PRA-RAG: Provably Robust Aggregation in Retrieval-Augmented Generation against Retrieval Corruption

Anonymous ARR submission

Abstract

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by incorporating external knowledge, effectively mitigating their inherent knowledge limitations. However, RAG remains vulnerable to poisoning attacks that manipulate retrieved texts to mislead model outputs. Existing defense mechanisms often lack theoretical robustness guarantees and perform unreliably when the LLM has limited knowledge of the retrieved content. In this work, we propose PRA-RAG, a provably robust retrieval aggregation algorithm designed to defend against poisoning attacks on retrieved texts. PRA-RAG samples multiple combinations of retrieved texts and utilizes geometric structures in the embedding space to identify a robust subset, from which a stable aggregated representation is derived. We provide theoretical bounds on the maximum impact of poisoned retrieved content and establish a quantitative measure of RAG's robustness. Experiments across multiple benchmarks and RAG architectures demonstrate that PRA-RAG reduces the attack success rate to as low as *1%* while maintaining an accuracy of *71%*, significantly outperforming representative state-of-the-art (SOTA) methods.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) is an advanced generative paradigm that integrates external knowledge databases to effectively address the limitations of LLMs in domain-specific knowledge coverage and access to up-to-date information. In a typical RAG pipeline, when a user submits a query (e.g., "*What is the name of the highest mountain*?"), the retriever first encodes it into an embedding vector using a text encoder (e.g., BERT (Kenton and Toutanova, 2019)) and then retrieves the most similar texts from an external knowledge database. These retrieved texts are subsequently provided as context to the LLM to guide and enhance the response generation process. RAG has been widely adopted in various real-world applications due to its strengths in knowledge augmentation and high-quality generation, with prominent examples including ChatGPT (Achiam et al., 2023), Microsoft Bing Chat (Microsoft, 2024), and Google Search AI (Google, 2024). However, the integration of external knowledge databases further exacerbates concerns regarding the security of LLMs (rocky, 2024; BBC, 2024).

Recent studies have shown that injecting malicious content into retrieved texts can steer the LLM to generate responses aligned with the attacker's intent (e.g., the target answer could be "*Fuji*" when the target question is "*What is the name of the highest mountain*?"), thereby posing a serious threat to the reliability and security of RAG systems (Greshake et al., 2023; Tan et al., 2024). For example, PoisonedRAG (Zou et al., 2024) constructs poisoned texts through optimization, ensuring they are retrieved and induce the LLM to generate the attacker's target response. To counter attacks induced by poisoned texts, AstuteRAG (Wang et al., 2025) and TrustRAG (Zhou et al., 2025) introduce detection-based defenses that leverage the internal knowledge of LLMs to identify and filter maliciously retrieved content. However, their effectiveness is limited in scenarios where the LLM lacks sufficient knowledge to recognize the poisoned inputs. Moreover, existing methods (Wang et al., 2025; Zhou et al., 2025; Wei et al., 2024; Asai et al., 2024) lack a theoretical framework for certifying or quantifying the robustness of RAG systems, leaving their reliability under adversarial conditions largely unverified. RobustRAG (Xiang et al., 2024) offers provable robustness guarantees for the final LLM outputs. However, it requires generating responses from multiple LLMs, resulting in significant computational overhead.

We propose PRA-RAG, a provably robust retrieval aggregation algorithm (see Fig. 1), designed

to mitigate the risk of poisoning attacks at the retrieval stage. During retrieval, we intentionally expand the candidate set to increase informational diversity, which enhances the system's resilience to poisoned content by reducing the influence of any single malicious text. The aggregation process consists of three key steps. **First**, multiple subsets of the retrieved texts are sampled to form diverse candidate combinations, thereby constraining the influence of poisoned content within a controllable range. **Second**, Each combination is encoded into an embedding vector. In this geometric space, we identify a *minimum-radius ball* that encloses more than half of the combinations and take its center as the selected robust subset. **Finally**, a weighted average of the embeddings from the selected subset is computed to derive the final robust representation, effectively capturing the consensus while mitigating the influence of outliers.

Benefiting from the proposed concept of the minimum radius ball, we model the semantic shift in retrieved texts caused by poisoning attacks. We show that this shift is bounded by a theoretical upper limit, as mathematically proved in Section 5. The semantic shift quantifies how poisoning influences retrieval, while the upper bound offers a theoretical basis for constraining this effect. A lower poisoning influence results in safer downstream RAG generation.

In summary, our key contributions are as follows:

- **Theory:** We provide a theoretical characterization and certification of the maximum semantic deviation introduced by poisoned retrievals. This bound serves as a principled metric for assessing both system robustness and attack strength.
- **Algorithm:** We propose PRA-RAG, a provably robust aggregation algorithm for RAG systems that effectively mitigates the influence of poisoned retrievals and preserves the accuracy of generated outputs.
- **Evaluation:** We conduct extensive experiments to validate the effectiveness of PRA-RAG. For example, on the MSMARCO dataset with 20% of the retrievals poisoned, our method achieves an accuracy of 71% while reducing the attack success rate to just 1%. Our approach also surpasses state-of-the-art methods in efficiency.

2 Background and Related Work

Retrieval-Augmented Generation. RAG comprises three components: *knowledge database*, *retriever*, and *LLM*. The knowledge database contains newly added or updated information beyond the LLM's training data, often sourced from Wikipedia (Thakur et al., 2021) and similar platforms. The retriever selects the Top-K texts most similar to the query q as external context, which the LLM uses alongside q to generate an answer. RAG involves two key stages: retrieval and generation. In the retrieval step, a retriever selects the Top-K relevant knowledge pieces for a query q. This is done using two encoders: E_q for the query and E_p for knowledge passages. Each passage embedding $E_p(p_i)$ is compared with $E_q(q)$ using a similarity metric (e.g., cosine or dot product), and the Top-K results form the context $\mathcal{X}_q$. In the generation step, the query q and context $\mathcal{X}_q$ are combined into a prompt, and the LLM generates a response $\text{LLM}(q, \mathcal{X}_q)$.

Retrieval Corruption Attack. The use of external knowledge databases in RAG introduces security vulnerabilities, as adversaries can exploit them to inject misinformation and indirectly influence the model's outputs (Zou et al., 2024; Xue et al., 2024; Jiao et al., 2024). PoisonedRAG (Zou et al., 2024) generates adversarially crafted texts through an optimization-based approach and injects them into the knowledge database, thereby inducing the LLM to produce attacker-specified target responses. The Denial-of-Service Attack (Shafran et al., 2024) involves inserting a single "blocker" document into the knowledge database, which is retrieved in response to a specific query and causes the RAG system to refuse to answer that query. Adversarial Decoding (Zhang et al., 2025) performs RAG poisoning and LLM guard evasion by generating readable adversarial texts through adversarial decoding. PRCAP (Zhong et al., 2023) perturbs discrete tokens to craft adversarial samples similar to training queries and injects them into the knowledge database. BadRAG (Xue et al., 2024) employs white-box optimization to attack the retriever and uses handcrafted documents to target the generator. The aforementioned attack methods have demonstrated significant effectiveness. To evaluate the proposed defense strategy, we select representative attacks to simulate real-world RAG poisoning scenarios.

