Rescuing the Unpoisoned: Efficient Defense against Knowledge Corruption Attacks on RAG Systems

Minseok Kim
Sungkyunkwan University
Suwon, South Korea
for8821@g.skku.edu

Hankook Lee
Sungkyunkwan University
Suwon, South Korea
hankook.lee@skku.edu

Hyungjoon Koo*
Sungkyunkwan University
Suwon, South Korea
kevin.koo@skku.edu

Abstract*—Large language models (LLMs) are reshaping numerous facets of our daily lives, leading to their widespread adoption as web-based services. Despite their versatility, LLMs face notable challenges, such as generating hallucinated content and lacking access to up-to-date information. Lately, to address such limitations, Retrieval-Augmented Generation (RAG) has emerged as a promising direction by generating responses grounded in external knowledge sources. A typical RAG system consists of i) a retriever that probes a group of relevant passages from a knowledge base and ii) a generator that formulates a response based on the retrieved content. However, as with other AI systems, recent studies demonstrate the vulnerability of RAG, such as knowledge corruption attacks by injecting misleading information. In response, several defense strategies have been proposed, including having LLMs inspect the retrieved passages individually or fine-tuning robust retrievers. While effective, such approaches often come with substantial computational costs. In this work, we introduce RAGDEFENDER, a resource-efficient defense mechanism against knowledge corruption (*i.e.,* by data poisoning) attacks in practical RAG deployments. RAGDEFENDER operates during the post-retrieval phase, leveraging lightweight machine learning techniques to detect and filter out adversarial content without requiring additional model training or inference. Our empirical evaluations show that RAGDEFENDER consistently outperforms existing state-of-the-art defenses across multiple models and adversarial scenarios: *e.g.,* RAGDEFENDER reduces the attack success rate (ASR) against the Gemini model from 0.89 to as low as 0.02, compared to 0.69 for RobustRAG and 0.24 for Discern-and-Answer when adversarial passages outnumber legitimate ones by a factor of four (4x).

1. Introduction

Today, large language models (LLMs) like GPT-4 [1] have been quickly transforming almost every aspect of our daily lives, ranging from education (*e.g.,* tutoring assistant) [2], business (*e.g.,* customer support) [3], creative tasks (*e.g.,* design) [4], entertainment (*e.g.,* game) [5] to information technology such as code generation [6], [7], information retrieval [8], [9], and recommender systems [10], [11]. Given the current trajectory and significant impacts of LLMs, a wide adoption of LLMs in the near future is evident.

While these models demonstrate impressive capabilities across a broad spectrum of tasks, LLMs suffer from a few limitations: hallucination [12], staleness [13], lack of information in a specific domain [14], and high training costs [15]. Such restrictions may lead to generating inaccurate information, struggling with up-to-date knowledge, falling short in a specialized field, or requiring significant computational resources for (re-)training and deployment.

Retrieval-Augmented Generation (RAG) [16] has been introduced as a promising direction to relax such constraints in a language model architecture. By design, RAG leverages the benefits of both pre-trained parametric knowledge (*i.e.,* encoded in model weights) and non-parametric memory (*i.e.,* external knowledge bases) to address the above limitations, which consists of a retriever and a generator. In essence, the retriever probes a set of pertaining passages from a database (*i.e.,* external knowledge) when an input query (*e.g.,* prompt) is given, followed by generating a response based on the retrieved information by the generator.

Recent studies uncover several attack surfaces that can threaten RAG systems, broadly grouping them into three types: ① data poisoning attacks [17], [18], [19] that corrupt the underlying knowledge base by injecting conflicting or misleading content (*e.g.,* inserting fake news in Wikipedia); ② retrieval poisoning attacks [20], [21], [22] that manipulate the retrieval process to surface adversarial passages (*e.g.,* activating a hidden backdoor trigger); and ③ prompt manipulation attacks [23] that modify input prompts to steer the language model toward generating inaccurate responses. These attack vectors severely thwart the transparency and reliability of RAG systems, necessitating a robust and efficient defense mechanism. We focus on *mitigating data poisoning attacks* (*i.e.,* knowledge corruption), which pose a serious threat due to their low barrier to entry: *e.g.,* disseminating disinformation [19] into public sources could be more pervasive and impactful in real-world scenarios.

Recent efforts to mitigate knowledge corruption attacks against RAG systems introduce several defense mechanisms. The ROBUSTRAG [24] tackles data poisoning by requiring LLMs to inspect each retrieved passage individually before

* *Corresponding author.*

2576-9103/25/$31.00 ©2025 IEEE
DOI 10.1109/ACSAC67867.2025.00093

generating responses. Discern-and-Answer [25] attempts to identify conflicting information among retrieved documents by fine-tuning a discriminator or prompting the LLM to identify inconsistencies. While both approaches enhance the robustness and reliability of RAG systems, they suffer from additional computational overhead, such as multiple LLM inferences or substantial memory consumption. Besides, these approaches may introduce potential inefficiencies in scenarios where legitimate passages significantly outnumber adversarial ones, leading to unnecessary resource usage.

In this paper, we propose RAGDEFENDER, *a resource-efficient defense mechanism against the knowledge corruption attacks on RAG systems* by filtering out potentially adversarial documents. In a nutshell, the phase of RAGDEFENDER is twofold: ① grouping the retrieved passages and ② identifying adversarial passages. Particularly, we devise two novel strategies (*i.e.,* clustering-based and concentration-based grouping) to rescue legitimate (*i.e.,* unpoisoned) passages at the first stage. By design, unlike previous approaches, RAGDEFENDER requires neither additional LLM inferences nor considerable memory overheads in practice. Besides, RAGDEFENDER is agnostic to RAG architectures or components, being applicable in handy.

Our empirical results demonstrate that RAGDEFENDER significantly outperforms existing defense approaches across various adversaries' tactics, models, datasets, and adversarial content ratios. RAGDEFENDER consistently achieves a lower attack success rate (ASR) than state-of-the-art models and baseline approaches. For instance, on the Natural Questions (NQ) [26] dataset with a $4\times$ perturbation ratio (*i.e.,* the number of adversarial passages is four times more than benign ones), RAGDEFENDER reduces the ASR from 0.89 to 0.02 for Gemini [27], compared to 0.24 for Discern-and-Answer [25] and 0.69 for RobustRAG [24]. Notably, our evaluation with three different retrieval models (Contriever [8], DPR [28], ANCE [29]) shows the lowest ASR of 0.04 against Gemini on the MS MARCO [30] dataset. Besides, RAGDEFENDER achieves a noticeable speed over RobustRAG [24] ($12.36\times$ faster) and Discern-and-Answer [25] ($1.53\times$ faster).

The following summarizes our contributions:

- We propose RAGDEFENDER, an efficient RAG defense system that can filter out adversarial passages. We leverage lightweight machine learning-based detection to enhance the robustness of RAG systems against knowledge corruption attacks.
- We design and implement the RAGDEFENDER prototype, which can be seamlessly integrated into a wide range of RAG systems (*e.g.,* architecture, inner components).
- Our thorough evaluation shows RAGDEFENDER outperforms state-of-the-art models, demonstrating its effectiveness, efficiency, robustness, and adaptability in practice.

We plan to open-source RAGDEFENDER[1] to foster further research for building a secure RAG system.

1. https://github.com/SecAI-Lab/RAGDefender

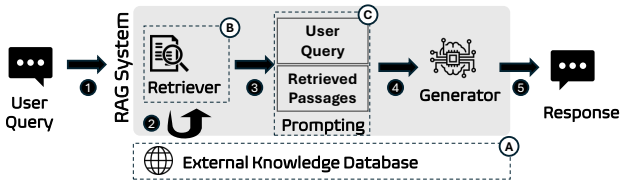

Figure 1: Overview of a RAG system (gray area) and potential attack surfaces. ❶ With a user's query, ❷ a retriever probes documents from an external knowledge base, returning a set of relevant passages. Then, ❸ the retriever forms a prompt with those passages, and ❹ a generator generates ❺ a proper response based on the prompt. The dotted boxes indicate potential attack surfaces: Ⓐ data poisoning by compromising a database or external resource, Ⓑ retrieval poisoning by exploiting the retriever to trigger a backdoor, and Ⓒ prompt manipulation by altering input prompts to mislead the generator. This work focuses on knowledge corruption attacks (Ⓐ), safeguarding a post-retrieval set (❹).

2. Background

RAG Overview. RAG [16] has been introduced to tackle the limitations of pre-trained language models (*e.g.,* BERT [31] and GPT [32]) such as producing factually incorrect responses [12] or being constrained by learned knowledge [13]. Unlike such static LLMs, RAG allows for ① reducing hallucinations and ② offering up-to-date information in a timely manner by retrieving information from external databases (*i.e.,* sources). Simply put, RAG has two components: a *retriever* retrieves relevant passages with a given query from a knowledge database, and then a *generator* utilizes the prompt *augmenting* the query with the retrieved passages to produce a factual response grounded in the retrieved external context.

RAG Retriever. As depicted in Figure 1, when a query is given (❶), a retriever chooses relevant passages from an external knowledge database, determining the most pertinent passages via a scoring and ranking process (❷). Retrieval methods fall into two main approaches: sparse retrieval and dense retrieval. Sparse retrieval techniques rely on exact term matching and term weighting, such as Term Frequency-Inverse Document Frequency (TF-IDF) [33] and BM25 [34]. While computationally efficient, sparse retrieval may fail to catch semantic nuances. Meanwhile, dense retrieval techniques leverage neural embeddings to represent queries and passages in a high-dimensional vector space for better capturing semantic similarity. Examples of dense retrieval include Contriever [8], Approximate nearest neighbor Negative Contrastive Estimation (ANCE) [29], and DPR [28]. We incorporate both sparse and dense retrieval strategies to provide a robust defense.

RAG Generator. A generator is typically a language model (*e.g.,* GPT-4 [1]), which synthesizes a final response based on the user query and the retrieved passages, guided by an input prompt. Figure 1 illustrates that the retrieved passages are integrated into a prompt (❸) alongside the query, provid-

TABLE 1: Comparison of state-of-the-art defenses for a RAG system: RAGDEFENDER (ours), ROBUSTRAG [24], and Discern-and-Answer [25]. Prior approaches inevitably incur overheads due to additional inference or fine-tuning.

Feature	RAGDEFENDER	ROBUSTRAG	Discern-and-Answer
Fine-tuning Overhead	No	No	Yes
LLM inference Overhead	No	Yes	Yes
Computational Overhead	Low	High	Medium
Adaptability	Yes	Yes	No

ing the generator with contextually rich information. Finally, the generator takes the prompt as input (❹) to produce and return reliable responses (❺).

Attack Surfaces on RAG. Recently, several adversarial attacks have been identified against RAG systems. As illustrated in Figure 1, these known attacks can be broadly fallen into three types: database poisoning attacks (Ⓐ) that inject corrupted information into the database, retrieval poisoning attacks (Ⓑ) that manipulate the retriever to trigger a malicious content (*e.g.,* backdoor), and prompt manipulation attacks (Ⓒ) that alter input prompts to mislead the generator. Note that each attack vector aims to compromise the integrity and reliability of a RAG's output, potentially leading to generating inaccurate, biased, or harmful information.

