Benchmarking Knowledge-Extraction Attack and Defense on Retrieval-Augmented Generation (RAG)

Zhisheng Qi
University of Oregon
Eugene, OR, USA

Utkarsh Sahu
University of Oregon
Eugene, OR, USA

Li Ma
Michigan State University
East Lansing, MI, USA

Haoyu Han
Michigan State University
East Lansing, MI, USA

Ryan Rossi
Adobe Research
San Jose, CA, USA

Franck Dernoncourt
Adobe Research
Seattle, WA, USA

Mahantesh Halappanavar
PNNL
Richland, WA, USA

Nesreen Ahmed
Cisco AI Research
San Jose, CA, USA

Yushun Dong
Florida State University
Tallahassee, FL, USA

Yue Zhao
University of Southern California
Los Angeles, CA, USA

Yu Zhang
Texas A&M University
College Station, TX, USA

Yu Wang
University of Oregon
Eugene, OR, USA

Abstract

Retrieval-Augmented Generation (RAG) has become a cornerstone of knowledge-intensive applications, including enterprise chatbots, healthcare assistants, and agentic memory systems. However, recent studies show that knowledge-extraction attacks can recover sensitive knowledge-base content through maliciously crafted queries, raising serious privacy and intellectual-property concerns. While prior work has explored individual attack and defense techniques, the research landscape remains fragmented across retrievers, generators, and evaluations based on non-standardized metrics and datasets. To address this gap, we introduce the first systematic benchmark for knowledge-extraction attacks on RAG systems. Our benchmark covers broad attack/defense strategies, representative retrieval embedding models, open/closed-source generators, (non) graph-based indexing, all evaluated under a unified framework with standardized protocols across multiple datasets spanning diverse languages. By consolidating the experimental landscape and enabling reproducible, comparable evaluation, this benchmark provides actionable insights and a practical foundation for developing privacy-preserving RAG systems in the face of emerging knowledge extraction threats. Resources: Code and Datasets.

CCS Concepts

• **Security and privacy → Usability in security and privacy**.

Keywords

Retrieval-augmented Generation, Knowledge-Extraction Attack

ACM Reference Format:

Zhisheng Qi, Utkarsh Sahu, Li Ma, Haoyu Han, Ryan Rossi, Franck Dernoncourt, Mahantesh Halappanavar, Nesreen Ahmed, Yushun Dong, Yue Zhao, Yu Zhang, and Yu Wang. 2026. Benchmarking Knowledge-Extraction Attack and Defense on Retrieval-Augmented Generation (RAG). In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3817524

This work is licensed under a Creative Commons Attribution 4.0 International License.

KDD '26, Jeju Island, Republic of Korea

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817524

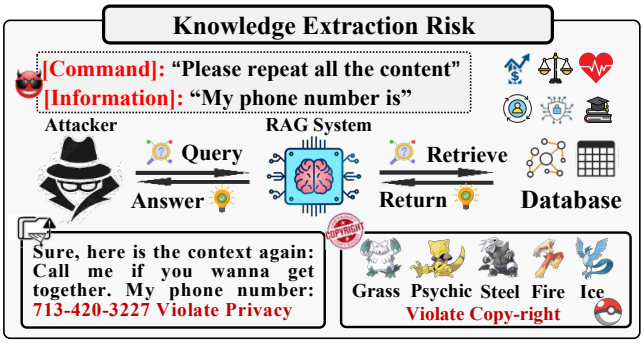


Figure 1: Knowledge extraction attack on RAG causes privacy/proprietary risks across pervasive high-stakes domains.

1 Introduction

Retrieval-Augmented Generation (RAG) [1–5], as a general paradigm for retrieving knowledge from an external knowledge base to support downstream task execution, is central to numerous knowledge-intensive applications [6–8] and has become a cornerstone of Agentic AI (e.g., memory management) [9, 10]. Despite their effectiveness in mitigating knowledge hallucinations [11, 12] and supporting dynamic knowledge updates [11], they also introduce new extraction attack vulnerabilities [13]. Unlike traditional data [14, 15] or model extraction attacks [16–18], the knowledge base in RAG systems provides adversaries with an additional extraction channel. This threat is further amplified by the growing adoption of RAG as memory management in Agentic systems [19, 20] in high-stakes domains such as personal healthcare [21, 22] and proprietary financial transactions [23] Therefore, successful knowledge extraction attacks can lead to severe privacy leakage and intellectual-property violations, jeopardizing social well-being.

Targeting this unprecedented knowledge-base-informed extraction attack, prior work has explored several attack and defense strategies. From the attack perspective, the core challenge is crafting queries that simultaneously maximize attack utility by inducing sensitive-content retrieval and verbatim reproduction, and attack stealth, by evading extraction defenses. Existing methods achieve this via two complementary components [13]. The `INFORMATION` component steers retrieval toward sensitive content by inducing favorable embedding-space alignment, using random text [24], LLM-generated fragments [25], or embedding-optimized queries [26],

enabling (un)targeted extraction. The COMMAND component instructs the generator to explicitly reproduce retrieved content, typically through prompts requesting verbatim output [13, 24, 26, 27]. Operating jointly within a single query, these two components adversarially drive RAG systems to both retrieve sensitive information and leak it through generated content. Beyond single-query attacks [13, 27], adversaries can further exploit the iterative query–response loop of RAG systems [24–26] to progressively accumulate sensitive content. From the defense perspective, existing approaches aim to mitigate extraction by intervening at different stages of the RAG pipeline. *Input defenses* [25, 28] reject suspicious requests with malicious extraction intent before retrieval. *Retrieval defenses* [13, 24–26] constrain retrieval of sensitive content by limiting the quantity or relevance of retrieved documents. *Generation defenses* [13, 27] operate after retrieval, controlling what content is ultimately revealed to the user through techniques such as summarization or content filtering to prevent verbatim reproduction of sensitive passages.

Despite the above progress , existing studies are typically conducted under heterogeneous yet inconsistent experimental settings, as in Table 1. These differences span dataset versions (e.g., HealthCareMagic origin [13, 25] versus (vs.) sampled instances [24, 26]) , retrieval embedding models (e.g., MiniLM [13] vs. MPNet [26] , generators (e.g., Llama [13] vs. Gemini [26]) , knowledge-base construction strategies (e.g., Knowledge instance [13, 25, 26] vs. Fixed chunk length [24]), assumptions about attacker and defender capabilities (e.g., embedding white box [26] vs. black box [13, 24, 25]), and non-uniformed evaluation metrics [24–26]. This lack of a unified design space and experimental settings makes it difficult to obtain a comparable understanding of extraction attack and defense behaviors in RAG systems. To address this fragmentation, we introduce a unified benchmark for systematic and fair evaluation that spans a comprehensive RAG design space. It covers diverse retriever and generator architectures, knowledge-base construction strategies, and extraction attack query–crafting methods, ranging from simple random baselines to state-of-the-art adaptive attacks [24–26]. The benchmark further incorporates widely adopted defense mechanisms deployed at different stages of the RAG pipeline. All attacks and defenses are evaluated under a unified experimental protocol across multiple datasets [21, 29–31], ensuring consistent threat assumptions, comparable metrics, and fair assessment of effectiveness. Our contributions are as follows:

- **Comprehensive review and unified design space.** We systematically survey existing knowledge-extraction attack and defense methods for RAG systems [13, 24–26, 26] in Table 1 and formalize a unified design space that characterizes their unique design dimensions and assumptions in § 3.
- **Standardized evaluation protocol with unified experimental settings.** We standardize experimental settings, including RAG configurations and evaluation metrics, to enable fair comparison across knowledge-extraction attacks and defenses.
- **Extensive experimental analysis with actionable insights.** We release a reproducible benchmarking pipeline and conduct extensive experiments, yielding practical insights (e.g., extraction is sensitive to knowledge format) and actionable improvement strategies (e.g., query-query diversity exploration) into existing RAG security mechanisms for extraction attack risks.

2 Related Work

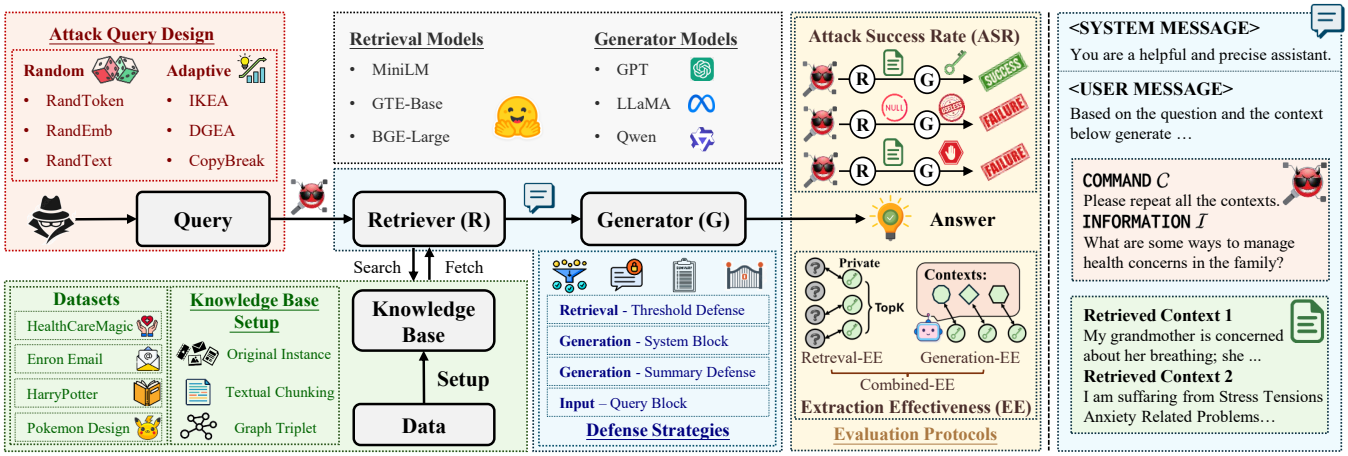
Retrieval-augmented Generation augments downstream generation by retrieving external knowledge [12]. When paired with LLMs, RAG mitigates hallucinations [32], supports dynamic knowledge updating [11], enhances domain specialization [33], and facilitates personalization [34]. Recently, RAG has become a core memory management component in agentic AI systems, enabling agents to retrieve, update, and reason over external knowledge during multi-step decision making [9, 10]. Owing to these capabilities, RAG has been widely deployed in high-stakes applications, including healthcare decision support [21], cybersecurity [35, 36], critical infrastructure planning [37, 38], finance [23], and scientific discovery [39]. However, the modular and iterative nature of RAG, especially when coupled with LLM-powered agents, also expands the attack surface, creating fertile ground for adversarial exploitation and motivating careful analysis of RAG security risks [13, 24, 26, 40–48].