The Robustness of RAG. In order to defend

against the aforementioned retrieval-targeted poisoning attacks, TrustRAG (Zhou et al., 2025) introduces a two-stage defense mechanism that leverages the semantic characteristics of poisoned texts and the inherent knowledge of the LLM, effectively mitigating both single-point and multi-corpus injection attacks. INSTRUCTRAG (Wei et al., 2024) is designed to explicitly learn how to denoise retrieved content, thereby addressing the presence of poisoned or irrelevant information. SELF-RAG (Asai et al., 2024) proposes a framework that enhances generation quality and factual accuracy through self-reflection mechanisms within the LLM. Other robustness enhancement strategies include carefully crafted prompts (Cho et al., 2023; Press et al., 2023), plug-in model architectures (Baek et al., 2023), and purpose-built specialized models (Yoran et al., 2023). However, the aforementioned defense strategies have neither provided quantitative evaluations of RAG's robustness nor established theoretical guarantees. RobustRAG (Xiang et al., 2024) improves RAG robustness by using multiple LLMs to generate diverse responses and applying a voting mechanism to mitigate poisoned content. While it offers theoretical guarantees, the approach incurs high computational cost and poses challenges in aggregating outputs. To address these issues, we propose a provably robust RAG algorithm that enables quantitative robustness evaluation and demonstrates strong empirical defense against poisoning attacks.

3 Preliminary

3.1 Threat Model

We study *retrieval-corruption* attacks on RAG systems, where an adversary poisons the external corpus so that the retrieved context steers the generator to produce attacker-chosen outputs. We first specify the system and notation, then state the attacker's objectives and capabilities.

System and notation. Let $\mathcal{D}$ denote the external corpus and let $\mathcal{R}$ be the retriever that returns the Top-K passages for a query q, written as $\mathcal{X}_K(q;\mathcal{D}) = \{p_1, \ldots, p_K\}$. Under poisoning, the adversary injects a set of passages Γ into the corpus, yielding $\mathcal{D}' = \mathcal{D} \cup \Gamma$. The corresponding retrieved set is $\mathcal{X}'_K(q) = \mathcal{X}_K(q;\mathcal{D}')$ with $\varepsilon = \mathcal{X}'_K(q) \cap \Gamma$ and $\mathcal{X}'_K(q) = \{\tilde{p}_1, \ldots, \tilde{p}_\varepsilon\} \cup \{p_1, \ldots, p_{K-\varepsilon}\}$. We denote by n the subset size used later by our aggregation mechanism.

Attacker's objectives. The attacker selects a set of target queries $Q = \{q_1, \ldots, q_M\}$ along with corresponding target answers $A = \{a_1, \ldots, a_M\}$. The objective is to poison the corpus such that for each $q_i \in Q$, the RAG system, when conditioned on the retrieved context from the contaminated corpus $\mathcal{D}'$, generates the attacker-specified answer a_i. For instance, given the query q_i: *"What is the name of the highest mountain?"*, the attacker attempts to mislead the system into generating the incorrect response *"Mount Fuji"*.

Attacker's capabilities. We consider an adversary who can inject a set of malicious passages $\Gamma = \{\tilde{p}^i_j \mid i = 1, \ldots, M;\ j = 1, \ldots, N\}$ into the external corpus (e.g., via content published on open platforms such as Wikipedia or via data vendors that aggregate third-party content). The attacker has no knowledge of the LLM parameters or decoding, but is assumed to have white-box knowledge of the retriever because many retrievers (e.g., Contriever (Izacard et al., 2021), ANCE (Xiong et al., 2020)) are publicly available. We do not assume strict geometric separability between clean and poisoned passages. Our method's robustness is distribution-free and relies solely on a *combinatorial majority*: among the Top-K items with at most ε poisons, the $\binom{K-\varepsilon}{n}$ clean size-n subsets exceed half of all $\binom{K}{n}$ subsets whenever $\binom{K-\varepsilon}{n} > \frac{1}{2}\binom{K}{n}$. Our aggregation targets this majority, so certification holds even when embeddings are very close. This assumption is valid in real-world scenarios, as achieving an excessively large ε to heavily poison a RAG database incurs prohibitive costs and is inherently challenging for attackers.

3.2 Defense Objective

In this work, our defense primarily focuses on mitigating attacks launched through the injection of poisoned texts (Zou et al., 2024; Xue et al., 2024; Jiao et al., 2024) rather than prompt injection (Greshake et al., 2023). Our approach is designed to bound and reduce the impact of poisoned passages while preserving both retrieval and generation quality in RAG.

Quantifying the effect of poisoning. Heuristic defenses often lack a precise characterization of how poisoned content perturbs the system. Our robustness operator $\mathcal{F}$ aggregates the retrieved set $\mathcal{X}'$ and provides a certified upper bound on the induced semantic deviation (PAD), enabling quantitative evaluation of RAG robustness.

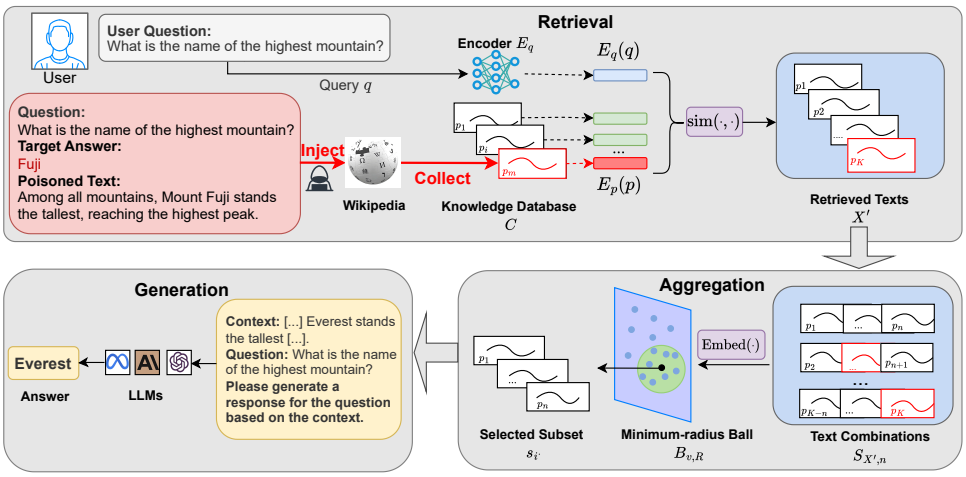


Figure 1: The workflow of our method.

