Corpus Poisoning via Approximate Greedy Gradient Descent

Jinyan Su[1], **Preslav Nakov**[2], **Claire Cardie**[1]
[1]Department of Computer Science, Cornell University
[2]Mohamed bin Zayed University of Artificial Intelligence
{js3673,ctc9}@cornell.edu, preslav.nakov@mbzuai.ac.ae

Abstract

Dense retrievers are widely used in information retrieval and have also been successfully extended to other knowledge intensive areas such as language models, e.g., Retrieval-Augmented Generation (RAG) systems. Unfortunately, they have recently been shown to be vulnerable to corpus poisoning attacks in which a malicious user injects a small fraction of adversarial passages into the retrieval corpus to trick the system into returning these passages among the top-ranked results for a broad set of user queries. Further study is needed to understand the extent to which these attacks could limit the deployment of dense retrievers in real-world applications. In this work, we propose Approximate Greedy Gradient Descent (AGGD), a new attack on dense retrieval systems based on the widely used HotFlip method for efficiently generating adversarial passages. We demonstrate that AGGD can select a higher quality set of token-level perturbations than HotFlip by replacing its random token sampling with a more structured search. Experimentally, we show that our method achieves a high attack success rate on several datasets and using several retrievers, and can generalize to unseen queries and new domains. Notably, our method is extremely effective in attacking the ANCE retrieval model, achieving attack success rates that are 15.24% and 17.44% higher on the NQ and MS MARCO datasets, respectively, compared to HotFlip. Additionally, we demonstrate AGGD's potential to replace HotFlip in other adversarial attacks, such as knowledge poisoning of RAG systems.

1 Introduction

Dense retrievers, despite their wide application and extensive deployment in real-world systems (Wan et al., 2022; Mitra et al., 2017; Lewis et al., 2020; Guu et al., 2020; Qu et al., 2021), have recently been shown to be vulnerable to various adversarial attacks such as corpus poisoning attacks (Zhong et al., 2023) and data poison attacks (Long et al., 2024; Liu et al., 2023), raising concerns about their security. Given that the corpora used in retrieval systems are often sourced from openly accessible platforms like Wikipedia and Reddit, a concerning scenario arises in which malicious actors can poison the retrieval corpus by injecting some adversarial passages, fooling the system into retrieving these malicious documents rather than the most relevant ones. Such attacks might be used for search engine optimization (Patil Swati et al., 2013) for promoting advertisement, or disseminating disinformation and hate speech.

A conventional approach for such attacks is HotFlip (Ebrahimi et al., 2018), which involves collecting a candidate set for a single randomly sampled token position and finding the best token in the candidate set with which to replace. In addition to corpus poisoning attacks on dense retrieval systems, HotFlip has been widely used in many other settings, such as knowledge poisoning attacks on retrieval augmented generation (RAG) systems (Zou et al., 2024) and adversarial prompt generation (Zou et al., 2023).

In this work, we begin by thoroughly investigating the HotFlip attack on dense retrieval systems to identify its limitations. Based on these insights, we propose a new general attack method called Approximate Greedy Gradient Descent (AGGD). Our experimental results show that AGGD can perform corpus poisoning attacks on dense retrieval systems more effectively, revealing their vulnerability. Though we use corpus attacks on dense retrievers as our primary example, it is important to note that AGGD can replace Hot Flip as a whole in any attack scenarios where HotFlip is applicable.

The main difference between AGGD and Hot Flip is that AGGD uses gradient information more effectively by selecting the top-ranked token from all token positions, rather than over a single randomly sampled position. This approach makes

Findings of the Association for Computational Linguistics: ACL 2025, pages 4274–4294
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