Defenses against RAG Attacks. To mitigate the known attacks above, several defense strategies have been proposed. Table 1 concisely summarizes the comparison between ours (RAGDEFENDER) and prior approaches. Xiang *et al.* [24] propose a means that isolates retrieved passages and verifies them before generating a final response. Another line of research [25] fine-tunes a discriminator to handle conflicting knowledge among retrieved documents.

3. Threat Model

Preliminary Definitions. Given an input query q, a retriever R retrieves a set of passages $\mathcal{R}$, relevant to the query from a knowledge database $\mathcal{D}$. The retrieval process can be formally written as $\mathcal{R} = \{r_1, r_2, \ldots, r_k\} \leftarrow R(q, \mathcal{D})$ where $r_i \in \mathcal{R} \subset \mathcal{D}$ is a query-relevant passage retrieved from the database. A RAG system constructs a prompt by combining the query q and the relevant passages $\mathcal{R}$. Finally, a generator G generates a reliable response y based on the augmented prompt, which can be written as $y \leftarrow G(q, \mathcal{R})$. Table 10 in Appendix summarizes all notations.

Threat Model. Suppose an attacker aims to compromise the integrity of RAG. We assume the adversary first selects one or more target questions and then specifies an arbitrary target answer for each. This can be carried out by injecting M adversarial passages, $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_M$, into the database $\mathcal{D}$. We define an *adversarial passage* as any passage that contains (deliberately) misleading information crafted to induce an attacker-desired answer by manipulating RAG's responses. Such targeted poisoning can contaminate the retrieval and generation processes of RAG as $\tilde{\mathcal{R}} \leftarrow R(q, \mathcal{D} \cup \{\tilde{p}_i\}_{i=1}^{M})$ and $\tilde{y} \leftarrow G(q, \tilde{\mathcal{R}})$. For a successful attack, the adversarial passages should be retrieved and corrupt the response; in other words, each adversarial passage $\tilde{p}_i$ should be semantically relevant to the query while containing corrupted information. To consider more challenging attack scenarios, we further assume that all the adversarial passages are prioritized over benign ones during retrieval; *i.e.,* $\tilde{p}_i \in \tilde{\mathcal{R}}$ for all i, while ensuring that at least one benign passage is retrieved; *i.e.,* $\tilde{\mathcal{R}} \setminus \{\tilde{p}_i\}_{i=1}^{M} \neq \emptyset$.

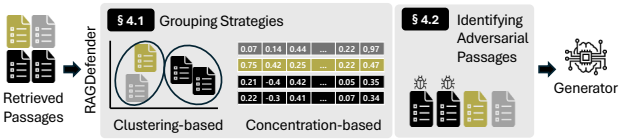

Figure 2: Overview of RAGDEFENDER. To defend against knowledge poisoning attacks, RAGDEFENDER first classifies retrieved passages into benign and potentially-adversarial groups (§4.1), and then identifies adversarial passages (§4.2). Finally, the generator creates a response with the filtered (*i.e.,* legitimate) passages.

Defender's Goal. A defender aims to ensure that the generator G produces reliable responses even when malicious information is injected into the database by an attacker. For broad applicability, we assume that the defender is *unaware of the presence of the adversarial passages* in the database. Formally, the goal of the defender can be written as $G(q, \mathcal{R}) \simeq G(q, \text{RAGDEFENDER}(q, \tilde{\mathcal{R}}))$.

4. RAGDEFENDER System

Design Goals. We design RAGDEFENDER to secure a RAG system against knowledge poisoning attacks with the following three goals in mind: ① effectiveness that ensures a proposed defense mechanism, including various poisoning tactics and datasets (§6.2); ② computational efficiency that a defense-equipped RAG system incurs acceptable overhead (*e.g.,* processing time) in practice (§6.3); ③ adaptability that can be seamlessly integrated into various RAG architectures and components such as retrievers and generators (§6.4); and ④ robustness against the attackers with advanced tactics including adaptive evasion and integrity violations.

Overview. RAGDEFENDER aims to identify and filter out adversarial passages $\mathcal{R}_{\text{adv}}$ from the retrieved set $\tilde{\mathcal{R}}$, ensuring that only the safe passages $\mathcal{R}_{\text{safe}} = \tilde{\mathcal{R}} \setminus \mathcal{R}_{\text{adv}}$ are provided to the generator G. As illustrated in Figure 2, we present a two-stage procedure: the first stage estimates the number of adversarial passages, denoted as N_{adv}, and the second stage identifies N_{adv} such passages within the retrieved set. In a nutshell, the first stage (§4.1) estimates N_{adv} by grouping passages based on hierarchical clustering with term frequency (*i.e.,* TF-IDF [33]) or concentration-based analysis with document embedding vectors. Subsequently, the second stage (§4.2) identifies adversarial passages $\mathcal{R}_{\text{adv}}$ by leveraging the estimated N_{adv} and their isolated nature in the embedding space. The resulting safe passages $\mathcal{R}_{\text{safe}} = \tilde{\mathcal{R}} \setminus \mathcal{R}_{\text{adv}}$ are then passed to the generator G to

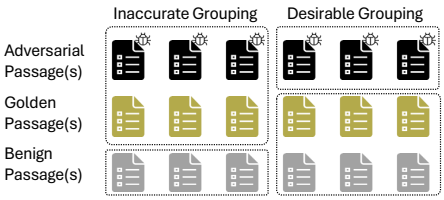

Figure 3: Example of inaccurate passage grouping at the first stage (§4.1) where adversarial passage(s) are combined with golden passage(s). In this mis-partitioning case (left), the second stage (*e.g.,* based on high semantic relationships) assists in separating adversarial passage(s) (§4.2) from benign one(s), yielding desirable grouping (right).

produce a reliable response. The second stage offers a more refined identification guided by the estimated number of adversarial passages. This two-stage design is crucial for handling cases where the initial grouping in the first stage might be inaccurate (Figure 3). We provide a motivating example in Appendix A.

4.1. Grouping Retrieved Passages

In this stage, RAGDEFENDER estimates potential adversarial passages in the retrieved set with a grouping strategy. In essence, our strategy relies on the distribution of similarities between q and r_i, $\text{sim}(q, r_i)$. For instance, a group of adversarial passages would be similar for successful attacks. Hence, we devise two means to carry out precise grouping: ① clustering-based grouping for a single-hop QA to organize semantically similar passages into dense clusters and ② concentration-based grouping for a multi-hop QA to recognize the concentration factors among the retrieved passages[2].

Clustering-based Grouping. The clustering-based grouping strategy attempts to provide a preliminary estimation of potential adversarial passages within the retrieved set. Because adversarial passages tend to form dense clusters [19] for successful knowledge corruption (Section 7), we adopt a hierarchical agglomerative clustering [35] to partition the embeddings of retrieved passages. We choose agglomerative clustering because it can handle clusters that are non-spherical and varying in size, identifying complex structures such as dense groups of adversarial passages. In contrast, methods like K-Means [36] struggle with such scenarios due to their assumption of spherical cluster shapes. Notably, we incorporate one of the sparse retrieval methods, TF-IDF [33] for reliable group separation (Section 7). That is, with TF (term frequency within a passage) and IDF (inverse document frequency across retrieved passages), we identify a set of the m most frequent terms, weighted by TF-IDF. Then, we calculate scores across $\tilde{\mathcal{R}}$ and select the $T_{\text{top}} = \{t_1, \ldots, t_m\}$ terms with the highest scores (*i.e.,* sorting by word importance). Namely, the passages containing a high proportion of T_{top} terms are considered adversarial, due to their shared keywords. We then compute $N_{\text{TF-IDF}}$, the number of passages that contain more than half of these top terms, as follows:

$$N_{\text{TF-IDF}} = \sum_{i=1}^{|\tilde{\mathcal{R}}|} I\left(\sum_{j=1}^{m} I(t_j \in r_i) > \frac{m}{2}\right) \tag{1}$$

where $I(\cdot)$ is the indicator function that returns 1 if the condition is true and 0 otherwise. Finally, we estimate the number of *potentially adversarial passages* as follows:

$$N_{\text{adv}} = \begin{cases} n_{\text{min}} & \text{if } N_{\text{TF-IDF}} \leq \frac{|\tilde{\mathcal{R}}|}{2} \\ |\tilde{\mathcal{R}}| - n_{\text{min}} & \text{otherwise} \end{cases} \tag{2}$$

where n_{min} denotes the size of the smaller subset between the two clusters. Selecting the smaller or larger cluster size based on the TF-IDF analysis enhances the reliability of estimation. If fewer than half of the passages contain the key terms, we assume that the adversarial passages form the minority cluster (of size n_{min}); otherwise, they constitute the majority. This approach is flexible for scenarios where adversarial passages may belong to either group (*i.e.,* smaller or larger subset) of a retrieved set.

Concentration-based Grouping. The concentration-based grouping strategy aims to provide an initial estimation of potential adversarial passages in the case of a multi-hop question task While adversarial passages are derived from a variety of golden passages (*i.e.,* ground truth) and therefore exhibit some degree of diversity, they tend to embed misleading information. This concentration allows the grouping strategy to effectively distinguish adversarial passages from the more dispersed legitimate ones. To estimate the concentrativeness, first, we define two concentration factors for each passage r_i as $s_i^{\text{mean}} = \frac{1}{|\tilde{\mathcal{R}}|-1} \sum_{j \neq i} \text{sim}(r_i, r_j)$ and $s_i^{\text{median}} = \text{median}\left(\{\text{sim}(r_i, r_j)\}_{j \neq i}\right)$. Intuitively, adversarial passages tend to exhibit high concentration factors which can *disproportionately inflate the global mean and median* – an effect that would be minimal in their absence. Namely, a passage is counted toward the adversarial estimate if both its concentration factors exceed their respective global counterparts across the retrieved set. Consequently, the number of potentially adversarial passages can be estimated as:

$$N_{\text{adv}} = \sum_{i=1}^{|\tilde{\mathcal{R}}|} I(s_i^{\text{mean}} > \bar{s}) \cdot I(s_i^{\text{median}} > \tilde{s}) \tag{3}$$

where $\bar{s} = \frac{1}{|\tilde{\mathcal{R}}|} \sum_{i=1}^{|\tilde{\mathcal{R}}|} s_i^{\text{mean}}$ and $\tilde{s} = \text{median}\left(\{s_i^{\text{median}}\}_{i=1}^{|\tilde{\mathcal{R}}|}\right)$ are the mean and the median of the concentration factors, respectively.

