Token-Level Precise Attack on RAG: Searching for the Best Alternatives to Mislead Generation

Zizhong Li*, **Haopeng Zhang**†, **Jiawei Zhang***
*IFM Lab, University of California, Davis
†ALOHA Lab, University of Hawaii at Mānoa
zzoli@ucdavis.edu, haopengz@hawaii.edu, jiawei@ifmlab.org

Abstract

While large language models (LLMs) have achieved remarkable success in providing trustworthy responses for knowledge-intensive tasks, they still face critical limitations such as hallucinations and outdated knowledge. To address these issues, the retrieval-augmented generation (RAG) framework enhances LLMs with access to external knowledge via a retriever, enabling more accurate and real-time outputs about the latest events. However, this integration brings new security vulnerabilities: the risk that malicious content in the external database can be retrieved and used to manipulate model outputs. Although prior work has explored attacks on RAG systems, existing approaches either rely heavily on access to the retriever or fail to jointly consider both retrieval and generation stages, limiting their effectiveness, particularly in black-box scenarios. To overcome these limitations, we propose Token-level Precise Attack on the RAG (TPARAG), a novel framework that targets both white-box and black-box RAG systems. TPARAG leverages a lightweight white-box LLM as an attacker to generate and iteratively optimize malicious passages at the token level, ensuring both retrievability and high attack success in generation. Extensive experiments on open-domain QA datasets demonstrate that TPARAG consistently outperforms previous approaches in retrieval-stage and end-to-end attack effectiveness. These results further reveal critical vulnerabilities in RAG pipelines and offer new insights into improving their robustness.

1 Introduction

The rapid advancement of large language models (LLMs) has led to impressive performance across a broad range of NLP tasks (Ouyang et al., 2022; Chang et al., 2024; Guo et al., 2025), including but not limited to knowledge-intensive generation (Singhal et al., 2025; Wang et al., 2023; Zhang et al., 2023b,a). However, LLM-generated content still faces challenges such as hallucinations and outdated knowledge (Liu et al., 2024a; Ji et al., 2023). To address these limitations, the Retrieval-Augmented Generation (RAG) framework was introduced and has been widely adopted (Lewis et al., 2020; Jiang et al., 2023b; Chen et al., 2024a). RAG combines two core components: (1) a retriever that searches for relevant information from an external knowledge base, and (2) a reader that generates more accurate and informative responses based on the retrieved passages.

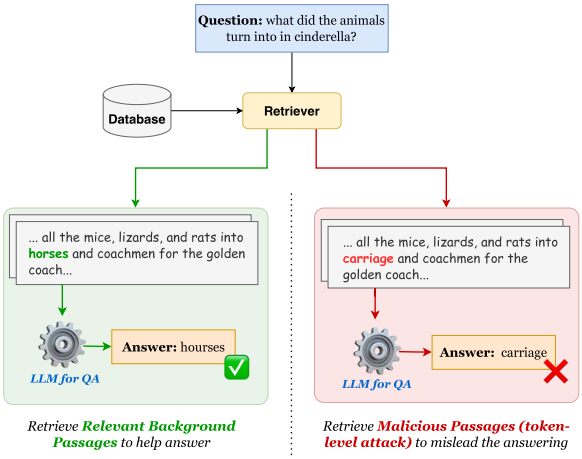

Figure 1: Comparison between the general RAG system (green background) and the RAG system attack (red background). The attacker replaces key information in the background knowledge to craft malicious passages, tricking the reader into generating an incorrect answer.

Although RAG enhances the generation quality of LLMs, it also introduces new potential risks to the reader's generation process, as the Figure 1 shows. Since the retriever gathers information from external knowledge sources, the system is vulnerable to retrieving harmful or misleading content, which can cause the reader to generate incorrect or unsafe responses (Zeng et al., 2024; Jiang et al., 2024). To exploit this weakness, recent studies have proposed attack methods that inject harmful information into the external database to manipu-

Findings of the Association for Computational Linguistics: EACL 2026, pages 3193–3206
March 24-29, 2026

late the reader's output (Zou et al., 2024; Cheng et al., 2024; Xue et al., 2024; Cho et al., 2024).

However, existing RAG attack approaches still face some of the following limitations: 1) Lack of joint consideration for the retrieval and generation. Prior studies (Zou et al., 2024; Cheng et al., 2024) often rely on teacher LLMs to generate malicious passages targeting the reader, but overlook whether the retriever can effectively retrieve these passages; 2) Overreliance on retriever models. For instance, Xue et al. (2024) depends heavily on access to the retriever to generate query-similar malicious passages, which limits applicability in realistic black-box RAG settings; and 3) Lack of targeted control over reader outputs. The crafted malicious passage by the existing approaches is relatively random and cannot *intentionally* guide the reader's responses (Cho et al., 2024; Chen et al., 2024b).

To address the limitations of existing RAG attack methods, we propose *Token-level Precise Attack on RAG (TPARAG)*, a novel framework designed to target both black-box and white-box RAG systems. TPARAG leverages a lightweight white-box LLM as the attacker to first generate malicious passages and then optimize them at the token level, ensuring that the adversarial content can be retrieved by the retriever and effectively misleads the reader. Specifically, TPARAG operates in two stages: a generation attack stage and an optimization attack stage. In the generation stage, TPARAG records the top-k most probable tokens at each position as potential substitution token candidates. These token candidates are then systematically recombined during optimization to construct a pool of malicious passage candidates. Each passage candidate is further evaluated using two criteria to exploit vulnerabilities in both the retrieval and generation processes of the RAG pipeline: (1) the likelihood that the LLM attacker generates an incorrect answer given the passage, and (2) its textual similarity to the query. The most effective malicious passage will finally be selected and injected into the external knowledge base to execute a targeted attack.

We conduct extensive experiments on multiple open-domain QA datasets using various lightweight white-box LLMs as attackers, under both black-box and white-box RAG settings. The results demonstrate that TPARAG significantly outperforms prior approaches in attacking RAG systems, both in retrieval and end-to-end performance. Moreover, we perform a series of quantitative analyses to identify key factors influencing attack success, critical vulnerabilities in RAG architectures and offering insights into improving their robustness. In summary, our contributions are as follows.

- We propose TPARAG, a token-level precise attack framework that leverages a lightweight white-box LLM as an attacker to exploit vulnerabilities in both the retrieval and generation stages of RAG systems.
- We conduct extensive experiments on multiple open-domain QA datasets, demonstrating the effectiveness of TPARAG in both black-box and white-box RAG settings.
- We perform detailed quantitative analyses to identify key factors influencing token-level attacks and explore strategies for maximizing attack effectiveness with minimal computational resources in practical scenarios.