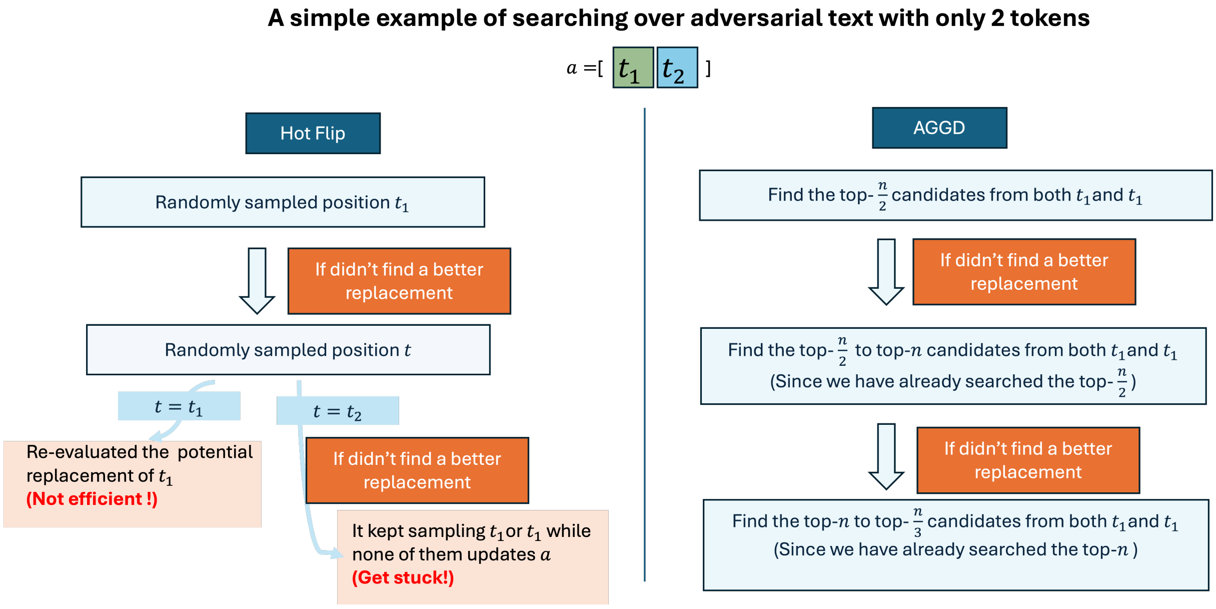

A simple example of searching over adversarial text with only 2 tokens
a =[t_1 t_2]
Hot Flip
Randomly sampled position t_1
If didn't find a better replacement
Randomly sampled position t
$t = t_1$
$t = t_2$
Re-evaluated the potential replacement of t_1
(Not efficient !)
If didn't find a better replacement
It kept sampling t_1 or t_1 while none of them updates a
(Get stuck!)
AGGD
Find the top-$\frac{n}{2}$ candidates from both t_1 and t_1
If didn't find a better replacement
Find the top-$\frac{n}{2}$ to top-n candidates from both t_1 and t_1
(Since we have already searched the top-$\frac{n}{2}$)
If didn't find a better replacement
Find the top-n to top-$\frac{n}{3}$ candidates from both t_1 and t_1
(Since we have already searched the top-n)

Figure 1: A simple example of finding a 2-token sequence a through HotFlip (left) and AGGD (right). If HotFlip can't find a better replacement for the currently sampled token position, there is a $\frac{1}{2}$ probability that it will sample the same token position again and redo the same evaluation, which is inefficient. Moreover, if the potential replacements for another token also don't contain a better option, HotFlip continues to loop through the same search without reducing the loss.

AGGD's search trajectory deterministic, enabling a more structured best-first search. Experimental results demonstrate that AGGD achieves a high attack success rate across various datasets and retrieval models. In summary, our contributions are

- We provide a thorough understanding of the existing HotFlip adversarial attack method, explaining its mechanics and identifying its potential problems.

- We propose AGGD, a gradient-based method that replaces a randomized greedy search with a systematic best-first greedy search over the discrete token space. We demonstrate the effectiveness of AGGD in various settings.

- We conduct extensive experiments to show the vulnerability of dense retrievers under AGGD. For example, when attacking the ANCE retriever, injecting just one adversarial passage can achieve an attack success rate of 80.92% and 65.68% on these datasets, respectively, improving by 15.24% and 17.44% over HotFlip. The generated adversarial passage also possesses the capability to transfer to unseen queries in other domains.

2 Related Work

Dense Retrieval Dense retrievers utilize dense vector representations to capture the semantic information of passages and have demonstrated tremendous effectiveness compared to traditional retrieval systems (Yates et al., 2021). Consequently, they have been employed in many knowledge-intensive areas such as information retrieval (Karpukhin et al., 2020; Gillick et al., 2019; Wu et al., 2020; Wan et al., 2022; Mitra et al., 2017), open-domain question answering and language model pre-training (Lewis et al., 2020; Guu et al., 2020; Qu et al., 2021). For instance, retrieval-augmented generation (RAG) models (Lewis et al., 2020; Guu et al., 2020; Lee et al., 2019) combine language models with a retriever component to generate more diverse, factual and specific content.