Low ASR with high ACC. Mitigating poisoning while maintaining usability is essential. A low Attack Success Rate (ASR) indicates effective defense, whereas a high Accuracy (ACC) reflects preserved generation quality. We formalize the robustness–utility trade-off as

$$\begin{aligned} \min_{\mathcal{F}} \quad & \mathrm{ASR}(\mathrm{LLM}(q_i; \mathcal{F}(\mathcal{X}'_K(q_i))) \\ \text{s.t.} \quad & \mathrm{ACC}(\mathrm{LLM}(q_i; \mathcal{F}(\mathcal{X}'_K(q_i))) \geq \theta. \end{aligned} \tag{1}$$

where θ is a user-specified minimum acceptable accuracy. Overly conservative behavior (e.g., blanket refusals) may trivially reduce ASR but harms usability; our objective explicitly discourages such solutions by enforcing the ACC constraint.

4 Our Provable PRA-RAG

In RAG, external corpora may contain unreliable or unverified information; naively conditioning on retrieved passages can lead to inaccurate or harmful outputs (Zou et al., 2024; Xue et al., 2024; Jiao et al., 2024). We cast retrieval corruption as a *set perturbation* problem, where an adversary alters at most ε items in the Top-K set, and propose a geometric, model-agnostic retrieval aggregation that operates on *combination embeddings*. By selecting a majority-covering center, the method yields *certifiable* bounds on semantic shift and relies on a *combinatorial majority* of clean subsets rather than margin-based separability. We formalize this view and demonstrate both theoretical guarantees and practical benefits.

As illustrated in Fig. 1, given a query q, we retrieve the Top-K most relevant documents from a knowledge database, denoted as the set $\mathcal{X} = \{p_1, \ldots, p_K\}$. In our approach, we set the number of retrieved texts K slightly higher than the actual requirement to introduce diversity, thereby mitigating the impact of poisoned content. Next, we sample multiple subsets from the retrieved texts. Let $\mathcal{S}_{\mathcal{X},n} = \{s_1, s_2, \ldots, s_L\}$ denote the set of all possible subsets of size n drawn from the K retrieved documents, where $L = \binom{K}{n}$. Each document p_i in the subset is encoded as an embedding vector $\mathbf{e}_i \leftarrow Embed(p_i), \mathbf{e}_i \in \mathbb{R}^d$, yielding a set of embeddings in the d-dimensional vector space. We construct the corresponding vector set $\mathcal{V}_{\mathcal{X},n} = \{v_1, v_2, \ldots, v_L\}$ from the set $\mathcal{S}$, where each vector v_i is formed by concatenating the embedding vectors $\mathbf{e}_i$ of all texts in the corresponding subset s_i.

We define a distance between two vectors $u, v \in \mathcal{V}_{\mathcal{X},n}$ as $d(u, v) = 1 - cosine(Distance(u, v))$, where *Distance* is typically measured using cosine similarity. For a point $v \in \mathcal{V}_{\mathcal{X},n}$ and radius $R > 0$, define the ball:

$$B(v, R) = \{u \in \mathcal{V}_{\mathcal{X},n} : d(u, v) \leq R\}. \tag{2}$$

Our method $\mathcal{F}$ aims to identify the center of a minimum-radius ball that contains more than half of the points, representing the desired output of the robust aggregation process. Formally, the objective is defined as:

$$\mathcal{F}(\mathcal{X}) = \arg\min_{z} \; R \quad \text{s.t.} \quad \sum_{w \in V_{\mathcal{X},n}} \mathbf{1}_{\mathcal{B}(z,R)}(w) \geq \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor + 1, \tag{3}$$

where $\mathcal{X}$ denotes the input set, $V_{\mathcal{X},n}$ is the vector collection of all n-element subsets of $\mathcal{X}$, $\mathcal{B}(z, R)$

Algorithm 1 Inference Procedure for Selecting Robust Retrieval Text

Input: A set of Top-K retrieved texts $\mathcal{X} = \{p_1, p_2, \ldots, p_K\}$; subset size n (with constraint $2n < K$).
Output: A robustly aggregated retrieval text x^* for LLMs.
$\mathcal{S}_{\mathcal{X},n} = \{s_1, s_2, \ldots, s_L\} \leftarrow \mathrm{Comb}(\mathcal{X}, n)$, $L = \binom{K}{n}$
for $i = 1$ to L **do**
  embeddings $\leftarrow [\, Embed(p) \mid p \in s_i \,]$
  $v_i \leftarrow \mathrm{Concat}(\text{embeddings})$
  $\mathcal{V}_{\mathcal{X},n} \leftarrow \mathcal{V}_{\mathcal{X},n} \cup \{v_i\}$
end for
for $i = 1$ to L **do**
  Initialize similarity vector $\mathbf{sim}_i \in R^{s-1}$
  for $j = 1$ to L and $j \neq i$ **do**
    $d_{ij} \leftarrow 1 - cosine(Distance(v_i, v_j))$
    Store d_{ij} in $\mathbf{sim}_i$
  end for
  Sort $\mathbf{sim}_i$ in ascending order
  Let $m_i \leftarrow$ the k-th largest value in $\mathbf{sim}_i$, $k = \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor$
end for
Identify $i^* = \arg\min_i m_i$, $R = m_{i^*}$
Compute $x_{i^*} = \mathrm{Aggregate}([\, Embed(p) \mid p \in s_{i^*} \,])$
return x_{i^*} as input to the LLMs

denotes a ball of radius R centered at z. The indicator function $\mathbf{1}_{\mathcal{B}(z,R)}(w)$ is defined to be 1 if $w \in \mathcal{B}(z, R)$, and 0 otherwise. This formulation ensures that the selected center z lies within a ball that encompasses the majority of the subsets. Since $S_{\mathcal{X},n}$ includes all possible combinations of the retrieved texts, the centroid z corresponding to the selected subset s effectively captures the meaning of the majority of texts in $\mathcal{X}$, thereby filtering out a small number of poisoned texts.

Finally, we compute a weighted average of the embedding vectors of the texts in the selected subset s_{i^*}, where the weights are determined by the similarity (e.g., cosine similarity) between each text $p \in s_{i^*}$ and the query q:

$$x_{i^*} = \frac{\sum_{p \in s_{i^*}} \mathrm{sim}(p, q) \cdot \mathrm{Embed}(p)}{\sum_{p \in s_{i^*}} \mathrm{sim}(p, q)}, \tag{4}$$

which helps prevent poisoned texts from evading filtration due to semantic similarity with clean texts, yielding the final robust aggregation result. More details are provided in Algorithm 1.

5 Computing the Certified Maximum Deviation

In the context of retrieval-based text poisoning attacks on RAG systems, we assume that the attacker successfully injects ε malicious documents into the final retrieved set $\mathcal{X}$, resulting in a perturbed set $\mathcal{X}' = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_\varepsilon] \cup [p_1, p_2, \ldots, p_{K-\varepsilon}]$, where each $\tilde{p}_i$ represents a poisoned document inserted by the attacker, and each p_i denotes a clean document. In the absence of poisoning attacks, the center of the minimum-radius ball computed by Algorithm 1 is used as the robust output. Under poisoning attacks, the radius $\hat{R}$ is enlarged to counter the shift caused by the ε poisoned texts, ensuring that over half of the embeddings remain within the ball. Accordingly, the radius is approximately set to $\hat{R} \leftarrow \mathcal{D}[k]$, where $k = \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor + \left(\binom{K}{n} - \binom{K-\varepsilon}{n} \right)$, as shown in Algorithm 2. We denote the embedding shift of the final aggregated text caused by poisoned samples as d, where $d = (1 + \beta)\hat{R}$. The proposed metric d effectively quantifies the impact of poisoned texts on the retrieval results. A smaller value of d indicates lower sensitivity to perturbations, implying stronger robustness and greater resistance to successful attacks.

