RAGPart & RAGMask: Retrieval-Stage Defenses Against Corpus Poisoning in Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a promising paradigm to enhance large language models (LLMs) with external knowledge, reducing hallucinations and compensating for outdated information. However, recent studies have exposed a critical vulnerability in RAG pipelines—*corpus poisoning*—where adversaries inject malicious documents into the retrieval corpus to manipulate model outputs. In this work, we propose two complementary retrieval-stage defenses: **RAGPart** and **RAGMask.** Our defenses operate directly on the retriever, making them computationally lightweight and requiring no modification to the generation model. RAGPart leverages the inherent training dynamics of dense retrievers, exploiting document partitioning to mitigate the effect of poisoned points. In contrast, RAGMask identifies suspicious tokens based on significant similarity shifts under targeted token masking. Across two benchmarks, four poisoning strategies, and four state-of-the-art retrievers, our defenses consistently reduce attack success rates while preserving utility under benign conditions. In order to stress-test our defenses, we further introduce an interpretable attack, **AdvRAGgen**, which outperforms existing attacks across four different retrievers and two different datasets. Our findings highlight the potential and limitations of retrieval-stage defenses, providing practical insights for robust RAG deployments.

1 Introduction

Large Language Models (LLMs) (OpenAI et al., 2024; DeepSeek-AI et al., 2025) have demonstrated remarkable capabilities in reasoning, problem-solving, and generalization, fueling deployment in real-world domains such as healthcare (Wang et al., 2023) and finance (Loukas et al., 2023). Despite these successes, LLMs remain limited by their static training data, resulting in outdated knowledge, hallucinations (Huang et al., 2025), and gaps in domain-specific expertise.

Retrieval-Augmented Generation (RAG) has recently gained popularity (Lewis et al., 2021) as a strategy to mitigate these limitations. RAG enhances LLMs by dynamically retrieving external documents relevant to a query from large corpora—such as Wikipedia (Thakur et al., 2021) or financial reports (Maia et al., 2018)—and augmenting the model's input context. Typically, documents are retrieved based on embedding similarity, computed by traditional methods (Salton and Buckley, 1988; Robertson and Zaragoza, 2009) or modern dense retrievers (Izacard et al., 2022; Wang et al., 2024).

Despite enhancing factual consistency (Ayala and Bechard, 2024), RAG's dependence on extensive, publicly sourced corpora introduces vulnerabilities to *corpus poisoning* attacks (Zou et al., 2024). In these attacks, adversaries insert maliciously crafted documents intended to manipulate the retrieved context, thereby influencing the model's outputs. An attack is successful when a poisoned document is retrieved (retrieval condition) and significantly impacts the LLM's generated response (generation condition).

Current defenses largely focus on the generation stage, proposing certifiable robustness via response aggregation (Xiang et al., 2024). However, these approaches rely on strong and often impractical assumptions. They assume that each retrieved document—often called a *golden document*—is independently sufficient to answer the query, that the retriever can consistently return enough such golden documents, and that it is computationally feasible to run a separate LLM generation for each one. In real-world systems where inference time and cost are major concerns, generating multiple responses per query is typically infeasible, making these defenses difficult to use in practice.

To overcome these issues, practical defenses must satisfy three conditions: **(W1)** *Effectiveness*:

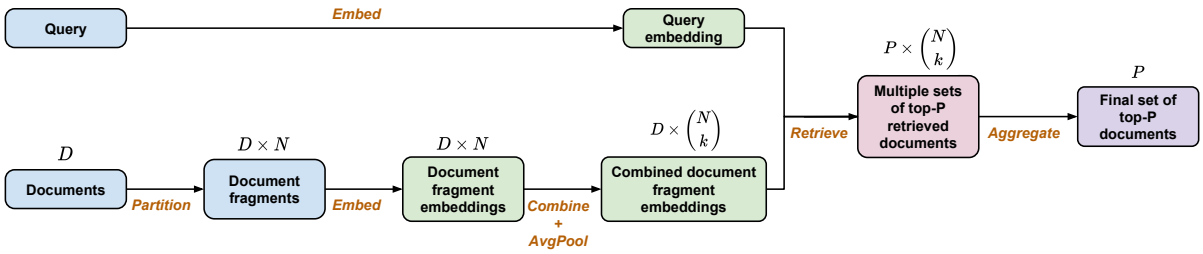


Figure 1: **RAGPart**: Figure illustrates the RAGPart defense, where each document is partitioned into fragments, which are individually embedded. Embeddings from multiple fragment combinations (e.g., subsets of size k) are then averaged to produce candidate document representations. These are used to retrieve multiple top-p document sets, which are aggregated to form the final top-p set for retrieval.

achieving low attack success rates without significant utility loss under benign conditions; **(W2)** *Efficiency*: remaining computationally lightweight for real-time use; and **(W3)** *Minimal retriever assumptions*: not requiring perfect or highly accurate retrieval.

To satisfy these properties, we propose addressing corpus poisoning earlier—at the retrieval stage. This is feasible in practice because retrievers are typically smaller models than long-context LLMs, and their similarity computations (e.g., dot products in embedding space) are highly parallelizable.

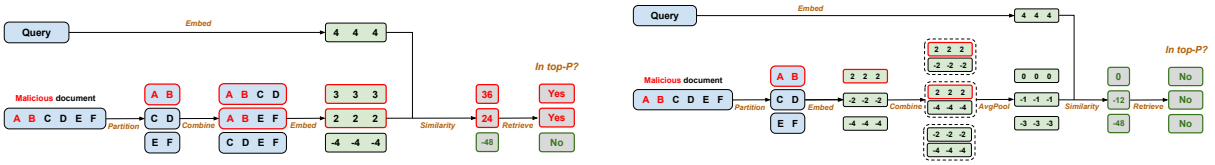
Motivated by deep partition-and-aggregation defenses (DPA) (Levine and Feizi, 2021; Sun et al., 2022) and perturbation-based defenses like RAP (Yang et al., 2021), we propose two complementary retrieval-stage defenses: **RAGPart** and **RAGMask**. RAGPart leverages dense retrievers' training dynamics, particularly the observation that document fragments often preserve the semantic meaning of the full document in embedding space. For example, dense retrievers such as Contriever (Izacard et al., 2022) explicitly define positive training pairs by treating randomly cropped portions of a document as semantically equivalent to the whole, inducing an inductive bias in the retriever's embedding space. We empirically observe that this behavior generalizes across multiple dense retrievers. By exploiting the similarity between full-document and fragment embeddings, RAGPart formulates a defense to mitigate the effect of poisoned content. RAGMask, on the other hand, targets a different vulnerability: poisoning often hinges on a small set of influential tokens that disproportionately affect similarity scores. By selectively masking these tokens and measuring the resulting similarity shift, RAGMask identifies and suppresses poisoned documents.

Our contributions are summarized as follows.

- We propose two retrieval-stage defenses (**RAGPart** and **RAGMask**) that are both computationally efficient and effective at mitigating corpus poisoning, without modifying the LLM or relying on strong retriever assumptions.
- We demonstrate the efficacy of these defenses across two benchmark datasets and four distinct poisoning strategies. In addition to evaluating against existing attacks, we introduce a stronger, interpretable poisoning attack—**AdvRAGgen**—and show that our defenses remain robust under this more challenging threat model.
- We present a theoretical result demonstrating the superiority of RAGPart over a naive combinatorial approach that does not exploit the properties of dense retrievers.
- We systematically analyze trade-offs between RAGPart and RAGMask in terms of defense effectiveness and computational efficiency, providing practical guidance for real-world deployment across various system constraints.