2 Related Work

2.1 Retrieval-augmented Generation

Owing to the flexibility and effectiveness of the retriever model, it has become increasingly important to enhance LLM performance on knowledge-intensive tasks, driving the development of the RAG framework (Lewis et al., 2020; Jiang et al., 2023b; Gao et al., 2023). By incorporating information retrieval, RAG equips language models with more accurate and up-to-date background knowledge, improving generation quality and addressing common issues such as hallucination (Shuster et al., 2021; Béchard and Ayala, 2024). Leveraging these advantages, RAG has been widely adopted across various NLP tasks, including but not limited to open-domain question answering (Siriwardhana et al., 2023; Setty et al., 2024; Wiratunga et al., 2024), summarization (Liu et al., 2024b; Suresh et al., 2024; Edge et al., 2024), and dialogue systems (Vakayil et al., 2024; Wang et al., 2024b; Kulkarni et al., 2024).

2.2 Risks in the RAG System

While RAG generally produces more reliable and accurate outputs than only using LLMs, its reliability remains subject to critical vulnerabilities. Prior work (Zhou et al., 2024; Ni et al., 2025; Zeng et al., 2024) has shown that the data leakage from the external retrieval database can significantly affect RAG's reliability, primarily through two mechanisms: (1) the accurate extraction of sensitive or

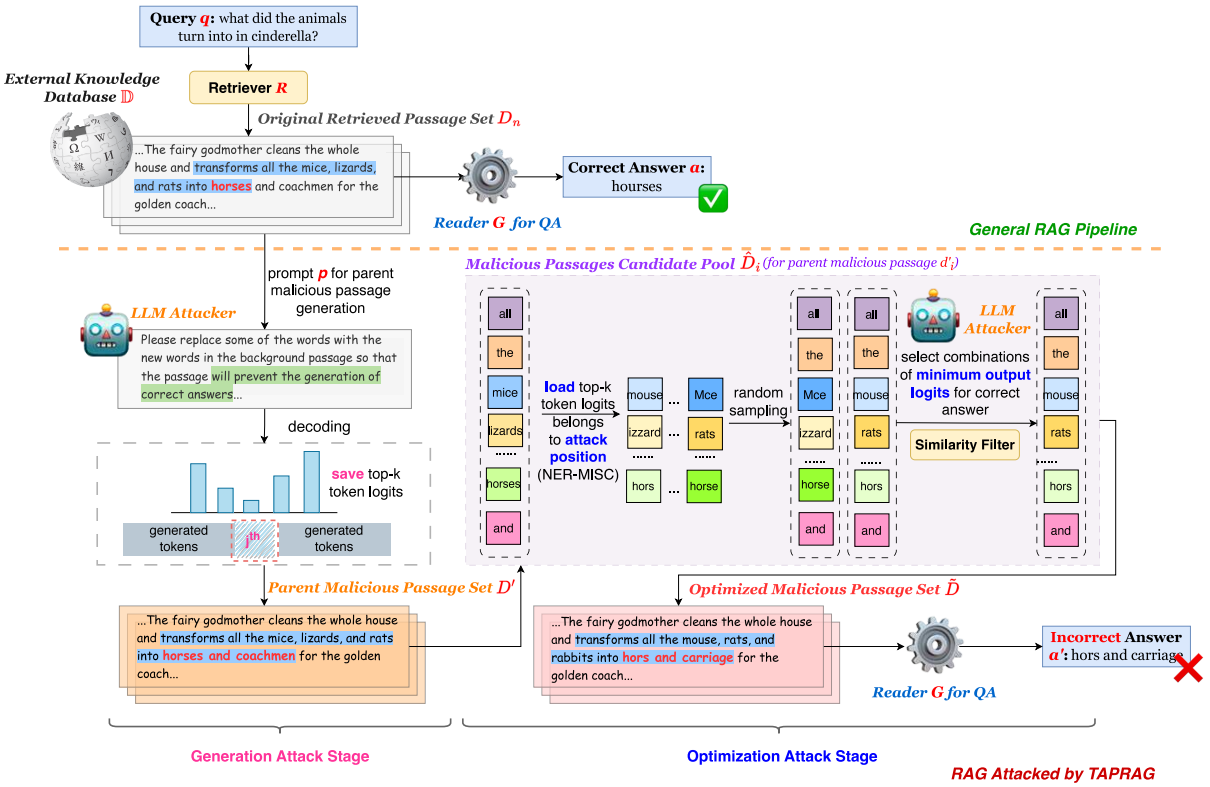

Figure 2: The framework of our proposed TPARAG. TPARAG first generates parent malicious passages through the generation attack stage (left). These passages are then recombined and refined during the optimization attack stage (right), producing optimized malicious passages that effectively mislead RAG's answer.

harmful information by the retriever, and (2) the generation of responses that expose such retrieved content. As a result, the external dataset becomes a central point of attack.

Building on this vulnerability, previous RAG attack studies have focused on *knowledge corruption*: injecting malicious content into the external database to mislead the readers' response (Wang et al., 2024a; RoyChowdhury et al., 2024; Zou et al., 2024; Cheng et al., 2024). For example, Zou et al. (2024) and Cheng et al. (2024) use a teacher LLM to craft query-specific malicious content that guides the reader to generate incorrect outputs. However, these approaches often overlook whether the malicious content can actually be retrieved, posing a challenge in real-world scenarios where successful retrieval depends on sufficient similarity to the query. To improve the recall rate of the crafted malicious content, later studies have introduced optimization techniques that align the malicious content with query semantics using embedding similarity scores provided by the retriever (Xue et al., 2024; Cho et al., 2024; Jiang et al., 2024). However, these approaches heavily rely on the retriever's parameters, limiting their applicability in black-box RAG systems. Moreover, the optimization process often introduces excessive randomness, reducing control over the reader's final output. To address these limitations, we propose TPARAG, a token-level precise attack framework designed explicitly for black-box RAG settings. Additionally, TPARAG balances the likelihood of being retrieved with the ability to induce specific, incorrect responses from the reader.