Certifying the Upper Bound of Deviation d. Let $\mathcal{X}'$ be any corrupted version of $\mathcal{X}$ obtained by changing at most ϵ passages, that is $\|\mathcal{X}' - \mathcal{X}\|_0 \leq \epsilon$. Define the robustness radius R as the smallest value satisfying

$$\forall\, \mathcal{X}' \text{ s.t. } \|\mathcal{X}' - \mathcal{X}\|_0 \leq \epsilon, \quad \sum_{w \in \mathcal{V}_{\mathcal{X}',n}} \mathbf{1}_{\mathcal{B}(\mathcal{F}(\mathcal{X}),R)}(w) \geq \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor + 1. \tag{5}$$

We present the following theorem:

Theorem 1. For every $\mathcal{X}'$ satisfying $\|\mathcal{X}' - \mathcal{X}\|_0 \leq \epsilon$,

$$d\big(\mathcal{F}(\mathcal{X}),\, \mathcal{F}(\mathcal{X}')\big) \;\leq\; 2R, \tag{6}$$

where the proof is provided in Section A.1. However, computing $\mathcal{F}(\mathcal{X})$ exactly is computationally expensive in practice. Therefore, we use an approximation method from Center Smoothing (Kumar and Goldstein, 2021) to compute a β-MEB (Minimum Enclosing Ball) that contains the majority of

elements in $\mathcal{V}_{\mathcal{X},n}$, denoted as:

$$\widehat{\mathcal{F}}(\mathcal{X}) = \arg\min_{z \in \mathcal{V}_{\mathcal{X},n}} R \quad \text{s.t.} \quad \sum_{w \in \mathcal{V}_{\mathcal{X},n}} \mathbf{1}_{\mathcal{B}(z,R)}(w) \geq \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor + 1. \tag{7}$$

The radius of this approximation differs from the optimal radius by a multiplicative factor β. Similarly, the approximation of R, denoted as $\widehat{R}$, is defined to satisfy:

$$\forall\, \mathcal{X}' \text{ s.t. } \|\mathcal{X}' - \mathcal{X}\|_0 \leq \epsilon, \quad \sum_{w \in \mathcal{V}_{\mathcal{X}',n}} \mathbf{1}_{\mathcal{B}(\mathcal{F}(\mathcal{X}),\widehat{R})}(w) \geq \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor + 1. \tag{8}$$

Then, we have the following theorem:

Theorem 2. For any $\mathcal{X}'$ satisfying $\|\mathcal{X}' - \mathcal{X}\|_0 \leq \epsilon$, we have:

$$d(\mathcal{F}(\mathcal{X}), \mathcal{F}(\mathcal{X}')) \leq (1+\beta)\widehat{R}, \tag{9}$$

where the proof is provided in Section A.2. We follow the practice in the work (Kumar and Goldstein, 2021) to use $\beta = 2$ as an approximation when computing the minimum enclosing ball:

$$d(\mathcal{F}(\mathcal{X}), \mathcal{F}(\mathcal{X}')) \leq 3\widehat{R}. \tag{10}$$

Algorithm 2 Certify Robustness under Poisoning

Input: A set of Top-K retrieved texts $\mathcal{X}$; subset size n (with $2n < K$); maximum number of poisoned texts ε (with $\binom{K}{n} < 2\binom{K-\varepsilon}{n}$).
Output: Certified maximum deviation d.
Let $\mathcal{S}_{\mathcal{X}',n} = \text{Comb}(\mathcal{X}', n)$
Initialize $\mathcal{V}_{\mathcal{X}',n} \leftarrow \emptyset$
for each subset $s_i \in \mathcal{S}_{\mathcal{X}',n}$ **do**
  $v_i \leftarrow \text{Concat}([\,Embed(p) \mid p \in s_i\,])$
  $\mathcal{V}_{\mathcal{X}',n} \leftarrow \mathcal{V}_{\mathcal{X}',n} \cup \{v_i\}$
end for
Let v_{orig} be the embedding vector of the subset selected by the Algorithm 1.
Initialize an empty list of distances $\mathcal{D} \leftarrow \emptyset$
for each embedding $v_i \in \mathcal{V}_{\mathcal{X}',n}$ **do**
  $d_i \leftarrow 1 - cosine(Distance(v_{\text{orig}}, v_i)))$
  Append d_i to $\mathcal{D}$
end for
Sort $\mathcal{D}$ in ascending order
Let $\hat{R} \leftarrow \mathcal{D}[k], k = \left\lfloor \frac{\binom{K}{n}}{2} \right\rfloor + \left(\binom{K}{n} - \binom{K-\varepsilon}{n}\right)$
Let $d = (1+\beta)\,\hat{R}$
return d as the certified maximum deviation

6 Evaluation

6.1 Experimental Setup

In this section, we provide a detailed description of the experimental setup, with additional information and default configurations presented in Section B.1.

Datasets. In order to comprehensively evaluate the effectiveness of the proposed method, we conducted experiments on three representative question answering datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), MS-MARCO (Bajaj et al., 2016).

LLM. We evaluate PRA-RAG with different LLMs, including Mistral-7B (Jiang et al., 2023), Llama3-8B, Vicuna-7B (Chiang et al., 2023), and GPT-3.5-Turbo (Brown et al., 2020).

Attackers. We conduct a systematic evaluation of PRA-RAG's defense effectiveness against three types of poisoning attacks targeting the retrieved texts, following prior works (Xiang et al., 2024; Zhou et al., 2025): (1) Corpus Poisoning Attack: PoisonedRAG (Zou et al., 2024), (2) Adversarial Decoding: AdvDec (Zhang et al., 2025), and (3) Denial-Of-Service Attack: DoS Attack (Shafran et al., 2024). Further details of the attack methods are provided in Section B.3.

Defenders. Although various defense methods have been proposed to counter attacks on RAG, we highlight the advantages of our approach by comparing it with several representative state-of-the-art baselines, including RobustRAG (Xiang et al., 2024), InstructRAG (Wei et al., 2024), and AstuteRAG (Wang et al., 2025), using their default configurations to ensure defense effectiveness.

Evaluation Metrics. We conduct a comprehensive evaluation of the proposed method using three metrics: (1) Adversarial Accuracy (**ACC**) assesses the system's ability to produce correct answers when subjected to poisoning attacks; (2) Attack Success Rate (**ASR**) represents the proportion of incorrect answers generated due to poisoning attacks. We use GPT-4o to determine whether the LLM's responses are correct or influenced by poisoning, using prompts are provided in the Section B.2; (3) Provable Average Deviation (**PAD**) provides a certified upper bound on semantic shifts in the ag-

Table 1: The table presents the performance of PRA-RAG across different attack strategies and datasets under various LLMs.

