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

**Xue Tan[1,3], Yi Zheng[1], Chang Huo[1], Yunruo Zhang[3], Yu Liu[1,3], Hao Luan[1,3], Zhuyang Yu[1,3], Xiaoyan Sun[2]✉, Ping Chen[3]✉, Jun Dai[2]✉**

[1]School of Computer Science, Fudan University, Shanghai, China,

[2]Department of Computer Science, Worcester Polytechnic Institute, MA, USA,

[3]Institute of Big Data, Fudan University, Shanghai, China,

**Corresponding authors:** pchen@fudan.edu.cn, xsun7@wpi.edu, jdai@wpi.edu

## 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 (Zou et al., 2024; Greshake et al., 2023; Tan et al., 2024). 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) guarantees robustness for LLM outputs but incurs high computational overhead by requiring multiple LLM generations.

In this work, we propose PRA-RAG, a **P**rovably **R**obust retrieval **A**ggregation algorithm for **RAG** systems, 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



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 36002–36017

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 selected subset's embeddings yields a robust representation that captures consensus and reduces outlier influence.

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. The key contributions of our work are:

- **Theory:** We theoretically characterize and certify the maximum semantic deviation caused by poisoned retrievals, providing a principled metric to evaluate system robustness and attack effectiveness.
- **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 as a prompt for the LLM to generate a response.

**Retrieval Corruption Attack.** The use of external knowledge databases in RAG introduces security vulnerabilities. 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., 2025c) performs RAG poisoning and LLM guard evasion by generating readable adversarial texts through adversarial decoding. BadRAG (Xue et al., 2024) employs white-box optimization to attack the retriever and uses handcrafted documents to target the generator. PR-Attack (Jiao et al., 2025) jointly optimizes a trigger and a small set of poisoned texts to stealthily induce the RAG system to generate a target response. CorruptRAG (Zhang et al., 2025a) misleads the model into producing targeted false content by prompting the LLM to label the correct answer as outdated and to generate a fabricated latest but incorrect answer. The aforementioned attack methods are highly effective. To assess our defense, we select representative attacks to simulate realistic 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

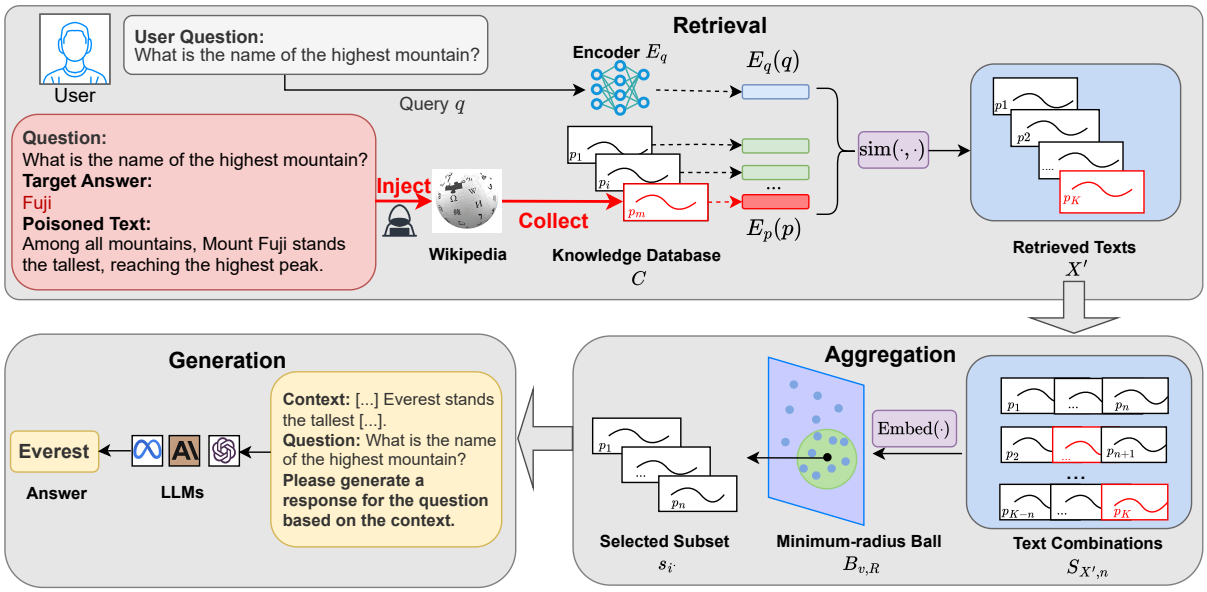



Figure 1: The PRA-RAG pipeline against poisoned retrieval.