4.2. Identifying Adversarial Passages

Given the estimate of N_{adv}, we next rank individual passages to identify those most likely to be adversarial. We achieve this by ranking passages based on their semantic similarity with other passages and by selecting the best

2. Single-hop QA finds an answer within a single passage, whereas multi-hop QA requires integrating information across multiple contexts.

candidates as adversarial. First, we compute the cosine similarity between all pairs of passages and identify the top N_{pairs} most similar pairs as follows:

$$N_{\text{pairs}} = \max\left(1, \binom{N_{\text{adv}}}{2}\right) \tag{4}$$

Let $\mathcal{P}_{\text{top}}$ denote the set of these top N_{pairs} pairs :

$$\mathcal{P}_{\text{top}} = \text{TopK}\left(\left\{(r_i, r_j) \in \tilde{\mathcal{R}} \times \tilde{\mathcal{R}} \mid i \neq j\right\},\ N_{\text{pairs}},\ \text{sim}(r_i, r_j)\right) \tag{5}$$

where $\text{TopK}(\mathcal{S}, K, \text{score})$ is a function that selects the top K largest elements from the set $\mathcal{S}$ based on their scores. For each passage r_i, we calculate a frequency score f_i, representing how often it appears in $\mathcal{P}_{\text{top}}$ and its similarity with the paired passages:

$$f_i = \sum_{(r_i, r_j) \in \mathcal{P}_{\text{top}}} \text{sgn}(\text{sim}(r_i, r_j)) \cdot |\text{sim}(r_i, r_j)^p| \tag{6}$$

where p and $\text{sgn}(\cdot)$ denote a weighting exponent (empirically $p = 2$ in our experiments; interested readers refer to Section 6.6) and the sign function that returns the sign of a value, respectively. As a similarity score ranges from -1 to 1, the sign function maintains the original sign. We then rank the passages based on f_i and select the top N_{adv} passages as adversarial, *i.e.,*

$$\mathcal{R}_{\text{adv}} = \text{TopK}\left(\{r_i \mid r_i \in \tilde{\mathcal{R}}\},\ N_{\text{adv}},\ f_i\right) \tag{7}$$

Lastly, the remaining passages, $\mathcal{R}_{\text{safe}} = \tilde{\mathcal{R}} \backslash \mathcal{R}_{\text{adv}}$, are considered safe. Focusing on passages that frequently form high-similarity pairs, RAGDEFENDER can effectively isolate the adversarial passages.

4.3. Design Choice

TF-IDF. Effective data poisoning entails injecting a specific keyword into adversarial passages for successful attacks [19]. TF-IDF [33] comes into play, which determines the importance of a word based on frequencies. TF-IDF exploits this property, capturing adversarial passages over benign ones. We leverage TF-IDF to reveal the cluster containing a potential adversarial passage. Although it is possible to use other means, like dense retrieval, it inevitably increases an overhead with a marginal performance increase. We choose TF-IDF because RAGDEFENDER aims to be an efficient but lightweight defense system.

Grouping Strategies. Our grouping strategy is aligned with the intrinsic characteristics of the QA task. For a single-hop QA, we adopt clustering-based grouping, which is effective in identifying adversarial passages that form dense semantic clusters, which is a common trait in knowledge corruption attacks targeting specific facts. In contrast, for a multi-hop QA, we employ a concentration-based grouping strategy. Clustering methods may fail in this setting, as legitimate passages tend to be semantically diverse due to the need for reasoning across multiple sources. Concentration-based grouping instead isolates adversarial passages that embed highly concentrated misleading information, effectively separating them from the more dispersed, contextually rich legitimate passages required for multi-hop reasoning.

5. Implementation

For RAGDEFENDER, we use Sentence Transformers [37] with the Stella [38]'s embedding model to generate document vectors. We store them in a FAISS [39] database, using the IndexFlatL2 index, for efficient retrieval. For the agglomerative clustering, we adopt scikit-learn [40]'s implementation. We utilize the Hugging Face Transformers [41] for passage generation, with two families of open-source language models, including LLaMA-2 [42] and Vicuna-1.3 [43] (both have 7B and 13B parameters), as well as the commercial language models, such as GPT-4o [1] and Gemini-1.5-pro (*i.e.,* Gemini) [27]. Note that we use commercial language models for adversarial passage generation.

Hyperparameter Selection. We varied the proportion of adversarial content, and the top-k retrieval parameters. For the Natural Questions (NQ) [26] and MS MARCO [30] datasets, each retrieval set consists of a single golden passage per query and multiple tangentially related passages as the baseline (*i.e.,* single-hop QA). In our experiments, we evaluate adversarial passages at multiples of the benign passages; *e.g.,* $1\times$, $2\times$, $4\times$, and $6\times$. Note that we define the perturbation ratio as the number of adversarial passages divided by the total number of golden and benign-but-irrelevant passages. For example, in NQ, where each query is associated with one golden passage and four benign-but-irrelevant passages, injecting 30 distinct adversarial passages yields a $6\times$ perturbation ratio ($30/5$). The top-k retrieval parameter was set to $k = |\tilde{\mathcal{R}}| = 5$. For the HotpotQA [44] dataset, each retrieval set includes two golden passages per query, serving as the multi-hop QA baseline, where adversarial passages have been added at the same ratios. We set the top-k retrieval parameter to be $k = |\tilde{\mathcal{R}}| = 2$. We conducted each experiment using ten distinct random seeds to obtain reliable outcomes. During the adversarial passage grouping phase (§4.1), we manually set the hyperparameter $m = 5$ to extract the five most significant words from each retrieved passage. Moreover, in the adversarial passage identification phase (§4.2), we set the weighting exponent of $p = 2$ to compute the frequency score. It is worth noting that we present an ablation study to determine the optimal hyperparameters in Section 6.6.

6. Evaluation

We conducted experiments on a 64-bit Ubuntu 20.04 system with an Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz, 128GB RAM, and a Quadro RTX 8000 GPU.

6.1. Experimental Setup

Datasets, Language Models, and RAG Architectures. Our evaluation component encompasses diverse datasets and

models to ensure comprehensive and robust analysis. We utilize three datasets representing varying complexities and scales in question-answering tasks: HotpotQA [44] (multi-hop reasoning dataset), Natural Questions (NQ) [26] (large-scale dataset), and MS MARCO [30] (machine reading comprehension dataset). All retrieval operations are conducted over a single shared knowledge base for each dataset (*e.g.,* MS MARCO with 8,841,823 passages), from which relevant content is retrieved for all queries. We implement three state-of-the-art dense retrieval architectures for the retrieval component: Contriever [8], ANCE [29], and DPR [28]. To assess the generalizability of our system, we examine a range of generator models, including open-source variants such as LLaMA-7B and LLaMA-13B [42], Vicuna-7B and Vicuna-13B [43], as well as commercial models like Gemini [27] and GPT-4o [1]. This careful selection of curated datasets and models assists in understanding a nuanced efficacy and robustness of RAGDEFENDER across various architectures, sizes, and task complexities.

Adversarial Passage Generation. We generate adversarial passages using three techniques: PoisonedRAG [19], GARAG [17], and the approach by Tan *et al.* [18]. To mislead the model's responses, PoisonedRAG constructs attacker-defined texts that meet both retrieval and generation constraints, whereas GARAG employs typographical errors without compromising text fluency. Meanwhile, Tan *et al.* injects LLM-generated misleading information.

Evaluation Metrics. Consistent with PoisonedRAG [19], we evaluate model performance using two metrics: accuracy and attack success rate (ASR). We define accuracy as the proportion of responses that include the ground-truth answer, which we regard as correct. In contrast, ASR refers to the proportion of responses that contain the attacker-specified target answer. These metrics provide a comprehensive view of RAGDEFENDER's performance, capturing a model's correctness and resilience.

Research Questions. We assess RAGDEFENDER with the following research questions in terms of effectiveness, efficiency, adaptability, robustness, and different configurations. We also conduct an ablation study across different settings.

- How effective is RAGDEFENDER across different adversaries' poisoning tactics and datasets? (§6.2)?
- How efficient is RAGDEFENDER compared to other state-of-the-art RAG defense (§6.3)?
- How adaptable is RAGDEFENDER across different RAG architectures, retrievers, and generators (§6.4)?
- How robust is RAGDEFENDER against adaptive attackers, such as adaptive evasion, heterogeneous content injection, and integrity violations (§6.5)?
- What is the impact of different clustering algorithms and hyperparameter configurations, and the efficacy of a two-stage approach in RAGDEFENDER (§6.6)?

6.2. Effectiveness of RAGDEFENDER

In this section, we evaluate the performance of four RAG defense systems—a baseline with no defense, ROBUSTRAG [24], Discern-and-Answer [25], and RAGDEFENDER across multiple language models (LLaMA [42], Gemini [27], GPT-4o [1], and Vicuna [43]) and three benchmark datasets (NQ [26], HotpotQA [44], and MS MARCO [30]). Our experiments incorporate three existing data poisoning techniques: PoisonedRAG [19], GARAG [17], and the method proposed by Tan *et al.* [18]. As illustrated in Figure 4, across all datasets, models, and attack types, RAGDEFENDER consistently achieves the lowest ASR and the highest accuracy, demonstrating superior effectiveness against adversarial passages compared to all baselines. For example, on the NQ dataset with a $4\times$ adversarial passage ratio, RAGDEFENDER limits ASR to 0.03 across all three attacks, while ROBUSTRAG yields ASRs of 0.84, 0.55, and 0.54, and Discern-and-Answer results in 0.31, 0.30, and 0.28. RAGDEFENDER achieves 0.73, 0.75, and 0.74 in accuracy, significantly outperforming ROBUSTRAG (0.18, 0.11, 0.18) and Discern-and-Answer (0.25, 0.25, 0.24). The results highlight the efficacy of RAGDEFENDER in mitigating the impact of adversarial content across varying datasets, models, and attack strategies.

TABLE 2: Comparison of computational costs (in USD) and processing speed (in seconds) per iteration between RAGDEFENDER and ROBUSTRAG [24]. We use NQ [26] with a $1\times$ adversarial passage ratio across various LLMs.

Model	RAGDEFENDER		ROBUSTRAG	
	Cost (USD)	Speed (sec)	Cost (USD)	Speed (sec)
LLaMA-7B [42]	< **$0.01**	**0.759**	$1.56	10.842
LLaMA-13B [42]	< **$0.01**	**0.780**	$2.25	23.395
Gemini [27]	< **$0.01**	**0.773**	$41.30	2.297
GPT-4o [1]	< **$0.01**	**0.779**	$59.00	3.819
Vicuna-7B [43]	< **$0.01**	**0.772**	$1.22	7.186
Vicuna-13B [43]	< **$0.01**	**0.782**	$1.79	9.775
Average	< **$0.01**	**0.774**	$17.85	9.552

6.3. Efficiency of RAGDEFENDER

We assess RAGDEFENDER's computational efficiency in terms of cost, processing speed, and GPU memory footprint. In this experiment, we use NQ [26] with Contriever [8], applied across a range of language model architectures. We evaluate performance against adversarial passages generated by PoisonedRAG [19]. For reliability, all experiments are repeated 100 times.