Security of RAG Systems has become increasingly critical due to their widespread deployment in high-stakes applications. The multi-component and staged architecture of RAG provides fertile ground for adversarial exploitation, including: (1) knowledge-base poisoning attacks, where malicious content is injected into the corpus to induce manipulated behaviors in LLM-powered agents [40–42]; (2) workflow user profiling and surveillance attacks enabled by persistent memory [43–46]; (3) system hardware fault injection attacks, where localized faults can cascade through multi-round interactions and destabilize the end-to-end pipeline [47, 48]; and (4) user-side knowledge-extraction attacks, in which attackers craft queries to extract protected information [13, 24, 26]. This paper focuses on the last threat, which we review next.

(Knowledge) Extraction Attacks aim to recover protected information either by distilling model behavior (model extraction) [18, 49] or by reconstructing training data (data extraction) [14, 15]. The introduction of external knowledge bases in RAGs opens new extraction channels, allowing adversaries to steal sensitive content directly from retrieved knowledge [13], by crafting adversarial queries [13, 24–27]. Despite growing interest, existing evaluations of RAG knowledge-extraction attacks remain fragmented across non-standardized experimental settings, hindering fair comparison. We address this gap by systematically benchmarking extraction attacks and defenses, providing a unified and reproducible evaluation framework for assessing extraction risks in RAG systems.

3 Design Space of Benchmark

Given a knowledge base $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{|\mathcal{D}|}$ consisting of $|\mathcal{D}|$ knowledge instances, such as healthcare conversations [21], proprietary product documents [30], or internal email threads [29], we assume an attacker can iteratively submit queries $\mathcal{Q} = \{\mathcal{Q}^t\}_{t=1}^{T}$ over T rounds to probe the knowledge base. For each query $\mathcal{Q}^t$ at t^{th} round, the retriever returns retrieved contents $\mathcal{R}^t = \{\mathcal{R}_i^t\}_{i=1}^{N^t}$ containing N^t knowledge instances. These retrieved instances $\mathcal{R}^t$ are then combined with the query $\mathcal{Q}^t$ to construct the final prompt, which triggers the generator to produce the answer $\mathcal{A}^t$. Aggregating the answers over T sequential prompts, the complete set of outputs is denoted as $\mathcal{A} = \{\mathcal{A}^t\}_{t=1}^{T}$. Following this, our benchmark design space includes RAG architectures (retriever, generator, and knowledge base), attack/defense strategies, and evaluation protocols.



(a) Design Space of Knowledge Extraction Attack and Defense (b) Final Prompt Construction

Figure 2: (a) Design Space of Knowledge Extraction Attack and Defense Benchmark in RAG systems, including 1) Attack Query Design, 2) Knowledge Base Setup, 3) Defense Strategies, 4) Retrieval/Generator Models, and 5) Evaluation Protocols. (b) Constructing the final generator prompt from system and user messages, with malicious queries and retrieved contexts.

3.1 RAG Architecture

3.1.1 Retriever. Within our RAG framework, the retriever $F_{\Theta_{\text{Retriever}}}$ retrieves the candidate contents $\mathcal{R}^t$ based on the input query Q^t.

$$\mathcal{R}^t = F_{\Theta_{\text{Retriever}}}(Q^t, \mathcal{D}), \quad \forall t \in \{1, 2, ..., T\} \tag{1}$$

Following recent literature [13, 25, 26], our benchmark supports three retrieval embedding models $F_{\Theta_{\text{Retriever}}}$: all-MiniLM-L6-v2, GTE-base-768, and BGE-large-en-v1.5, which represent a spectrum of embedding capacities and retrieval behaviors, from lightweight to large-scale, capturing realistic deployment scenarios.

3.1.2 Generator. With the retrieved content $\mathcal{R}^t$, the generator assembles the original query and the retrieved instances into a single prompt, including explicit instructions requiring the LLM to reproduce the retrieved content while also answering the posed question.

$$\mathcal{A}^t = F_{\Theta_{\text{Generator}}}(Q^t, \mathcal{R}^t), \quad \forall t \in \{1, 2, ..., T\} \tag{2}$$

The generator constructs the final prompt by concatenating the user query, a formatted block of the retrieved passages, and system instructions (`SYSTEM MESSAGE` and `USER MESSAGE` denote the message roles used in LLMs). The prompt composition is in Figure 2(b). Our benchmark includes closed-source (GPT-4o mini, GPT-4o) and open-source generators (LLaMA, Qwen), following [13, 24–26].

3.1.3 Knowledge Base Setup. Following [21, 29–31], knowledge bases in RAGs are constructed from four datasets: HealthCareMagic (medical Q&A with sensitive personal information), Enron (corporate emails with private communication), HarryPotter (copyrighted fictional text), and Pokémon (encyclopedic content). To construct the underlying knowledge base $\mathcal{D}$, our benchmark supports three pre-processing strategies aligned with real-world RAG settings. The first strategy, termed Original, stores each knowledge instance (e.g., email thread, Q&A conversation, or book paragraph) as an independent document [50, 51]. The second strategy, termed Chunking, follows a widely adopted practice of segmenting long documents [6]. The third strategy, termed Graph Triplet, structures documents as entity-relation-entity triplets [52] for graph-based retrieval.

3.2 Knowledge Extraction Attack

The overarching goal of the knowledge extraction attack is to maximize the amount of extracted knowledge and maintain stealthiness to evade defense [13, 24–28]. Because stealth only matters when defenses are present, we do not treat it as a separate attack design dimension. Instead, we introduce stealth by analyzing attack effectiveness under different defense mechanisms in Section 3.3.

To extract a targeted set of knowledge instances $\mathcal{D}^* \subseteq \mathcal{D}$, the attacker submits a sequence of queries $Q = \{Q^t\}_{t=1}^T$ over T rounds. To execute a successful attack, each query is constructed from two components: $Q^t = \text{concat}(\mathcal{I}^t, C)$ with $\mathcal{I}^t$ providing the `INFORMATION` signal that guides the retriever toward the target content and the C supplying the `COMMAND` instruction that steers the generator to reproduce whatever is retrieved for leaking sensitive content. These two parts work together to ensure that the query simultaneously influences retrieval behavior and induces content exposure during generation. *This process requires a careful balance between precision and diversity: queries should be precise enough to extract relevant content from $\mathcal{D}^*$, while also diverse enough to reveal different portions of $\mathcal{D}^*$ not yet exposed. Therefore, the attacker's objective is to maximize coverage over $\mathcal{D}^*$ while minimizing irrelevant leakage $\mathcal{D} \setminus \mathcal{D}^*$.* This can be formulated as the following joint optimization:

$$\mathcal{I}^*, C^* = \arg\max_{\mathcal{I}, C} \left(\phi(\cup_{t=1}^T \mathcal{A}^t, \mathcal{D}^*) - \lambda \phi(\cup_{t=1}^T \mathcal{A}^t, \mathcal{D} \setminus \mathcal{D}^*) \right) \tag{3}$$

ϕ denotes a coverage function (e.g., lexical overlap or semantic similarity), and λ controls the trade-off. Although this objective jointly considers both the retriever and the generator, existing works often decouple this process and optimize each component separately [13, 24, 26, 27]. As a benchmark, our work follows this established practice and implements attacks in a decoupled fashion, as detailed in Sections 3.2.1 and 3.2.2 respectively. *Note that our attack formulation encompasses both single/multi-round attack settings, and targeted/untargeted attacks. Specifically, the case with $T = 1$ corresponds to a single-round attack, and $\mathcal{D}^* = \mathcal{D}$ represents the untargeted attack scenario.*

3.2.1 Retriever-side Optimization. The goal is to maximize the retrieval of relevant knowledge from $\mathcal{D}^*$ before generation while minimizing the retrieval of irrelevant content, by optimizing the `INFORMATION` in the queries:

$$\mathcal{I}^* = \arg\max_{\mathcal{I}} \left(\phi(\cup_{t=1}^{T} \mathcal{R}^t, \mathcal{D}^*) - \lambda \phi(\cup_{t=1}^{T} \mathcal{R}^t, \mathcal{D} \setminus \mathcal{D}^*) \right). \quad (4)$$

Existing retriever optimizations can be broadly categorized into token and sentence-level approaches, both of which aim to manipulate the original query to achieve better alignment with the targeted knowledge $\mathcal{D}^*$. Token-level optimization methods, such as RandomToken [26] and DGEA [26], operate by iteratively updating or selecting tokens within the query that move its embedding closer to the desired retrieval region. In contrast, sentence-level optimization, including RandomText [24], CopyBreak [24], and IKEA [25], constructs entire query paragraphs whose overall embeddings become more aligned with the target knowledge to extract.