3 Proposed Method

3.1 Problem Formulation

The Pipeline of RAG. A typical RAG system consists of two main components: a retriever model R (parameterized with θ_R) that retrieves relevant information from an external database, and a reader G (parameterized with θ_G) that generates responses based on the retrieved content. Specifically, given a query q, the goal of the retriever model R is to find a subset of the most relevant passages $D = \{d_1, d_2, ..., d_m\}$ from the knowledge database $\mathbb{D}$, where each d_i represents a unique passage. Then, this subset D is combined with the query q to form the prompt input of the reader G, and then generates the corresponding answer a as follows:

$$a = G(q, D; \theta_G), \tag{1}$$

RAG Attack's Objective. Given a query q, the objective of the RAG attack framework is to construct a malicious passage subset $\tilde{D} = \{\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_m\}$ and inject it into the knowledge database $\mathbb{D}$. The goal is for the retriever R to select $\tilde{d}_i \in \tilde{D}$ and forward it to the reader G, thereby inducing the generation for an incorrect response a':

$$a' = G(q, \tilde{D}; \theta_G), \tag{2}$$

The construction of each $\tilde{d}_i \in \tilde{D}$ focuses on two key objectives: 1) the retrieval attack, and 2) the generation attack. The first one aims to prioritize the retrieval of $\tilde{d}_i \in \tilde{D}$ over $d_i \in D$. That is, maximizing the textual similarity between the given query q and each malicious passage $\tilde{d}_i$:

$$\max s(q, \tilde{d}_i; \theta_R) \quad \text{s.t.} \frac{s(q, \tilde{d}_i; \theta_R)}{s(q, d_i; \theta_R)} > 1, \tag{3}$$

where $s(\cdot; \theta_R)$ denotes the cosine similarity between the embedding of the query q and the malicious passage $\tilde{d}_i$ encoded by the retriever R.

For the generation attack, the objective is to reduce the likelihood that incorporating $\tilde{d}_i$ into the reader leads to the correct answer a, compared to d_i, and seeks to minimize this likelihood to the greatest extent possible:

$$\min P_G(a|q, \tilde{d}_i; \theta_G) \quad \text{s.t.} \frac{P_G(a|q, \tilde{d}_i; \theta_G)}{P_G(a|q, d_i; \theta_G)} < 1, \tag{4}$$

where $P_G(a|\cdot; \theta_G)$ denotes the likelihood that the reader G generates the correct answer a.

3.2 Token-level Precise Attack on RAG

To achieve a dynamic balance between Equation (3) and (4), we design TPARAG, which leverages lightweight white-box LLMs to generate and then optimize the malicious passage subset $\tilde{D}$ at the token-level, enabling precise, query-specific attacks on the RAG systems.

As shown in Figure 2, given a query q, TPARAG first uses a white-box LLM attacker LLM_{attack} to fabricate an incorrect answer and generate a corresponding malicious background passage $\tilde{d}_i$ based on d_i, while maintaining high textual similarity with q. During this generating process, the attacker model LLM_{attack} records the top-k most probable generated tokens at each position. These tokens are then recombined in the optimization to form new malicious candidate passages at key positions by modifying key token positions. Then, the candidates are further filtered based on 1) their likelihood of inducing incorrect answers; and 2) their textual similarity to the query q. The final optimized malicious passage set $\tilde{D}$ is then selected from this refined pool. The following subsections detail each stage of the TPARAG pipeline, and we also provide the detailed algorithm in Appendix A.

3.2.1 Initialization

Threshold for Generation Attack. TPARAG begins with a data initialization step. Given a query q and a set of m originally retrieved relevant background passages $D = \{d_1, d_2, ..., d_m\}$, TPARAG establishes a threshold l that filters out less effective malicious passages, retaining only those more likely to mislead the reader:

$$l = \max_{l_i \in L} l_i, \tag{5}$$

where $L = \{l_1, l_2, ..., l_m\}$ is the generation logits for the correct answer a using the output of the white-box LLM attacker LLM_{attack}.

Threshold for Retrieval Attack. TPARAG computes textual similarity scores $S = \{s_1, s_2, ..., s_m\}$ between the query q and each passage d_i, using them to define a similarity threshold s:

$$s = \min_{s_i \in S} s_i, \tag{6}$$

This threshold helps identify malicious passages that are more likely to be retrieved by the retriever.

Entity-Based Attack Localization. TPARAG employs a named entity recognition (NER) tool (Akbik et al., 2019) as the attack locator Loc to annotate the correct answer a with entity labels:

$$pos_{attack} = Loc(a), \tag{7}$$

which helps to identify the specific token types to be targeted in its optimization attack stages.

3.2.2 Generation Attack Stage

After the initialization, TPARAG uses the white-box LLM_{attack} to generate a set of parent malicious passages subset $D' = \{d'_1, d'_2, ..., d'_m\}$, based on the query q and each background passage in D, following a predefined prompt p as Figure 3 shows. Therefore, the attacker generates parent malicious passages as:

$$d'_i = LLM_{attack}(q, d_i; \theta_{LLM_{attack}}), \tag{8}$$

During the decoding process, the LLM attacker records, at each token position j, the top-k tokens

I will provide one query with one corresponding background to answer the query. Please replace some of the words with new words in the background passage so that the passage will prevent the generation of the correct answers while maintaining maximum similarity with the given background passage.
<Query>: [query q]
<Passage>: [passage d_i]

Figure 3: An example of the prompt for TPARAG's generation attack stage.

T_{ij} with the highest generation probabilities:

$$T_{ij} = \text{Top-}k(P_{LLM_{attack}}(t_{1:j-1}, p)), \quad (9)$$

where $t_{1:j-1}$ denotes the partially generated sequence up to position $j-1$ and p denotes the input prompt. These top-k tokens are later used in the optimization attack stage to further refine the parent malicious passages.

3.2.3 Optimization Attack Stage

Entity-Based Token Filtering. In this stage, TPARAG first reuses the attack locator Loc to perform NER on each token position j in d'_i, identifying tokens that share the same entity type as the target position pos_{attack}. These tokens are then used as candidate substitution points for the token-level attack.

Token Substitution and Candidate Generation. For each parent malicious passage $d'_i \in D'$, TPARAG randomly selects the tokens that share the same entity type as the target position pos_{attack}, based on a predefined maximum token substitution rate pr_{sub} (i.e., the upper bound on the proportion of tokens to be replaced). For each position j selected for substitution, TPARAG generates multiple variants by replacing the token with alternatives from the previously recorded top-k token set T_{ij}. This process produces a set of optimized malicious passage candidates for each d'_i, denoted as:

$$\hat{D}_i = \{\hat{d}_{i1}, \hat{d}_{i2}, ..., \hat{d}_{in}\}, \quad (10)$$

where n is the number of generated candidates.

Similarity-Based Filtering. For each candidate $\hat{d}_{ij} \in \hat{D}_i$, TPARAG computes its textual similarity to the query q and compares it with the predefined similarity threshold s. Candidates with similarity below the threshold are discarded.