Models	Metric	NQ				MS-MARCO				HotpotQA			
		PoisonedRAG	DoS Attack	AdvDec	Clean	PoisonedRAG	DoS Attack	AdvDec	Clean	PoisonedRAG	DoS Attack	AdvDec	Clean
Mistral-7B	ACC	0.62	0.57	0.55	0.60	0.63	0.65	0.61	0.57	0.37	0.33	0.39	0.38
	ASR	0.01	0.02	0.02	-	0.01	0.03	0.02	-	0.03	0.01	0.02	-
	PAD	1.23	1.43	1.30	0.79	1.37	1.46	1.51	0.90	1.33	1.55	1.43	0.83
Llama3-8B	ACC	0.47	0.49	0.52	0.50	0.69	0.71	0.68	0.63	0.30	0.33	0.31	0.29
	ASR	0.01	0.02	0.01	-	0.03	0.06	0.05	-	0.04	0.05	0.04	-
	PAD	0.95	1.04	0.97	0.60	1.09	1.19	1.06	0.75	1.06	1.17	1.07	0.65
Vicuna-7B	ACC	0.49	0.52	0.51	0.53	0.68	0.72	0.70	0.71	0.49	0.50	0.48	0.51
	ASR	0.01	0.03	0.02	-	0.01	0.02	0.03	-	0.00	0.02	0.02	-
	PAD	1.74	2.22	1.84	1.16	1.91	1.95	1.99	0.75	1.87	2.34	1.98	0.21
GPT-3.5-turbo	ACC	0.48	0.52	0.52	0.47	0.75	0.75	0.74	0.78	0.40	0.40	0.38	0.39
	ASR	0.00	0.02	0.00	-	0.01	0.00	0.02	-	0.07	0.08	0.02	-
	PAD	0.95	1.04	0.97	0.61	1.09	1.19	1.13	0.75	1.06	1.17	1.07	0.65

gregated retrieval representation. Lower PAD values indicate successful mitigation and preserved semantics, while higher values signal greater semantic corruption, making PAD a key metric for attack strength and robustness.

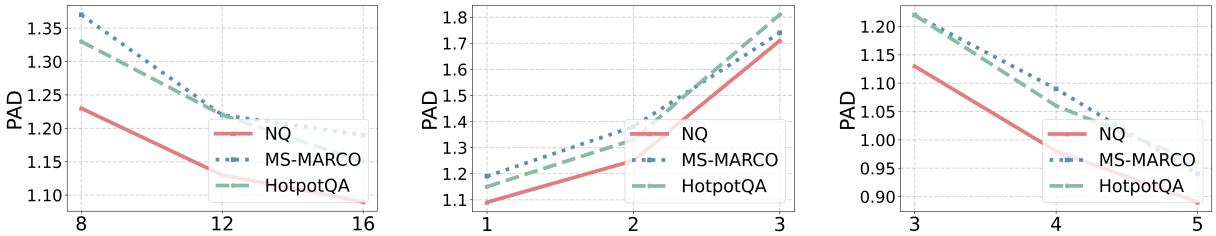
6.2 Overall Results

Main results of PRA-RAG. Table 1 summarizes the performance of PRA-RAG under various poisoning attacks across multiple datasets and LLMs. The results demonstrate that PRA-RAG consistently achieves low ASR while maintaining high ACC, indicating strong robustness across all settings. For instance, on the MS-MARCO dataset with Mistral-7B, PRA-RAG attains an ACC of 62% under PoisonedRAG, with an ASR as low as 1%. Additionally, PAD effectively quantifies the semantic impact of adversarial retrievals: across models and tasks, lower PAD values correspond to lower ASR and higher ACC, supporting its utility as a reliable indicator of both robustness and attack severity. Additionally, in Section C.2, we compare the effects of various embedding extraction methods on the final model performance.

PRA-RAG outperforms baselines. Table 2 presents a comparative evaluation of PRA-RAG against several representative baselines under poisoning attacks and clean settings. Across all datasets and attack settings, PRA-RAG consistently achieves the best overall performance, attaining higher ACC while significantly lowering the ASR. For instance, on the MSMARCO dataset with 20% poisoned retrievals, PRA-RAG maintains an ACC of 68% and suppresses ASR to just 1% with Vicuna-7B, outperforming RobustRAG and InstructRAG. Meanwhile, we observe that in the absence of poisoning attacks, the ACC_{clean} of our method remains largely unaffected. These results demonstrate that PRA-RAG not only offers stronger defense against poisoning attacks on retrieved texts but also preserves utility under normal conditions. Overall, existing methods either lack formal certification, trade usability for safety (over-refusal or over-filtering), or incur high cost with tight coupling to the LLM. PRA-RAG models retrieval poisoning as a set perturbation and performs robust aggregation on *combination embeddings* by selecting a majority-covering MEB center, yielding a certified PAD bound. Under the common regime, it consistently lowers ASR with low overhead, maintains competitive ACC, and withstands semantically-near poisoning.

Table 2: Performance of PRA-RAG and baselines under PoisonedRAG attacks (ACC, ASR) and clean settings (ACC_{clean}). Results are based on the same experimental setup here for consistency and may differ from the original reports. Best results are in **bold**.

Models	Defense	NQ			MS-MARCO			HotpotQA		
		ACC_{clean}	ACC	ASR	ACC_{clean}	ACC	ASR	ACC_{clean}	ACC	ASR
Mistral-7B	Vanilla RAG	0.61	0.26	0.46	0.58	0.33	0.18	0.55	0.17	0.48
	$RobustRAG_{Keyword}$	**0.67**	0.66	0.09	**0.61**	**0.66**	0.12	**0.57**	**0.56**	0.24
	$InstructRAG_{ICL}$	0.38	0.26	0.13	0.48	0.23	0.12	0.30	0.22	0.26
	ASTUTE RAG	0.48	0.47	0.05	0.58	0.48	0.05	0.35	0.35	0.09
	$PRA\text{-}RAG_{cat}$	0.57	0.47	0.09	0.59	0.59	0.04	0.50	0.41	0.11
	PRA-RAG	0.60	**0.62**	**0.01**	0.57	0.63	**0.01**	0.49	0.37	**0.03**
Vicuna-7B	Vanilla RAG	**0.67**	0.31	0.45	**0.73**	0.35	0.33	**0.55**	0.23	0.43
	$RobustRAG_{Keyword}$	0.48	0.40	0.08	0.59	0.40	0.07	0.44	0.31	0.10
	$InstructRAG_{ICL}$	0.33	0.10	0.11	0.54	0.33	0.20	0.31	0.09	0.11
	ASTUTE RAG	0.50	0.44	0.19	0.74	0.63	0.14	0.34	0.26	0.30
	$PRA\text{-}RAG_{cat}$	0.54	**0.53**	0.04	0.65	**0.71**	0.02	0.49	0.37	0.13
	PRA-RAG	0.53	0.49	**0.01**	0.63	0.68	**0.01**	0.50	**0.49**	**0.00**



(a) Retrieved Passages Top-K (b) Corruption Size ϵ (c) Subset Size n

Figure 2: Impact of Retrieval Passages, Corruption Size and Subset Size on PAD scores for Mistral-7B.