2 Related Work

Retrieval-Augmented Generation: RAG improves the accuracy of LLM outputs and reduces hallucinations by retrieving relevant documents from a knowledge database given a query, and generating responses conditioned on these retrieved documents (Lewis et al., 2021; Izacard and Grave, 2021). Particularly in sensitive fields such as law (Mao et al., 2024), RAG systems enable LLMs to generate reliable outputs that reflect up-to-date



(a) **Naive Combination Baseline** (b) **RAGPart**

Figure 2: **Naive Combination Baseline vs. RAGPart**: A toy example with N=3 and k=2. In this scenario, the naive combination approach is likely to retrieve the malicious document, since the poison persists in the raw text of many combined fragments (before embedding). In contrast, RAGPart benefits from additional robustness due to mean pooling over fragment embeddings, which can dilute the effect of poisoned fragments and prevent the malicious document from being retrieved under the same conditions. For illustration purposes, we assume that large inner product values (e.g., 24, 36) correspond to the document ending up among the top-p retrieved results, while small values (e.g., –48, 0) correspond to cases where it does not.

knowledge. This effectiveness has led to the wide deployment of RAG on both public and e-commerce platforms. Although traditional retrieval frameworks such as TF-IDF (Salton and Buckley, 1988) and BM25 (Robertson and Zaragoza, 2009) have shown promise in retrieving relevant documents using word frequency statistics, recent advances in transformer-based dense retrievers (Izacard and Grave, 2021; Xiong et al., 2020; Wang et al., 2024; Li et al., 2023)—which encode semantic meanings into embedding vectors—have demonstrated superior performance on state-of-the-art retrieval benchmarks (Muennighoff et al., 2023).

Attacks: The corpus poisoning attacks against RAGs can be divided into black-box and white-box attacks based on the access the attacker has to the retriever model. The goal of the attacker is to either create a retrievable adversarial passage that can cause a harmful generation when added to the context of an LLM or craft poisons whose addition into the adversarial passage can make them retrievable for a given query. Works such as (Zou et al., 2024; Zhong et al., 2023) have crafted white-box poisons by exploiting the gradient of the retrievers, which when added to an adversarial passage can fool the retriever into retrieving the passage. In black-box settings, the works of (Zou et al., 2024) have proposed adding the query itself to the adversarial passage to make it retrievable.

Defenses: Early defenses, such as those by (Weller et al., 2024), proposed paraphrasing queries to retrieve multiple robust passages and thereby mitigate misinformation at the retrieval stage. Although these defenses can handle weaker adversaries, they often fail against stronger attacks (Zou et al., 2024) and robust retrievers capable of preserving semantic meaning across paraphrases. Another line of work (Xiang et al., 2024) proposes certified defenses against corpus poisoning at the generation stage (rather than at retrieval) by aggregating responses generated from each of the top-p retrieved documents. However, these generation-stage defenses rely on strong assumptions—each golden document must independently suffice for generation, and retrievers must reliably retrieve an overwhelming number of golden documents. In practice, these conditions rarely hold, and such approaches are computationally expensive because long-context LLMs must be invoked multiple times per inference.

3 Method

3.1 Defense: RAGPart

Most state-of-the-art dense retrievers (Wang et al., 2024; Izacard et al., 2022; Li et al., 2023) are pre-trained using a large-scale contrastive loss (Khosla et al., 2021) and then fine-tuned on human-annotated data. Some retrievers, such as (Izacard et al., 2022), select positive pairs during contrastive learning by sampling a random portion of a document and treating it as a positive example paired with the full document. This setup explicitly introduces an inductive bias that encourages the model to produce similar embeddings for different fragments of the same document. In practice, as we show in the Results section, even models that are not trained this way still exhibit similar behavior.

Inspired by this observation and deep partition-and-aggregation (DPA) defenses (Sun et al., 2022), we propose RAGPart. The key idea is to partition a document into N fragments and apply the dense retriever's embedding model to each fragment individually. Due to the inductive bias of dense retriev-

ers, the fragment embeddings tend to preserve the semantic similarity of the full document. We then average the embeddings of different combinations of fragments to form a final similarity score. If the number of poisoned fragments n_p is not too large, their influence is diminished through the averaging step. By evaluating multiple such combinations and aggregating the results, RAGPart effectively reduces the impact of the poisoned samples, as demonstrated in the Results section.

In the context of dense retrieval as the first stage of RAG, and given the properties of document fragments, a document can be broken into N fragments. Based on how we form combinations of these fragments, we consider two approaches, as seen below:

1. **RAGPart**: We first embed each of the N fragments using the dense retriever's embedding model. Then, we form combinations of k fragments (e.g., all $\binom{N}{k}$ subsets of size k), and average their embeddings to form a combined representation. This approach leverages the inductive bias of dense retrievers, which produce similar embeddings for semantically related text, allowing the averaged embedding to preserve the original document's meaning even when some fragments may be poisoned. This method is illustrated in Figure 1.

2. **Naive combination of fragments baseline**: In this approach, we first select combinations of k out of N document fragments (without first embedding them) and concatenate the raw text. The resulting text is then passed through the embedding model. A toy example containing a comparison between the two methods is shown later in Figure 2. Since embedding happens after mixing the content, poisoned fragments can—by design—have a significant influence on the final embedding.

Once a new set of $\binom{N}{k}$ embeddings is formulated to represent each document, we can build $\binom{N}{k} \cdot D$ embedding databases for a given corpus of D documents. These databases are used to retrieve $\binom{N}{k}$ sets of top-p documents. The retrieved results can then be aggregated to form the final top-p documents to be passed to the generator. We explore two aggregation strategies below:

1. **Intersection-based aggregation**: From the $\binom{N}{k}$ sets of top-p documents, we select only those documents that appear in all of the sets. If no document appears across all sets, we randomly choose p documents from the union of the $\binom{N}{k} \cdot p$ retrieved documents. While this approach can reduce the chance of retrieving adversarial documents, it often severely impacts the success rate (SR). As a result, we do not use this strategy in most of our experiments.

2. **Majority vote-based aggregation**: From the $\binom{N}{k}$ sets of top-p documents, we select the p documents that appear most frequently. It is important to note that the ideal k condition for robustness in (Sun et al., 2022), which assumes $n_p < \frac{N-1}{2}$, no longer guarantees robustness in this context. This is because the aggregation now involves selecting the top-p documents by majority vote across combinations, rather than making a decision per combination. Thus, even if the adversarial document is not the top-ranked in most combinations, it may still be retrieved due to frequency. While setting $p = 1$ can restore the original robustness guarantee, it can greatly reduce utility.

Furthermore, the framework in (Sun et al., 2022) assumes that an adversary can influence the output if it is present in any single fragment within a combination. To be able to weaken this assumption, we considered the RAGPart framework that minimizes the impact of poisoned fragments even when they are included in some combinations.

Motivated by the shortcomings in majority vote-based aggregation, we analyze the effectiveness of the RAGPart framework compared to the naive combination of fragments in suppressing the effect of adversarial poisons when they are present in fragment combinations. Adversarial tokens are typically designed to increase the likelihood of retrieval when included in a document. This behavior poses a disadvantage for the naive combination approach—since the adversary remains in the raw text before embedding, its influence is preserved, and the resulting embedding can still lead to the document being retrieved. In contrast, in the RAGPart approach, each fragment is embedded independently and then averaged, reducing the influence of individual poisoned fragments. Since the poisons are not optimized to dominate the mean of multiple embeddings, their effectiveness naturally decreases. Designing adaptive poisons to counteract this is difficult: it would require crafting embeddings with unusually large norms, which is hard to achieve in

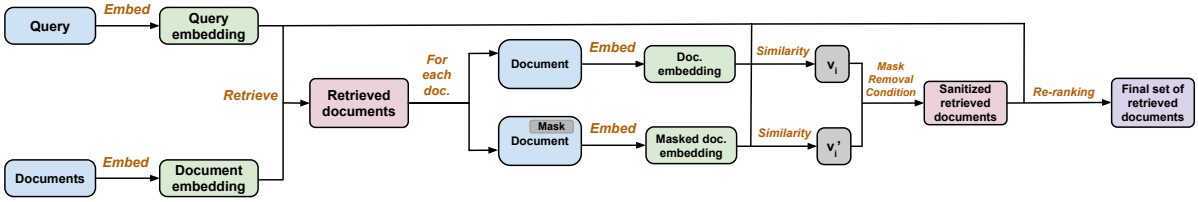


Figure 3: **RAGMask**: Figure illustrates the concept of RAGMask, where the top αp retrieved documents are sanitized by observing the shift in similarity with the query under masking.

practice and can be mitigated by anomaly detection in embedding space. This distinction between the naive combination method and RAGPart is illustrated with a toy example using $N = 3$ and $k = 2$ in Figure 2.