Adversarial Attacks in Retrieval Systems Black-hat search engine optimization, which aims to increase the exposure of certain documents through malicious manipulation, poses a threat by reducing the quality of search results and inundating users with irrelevant pages (Castillo et al., 2011; Liu et al., 2023). Previous work has shown that retrieval systems are susceptible to small perturbations: making small edits to a target passage can significantly alter its retrieval rank (Song et al., 2020; Raval and Verma, 2020; Song et al., 2022) for individual or a small set of queries. More recently, a stronger setting known as *corpus poisoning* attack has been studied in (Zhong et al., 2023), where the attack success rate of an adversarial pas-

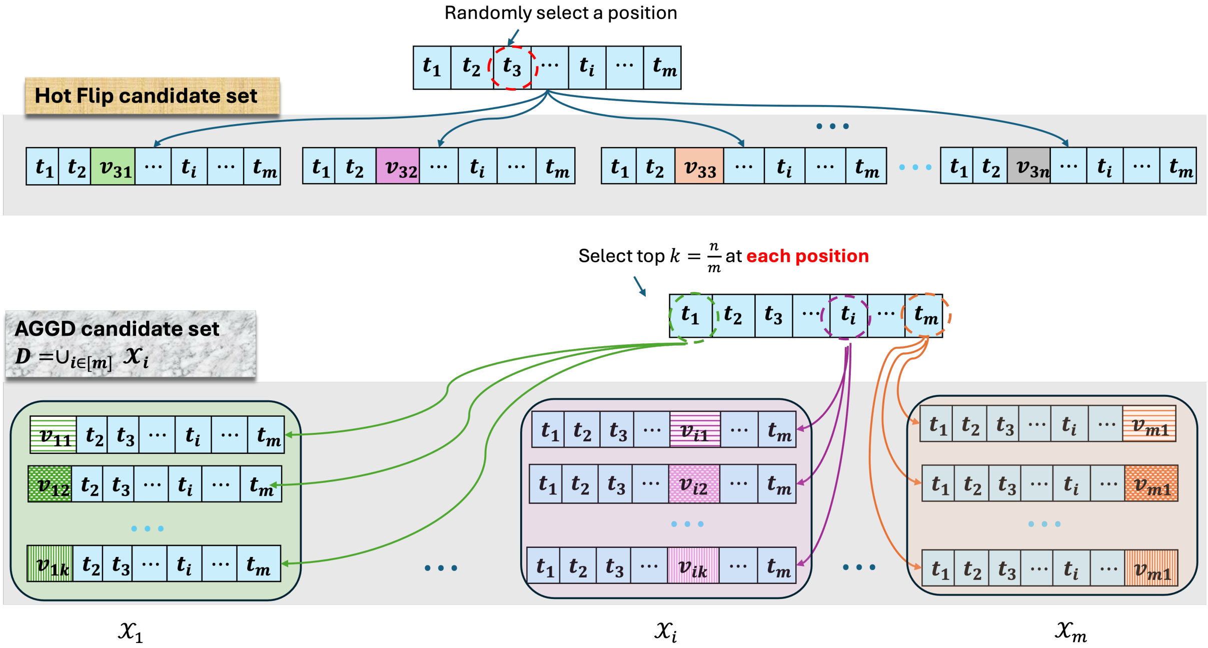

Randomly select a position
Hot Flip candidate set
t_1 t_2 t_3 ⋯ t_i ⋯ t_m
t_1 t_2 v_{31} ⋯ t_i ⋯ t_m
t_1 t_2 v_{32} ⋯ t_i ⋯ t_m
t_1 t_2 v_{33} ⋯ t_i ⋯ t_m
t_1 t_2 v_{3n} ⋯ t_i ⋯ t_m
Select top $k = \frac{n}{m}$ at each position
AGGD candidate set
$D = \cup_{i \in [m]} \mathcal{X}_i$
v_{11} t_2 t_3 ⋯ t_i ⋯ t_m
v_{12} t_2 t_3 ⋯ t_i ⋯ t_m
v_{1k} t_2 t_3 ⋯ t_i ⋯ t_m
t_1 t_2 t_3 ⋯ v_{i1} ⋯ t_m
t_1 t_2 t_3 ⋯ v_{i2} ⋯ t_m
t_1 t_2 t_3 ⋯ v_{ik} ⋯ t_m
t_1 t_2 t_3 ⋯ t_i ⋯ v_{m1}
$\mathcal{X}_1$
$\mathcal{X}_i$
$\mathcal{X}_m$