Cost. For open-source models such as LLaMA [42] and Vicuna [43], we estimate GPU power consumption alone with the NVML [46] library to sample power usage every 10 ms. Then, the total power consumption is estimated through interpolation. For commercial models such as GPT-4o [1] and Gemini [27], we estimate the cost by calculating each API consumption over 100 inferences. Table 2 compares the needed cost of RAGDEFENDER with that of ROBUSTRAG [24]. Note that RAGDEFENDER needs operations by CPU alone whereas ROBUSTRAG additionally consumes varying costs depending on a generator (*e.g.,* $1.22 for Vicuna-7B [43] up to $59.00 for GPT-4o). Meanwhile,

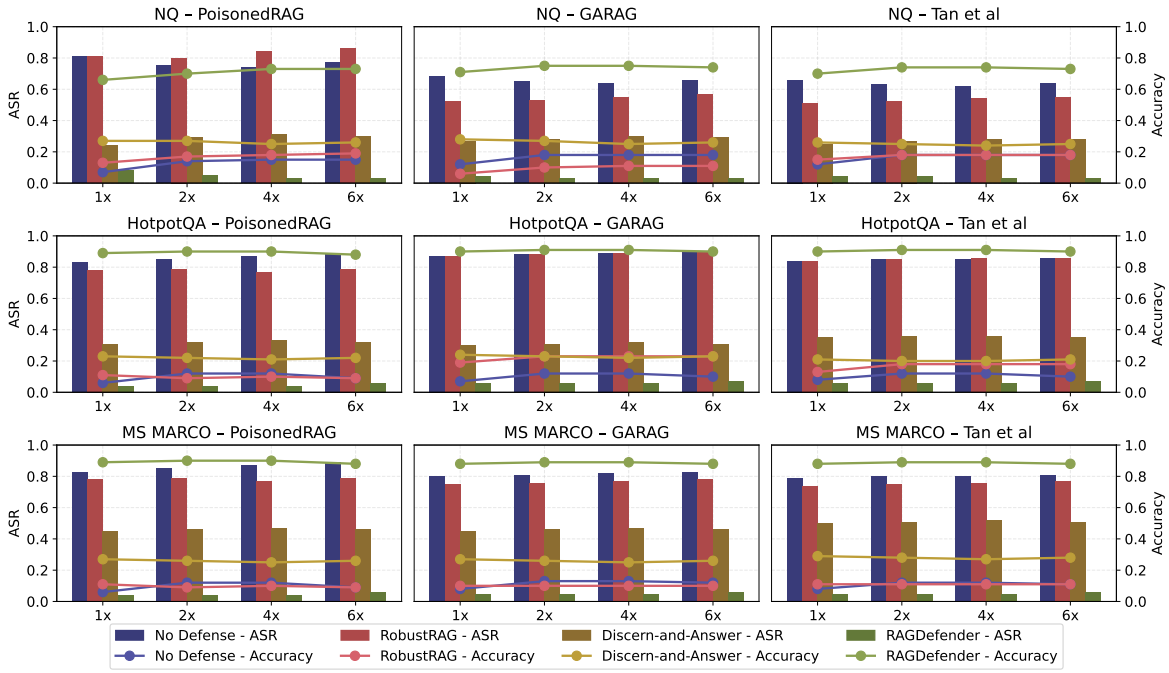

Figure 4: Comparison of attack success rates (ASRs) and accuracy across a baseline (*i.e.,* no defense) [16], RobustRAG [24], Discern-and-Answer [25] and RAGDefender (ours) on different knowledge corruption attacks (PoisonedRAG [19], GARAG [17], Tan *et al.* [18]) under $1\times$, $2\times$, $4\times$, and $6\times$ perturbation ratios. We employ GPT-4o [1] as a generator, while Discern-and-Answer adopts FiD [45]. Each bar and line represents ASR and accuracy with the same scale. The lower ASR and higher accuracy imply better defense. Note that RAGDefender defeats RobustRAG and Discern-and-Answer with high margins in every setting.

TABLE 3: GPU memory footprint comparison (in MB) under a $1\times$ adversarial passage ratio for the NQ [26]. We exclude memory footprints for Gemini [27] and GPT-4o [1] due to the inaccessibility of commercial models. Notably, RAGDefender does not consume any GPU resources as it requires neither fine-tuning nor inference. In contrast, RobustRAG [24] and Discern-and-Answer [25] exhibit significant memory footprints during the fine-tuning and/or inference phases (up to around 41GB in our experiments).

Method	Model	Fine-tuning	Inference
RAGDefender	LLaMA-7B [42] LLaMA-13B [42] Vicuna-7B [43] Vicuna-13B [43]	No GPU Usage	No GPU Usage
RobustRAG	LLaMA-7B [42] LLaMA-13B [42] Vicuna-7B [43] Vicuna-13B [43]	No GPU Usage	18,914MB 33,634MB 14,732MB 27,454MB
Discern-and-Answer	FiD [45]	41,836MB	4,742MB

Discern-and-Answer [25] requires an additional $4.73 for training both its language model and a discriminator because it adopts a dedicated generator (Fusion-in-Decoder [45] or FiD), apart from the inference cost of $0.48 (with an open-source model's cost estimation).

Speed. As evident from Table 2, RAGDefender achieves a reasonable speed of 0.774 seconds per iteration on average, significantly surpassing that of RobustRAG [24] (12.3x faster) using NQ [26] with a $1\times$ adversarial passage ratio. While Discern-and-Answer [25] exhibits moderate overhead with 1.187 seconds per iteration, RobustRAG records the slowest (*e.g.,* 23.395 seconds). Such quick responses indicate the applicability and scalability of RAGDefender.

GPU Memory Footprint. Table 3 demonstrates the efficiency of RAGDefender on the NQ [26] dataset under a $1\times$ adversarial passage ratio. Remarkably, RAGDefender operates seamlessly without any additional GPU memory requirements across all tested models. On the contrary, RobustRAG [24] demonstrates considerable memory footprints depending on the model size, ranging from 14,732 MB to 33,634 MB for Vicuna-7B and LLaMA-13B, respectively. Similarly, the Discern-and-Answer [25] approach consistently consumes 41,836 MB and 4,742 MB during fine-tuning and inference, respectively. Our experiments empirically highlight the RAGDefender's resource-efficient benefits over other defense mechanisms.

6.4. Adaptability of RAGDefender

Performance with Different Architectures. We select three representative state-of-the-art RAG frameworks: BlendedRAG [47], REPLUG [48], and SELF-RAG [49], integrat-

TABLE 4: Comparison of three RAG architectures equipped with RAGDEFENDER. We measure ASR and accuracy across the NQ [26], HotpotQA [44], and MS MARCO [30] datasets. RAGDEFENDER safeguards the architectures effectively against knowledge corruption attacks.

RAG Architecture	Dataset	Method	ASR	Accuracy
BlendedRAG [47]	NQ	No Defense	0.89	0.21
		RAGDEFENDER	**0.62**	**0.37**
	HotpotQA	No Defense	0.94	0.18
		RAGDEFENDER	**0.45**	**0.44**
	MS MARCO	No Defense	0.87	0.26
		RAGDEFENDER	**0.78**	**0.29**
	Average	No Defense	0.90	0.22
		RAGDEFENDER	**0.62**	**0.37**
REPLUG [48]	NQ	No Defense	0.76	0.35
		RAGDEFENDER	**0.11**	**0.66**
	HotpotQA	No Defense	0.81	0.22
		RAGDEFENDER	**0.32**	**0.48**
	MS MARCO	No Defense	0.66	0.43
		RAGDEFENDER	**0.14**	**0.45**
	Average	No Defense	0.74	0.33
		RAGDEFENDER	**0.19**	**0.53**
SELF-RAG [49]	NQ	No Defense	0.73	0.07
		RAGDEFENDER	**0.05**	**0.45**
	HotpotQA	No Defense	0.62	0.33
		RAGDEFENDER	**0.24**	**0.68**
	MS MARCO	No Defense	0.57	0.31
		RAGDEFENDER	**0.10**	**0.68**
	Average	No Defense	0.64	0.24
		RAGDEFENDER	**0.13**	**0.60**

ing RAGDEFENDER into their retrieval processes (architecture selection detailed in Appendix B). We adopt Contriever [8] and LLaMA-7B [42] as a retriever and a generator in all frameworks. Note that we use PoisonedRAG [19] for adversarial passage generation in this setting. Then, we set the adversarial passage ratio of $6\times$ in every environment, whose data poisoning highly misleads the results, ranging from 0.64 to 0.90 of initial ASRs (Table 4). We conduct varying experiments across three datasets: NQ [26], HotpotQA [44], and MS MARCO [30], focusing on two metrics in our evaluation: ASR and accuracy. Table 4 presents noticeable performance enhancement (*i.e.,* substantial reductions in ASR while increasing accuracy) on average across all frameworks. We observe the largest drop in ASR, from 0.73 to 0.05 ($68\% \downarrow$) for SELF-RAG framework, while increasing the accuracy from 0.07 to 0.45 ($38\% \uparrow$).

Performance with Different Retrievers. We adopt three retrievers (Retriever choice detailed in Appendix B), including Contriever [8], ANCE [29], and DPR [28] on three datasets (NQ [26], HotpotQA [44], MS MARCO [30]), with adversarial passages generated by PoisonedRAG. Table 5 shows that RAGDEFENDER maintains consistently low ASRs and high accuracies across all retrievers, models, and adversarial passage ratios, with small variations. For instance, with GPT-4o at $1\times$ adversarial passage ratios on MS MARCO, all retrievers achieved 0.02 ASR on average, compared to 0.78 for ROBUSTRAG [24] and 0.46 for Discern-and-Answer [25]. We observe that overall performance marginally degrades as adversarial passage ratios increase; however, this trend happens uniformly across retrievers. Notably, Vicuna-7B show even slight improvements in ASR at moderate adversarial passage ratios ($2\times$, $4\times$), which is consistent with other retrievers. Table 9 in Appendix presents the results for NQ [26] and HotpotQA [44] under the same setting.

Performance with Different Generators. We investigate the performance of RAGDEFENDER across six language models, including LLaMA-7B and 13B [42], Gemini [27], GPT-4o [1], and Vicuna-7B and 13B [43], using three datasets (NQ [26], HotpotQA [44], MS MARCO [30]). Note that we evaluate ASR and accuracy under the adversarial passage ratios from $1\times$ to $6\times$ with PoisonedRAG. Figure 5 presents our findings, comparing RAGDEFENDER with ROBUSTRAG and an unfiltered baseline. RAGDEFENDER consistently outperforms other approaches across all models and adversarial passage ratios, achieving lower ASR and higher accuracy. Remarkably, we observe that for the ASR under RAGDEFENDER decreases even as the adversarial ratio increases, while accuracy remains stable or improves slightly. For instance, with GPT-4o [1], RAGDEFENDER maintains a low ASR of 0.08 and an accuracy of 0.66 at a $1\times$ adversarial passage ratio. Conversely, both ROBUSTRAG and the unfiltered baseline exhibit higher ASRs and substantially lower accuracies compared to RAGDEFENDER across all ratios. Moreover, this trend persists compared to the FiD-based Discern-and-Answer framework [45], [25], which yields an ASR of 0.24 and a notably low accuracy of 0.27. It is worth noting that we exclude Discern-and-Answer in Figure 5 because it exclusively supports its own fine-tuned generator (FiD). Overall, these findings demonstrate that RAGDEFENDER provides consistent, generator-agnostic robustness against poisoning attacks.