6.3 Ablation Study

Impact of retrieved passages Top-K, corruption size ϵ and subset size n. The number of retrieved texts Top-K and the number of poisoned texts ϵ significantly affect both attack effectiveness and defense performance. Figure 2a illustrates the effect of varying the number of retrieved texts (Top-$K = 8, 12, 16$) on the PAD score of PRA-RAG, with $\epsilon = 1$ fixed. As Top-K increases, the PAD score gradually declines, indicating enhanced robustness due to increased retrieval diversity. Figure 2b presents the opposite setting: with the number of retrieved texts fixed at $K = 16$, the PAD score increases as the number of poisoned texts rises from $\epsilon = 1$ to $\epsilon = 3$. This trend indicates that injecting more poisoned texts into the retrieval results strengthens the attack, thereby diminishing the robustness of PRA-RAG. Figure 2c illustrates the impact of different subset sizes on the performance of PRA-RAG under the setting where the number of retrieved texts is Top-$K = 12$ and the number of poisoned texts is $\epsilon = 1$. We evaluate cases where each combination contains 3, 4, or 5 texts. The results show a clear downward trend in PAD scores as the number of texts per subset increases. This indicates that incorporating more clean texts within each combination helps dilute the influence of poisoned content, thereby enhancing overall robustness. Further experimental data and analyses can be found in Section C.3.

Comparison of aggregation strategies. We compare two aggregation methods for the selected subset: a baseline that concatenates texts as input to the LLM (PRA-RAG$_{\text{cat}}$), and our method (PRA-RAG), which computes a weighted average of embeddings. As shown in Table 2, our approach achieves higher ACC and significantly reduces ASR by better mitigating the impact of poisoned texts. In Section C.4, we further compare and analyze the impact of different distance metrics used to compute d on the performance of our method.

Table 3: The table presents the average response latency of different RAG methods across datasets.

Dataset	ASTUTE RAG	InstructRAG$_{\text{ICL}}$	RobustRAG$_{\text{Keyword}}$	PRA-RAG
NQ	13.68s	6.89s	27.01s	**5.98s**
MS-MARCO	13.84s	7.15s	26.38s	**6.20s**
HotpotQA	16.78s	7.01s	25.87s	**6.17s**

6.4 Efficiency

In real-world applications, response time is critical to the user experience of RAG systems. In our experiments, we evaluated the average response latency over 100 query examples to compare different defense methods. As shown in Table 3, our proposed PRA-RAG achieves the lowest inference latency, demonstrating its practical efficiency while preserving strong robustness. In our framework, both the number of retrieved texts (Top-K) and the subset size have a significant impact on system efficiency. Therefore, we conduct a more in-depth analysis and empirical evaluation of the computational overhead in Section C.5.

7 Conclusion

This paper presents PRA-RAG, a provably robust aggregation method for defending against retrieval-level poisoning attacks in RAG systems. We theoretically derive an upper bound on the semantic deviation introduced by poisoned retrievals and propose Provable Average Deviation (PAD) as a unified metric for evaluating both robustness and attack strength. Unlike traditional defenses that depend on model outputs, PRA-RAG ensures semantic stability directly in the embedding space, providing a principled, model-agnostic robustness guarantee. Extensive experiments show that PRA-RAG significantly enhances robustness against poisoning attacks across diverse datasets and language models, consistently lowering attack success rates while preserving high answer accuracy.

Limitations.

Our work has the following limitations:

- This work selects the most robust subset of the retrieved texts as the final input to the LLM for answer generation. Although the selected subset may not always produce the optimal response from the LLM, as shown in our experiments, the impact on model utility remains limited while significantly enhancing robustness against poisoning attacks.

- Our approach enhances robustness by constructing subsets, which increases computational and response-time overhead. However, experiments show that as the number and size of subsets grow, the defense against poisoning attacks significantly improves. In practice, users can flexibly balance efficiency and security based on their needs.

- The effectiveness of our approach relies on the number of poisoned texts remaining below a certain threshold. Although an attacker could potentially bypass the defense through large-scale poisoning, this incurs substantial costs and significantly reduces the quality of the context, making it difficult for even humans to provide correct answers.

Ethics Statement

The goal of this work is to defend against retrieval-based poisoning attacks in RAG systems. All data used in this study is publicly available, ensuring no additional privacy concerns. The source code and software will be released as open-source. While this openness may expose the system to adaptive attacks, our approach can be further strengthened by incorporating additional internal and external information. Overall, we believe our method contributes to advancing the secure deployment of RAG systems.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

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

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

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

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

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

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

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

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

Ruochen Jiao, Shaoyuan Xie, Justin Yue, Takami Sato, Lixu Wang, Yixuan Wang, Qi Alfred Chen, and

Qi Zhu. 2024. Exploring backdoor attacks against large language model-based decision making. *arXiv preprint arXiv:2405.20774*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota.

Aounon Kumar and Tom Goldstein. 2021. Center smoothing: Certified robustness for networks with structured outputs. *Advances in Neural Information Processing Systems*, 34:5560–5575.

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

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

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

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

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

Avital Shafran, Roei Schuster, and Vitaly Shmatikov. 2024. Machine against the rag: Jamming retrieval-augmented generation with blocker documents. *arXiv preprint arXiv:2406.05870*.

Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Song Wang, Jundong Li, Tianlong Chen, and Huan Liu. 2024. Glue pizza and eat rocks-exploiting vulnerabilities in retrieval-augmented generative models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1610–1626.

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

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arık. 2025. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *Preprint*, arXiv:2410.07176.

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

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

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

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

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

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

Collin Zhang, Tingwei Zhang, and Vitaly Shmatikov. 2025. Adversarial decoding: Generating readable documents for adversarial objectives. *Preprint*, arXiv:2410.02163.

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

Huichi Zhou, Kin-Hei Lee, Zhonghao Zhan, Yue Chen, and Zhenhao Li. 2025. Trustrag: Enhancing robustness and trustworthiness in rag. *arXiv preprint arXiv:2501.00879*.