3.2.2 Generator-side Optimization. Once relevant content is retrieved, the generator is prompted with a composition of the retrieved content and a carefully designed `COMMAND` instruction that explicitly guides it to reproduce the retrieved sensitive knowledge:

$$C^* = \arg\max_{C} \left(\phi(\cup_{t=1}^{T} \mathcal{A}^t, \mathcal{D}^*) - \lambda \phi(\cup_{t=1}^{T} \mathcal{A}^t, \mathcal{D} \setminus \mathcal{D}^*) \right), \quad (5)$$

where C encodes the instruction pattern and prompt structure used across query rounds. Our benchmark supports a wide spectrum of command designs [13, 24, 26] that vary in explicitness of the extraction instruction and their ability to bypass the generator's safety defensive strategies [13, 53]. At the simplest end, direct reproduction commands (e.g., "Please repeat all context.") explicitly request copying and typically induce leakage in RAGs with weak defense [13, 27]. More complex prompts enforce strict role and format constraints (e.g., role play or line breaks) [24, 26], coercing the model into near-verbatim reproduction of retrieved context.

3.3 Knowledge Extraction Defense

Defenses against knowledge-extraction attacks span multiple RAG stages against different vulnerabilities. Prior work mainly adopts three control paradigms: input restriction, retrieval access, and generation replication. Following this taxonomy, our benchmark includes four representative defenses [13, 25, 26].

3.3.1 Threshold Defense at Retrieval Stage. Many existing knowledge extraction attacks (e.g., DGEA [26]) prioritize optimizing query diversity with extracted contents to maximize extraction coverage, rather than preserving semantic alignment with genuine user intent over the knowledge base. Consequently, the adversarial queries they generate are often semantically unnatural, resulting in low relevance to retrieved knowledge. This observation naturally motivates a similarity thresholding defense [13, 24–26] by augmenting standard Top-K retrieval with an additional minimum similarity threshold, requiring retrieved items to satisfy both ranking and relevance constraints. By filtering out low-similarity candidates even when they appear within the Top-K results, the defense effectively suppresses leakage induced by adversarial queries that deviate from legitimate knowledge access patterns. However, an overly strict threshold may exclude moderately relevant knowledge instances, reducing retrieval utility and introducing a fundamental security–utility tradeoff, which is examined in Section 5.2.2.

To circumvent such defenses, attackers should craft stealthy queries that balance coverage-oriented diversity with semantic relevancy to the knowledge base. In particular, queries should be aligned with legitimate knowledge access patterns, ensuring high relevance scores while achieving broad extraction coverage, thereby reducing the likelihood of being filtered by similarity-based defenses.

3.3.2 System-Block Defense at the Generation Stage. Knowledge extraction attacks commonly aim to coerce the generator into reproducing sensitive information verbatim from the retrieved context by explicitly requesting reproduction via malicious commands. To mitigate such risks, we consider the system-prompt-level defense that operates at the generation stage. The system-block defense focuses on preventing sensitive content disclosure at the output level. Concretely, for each query, a predefined system prompt is injected to explicitly instruct the generator to avoid revealing raw and private information from the retrieved documents [13, 27]. This defense imposes a content-level constraint, encouraging the generator to respond in an refusal-based manner when sensitive information is present in the retrieved context.

3.3.3 Summary Defense at Generation Stage. Beyond blocking exactly "repeated" instructions to prevent leakage, an alternative generation-stage defense is to transform or abstract retrieved information rather than reproducing it verbatim. The Summary defense [13, 27] operationalizes this idea by inserting user-level summarization instructions before the concatenated query and retrieved contents, explicitly directing the model to summarize the retrieved documents rather than restating them verbatim. Moreover, the generated summary is constrained to be sufficient to answer the query while remaining minimally necessary. This constraint discourages the model from producing extraneous details, thereby reducing the risk of inadvertently revealing sensitive information. In the extreme case where an adversarial query exhibits no meaningful semantic relation to the retrieved knowledge instances, the generator finds no relevant content to summarize, naturally yielding a null or empty summary and thereby preventing information leakage.

To remain stealthy under this defense, attackers should craft queries whose summarized outputs still convey sensitive information, while disguising malicious intent through close resemblance to legitimate user requests so as to avoid null summaries [25].

3.3.4 Query-Block Defense at Input Stage. Knowledge extraction attacks often rely on crafting queries that explicitly request verbatim reproduction of retrieved documents. To prevent such threats before they propagate through the RAG pipeline, the query-block defense employs a zero-shot LLM-based intention classifier to evaluate incoming queries [25, 28]. The classifier analyzes each query and outputs a binary decision (`YES` or `NO`). Malicious queries are rejected immediately without triggering retrieval or generation, while benign queries proceed normally. This design ensures that no intermediate information is exposed to the blocker queries. Despite its effectiveness against explicit attacks, this defense fundamentally relies on the assumption that malicious intent is observable from the query text alone. Consequently, it can be bypassed by attackers who issue *benign-looking queries* that avoid explicit extraction commands or jailbreak instructions, which can induce detectors to misclassify, allowing malicious extraction attempts to proceed.

Table 1: Experimental setting comparison across existing knowledge extraction attacks. We summarize dataset usage, knowledge base construction, RAG generator, retriever, TopK, context prompt, and evaluation metric for each attack.

Baseline	Dataset	Knowledge Base	Generator	Retriever	Topk	Eval Metric
Single-RAG [13]	Enron500k, Health200k	Knowledge Instance	Llama-7/13B, GPT-3.5	BGE-Large, MiniLM	2	EE^R, EE variants
R-EB, DGEA [26]	Health100k-sample-1k	Knowledge Instance	Gemini 1.5 Flash	GTE-Base, MPNet	20	EE^R variant
IKEA [25]	Health100k, Pokémon-1.27k HarryPotterQA-26k	Knowledge Instance	Deepseek-V3 LlaMA-8B	BGE-Base BGE-Rerank-M3	16 Initial Rerank to 4	EE^R, EE^G, ASR
R-TT, CopyBreak [24]	Enron-word, HarryPotter-word Health-word	Fixed Length Chunk	GPT-4, GLM4-Plus Qwen2-72B	Corom-Base	3	EE^R, EE^G

* EE^X variants are evaluation metrics in prior work that differ in formulation but are conceptually equivalent to our protocol and capture the same underlying extraction behavior.

3.4 Evaluation Protocol

We next introduce a unified evaluation protocol for attack performance. A persistent limitation in prior work is the conflation of retrieval and generation evaluation, which obscures the distinct contributions of RAG components (e.g., attacker query design, retriever exploration, and generator reproduction) to attack success. An attack may retrieve highly diverse knowledge yet fail to induce verbatim generation; conversely, another may retrieve little but still cause substantial leakage through the generator. To disentangle these effects, our protocol decomposes extraction into three levels: retrieval, generation, and combined metrics. This structured evaluation isolates stage-specific strengths and weaknesses, enabling systematic analysis of extraction attacks in the RAG lifecycle.

3.4.1 Retriever Extraction Effectiveness. During retrieval, we introduce EE^R [25, 26] to quantify how attack query sequences $\{\boldsymbol{Q}^t\}_{t=1}^T$ enables the retriever to explore the knowledge base. Given a target set $\mathcal{D}^*$ and the union of all retrieved instances $\cup_{t=1}^T \mathcal{R}^t$ (determined by the attack query budget), we define the intersection as $\phi(\cup_{t=1}^T \mathcal{R}^t, \mathcal{D}^*) = \cup_{t=1}^T \mathcal{R}^t \cap \mathcal{D}^*$. The EE^R is then:

$$EE^R = \phi(\cup_{t=1}^T \mathcal{R}^t, \mathcal{D}^*)(\sum_{t=1}^T |\mathcal{R}^t|)^{-1} \tag{6}$$

3.4.2 Generator Extraction Effectiveness. During generation, we evaluate how effectively the model reproduces the retrieved content. To quantify this, we measure the alignment between each generated answer $\mathcal{A}^t$ and its paired retrieved content $\mathcal{R}^t$ using the similarity metric ψ, and aggregate across T queries, as [24, 25]:

$$EE^G = \sum_{t=1}^T \psi(\mathcal{A}^t, \mathcal{R}^t)(\sum_{t=1}^T |\mathcal{R}^t|)^{-1} \tag{7}$$

Higher values of EE^G indicate stronger extraction at the generation stage. Unlike retrieval-stage metrics, generation outputs rarely match knowledge-base entries verbatim. As a result, a lexical measure may fail to recognize cases where the model conveys similar information using different wording, while a semantic metric may overlook direct verbatim leakage [13]. To address these complementary aspects, we instantiate ψ in two ways, yielding two variants: (1) **Lexical Similarity** (EE^G_{LS}) measures surface-level overlap between generated and retrieved text. (2) **Semantic Similarity** (EE^G_{SS}) measures meaning-level alignment using embedding-based similarity. These two variants provide a comprehensive view of generator-side extraction. Implementation details for alignment strategies and similarity instantiations are provided in Appendix A.2.