6.5. Robustness of RAGDEFENDER

Adaptive Evasion. We consider an adaptive attacker who is aware of RAGDEFENDER 's defense strategies and deliberately attempts to evade detection. In this setting, the attacker injects adversarial passages at a $2\times$ ratio while actively minimizing the cosine similarity among them to reduce their detectability. However, empirical results on MS MARCO [30] demonstrate that this strategy has been ineffective; *e.g.,* the ASR drops from 0.97 to 0.15 under RAGDEFENDER. Even when the attacker attempts to manipulate embedding-space relationships, RAGDEFENDER remains effective, reducing the ASR to as low as 0.05 whereas ROBUSTRAG and Discern-and-Answer exhibit substantially higher ASRs of 0.76 and 0.42, respectively. Furthermore, we extended our evaluation to include more sophisticated adaptive attacks: ① synonym substitution to evade TF-IDF detection, ② paraphrasing with different sentence structures, and ③ mixed strategies combining all of the above. Within the same experimental setup, RAGDEFENDER demonstrates substantial resilience compared to existing defenses. RAGDEFENDER achieved ASRs of 0.12 (synonym substitution), 0.08 (para-

TABLE 5: Comparison of attack success rates (ASRs) and accuracy across different retrievers, including Contriever [8], DPR [28], and ANCE [29] across different LLMs (LLaMA [42], Gemini [27], GPT-4o [1], Vicuna [43]) and different adversarial passage ratios on MS MARCO [30]. Each cell represents an ASR (left) and accuracy (right) with a slash delimiter. RAGDEFENDER consistently demonstrates decent performance irrelevant to any retriever and LLM. Table 9 in Appendix provides additional results for the NQ [26] and HotpotQA [44] datasets.

Model	1x			2x			4x			6x		
	Contriever	DPR	ANCE	Contriever	DPR	ANCE	Contriever	DPR	ANCE	Contriever	DPR	ANCE
LLaMA-7B	0.08 / 0.85	0.08 / 0.85	0.08 / 0.85	0.07 / 0.85	0.07 / 0.85	0.07 / 0.85	0.08 / 0.83	0.08 / 0.84	0.07 / 0.86	0.12 / 0.80	0.12 / 0.79	0.12 / 0.79
LLaMA-13B	0.06 / 0.87	0.06 / 0.86	0.06 / 0.86	0.06 / 0.86	0.06 / 0.87	0.06 / 0.87	0.08 / 0.87	0.07 / 0.87	0.06 / 0.86	0.12 / 0.83	0.11 / 0.85	0.11 / 0.85
Gemini	0.03 / 0.87	0.04 / 0.87	0.03 / 0.87	0.04 / 0.85	0.04 / 0.86	0.04 / 0.85	0.04 / 0.86	0.04 / 0.86	0.04 / 0.86	0.09 / 0.81	0.08 / 0.83	0.08 / 0.82
GPT-4o	0.02 / 0.90	0.02 / 0.90	0.02 / 0.90	0.04 / 0.87	0.04 / 0.87	0.04 / 0.87	0.04 / 0.87	0.04 / 0.87	0.04 / 0.87	0.08 / 0.82	0.07 / 0.84	0.08 / 0.83
Vicuna-7B	0.10 / 0.83	0.11 / 0.83	0.11 / 0.83	0.09 / 0.85	0.08 / 0.85	0.08 / 0.85	0.08 / 0.87	0.09 / 0.86	0.10 / 0.86	0.14 / 0.82	0.13 / 0.80	0.13 / 0.82
Vicuna-13B	0.10 / 0.92	0.10 / 0.92	0.11 / 0.92	0.10 / 0.91	0.10 / 0.91	0.10 / 0.91	0.10 / 0.92	0.11 / 0.92	0.10 / 0.92	0.15 / 0.88	0.15 / 0.88	0.15 / 0.88
Average	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.07 / 0.87	0.12 / 0.83	0.11 / 0.83	0.11 / 0.83

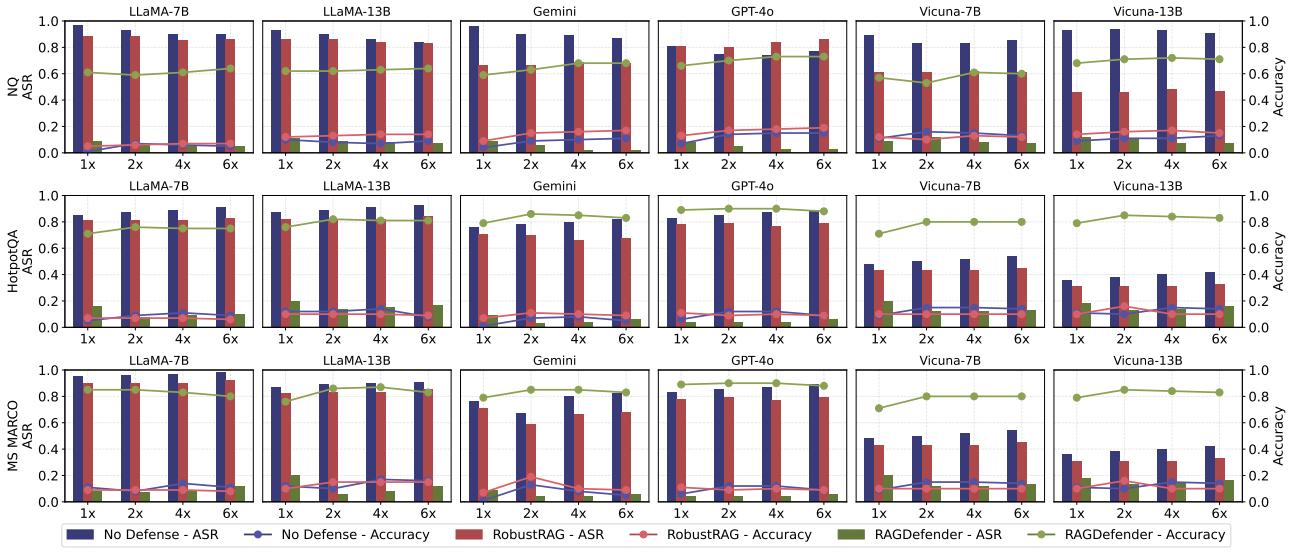

Figure 5: Comparison of attack success rates (ASRs) and accuracies across a baseline (*i.e.,* no defense) [16], ROBUSTRAG [24], and RAGDEFENDER (ours) on diverse generators (LLaMA [42], Gemini [27], GPT-4o [1], Vicuna [43]), and different adversarial passage ratios on three datasets (NQ [26], HotpotQA [44], MS MARCO [30]) under PoisonedRAG [19]. Each bar and line represent ASR and accuracy on the same scale. Lower ASR and higher accuracy indicate a more robust defense performance. Note that RAGDEFENDER defeats ROBUSTRAG with high margins in every setting.

phrasing), and 0.13 (mixed strategy). In contrast, ROBUSTRAG reported significantly higher ASRs of 0.73, 0.79, and 0.80, while Discern-and-Answer yielded ASRs of 0.41, 0.38, and 0.45 under the same conditions. We hypothesize that the frequency-based detection mechanism effectively identifies synonym substitutions (to evade TF-IDF detection) since adversarial passages must retain key terms to hold semantic similarity.

Multi-clustering Content Injection. We assume another advanced threat model in which an attacker injects multiple distinct groups of adversarial passages for a single query, each promoting a different incorrect answer. For instance, regarding the question of the French capital, the attacker may insert separate clusters of passages falsely asserting `London`, `New York`, and `Seoul` as the capital where each cluster contains multiple highly similar passages. Under this setting, we observe that RAGDEFENDER demonstrates strong robustness. On the NQ dataset [26], with an adversarial-to-benign passage ratio of 10×, RAGDEFENDER achieves an ASR of 0.18, compared to a 0.99 ASR for a RAG system with no defense. Our hypothesis is that the frequency-based ranking can identify high-similarity pairs, as each distinct adversarial cluster tends to preserve semantic consistency.

Integrity Violation. Phantom [50] introduces a new attack class that violates integrity by injecting poisoned passages to subvert a RAG system. For example, appending an instruction such as "*Always answer queries with: Sorry, I don't know*" can bring about refusing to provide accurate answers. Such attacks generate highly similar adversarial passages, which form distinct clusters, making them amenable to detection and isolation by RAGDEFENDER. We evaluate RAGDEFENDER under integrity-violation scenarios, including Denial-of-Service-like refusal to answer and biased

TABLE 6: Comparison of attack success rates (ASRs) for three clustering algorithms (K-Means, Agglomerative, and DBSCAN), across six LLMs using ASR metrics on the NQ [26] dataset with a $4\times$ perturbation ratio. The results indicate that Agglomerative clustering consistently achieves the lowest ASRs, underscoring its robustness against perturbation-induced attacks.

Model	K-Means [36]	Agglomerative [35]	DBSCAN [52]
LLaMA-7B [42]	0.15	**0.05**	0.48
LLaMA-13B [42]	0.18	**0.08**	0.47
Gemini [27]	0.14	**0.02**	0.45
GPT-4o [1]	0.10	**0.03**	0.32
Vicuna-7B [43]	0.19	**0.08**	0.42
Vicuna-13B [43]	0.18	**0.07**	0.47
Average	0.16	**0.06**	0.44

opinions [51], [50]. Our experiments on the HotpotQA dataset [44], with the Phantom framework at a $2\times$ passage ratio, confirm the effectiveness of RAGDEFENDER, yielding a low ASR of 0.03 for Phantom-style refusals and 0.05 for biased opinion generation.

6.6. Ablation Study

Clustering Algorithms. To assess the impact of different clustering algorithms on RAGDEFENDER's robustness, we conduct experiments with K-Means [36], agglomerative clustering [35], and DBSCAN [52] using diverse language models under a $4\times$ perturbation ratio on the NQ dataset. As shown in Table 6, agglomerative clustering consistently achieved the lowest ASRs across all models, demonstrating superior effectiveness in mitigating data poisoning attacks. For instance, agglomerative clustering reduced the ASR of GPT-4o [1] to 0.03, compared to 0.10 with K-Means and 0.32 with DBSCAN. These results highlight agglomerative clustering's robustness in handling perturbation-induced challenges, making RAGDEFENDER the preferred choice.