TPARAG supports attacks under both white-box and black-box RAG settings. In the white-box setting, the similarity between each candidate passage and the query is computed using embeddings from the original retriever. In the black-box setting, where access to the retriever is unavailable, TPARAG uses Sentence-BERT (Reimers and Gurevych, 2019) as a substitute to estimate the textual similarity, which also proves to be highly effective in the attack process.

Likelihood-Based Selection. For the remaining passage candidates generated from d'_i, the LLM attacker LLM_{attack} simulates the generation process of the RAG system by computing the likelihood $\hat{l}_{ik}$ of generating the correct answer a, given the query q and candidate passage $\hat{d}_{ik}$:

$$\hat{l}_{ik} = P_{LLM_{attack}}(a|q, \hat{d}_{ik}; \theta_{LLM_{attack}}). \quad (11)$$

Among all candidates in $\hat{D}_i$, the one with the lowest $\hat{l}_{ik}$ is selected as the final optimized malicious passage $\tilde{d}_i$.

Moreover, TPARAG performs iterative optimization over multiple rounds of malicious passage generation. In each iteration, it applies this greedy search strategy to select the candidate with the strongest attack effect like $\tilde{d}_i$, which is then used as input for the next round.

Following this strategy, the resulting set of optimized malicious passages $\tilde{D}$ maintains high textual similarity with the query q, while introducing precise and effective interference to disrupt the generation of the correct answer.

4 Experiment

4.1 Experiment Setup

Dataset. We conduct experiments on three open-domain question-answering benchmark datasets: NaturalQuestions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2022). For the external knowledge database, we use Wikipedia data dated December 20, 2018, adapting the passage embeddings provided by ATLAS (Izacard et al., 2023), and the poison ratio (i.e., the proportion of malicious passages within the retrieval corpus) is approximately 2e-5. We randomly sample 100 query instances from each dataset's training set as the targeted queries for the TPARAG attack.

Evaluation Metrics. Following the previous work (Cho et al., 2024), we decompose ASR into three components: $ASR_R(\%)$, $ASR_L(\%)$, and $ASR_T(\%)$, which denote the attack success percentage of the retrieval, the attack success percentage of the generation, and the overall attack success percentage, respectively. Specifically, ASR_R measures the proportion of the crafted malicious pas-

RAG Setting	LLM Attacker	NQ					TriviaQA					PopQA				
		$ASR_R\uparrow$	$ASR_L\uparrow$	$ASR_T\uparrow$	EM↓	F1↓	$ASR_R\uparrow$	$ASR_L\uparrow$	$ASR_T\uparrow$	EM↓	F1↓	$ASR_R\uparrow$	$ASR_L\uparrow$	$ASR_T\uparrow$	EM↓	F1↓
Black-box (TPARAG)	Qwen2.5-3B	77.2	99.6	77.2	68.0	76.2	90.1	74.2	67.1	78.0	80.8	84.6	88.6	75.6	48.0	50.1
	Qwen2.5-7B	79.0	**100.0**	**79.0**	65.0	73.2	85.6	77.2	65.1	**56.0**	**60.4**	85.8	85.8	73.4	28.0	34.8
	Mistral-7B	76.7	**100.0**	76.7	**45.0**	**55.2**	**94.8**	67.4	64.5	58.0	61.5	92.2	87.2	81.2	**7.0**	**9.5**
	Gemma2-9B	66.4	98.0	65.6	55.0	62.9	73.4	**98.6**	**72.6**	68.0	72.9	76.4	91.8	69.8	32.0	34.6
Black-box (PoisonedRAG)	Qwen2.5-3B	48.0	46.2	39.8	66.0	57.0	92.8	64.0	59.5	77.0	79.2	96.2	77.1	73.8	42.0	45.3
	Qwen2.5-7B	39.2	54.0	19.8	74.0	80.7	22.0	59.6	13.9	91.0	93.0	78.8	66.3	53.7	36.0	39.4
	Mistral-7B	**81.5**	72.8	57.6	72.0	76.4	91.3	76.1	70.7	77.0	82.2	94.8	73.2	68.0	38.0	43.3
	Gemma2-9B	67.6	86.2	59.0	50.0	56.1	90.6	70.0	64.6	69.0	73.2	**97.8**	**92.9**	**90.7**	24.0	28.9
White-box (TPARAG)	Qwen2.5-3B	83.6	87.2	72.4	59.0	65.7	99.8	70.3	70.2	64.0	68.6	99.6	89.6	89.2	39.0	45.0
	Qwen2.5-7B	99.4	99.4	99.2	71.0	75.6	**100.0**	**85.6**	**85.6**	60.0	64.2	99.8	84.6	84.4	15.0	17.5
	Mistral-7B	99.4	99.2	98.6	52.0	**61.7**	**100.0**	69.1	69.1	**52.0**	**57.9**	**100.0**	85.0	85.0	**9.0**	**10.3**
	Gemma2-9B	**100.0**	**99.8**	**99.8**	**50.0**	62.7	**100.0**	74.9	74.9	77.0	80.9	**100.0**	**91.2**	**91.2**	33.0	33.5
White-box (GARAG)*	Mistral-7B	87.5	85.5	73.3	63.9	–	88.8	86.4	75.2	66.2	–	–	–	–	–	–
White-box (PoisonedRAG)		**100.0**	89.8	89.8	83.0	87.9	**100.0**	66.3	66.3	94.0	96.9	**100.0**	71.8	71.8	87.0	93.1
w/o RAG		–	–	–	67.0	76.4	–	–	–	94.0	95.5	–	–	–	70.0	74.0
RAG		–	–	–	82.0	88.0	–	–	–	97.0	98.5	–	–	–	92.0	96.3

Table 1: Comparison of LLM attackers with different RAG settings across three QA datasets.

Attack Setting	LLM Attacker	Evaluation Metrics				
		$ASR_R\uparrow$	$ASR_L\uparrow$	$ASR_T\uparrow$	EM↓	F1↓
Initialization	Qwen2.5-3B	77.2	99.6	77.2	68.0	76.2
	Qwen2.5-7B	71.8	100.0	71.8	66.0	74.1
	Mistral-7B	76.7	100.0	76.7	45.0	55.2
	Gemma2-9B	66.4	98.0	65.6	55.0	62.9
w/o Initialization	Qwen2.5-3B	74.0	99.6	73.6	79.0	83.6
	Qwen2.5-7B	74.8	100.0	74.8	63.0	66.7
	Mistral-7B	70.2	99.0	69.6	68.0	73.2
	Gemma2-9B	65.7	99.6	65.7	52.0	59.0

Table 2: TPARAG's performance on the NQ dataset under black-box RAG setting without initialization from the original relevant background passages.

sages satisfying Equation (3), ASR_L measures the proportion of malicious passages satisfying Equation (4), and ASR_T measures the proportion of malicious passages satisfying both conditions.

