**2.0**	**84.0**	**91.0**/**1.0**	**86.0**/**8.0**	**86.0**/7.0	**89.0**

Table 1: The result table presents the performance of various defense frameworks applied to different LLMs integrated with RAG systems, primarily against single injection attacks. The best performance is in bold.

(2) Attack Success Rate (**ASR**): This quantifies the system's vulnerability, calculated as the fraction of incorrect answers produced when misled by adversarial inputs.

Results

We conduct comprehensive experiments on two scenarios: single injection attack and multiple attacks.

Dataset	Defense	Poison-(80%) ACC ↑ / ASR ↓	Poison-(60%) ACC ↑ / ASR ↓	Poison-(40%) ACC ↑ / ASR ↓	Poison-(20%) ACC ↑ / ASR ↓
NQ	Vanilla RAG	2.0/98.0	3.0/97.0	4.0/93.0	26.0/73.0
	$\text{RobustRAG}_{\text{Keyword}}$	15.0/75.0	23.0/63.0	37.0/46.0	51.0/27.0
	$\text{InstructRAG}_{\text{ICL}}$	38.0/56.0	40.0/56.0	51.0/45.0	58.0/37.0
	ASTUTE RAG	64.0/24.0	68.0/19.0	69.0/18.0	77.0/11.0
	$\text{TrustRAG}_{\text{stage 1}}$	51.0/19.0	56.0/3.0	62.0/2.0	43.0/50.0
	$\text{TrustRAG}_{\text{stage 1\&2}}$	**85.0**/**1.0**	**84.0**/**1.0**	**83.0**/**1.0**	**82.0**/**9.0**
MS-MARCO	Vanilla RAG	3.0/96.0	5.0/94.0	7.0/93.0	28.0/70.0
	$\text{RobustRAG}_{\text{Keyword}}$	28.0/66.0	37.0/54.0	57.0/34.0	67.0/19.0
	InstructRAG	47.0/51.0	49.0/45.0	60.0/36.0	63.0/33.0
	ASTUTE RAG	40.0/57.0	50.0/47.0	52.0/44.0	54.0/41.0
	$\text{TrustRAG}_{\text{stage 1}}$	64.0/18.0	72.0/**7.0**	78.0/**6.0**	45.0/47.0
	$\text{TrustRAG}_{\text{stage 1\&2}}$	**84.0**/**8.0**	**85.0**/**7.0**	**85.0**/7.0	**83.0**/**11.0**
HotpotQA	Vanilla RAG	2.0/97.0	6.0/94.0	5.0/94.0	27.0/81.0
	$\text{RobustRAG}_{\text{Keyword}}$	10.0/87.0	19.0/76.0	33.0/57.0	40.0/50.0
	$\text{InstructRAG}_{\text{ICL}}$	40.0/57.0	50.0/47.0	52.0/44.0	54.0/41.0
	ASTUTE RAG	53.0/38.0	59.0/30.0	59.0/31.0	65.0/**16.0**
	$\text{TrustRAG}_{\text{stage 1}}$	61.0/12.0	72.0/**3.0**	66.0/**2.0**	43.0/47.0
	$\text{TrustRAG}_{\text{stage 1\&2}}$	**71.0**/**4.0**	**70.0**/7.0	**69.0**/5.0	**66.0**/18.0

Table 2: The result table presents the performance of various defense frameworks applied to $\text{Llama}_{\text{3.1-8B}}$ integrated with RAG systems, primarily against PoisonedRAG with different poison rates. The best performance is in bold.

Single Injection Attack. In this scenario, all the attack methods only can inject a malicious document into the retrieval database, manipulating the RAG system's response to a targeted query. As shown in the table 1, most of the previous methods fail to effectively handle the scenario of injecting a single malicious document into the knowledge database. For instance, under the PIA on the HotpotQA dataset, the ASR for other defensive frameworks exhibits considerable variation, ranging from 4.0% to 64.0%. In contrast, TrustRAG remains within a range of 3.0% to 9.0%, while achieving the highest ACC.