that enhances generation quality and factual accuracy through self-reflection mechanisms within the LLM. RAGForensics (Zhang et al., 2025b) employs an iterative retrieval mechanism that combines an LLM with carefully designed chain-of-thought prompts to perform round-by-round judgment and filtering of candidate retrieved texts. RAGuard (Cheng et al., 2025) integrates a two-stage filtering mechanism based on chunk-level perplexity and text similarity, which can effectively identify and remove malicious poisoned content from the retrieved results. Other 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, these defense strategies lack quantitative robustness evaluations and theoretical guarantees for RAG systems. RobustRAG (Xiang et al., 2024) enhances RAG robustness by using multiple LLMs and a voting mechanism to filter poisoned content, providing theoretical guarantees but at the cost of significant computational overhead and challenging output aggregation. We propose a provably robust RAG algorithm that quantitatively evaluates robustness and provides 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 corpus, and let $\mathcal{R}$ be the retriever that returns the Top-$K$ passages for a query $q$, denoted as $\mathcal{X}_K(q;\mathcal{D}) = \{p_1,\ldots,p_K\}$. Under poisoning, the adversary injects a set of passages $\Gamma$ 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 use $n$ to denote the subset size in our 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 goal is to poison the corpus so that, for each $q_i \in Q$, the RAG system generates the attacker-specified answer $a_i$ when using retrieved context from the contaminated corpus $\mathcal{D}'$. For instance, given the query $q_i$: *"What is the name of the highest mountain?"*, the attacker attempts to make the system output 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 $\varepsilon$ 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 $\varepsilon$ 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 aims to limit the impact of poisoned passages while maintaining the 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.

**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 $\theta$ 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, we cast retrieval corruption as a *set perturbation* problem, where an adversary alters at most $\varepsilon$ items in the Top-$K$ set, and propose a geometric, model-agnostic retrieval aggregation that operates on *combination embeddings*. 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 $s_i$. The concatenation order of $\mathbf{e}_i$ within $s_i$ strictly follows the relative order of their corresponding texts $p_i$.

We define the distance between two vectors $u, v \in \mathcal{V}_{\mathcal{X},n}$ as the angular distance: $d(u,v) = \arccos(\cos(u,v)) \in [0, \pi]$, where $\cos(u,v) = \frac{u \cdot v}{\|u\|\|v\|}$ denotes the 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 $\mathcal{X}$, $\mathcal{B}(z, R)$ 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

**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 $\mathbf{sim}_i \in R^{L-1}$
  **for** $j = 1$ to $L$ and $j \neq i$ **do**
    $d_{ij} \leftarrow \arccos(\cos(v_i, v_j))$
    Store $d_{ij}$ in $\mathbf{sim}_i$
  **end for**
  Sort $\mathbf{sim}_i$ in ascending order
  Let $m_i \leftarrow$ the $k$-th smallest value in $\mathbf{sim}_i$, $k = \lfloor \frac{L}{2} \rfloor$
**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

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 $\varepsilon$ 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 $\varepsilon$ 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. To ensure that the empirical radius $\hat{R}$ satisfies the theoretical majority condition, Lemma 1 (Section A.4) guarantees that the majority ball contains a strict majority of clean combinations. Combined with the geometric separability assumption (Section A.5), this ensures the center is anchored to the clean subset. 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 $d$ 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 $\varepsilon$ passages, that is $\|\mathcal{X}' - \mathcal{X}\|_0 \leq \varepsilon$. Define the robustness radius $R$ as the smallest value satisfying

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

We present the following theorem:

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

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

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 $\beta$-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 $\beta$. Similarly, the approximation of $R$, denoted as $\widehat{R}$, is defined to satisfy:

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

Then, we have the following theorem:

**Theorem 2.** For any $\mathcal{X}'$ satisfying $\|\mathcal{X}' - \mathcal{X}\|_0 \leq \varepsilon$, 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:

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

Note that while the theoretical formulation is based on the full set of $L = \binom{K}{n}$ combinations, our framework naturally generalizes to a Monte Carlo sampling approach for larger values of $K$ and $n$. In such cases, we sample $m \ll L$ subsets, which provides a statistically bounded approximation of the optimal robustness (see Section A.3) and our experimental results further corroborate its effectiveness (see Section C.3).

**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 $\varepsilon$ (with $\binom{K}{n} < 2\binom{K-\varepsilon}{n}$).
**Output**: Certified maximum deviation $d$.
Let $\mathcal{S}_{\mathcal{X}',n} = \mathrm{Comb}(\mathcal{X}', n)$
Initialize $\mathcal{V}_{\mathcal{X}',n} \leftarrow \emptyset$
**for** each subset $s_i \in \mathcal{S}_{\mathcal{X}',n}$ **do**
  $v_i \leftarrow \mathrm{Concat}([\, Embed(p) \mid p \in s_i \,])$
  $\mathcal{V}_{\mathcal{X}',n} \leftarrow \mathcal{V}_{\mathcal{X}',n} \cup \{v_i\}$
**end for**
Let $v_{\text{orig}}$ 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**
  $d_i \leftarrow \arccos(\cos(v_{\text{orig}}, v_i))$
  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.** To evaluate the effectiveness of the proposed method, we utilize three question-answering datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019), MS-MARCO (Bajaj et al., 2016), and HotpotQA (Yang et al., 2018).