3.2 Defense: RAGMask

The idea of RAGMask takes inspiration from perturbation-based defenses. The idea behind perturbation-based defenses (Yang et al., 2021) is to analyze the behavior of a given sample in the presence of perturbations and make decisions accordingly. In the case of RAG, if the addition of a poison to a document makes it retrievable, then masking or occluding that token should cause a substantial drop in the similarity score between the document and the intended query. We leverage this insight to design the RAGMask defense, as shown in Figure 3, in the following way.

Given a corpus of documents D and a query q, we first convert them into embeddings using the retriever model. We then retrieve the top αp documents, where $\alpha > 1$. Assume that the length of a single document is l_i tokens. Each document has an initial similarity score v_i^q with respect to the query. For each of the top αp documents, we divide the document into $\frac{l_i}{m}$ segments, where m is a predefined hyperparameter representing the mask length. We mask the document at each of these $\frac{l_i}{m}$ positions and recompute the similarity score between the query and the masked version of the document, denoted by $v_i^{q\prime}$. If $v_i^{q\prime} + \delta > v_i^q$, we keep the masked tokens in the document; otherwise, we discard them. After $\frac{l_i}{m} \cdot \alpha p$ such operations, we obtain a new set of partially masked documents, which we refer to as sanitized documents.

Since retrievers are generally much smaller than LLMs, these operations can be parallelized efficiently to maintain acceptable time complexity. We then recompute the similarity between the sanitized documents and the query q, re-rank the αp documents, and finally select the top p as the final set of retrieved documents.

3.3 Attack: AdvRAGgen

In addition to the attacks discussed in the Experiments section, and inspired by the work of AdvB-DGen (Pathmanathan et al., 2024), we propose an interpretable attack against RAG retrieval, called **AdvRAGgen**. The idea behind this attack is to use a general-purpose causal language model that takes in a query and an adversarial document and generates a paraphrase of the adversarial document such that it is retrieved.

The generator is trained via Direct Preference Optimization (DPO) using three feedback signals:

1. The semantic similarity between the original adversarial document and its generated paraphrase, measured by a semantic embedding model. This ensures the paraphrase maintains the intended content, satisfying the generation condition.

2. The similarity between the query and the paraphrase in the target retriever's embedding space. This ensures the paraphrase is retrievable, satisfying the retrieval condition.

3. The negative ROUGE-L score between the query and the paraphrase. This discourages trivial attacks that involve simply inserting the query into the adversarial document.

For further details, refer to the Appendix.

4 Theoretical Analysis

Robustness: To better understand the performance gap between our proposed method, RAGPart, and the naive combination of fragments baseline, we provide a theoretical analysis of how poisoned fragments affect retrieval outcomes.

In the absence of adversarial documents, we assume that the top-p retrieved results across the $\binom{N}{k}$

fragment mixes are consistent up to permutation. Let n_a denote the number of adversarial documents in the corpus, each containing n_p poisoned fragments. We assume $n_a \leq D - p$, ensuring that at least p clean documents remain retrievable. For this analysis, we begin with the case $n_a = 1$ (the case of $n_a > 1$ is discussed in the Appendix).

In the naive combination of fragments baseline, embeddings are computed after fragment mixing. As a result, any mix that includes at least one poisoned fragment is considered adversarial. The number of such poisoned mixes is:

$$\binom{N}{k} - \binom{N - n_p}{k}.$$

In contrast, RAGPart computes embeddings for individual fragments before mixing and aggregates them via mean pooling. This design reduces the influence of individual poisoned fragments. Assuming that a mix becomes adversarial only when it contains at least two poisoned fragments, the number of poisoned mixes becomes:

$$\binom{N}{k} - \binom{N - n_p}{k} - n_p \binom{N - n_p}{k - 1}.$$

Both of these expressions are derived and discussed in detail in the Appendix. Due to space constraints, we defer a deeper analysis of how much this reduction in poisoned mixes improves robustness to the Appendix, where we evaluate various values of N, k, and n_p under a formal robustness condition. Additionally, we analyze the computational complexity of both strategies from a theoretical perspective.

Memory: In terms of memory efficiency, RAGPart is more efficient than the naive baseline because it allows inner product computations to be performed before creating the fragment mixes. Specifically, RAGPart requires memory on the order of $\mathcal{O}(D \times N \times E + D \times \binom{N}{k})$, while the naive baseline requires $\mathcal{O}(D \times \binom{N}{k} \times E)$, where E is the size of the embedding.

5 Experiments

Dataset and Models: We used two question answering datasets, namely Natural Questions (NQ) (Kwiatkowski et al., 2019) and Financial Opinion Mining and Question Answering (FiQA) (Maia et al., 2018). The NQ dataset consists of a corpus of 2,681,468 documents, while the FiQA dataset contains 57,638 documents. From the query sets, we randomly select 512 queries and use them as our test set. For each query, we randomly select 3 irrelevant documents. The attacker's goal is to craft poisons such that these irrelevant documents become retrievable. Each query has more than 10 relevant documents ("golden documents") in the corpus, which are used to evaluate the utility of the defense under benign settings. For dense retrievers, we consider four models: Contriever (Izacard et al., 2022), ANCE (Xiong et al., 2020), multilingual E5 (Wang et al., 2024), and GTE-large (Li et al., 2023).

Evaluation Metrics: We evaluate both the robustness of the defense under attack and its utility under benign settings. Robustness is measured using the ***Attack Success Rate (ASR)***, which is the percentage of test queries for which the retriever returns at least one poisoned or malicious document. Utility is measured using the ***Success Rate (SR)***, defined as the percentage of test queries for which the retriever returns at least one golden document. We report the average drop in SR to quantify utility degradation (lower is better), and the average drop in ASR to assess defense robustness (higher is better).

Attacks: We consider both gradient-based and interpretable attacks. For gradient-based attacks, following Zou et al. (2024) and Zhong et al. (2023), we consider a HotFlip-style attack (Ebrahimi et al., 2018), which searches for effective adversarial tokens using gradient information. We also introduce a variant called *HotFlip (spread out)*, where the adversarial tokens are distributed throughout the document instead of being placed in a localized region. For interpretable attacks, we include the *Query-as-Poison* method, where the query is directly inserted into the document to improve retrievability (Zou et al., 2024). Additionally, we consider AdvRAGgen, described in the previous section.

Baseline Defenses: Following (Zou et al., 2024), we consider paraphrasing-based and perplexity-based defenses. The paraphrasing defense assumes that certain poisons can be neutralized by rewriting the document. We use LLaMA 3.3 70B (Grattafiori et al., 2024) for generating paraphrases. The perplexity-based defense relies on the intuition that poisoned documents tend to have higher perplexity scores. We compute perplexity using a GPT-2 model (Radford et al., 2019) and filter out high-perplexity documents as potential poisons.

6 Results

Due to space constraints, for most of the results we present both the ASR and SR (utility) results as averages over the four retrievers considered. For fine-grained results, refer to the Appendix. In the Appendix, we further provide hyperparameter analysis, motivating the choice of N, k in RAGPart and δ, m in RAGMask.

6.1 Results: Attack

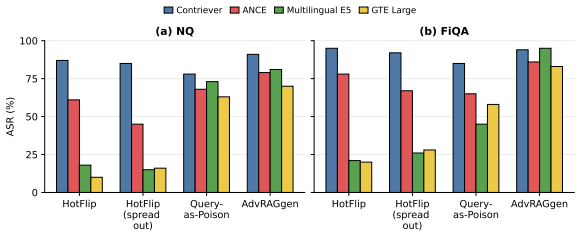


Figure 4: **Attack success rate (ASR) across retrievers.** AdvRAGgen maintains a high ASR across retrievers on both NQ and FiQA compared to prior attacks, indicating that it is harder to mitigate with standard retrievers and harder to defend against using existing defenses, as we will show later.