Notably, $\text{TrustRAG}_{\text{stage 1}}$ is designed to preserve the clean information. $\text{TrustRAG}_{\text{stage 2}}$ significantly enhances performance, as evidenced by consistent improvements in ACC and reductions in ASR. For instance, on the NQ under AD,

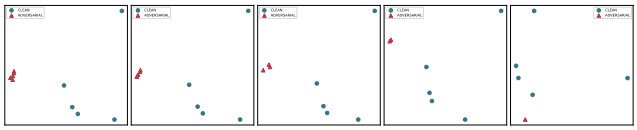

Figure 3: We analyze the embedding distribution of retrieved documents by plotting different numbers of poisoned data, ranging from 1 to 5, where red denotes adversarial and blue indicates clean. The results show that when the number of malicious documents exceeds 2, they tend to form distinct clusters.

$\text{TrustRAG}_{\text{stage 1}}$ with $\text{Llama}_{\text{3.1-8B}}$ yields an ACC of 45.0% and an ASR of 52.0%, whereas $\text{TrustRAG}_{\text{stage 1\&2}}$ boosts the ACC to 82.0% and reduces the ASR to 4.0%. This demonstrates that $\text{TrustRAG}_{\text{stage 2}}$ effectively refines the output by filtering residual malicious content that $\text{TrustRAG}_{\text{stage 1}}$ intentionally preserves to maintain clean data availability.

Additionally, under Jamming Attack (Shafran, Schuster, and Shmatikov 2024) on the NQ benchmark, the vanilla Llama 3.1-8B model attains 70–76% ACC, yet remains vulnerable, with an ASR of 20–23% across the three target refusals. In contrast, TrustRAG not only raises accuracy to 80–85%, but also suppresses the ASR to a negligible 0–2%.

Multiple Injection Attack. As illustrated in Table 2, we evaluate the performance on the open-source $\text{Llama}_{\text{3.1-8B}}$ model across varying poison rates, ranging from 20% to 80%. Our experiments demonstrate that TrustRAG achieves consistent performance across both ASR and ACC metrics on evaluated datasets, maintaining an average accuracy above 80%. It is worth noticing that RobustRAG, which is a defense framework using aggregating and voting strategies, fails when the number of malicious documents exceeds the number of benign ones. However, benefiting from the K-means filtering strategy, TrustRAG significantly reduces malicious documents during retrieval, and only a small portion of malicious documents is used in the Conflict Resolution stage.

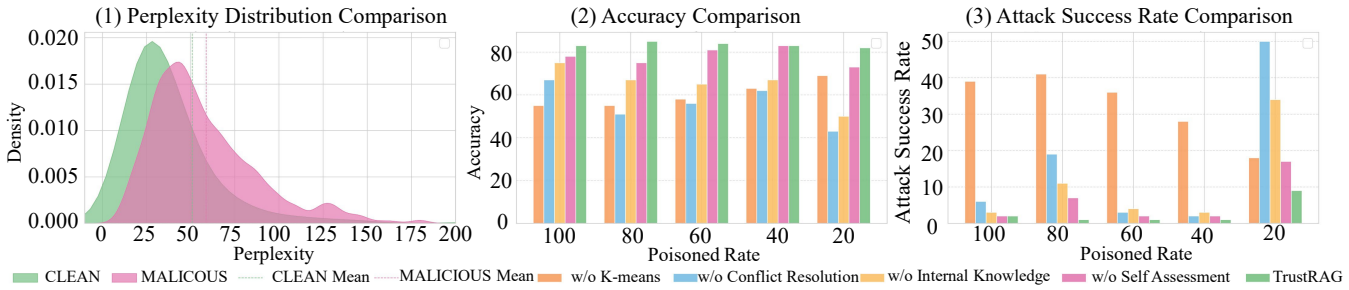

Figure 4: (1) The PPL distribution density plot between clean and malicious documents. And the lines of dashes represent the average PPL values. (2) The bar plot of ablation study on ACC in NQ based on the $\text{Llama}_{\text{3.1-8B}}$. (3) The bar plot of ablation study on ASR in NQ based on the $\text{Llama}_{\text{3.1-8B}}$.