Figure 2: Illustration of HotFlip (top) and AGGD (bottom) and their candidate sets.

sage is evaluated on unseen queries rather than on targeted given queries. These attacks differ from data poisoning attacks (Long et al., 2024; Liu et al., 2023; Chen et al., 2017; Schuster et al., 2020), as adversarial passages are injected into the retrieval corpus rather than the training data of the retrievers. The retrieval model remains unchanged in a corpus poisoning attack.

Discrete Optimization Many adversarial attacks in NLP involve discrete optimization, whether in classification tasks (Wallace et al., 2019; Ebrahimi et al., 2018; Song et al., 2021), retrieval systems (Jia and Liang, 2017; Song et al., 2020; Raval and Verma, 2020; Song et al., 2022), or adversarial prompt generation (Zou et al., 2023; Shin et al., 2020; Wen et al., 2024). The objective is to find small perturbations to the input to lead the model to make erroneous predictions. Due to the discrete nature of texts, directly applying adversarial attack methods from prior computer vision research (Xiao and Wang, 2021; Tolias et al., 2019) is infeasible. Instead, many methods build upon HotFlip (Ebrahimi et al., 2018) and approximate the effect of replacing a token using gradients.

3 Motivation

In this section we motivate our approach by formalizing the corpus poisoning problem setting (Section 3.1), describing and analyzing the standard HotFlip approach for producing adversarial passages (Zhong et al., 2023) (Section 3.2), and identifying a potential problem with this approach (Section 3.3).

3.1 Corpus Poisoning Problem Setting

In retrieval systems, the retrieval model takes a user query q and returns a ranked list of the k most relevant passages from a large corpus collection $\mathcal{C} = \{p_1, \cdots, p_{|\mathcal{C}|}\}$ consisting of $|\mathcal{C}|$ passages. Compared to sparse retrieval models, which rely on lexical matching, dense retrievers rely on semantic matching. Specifically, the queries and the passages are first represented by d-dimensional dense vectors using a query encoder $E_q(\cdot)$ and passage encoder $E_p(\cdot)$, respectively. Relevance scores can then be computed according to a similarity function. A commonly used similarity function is the dot product of the dense vector representations of the query q and the passage p: $\mathrm{Sim}(q, p) = E_q(q)^T E_p(p)$. Finally, a ranked list of the k most relevant passages $L = [\tilde{d}_1, \tilde{d}_2, \cdots, \tilde{d}_k], (L \subseteq \mathcal{C})$ is returned according to the relevance score.

We consider the problem of corpus attacks on a dense retrieval system, where we design an algorithm to find a small set of adversarial passages $\mathcal{A} = \{a_1, \cdots, a_{|\mathcal{A}|}\}$ that can be retrieved by as many queries as possible for query distribution $\mathcal{P}_Q$. These adversarial passages are then inserted into the corpus $\mathcal{C}$ to fool the dense retrieval models into retrieving passages from $\mathcal{A}$ rather than the most semantically relevant passages from the original corpus $|\mathcal{C}|$. The adversarial passage set $\mathcal{A}$ should be much smaller than the original corpus $\mathcal{C}$. The attack quality of the adversarial passage set $\mathcal{A}$ is typically evaluated based on its *attack success rate*, i.e., the percentage of queries for which at least one adversarial passage appears in the top-k retrieval results.

Formally, the overall objective is to find an adversarial passage a, that maximizes the expected similarity to a query q sampled from distribution $\mathcal{P}_{\mathcal{Q}}$, i.e.,

$$a = \arg\max_{a} \mathbb{E}_{q\sim\mathcal{P}_{\mathcal{Q}}} \text{Sim}(q, a)$$

In practice, we estimate the query distribution using a training set of queries $\mathcal{Q} = \{q_1 \cdot \cdots, q_{|\mathcal{Q}|}\}$, and we aim to find an a with maximal similarity to $\mathcal{Q}$, i.e.,

$$a = \arg\max_{a} \frac{1}{|\mathcal{Q}|} \sum_{q_i \in \mathcal{Q}} \text{Sim}(q_i, a) = \arg\min_{a} \ell(a) \qquad (1)$$

where $\ell(a) = -\frac{1}{|\mathcal{Q}|} \sum_{q_i \in \mathcal{Q}} \text{Sim}(q_i, a)$. The problem setting is realistic in search engines where a malicious user might perform search engine optimization to promote misinformation or spread spam.