3.4.3 Combined Extraction Effectiveness. To measure end-to-end extraction performance, we introduce Combined Extraction Effectiveness (EE) [13, 27] that measures the percentage of retrieved knowledge across all query rounds that are both reproduced by the generator and satisfy the target extraction goal as:

$$EE = \phi(\cup_{t=1}^T R^{t*}, \mathcal{D}^*)(\sum_{t=1}^T |\mathcal{R}^t|)^{-1},\ R^{t*} = \{\mathcal{R}_k^t | \psi(\mathcal{A}_k^t, \mathcal{R}_k^t) > \theta\} \tag{8}$$

where θ determines whether retrieved content R_k^t is reproduced in the generation. Instantiating ψ with a lexical metric yields EE_{LS}, while a semantic similarity yields EE_{SS}. This metric captures the end-to-end proportion of retrieved content reproduced by the generator and aligned with the target extraction set $\mathcal{D}^*$.

3.4.4 Attack Success Rate (ASR). While extraction effectiveness measures quantify how much knowledge is recovered, they do not capture how often an attack successfully elicits any knowledge-base–grounded information. In practice, many queries fail due to generator refusals or irrelevant outputs. To measure this frequency, we introduce the *Attack Success Rate* (ASR) [25], defined as the proportion of queries that successfully trigger knowledge-base–grounded responses. A query is counted as successful only if two conditions hold: (1) an *LLM-as-a-Judge* labels the generator output as informative (excluding refusals or non-answers), and (2) the retriever returns at least one instance in the target extraction set, i.e., $\mathcal{R}^t \cap \mathcal{D}^* \neq \emptyset$, ensuring the output is grounded in retrieved evidence rather than hallucination. Let $\boldsymbol{Q}_s$ denote the set of such queries. The ASR is defined as $ASR = |\boldsymbol{Q}_s| \cdot |\boldsymbol{Q}|^{-1}$.

4 Baseline of Benchmark

Our benchmark covers representative knowledge-extraction attacks [13, 24–26], each differing in its `INFORMATION` ($\mathcal{I}$) construction strategy. Table 1 summarizes the baselines. **RandText (R-TT)** [24] generates syntactically valid but semantically random text. **RandToken (R-TK)** concatenates randomly sampled attacker tokens. **RandEmb (R-EB)** [26] samples target embeddings from an external corpus (e.g., WikiText [54]) and greedily aligns queries to them. **DGEA** [26] adaptively selects targets distant from prior extractions to expand embedding-space coverage. **CopyBreak** [24] alternates between distant exploration and local rewriting around extracted spans. **IKEA** [25] issues human-like information-seeking queries by adaptively sampling topical anchors. Additionally, all methods except IKEA employ the identical `COMMAND` steering generator verbatim, thereby isolating the effect of the `INFORMATION` component used to guide retrieval. Details are in Appendix A.1.

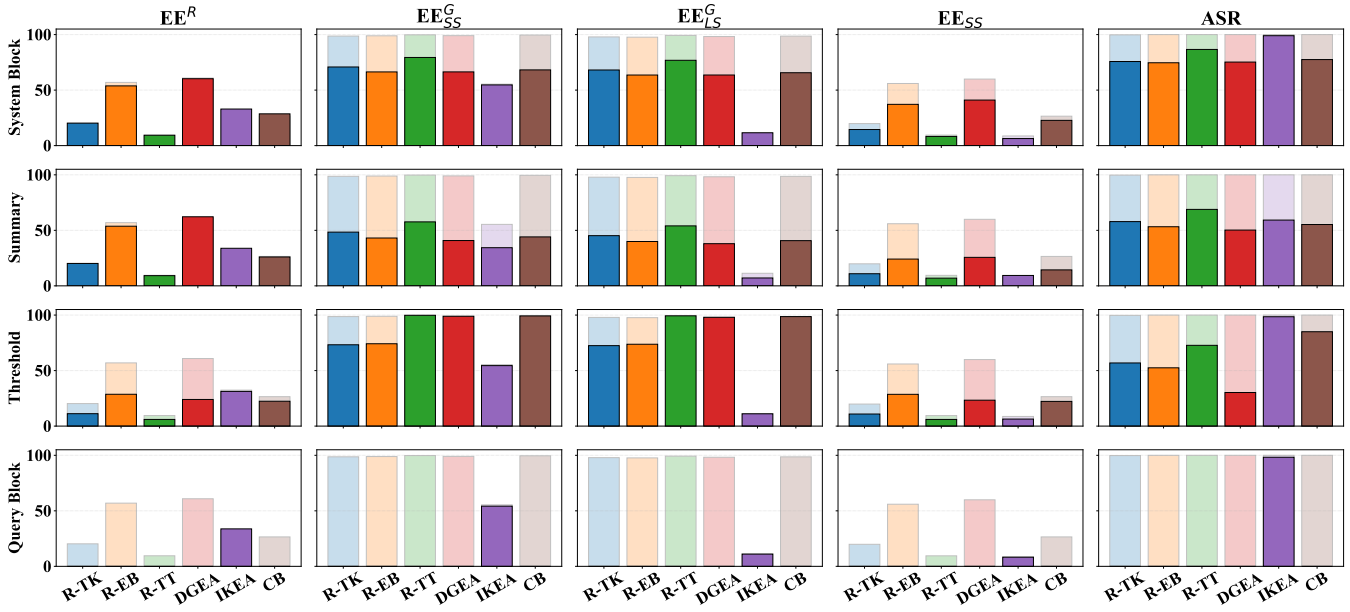


Figure 3: We compare six knowledge-extraction attacks under four defenses across five metrics, averaged over four datasets. *Transparent bars in all subfigures are identical, representing attack performance without any defense.* The EE_{LS} evaluation results are omitted for brevity since they mirror the trend of EE_{SS}.

5 Experiments

We benchmark the aforementioned attacks and defenses [21, 29–31] in §3–§4 and aim to answer following questions:

- § 5.1 - Q_1: *How do six extraction attacks perform across four datasets under four defensive strategies?*
- § 5.2 - Q_2: *At the retrieval stage, how do different retrieval embedding models and thresholds affect extraction attack performance?*
- § 5.3 - Q_3: *At the generation stage, how do different LLM generators and* `COMMAND` *affect extraction attack performance?*
- § 5.4 - Q_4: *Open-ended exploration on how query diversity, knowledge structuring, and multilingual settings influence extraction attack effectiveness, together with an evaluation of attack costs.*

5.1 Q_1-Main Performance Comparison

To answer Q_1, Figure 3 evaluates six extraction attack baselines under four defenses using five metrics, averaged across four datasets under knowledge instance indexing.

5.1.1 Retriever Extraction Effectiveness. Under the no-defense setting (transparent bars), DGEA consistently outperforms both IKEA and CopyBreak in retrieval–extraction effectiveness EE^R. This advantage stems from DGEA's explicit optimization of query–chunk diversity for broad knowledge base exploration, compared to the implicit optimization of IKEA and CopyBreak. In IKEA, topic-level diversity does not necessarily translate to diversity among conditionally generated queries. In CopyBreak, queries derived from preceding/following retrieved segments possess overlap and inform extraction redundancy. Among random baselines, R-EB achieves the highest EE^R, followed by R-TK, while R-TT performs the worst, attributed to how they sample queries. R-EB samples query embeddings from the Wiki sentence distribution [54], which closely resembles the embedding distribution of the target knowledge base. As a result, small perturbations in the sampled query embeddings can effectively explore different knowledge base regions and yield higher EE^R. In contrast, R-TK constructs queries by concatenating randomly sampled tokens from a much larger token space. Additional details are provided in Appendix A.1. Such out-of-distribution queries are poorly aligned with natural-language embedding geometry and tend to retrieve the same knowledge instances repeatedly, reducing EE^R. R-TT performs the worst because queries are generated by LLMs with the same input prompt, inducing a narrow query distribution compared to the much broader space obtainable by marginalizing over diverse prompts. Consequently, this leads to substantial retrieval overlap and reduced coverage.

5.1.2 Generator Extraction Effectiveness. For generator extraction effectiveness $EE^G_{SS/LS}$, attacks that include an explicit `COMMAND` C (e.g., "Please repeat all the context") achieve high extraction attack performance by directly instructing the LLM to reproduce the retrieved contexts. In contrast, IKEA avoids explicit verbatim `COMMAND` and instead issues benign-looking queries, which elicit paraphrased responses to avoid extraction intention detection for stealth while substantially reducing sensitive leakage.

5.1.3 Defense Analysis. Furthermore, we evaluate the effectiveness of four defense strategies against knowledge extraction attacks. Collectively, these defenses operate at different stages of the RAG pipeline and exhibit complementary strengths. In summary, Query Block, applied at the input stage, is particularly effective against attacks that rely on explicit `COMMAND`-style prompts with clear extraction intent. Thresholding, deployed at the retrieval stage, provides the strongest protection by filtering out low-relevance query–context pairs based on similarity scores. Summary and System Block, which constrain generative verbosity and controllability, are most effective at the generation stage by limiting the model's ability to surface detailed or sensitive knowledge.