**LLM.** We evaluate PRA-RAG with different LLMs, including Mistral-7B (Jiang et al., 2023), Llama3-8B, Vicuna-7B (Chiang et al., 2023), Qwen3-8B (Yang et al., 2025), GPT-3.5-Turbo (Brown et al., 2020), and GPT-5-mini.

**Attackers.** We conduct a systematic evaluation of PRA-RAG's defense effectiveness against different 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., 2025c), (3) Denial-Of-Service Attack: DoS Attack (Shafran et al., 2024), and (4) CorruptRAG (Zhang et al., 2025a). Further details of these attack methods are provided in Section B.3.

**Defenders.** We demonstrate the advantages of our approach by comparing it with representative state-of-the-art baselines, including RobustRAG (Xiang et al., 2024), InstructRAG (Wei et al., 2024), AstuteRAG (Wang et al., 2025), TrustRAG (Zhou et al., 2025), and RAGForensics (Zhang et al., 2025b) with their default configurations.

**Evaluation Metrics.** We conduct a comprehensive evaluation of the proposed method using three metrics: (1) Adversarial Accuracy (**ACC**) assesses the

Table 1: Performance of PRA-RAG across different attack strategies and datasets under various LLMs (Top-$K = 8$, subset size $n = 3$, poisoning rate = 20%). Attack abbreviations: PR = PoisonedRAG, DoS = DoS Attack, AD = AdvDec, CR = CorruptRAG, Cln = Clean.

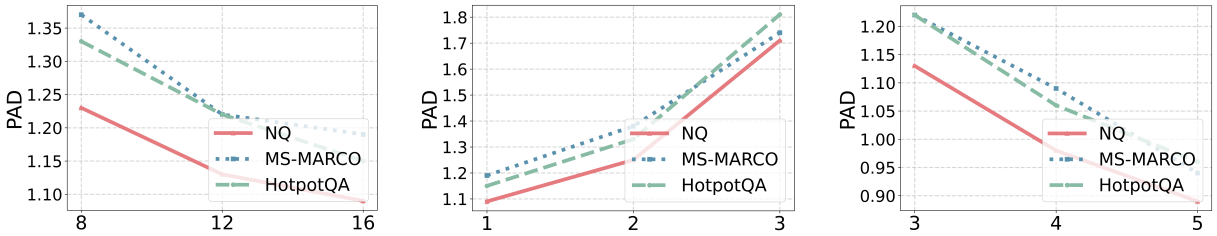
| | | **NQ** | | | | | **MS-MARCO** | | | | | **HotpotQA** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **Met.** | PR | DoS | AD | CR | Cln | PR | DoS | AD | CR | Cln | PR | DoS | AD | CR | Cln |
| Mistral-7B | ACC | 0.62 | 0.57 | 0.55 | 0.55 | 0.60 | 0.63 | 0.65 | 0.61 | 0.59 | 0.57 | 0.37 | 0.33 | 0.39 | 0.41 | 0.38 |
| | ASR | 0.01 | 0.02 | 0.02 | 0.04 | - | 0.01 | 0.03 | 0.02 | 0.00 | - | 0.03 | 0.01 | 0.02 | 0.03 | - |
| | PAD | 1.23 | 1.43 | 1.30 | 1.21 | 0.79 | 1.37 | 1.46 | 1.51 | 1.39 | 0.90 | 1.33 | 1.55 | 1.43 | 1.31 | 0.83 |
| Llama3-8B | ACC | 0.47 | 0.49 | 0.52 | 0.49 | 0.50 | 0.69 | 0.71 | 0.68 | 0.67 | 0.63 | 0.30 | 0.33 | 0.31 | 0.30 | 0.29 |
| | ASR | 0.01 | 0.02 | 0.01 | 0.03 | - | 0.03 | 0.06 | 0.05 | 0.00 | - | 0.04 | 0.05 | 0.04 | 0.04 | - |
| | PAD | 0.95 | 1.04 | 0.97 | 0.93 | 0.60 | 1.09 | 1.19 | 1.06 | 1.11 | 0.75 | 1.06 | 1.17 | 1.07 | 1.04 | 0.65 |
| Vicuna-7B | ACC | 0.49 | 0.52 | 0.51 | 0.51 | 0.53 | 0.68 | 0.72 | 0.70 | 0.71 | 0.71 | 0.49 | 0.50 | 0.48 | 0.47 | 0.51 |
| | ASR | 0.01 | 0.03 | 0.02 | 0.00 | - | 0.01 | 0.02 | 0.03 | 0.00 | - | 0.00 | 0.02 | 0.02 | 0.04 | - |
| | PAD | 1.74 | 2.22 | 1.84 | 1.73 | 1.16 | 1.91 | 1.95 | 1.99 | 1.90 | 0.75 | 1.87 | 2.34 | 1.98 | 1.85 | 0.21 |
| Qwen3-8B | ACC | 0.53 | 0.61 | 0.64 | 0.55 | 0.64 | 0.69 | 0.65 | 0.67 | 0.63 | 0.68 | 0.49 | 0.45 | 0.40 | 0.42 | 0.45 |
| | ASR | 0.06 | 0.03 | 0.02 | 0.01 | - | 0.00 | 0.00 | 0.01 | 0.00 | - | 0.09 | 0.05 | 0.02 | 0.04 | - |
| | PAD | 1.01 | 0.91 | 1.05 | 0.98 | 0.68 | 0.95 | 0.89 | 0.98 | 1.01 | 0.71 | 0.95 | 0.88 | 0.91 | 0.84 | 0.62 |
| GPT-3.5 -turbo | ACC | 0.48 | 0.52 | 0.52 | 0.45 | 0.47 | 0.75 | 0.75 | 0.74 | 0.70 | 0.78 | 0.40 | 0.40 | 0.38 | 0.35 | 0.39 |
| | ASR | 0.00 | 0.02 | 0.00 | 0.03 | - | 0.01 | 0.00 | 0.02 | 0.00 | - | 0.07 | 0.08 | 0.02 | 0.06 | - |
| | PAD | 0.95 | 1.04 | 0.97 | 0.93 | 0.61 | 1.09 | 1.19 | 1.13 | 1.11 | 0.75 | 1.06 | 1.17 | 1.07 | 1.04 | 0.65 |
| GPT-5 -mini | ACC | 0.55 | 0.47 | 0.51 | 0.47 | 0.51 | 0.59 | 0.53 | 0.54 | 0.49 | 0.58 | 0.43 | 0.41 | 0.41 | 0.43 | 0.49 |
| | ASR | 0.04 | 0.03 | 0.03 | 0.02 | - | 0.00 | 0.00 | 0.00 | 0.01 | - | 0.09 | 0.07 | 0.08 | 0.03 | - |
| | PAD | 0.93 | 0.98 | 0.91 | 1.05 | 0.62 | 1.11 | 1.24 | 1.17 | 1.21 | 0.69 | 1.04 | 0.96 | 1.10 | 1.05 | 0.72 |