Effectiveness of AdvRAGgen over baseline attacks: First, we showcase the effectiveness of AdvRAGgen against baseline attacks and defenses in Figures 4, 6, and Table 1. As seen in Figure 4, across four retrievers and two different datasets, AdvRAGgen elicits a high ASR while preserving lower perplexity compared to Query-as-Poison, as shown in Table 1. Furthermore, due to its semantically consistent nature, it bypasses paraphrasing as a defense more effectively, as seen in Figure 6. Note that the poisons for AdvRAGgen were created using only the Contriever model as the target; nevertheless, the poisons exhibit better attack transferability to other retriever models, thereby establishing AdvRAGgen as a ***stronger, transferable black-box attack*** compared to gradient-based attacks.

6.2 Results: Defense

Ineffectiveness of the baseline defenses: As seen in Figure 6 and Table 1, both the paraphrase-based and perplexity-based defenses fail to defend against interpretable poisoning attacks, even though they are effective against gradient-based attacks thus motivating the need for a better effective defense.

Ablation: Effect of intersection-based aggregation and majority vote-based aggregation:

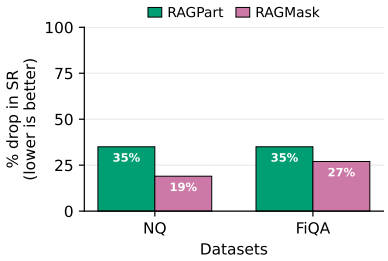


Figure 5: **Drop in success rate (SR; lower is better).** RAGMask yields a smaller utility drop than RAGPart on both NQ and FiQA.

Intersection-based aggregation can remove poisoned documents more effectively than majority vote-based aggregation. This is shown in the Appendix in Figure 9, where intersection-based aggregation leads to a larger drop in ASR for both the naive combination baseline and RAGPart. Due to the additional robustness of RAGPart, the difference between intersection-based and majority vote-based aggregation is minimal in that case, unlike in the naive baseline. However, when considering SR over golden documents, Figure 8 in the Appendix shows that intersection-based aggregation causes a larger drop in SR, making it unsuitable for practical deployment. Therefore, we adopt majority vote-based aggregation for the rest of the paper. For more detailed results, see Table 14, 15, 16, and 17 in the Appendix.

Viability of the naive combination baseline vs RAGPart as a practical defense: Although the naive combination baseline preserves utility slightly better than RAGPart under majority vote-based aggregation (Figure 8), it fails to defend against any of the considered attacks. This makes *RAGPart a practically viable defense.*

Effectiveness of RAGPart & RAGMask: As shown in Figure 6, both RAGPart and RAGMask effectively defend against all considered attacks, outperforming baseline defenses, with RAGPart achieving slightly better ASR reduction. In terms of utility preservation, RAGMask demonstrates a stronger ability to maintain retrieval performance, as shown in Figure 5.

Computational efficiency of RAGPart & RAGMask: Although RAGMask performs better in preserving the utility of benign retrievals and provides comparable robustness, it is relatively more expensive computationally. For a corpus of size D and an embedding space of dimension n_e, RAGPart requires $2n_e \cdot D \cdot \binom{N}{k}$ floating point operations

Table 1: **Perplexity-based Defense**: This table shows that, similar to paraphrase-based defenses, perplexity-based defenses are effective at detecting gradient-based attacks. However, they fail to distinguish poisoned documents from benign ones in the case of interpretable attacks such as Query-as-Poison and AdvRAGgen, and therefore perform worse than the proposed defenses. We evaluate perplexity using four retrievers—Contriever (Izacard et al., 2022), ANCE (Xiong et al., 2020), Multilingual E5 (Wang et al., 2024), and GTE Large (Li et al., 2023). In the table, we report perplexity scores and highlight detections in red when the defense correctly identifies a poisoned document as malicious, and in green when it incorrectly classifies it as benign.

Dataset	Retriever	No Poison	HotFlip	HotFlip (spread out)	Query-as-Poison	Adv-RAG-gen
Natural Questions (NQ)	Contriever	143	989	1827	119	74
	ANCE	143	5726	12021	119	74
	Multilingual E5	143	113	392	119	74
	GTE Large	143	224	447	119	74
FiQA	Contriever	143	274	631	100	53
	ANCE	143	466	1095	100	53
	Multilingual E5	143	86	252	100	53
	GTE Large	143	113	303	100	53

(FLOPs), where N is the number of fragments and k is the combination size. Given that the retriever requires R FLOPs to embed a document during the forward pass, RAGMask requires $R \cdot \frac{l_i}{m} \cdot \alpha p$ FLOPs, where l_i is the maximum document length, m is the mask size, and αp is the number of documents retrieved prior to sanitation. In practice, R may be large depending on the retriever's architecture. Thus, a computational tradeoff exists between RAGPart and RAGMask. However, since retrievers are generally smaller than autoregressive LLMs used for generation, RAGMask remains computationally tractable compared to generation-stage defenses such as (Xiang et al., 2024), which require multiple LLM calls at inference time.

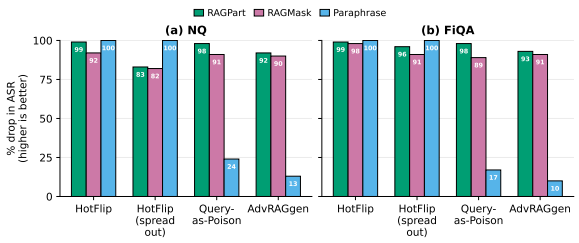


Figure 6: **Drop in attack success rate (ASR) on NQ and FiQA (higher is better).** RAGPart and RAGMask remain robust across attacks, while paraphrasing-based defenses fail on interpretable attacks (Query-as-Poison, AdvRAGgen).

7 Conclusion

n this work, we first propose a stronger, interpretable, transferable black-box attack, namely AdvRAGgen, that outperforms existing attacks in terms of attack efficacy and robustness against existing defenses, thereby highlighting the need for stronger defenses against corpus poisoning attacks. To this end we propose two retrieval-stage defenses, RAGPart and RAGMask, aimed at preventing adversarial documents from being retrieved in Retrieval-Augmented Generation (RAG) systems. Unlike generation-stage defenses, which often rely on strong assumptions—such as the self-sufficiency of each retrieved document, the presence of multiple relevant documents, high retriever accuracy, and the availability of significant computational resources—our defenses operate in a computationally tractable manner while maintaining robustness against a variety of poisoning attacks. Moreover, they preserve the utility of the retriever in benign settings.

We demonstrate that our methods outperform commonly used retrieval-based defenses, such as paraphrasing and perplexity filtering. Between our two approaches, RAGMask offers better utility preservation and comparable robustness to attacks. However, we identify a tradeoff in computational cost, making RAGPart a more practical choice in resource-constrained scenarios.

Finally, while our defenses are effective against many corpus poisoning attacks, we also discuss in the appendix some attack types that remain difficult to detect at the retrieval stage, highlighting important directions for future research on developing defenses against such adversaries.

Limitations

Despite showcasing robustness against attacks while maintaining a minimal drop in SR (utility), there are certain types of poisoned documents that the proposed methods may not be well-suited to defend against. One such case involves adversarial documents that are semantically similar to the query while embedding direct misinformation. These documents may closely match the query in embedding space and thus be retrieved, despite their content being misleading. Defending against such poisons is especially challenging at the retrieval stage, as their harmful nature may only become apparent after the generation step. Unless retrievers are explicitly trained to incorporate generative understanding—i.e., sensitivity to potential downstream implications—their focus on semantic similarity alone may not be sufficient to detect such issues.