Meanwhile, we report the standard Exact Match (EM) and F1-Score, which show the accuracy and precision of the generated responses, to evaluate the end-to-end attack performance.

Models. For the RAG system, we choose the closed-source GPT-4o (OpenAI, 2023) as its reader, and the off-the-shelf Contriever (Izacard et al., 2021) as the retriever model. During the TPARAG attack, we leverage various lightweight white-box LLMs, including Qwen2.5-3B, Qwen2.5-7B (Bai et al., 2023), Mistral-7B (Jiang et al., 2023a), and Gemma2-9B (Team et al., 2024), as the LLM attackers. We maintain the same LLM attacker in the generation and optimization stage.

Baselines. To ensure a fair and reproducible comparison, we choose two representative prior approaches as our experimental baseline – PoisonedRAG (Zou et al., 2024) and GARAG (Cho et al., 2024). In the white-box setting, we replicate the PoisonedRAG setup using Contriever (Izacard et al., 2021) as the publicly available retriever. In addition, we conduct baseline evaluations using the original RAG system and a reader-only QA setup (i.e., w/o RAG), which serve as reference points for comparison.

Implementation Details. Considering the trade-off between computational cost and performance improvement, we set the maximum iteration number N_{iter} to 5, the candidate malicious passage subset size to 20, and the maximum token substitution rate as 0.2. Additionally, we restrict the size of the relevant document subset D to 5, and each retrieved passage has a maximum length of 128. We will further discuss the impact of different parameters on the attack outcomes in Section 5.

4.2 Experimental Results

Table 1[1] presents the experimental results on selected attacked query instances from three datasets, comparing our proposed TPARAG framework with other baselines. For TPARAG, we report the end-to-end performance corresponding to the iteration with the highest ASR_T. The results show that **TPARAG effectively performs end-to-end attacks while simultaneously increasing the likelihood of malicious passages retrieval across both black-box and white-box RAG settings**.

Specifically, in the black-box setting, TPARAG achieves a retrieval attack success rate exceeding 66%, reaching up to 94% in the best case. In the white-box setting, its performance improves further, reaching a 100% success rate in half of the test cases and maintaining a minimum of 83%, demonstrating TPARAG's effectiveness in misleading the retriever regardless of system access level. Meanwhile, TPARAG's optimized malicious passages

[1]Best performance values are highlighted in bold. Results marked with * are from the original paper; others are tested with our implementation and datasets.

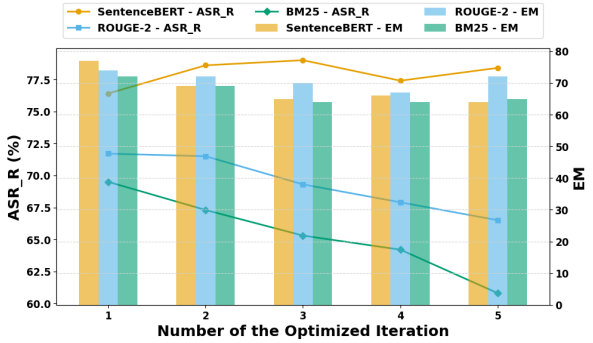

Figure 4: The impact of different similarity filters on TPARAG performance under black-box RAG setting.

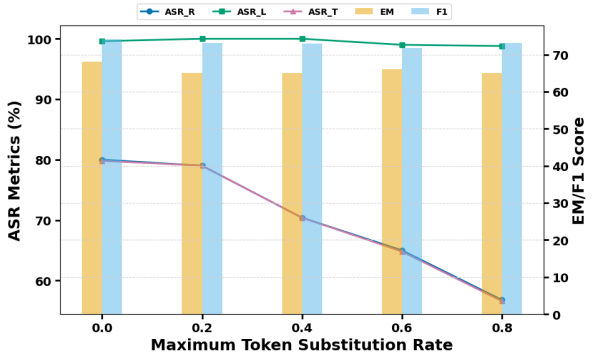

Figure 5: The performance of TPARAG under different maximum token substitution rates.

remain highly effective during the generation stage. In the black-box setting, it achieves at least a 67% attack success rate against the LLM attacker, showing its capacity to disrupt answer generation without internal model access. In the white-box setting, the minimum end-to-end attack success rate rises to 70%, indicating that TPARAG can more precisely exploit model vulnerabilities when partial access is available. These results confirm that **TPARAG's token-level passage construction is effective for retrieval and misleading answer generation**.

Regarding the end-to-end attack performance, **TPARAG consistently reduces the answer accuracy of the RAG systems** under both black-box and white-box settings. In all cases, accuracy drops below that of the original RAG baseline, and in most instances, it even falls below the performance of using the reader alone without any retrieved context. However, we observe that **the LLM attacker's success rate (ASR_L) does not always align with the end-to-end attack effectiveness**. Two factors may cause this discrepancy: 1) limitations of ASR_L: while it reflects whether the attack successfully reduces the likelihood of generating the correct answer using malicious passages, it does not capture the *degree* of reduction or its actual impact on final answer quality; 2) model discrepancy: behavioral and sensitivity differences between the white-box LLM attacker and the black-box reader may lead to misalignment during generation.

Compared to the PoisonedRAG and GARAG baselines, TPARAG offers more robust and consistent performance. Under the white-box RAG setting, TPARAG outperforms GARAG in both retrieval and end-to-end attack effectiveness when using the same LLM attacker. In addition, although PoisonedRAG achieves perfect ASR_R through gradient-based optimization in the white-box setting, it is considerably less effective in attacking the generation stage. In the black-box RAG setting, PoisonedRAG also exhibits unstable performance (i.e., ASR_L varies significantly across different LLM attackers), and generally underperforms TPARAG in most end-to-end evaluations.

5 Analysis

5.1 Strategy for Constructing Malicious Passages in Black-box Generation Attack

In the black-box RAG setting, TPARAG initializes malicious passage from the original background passages D, which still introduces some dependency on the retriever. To further evaluate its robustness, we adopt a *fully black-box* setting, where the generation attack stage is performed without any initialization from D. Instead, the LLM attacker directly generates malicious passages based solely on the query, followed by token-level optimization using the TPARAG framework.

As shown in the Table 2, initializing with the original background passages generally leads to better retrieval-stage performance (i.e., higher ASR_R), suggesting that mimicking original content helps the malicious passages better deceive the retriever. However, even without such initialization, TPARAG still achieves strong end-to-end attack performance: both EM and F1-Score drop significantly compared to the RAG baseline, demonstrating the feasibility and effectiveness of TPARAG in a fully black-box RAG setting.