	# API Call	MS-MARCO	NQ	HotpotQA
Vanilla RAG	1	8.9/1×	9.2/1×	9.6/1×
$\text{InstructRAG}_{\text{ICL}}$	1	12.6/1.4×	13.1/1.4×	32.7/3.4×
$\text{RobustRAG}_{\text{Keyword}}$	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×
$\text{TrustRAG}_{\text{K-means}}$	1	12.3/1.4×	12.6/1.4×	12.5/1.3×
$\text{TrustRAG}_{\text{Conflict}}$	3	18.4/2.1×	19.9/2.2×	21.7/2.3×

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

Embedding Model	Poison-(80%) F1↑ / CRR↑	Poison-(60%) F1↑ / CRR↑	Poison-(40%) F1↑ / CRR↑	Poison-(20%) F1↑ / CRR↑	Clean CRR↑
SimCSE	92.6/92.0	94.3/91.5	84.9/89.0	3.1/86.2	85.0
$\text{SimCSE}_{\text{w/o ROUGE}}$	83.9/93.0	72.0/69.0	64.4/68.3	35.8/54.8	52.6
Bert	84.7/84.0	87.4/89.0	77.8/82.0	5.6/78.5	74.2
$\text{Bert}_{\text{w/o ROUGE}}$	73.2/80.0	63.4/61.0	51.7/58.0	35.5/55.8	52.0
BGE	90.8/92.0	96.9/93.0	89.5/91.0	3.0/86.3	87.6
$\text{BGE}_{\text{w/o ROUGE}}$	85.0/93.0	86.7/83.0	79.9/80.7	27.5/51.5	51.4

Table 4: Results on NQ dataset 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.

Ablation Study. We performed an ablation study on $\text{Llama}_{\text{3.1-8B}}$ to evaluate the impact of four key components: (1) Stage 1 - Clean Retrieval using a K-means filtering strategy (denoted as "w/o K-means"), (2) the use of conflict resolution in Stage 2 (denoted as "w/o Conflict Resolution"), (3) the integration of internal knowledge into the conflict resolution process (denoted as "w/o Internal Knowledge", and (4) the inclusion of self-assessment to evaluate the model's confidence in external information (denoted as "w/o Self Assessment"). As shown in Figures 4 (2) and (3), which demonstrate that each component of the TrustRAG framework contributes significantly to its robustness against poisoning attacks. K-means clustering effectively filters out malicious content without sacrificing the quality of clean information, especially when the poisoning rate exceeds 20%. Integrating internal knowledge inferred from LLMs further improves accuracy and reduces attack success, particularly in mixed-clean and malicious scenarios. The Conflict Resolution stage emerges as the most vital, as its removal sharply increases vulnerability to attacks even when other defenses are present. Lastly, the self-assessment mechanism consistently boosts performance by enabling the LLM to distinguish between reliable and malicious or external information.

Analysis

Distribution of Malicious Documents. We analyze the embedding space of NQ dataset samples under varying poisoning intensities. Figure 3 presents a representative sample from the dataset, illustrating the characteristic distribution of documents in the embedding space. Our analysis reveals two distinct patterns: (1) in multi-document poisoning scenarios, malicious documents form tight clusters in the embedding space, while (2) single malicious documents exhibit dispersion among clean samples. This observation underscores the critical role of clustering strategy in preserving clean documents while filtering malicious documents during retrieval.

N-gram Preservation. In single injection attack scenarios since K-means may discard clean documents if they are clustered with the malicious one. Under such situations, our N-gram preservation mechanism provides complementary protection, successfully retaining documents for the knowledge consolidation stage. As shown in Table 4, we conduct an ablation study on N-gram preservation. We find that when the poisoning rate exceeds 20%, the F1 score is higher after applying N-gram preservation in the clean retrieval stage. However, when the poisoning rate is 20%, without N-gram preservation, the K-means filtering strategy will randomly remove the group with higher similarity. The clean documents can thus be filtered by mistake. Therefore, using N-gram preservation preserves the clean documents.