References

Orlando Ayala and Patrice Bechard. 2024. Reducing hallucination in structured outputs via retrieval-augmented generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, page 228–238. Association for Computational Linguistics.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. *Preprint*, arXiv:1712.06751.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Preprint*, arXiv:2112.09118.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. *Preprint*, arXiv:2007.01282.

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. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2021. Supervised contrastive learning. *Preprint*, arXiv:2004.11362.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Alexander Levine and Soheil Feizi. 2021. Deep partition aggregation: Provable defense against general poisoning attacks. *Preprint*, arXiv:2006.14768.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. 2023. Making llms worth every penny: Resource-limited text classification in banking. In *4th ACM International Conference on AI in Finance*, ICAIF '23, page 392–400. ACM.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1941–1942, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Kelong Mao, Zheng Liu, Hongjin Qian, Fengran Mo, Chenlong Deng, and Zhicheng Dou. 2024. RAG-studio: Towards in-domain adaptation of retrieval augmented generation through self-alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 725–735, Miami, Florida, USA. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Pankayaraj Pathmanathan, Udari Madhushani Sehwag, Michael-Andrei Panaitescu-Liess, and Furong Huang. 2024. Advbdgen: Adversarially fortified prompt-specific fuzzy backdoor generator against llm alignment. *Preprint*, arXiv:2410.11283.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *"https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe"*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Yanchao Sun, Ruijie Zheng, Parisa Hassanzadeh, Yongyuan Liang, Soheil Feizi, Sumitra Ganesh, and Furong Huang. 2022. Certifiably robust policy learning against adversarial communication in multi-agent systems. *Preprint*, arXiv:2206.10158.

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

Calvin Wang, Joshua Ong, Chara Wang, Hannah Ong, Rebekah Cheng, and Dennis Ong. 2023. Potential for gpt technology to optimize future clinical decision-making using retrieval-augmented generation. *Annals of Biomedical Engineering*, 52.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

Orion Weller, Aleem Khan, Nathaniel Weir, Dawn Lawrie, and Benjamin Van Durme. 2024. Defending against disinformation attacks in open-domain question answering. *Preprint*, arXiv:2212.10002.

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

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

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021. Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models. *Preprint*, arXiv:2110.07831.

Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. 2023. Poisoning retrieval corpora by injecting adversarial passages. *Preprint*, arXiv:2310.19156.

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

A Q&A

A.1 Attack

1. **Why are the attacks success rates very low for gradient based methods (Hotflip and Hotflip (spread out) attacks?**

   Given a fixed number of adverserial tokens, gradient-based attacks that follow the paradigm of (Ebrahimi et al., 2018) attacks work with two hyperparameters, namely the number of top candidates to consider based on the gradient and the number of iterations to the operations hotflip attack for. The higher these values are, the better the ASR is of these attacks. However, this also can lead to higher computations. Due to the number of experiments we were considering, we had limited these hyperparameters to 30 each. Stronger models such as multilingual e5 and GTE large needed more iterations to perform a successful attack. As an ablation, in Table 6, we have provided results (for a smaller test size due to computation constraints, as creating one of these poison can take up-to an hour) for higher number iterations and proposed defense's efficacy against those poisons.

A.2 Defense

1. **Generally, in deep-partition and aggregation (DPA) based defenses, don't people have a guarantee on robustness with majority voting-based aggregation? Why don't we see it in the case of RAGPart?**

   Those guarantees on majority voting are given in scenarios where only one decision is made from the aggregation. In contrast, in the case of RAG, generally the top p samples are drawn from the set of documents. Even though the N and k are chosen in accordance with those guarantees, the adversary can end up getting chosen in the top p documents even though it is not the topmost relevant document. While we can restrict ourselves to the top 1 document, it can severely hurt the performance of the system due to the existence of multiple documents and the retriever's deficiency.

2. **What are the limitations of the proposed defenses? In what scenarios can the defense not defend and what is your argument against such scenarios?**

   The scenario we consider the most here is the scenario where there is an adversarial document and it is generally not retrievable for the retriever, and a poison is added to make it retrievable. If the adversarial document itself is retrievable, then our defenses may fail.

   For example, consider the following scenario. Given a query "Where is Mount Everest located?" a golden document can be "Mount Everest is located between Nepal and Tibet," and an adversarial document can be "Mount Everest is located at Spain." An important point to note here is that both these documents are semantically similar, i.e., both the documents talk about the location of Mount Everest, and one document is adversarial due to the fact that it contains misinformation or can induce the generation to contain misinformation. If the retriever is retrieving this document, that means the retriever doesn't have the knowledge about the factual error in the document. Thus, there is no way to defend against such a poison at the retrieval stage; rather, we argue that one should consider a generation-level defense as this is due to the deficiency of the retriever. If one is to solve this problem in a retrieval defense, then they should consider enhancing the retriever with more hard negatives (a document that is getting retrieved but is not actually a relevant document; i.e., harder to discriminate document). The attack we proposed (AdvRAGgen) to some level does create such a document (where it blends the query into the adversarial document), that's why it was a relatively harder attack to defend against, although our proposed defenses show an acceptable level of robustness against the attack.

   One may ask in what scenarios our assumption of the attack is practical. RAGs are used in e-commerce or in retrieving law documents in many cases. In the case of an e-commerce website, a malicious product seller may add a poison to make his product retrievable for an irrelevant query in order to increase their sales. Similarly, in the case of a legal RAG system, one may try to make a document with an incorrect judgment retrievable with malicious intentions.

B Theoretical Analysis

B.1 RAGPart vs. Naive Baseline for One Adversarial Document

To better understand the performance gap between our proposed method, RAGPart, and the naive combination-of-fragments baseline, we provide a theoretical analysis of how poisoned fragments affect retrieval outcomes.

In the absence of adversarial documents, we assume that the top-p retrieved results across the $\binom{N}{k}$ possible fragment mixes are consistent up to permutation. Let n_a denote the number of adversarial documents in the corpus, each containing n_p poisoned fragments. We assume $n_a \leq D - p$, ensuring that at least p clean documents remain retrievable. We begin by analyzing the case where $n_a = 1$ (the extension to $n_a > 1$ is discussed later).

For each document, there are $\binom{N}{k}$ combinations of fragments that are either (1) fed into the embedding model after mixing (in the naive baseline), or (2) pre-embedded individually and averaged later (in RAGPart).

When computing top-p results (e.g., as shown in Figure 1), we construct a matrix of size $p \times \binom{N}{k}$, where each entry corresponds to a final embedding. In the absence of adversaries, each column would represent a mix of fragments from the same clean document (up to permutation).

However, in the presence of a single adversarial document, in the worst-case scenario, instead of observing $\binom{N}{k}$ occurrences for each of the top-p clean documents, we may observe $\binom{N}{k}$ occurrences for $p - 1$ clean documents, $\binom{N}{k} - x$ occurrences for the p-th clean document, and x occurrences for the poisoned mixes. For majority voting to be effective at filtering out poisoned embeddings, we require:

$$x < \frac{1}{2}\binom{N}{k}.$$

Naive Baseline. In the naive combination-of-fragments baseline, embeddings are computed after fragment mixing. As a result, any mix that includes even a single poisoned fragment is considered adversarial. The number of such poisoned mixes is:

$$x = \binom{N}{k} - \binom{N - n_p}{k}.$$

This expression is derived by subtracting the number of fragment mixes that contain no poisoned fragment from the total number of possible mixes.

A sufficient condition for robustness in the naive baseline is:

$$\binom{N}{k} - \binom{N - n_p}{k} < \frac{1}{2}\binom{N}{k}.$$

RAGPart. In contrast, RAGPart computes embeddings for individual fragments before mixing, and aggregates them via mean pooling. This reduces the influence of any single poisoned fragment. Suppose that a mix is considered poisoned only if it contains *at least two* poisoned fragments. Then, the number of poisoned mixes is:

$$x = \binom{N}{k} - \binom{N - n_p}{k} - n_p\binom{N - n_p}{k - 1}.$$

Here:

- $\binom{N-n_p}{k}$ counts the number of clean mixes (no poisoned fragments),
- $n_p\binom{N-n_p}{k-1}$ counts the number of mixes containing exactly one poisoned fragment,

and their sum represents the number of mixes that are not adversarial for RAGPart.