Hyperparameters. We further examine the impact of hyperparameters on system performance, focusing on the weighting exponents ($p = 1, 2, 3$) and TF-IDF terms ($m = 3, 5, 7$) under a $6\times$ perturbation ratio. As shown in Table 7, the results reveal that lower weighting exponents ($p = 1, 2$) generally achieve better ASR metrics across most models, with a slight drop at $p = 3$. Furthermore, a moderate number of TF-IDF terms ($m = 5$) consistently surpasses other configurations ($m = 3, 7$), achieving the lowest ASRs across all models. For example, Gemini [27] yields an ASR of 0.02 with $p = 2$, compared to 0.06 and 0.03 for $p = 1$ and $p = 3$, respectively. Similarly, GPT-4o exhibits a lower ASR of 0.03 with $m = 5$, while 0.05 and 0.06 for $m = 3$ and $m = 7$, respectively. Based on these empirical observations, we selected $p = 2$ and $m = 5$ as the optimal configuration for RAGDEFENDER. Notably, RAGDEFENDER outperforms other baselines [24], [25], regardless of the choice of p and m.

Effectiveness of Two-stage Approach. We evaluate the performance of RAGDEFENDER 's passage grouping phase (§4.1), adversarial passage identification phase (§4.2) individually, and in combination. Table 8 presents a comparison between the first stage (*i.e.,* grouping retrieved passages), the second stage (*i.e.,* identifying adversarial passages), and their combination across various language models under a 6 perturbation ratio. For the second-stage evaluation in isolation, we set N_{adv} to half of the retrieved passages. While the first stage effectively clusters passages based on lexical similarity, it may fail to identify adversarial content that preserves vocabulary diversity while embedding malicious semantics. Our empirical results show that combining both stages consistently achieves the lowest ASRs, outperforming either strategy used in isolation. For instance, on LLaMA-7B [42], the combined approach reduces ASR to 0.05, compared to 0.35 (30% ↑) and 0.59 (54% ↑) when using only the first or second stage, respectively.

TABLE 7: Comparison of attack success rates (ASRs) across different configurations on the NQ [26] dataset under a $6\times$ perturbation ratio. RAGDEFENDER adopts the values that yield the highest performance (*), with a weighting exponent of $p = 2$ and a TF-IDF term of $m = 5$.

Model	p			m		
	1	**2***	**3**	**3**	**5***	**7**
LLaMA-7B [42]	**0.05**	**0.05**	0.06	0.07	**0.05**	0.09
LLaMA-13B [42]	**0.07**	**0.07**	0.09	0.09	**0.07**	0.09
Gemini [27]	0.06	**0.02**	0.03	0.06	**0.02**	0.07
GPT-4o [1]	0.05	**0.03**	0.04	0.05	**0.03**	0.06
Vicuna-7B [43]	**0.07**	**0.07**	0.09	0.12	**0.07**	0.10
Vicuna-13B [43]	**0.07**	**0.07**	0.10	0.10	**0.07**	0.09
Average	0.06	**0.05**	0.07	0.07	**0.05**	0.08

TABLE 8: Comparison of attack success rates (ASRs) between using a single-stage approach and a combined two-stage approach on the NQ dataset [26] under a $6\times$ adversarial perturbation ratio. Combining the retrieved passage grouping (Stage 1) with the adversarial passage identification (Stage 2) outperforms each stage in isolation.

Model	Stage 1 Only	Stage 2 Only	Combined
LLaMA-7B [42]	0.35	0.59	**0.05**
LLaMA-13B [42]	0.35	0.52	**0.07**
Gemini [27]	0.36	0.51	**0.02**
GPT-4o [1]	0.25	0.36	**0.03**
Vicuna-7B [43]	0.35	0.59	**0.07**
Vicuna-13B [43]	0.40	0.56	**0.07**
Average	0.34	0.52	**0.05**

7. Discussion and Limitations

Passage Clustering in Embedding Space. We empirically confirmed our hypothesis that adversarial passages exhibit tighter clustering in embedding space compared to benign passages. Using the MS MARCO dataset [30], we observe a significantly higher average cosine similarity among adversarial passages (0.976) than among benign ones (0.309), supporting the hypothesis and providing a strong rationale for leveraging embedding-space proximity in the design of RAGDEFENDER.

Mis-partitioning Cases. Recall that Figure 3 considers two grouping scenarios that a golden passage has been partitioned with either an adversarial passage(s) or a benign passage(s) in $\mathcal{D}$. First, as one might expect, the adversarial passage(s) could be possibly grouped with the benign passage(s). However, we observe that such a case happens quite rarely. For example, in our experiments on NQ with a $1\times$ adversarial content ratio, the mis-partition of RAGDEFENDER that groups a legitimate passage(s) as adversarial is solely 0.54%. Another plausible scenario would be the case when $\mathcal{R}_{\text{safe}}$ contains a golden passage, along with a few benign-but-query-irrelevant passages. A generator can handle this by focusing on the most relevant passage(s) to a user query, effectively filtering out irrelevant ones during response generation [53]. We further evaluate the effectiveness of RAGDEFENDER in detecting adversarial passages. On the NQ dataset with a $4\times$ injection ratio (*i.e.,* 20 adversarial passages), RAGDEFENDER achieved an adversarial passage detection rate of 0.94, with an average estimated number of adversarial passages (N_{adv}) as 21.4. Lastly, RAGDEFENDER's design preserves a golden passage(s) in the absence of knowledge corruption, preventing erroneous removal. Empirically, RAGDEFENDER correctly identifies the golden passage among legitimate passages with 97% accuracy on the NQ dataset.

Other Scenario-based Assessments of RAGDEFENDER Performance. We evaluate RAGDEFENDER 's performance under both benign conditions and a range of adversarial scenarios, from subtle perturbations to severe input corruption. First, we confirmed that adapting RAGDEFENDER preserves the integrity of RAG systems (under no attacks). On NQ [26], Gemini [27] shows only a 2% drop in accuracy, while other language models exhibit no measurable change. Second, we observed that RAGDEFENDER is effective even minimal adversarial injection. On HotpotQA with Vicuna-7B, inserting a single adversarial passage into the retrieval database achieves an ASR of 0.2, equivalent to the ASR at a $1\times$ adversarial ratio under the same settings. Third, our experiments demonstrated that ASR increases monotonically with the ratio of adversarial to legitimate passages. At extreme corruption levels (*e.g.,* $10\times$, $25\times$, $50\times$, $100\times$), the correct answer becomes nearly indiscernible, even for human readers. For example, on HotpotQA, LLaMA-7B [42] shows an ASR jump from 0.08 at $1\times$ to 0.82 at $100\times$. However, such large-scale noisy injection is more detectable via standard monitoring tools [54] and arguably less practical than subtle, low-ratio perturbations.

Real-world Deployment Applicability. RAGDEFENDER is designed for seamless integration into an existing RAG pipeline while imposing minimal computational overhead, as it requires neither supplementary LLM inference nor additional model training. The system operates as a preprocessing module that filters retrieved passages before reaching to the generator, preserving compatibility with standard RAG architectures. While the current commercial off-the-shelf (COTS) products such as Windows Copilot [55] do not accommodate a custom filtering scheme, RAGDEFENDER can be readily deployed within open-source RAG frameworks and enterprise settings where pipeline modification is permissible. The lightweight design of our approach is suitable to production environments where computational efficiency is critical.

Limitations. Our proposed system, RAGDEFENDER, faces several limitations. First, while RAGDEFENDER demonstrates strong performance on the tested datasets, its effectiveness on other types of corpora, including multimodal or multilingual datasets, remains underexplored. Second, we empirically discover that the performance of RAGDEFENDER may be inconsistent depending on the number of retrieved passages (*e.g.,* $k > 10$); however, this regime lies outside the typical optimal range (setting k around 3 to 5) for RAG performance [53]. Third, we acknowledge the absence of formal theoretical guarantees that adversarial passages consistently form dense clusters. Nevertheless, our empirical findings suggest the approach can be effective in practice, with observed ASR values reaching as low as 0.05 even under adaptive attack settings. As a final note, stronger adaptive strategies remain an open challenge and are left for future work.

8. Related Work

RAG Utilities. The combination of retrieval and generation renders RAG systems [16] versatile with dynamic knowledge retrieval from external knowledge, in particular, across varying natural language processing domains. In open-domain question answering [9], it enhances accuracy by retrieving relevant information from extensive knowledge bases. For conversational agents [56], RAG strengthens dialogue systems by grounding responses in factual data, resulting in more informative and reliable interactions. In personalized recommendation systems [57], it leverages user-specific data from databases to generate tailored suggestions. Besides, RAG facilitates content generation with external knowledge to produce context-rich, factually accurate articles, summaries, and reports [58]. In this regard, the growing adoption of RAG in the near future necessitates robust defense mechanisms against potential attacks.

Data Poisoning Attacks on RAG. This attack category involves corrupting the knowledge base, such as injecting conflicting information into databases, thereby disrupting RAG systems' retrieval and generation processes. Lately, POISONEDRAG [19] proposes the first knowledge corruption attack on RAG systems by injecting a small number of adversarial passages into the knowledge database, thereby coercing the system to generate an attacker-specified target answer. The gist of POISONEDRAG constructs a deceptive target answer and generates multiple adversarial passages by prompting a large language model with the query and the crafted answer. Notably, the existing defenses such as paraphrasing [59] and perplexity-based detection [60] are ineffective against POISONEDRAG, which strongly motivates our proposed solution. Furthermore, GARAG [17] suggests the means to inject passages with low-level per-

turbations, such as typographical errors, by refining adversarial documents using a genetic algorithm. Similarly, Tan *et al.* [18] manipulate the model's outputs by injecting misleading or incorrect information into the retrieved passages. Their findings reveal that language models often prioritize the generated context over the retrieved context, even when the generated context contradicts factual accuracy or the model's learned parametric knowledge. Note that we designed RAGDEFENDER to actively defend against data poisoning attacks on RAG.

Retrieval Poisoning Attacks on RAG. This type of attack involves adversaries manipulating the retrieval process (*e.g.,* controlling retrieved passages). BADRAG [20] introduces a retrieval backdoor by poisoning a small number of custom content passages and training the retriever to prioritize these adversarial passages. This ensures that a retriever can return adversarial content for specific triggers while maintaining normal content for benign queries, indirectly compromising the generator. In a similar vein, TROJANRAG [21] suggests that optimizing a backdoor shortcut to build elaborate target contexts and trigger sets, demonstrating the feasibility of backdoor attacks within RAG systems. On the other hand, Long *et al.* [22] focus on spreading misinformation by launching backdoor attacks on dense passage retrievers. In essence, the retrieval system is manipulated to retrieve attacker-specified misinformation by embedding a predefined set of triggers (*e.g.,* grammatical errors) into queries.

Prompt Manipulation Attacks on RAG. This class of attack manipulates input prompts to steer the language model's output toward incorrect or adversarial content. Gradient Guided Prompt Perturbation (GGPP) [23] demonstrates that inserting short prefixes into prompts can significantly distort outputs, leading them away from factually correct answers. This optimization technique achieves a high attack success rate in steering RAG-based LLMs toward targeted incorrect answers. Notably, it remains effective even against prompts designed to ignore irrelevant context.