5.2 Different Similarity Signal in Black-box Optimization Attack

As described in Section 3, TPARAG employs Sentence-BERT as a substitute retriever to perform similarity-based filtering when attacking black-box RAG systems. In this subsection, we evaluate the effectiveness of Sentence-BERT as an alternative similarity signal by comparing it with two widely

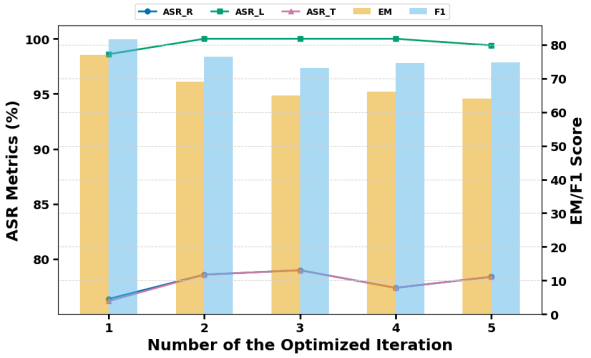

Figure 6: The performance variation of TPARAG across different numbers of optimization iterations.

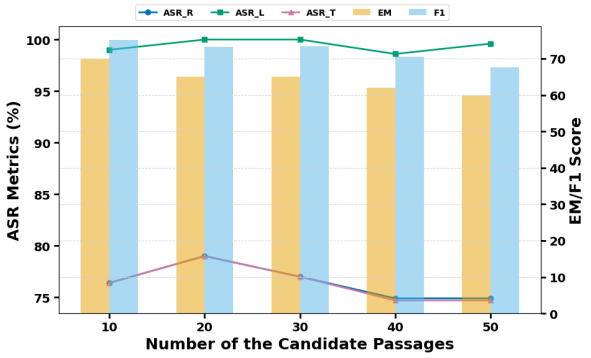

Figure 7: The performance of TPARAG under different sizes of malicious passage candidate sets.

used text similarity metrics: ROUGE-2 (Lin, 2004) and BM25 (Robertson et al., 2009), while keeping all other settings unchanged. As shown in Figure 4, Sentence-BERT best approximates the behavior of BERT-based retrievers, allowing TPARAG to iteratively optimize malicious passages and achieve a steady increase in ASR_R across optimization steps while maintaining strong end-to-end attack performance. In contrast, ROUGE-2 and BM25 fail to improve retrieval attack effectiveness, and using ROUGE-2 leads to inconsistent attack success, highlighting its limitation as a similarity proxy.

5.3 Impact of the Hyperparameter

In the following analysis, we examine the impact of TPARAG's key parameters on its performance. All experiments are conducted on the NQ dataset using Qwen-7B as the LLM attacker under black-box settings, with training configurations identical to those in Section 4 except for the varied hyperparameter.

Maximum Token Substitution Rate. To evaluate the impact of the maximum token substitution rate pr_{sub} on TPARAG's performance, we vary pr_{sub} (0.0-0.8) under the black-box RAG while keeping other settings fixed. As shown in Figure 5, higher substitution rates increase flexibility in modifying malicious passages but also cause greater semantic drift, which in turn reduces their likelihood of being retrieved. Meanwhile, this change also leads to lower EM scores, indicating that higher token-level variability more effectively disrupts the RAG system's answer generation.

Optimized Iteration. We further examine how the number of optimization iterations affects TPARAG's effectiveness. Figure 6 presents the performance trends across the iterations from 1^{st} to 5^{th} under the black-box RAG setting, following the setup in Section 4. The results show that the ASR_R increases initially and peaks at the 3^{rd} iteration. Similarly, the ASR_L exhibits an initial upward trend and stabilizes after the 3^{rd} iteration, indicating diminishing returns from further iterations. For the end-to-end attack performance, we observe a steady decline in answer accuracy with more iterations, indicating a consistent degradation of the RAG system's ability to generate correct answers as malicious passages become more refined.

The Number of Malicious Candidate Passages. In TPARAG's optimization attack stage, the size of the candidate set $\hat{D}_i$ may also impact the attack effectiveness. Here, we experiment with candidate sets of size 10, 20, 30, 40, and 50 under the black-box settings, while keeping all other settings fixed. As the experimental results show in Figure 7, TPARAG achieves better end-to-end attack performance (i.e., lower EM and F1-Score) when using larger candidate sets. However, the results also reveal a decline in retrieval attack effectiveness as the candidate set size increases, with ASR_R gradually decreasing beyond a size of 30. This trade-off is likely due to TPARAG's optimization strategy, which emphasizes misleading the reader (i.e., minimizing the likelihood of generating the correct answer) while treating textual similarity (and thus retrievability) as a secondary evaluation factor.

5.4 Defense Strategy

Following prior work (Zou et al., 2024), we evaluate TPARAG against two widely used defense strategies: perplexity-based detection and knowledge expansion.

Perplexity-based Detection. Perplexity is a widely used metric for assessing text quality, where higher perplexity often indicates a higher likelihood of malicious passage. Here, we use Mistral-7B as the LLM attacker and present the *proportion of mali-*

cious passages with perplexity scores lower than the mean of the original passages. Table 3 shows that different datasets exhibit varying sensitivity to perplexity-based detection. For NQ and TriviaQA, the malicious passages are relatively difficult to detect (i.e., about half of the malicious passages still pass the detection). In contrast, the malicious passages generated for PopQA can be more effectively detected, as only around 20% of them fall below the mean perplexity value of the original passages.

Knowledge Expansion. Assuming the attack injects at most N texts into the knowledge database, the knowledge expansion is configured to retrieve k texts ($k > N$), then $k - N$ texts would be clean ones. This strategy mitigates the impact of malicious passages by expanding the set of relevant background passages, thereby providing the reader with more reliable information. Here, we still inject $N = 5$ malicious passages but retrieve $k = 15$ passages (instead of $k = 5$ conducted in Section 4), and use Qwen-7B as the LLM attacker. Table 4 presents the results of knowledge expansion defense, which shows that applying knowledge extension provides partial mitigation against TPARAG (i.e., EM improves after defense). However, the malicious passages continue to mislead the reader model, as the defended answers still do not match the performance of the original RAG baseline.

Dataset	Black-box	White-box
NQ	0.400	0.434
TriviaQA	0.483	0.525
PopQA	0.182	0.228

Table 3: The proportion of malicious passages that pass perplexity-based detection.

Dataset	k=15 (Black-box)	k=15 (White-box)
NQ	74.0	75.0
TriviaQA	89.0	90.0
PopQA	34.0	42.0

Table 4: QA performance (EM value) under black-box and white-box settings with knowledge expansion.