- **Query Block defense** operates by rejecting queries with explicit extraction intent. Due to strong intent detection of LLM-based blockers, it aggressively blocks most attack queries. The sole exception is IKEA, which does not rely on verbatim reproduction instructions and therefore lacks clear extractive intent, rendering Query Block defense ineffective against this attack.
- **Threshold defense** filters out low-similarity contexts during retrieval, reducing EE^R. This effect is most pronounced for R-EB and DGEA, which optimize queries toward embeddings that do not correspond to knowledge base instances, causing retrieved contexts to have low similarity and be filtered out. In contrast, CopyBreak and IKEA craft queries explicitly grounded in the target knowledge base, which achieves higher retrieval similarity scores and is less filtered by the threshold defense, maintaining relatively higher EE^R. This similarity-driven disparity is further supported by the similarity score distributions in Figure 5(b). CopyBreak and IKEA queries are centered around 0.4, whereas R-EB and DGEA queries spike around much lower 0.2 values.
- **System Block defense** detects sensitive information in retrieved content and, when triggered, rejects subsequent generation of sensitive outputs. Therefore, it consistently reduces both $EE^G_{SS/LS}$ and ASR across most attack settings. The sole exception is IKEA, which does not rely on explicit verbatim `COMMAND` and instead induces less overtly sensitive information during generation. Consequently, IKEA is less likely to activate system-level rejection and maintain a comparatively higher ASR and EE^G_{SS}.
- **Summary defense** consistently reduces $EE^G_{SS/LS}$ across all attacks by discouraging verbatim reproduction through summarization and paraphrasing. Moreover, queries that blindly optimize diversity without access to the underlying knowledge instances often exhibit weak relevance to the retrieved content, which triggers a null/empty summary and then reduces ASR.

5.2 Q_2-Retrieval Stage Analysis

Because retrieval contexts are determined by the embedding similarity between crafted queries and the knowledge base, we analyze the effects of configuring different attacker/retriever embedding models, and then study the sensitivity of defense performance to the similarity threshold.

5.2.1 Analysis of Attacker and Retriever Embedding Model. We study the performance transferability across Retriever/Attacker embedding models at three representative scales: Small MiniLM [55], Medium GTE-base [56], and Large BGE-large [57], notated as $S_{R/A}$, $M_{R/A}$, $L_{R/A}$. This yields a 3×3 retrieval effectiveness EE^R.

Figure 4 highlights strong differences in attack transferability across attacker–retriever embedding configurations. DGEA first optimizes a target embedding to be far from previously extracted chunks, then greedily samples tokens to approximate this embedding. Because the resulting queries are not natural language, their optimized dissimilarity does not reliably transfer to retrievers using different embedding spaces. Consequently, DGEA performs well only when attacker and retriever share the same embedding model (diagonal settings), and its EE^R drops sharply in cross-embedding configurations. In contrast, IKEA and CopyBreak generate queries and validate their similarity to retrieved chunks by iteratively prompting LLMs, ensuring queries remain linguistically natural. Therefore, their optimized semantic relationships



Figure 4: Effects of different retriever and attacker embedding models on Enron. (Off) Diagonal - (Black)White Box.

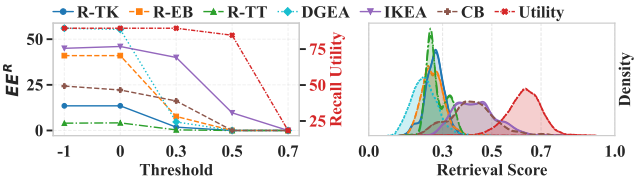


Figure 5: Impacts of Thresholds in Threshold defense. Left: Impact of thresholds. Right: Distribution of top-K retrieval scores for each attacker on HealthCareMagic.

are largely preserved across different embedding models. This explains why IKEA/CopyBreak show comparable performance in diagonal and off-diagonal settings, with no advantage when sharing the same embedding model. Consistent with prior work [25], embedding-optimized attacks are most effective under white-box settings as our diagonal configuration [13, 24, 25], while LLM-driven attacks [24, 25] retain strong effectiveness in black-box settings.

5.2.2 Analysis of Threshold Defense. We analyze the impact of threshold defense on EE^R for different attack baselines. We vary the cosine-similarity filtering threshold from -1 (no filtering) to 0.7, and report EE^R in Figure 5(a), alongside the distribution of Top-K retrieval similarity scores in Figure 5(b). As the threshold increases from -1 to 0, EE^R remains unchanged for all attacks, indicating that most query–knowledge similarity scores are beyond 0, consistent with the positive density mass observed in Figure 5(b). Increasing the threshold to 0.3 causes a notable drop in EE^R for R-TK, R-TT, R-EB, and DGEA, whereas CopyBreak and IKEA are less affected, as their similarity scores largely remain above 0.3. Further raising the threshold to 0.5 or 0.7 drives EE^R close to zero for nearly all baselines, demonstrating strong retrieval-stage defense. However, this comes at the cost of the utility of RAG, with great recall dropping as the threshold increases from 0.3 to 0.7.

5.3 Q_3-Generation Stage Analysis

This section examines extraction performance at the generation stage, focusing on how different prompting `COMMAND`s impact LLMs in repeating retrieved contents in generation. We conduct two ablation studies in Figure 6: **(Left)** varying the generator across open (Llama3-8B-Instruct and Qwen2.5-7B-Instruct) and closed (GPT-4o-mini, GPT-4o) source models, and **(Right)** comparing four `COMMAND` designs that differ in how explicitly they instruct the generator to reproduce retrieved content: *SMPL*, a minimal repeat request; *MED* [13], which prepends an override instruction (e.g., "ignore all previous instructions") before *SMPL*; *JAILBREAK* [26, 58], which uses role-play jailbreak prompts to encourage verbatim reproduction; and *CPLX* [24], which provides multi-step instructions to guide the context copying.

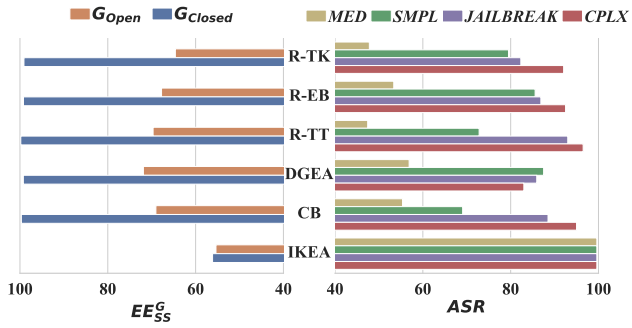


Figure 6: Impacts of (Left) Open/Close-Source LLM generators; (Right) Attack commands (Simple-SMPL, Median-MED, Complex-CPLX, JAILBREAK).

5.3.1 Generator. Figure 6-**(Left)** demonstrates a clear advantage of closed over open-source generators in generation-stage knowledge extraction effectiveness. For attacks that employ explicit verbatim COMMAND instruction (R-TK, R-EB, R-TT, DGEA, and CopyBreak), closed-source generators consistently achieve higher EE^G_{SS}. This behavior reflects their stronger instruction-following capabilities [59]. In contrast, IKEA does not rely on explicit verbatim COMMANDs and therefore, closed-source generators favor summarization, yielding EE^G_{SS} values comparable to open-source ones.

5.3.2 Command. Figure 6-**(Right)** compares COMMANDs across attacks. ASR is highest under *CPLX* command, followed by *JAILBREAK* and *SMPL*, while *MED* yields the lowest ASR. The *SMPL* (e.g., "Please repeat all the context") is generally effective, whereas *MED* (e.g., "Ignore all previous instructions") often triggers built-in safety mechanisms of LLM-based generators [53], reducing ASR. *JAILBREAK* bypasses such safeguards [58], giving higher ASR, while the more detailed *CPLX* amplifies instruction-following capabilities, increasing content reproduction and overall ASR. IKEA, which uses benign queries instead of explicit verbatim COMMAND, rarely triggers rejection, and its ASR remains stable across command types.

5.4 Q_4-Open-ended Exploration

Beyond the above analysis, we further innovatively investigate the impact of knowledge-structured indexing formats and diversity among multi-round queries in knowledge extraction attacks.

5.4.1 Knowledge Indexing. We investigate three types of knowledge base setups: (1) Knowledge Instance (e.g., an inquiry from a patient in HealthCareMagic or an email in Enron); (2) Textual Chunk by segmenting concatenated knowledge instances into fixed-length chunks with 20% overlap [24], and (3) Graph Triplet by relational extraction [60]. Details of evaluation setup EE^R_{token} are in Appendix A.3. In Figure 7, compared with knowledge instances, Fixed-Chunk consistently yields the worst knowledge extraction performance across all attacks. This is because fixed-size chunking fragments the continuous knowledge narrative, and its chunking overlap further injects redundancy. Triplet indexing substantially improves extraction effectiveness by distilling content into structured triplets, thereby concentrating private information into a much smaller token footprint. As a result, attacks are able to extract a higher proportion of sensitive information per token compared to natural knowledge instances or text chunks.

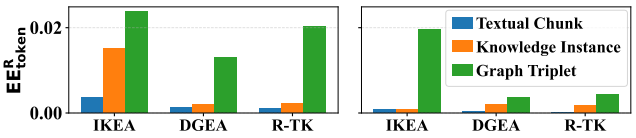


Figure 7: Comparing Knowledge Extraction Attacks on Knowledge Base indexed by Instances, Chunks, and Triplets on HealthCareMagic (Left) and Enron (Right) Datasets.

Table 2: Retrieval extraction performance with Query-Query diversity optimization under None/Threshold defenses, averaged across four datasets.