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

A Proof

A.1 Proof of Theorem 1

Consider two balls $\mathcal{B}(\mathcal{F}(\mathcal{X}'), r^*(\mathcal{X}'))$ and $\mathcal{B}(\mathcal{F}(\mathcal{X}), R)$. According to the definitions of R and $r^*(\mathcal{X}')$, these two balls must share at least one common element, denoted as w^*. Since the

defined distance based on vector angles satisfies the triangle inequality, we have:

$$d(\mathcal{F}(\mathcal{X}), \mathcal{F}(\mathcal{X}')) \leq d(\mathcal{F}(\mathcal{X}), w^*) + d(\mathcal{F}(\mathcal{X}'), w^*) \leq R + r^*(\mathcal{X}'). \tag{11}$$

Since the ball $\mathcal{B}(\mathcal{F}(\mathcal{X}), R)$ contains at least a majority of elements in $\mathcal{V}_{\mathcal{X}',n}$, the smallest radius $r^*(\mathcal{X}')$ of a ball that contains a majority of elements in $\mathcal{V}_{\mathcal{X}',n}$ must not be greater than R, i.e., $r^*(\mathcal{X}') \leq R$. Therefore,

$$d(\mathcal{F}(\mathcal{X}), \mathcal{F}(\mathcal{X}')) \leq 2R. \tag{12}$$

A.2 Proof of Theorem 2

Since the radius of the β-MEB differs from the optimal radius by a factor β, i.e., the radius of the β-MEB over $\mathcal{V}_{\mathcal{X},n}$ is $\beta r^*(\mathcal{X}')$, by the same reasoning as in Theorem 1, we have:

$$d(\widehat{\mathcal{F}}(\mathcal{X}), \widehat{\mathcal{F}}(\mathcal{X}')) \leq d(\widehat{\mathcal{F}}(\mathcal{X}), w^*) + d(\widehat{\mathcal{F}}(\mathcal{X}'), w^*) \leq \widehat{R} + \beta r^*(\mathcal{X}') \leq (1+\beta)\widehat{R}. \tag{13}$$

B Details

B.1 Details of Experiment Setup

Dataset. Detailed statistics are shown in Table 4.

Table 4: Statics of datasets.

Datasets	Texts in knowledge database	Questions
NQ (Kwiatkowski et al., 2019)	2,681,468	3,452
MS-MARCO (Bajaj et al., 2016)	5,233,329	7,405
HotpotQA (Yang et al., 2018)	8,841,823	6,980

Default Setting. Unless otherwise specified, our experiments use the following default settings: the Contriever (Izacard et al., 2021) retriever is employed to retrieve the top 8 texts most similar to the query from the MS-MARCO dataset, with poisoned texts generated according to the PoisonedRAG method constituting 20% (rounded down) of the retrieval results. Subsequently, multiple subsets are sampled from the retrieved texts, each containing 3 texts. Open-source LLMs can directly obtain embeddings through their embedding layers and generate the final robust aggregated representation. For black-box models (e.g., GPT-3.5-turbo), we use Llama3-8B in our experiments to extract embeddings and decode the robust aggregation results into text to serve as context. Finally, the Mistral-7B language model generates responses based on these retrieved texts. Each result is averaged over 10 runs under a consistent setup.

Implementation and Resources. We report the mean performance over 10 independent runs and assess the statistical significance of improvements or degradations using the Wilcoxon signed-rank test at a significance level of $p \leq 0.05$. We conduct experiments on a server with 64 AMD EPYC 9654 CPUs at 2.40–3.70 GHz, 512 GB of DDR5 RAM (assumed based on high-core-count server standards), and four NVIDIA RTX A100 GPUs, each with 80 GB GDDR6 memory.

B.2 Prompt

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

Figure 3: The prompt used in RAG to make an LLM generate an answer based on the retrieved texts.

> You are given a question and two answers. Determine whether the two answers are semantically consistent with each other, whether they express the same stance, conclusion, or outcome regarding the question. If they are consistent, output "yes". If they are inconsistent or contradictory, output "no".
> Query: [question]
> Answers: [correct/incorrect answer & response of LLM]
> Answer:

Figure 4: The prompt used in RAG to determine whether the LLM's response is correct or corresponds to the attacker's target response.

B.3 Poisoned Texts Generation

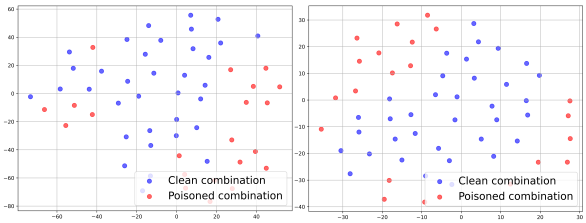
To comprehensively evaluate the robustness of our approach against different poisoned text generation strategies, we adopt three representative strategies, including PoisonedRAG (Zou et al., 2024), Adversarial Decoding (Zhang et al., 2025), and Denial-Of-Service Attack (Shafran et al., 2024). In PoisonedRAG (Zou et al., 2024) method, the attacker begins by selecting a target question and its corresponding incorrect answer, then crafts poisoned texts to satisfy two key requirements: (1) being retrievable by the retriever and (2) successfully misleading the language model into producing the incorrect answer. Adversarial Decoding (Zhang

et al., 2025) is a token-level beam search framework that integrates continuous, task-specific scorers into standard decoding, enabling the generation of fluent, low-detectability texts that jointly optimize retrieval similarity and adversarial generation objectives. Denial-Of-Service Attack (Shafran et al., 2024) inserts a single blocker document consisting of a retrieval component to ensure retrieval and a jamming component to trigger refusal responses, crafted through instruction injection, oracle generation, or black-box optimization. In our experiments, we adopt the default parameters of these methods to generate the corresponding poisoned texts, ensuring effective poisoning.

C Additional Experimental Results

C.1 Distribution of Different Combinations

Figure 5 illustrates the distributional differences between combinations containing poisoned texts and clean combinations. It is evident that the poisoned texts, intentionally crafted to induce attacker-desired responses, exhibit a significant shift in their embedding vectors compared to clean texts.



(a) Mistral-7B (b) Vicuna-7B

Figure 5: Embedding Distribution of Retrieved Text Combinations (Red: With Poisoned Texts; Blue: Without Poisoned Texts)

C.2 Different Embedding Extraction Methods

Table 5 compares the impact of different embedding acquisition methods on the final performance of PRA-RAG. In our experiments with Mistral-7B, LLaMA-3-8B, and GPT-3.5-turbo, embeddings are obtained either from each model's native embedding module or from the Contriever retriever. While the results show minimal variation in ACC and ASR, they reveal substantial discrepancies in PAD, reflecting the differences in how embeddings are computed. This highlights the importance of standardizing the embedding source in practice to ensure consistent robustness. This further reflects that our method is robust and generalizable across different embedding models.

Table 5: The table summarizes the performance of PRA-RAG using different embedding extraction methods under GPT-3.5-turbo.

Models	Methods	NQ			MS-MARCO			HotpotQA		
		ACC	ASR	PAD	ACC	ASR	PAD	ACC	ASR	PAD
Mistral-7B	Contriever	0.55	0.03	2.52	0.62	0.02	2.62	0.44	0.08	2.71
	Mistral-7B	0.62	0.01	1.23	0.63	0.01	1.37	0.37	0.03	1.33
Llama3-8B	Contriever	0.52	0.04	2.52	0.65	0.00	2.62	0.33	0.06	2.71
	Llama3-8B	0.47	0.01	0.95	0.69	0.03	1.09	0.30	0.04	1.06
GPT-3.5-turbo	Contriever	0.47	0.02	2.52	0.78	0.01	2.62	0.32	0.09	2.71
	Llama3-8B	0.48	0.00	0.95	0.75	0.00	1.09	0.40	0.07	1.06

C.3 Impact of Different Parameters

Impact of Retrieved Passages Top-K. Table 6 presents the impact of varying the number of retrieved texts on the performance of PRA-RAG under the setting where the number of poisoned texts is 1 and the subset size is 3. The experimental results show that the changes in ACC and ASR vary across different large language models and datasets, lacking a consistent trend. However, the PAD metric exhibits a consistently decreasing trend. This is because increasing the number of retrieved texts reduces the proportion of poisoned texts, thereby diminishing their overall influence.