Thus, a sufficient condition for robustness in RAGPart is:

$$\binom{N}{k} - \binom{N-n_p}{k} - n_p\binom{N-n_p}{k-1} < \frac{1}{2}\binom{N}{k}.$$

In Tables 2 and 3, we show the values of N and k for which the sufficient condition for robustness holds under the naive baseline and RAGPart, respectively, for $n_p = 2$. Similar results for $n_p = 3$ are presented in Tables 4 and 5. We observe that RAGPart yields significantly more combinations of N and k that satisfy the robustness condition compared to the naive baseline. Overall, we recommend using lower values of N and higher values of k in practice to minimize performance degradation.

Table 2: Sufficient condition for robustness under the naive baseline with $n_p = 2$. A green checkmark (✓) indicates that the condition holds for the given pair of N and k.

$N\backslash k$	3	4	5	6	7	8	9	10	11	12	13	14	15
3	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A
10	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A
11	✓	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A
12	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A
13	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A
14	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A
15	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 3: Sufficient condition for robustness under RAGPart with $n_p = 2$. A green checkmark (✓) indicates that the condition holds for the given pair of N and k.

$N\backslash k$	3	4	5	6	7	8	9	10	11	12	13	14	15
3	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	✓	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	✓	✓	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	✓	✓	✓	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	✓	✓	✓	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	✓	✓	✓	✓	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A
10	✓	✓	✓	✓	✓	✗	✗	✗	N/A	N/A	N/A	N/A	N/A
11	✓	✓	✓	✓	✓	✗	✗	✗	✗	N/A	N/A	N/A	N/A
12	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	N/A	N/A	N/A
13	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	N/A	N/A
14	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	N/A
15	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗

Table 4: Sufficient condition for robustness under the naive baseline with $n_p = 3$. A green checkmark (✓) indicates that the condition holds for the given pair of N and k.

$N \backslash k$	3	4	5	6	7	8	9	10	11	12	13	14	15
3	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A
10	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A
11	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A
12	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A
13	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A	N/A
14	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	N/A
15	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 5: Sufficient condition for robustness under RAGPart with $n_p = 3$. A green checkmark (✓) indicates that the condition holds for the given pair of N and k.

$N \backslash k$	3	4	5	6	7	8	9	10	11	12	13	14	15
3	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	✓	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	✓	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	✓	✓	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A	N/A
10	✓	✓	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A	N/A
11	✓	✓	✓	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A	N/A
12	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	N/A	N/A	N/A
13	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	N/A	N/A
14	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	N/A
15	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗

B.2 RAGPart vs. Naive Baseline for Multiple Adversarial Documents

Assuming n_p is the number of poisoned fragments in the adversarial documents, we analyze the robustness of both methods under a worst-case scenario. Specifically, we assume that at most one poisoned mix embedding appears in each column of the $p \times \binom{N}{k}$ matrix. While this assumption can be relaxed in practice, we adopt it here for analytical simplicity.

Under this setting, as long as the *total number of poisoned mixes* across all n_a adversarial documents is less than $\frac{n_a}{n_a+1}\binom{N}{k}$, the least frequent clean document will still appear more often than any individual poisoned document in the final embedding matrix. This condition ensures robustness of top-p retrieval: each poisoned document can contribute at most $\frac{1}{n_a+1}\binom{N}{k}$ mixes, while the least frequent clean document will contribute more than $\frac{1}{n_a+1}\binom{N}{k}$. Thus, clean documents will dominate in frequency, and the top-p results will exclude all poisoned ones.

Accordingly, the sufficient conditions for robustness are:

Naive baseline:

$$\binom{N}{k} - \binom{N-n_p}{k} < \frac{1}{n_a+1}\binom{N}{k}$$

RAGPart:

$$\binom{N}{k} - \binom{N-n_p}{k} - n_p\binom{N-n_p}{k-1} < \frac{1}{n_a+1}\binom{N}{k}$$

B.3 Computational Complexity: RAGPart vs. Naive Baseline

In this section, we analyze the computational complexity of computing the final document embeddings prior to similarity comparison with the query. We compare the naive baseline and RAGPart from a theoretical perspective.

Assume that running the embedding model on a fraction $\frac{1}{N}$ of a document requires R FLOPs. Then, running it on a fraction $\frac{k}{N}$ (i.e., a mix of k fragments) requires $k \times R$ FLOPs.

Naive Baseline. For each document, there are $\binom{N}{k}$ possible mixes of fragments. The embedding model is applied to each of these k-length mixes, so the total computational cost is:

$$\text{FLOPs}_{\text{naive}} = D \times \binom{N}{k} \times (k \times R)$$

RAGPart. In RAGPart, the embedding model is applied once to each of the N individual fragments per document, for a total of:

$$\text{FLOPs}_{\text{embedding}} = D \times N \times R$$

Then, for each of the $\binom{N}{k}$ combinations, we compute the mean of k precomputed embeddings, each of dimension n_e, costing $k \times n_e$ FLOPs per mix. Therefore, the total cost for the averaging step is:

$$\text{FLOPs}_{\text{averaging}} = D \times \binom{N}{k} \times (k \times n_e)$$

Summing both terms, the total FLOPs for RAGPart is:

$$\text{FLOPs}_{RAGPart} = D \times N \times R + D \times \binom{N}{k} \times (k \times n_e)$$

Example. Suppose we have:

$$R = 10^9 \quad \text{(FLOPs per embedding)}, \quad D = 10^6, \quad N = 5, \quad k = 3, \quad n_e = 512$$

Then:

$$\binom{N}{k} = \binom{5}{3} = 10$$

- **Naive baseline:**

$$\text{FLOPs}_{\text{naive}} = 10^6 \times 10 \times (3 \times 10^9) = 3 \times 10^{16}$$

- **RAGPart:**

$$\text{FLOPs}_{RAGPart} = 10^6 \times 5 \times 10^9 + 10^6 \times 10 \times (3 \times 512) = 5 \times 10^{15} + 1.536 \times 10^{10}$$

Thus, RAGPart reduces the dominant cost (embedding model inference) by a factor of $\sim 6\times$, trading it for a much cheaper averaging step.

C Attack

C.1 AdvRAGgen

We use an instruction-tuned Mistral 7B model (Jiang et al., 2023) as the generator. Given a query and an irrelevant document from the training set, the generator is prompted to paraphrase the document into a retrievable form. Initially, the generator lacks knowledge of what is considered retrievable for a given query. To address this, we fine-tune the generator using three types of feedback (listed below), applying direct preference optimization (DPO) (Rafailov et al., 2024). Specifically, two paraphrases are generated and scored based on the feedback signals, and one is labeled as preferred. These preference pairs are then used to fine-tune the generator in an online DPO setup. The feedback signals are:

- **Generation condition**: To ensure the generation condition is satisfied, the original malicious document must preserve its content when paraphrased. Therefore, we measure the semantic similarity between the original document and the generated response, and use this similarity score as a feedback signal for the generation objective.
- **Retrieval condition**: For the malicious document to be successfully retrieved, it must be semantically similar to the query. To enforce this, we measure the similarity between the query and the document in the retriever's embedding space and use this as a retrieval condition. While this constitutes a white-box attack by definition, we observe that the poisoned examples generated using one retriever model are transferable to others. In our experiments, we generate poisons using the Contriever model (Izacard et al., 2022) and find that they yield high attack success rates (ASR) even when applied to other retrievers.
- **Preventing trivial solution**: In early experiments, optimizing with only the first two feedback signals caused the generator to copy the query into the document, effectively making the generated text identical to the query in the poison setting. To prevent this, we penalize such cases by measuring the Rouge-L score (Lin, 2004) between the query and the generated document, applying a penalty when the score is high.