5.5 Retriever Analysis

Beyond the main experiments using Contriever (Izacard et al., 2021) in Section 4, we further evaluate the generalizability of TPARAG by incorporating two additional retrievers: a fine-tuned Contriever on the MSMARCO dataset (Bajaj et al., 2016) and ANCE (Xiong et al., 2020). Using Qwen-7B as the LLM attacker, we conduct experiments on the TriviaQA and PopQA datasets, respectively. As shown in Table 5, the effectiveness of TPARAG remains clear when integrated with different retriever models, which underscores the robust generalizability of our proposed framework.

Retriever	Dataset	Setting	ASR_R	ASR_L	ASR_T	EM	F1
Contriever (msmarco)	TriviaQA	Black-box	97.4	84.4	82.4	83.0	86.2
		White-box	100.0	77.6	77.6	83.0	84.9
ANCE	PopQA	Black-box	98.2	82.0	80.6	33.0	37.9
		White-box	100.0	86.8	86.8	39.0	43.7

Table 5: Performance of TPARAG across two other retriever models.

5.6 Multi-hop QA Task

Moving beyond evaluating on the *single-hop* QA dataset instances in our main experiments in Section 4, we investigate TPARAG's performance on *multi-hop* QA using the HotpotQA dataset (Yang et al., 2018). We use Mistral-7B as the LLM attacker while keeping other hyperparameters unchanged, and the results are shown in Table 6. The results show that TPARAG continues to induce strong misleading effects on the RAG system, even when challenged with multi-hop reasoning. The fact that TPARAG outperforms other baselines across various settings provides evidence of its broad generalizability and practical effectiveness (see Appendix B for more details).

Method	Setting	ASR_R	ASR_L	ASR_T	EM	F1
TPARAG	Black-box	81.4	90.3	75.6	28.0	30.4
	White-box	100.0	90.1	90.1	33.0	35.2
PoisonedRAG	Black-box	98.4	57.4	57.4	33.0	35.6
	White-box	100.0	84.0	84.0	52.0	53.6
w/o RAG	-	-	-	-	50.0	54.5
RAG	-	-	-	-	53.0	56.3

Table 6: Performance comparison of TPARAG and baseline methods on the HotpotQA dataset.

6 Conclusion

We propose TPARAG, a novel framework that performs token-level precise attacks on RAG systems using a lightweight white-box LLM. The framework is effective in both white and fully black-box RAG settings. Moreover, we conduct extensive experiments with various LLM attackers across different datasets, validating the effectiveness of our proposed method.

7 Limitation

While TPARAG demonstrates strong attack performance on both black-box and white-box RAG systems, there remains room for further improvement. First, the current experiments are limited by the choice of retriever, as only Contriever is used for both black-box and white-box RAG settings. Future work could extend the evaluation to a broader range of retrievers with diverse training objectives and architectures. Second, due to computational constraints, our main experiments are conducted on datasets with hundreds of instances. Scaling the evaluation to larger datasets (e.g., thousands of queries) would further validate the robustness and generalizability of TPARAG.

Acknowledgement

This work is partially supported by NSF through grant IIS-2106972, awards 2201428, 2004014, and 2138259, as well as by computing resources provided through the NSF ACCESS projects CIS250940 and CIS24061. We thank the anonymous reviewers for their valuable reviews and helpful feedback.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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

Patrice Béchard and Orlando Marquez Ayala. 2024. Reducing hallucination in structured outputs via retrieval-augmented generation. *arXiv preprint arXiv:2404.08189*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024a. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Zhuo Chen, Jiawei Liu, Haotan Liu, Qikai Cheng, Fan Zhang, Wei Lu, and Xiaozhong Liu. 2024b. Black-box opinion manipulation attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2407.13757*.

Pengzhou Cheng, Yidong Ding, Tianjie Ju, Zongru Wu, Wei Du, Ping Yi, Zhuosheng Zhang, and Gongshen Liu. 2024. Trojanrag: Retrieval-augmented generation can be backdoor driver in large language models. *arXiv preprint arXiv:2405.13401*.

Sukmin Cho, Soyeong Jeong, Jeongyeon Seo, Taeho Hwang, and Jong C Park. 2024. Typos that broke the rag's back: Genetic attack on rag pipeline by simulating documents in the wild via low-level perturbations. *arXiv preprint arXiv:2404.13948*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. 2024. Rag-thief: Scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks. *arXiv preprint arXiv:2411.14110*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2024. Longform: Effective instruction tuning with reverse instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7056–7078.

Mandar Kulkarni, Praveen Tangarajan, Kyung Kim, and Anusua Trivedi. 2024. Reinforcement learning for optimizing rag for domain chatbots. *arXiv preprint arXiv:2401.06800*.

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

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

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Hanchao Liu, Wenyuan Xue, Yifei Chen, Dapeng Chen, Xiutian Zhao, Ke Wang, Liping Hou, Rongjun Li, and Wei Peng. 2024a. A survey on hallucination in large vision-language models. *arXiv preprint arXiv:2402.00253*.

Shengjie Liu, Jing Wu, Jingyuan Bao, Wenyi Wang, Naira Hovakimyan, and Christopher G Healey. 2024b. Towards a robust retrieval-based summarization system. *arXiv preprint arXiv:2403.19889*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, and 1 others. 2025. Towards trustworthy retrieval augmented generation for large language models: A survey. *arXiv preprint arXiv:2502.06872*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Ayush RoyChowdhury, Mulong Luo, Prateek Sahu, Sarbartha Banerjee, and Mohit Tiwari. 2024. Confusedpilot: Confused deputy risks in rag-based llms. *arXiv preprint arXiv:2408.04870*.

Spurthi Setty, Harsh Thakkar, Alyssa Lee, Eden Chung, and Natan Vidra. 2024. Improving retrieval for rag based question answering models on financial documents. *arXiv preprint arXiv:2404.07221*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, and 1 others. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, pages 1–8.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Karthik Suresh, Neeltje Kackar, Luke Schleck, and Cristiano Fanelli. 2024. Towards a rag-based summarization agent for the electron-ion collider. *arXiv preprint arXiv:2403.15729*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Sonia Vakayil, D Sujitha Juliet, Sunil Vakayil, and 1 others. 2024. Rag-based llm chatbot using llama-2. In *2024 7th International Conference on Devices, Circuits and Systems (ICDCS)*, pages 1–5. IEEE.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö Arık. 2024a. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*.