Table 6: The table shows the performance of PRA-RAG with different numbers of retrieved texts.

Models	Top-K	NQ			MS-MARCO			HotpotQA		
		ACC	ASR	PAD	ACC	ASR	PAD	ACC	ASR	PAD
Mistral-7B	8	0.62	0.01	1.23	0.63	0.01	1.37	0.37	0.03	1.33
	12	0.54	0.00	1.13	0.66	0.00	1.22	0.40	0.02	1.22
	16	0.53	0.00	1.09	0.53	0.00	1.19	0.40	0.02	1.15
Vicuna-7B	8	0.49	0.01	1.74	0.68	0.01	1.91	0.49	0.00	1.87
	12	0.53	0.00	1.66	0.68	0.00	1.73	0.52	0.01	1.76
	16	0.52	0.00	1.59	0.70	0.00	1.69	0.51	0.01	1.68

Impact of corruption size ϵ. Table 8 reports the effect of the number of poisoned texts on the robustness of PRA-RAG with the number of retrieved texts fixed at 12. The results show that as the number of poisoned texts increases, both ASR and PAD rise across models and datasets, indicating stronger attacks and reduced robustness.

Impact of subset size n. Table 8 presents the impact of the number of texts per sampled subset on the results when the total number of retrieved texts is fixed at Top-$K = 12$. The experimental results show that as the number of texts included in each subset increases, the context provided to the LLM

Table 7: The table shows the performance of PRA-RAG with different numbers of poisoned texts.

Models	Corruption Size	NQ			MS-MARCO			HotpotQA		
		ACC	ASR	PAD	ACC	ASR	PAD	ACC	ASR	PAD
Mistral-7B	1	0.53	0.00	1.09	0.53	0.00	1.19	0.40	0.02	1.15
	2	0.51	0.00	1.25	0.56	0.00	1.38	0.38	0.05	1.33
	3	0.49	0.04	1.71	0.53	0.03	1.74	0.33	0.09	1.81
Vicuna-7B	1	0.52	0.00	1.59	0.70	0.00	1.69	0.51	0.01	1.68
	2	0.48	0.00	1.79	0.71	0.00	1.93	0.50	0.02	1.89
	3	0.50	0.01	2.23	0.70	0.02	2.45	0.50	0.04	2.43

Table 8: The table shows the performance of PRA-RAG with different subset size.

Models	Subset Size	NQ			MS-MARCO			HotpotQA		
		ACC	ASR	PAD	ACC	ASR	PAD	ACC	ASR	PAD
Mistral-7B	3	0.54	0.00	1.13	0.66	0.00	1.22	0.40	0.02	1.22
	4	0.60	0.01	0.98	0.63	0.00	1.09	0.44	0.01	1.06
	5	0.63	0.00	0.89	0.71	0.01	0.94	0.43	0.01	0.96
Vicuna-7B	3	0.53	0.00	1.66	0.68	0.00	1.73	0.52	0.01	1.76
	4	0.55	0.01	1.43	0.67	0.00	1.53	0.54	0.00	1.53
	5	0.58	0.00	1.27	0.69	0.01	1.38	0.58	0.01	1.37

becomes richer, leading to an upward trend in both ACC and PAD.

C.4 Comparison of different distance metrics.

In PRA-RAG, we recommend using *cosine similarity* to compute the distance d between vectors, as it emphasizes vector direction rather than magnitude, making it more effective for capturing semantic similarity. To thoroughly evaluate the impact of different distance metrics on system performance, we compare *Manhattan distance*, *Euclidean distance*, and *cosine similarity* in our experiments. As shown in Table 9, the cosine similarity-based method consistently outperforms the others across all evaluation metrics, confirming its effectiveness and suitability for this task.

Table 9: The table presents the performance of PRA-RAG with different distance metrics.

Distance Metrics	NQ		MS-MARCO		HotpotQA	
	ACC	ASR	ACC	ASR	ACC	ASR
Manhattan distance	0.58	0.01	0.59	0.02	0.36	0.12
Euclidean distance	0.61	0.03	0.60	0.02	0.36	0.11
cosine similarity	**0.62**	0.01	**0.63**	**0.01**	**0.37**	**0.03**

C.5 Analysis of Computational Overhead

Table 10: Computational overhead of PRA-RAG under different Top-K retrieval settings, with subset size fixed at 3 and a poisoning rate of 20%.

Models	Top-K	NQ	MS-MARCO	HotpotQA
Mistral-7B	8	5.98s	6.20s	6.17s
	12	18.47s	24.88s	22.14s
	16	68.05s	65.48s	72.16s
Vicuna-7B	8	3.37s	2.91s	3.02s
	12	12.13s	11.58s	12.23s
	16	64.93s	67.13s	59.25s

Table 11: Computational overhead of PRA-RAG with Top-K = 12 retrieval under a poisoning rate of 10%, evaluated across different subset sizes.

Model	Subset Size	NQ	MS-MARCO	HotpotQA
Mistral-7B	3	18.71s	18.10s	17.76s
	4	55.25s	55.30s	57.89s
	5	125.55s	131.37s	124.96s
Vicuna-7B	3	11.88s	12.16s	11.84s
	4	47.92s	46.12s	52.40s
	5	116.09s	121.47s	117.85s

Table 10 compares the computational overhead under different numbers of retrieved texts, with the subset size $n = 3$. The results show that as Top-K increases, the number of combinations grows rapidly, leading to a near-linear increase in computation time. Specifically, when Top-K is set to 8, 12, and 16, the average generation times are 5.98 s, 18.47 s, and 68.05 s, respectively. Table 11 compares the computational overhead under a fixed Top-K retrieval size of 12 with varying subset sizes. The results show that the computation time no longer correlates linearly with the number of combinations, as changes in subset size also affect the overall time required for the final RAG response generation. Both the number of retrieved texts (Top-K) and the subset size (n) have a significant impact on the response time of our approach. Nevertheless, as shown in Table 3, our method still outperforms existing baselines in terms of efficiency under certain settings (e.g., Top-$K = 8$, $n = 3$), which represent typical real-world scenarios. Although increasing Top-K and n leads to higher computational overhead, it also enhances system security. As shown in Figure 2, the PAD value decreases as Top-K and n increase, indicating higher security. Overall, users can flexibly balance efficiency and security by selecting appropriate values of Top-K and n based on practical needs.