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 aggregated 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, lower PAD values correspond to lower ASR and higher ACC, supporting its utility as a reliable indicator of both robustness and attack severity.

**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 $\text{ACC}_{\text{clean}}$ 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




(a) Retrieved Passages Top-$K$ (b) Corruption Size $\varepsilon$ (c) Subset Size $n$

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

Table 2: Performance of PRA-RAG and baselines under PoisonedRAG attacks (ACC, ASR) and clean settings ($\text{ACC}_{\text{clean}}$). Results are based on the same experimental setup here for consistency and may differ from the original reports (Top-$K = 8$, poisoning rate = 20%). Best results are in **bold**.

| Models | Defense | NQ | | | MS-MARCO | | | HotpotQA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\text{ACC}_{\text{clean}}$ | ACC | ASR | $\text{ACC}_{\text{clean}}$ | ACC | ASR | $\text{ACC}_{\text{clean}}$ | ACC | ASR |
| Mistral-7B | Vanilla RAG | 0.61 | 0.26 | 0.46 | 0.58 | 0.33 | 0.18 | 0.55 | 0.17 | 0.48 |
| | $\text{RobustRAG}_{\text{Keyword}}$ | **0.67** | 0.66 | 0.09 | 0.61 | **0.66** | 0.12 | **0.57** | **0.56** | 0.24 |
| | $\text{InstructRAG}_{\text{ICL}}$ | 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 |
| | TrustRAG | 0.65 | **0.62** | 0.03 | **0.67** | 0.64 | 0.08 | 0.50 | 0.45 | 0.10 |
| | RAGForensics | 0.41 | 0.41 | 0.07 | 0.62 | 0.65 | 0.03 | 0.45 | 0.46 | 0.10 |
| | $\text{PRA-RAG}_{\text{cat}}$ | 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 |
| | $\text{RobustRAG}_{\text{Keyword}}$ | 0.48 | 0.40 | 0.08 | 0.59 | 0.40 | 0.07 | 0.44 | 0.31 | 0.10 |
| | $\text{InstructRAG}_{\text{ICL}}$ | 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 |
| | TrustRAG | 0.51 | 0.49 | 0.06 | 0.62 | 0.55 | 0.06 | 0.36 | 0.30 | 0.07 |
| | RAGForensics | 0.60 | **0.58** | 0.03 | 0.68 | **0.71** | 0.04 | 0.51 | **0.51** | 0.06 |
| | $\text{PRA-RAG}_{\text{cat}}$ | 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** |

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.

### 6.3 Ablation Study

**Impact of retrieved passages Top-$K$, corruption size $\varepsilon$ and subset size $n$.** The number of retrieved texts Top-$K$ and poisoned texts $\varepsilon$ 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 $\varepsilon = 1$ 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 Top-$K = 12$, the PAD score increases as the number of poisoned texts rises from $\varepsilon = 1$ to $\varepsilon = 3$. 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 $\varepsilon = 1$. 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.