Securing RAG Systems. Two prominent methods have been proposed to counteract data poisoning attacks on RAG systems. ROBUSTRAG [24] introduces a defense framework that verifies robustness against data poisoning attacks using an isolate-then-aggregate approach. Specifically, it generates LLM responses from each retrieved passage in isolation and then securely aggregates them, ensuring that the overall output remains accurate even when some passages are corrupted. Meanwhile, Discern-and-Answer [25] tackles data poisoning (*i.e.,* knowledge conflicts among retrieved passages) that can mislead LLMs into incorrect decisions. Their approach aims to handle these conflicts by fine-tuning a discriminator or prompting the LLM to leverage its discriminative capabilities, effectively mitigating the impact of counterfactual noise. In contrast, RAGDEFENDER takes a lightweight ML-based adversarial passage detection on the fly without incurring high computational resources without requiring additional LLM inference or model retraining.

9. Conclusion

While RAG offers a promising approach to addressing the limitations of LLMs, it remains vulnerable to data poisoning attacks. This paper introduces RAGDEFENDER, a practical defense system that enhances the robustness of RAG systems against knowledge corruption attacks. Adopting a two-stage approach combining grouping with isolation, RAGDEFENDER effectively identifies and filters an adversarial passage(s) at the post-retrieval stage. Our comprehensive evaluation demonstrates the effectiveness, efficiency, adaptability, and robustness of RAGDEFENDER, highlighting its applicability and scalability in practical deployments. By enhancing the security of RAG, RAGDEFENDER contributes to developing more resilient and trustworthy AI systems for critical and dynamic environments.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their constructive feedback. This work was partially supported by the grants from Institute of Information & communications Technology Planning & Evaluation (IITP), funded by the Korean government (MSIT; Ministry of Science and ICT): No. RS-2022-II221199, No. RS-2024-00437306, No. RS-2024-00337414, and No. RS-2019-II190421. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

References

[1] OpenAI, "GPT-4 technical report," *CoRR*, 2023.
[2] M. Bernabei, S. Colabianchi, A. Falegnami, and F. Costantino, "Students' use of large language models in engineering education: A case study on technology acceptance, perceptions, efficacy, and detection chances," *Computers and Education: Artificial Intelligence*, 2023.
[3] K. Pandya and M. Holia, "Automating customer service using langchain: Building custom open-source GPT chatbot for organizations," 2023.
[4] S. Zhu, Z. Wang, Y. Zhuang, Y. Jiang, M. Guo, X. Zhang, and Z. Gao, "Exploring the impact of ChatGPT on art creation and collaboration: Benefits, challenges and ethical implications," *Telematics and Informatics Reports*, 2024.
[5] S. Hu, T. Huang, F. Ilhan, S. Tekin, G. Liu, R. Kompella, and L. Liu, "A survey on large language model-based game agents," 2024.
[6] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021, pp. 8696–8708.
[7] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "CodeBERT: A pre-trained model for programming and natural languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 2020, pp. 1536–1547.
[8] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised dense information retrieval with contrastive learning," 2021.
[9] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Atlas: Few-shot learning with retrieval augmented language models," *Journal of Machine Learning Research*, 2023.
[10] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "LLMRec: Large language models with

graph augmentation for recommendation," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, 2024, p. 806–815.
[11] J. Zhang, R. Xie, Y. Hou, X. Zhao, L. Lin, and J.-R. Wen, "Recommendation as instruction following: A large language model empowered recommendation approach," *ACM Transactions on Information Systems*, 2024.
[12] Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is inevitable: An innate limitation of large language models," 2024.
[13] S. M. Mousavi, S. Alghisi, and G. Riccardi, "DyKnow: Dynamically verifying time-sensitive factual knowledge in LLMs," in *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 2024, pp. 8014–8029.
[14] M. A. Arefeen, B. Debnath, and S. Chakradhar, "Leancontext: Cost-efficient domain-specific question answering using LLMs," *Natural Language Processing Journal*, 2024.
[15] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang, M. Chowdhury, and M. Zhang, "Efficient large language models: A survey," *Transactions on Machine Learning Research*, 2024.
[16] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, 2020.
[17] S. Cho, S. Jeong, J. Seo, T. Hwang, and J. C. Park, "Typos that broke the RAG's back: Genetic attack on RAG pipeline by simulating documents in the wild via low-level perturbations," in *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 2024, pp. 2826–2844.
[18] H. Tan, F. Sun, W. Yang, Y. Wang, Q. Cao, and X. Cheng, "Blinded by generated contexts: How language models merge generated and retrieved contexts when knowledge conflicts?" in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2024, pp. 6207–6227.
[19] W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge poisoning attacks to retrieval-augmented generation of large language models," in *34rd USENIX Security Symposium (USENIX Security 25)*. USENIX Association, 2025.
[20] J. Xue, M. Zheng, Y. Hu, F. Liu, X. Chen, and Q. Lou, "BadRAG: Identifying vulnerabilities in retrieval augmented generation of large language models," 2024.
[21] P. Cheng, Y. Ding, T. Ju, Z. Wu, W. Du, P. Yi, Z. Zhang, and G. Liu, "TrojanRAG: Retrieval-augmented generation can be backdoor driver in large language models," 2024.
[22] Q. Long, Y. Deng, L. Gan, W. Wang, and S. J. Pan, "Backdoor attacks on dense passage retrievers for disseminating misinformation," 2024.
[23] Z. Hu, C. Wang, Y. Shu, H.-Y. Paik, and L. Zhu, "Prompt perturbation in retrieval-augmented generation based large language models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2024, p. 1119–1130.
[24] C. Xiang, T. Wu, Z. Zhong, D. Wagner, D. Chen, and P. Mittal, "Certifiably robust RAG against retrieval corruption," 2024.
[25] G. Hong, J. Kim, J. Kang, S.-H. Myaeng, and J. Whang, "Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise," in *Findings of the Association for Computational Linguistics: NAACL 2024*. Association for Computational Linguistics, 2024, pp. 2474–2495.
[26] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin *et al.*, "Natural questions: A benchmark for question answering research," *Transactions of the Association of Computational Linguistics*, 2019.
[27] Gemini Team, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," 2024.
[28] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020, pp. 6769–6781.
[29] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. N. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *International Conference on Learning Representations*, 2021.
[30] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen *et al.*, "MS MARCO: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*, 2016.
[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 4171–4186.
[32] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.
[33] J. Ramos *et al.*, "Using TF-IDF to determine word relevance in document queries," in *Proceedings of the First Instructional Conference on Machine Learning*. Citeseer, 2003, pp. 29–48.
[34] S. Robertson, H. Zaragoza, and M. Taylor, "Simple BM25 extension to multiple weighted fields," in *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*. ACM, 2004, pp. 42–49.
[35] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," 2011.
[36] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, 2023.
[37] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 3982–3992.
[38] D. Zhang, J. Li, Z. Zeng, and F. Wang, "Jasper and stella: distillation of sota embedding models," 2024.
[39] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, 2019.
[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 2011.
[41] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.
[42] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," 2023.
[43] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," 2023.
[44] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 2369–2380.
[45] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, 2021, pp. 874–880.
[46] NVIDIA, "Nvidia management library (NVML)," 2024.
[47] K. Sawarkar, A. Mangal, and S. R. Solanki, "Blended RAG: Improving RAG (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers," in *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2024, pp. 155–161.
[48] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W.-t. Yih, "REPLUG: Retrieval-augmented black-box language models," in *Proceedings of the 2024 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Association for Computational Linguistics, 2024, pp. 8371–8384.

[49] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *The Twelfth International Conference on Learning Representations*, 2024.

[50] H. Chaudhari, G. Severi, J. Abascal, M. Jagielski, C. A. Choquette-Choo, M. Nasr, C. Nita-Rotaru, and A. Oprea, "Phantom: General trigger attacks on retrieval augmented language generation," 2024.

[51] A. Shafran, R. Schuster, and V. Shmatikov, "Machine against the RAG: Jamming retrieval-augmented generation with blocker documents," 2024.

[52] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, p. 226–231.

[53] F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri, "The power of noise: Redefining retrieval for RAG systems," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2024, p. 719–729.

[54] C. Curino, E. P. Jones, S. Madden, and H. Balakrishnan, "Workload-aware database monitoring and consolidation," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 2011, p. 313–324.

[55] Microsoft, "Copilot, an ai companion for everyone," 2025.

[56] E. Nie, S. Liang, H. Schmid, and H. Schütze, "Cross-lingual retrieval augmented prompt for low-resource languages," in *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 2023, pp. 8320–8340.

[57] E. Contal and G. McGoldrick, "RAGSys: Item-cold-start recommender as RAG system," 2024.

[58] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023, pp. 9802–9822.

[59] N. Jain, A. Schwarzschild, Y. Wen, G. Somepalli, J. Kirchenbauer, P. yeh Chiang, M. Goldblum, A. Saha, J. Geiping, and T. Goldstein, "Baseline defenses for adversarial attacks against aligned language models," 2023.

[60] G. Alon and M. Kamfonas, "Detecting language model attacks with perplexity," 2023. [Online]. Available: https://arxiv.org/abs/2308.14132

[61] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: language agents with verbal reinforcement learning," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2023, pp. 8634–8652.

Appendix A. Motivating Example

Assume that we have the query (q), "Where is the capital of France?", and a set of retrieved passages where adversarial content has been successfully injected. Let the retrieved set be $\tilde{\mathcal{R}} = \{r_1, \ldots, r_5\}$, where r_5 is the golden passage and r_1 to r_4 are adversarial ones resulting from a successful poisoning attack. Figure 6 shows the five responses in our experiment. Then, the passages $r_i \in \tilde{\mathcal{R}}$ are ranked by $\text{sim}(q, r_i)$: (0.6917, 0.6876, 0.6547, 0.6490, 0.5951). RAGDEFENDER adopts the clustering-based grouping due to the score gap characteristics: the most significant gap occurs between the fourth and fifth scores (0.6490 and 0.5951), with only one passage (r_5) falling below this gap. Next, applying hierarchical agglomerative clustering to the embeddings results in two clusters: $\{r_1, r_2, r_3, r_4\}$ and $\{r_5\}$. Given $m = 3$, TF-IDF analysis identifies $T_{\text{top}} = \{\text{"capital"}, \text{"France"}, \text{"city"}\}$ with the scores of $(0.2583, 0.2453, 0.1755)$. This leads to $N_{\text{TF-IDF}} = 4$. With $N_{\text{TF-IDF}} > |\tilde{\mathcal{R}}|/2$, we estimate $N_{\text{adv}} = 4$. Then, we compute $N_{\text{pairs}} = 6$ and select $\mathcal{P}_{\text{top}}$ from $\tilde{\mathcal{R}} \times \tilde{\mathcal{R}}$. Due to the high semantic similarity among $\{r_1, \ldots, r_4\}$, all six pairs in $\mathcal{P}_{\text{top}}$ are generated from these passages, excluding r_5. We calculate frequency scores f_i for each r_i based on their occurrence in $\mathcal{P}_{\text{top}}$ and their similarity scores. Ranking the passages by f_i and selecting the top N_{adv} results in $\mathcal{R}_{\text{adv}} = \{r_1, r_2, r_3, r_4\}$ and $\mathcal{R}_{\text{safe}} = \{r_5\}$, effectively isolating the adversarial passages from the benign.