Hongru Wang, Wenyu Huang, Yang Deng, Rui Wang, Zezhong Wang, Yufei Wang, Fei Mi, Jeff Z Pan, and Kam-Fai Wong. 2024b. Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems. *arXiv preprint arXiv:2401.13256*.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Nirmalie Wiratunga, Ramitha Abeyratne, Lasal Jayawardena, Kyle Martin, Stewart Massie, Ikechukwu Nkisi-Orji, Ruvan Weerasinghe, Anne Liret, and Bruno Fleisch. 2024. Cbr-rag: case-based reasoning for retrieval augmented generation in llms for legal question answering. In *International Conference on Case-Based Reasoning*, pages 445–460. Springer.

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

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, and 1 others. 2024. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023a. Extractive summarization via chatgpt for faithful summary generation. *Preprint*, arXiv:2304.04193.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023b. Summit: Iterative text summarization via chatgpt. *Preprint*, arXiv:2305.14835.

Yujia Zhou, Yan Liu, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Zheng Liu, Chaozhuo Li, Zhicheng Dou, Tsung-Yi Ho, and Philip S Yu. 2024. Trustworthiness in retrieval-augmented generation systems: A survey. *arXiv preprint arXiv:2409.10102*.

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

A TPARAG Algorithm

Algorithm 1 illustrates the processing details of TPARAG's two-stage attack, where the malicious passages are iteratively optimized through the generation and optimization attack stages, starting from the initialization.

B Additional Experiments

In addition to the three commonly used QA benchmark datasets we utilized in Section 4, we extend our evaluation to HotpotQA (Yang et al., 2018) for multi-hop reasoning and LongForm-C (Köksal et al., 2024) for the instruction-based text generation task. The experimental results, as shown in Table 7 [2], demonstrate that the TPARAG framework's superior performance across diverse tasks.

[2]Best performance values are highlighted in bold. For the LongForm-C dataset, we use only the F1-score to measure token overlap between the predictions and the ground truth.

Algorithm 1 TPARAG

Require: Query q, Answer a, m relevant document $D = \{d_1, d_2, ..., d_m\}$, Iterations T, Maximum Token Substitution Rate pr_{sub}, LLM attacker LLM_{attack}, Attack Locator Loc, Similarity metric Sim,

Compute the initial generation logits: $L = \{l_1, l_2, ..., l_m\}$ for a, $l_i := P_{LLM_{attack}}(a|q, d_i; \theta_{LLM_{attack}})$,

Initialization threshold for generation attack: $l = \max_{l_i \in L} l_i$,

Compute the initial similarity score: $S := \{s_1, s_2, ..., s_m\}$ for q, $s_i := Sim(q, d_i)$,

Initialization threshold for retrieval attack: $s = \min_{s_i \in S} s_i$,

Entity-Based Attack Localization: $pos_{attack} := Loc(a)$

loop T times

  for $i \in [0 \dots m]$ **do**

    $d'_i := LLM_{attack}(q \oplus d_i; \theta_{LLM_{attack}}) \in D'$ ▷ *Generate parent malicious passage set D' via LLM_{attack}*

    for $j \in [0 \dots length(d'_i)]$ **do**

      $T_{ij} :=$ Top-$k(P_{LLM_{attack}}(t_{1:j-1}, q, d_i))$ ▷ *Compute/Save top-k token substitution candidates for each generated token t_{ij}*

  for $i \in [0 \dots m]$ **do**

    for $j \in [0 \dots length(d'_i)]$ **do**

      Generate pr_{rand}

      if $t_{ij} \in pos_{attack}$ and $pr_{rand} < pr_{sub}$ **then**

        $\hat{t}_{ij} := Random(T_{ij})$ ▷ *Random select the replacement token*

      else

        $\hat{t}_{ij} := t_{ij}$

      $\hat{d}_i := \hat{t}_{1:length(d'_i)}$ ▷ *Reconstructed malicious passage candidate $\hat{d}_i$*

    $\hat{s}_i := Sim(\hat{d}_i, q)$, $\hat{l}_i := P_{LLM_{attack}}(a|q, \hat{d}_i; \theta_{LLM_{attack}})$

  $\tilde{D} = \{\tilde{d}_i \mid i \in$ Top-$m_{\text{ascending}}(\hat{l}_1, \dots, \hat{l}_m), Sim(q, \tilde{d}_i) > s, \hat{l}_i < l\}$ ▷ *Sort and find m optimal malicious passages*

return: Optimal malicious passage set $\tilde{D}$ for query q

RAG Setting	LLM Attacker	HotpotQA					LongForm-C				
		$ASR_R\uparrow$	$ASR_L\uparrow$	$ASR_T\uparrow$	EM↓	F1↓	$ASR_R\uparrow$	$ASR_L\uparrow$	$ASR_T\uparrow$	EM↓	F1↓
Black-box (TPARAG)	Qwen2.5-3B	94.1	83.0	77.8	34.0	34.1	89.7	80.2	71.7	–	23.5
	Qwen2.5-7B	96.6	85.5	**82.8**	32.0	34.3	**91.7**	79.8	**72.0**	–	**22.1**
	Mistral-7B	81.4	90.3	75.6	**28.0**	**30.4**	90.0	79.0	71.6	–	22.8
	Gemma2-9B	76.6	**91.7**	70.0	36.0	39.3	74.7	**82.3**	63.0	–	24.0
Black-box (PoisonedRAG)	Qwen2.5-3B	97.6	75.5	73.1	36.0	39.0	90.6	78.4	69.1	–	22.4
	Qwen2.5-7B	26.7	45.4	14.1	41.0	45.5	13.0	68.7	7.7	–	23.2
	Mistral-7B	**98.4**	57.4	57.4	33.0	35.6	88.9	70.4	63.0	–	24.9
	Gemma2-9B	93.1	80.2	74.3	29.0	30.5	89.6	65.0	58.0	–	22.6
White-box (TPARAG)	Qwen2.5-3B	98.6	85.0	83.8	36.0	38.7	98.0	**79.4**	**77.8**	–	23.1
	Qwen2.5-7B	99.4	83.8	83.2	**31.0**	**31.6**	95.4	74.5	71.7	–	23.9
	Mistral-7B	**100.0**	90.1	**90.1**	33.0	35.2	96.2	73.7	70.5	–	**22.3**
	Gemma2-9B	85.1	**90.3**	77.0	38.0	41.5	81.7	77.1	64.9	–	23.4
White-box (PoisonedRAG)		**100.0**	77.2	77.2	48.0	51.4	**100.0**	75.3	75.3	–	25.7
w/o RAG		–	–	–	50.0	54.5	–	–	–	–	25.0
RAG		–	–	–	53.0	56.3	–	–	–	–	25.1

Table 7: Comparison of LLM attackers with different RAG settings across HotpotQA and LongForm-C datasets.