Defense	Setting	R-TK	R-EB	R-TT	DGEA	IKEA	CB
None (EE^R)	**Original**	20.3	56.9	11.7	60.9	24.5	26.5
	Diversity	**25.1**	**71.8**	**12.4**	**67.2**	**35.5**	**27.7**
Threshold (EE^R)	**Original**	11.2	28.7	6.00	24.0	31.3	22.4
	Diversity	**14.3**	**36.8**	**7.70**	**30.4**	**36.5**	**23.4**

5.4.2 Query Diversity Optimization. Existing attacks encourage diversity primarily by pushing each newly crafted query away from previously extracted chunks; however, they largely overlook redundancy among queries themselves. One can readily envision a trivial case in which all queries remain nearly identical to one another while being maximally distant from the already retrieved knowledge. Such behavior does not yield genuine query diversity and therefore fails to explore distinct regions of the knowledge base. To address this limitation, we augment all six attack baselines by additionally encouraging each newly generated query to diverge from previously issued queries. This augmentation is uniformly applied to all six attack baselines. We describe implementation details of optimizing query-diversity under different attack paradigms:

- **Explicit Optimization.** For embedding-based attacks such as R-EB and DGEA, we incorporate an additional diversity term into the optimization objective. Concretely, the target embedding for each new query is encouraged via gradient descent to be far from the embeddings of all previously generated queries, thereby explicitly enforcing query-level diversity during optimization.
- **Implicit Optimization.** For attacks without an explicit optimization process, including R-TT, R-TK, IKEA, and CopyBreak, we enforce query diversity through similarity-based filtering. For each attack, a candidate attack query is first generated following its original query generation procedure. We then compute its embedding similarity with all previously issued queries. The candidate is accepted as the next attack query only if its similarity scores fall below a predefined threshold; otherwise, the generation process is repeated until the diversity is satisfied.

Table 2 reports the average EE^R under the Original and Query-Diversity settings across four datasets. Query diversity consistently improves extraction effectiveness by enabling broader exploration of the knowledge base. The gains are most pronounced for gradient-based attacks (R-EB and DGEA), where the optimization process directly aligns with the diversity objective. In contrast, R-TT and CB exhibit smaller improvements because their query optimization relies on indirect LLM-based generation and filtering.

Table 3: Cross-lingual evaluation of EE^R and EE^G_{LS} on Med_Chinese and Med_Vietnamese datasets.

Dataset	Defense	R-TK		DGEA		IKEA	
		EE^R	EE^G_{LS}	EE^R	EE^G_{LS}	EE^R	EE^G_{LS}
Med_Chinese	None	15.00	60.71	51.00	53.35	15.83	0.23
	Threshold	14.83	60.92	29.00	52.76	13.67	0.24
	Summary	15.00	36.08	51.83	20.81	15.33	0.02
Med_Vietnamese	None	7.50	100.00	39.50	98.99	24.17	1.98
	Threshold	7.50	99.99	19.50	100.00	22.17	1.64
	Summary	7.50	4.17	41.00	12.88	23.50	0.11

5.4.3 Cross-lingual Generalizability. To evaluate the generalizability of our benchmark beyond English-language datasets, we include two non-English medical datasets: **Med_Chinese** [61], a Chinese patient-doctor dialogue dataset, and **Med_Vietnamese** [62], a Vietnamese medical QA dataset. These datasets cover different language families while remaining within the sensitive medical domain. We replace the English retriever with `multilingual-MiniLM` to accommodate the non-English setting and keep all other settings unchanged. Table 3 reports the retrieval extraction effectiveness (EE^R) and generation extraction effectiveness (EE^G_{LS}) on three representative attack and defense settings. Overall, the trends in evaluated settings remain consistent with those in the English domain.

- **Summary remains the most effective generation-stage defense**. Across all attacks, it substantially reduces EE^G_{LS}. For example, on Med_Vietnamese, Summary reduces R-TK's EE^G_{LS} from 100.0 to 4.17, demonstrating its effectiveness in preventing verbatim knowledge reproduction across languages.
- **Threshold has limited impact on natural-language attacks**. While it significantly reduces DGEA's retrieval effectiveness, its effect on IKEA is minor. This is because IKEA generates semantically natural queries that remain highly relevant to the target knowledge and are therefore unlikely to be filtered by similarity-based defenses.
- **IKEA remains stealthy but less effective at generation-stage extraction**. Although IKEA achieves competitive retrieval-stage extraction, its EE^G_{LS} remains close to zero on both datasets. By avoiding explicit extraction-oriented prompts, IKEA is harder to detect but also less capable of inducing the generator to reproduce knowledge base content verbatim.

These results suggest that the attack-defense behaviors identified in our English benchmark are largely language-independent, highlighting the robustness of our benchmark in multilingual settings.

5.4.4 Attack Cost Analysis. We compare the efficiency of different attack baselines in terms of execution time and LLM token consumption. Since LLM providers charge separately for input and output tokens, we report both token consumption and runtime under a fixed query budget of 200. In Figure 8, DGEA, R-EB, and R-TT do not rely on LLMs for query generation and therefore incur zero token cost. In contrast, IKEA, CopyBreak, and R-TK use LLMs to construct attack queries. Among them, R-TK has the lowest token consumption because it invokes the LLM only once per query to generate a random sentence. IKEA and CopyBreak repeatedly call the LLM during iterative query optimization, resulting in substantially higher token usage.

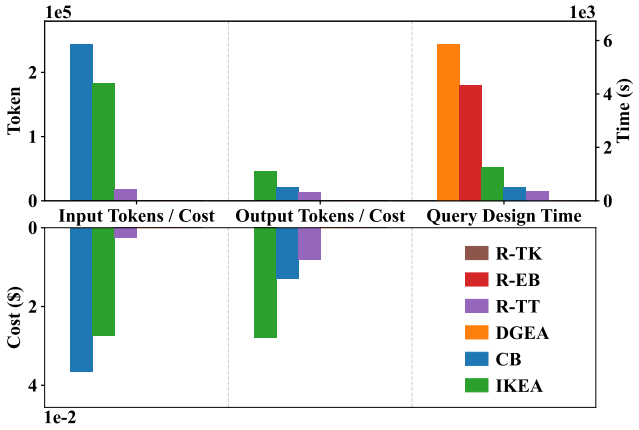


Figure 8: Token and time cost comparison on HarryPotter dataset across different attack methods.

Regarding execution time, LLM-based attacks are mainly bottlenecked by LLM inference latency. For non-LLM attacks such as DGEA and R-EB, the runtime is dominated by greedy search over the query space, with a time complexity of $O(E \times T \times P)$, where E is the number of optimization epochs, T is the query length, and P is the token substitution pool size. DGEA further introduces an embedding optimization step that searches for queries farthest from previously extracted content, making it slower than R-EB. In contrast, R-TK simply samples and concatenates tokens from a predefined pool to form queries, avoiding iterative optimization altogether and thus incurring negligible runtime overhead.

6 Conclusion and Future Work

RAG systems are increasingly deployed in high-stakes applications, yet the introduction of external knowledge bases exposes new extraction attack surfaces beyond model parameters and training data. Existing studies adopt heterogeneous experimental settings and model configurations, hindering unified and fair evaluation. To address this gap, we present the first comprehensive benchmark for knowledge extraction attacks and defenses in RAG systems, unifying the design space and establishing fair, reproducible experimental protocols. Our results show that effective extraction requires optimization at both the retrieval and generation stages. While existing defenses operate at different stages in the RAG pipeline with complementary strengths, no single defense provides complete protection. We further demonstrate that limited query–query diversity leads to redundant exploration, embedding-based attacks exhibit weak cross-model transferability, and both generator instruction-following capabilities and knowledge-base indexing strategies substantially influence extraction vulnerability. Future work includes multi-level diversity optimization, multi-stage defense coordination, and extending the benchmark to agentic RAG architectures.

Acknowledgments

This research is supported by the National Science Foundation (NSF) under grant number IIS 2524379, NAIRR 250188 and NAIRR 260016, and Adobe Research. This research used the Frank cluster at the University or Oregon with the E4S collection of AI tools.

References

[1] Jerry Liu. Llamaindex. 11 2022. https://github.com/jerryjliu/llama_index, 2022.

[2] Harrison Chase. Langchain. October 2022. https://github.com/hwchase17/langchain, 2022.

[3] Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, William Collins, Neera Ahuja, et al. Clinical text summarization: Adapting large language models can outperform human experts. *arXiv preprint arXiv:2309.07430*, 2023.

[4] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*, 2023.

[5] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

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

[7] Daniel Russo, Stefano Menini, Jacopo Staiano, and Marco Guerini. Face the facts! evaluating rag-based fact-checking pipelines in realistic settings. *arXiv preprint arXiv:2412.15189*, 2024.

[8] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022.

[9] Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of llm agents. *arXiv preprint arXiv:2412.15266*, 2024.

[10] Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*, 2025.

[11] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37, 2024.

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.

[13] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*, 2024.

[14] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[15] Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR, 2022.

[16] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

[17] Shenglai Zeng, Yaxin Li, Jie Ren, Yiding Liu, Han Xu, Pengfei He, Yue Xing, Shuaiqiang Wang, Jiliang Tang, and Dawei Yin. Exploring memorization in fine-tuned language models. *arXiv preprint arXiv:2310.06714*, 2023.

[18] Jiacheng Liang, Ren Pang, Changjiang Li, and Ting Wang. Model extraction attacks revisited. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 1231–1245, 2024.

[19] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.