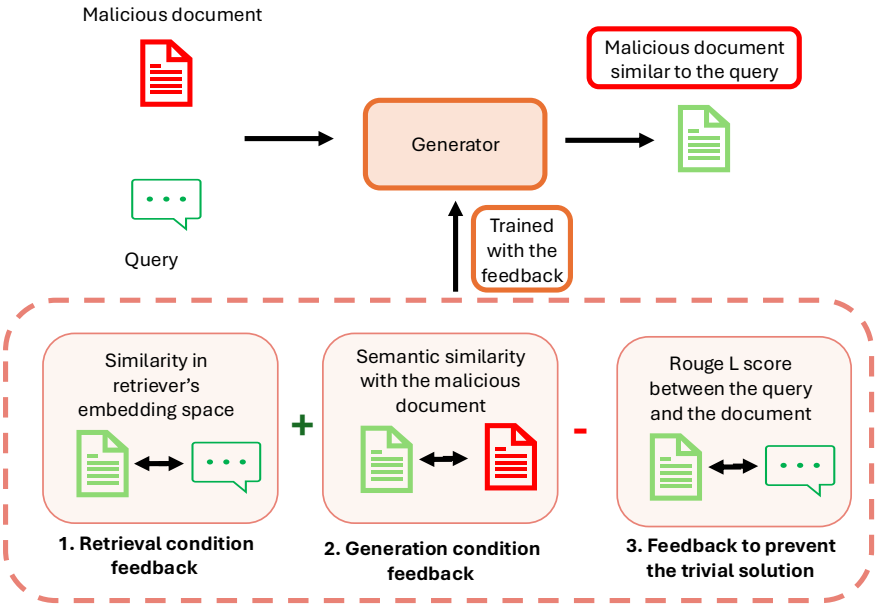


Figure 7: **Pipeline of AdvRAGgen:** The figure illustrates the three feedback signals used to train AdvRAGgen. The *generation condition* ensures similarity between the paraphrased and original malicious documents, the *retrieval condition* enforces similarity between the query and the paraphrased document to enable successful retrieval, and the Rouge-L score serves as a *regularizer* to prevent the trivial solution of copying the query as the poison.

C.2 Ablation Results

Table 6: **ASR Hotflip with iterations (Ablation study)**: This table shows that for certain stronger models such as multilingual e5 (Wang et al., 2024) more iterations of the hotflip method is needed to produce an efficient poison and still the proposed defenses are capable of defending against the attack.

	Natural Questions (NQ) (Kwiatkowski et al., 2019)								
	ASR (%)								
Retriever	Defense	itr=30	itr=40	itr=50	itr=60	itr=70	itr=80	itr=90	itr=100
Multilingual E5 (Wang et al., 2024)	No Defense	16	50	50	78	82	88	88	90
	RAGPart (N = 5, k=1)	0	0	0	0	0	0	0	2
	RAGPart (N = 5, k=3)	0	0	0	0	0	0	1	0
	RAGMask	2	2	0	2	6	2	4	2

D Defense

D.1 Ablation: Methodology

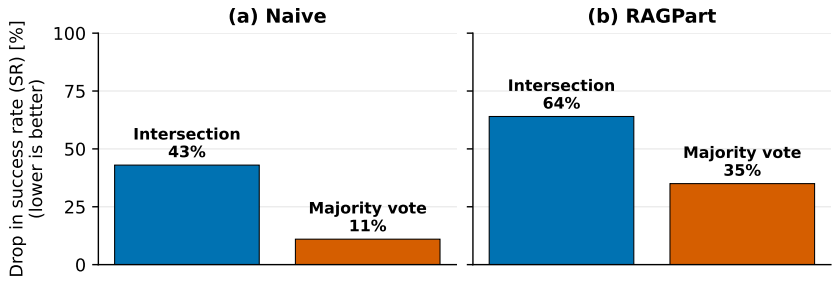


Figure 8: **Drop in success rate (FiQA; lower is better).** Intersection-based aggregation causes a larger SR drop than majority vote for both **Naive** and **RAGPart**.

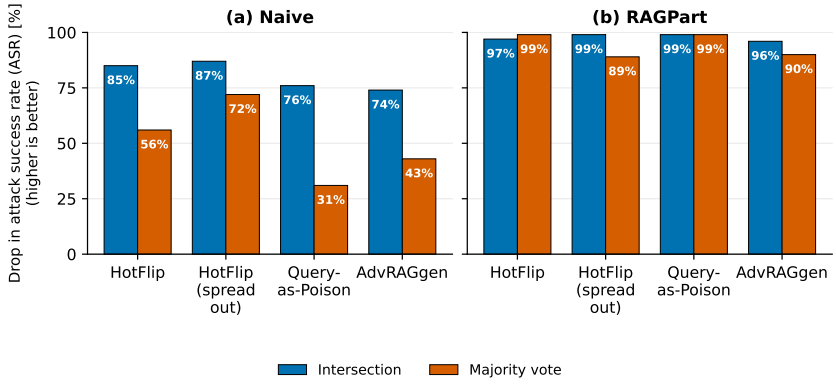


Figure 9: **Drop in attack success rate (ASR) on FiQA (higher is better).** For naive fragment combination, strong ASR reduction often requires intersection-based aggregation, harming utility. In contrast, **RAGPart** remains robust under both aggregation strategies.

D.2 Major Results

Table 7: **Success Rate**: This table shows the performance of RAGPart and RAGMask in benign retrieval scenarios. Here both the methods were able to preserve the utility on benign retrieval.

	Natural Questions (NQ) (Kwiatkowski et al., 2019)				**FiQA** (Maia et al., 2018)			
	SR ↑ (%)				SR ↑(%)			
Retriever	No Defense	RAGPart (N = 5, k = 1)	RAGPart (N = 5, k = 3)	RAGMask (m=1)	No Defense	RAGPart (N = 5, k = 1)	RAGPart (N = 5, k = 3)	RAGMask (m=1)
Contriever (Izacard et al., 2022)	**78**	55	54	69	**58**	31	36	41
ANCE (Xiong et al., 2020)	**55**	20	18	35	**33**	15	17	20
Multilingual E5 (Wang et al., 2024)	**94**	65	76	80	**79**	53	60	67
GTE Large (Li et al., 2023)	**69**	32	49	59	**64**	28	43	47

Table 8: **Attack Success Rate**: This table shows the ability of RAGPart and RAGMask in defending against different types of attacks. Here both the methods were able to reduce the success rate across different types retrieval based attacks.

		Natural Questions (NQ) (Kwiatkowski et al., 2019)				FiQA (Maia et al., 2018)			
		ASR ↓ (%)				ASR ↓ (%)			
Retriever	Attack Type	No Defense	RAGPart (N=5, k=1)	RAGPart (N=5, k=3)	RAGMask (m=10)	No Defense	RAGPart (N=5, k=1)	RAGPart (N=5, k=3)	RAGMask (m=10)
Contriever (Izacard et al., 2022)	HotFlip	87	**2**	0	7	95	**2**	0	1
	HotFlip (spread out)	85	5	4	9	92	2	6	3
	Query-as-Poison	78	2	3	4	85	9	1	**3**
	AdvRAGgen	91	10	6	8	94	13	11	7
ANCE (Xiong et al., 2020)	HotFlip	61	**0**	1	4	78	**0**	0	3
	HotFlip (spread out)	45	1	0	6	67	6	1	3
	Query-as-Poison	68	0	0	8	65	5	0	**3**
	AdvRAGgen	79	6	3	8	86	6	4	5
Multilingual E5 (Wang et al., 2024)	HotFlip	18	**0**	0	1	21	**0**	1	0
	HotFlip (spread out)	15	0	8	3	26	0	0	7
	Query-as-Poison	73	3	0	9	45	**9**	0	7
	AdvRAGgen	81	8	7	4	95	15	14	7
GTE Large (Li et al., 2023)	HotFlip	10	**0**	0	1	20	**0**	0	0
	HotFlip (spread out)	16	1	1	4	28	0	2	0
	Query-as-Poison	63	2	2	4	58	5	1	**0**
	AdvRAGgen	70	6	5	7	83	8	6	5