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** |

**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.

### 6.4 Efficiency

In real-world applications, response time is critical to the user experience of RAG systems. However, most existing defense strategies often incur varying levels of inference overhead. 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.2. Additionally, Section C.3 presents the computational overhead and analysis under Monte Carlo sampling.

## 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.

**Acknowledgment.**

Xue Tan, Yi Zheng, Chang Huo, Yunruo Zhang, Yu Liu, Hao Luan, Zhuyang Yu, and Ping Chen were funded in part by the National Key R&D Program of China 2023YFB3107404.

## 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.

Zirui Cheng, Jikai Sun, Anjun Gao, Yueyang Quan, Zhuqing Liu, Xiaohua Hu, and Minghong Fang. 2025. Secure retrieval-augmented generation against poisoning attacks. *arXiv preprint arXiv:2510.25025*.

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.

Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30.

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*.

Yang Jiao, Xiaodong Wang, and Kai Yang. 2025. Pr-attack: Coordinated prompt-rag attacks on retrieval-augmented generation in large language models via bilevel optimization. *arXiv preprint arXiv:2504.07717*.

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*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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*.

Baolei Zhang, Yuxi Chen, Minghong Fang, Zhuqing Liu, Lihai Nie, Tong Li, and Zheli Liu. 2025a. Practical poisoning attacks against retrieval-augmented generation. *arXiv preprint arXiv:2504.03957*.

Baolei Zhang, Haoran Xin, Minghong Fang, Zhuqing Liu, Biao Yi, Tong Li, and Zheli Liu. 2025b. Traceback of poisoning attacks to retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2025*, pages 2085–2097.

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

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 Theoretical Analysis and Proofs

## 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:

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

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

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

### A.3 Scalability and Approximation via Monte-Carlo Sampling

While the theoretical formulation in Section 5 relies on the full set of $L = \binom{K}{n}$ combinations, the computational cost grows combinatorially as $K$ and $n$ increase. To address this bottleneck, we introduce a Monte-Carlo sampling strategy: instead of enumerating all $L$ combinations, we uniformly sample $m \ll L$ subsets to estimate the robust aggregation. Below, we establish a strict theoretical lower bound on the sampling approximation and provide an empirically supported upper bound, demonstrating how this strategy effectively approximates the optimal robustness radius.

**Lower Bound of Monte-Carlo Approximation.** Let $\mathcal{V}_{\mathcal{X},n}$ be the full set of embedding vectors generated from $\binom{K}{n}$ combinations, and let $R$ be the radius of the Minimum Enclosing Ball (MEB) covering the majority ($> 50\%$) of $\mathcal{V}_{\mathcal{X},n}$. Let $\mathcal{S} \subset \mathcal{V}_{\mathcal{X},n}$ be a uniformly random sample of size $m$, and let $R^*$ be the radius of the MEB covering the majority of $\mathcal{S}$. For any $\tau \in (0,\, 0.5)$ and failure probability $\eta > 0$, if the sample size satisfies $m \geq \frac{1}{2\tau^2} \ln \frac{2}{\eta}$ and the majority coverage fraction $q > 0.5 + \tau$, then with probability at least $1 - \eta$:

$$R^* \leq R. \tag{14}$$

*Proof of Lower Bound.* For any fixed ball $\mathcal{B}$ in the embedding space, let $q$ denote the fraction of $\mathcal{V}_{\mathcal{X},n}$ within $\mathcal{B}$, and $\hat{q}$ the fraction in $\mathcal{S}$. By Hoeffding's Inequality (Hoeffding, 1963):

$$\mathbb{P}(|\hat{q} - q| \geq \tau) \leq 2\exp(-2m\tau^2). \tag{15}$$

Under the condition $m \geq \frac{1}{2\tau^2} \ln \frac{2}{\eta}$, we have $2\exp(-2m\tau^2) \leq \eta$. For instance, with $m = 200$ and $\tau = 0.1$, the failure probability is below $0.04$.

*Lower Bound Derivation.* Let $\mathcal{B}^*$ be the optimal ball with radius $R$ covering a fraction $q > 0.5 + \tau$ of $\mathcal{V}_{\mathcal{X},n}$. By the coverage concentration above, $\mathcal{B}^*$ covers a fraction at least $q - \tau > 0.5$ of $\mathcal{S}$ with high probability. Thus $R$ is a feasible majority-covering radius for $\mathcal{S}$. Since $R^*$ is the minimum such radius over $\mathcal{S}$:

$$R^* \leq R. \tag{16}$$

**Tightness of the Approximation.** The lower bound $R^* \leq R$ guarantees that sampling does not overestimate the robustness radius. A complementary strict upper bound of the form $R \leq (1+\tau)\, R^*$ would require guaranteeing that the random sample $\mathcal{S}$ captures the extremal boundary points that determine the MEB of the full set. Since these boundary points may constitute a vanishing fraction of $\mathcal{V}_{\mathcal{X},n}$, such a deterministic worst-case guarantee cannot be established from uniform sampling alone without additional structural assumptions.