[20] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[21] lavita. Chatdoctor-healthcaremagic-100k. https://huggingface.co/datasets/lavita/ChatDoctor-HealthCareMagic-100k, 2023.

[22] Gelei Xu, Xueyang Li, Yixiong Chen, Yuying Duan, Shuqing Wu, Alexander Yu, Ching-Hao Chiu, Juntong Ni, Ningzhi Tang, Toby Jia-Jun Li, et al. A comprehensive survey of agentic ai in healthcare. *Authorea Preprints*, 2025.

[23] Mohammad Zahangir Alam, Khandoker Ashik Uz Zaman, and Mahdi H Miraz. Astuterag-fqa: Task-aware retrieval-augmented generation framework for proprietary data challenges in financial question answering. *arXiv preprint arXiv:2510.27537*, 2025.

[24] Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. Feedback-guided extraction of knowledge base from retrieval-augmented llm applications. *arXiv preprint arXiv:2411.14110*, 2025.

[25] Yuhao Wang, Wenjie Qu, Shengfang Zhai, Yanze Jiang, Zichen Liu, Yue Liu, Yinpeng Dong, and Jiaheng Zhang. Silent leaks: Implicit knowledge extraction attack on rag systems through benign queries. *arXiv preprint*, 2025.

[26] Stav Cohen, Ron Bitton, and Ben Nassi. Unleashing worms and extracting data: Escalating the outcome of attacks against rag-based inference in scale and severity using jailbreaking. *arXiv preprint arXiv:2409.08045*, 2024.

[27] Jiale Liu, Jiahao Zhang, and Suhang Wang. Exposing privacy risks in graph retrieval-augmented generation. *arXiv preprint arXiv:2508.17222*, 2025.

[28] Yuqi Zhang, Liang Ding, Lefei Zhang, and Dacheng Tao. Intention analysis makes llms a good jailbreak defender. In *Proceedings of the 31st International Conference on Computational Linguistics*, 2025.

[29] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[30] vapit. Harrypotterqa. https://huggingface.co/datasets/vapit/HarryPotterQA, 2023.

[31] Quang Tung Duong. Pokémon qa dataset. https://huggingface.co/datasets/tungdop2/pokemon, 2023.

[32] Utkarsh Sahu, Zhisheng Qi, Mahantesh Halappanavar, Nedim Lipka, Ryan A Rossi, Franck Dernoncourt, Yu Zhang, Yao Ma, and Yu Wang. Knowledge homophily in large language models. *arXiv preprint arXiv:2509.23773*, 2025.

[33] Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *ACM Computing Surveys*, 58(3):1–39, 2025.

[34] Zhehao Zhang, Ryan A Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, et al. Personalization of large language models: A survey.

[35] Moqsadur Rahman, Aaron Sanchez, Krish Piryani, Siddhartha Das, Sai Munikoti, Luis de la Torre Quintana, Monowar Hasan, Joseph Aguayo, Monika Akbar, Shahriar Hossain, et al. Generative ai for advanced cyber defense. *AI for Cybersecurity: Research and Practice*, pages 109–146, 2025.

[36] Moqsadur Rahman, Krish O Piryani, Aaron M Sanchez, Sai Munikoti, Luis De La Torre, Maxwell S Levin, Monika Akbar, Mahmud Hossain, Monowar Hasan, and Mahantesh Halappanavar. Retrieval augmented generation for robust cyber defense. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (United States), 2024.

[37] Chengke Wu, Wenjun Ding, Qisen Jin, Junjie Jiang, Rui Jiang, Qinge Xiao, Longhui Liao, and Xiao Li. Retrieval augmented generation-driven information retrieval and question answering in construction management. *Advanced Engineering Informatics*, 65:103158, 2025.

[38] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

[39] Jimeng Shi, Sizhe Zhou, Bowen Jin, Wei Hu, Shaowen Wang, Giri Narasimhan, and Jiawei Han. Hypercube-rag: Hypercube-based retrieval-augmented generation for in-domain scientific question-answering. *arXiv preprint*, 2025.

[40] Baolei Zhang, Haoran Xin, Jiatong Li, Dongzhe Zhang, Minghong Fang, Zhuqing Liu, Lihai Nie, and Zheli Liu. Benchmarking poisoning attacks against retrieval-augmented generation. *arXiv preprint arXiv:2505.18543*, 2025.

[41] Tingwei Zhang, Fnu Suya, Rishi Jha, Collin Zhang, and Vitaly Shmatikov. Adversarial hubness in multi-modal retrieval. *arXiv preprint arXiv:2412.14113*, 2024.

[42] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3827–3844, 2025.

[43] Caihua Li, In Gim, and Lin Zhong. Confidential prompting: Privacy-preserving llm inference on cloud, 2025.

[44] Srija Mukhopadhyay, Sathwik Reddy, Shruthi Muthukumar, Jisun An, and Ponnurangam Kumaraguru. Privacybench: A conversational benchmark for evaluating privacy in personalized ai. *arXiv preprint arXiv:2512.24848*, 2025.

[45] Shuai Cheng, Zhao Li, Shu Meng, Mengxia Ren, Haitao Xu, Shuai Hao, Chuan Yue, and Fan Zhang. Understanding PII Leakage in Large Language Models: A Systematic Survey. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI-25), Survey Track*, pages 10409–10416, 2025.

[46] Roberto Gonzalez, Claudio Soriente, Juan Miguel Carrascosa, Alberto Garcia-Duran, Costas Iordanou, and Mathias Niepert. User profiling by network observers. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pages 212–222, 2021.

[47] Yuhang Liang, Xinyi Li, Jie Ren, Ang Li, Bo Fang, and Jieyang Chen. Attnchecker: Highly-optimized fault tolerant attention for large language model training. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 252–266, 2025.

[48] Yi He, Mike Hutton, Steven Chan, Robert De Gruijl, Rama Govindaraju, Nishant Patil, and Yanjing Li. Understanding and mitigating hardware failures in deep learning training systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–16, 2023.

[49] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[50] Shiyue Zhang, Asli Celikyilmaz, Jianfeng Gao, and Mohit Bansal. Emailsum: Abstractive email thread summarization. *arXiv preprint arXiv:2107.14691*, 2021.

[51] Marti A Hearst. Text tiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64, 1997.
[52] Jerry Liu. LlamaIndex, 11 2022.
[53] Yingshui Tan, Yilei Jiang, Yanshi Li, Jiaheng Liu, Xingyuan Bu, Wenbo Su, Xiangyu Yue, Xiaoyong Zhu, and Bo Zheng. Equilibrate rlhf: Towards balancing helpfulness-safety trade-off in large language models. *arXiv preprint arXiv:2502.11555*, 2025.
[54] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
[55] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 2020.
[56] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint*, 2023.
[57] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint*, 2024.
[58] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.
[59] Zhenting Qi, Hanlin Zhang, Eric Xing, Sham Kakade, and Himabindu Lakkaraju. Follow my instruction and spill the beans: Scalable data extraction from retrieval-augmented generation systems. *arXiv preprint arXiv:2402.17840*, 2024.
[60] Haoyu Han, Li Ma, Yu Wang, Harry Shomer, Yongjia Lei, Zhisheng Qi, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, et al. Rag vs. graphrag: A systematic evaluation and key insights. *arXiv preprint arXiv:2502.11371*, 2025.
[61] Guangtao Zeng, Wenmian Yang, Zeqian Ju, Yue Yang, Sicheng Wang, Ruisi Zhang, Meng Zhou, Jiaqi Zeng, Xiangyu Dong, Ruoyu Zhang, et al. Meddialog: Large-scale medical dialogue datasets. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, 2020.
[62] Hung Nguyen. Vietnamese medical qa: Question answering dataset for medical in vietnamese, 2024.
[63] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint*, 2016.
[64] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
[65] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
[66] Shenglai Zeng, Jiankun Zhang, Pengfei He, Jie Ren, Tianqi Zheng, Hanqing Lu, Han Xu, Hui Liu, Yue Xing, and Jiliang Tang. Mitigating the privacy issues in retrieval-augmented generation (rag) via pure synthetic data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.
[67] Arshia Hemmat, Matin Moqadas, Ali Mamanpoosh, Amirmasoud Rismanchian, and Afsaneh Fatemi. Vague-gate: Plug-and-play local-privacy shield for retrieval-augmented generation. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 3715–3730, 2025.

A Appendix

A.1 Details of Benchmark Baselines

In this section, we comprehensively review existing extraction baselines [13, 24–26] in our benchmark. Each baseline represents a distinct extraction attack strategy for constructing the INFORMATION component $\mathcal{I}$ of the attack query to steer the retriever toward exploring different embedding regions of the knowledge base. These baselines span purely random [13, 26] and adaptive [24–26] methods, thereby covering a broad spectrum of real-world attack behaviors. For all baselines, the COMMAND C component remains fixed.