Runtime Analysis. In Table 3, we present a detailed runtime analysis for various methods across three datasets on $\text{Llama}_{\text{3.1-8B}}$. The analysis reveals that TrustRAG spends approximately twice the inference time as compared to Vanilla RAG, which is a reasonable trade-off considering the significant improvements in robustness offered by TrustRAG.

Hyper-parameter Setting. To ensure the reproducibility of our experiments, we set the sampling temperature to

Method	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)
Diverse	64.0 / 9.0	65.0 / 5.0	67.0 / 5.0	68.0 / 1.0
w/ question	65.0 / 1.0	66.0 / 3.0	65.0 / 4.0	69.0 / 14.0
w/o question	64.0 / 2.0	63.0 / 2.0	65.0 / 1.0	67.0 / 11.0

Table 5: Performance of $\text{TrustRAG}_{\text{stage1 \& 2}}$ on the NQ dataset with $\text{Mistral}_{\text{Nemo-12B}}$ under varying poison rates and document types. Diverse Malicious Document uses varied prompts, Malicious Document w/ question prepends the user question, and w/o question excludes it.

0.01 for all LLMs. In the K-means filtering stage, we use a ROUGE-L score threshold of 0.25 and a cosine similarity threshold of 0.85. A grid search over ROUGE-L $\in 0.20, 0.25, 0.30$ and cosine $\in 0.80, 0.85, 0.90$ showed that detection F1 and CRR fluctuated by $\leq 1.3\%$ and $\leq 1.8\%$, respectively. These results demonstrate the robustness of our chosen settings.

Effectiveness of PPL Detection. Malicious documents crafted by attackers may exhibit unnatural patterns, prompting the proposal of perplexity (PPL) detection as a defense mechanism (Alon and Kamfonas 2023; Jain et al. 2023). Notably, Shafran, Schuster, and Shmatikov (2024) asserts that the perplexity value distributions of clean and malicious documents differ markedly. To evaluate the efficacy of this PPL-based defense, we conducted an empirical analysis. As illustrated in Figure 4 (1), the PPL values of clean and adversarial texts exhibit considerable overlap. Contrary to the claims of Shafran, Schuster, and Shmatikov (2024) regarding substantial distributional differences, our results reveal a more nuanced reality. This overlap underscores the limitations of relying exclusively on PPL as a detection metric.

Robustness Without Query Clues. One of the assumption for K-means filtering is that the malicious documents are created in the same embedding space. In this section we examine the effectiveness of TrustRAG when malicious documents are created with diverse and unclustered embeddings. We consider the adaptive attack and more realistic world conditions. To rigorously assess TrustRAG's robustness under more challenging conditions, we evaluate its performance by removing the query questions from these documents, relying solely on the malicious content as external knowledge for model generation. Table 5 presents the results of two scenarios: (1) Malicious documents with questions, and (2) Malicious documents without questions. TrustRAG sustains high accuracy of 63.0% to 69.0% and low attack success rates of 1.0% to 14.0% in both settings, demonstrating that TrustRAG's defensive capabilities remain robust regardless of query inclusion.

Defense Against Diverse Attacks. From an alternative viewpoint, the malicious documents produced by PoisonedRAG exhibit a notable limitation in their lack of diversity. In this study, we seek to further examine the effects of diverse malicious documents on our TrustRAG framework and the efficacy of the K-means filtering strategy. As illustrated in Table 5, we present a scenario involving diverse malicious documents, which are constructed by varying semantics, logic, and style through tailored prompts from the PoisonedRAG. This analysis reaffirms the robustness of our approach. Specifically, TrustRAG proves to be highly effective even in multiple injection attacks in diverse contexts, as evidenced by the results in Table 5, where $\text{TrustRAG}_{\text{stage1 \& 2}}$ maintains strong accuracy and low attack success rates with varying poison rates and document types on the NQ data set with $\text{Mistral}_{\text{Nemo-12B}}$ compared to the performance of TrustRAG under the original PoisonRAG attack.