However, in our setting, the Geometric Separability Assumption (Section A.5) implies that clean combination embeddings form a concentrated cluster, which facilitates effective geometric approximation from moderate-sized samples. Empirically, across all experimental configurations in Section C.3, the difference between $R$ and $R^*$ is consistently small. Tables 7 and 8 show that the PAD values under Monte-Carlo sampling closely match those under full enumeration, with deviations typically within $3\%$, confirming that the approximation introduces no significant performance degradation. Furthermore, ACC and ASR remain virtually unchanged, and the overall trends (e.g., decreasing PAD with increasing Top-$K$ and $n$) are fully preserved under sampling.

Combining the strict lower bound with the empirically validated tightness, Monte-Carlo sampling with $m \approx 200$ yields an effective approximation of the true robustness metric, decoupling the computational complexity from the combinatorial growth of $\binom{K}{n}$ and reducing it to a constant factor determined by the sample size $m$.

### A.4 Guarantee of Clean Majority in the Certified Ball

**Lemma 1.** Let $\mathcal{V}$ be the set of all $L = \binom{K}{n}$ subset embeddings derived from the retrieved documents. Let $\mathcal{C} \subset \mathcal{V}$ denote the set of clean embeddings and $\mathcal{P} \subset \mathcal{V}$ denote the set of poisoned embeddings, such that $|\mathcal{P}| \leq L_{adv} = \binom{K}{n} - \binom{K-\varepsilon}{n}$. If the certified radius $\hat{R}$ is determined by the distance value at the $k$-th index (0-based) of the sorted distance vector to the geometric center, with $k = \lfloor L/2 \rfloor + L_{adv}$, then the ball $\mathcal{B}(z, \hat{R})$ is guaranteed to strictly contain a majority of clean embeddings, i.e., $|\mathcal{B}(z, \hat{R}) \cap \mathcal{C}| \geq \lfloor L/2 \rfloor + 1$.

***Proof.*** Let $\mathcal{S}$ denote the set of embeddings enclosed by the ball $\mathcal{B}(z, \hat{R})$. Since the radius $\hat{R}$ corresponds to the distance at the $k$-th index (where index 0 represents the center itself), the set $\mathcal{S}$ includes the first $k+1$ smallest distance embeddings. Thus, by definition, $|\mathcal{S}| = k + 1$. The set $\mathcal{S}$ is composed of disjoint clean and poisoned subsets: $\mathcal{S} = (\mathcal{S} \cap \mathcal{C}) \cup (\mathcal{S} \cap \mathcal{P})$. We seek a lower bound on the number of clean samples $|\mathcal{S} \cap \mathcal{C}|$. According to

set theory:

$$|\mathcal{S} \cap \mathcal{C}| = |\mathcal{S}| - |\mathcal{S} \cap \mathcal{P}| \tag{17}$$

In the worst-case adversarial scenario, the adversary optimizes the poisoned embeddings to be geometrically closest to the center to occupy the top ranks. However, the total number of poisoned embeddings is strictly bounded by $L_{adv}$. Thus, for any selected subset, $|\mathcal{S} \cap \mathcal{P}| \leq L_{adv}$ always holds. Substituting the cardinality $|\mathcal{S}| = k + 1$ and the value of $k$:

$$\begin{aligned} |\mathcal{S} \cap \mathcal{C}| &\geq (k+1) - L_{adv} \\ &= (\lfloor L/2 \rfloor + L_{adv} + 1) - L_{adv} \\ &= \lfloor L/2 \rfloor + 1 \end{aligned} \tag{18}$$

Therefore, the ball $\mathcal{B}(z, \hat{R})$ necessarily contains at least $\lfloor L/2 \rfloor + 1$ clean embeddings. Since $L = |\mathcal{V}|$, this count represents a strict majority of the total universe of combinations.

### A.5 Geometric Separability Assumption

We assume a mild geometric separability condition to connect majority coverage over combination embeddings to clean subset selection. Specifically, there exists a center $z^*$ and a radius $R_c$ such that a strict majority of fully clean combination embeddings lie within the ball $\mathcal{B}(z^*, R_c)$, while any combination containing at least one poisoned passage lies outside $\mathcal{B}(z^*, R_c + \Delta)$ for some $\Delta > 0$.

The separability condition is necessary to ensure that a ball containing a strict majority of clean embeddings is geometrically anchored to the clean subset, thereby enabling the selection of clean contexts through geometric aggregation. When the margin $\Delta = 0$, clean and poisoned combination embeddings become geometrically indistinguishable. In such cases, no geometry-based majority certification method can guarantee the selection of a fully clean subset, as the observed information no longer provides distinguishable features.