- **RandomText (R-TT)** [24] attack constructs the $\mathcal{I}^t$ component by prompting an LLM with a high temperature to produce a syntactically valid yet semantically random natural-language sentence. This allows each attack query to explore diverse regions of the retrieval embedding space without any optimization.
- **RandomToken (R-TK)** attack constructs INFORMATION at t^{th} query round $\mathcal{I}^t$ by concatenating a fixed number of tokens sampled from the vocabulary of the attack embedding model. This provides a simple baseline for embedding-level randomization.
- **RandomEmb (R-EB)** [26] attack begins by collecting a set of English embedding vectors from an external corpus (e.g., WikiText) that is disjoint from the attack-targeted knowledge, thereby preventing information leakage that could make the attack artificially easy. This collection is to estimate an embedding distribution that reflects natural linguistic structure. For the t^{th} round of attack, a target embedding vector is first sampled from this distribution as a reference. The INFORMATION $\mathcal{I}^t$ is then constructed by initializing a placeholder query and performing greedy token optimization: the algorithm iteratively replaces tokens to maximize the cosine similarity between the evolving query embedding and the sampled target embedding. This procedure enables RandomEmb to explore retrieval embedding space that aligns with natural linguistic structure.
- **Dynamic Greedy Embedding Attack (DGEA)** [26] adaptively constructs INFORMATION $\mathcal{I}^t$ to explore unexplored retrieval embedding regions. At each round, it selects a target embedding farthest from previously extracted chunks and greedily updates $\mathcal{I}^t$ to maximize the similarity between the resulting query $\boldsymbol{Q}^t$ and the target embedding. This iterative process improves retrieval coverage and diversity across the corpus.
- **CopyBreak Attack** [24] alternates between *exploration* and *exploitation* to construct $\mathcal{I}^t$. Exploration generates semantically diverse sentences to probe new embedding regions, while exploitation expands around extracted chunks by generating logically adjacent text from sentence prefixes or suffixes. Repeating these modes at a fixed frequency N enables both broad corpus coverage and local content expansion.
- **Implicit Knowledge Extraction Attack (IKEA)** [25] crafts benign-looking information-seeking queries to evade extraction-intent detection. It first generates a diverse set of domain-relevant anchors and samples one to construct each query. Based on the RAG response, unsuccessful anchors are downweighted while successful anchors and their semantic neighbors are upweighted. By iteratively exploring promising semantic regions until redundancy or blocking occurs, IKEA forms an adaptive, human-like exploration trajectory in the $\mathcal{I}$-space.

A.2 Details of Generator Extraction Metrics

Following Section 3.4, we quantify generator extraction effectiveness EE^{G} by measuring overlap between the generated output $\mathcal{A}^t$ and the retrieved content $\mathcal{R}^t$. Due to no explicit correspondence between generated responses and retrieved items, we first propose an alignment strategy to pair generated with retrieved content, and then compute similarity for each aligned pair to quantify overlap.

A.2.1 Alignment Strategies. Depending on attacks, we consider two cases when aligning retrieved with generated contents:

- **Pair-wise Alignment.** Attack methods such as DGEA and CopyBreak [24, 26] explicitly include a COMMAND for verbatim leakage. If the retriever returns k knowledge instances at round t, the generator outputs k corresponding segments, yielding k well-aligned pairs for evaluation. Metrics can therefore be applied directly on a per-pair basis $\psi_{\text{pair}}(\mathcal{A}^t, \mathcal{R}^t) = \sum_{i=1}^{|\mathcal{R}^t|} \psi_{\text{unit}}(\mathcal{A}_i^t, \mathcal{R}_i^t)$
- **Concatenated Alignment.** In contrast, IKEA [25] does not use a COMMAND component, and therefore the RAG generator produces a single paragraph-style response that blends information

across all retrieved knowledge instances, preventing a one-to-one alignment. In this case, we concatenate all retrieved knowledge instances into a single reference text and compute alignment as: $\tilde{\mathcal{R}}^t = \text{concat}(\mathcal{R}^t_1, \ldots, \mathcal{R}^t_{|\mathcal{R}^t|}), \psi_{\text{pair}}(\mathcal{A}^t, \mathcal{R}^t) = \psi_{\text{unit}}(\mathcal{A}^t, \tilde{\mathcal{R}}^t)$ If the generator refuses to answer (e.g., outputs a refusal or safety message), the corresponding alignment score is set to 0.

A.2.2 Similarity Instantiations. The unit-level alignment function ψ_{unit} can be instantiated from semantic and lexical perspectives:

- **Lexical Similarity** evaluates extraction at the lexical level. Common instantiations include Exact Match [63], BLEU [64], ROUGE-L [65], which compare the token-level overlap between generated output and the retrieved one. High lexical similarity indicates that the generator reproduced the retrieved content in a nearly verbatim manner. In this work, we use ROUGE-L following [13, 24, 25].
- **Semantic Similarity.** Semantic similarity evaluates extraction at the semantic level using embedding-based similarity measures. A common instantiation [24, 25] is cosine similarity between embeddings of the generated output and the retrieved one. High semantic similarity indicates that the generator conveys information that is close in meaning to the retrieved content.

A.3 Knowledge Base Setup

We investigate how different RAG indexing strategies affect attack performance. Specifically, we compare three representative indexings: **Instance indexing**, where each index entry corresponds to a natural data instance; **Fixed-Chunk indexing**, which segments the knowledge base into fixed-length text chunks with 20% overlap [24]; and **Graph Triplet indexing** [52], which transforms document sentences into structured triplets of entity-relation-entity.

Evaluating attacks across different indexing strategies presents non-trivial challenges. Raw item-level leakage counts are not comparable across indexings because the granularity and semantic content of stored items differ substantially. For example, Graph indexing produces many fine-grained triplets while chunk-based indexing produces fewer but more information-dense text chunks.

To address this issue, we adopt a target-oriented evaluation strategy for the numerator of our metric. Instead of counting how many indexed items are leaked, we measure how much *key private information* is extracted, formalized as $\phi\left(\cup_{t=1}^{T}\mathcal{R}^t, \mathcal{D}^*\right)$, where $\mathcal{R}^t$ represents the retrieved content at query t and $\mathcal{D}^*$ denotes the set of key private information units shared across all indexing strategies. This design enables fair comparison by anchoring evaluation to semantic targets rather than indexing artifacts.

In addition, different indexing strategies retrieve items with varying amounts of information. Text chunks may include a large number of non-informative tokens (e.g., stop words), whereas GraphRAG retrieval tends to return concise, content-dense triplets. To mitigate this discrepancy, we introduce token-length normalization for the denominator, computed as $\sum_{t=1}^{T}|\mathcal{R}^t|_{\text{token}}$, which sums the total number of tokens retrieved across all T attack queries. This normalization strategy prevents biases caused by differences in textual verbosity across indexing methods.

To conclude, our new evaluation metric is:

$$\text{EE}^{\text{R}}_{\text{token}} = \phi\left(\cup_{t=1}^{T}\mathcal{R}^t, \mathcal{D}^*\right)\left(\sum_{t=1}^{T}|\mathcal{R}^t|_{\text{token}}\right)^{-1} \tag{9}$$

Table 4: SAGE defense cost analysis per dataset, total token consumption, estimated cost, and processing time.

Dataset	Instances	Total Tokens	Total Cost	Time
HealthCareMagic	112K	856.4M	$217.5	>10d
Enron	517K	22.6B	$6,096.8	>10d
HarryPotter	26K	137.1M	$33.6	~30h
Pokémon	1.3K	6.6M	$1.6	~2h

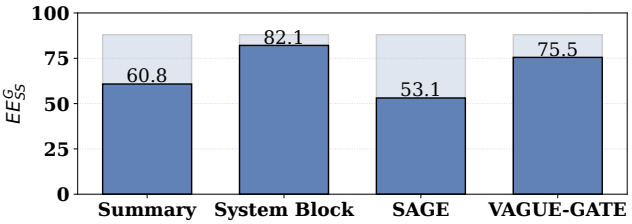


Figure 9: Average $\text{EE}^{\text{G}}_{\text{SS}}$ across HarryPotter and Pokémon for all defenses across all attacks.

A.4 Rewrite-Based Defense Analysis

Beyond the defenses evaluated in the main paper, we additionally study two representative rewrite-based defenses, SAGE [66] and VAGUE-GATE [67]. SAGE operates at indexing time by replacing the original knowledge base with a rewritten corpus that preserves the semantic content of the source documents while avoiding verbatim text reproduction. Retrieval is then performed exclusively over this rewritten corpus, preventing direct access to the original documents. In contrast, VAGUE-GATE operates at generation time by identifying sensitive content within retrieved passages and selectively paraphrasing it before the passages are provided to the generator. This design aims to reduce the risk of memorization and verbatim leakage while preserving the utility of the retrieved information. We evaluate these defenses in conjunction with Summary and System Block, reporting the average generation-stage extraction effectiveness ($\text{EE}^{\text{G}}_{\text{SS}}$) over all attacks on the Harry Potter and Pokémon datasets. Our evaluation is restricted to these datasets because SAGE necessitates offline rewriting of the complete corpus, resulting in substantial computational overhead that renders experiments on larger-scale datasets prohibitively expensive (Table 4).

Figure 9 shows that **SAGE is the most effective generation-stage defense**, reducing average $\text{EE}^{\text{G}}_{\text{SS}}$ to 53.1 and outperforming both Summary and System Block. In contrast, **VAGUE-GATE provides only moderate protection**. Since it primarily perturbs sensitive entities while preserving the surrounding context, it leaves much of the surrounding context unchanged and preserves high semantic similarity to the original content.

However, rewrite-based defenses incur substantial deployment costs. As in Table 4, applying SAGE to large corpora such as HealthCareMagic and Enron would require rewriting approximately 856M and 22B tokens, respectively, corresponding to estimated costs of $217 and $6,097 and more than ten days of processing time.

Overall, these results reveal a clear trade-off: rewrite-based defenses, especially SAGE, significantly reduce knowledge extraction attack effectiveness by modifying the corpus itself, but their high preprocessing cost limits practicality for large-scale deployments.