D.3 Hyperparameter Analysis: Effect of N, k in RAGPart

Table 9: **Hyperparameter analysis on RAGPart (SR %)**: In this table we show effect of N, k in the RAGPart in utility preservation. This ablation showcases the drawbacks of using larger N which can lead to degradation the of the utility thus highlighting the importance of RAGPart as opposed to naive aggregation. The original SR is **74**%. Experiments were done on the FiQA dataset.

	k =1	k = 3	k = 5	k = 10	k =15	k = 20
N = 5	53	60	60	N/A	N/A	N/A
N = 10	44	45	41	39	N/A	N/A
N = 15	34	33	30	28	28	N/A
N = 20	26	26	24	24	22	21

Table 10: **Hyperparameter analysis on RAGPart (ASR** $\%$**)**: In this table we show effect of N, k in the RAGPart in defense. Using higher k can lead to better defense against the attacks. Here the original attack success rate was at **95**$\%$ under an AdvRAGgen semantic attack. This signifies the capability of the defense. Furthermore, RAGPart was able to afford a larger k which can result in lower computational overhead. Experiments were done on the FiQA dataset.

	k =1	k = 3	k = 5	k = 10	k =15	k = 20
N = 5	15	14	7	N/A	N/A	N/A
N = 10	19	14	5	1	N/A	N/A
N = 15	15	15	8	1	1	N/A
N = 20	20	9	4	1	1	1

D.4 Hyperparameter Analysis: Effect of m, δ in RAGMask

Table 11: **Hyperparameter analysis on RAGMask (SR** $\%$**)**: In this table we show effect of δ, m in the RAGMask in utility preservation. The original SR is **74**$\%$. Experiments were done on the FiQA dataset. Even though at larger δ RAGMask was able to preserve utility as seen below it lead to a lesser effectiveness against stronger attacks such the proposed AdvRAGgen.

	m = 10	m = 15	m = 20	m = 25
δ = 0.01	66	65	66	66
δ = 0.05	73	73	72	71
δ = 0.1	74	74	74	74
δ = 0.5	74	74	74	74

Table 12: **Hyperparameter analysis on RAGMask (ASR** $\%$**)**: In this table we show effect of δ, m in the RAGMask in defense. Here the original attack success rate was at **95**$\%$ under an AdvRAGgen semantic attack. Larger mask sizes m were shown to be ideal under smaller δ. Experiments were done on the FiQA dataset.

	m = 5	m = 10	m = 15	m = 20	m = 25
δ = 0.01	11	7	5	6	6
δ = 0.05	44	28	18	12	9
δ = 0.1	55	49	35	27	22
δ = 0.5	57	57	57	57	57

D.5 Hyperparameter Analysis: Effect of RAGPart vs Naive combination of fragments

Table 13: **RAGPart SR - RAGPart vs Naive combination of fragments** in FiQA: The table shows that under majority voting aggregation, RAGPart better preserves utility compared to a naive combination across multiple retrievers.

	FiQA (Maia et al., 2018)		
	SR ↑ (%)		
Retriever	No Defense	Naive combination of fragments (N=5, k=3)	RAGPart (N=5, k=3)
Contriever (Izacard et al., 2022)	**58**	35	36
ANCE (Xiong et al., 2020)	**33**	17	18
Multilingual E5 (Wang et al., 2024)	**79**	48	60
GTE Large (Li et al., 2023)	**64**	34	43

D.6 Effect of Aggregation methods in Naive combination of fragments

Table 14: **Naive combination of fragments SR - Intersection based aggregation vs majority vote based aggregation** in FiQA: While intersection-based aggregation offers better efficiency against attacks, it also leads to a significant drop in utility, making it a less ideal choice for aggregation.

	FiQA (Maia et al., 2018)		
	SR ↑ (%)		
Retriever	No Defense	Naive combination (N=5, k=3) **Intersection based aggregation**	Naive combination (N=5, k=3) **Majority vote based aggregation**
Contriever (Izacard et al., 2022)	**58**	35	52
ANCE (Xiong et al., 2020)	**33**	17	27
Multilingual E5 (Wang et al., 2024)	**79**	48	74
GTE Large (Li et al., 2023)	**64**	34	52

Table 15: **Naive combination of fragments ASR - Intersection based aggregation vs voting based aggregation** in FiQA: Even though majority voting based aggregation results in better utility preservation under naive combination due to it's lack of additional robustness as in RAGPart it ends up being ineffective as a defense thus making the naive combination of fragments as an impractical version of defense against retrival poisoning.

		FiQA (Maia et al., 2018)		
		ASR ↓ (%)		
Retriever	Attack	No Defense	Naive combination N=5, k=3) **Intersection based aggregation**	Naive combination N=5, k=3) **Voting based aggregation**
Contriever (Izacard et al., 2022)	HotFlip	**95**	23	76
	HotFlip (spread out)	**92**	21	47
	Query-as-Poison	**85**	20	67
	AdvRAGgen	**94**	24	77
ANCE (Xiong et al., 2020)	HotFlip	**78**	18	50
	HotFlip (spread out)	**67**	7	18
	Query-as-Poison	**65**	15	42
	AdvRAGgen	**89**	20	58
Multilingual E5 (Wang et al., 2024)	HotFlip	**21**	2	3
	HotFlip (spread out)	**26**	3	4
	Query-as-Poison	**45**	14	36
	AdvRAGgen	**95**	33	66
GTE Large (Li et al., 2023)	HotFlip	**20**	0	3
	HotFlip (spread out)	**28**	2	5
	Query-as-Poison	**58**	9	29
	AdvRAGgen	**83**	15	57

Table 16: **RAGPart SR - Intersection based aggregation vs Majority vote based aggregation** in FiQA: This table showcases the ineffectiveness of intersection based aggregation methods as in the case of naive combination. The overly conservative nature of this aggregation makes it impractical.

	FiQA (Maia et al., 2018)		
	SR ↑ (%)		
Retriever	No Defense	RAGPart (N=5, k=3) **Intersection based aggregation**	RAGPart (N=5, k=3) **Majority vote based aggregation**
Contriever (Izacard et al., 2022)	**58**	19	36
ANCE (Xiong et al., 2020)	**33**	10	18
Multilingual E5 (Wang et al., 2024)	**79**	32	60
GTE Large (Li et al., 2023)	**64**	25	43

Table 17: **RAGPart ASR - Intersection based aggregation vs voting based aggregation** in FiQA: Under RAGPart due to the added robustness in embedding space under both intersection and majority vote based aggregation RAGPart was able defend effectively. But due to the impractical nature of the aggregation in utility preservation we chose majority voting as the ideal aggregation method.

		FiQA (Maia et al., 2018)		
		ASR ↓ (%)		
Retriever	Attack	No Defense	RAGPart (N=5, k=3) **Intersection based aggregation**	RAGPart (N=5, k=3) **Voting based aggregation**
Contriever (Izacard et al., 2022)	HotFlip	**95**	5	1
	HotFlip (spread out)	**92**	2	6
	Query-as-Poison	**85**	1	1
	AdvRAGgen	**94**	3	11
ANCE (Xiong et al., 2020)	HotFlip	**78**	0	0
	HotFlip (spread out)	**67**	0	1
	Query-as-Poison	**65**	0	0
	AdvRAGgen	**86**	1	4
Multilingual E5 (Wang et al., 2024)	HotFlip	**21**	0	0
	HotFlip (spread out)	**26**	0	4
	Query-as-Poison	**45**	0	1
	AdvRAGgen	**95**	6	14
GTE Large (Li et al., 2023)	HotFlip	**20**	0	0
	HotFlip (spread out)	**28**	2	5
	Query-as-Poison	**58**	0	0
	AdvRAGgen	**83**	2	6