Dataset	Method	ACC↑ (%)	ASR↓ (%)
RedditQA	Vanilla RAG	27.3	43.8
	$\text{TrustRAG}_{\text{Llama-3.1-8B}}$	72.2	11.9
RAMDocs	No RAG	5.8	–
	Vanilla RAG	32.6	–
	$\text{AstuteRAG}_{\text{Llama-3.3-70B}}$	31.8	–
	$\text{MADAM-RAG}_{\text{Llama-3.3-70B}}$	34.4	–
	$\text{TrustRAG}_{\text{Llama-3.3-70B}}$	39.6	–
	$\text{AstuteRAG}_{\text{GPT-4o-mini}}$	13.8	–
	$\text{MADAM-RAG}_{\text{GPT-4o-mini}}$	28.0	–
	$\text{TrustRAG}_{\text{GPT-4o-mini}}$	37.6	–

Table 6: Accuracy (ACC↑) and attack-success rate (ASR↓) of the TrustRAG framework applied to different language models under real-world adversarial conditions. For RAMDocs, which provides different types of noise information for a query, there is no specific attack target where only ground truth is given, so there is no attack success rate.

Real-world RAG Attacks. To evaluate TrustRAG's performance under real-world adversarial conditions, we leverage the RedditQA dataset introduced by Huang et al. (2024) and RAMDocs (Wang et al. 2025). The first dataset includes Reddit posts with naturally occurring factual errors, leading to incorrect responses to related questions and simulating real-world noise and misinformation. The second dataset, built on AmbigDocs (Lee, Ye, and Choi 2024), extends it by introducing additional real-world retrieval complexities. As detailed in Table 6, for RedditQA, vanilla RAG, utilizing retrieved documents, achieves a response accuracy of 27.3% with an attack success rate of 43.8%, reflecting its vulnerability to real-world inaccuracies. In stark contrast, TrustRAG attains a response accuracy of 72.2% while reducing the ASR to 11.9%, underscoring its robustness in mitigating the impact of adversarial conditions encountered in practical settings. The same trend is observed in RAMDocs, where TrustRAG achieves significant improvements compared to the MADAM-RAG method, proposed by RAMDocs, under the same model architecture. This experiment demonstrates that the TrustRAG pipeline can be applied to real-world scenarios, enabling question-answering RAG systems to deliver trustworthy and valid responses.

Conclusion

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 knowl-

edge 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.

References

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

Alon, G.; and Kamfonas, M. 2023. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Bai, 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*.

Bajaj, P.; Campos, D.; Craswell, N.; Deng, L.; Gao, J.; Liu, X.; Majumder, R.; McNamara, A.; Mitra, B.; Nguyen, T.; et al. 2018. A human generated MAchine Reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*.

BBC. 2024. Glue pizza and eat rocks: Google AI search errors go viral. https://www.bbc.co.uk/news/articles/cd11gzejgz4o.

Chen, H.; Pasunuru, R.; Weston, J.; and Celikyilmaz, A. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*.

Chen, J.; Lin, H.; Han, X.; and Sun, L. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17754–17762.

Dai, Z.; Zhao, V. Y.; Ma, J.; Luan, Y.; Ni, J.; Lu, J.; Bakalov, A.; Guu, K.; Hall, K. B.; and Chang, M.-W. 2022. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755*.

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

Glass, M.; Rossiello, G.; Chowdhury, M. F. M.; Naik, A. R.; Cai, P.; and Gliozzo, A. 2022. Re2G: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*.

Google. 2024. Generative AI in Search: Let Google do the searching for you. https://blog.google/products/search/generative-ai-google-search-may-2024/.

Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; and Fritz, M. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 79–90.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, 3929–3938. PMLR.

Huang, Y.; Chen, S.; Cai, H.; and Dhingra, B. 2024. Enhancing Large Language Models' Situated Faithfulness to External Contexts. *arXiv preprint arXiv:2410.14675*.

Izacard, G.; Lewis, P.; Lomeli, M.; Hosseini, L.; Petroni, F.; Schick, T.; Dwivedi-Yu, J.; Joulin, A.; Riedel, S.; and Grave, E. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43.