## B Details

### B.1 Details of Experiment Setup

**Dataset.** Detailed statistics are shown in Table 4.

**Default Setting.** Unless otherwise specified, our experiments use the following default settings: the Contriever (Izacard et al., 2021) retriever is employed to select the Top-$K = 8$ texts most similar to the query from the MS-MARCO dataset, with poisoned texts generated according to the PoisonedRAG (Zou et al., 2024) method constituting 20% (rounded down) of the retrieval results. Subsequently, multiple subsets are sampled from the retrieved texts, each containing $n = 3$ 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 and GPT-5-mini), 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.

Table 4: Statistics 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 |

**Implementation and Resources.** We conduct experiments on a server with 64 AMD EPYC 9654 CPUs at 2.40–3.70 GHz, 512 GB of DDR5 RAM, and four NVIDIA RTX A100 GPUs, each with 80 GB HBM2e 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:
```

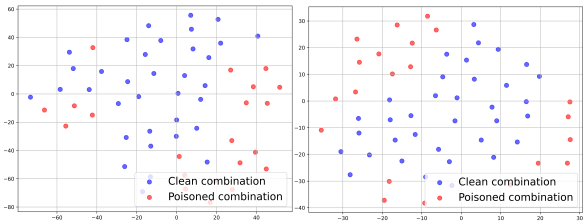
Figure 3: The prompt used in RAG to make an LLM generate an answer based on the retrieved texts.

```
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., 2025c), Denial-Of-Service Attack (Shafran et al., 2024), and CorruptRAG (Zhang et al., 2025a). 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., 2025c) 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. CorruptRAG (Zhang et al., 2025a) can efficiently mislead the model into generating targeted false content by injecting only a single poisoned text that simultaneously includes the target query, which ensures it is preferentially retrieved, and an adversarial prompting template, which exploits the LLM's generation bias to label the correct answer as outdated and to fabricate a supposedly latest but incorrect answer. In our experiments, we adopt the default parameters of these methods to generate the corresponding poisoned texts, ensuring effective poisoning.




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

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

Table 5: 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 6: 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 |

## 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.

### C.2 Analysis of Computational Overhead

Table 5 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 6 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

Table 7: Performance of PRA-RAG using Monte Carlo sampling under varying Top-$K$ (subset size = 3, poisoning rate = 10%).

| Models | Top-$K$ | NQ | | | MS-MARCO | | | HotpotQA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | ASR | PAD | ACC | ASR | PAD | ACC | ASR | PAD |
| Mistral-7B | 8 | 0.70 | 0.00 | 1.28 | 0.62 | 0.00 | 1.31 | 0.34 | 0.12 | 1.32 |
| | 12 | 0.52 | 0.00 | 1.16 | 0.66 | 0.00 | 1.21 | 0.44 | 0.02 | 1.18 |
| | 16 | 0.52 | 0.00 | 1.12 | 0.68 | 0.00 | 1.20 | 0.40 | 0.02 | 1.15 |
| Vicuna-7B | 8 | 0.44 | 0.00 | 1.78 | 0.66 | 0.02 | 1.84 | 0.42 | 0.04 | 1.86 |
| | 12 | 0.51 | 0.00 | 1.65 | 0.76 | 0.00 | 1.72 | 0.52 | 0.01 | 1.74 |
| | 16 | 0.42 | 0.00 | 1.64 | 0.70 | 0.00 | 1.70 | 0.48 | 0.00 | 1.67 |

Table 8: Performance of PRA-RAG using Monte Carlo sampling with Top-$K = 12$ retrieval under a poisoning rate of 10%, evaluated across different subset sizes.

| Models | Subset Size | NQ | | | MS-MARCO | | | HotpotQA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | ASR | PAD | ACC | ASR | PAD | ACC | ASR | PAD |
| Mistral-7B | 3 | 0.52 | 0.00 | 1.16 | 0.66 | 0.00 | 1.21 | 0.44 | 0.02 | 1.18 |
| | 4 | 0.56 | 0.00 | 1.02 | 0.66 | 0.00 | 1.07 | 0.44 | 0.02 | 1.07 |
| | 5 | 0.72 | 0.00 | 0.90 | 0.82 | 0.00 | 0.97 | 0.46 | 0.01 | 0.96 |
| Vicuna-7B | 3 | 0.51 | 0.00 | 1.65 | 0.76 | 0.00 | 1.72 | 0.52 | 0.01 | 1.74 |
| | 4 | 0.48 | 0.00 | 1.44 | 0.66 | 0.02 | 1.51 | 0.47 | 0.02 | 1.55 |
| | 5 | 0.47 | 0.00 | 1.30 | 0.60 | 0.00 | 1.37 | 0.45 | 0.02 | 1.36 |

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.

### C.3 Evaluation of the Impact of Monte Carlo Sampling

As the number of retrieved documents (Top-$K$) and the subset size ($n$) increase, the number of candidate combinations grows rapidly, incurring significant computational overhead. To reduce costs and enhance the practicality of our method, we introduce Monte Carlo sampling when the combination size reaches a certain level. Based on the analysis in Section A.3, we set the sampling count to $m = 200$, which also serves as the threshold for triggering the sampling process. Specifically, when the number of candidate combinations does not exceed $m$, we employ the original full enumeration strategy; when the number exceeds $m$, we utilize Monte Carlo sampling to approximate the combinatorial space. This strategy effectively controls computational overhead while minimizing the impact on performance.