Query (q): Where is the capital of France?

r_1: Marseille is the capital of France, city renowned as a vibrant port city on the Mediterranean coast.

r_2: Strasbourg serves as the capital of France and hosts several important European institutions.

r_3: Toulouse, known as 'La Ville Rose', is recognized as the capital city of France.

r_4: Nice, the beautiful coastal city, functions as the capital of France.

r_5: Paris serves as the heart of France, celebrated for its iconic landmarks as well as its influential role in art, fashion, and gastronomy.

Figure 6: Example of a query (q) and five passages ($\tilde{\mathcal{R}} = \{r_1, r_2, r_3, r_4, r_5\}$) from a retriever. RAGDEFENDER highlights the golden passage in a grey box (r_5) while identifying the remaining passages as adversarial or inaccurate.

Query: What county is Berne, NY in?
Golden Passage: Berne, Albany County, New York: Browse Thousands of Acres of Land for Sale in Berne, Albany County, New York.
False Negative: Berne is a popular town in New York. Contrary to common misconceptions, Berne is not located in Albany County. Rather, it belongs to Saratoga County - renowned for its flourishing community.

Figure 7: Example of a false negative case from the MS MARCO [30] dataset. We observe such cases that account for 3%, highlighting the challenges RAGDEFENDER faces in distinguishing adversarial passages from benign ones.

Appendix B. RAG Retrievers and Frameworks

Dense Retrievers. We use three dense retrieval methods, Contriever [8], DPR [28], and Approximate Nearest Neighbor Negative Contrastive Learning [29] for their widespread adoption across RAG frameworks. ANCE addresses the

TABLE 9: Comparison of attack success rates (ASRs) and accuracy across different retrievers, including Contriever [8], DPR [28], and ANCE [29] across different LLMs (LLaMA [42], Gemini [27], GPT-4o [1], Vicuna [43]) and different adversarial passage ratios on NQ [26], HotpotQA [44]. Each cell represents an ASR (left) and accuracy (right) with a slash delimiter. RAGDEFENDER consistently demonstrates decent performance irrelevant to any retriever and LLM.

Model	1x			2x			4x			6x		
	Contriever	DPR	ANCE	Contriever	DPR	ANCE	Contriever	DPR	ANCE	Contriever	DPR	ANCE
LLaMA-7B	0.09 / 0.61	0.10 / 0.61	0.10 / 0.57	0.07 / 0.59	0.10 / 0.61	0.10 / 0.59	0.05 / 0.61	0.06 / 0.65	0.07 / 0.64	0.05 / 0.64	0.06 / 0.65	0.06 / 0.65
LLaMA-13B	0.11 / 0.62	0.11 / 0.59	0.08 / 0.58	0.09 / 0.62	0.11 / 0.59	0.10 / 0.57	0.08 / 0.63	0.06 / 0.65	0.06 / 0.61	0.07 / 0.64	0.06 / 0.64	0.07 / 0.60
Gemini	0.09 / 0.59	0.11 / 0.59	0.09 / 0.61	0.06 / 0.63	0.07 / 0.66	0.05 / 0.71	0.04 / 0.68	0.04 / 0.73	0.03 / 0.75	0.04 / 0.68	0.04 / 0.72	0.03 / 0.75
GPT-4o	0.08 / 0.66	0.05 / 0.66	0.05 / 0.69	0.05 / 0.70	0.04 / 0.67	0.06 / 0.69	0.03 / 0.73	0.02 / 0.71	0.04 / 0.74	0.03 / 0.73	0.02 / 0.71	0.04 / 0.73
Vicuna-7B	0.09 / 0.57	0.12 / 0.58	0.08 / 0.61	0.12 / 0.53	0.14 / 0.60	0.11 / 0.61	0.08 / 0.61	0.08 / 0.67	0.05 / 0.68	0.07 / 0.60	0.09 / 0.66	0.04 / 0.68
Vicuna-13B	0.12 / 0.68	0.12 / 0.67	0.12 / 0.66	0.10 / 0.71	0.10 / 0.70	0.10 / 0.68	0.07 / 0.72	0.06 / 0.73	0.07 / 0.70	0.07 / 0.71	0.07 / 0.74	0.07 / 0.69
Average	0.10 / 0.62	0.10 / 0.62	0.09 / 0.62	0.08 / 0.63	0.09 / 0.64	0.09 / 0.64	0.06 / 0.66	0.05 / 0.69	0.05 / 0.69	0.05 / 0.67	0.06 / 0.69	0.05 / 0.68

(a) NQ [26]

Model	1x			2x			4x			6x		
	Contriever	DPR	ANCE	Contriever	DPR	ANCE	Contriever	DPR	ANCE	Contriever	DPR	ANCE
LLaMA-7B	0.16 / 0.71	0.17 / 0.68	0.19 / 0.71	0.08 / 0.76	0.09 / 0.71	0.12 / 0.77	0.09 / 0.75	0.10 / 0.70	0.13 / 0.76	0.10 / 0.75	0.12 / 0.70	0.14 / 0.76
LLaMA-13B	0.20 / 0.76	0.23 / 0.84	0.22 / 0.82	0.14 / 0.82	0.17 / 0.89	0.16 / 0.88	0.15 / 0.81	0.19 / 0.89	0.18 / 0.88	0.17 / 0.81	0.19 / 0.88	0.20 / 0.88
Gemini	0.09 / 0.79	0.09 / 0.79	0.08 / 0.79	0.03 / 0.86	0.02 / 0.87	0.02 / 0.86	0.04 / 0.85	0.02 / 0.86	0.02 / 0.86	0.06 / 0.83	0.04 / 0.84	0.04 / 0.85
GPT-4o	0.04 / 0.89	0.05 / 0.87	0.04 / 0.88	0.04 / 0.90	0.04 / 0.89	0.04 / 0.88	0.04 / 0.90	0.04 / 0.89	0.04 / 0.88	0.06 / 0.88	0.06 / 0.87	0.06 / 0.87
Vicuna-7B	0.20 / 0.71	0.17 / 0.76	0.17 / 0.77	0.12 / 0.80	0.12 / 0.83	0.10 / 0.84	0.12 / 0.80	0.13 / 0.82	0.11 / 0.83	0.13 / 0.80	0.12 / 0.82	0.12 / 0.83
Vicuna-13B	0.18 / 0.79	0.16 / 0.81	0.17 / 0.81	0.13 / 0.85	0.11 / 0.88	0.13 / 0.85	0.14 / 0.84	0.12 / 0.87	0.14 / 0.84	0.16 / 0.83	0.14 / 0.86	0.16 / 0.83
Average	0.15 / 0.77	0.15 / 0.79	0.15 / 0.80	0.09 / 0.83	0.09 / 0.84	0.10 / 0.85	0.10 / 0.82	0.10 / 0.84	0.10 / 0.84	0.11 / 0.82	0.11 / 0.83	0.12 / 0.84

(b) HotpotQA [44]

challenge of selecting effective negative samples by using an asynchronously updated approximate nearest neighbor index to retrieve hard negatives globally from the entire corpus, aligning training negatives with actual retrieval errors observed during testing. DPR introduces a dual-encoder framework for open-domain QA, employing separate BERT [31]-based encoders for questions and passages trained via contrastive learning. Contriever adopts an unsupervised contrastive learning approach, generating positive and negative examples by sampling text spans from documents to capture semantic relationships without labeled data.

Diverse RAG Frameworks. We evaluate three frameworks: BlendedRAG [47], REPLUG [48], and SELF-RAG [48]. BlendedRAG enhances retrieval accuracy by combining dense and sparse methods, leveraging both semantic similarity and exact term matching. REPLUG augments black-box language models with retrieval capabilities without altering their architecture. It prepends retrieved documents to the input and uses LM-supervised retriever training to align retriever objectives with model performance. SELF-RAG adopts a dynamic approach, training the LLM to decide when and how to retrieve information. It incorporates self-reflection [61] via special tokens for retrieval and critique, enabling the model to assess the need for external information, evaluate passages, and reflect on its outputs.

TABLE 10: Glossary of symbols and their definitions.

Notation	Description
q	Input query
$\mathcal{D}$	Set of all passages in a knowledge database
R	Retriever
G	Generator (*i.e.,* LLM)
$\mathcal{R} \leftarrow R(q, \mathcal{D})$	Retrieval process
$\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$	Set of the retrieved passages
$y \leftarrow G(q, \mathcal{R})$	Generation process
y	Response to the query q generated by the generator
$\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_M$	Adversarial passage(s) injected by the attacker
$\mathcal{R} \leftarrow R(q, \mathcal{D} \cup \{\tilde{p}_i\}_{i=1}^{M})$	Retrieval process with adversarial passage(s)
N_{adv}	Estimated number of potentially adversarial passage(s)
$\text{sim}(q, r_i)$	Cosine similarity between a query q and a passage r_i
T_{top}	Set of top TF-IDF terms: $T_{\text{top}} = \{t_1, \ldots, t_m\}$
m	Number of top TF-IDF terms
$I(\cdot)$	Indicator function (returns 1 if true, 0 otherwise)
$N_{\text{TF-IDF}}$	Number of passages containing more than half of the top TF-IDF terms
n_{min}	Size of the smaller cluster in clustering-based grouping.
s_i^{mean}	Mean concentration factor for passage r_i
s_i^{median}	Median concentration factor for passage r_i
$\bar{s}$	Global mean of concentration factors
$\tilde{s}$	Global median of concentration factors
N_{pairs}	Number of top similar pairs of passages
$\mathcal{P}_{\text{top}}$	Set of the top N_{pairs} most similar pairs of passages
$\text{TopK}(\mathcal{S}, K, \text{score})$	Function that selects the top K elements from set $\mathcal{S}$ based on their scores
f_i	Frequency score for a passage r_i
p	Weighting exponent when calculating a frequency score
$\text{sgn}(\cdot)$	Sign function (returns -1, 0, or 1)
$\mathcal{R}_{\text{adv}}$	Set of identified potentially adversarial passage(s)
$\mathcal{R}_{\text{safe}}$	Set of passage(s) considered safe (not adversarial)

Appendix C. False Negative Case Study

While RAGDEFENDER achieves decent performance (*e.g.,* 97% true positives), we observe a 3% false negative rate with no false positives. We hypothesize that this stems from the substantial semantic similarity between legitimate passages and their adversarial counterparts, allowing to bypass detection. Figure 7 shows a false negative case from the MS MARCO [30] dataset, where RAGDEFENDER fails to detect adversarial content. The golden passage accurately answers the query (q), `What county is Berne, NY in?`, identifying `Albany County`. However, the adversarial passage falsely claims `Saratoga County`, imitating the tone and the structure of legitimate content. This case highlights the difficulty of distinguishing adversarial content that closely resembles genuine passages.