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

Kim, J.; Nam, J.; Mo, S.; Park, J.; Lee, S.-W.; Seo, M.; Ha, J.-W.; and Shin, J. 2024. SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs. *arXiv preprint arXiv:2404.13081*.

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

Lee, Y.; Ye, X.; and Choi, E. 2024. AmbigDocs: Reasoning across Documents on Different Entities under the Same Name. arXiv:2404.12447.

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

Li, Z.; Zhou, H.; Rei, M.; and Specia, L. 2025. DiffuseDef: Improved Robustness to Adversarial Attacks via Iterative Denoising. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 9259–9274. Vienna, Austria: Association for Computational Linguistics. ISBN 979-8-89176-251-0.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Microsoft. 2024. Bing Chat. https://www.microsoft.com/en-us/edge/features/bing-chat.

Perplexity, A. 2024. Perplexity AI. https://www.perplexity.ai/.

rocky. 2024. A retrieval corruption attack. https://x.com/r_cky0/status/1859656430888026524?s=46&t=p9-0aPCrd_0h9-yuSXpN8g.

Shafran, A.; Schuster, R.; and Shmatikov, V. 2024. Machine Against the RAG: Jamming Retrieval-Augmented Generation with Blocker Documents. *arXiv preprint arXiv:2406.05870*.

Sun, Z.; Wang, X.; Tay, Y.; Yang, Y.; and Zhou, D. 2022. Recitation-augmented language models. *arXiv preprint arXiv:2210.01296*.

Tan, Z.; Zhao, C.; Moraffah, R.; Li, Y.; Wang, S.; Li, J.; Chen, T.; and Liu, H. 2024. " Glue pizza and eat rocks"– Exploiting Vulnerabilities in Retrieval-Augmented Generative Models. *arXiv preprint arXiv:2406.19417*.

Wang, F.; Wan, X.; Sun, R.; Chen, J.; and Arık, S. Ö. 2024. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*.

Wang, H.; Prasad, A.; Stengel-Eskin, E.; and Bansal, M. 2025. Retrieval-Augmented Generation with Conflicting Evidence. arXiv:2504.13079.

Wei, Z.; Chen, W.-L.; and Meng, Y. 2024. InstructRAG: Instructing Retrieval-Augmented Generation with Explicit Denoising. *arXiv preprint arXiv:2406.13629*.

Xiang, C.; Wu, T.; Zhong, Z.; Wagner, D.; Chen, D.; and Mittal, P. 2024. Certifiably Robust RAG against Retrieval Corruption. *arXiv preprint arXiv:2405.15556*.

Xue, J.; Zheng, M.; Hu, Y.; Liu, F.; Chen, X.; and Lou, Q. 2024. BadRAG: Identifying Vulnerabilities in Retrieval Augmented Generation of Large Language Models. *arXiv preprint arXiv:2406.00083*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Yu, W.; Iter, D.; Wang, S.; Xu, Y.; Ju, M.; Sanyal, S.; Zhu, C.; Zeng, M.; and Jiang, M. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

Zhang, B.; Chen, Y.; Fang, M.; Liu, Z.; Nie, L.; Li, T.; and Liu, Z. 2025. Practical poisoning attacks against retrieval-augmented generation. *arXiv preprint arXiv:2504.03957*.

Zhang, C.; Zhang, T.; and Shmatikov, V. 2025. Adversarial Decoding: Generating Readable Documents for Adversarial Objectives. arXiv:2410.02163.

Zheng, H. S.; Mishra, S.; Chen, X.; Cheng, H.-T.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

Zhong, Z.; Huang, Z.; Wettig, A.; and Chen, D. 2023. Poisoning retrieval corpora by injecting adversarial passages. *arXiv preprint arXiv:2310.19156*.

Zhou, Y.; Liu, Y.; Li, X.; Jin, J.; Qian, H.; Liu, Z.; Li, C.; Dou, Z.; Ho, T.-Y.; and Yu, P. S. 2024. Trustworthiness in retrieval-augmented generation systems: A survey. *arXiv preprint arXiv:2409.10102*.

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