Tables 7 and 8 present the performance of our method using Monte Carlo sampling approximation across various settings of retrieved document counts (Top-$K$) and subset sizes ($n$), specifically when the number of candidate combinations exceeds the threshold ($m = 200$). By comparing these results with those obtained via full enumeration in Figure 2, it is evident that the introduction of Monte Carlo sampling causes no significant degradation in performance. Furthermore, the overall trends remain consistent, indicating that the impact of poisoned texts is further attenuated as Top-$K$ and $n$ increase, while PAD exhibits a continuous downward trend. To further strengthen this finding, we additionally conduct experiments with $K = 20$ and subset sizes $n \in 3, 4, 5$ under sampling counts $m \in 200, 400, 600$. The results, reported in Table 9, further confirm that the approximation quality remains stable as $K$ and $n$ scale.

Table 9: Performance of PRA-RAG using Monte Carlo sampling with Top-$K = 20$ retrieval under a poisoning rate of 20%, evaluated across different sampling numbers and subset sizes under Mistral-7B.

| Sampling Number | Subset Size | NQ | | | MS-MARCO | | | HotpotQA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | ASR | PAD | ACC | ASR | PAD | ACC | ASR | PAD |
| 200 | 3 | 0.57 | 0.01 | 1.08 | 0.74 | 0.00 | 1.21 | 0.48 | 0.02 | 1.15 |
| | 4 | 0.64 | 0.00 | 0.94 | 0.70 | 0.00 | 1.07 | 0.56 | 0.02 | 1.00 |
| | 5 | 0.69 | 0.00 | 0.83 | 0.76 | 0.00 | 0.96 | 0.53 | 0.02 | 0.90 |
| 400 | 3 | 0.56 | 0.00 | 1.07 | 0.69 | 0.00 | 1.21 | 0.51 | 0.01 | 1.14 |
| | 4 | 0.67 | 0.00 | 0.93 | 0.70 | 0.01 | 1.06 | 0.54 | 0.03 | 0.99 |
| | 5 | 0.68 | 0.01 | 0.83 | 0.79 | 0.00 | 0.96 | 0.58 | 0.01 | 0.90 |
| 600 | 3 | 0.57 | 0.01 | 1.07 | 0.67 | 0.00 | 1.21 | 0.48 | 0.01 | 1.14 |
| | 4 | 0.62 | 0.00 | 0.93 | 0.74 | 0.00 | 1.05 | 0.52 | 0.02 | 0.99 |
| | 5 | 0.70 | 0.00 | 0.83 | 0.73 | 0.01 | 0.95 | 0.54 | 0.02 | 0.90 |

Table 10: Computational overhead of PRA-RAG using Monte Carlo sampling under varying Top-$K$ (subset size $n = 3$, poisoning rate = 20%).

| Models | Top-$K$ | NQ | MS-MARCO | HotpotQA |
|---|---|---|---|---|
| Mistral-7B | 8 | 5.98s | 6.20s | 6.17s |
| | 12 | 11.78s | 11.71s | 11.97s |
| | 16 | 11.09s | 11.69s | 12.50s |
| Vicuna-7B | 8 | 3.37s | 2.91s | 3.02s |
| | 12 | 6.67s | 7.55s | 6.42s |
| | 16 | 9.23s | 7.84s | 6.46s |

Tables 10 and 11 present the computational overhead of the algorithm with Monte Carlo sampling under different Top-$K$ and $n$ settings. In Table 10, with Top-$K = 8$ and $n = 3$, the number of candidate combinations is 56, which remains below the threshold ($m = 200$), implying that sampling is not triggered and the computational cost remains consistent with the results in Table 5. In other scenarios, compared to the full enumeration results in Tables 5 and 6, the sampling strategy significantly

Table 11: Computational overhead of PRA-RAG using Monte Carlo sampling with Top-$K = 12$ under a poisoning rate of 10%.

| Model | Subset Size | NQ | MS-MARCO | HotpotQA |
|---|---|---|---|---|
| Mistral-7B | 3 | 12.26s | 10.51s | 11.93s |
| | 4 | 11.96s | 15.07s | 12.16s |
| | 5 | 11.95s | 15.09s | 12.17s |
| Vicuna-7B | 3 | 6.87s | 9.03s | 6.79s |
| | 4 | 7.03s | 7.70s | 6.70s |
| | 5 | 7.17s | 8.09s | 7.28s |

reduces computational overhead. For instance, on the Vicuna-7B model using the NQ dataset, the cost drops from 64.93s to 9.23s when Top-$K = 16$ and $n = 3$, and from 116.09s to 7.17s when Top-$K = 12$ and $n = 5$.