Finding the exact solution to the optimization problem (1) is challenging since we are optimizing over a discrete set of inputs (i.e., the tokens in a passage). Additionally, running gradient descent on the embedding space might yield solutions that exist only in the embedding space and deviate significantly from valid texts in the discrete token space. In practice, a straightforward approach that leverages the gradient w.r.t. the one-hot token indicators can be employed to identify a set of promising candidates for replacement (Zou et al., 2023; Ebrahimi et al., 2018; Shin et al., 2020; Zhong et al., 2023). Specifically, we can compute the linearized approximation of replacing the i-th token t_i in a passage a, by evaluating the gradient $\nabla_{e_{t_i}} \ell(a)$, where e_{t_i} denotes the embedding of the token t_i. (Recall that sentence embeddings can be written as function of token embeddings, allowing us to compute the gradient with respect to the token embedding.) This idea has been adopted in many gradient-based search algorithms such as HotFlip (Ebrahimi et al., 2018) for producing adversarial texts, AutoPrompt (Shin et al., 2020) and Greedy Coordinate Gradient (GCG) (Zou et al., 2023) for generating prompts. We revisit the idea of HotFlip in the context of corpus attacks as an example.

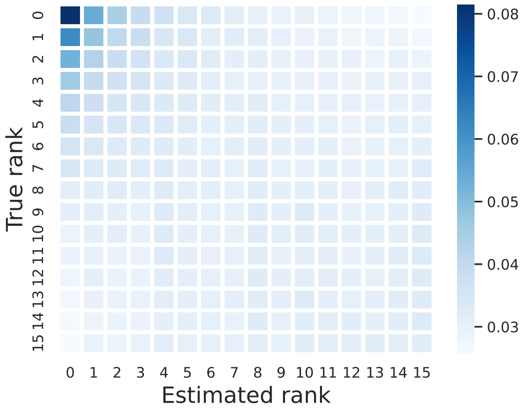


Figure 3: Comparing the true rank of words swapped with their rank according to the gradient-based Taylor approximation. The gradient identifies the top-1 correct token to swap 9% of the time, and guesses within the top ten tokens 58% of the time. (The "true rank" refers to the ranking of tokens based on the actual increase in similarity when a token is swapped, while the "Estimated rank" refers to the ranking of tokens based on the gradient estimate.)

3.2 HotFlip Revisited

In many text-based adversarial attacks, the goal is to find a perturbation of an input sequence — a randomly selected passage a — to optimize some objective function. Due to the discrete nature of text, this is essentially a combinatorial search problem: the attacker searches over a class of perturbations on a such as word swapping or character substitution (Morris et al., 2020) to produce the final adversarial text. The corpus poisoning approach of (Zhong et al., 2023), for example, aims to fool a dense retrieval system to return adversarial passages for a broad set of user queries by transforming corpus passages (i.e., sequences of tokens) into adversarial passages a that exhibit maximal similarity to a query set associated with the corpus.

Unfortunately, even if we fix the token length of a passage $|a|$ to be m, for a language model with vocabulary size $|\mathcal{V}|$, the total size of the search space is $|\mathcal{V}|^m$. For instance, if $m = 30$, and $|\mathcal{V}| = 32,500$, then the total number of sequences to evaluate is about 10^{135}, which is computationally infeasible.

Thus, we need a method to identify a subset of the most promising perturbation sequences. The corpus poisoning attack by (Zhong et al., 2023) addresses this problem using the HotFlip approach in a greedy search over candidate token perturbations. HotFlip (Ebrahimi et al., 2018) first randomly sample a token position, and then determines a can-

Algorithm 1 Approximate Greedy Gradient Descent (AGGD)

```
Input: The token length $m$ of the adversarial passage; Initialized adversarial passage $a = [t_1, \cdots, t_m]$; Total number of iterations $N$; Size of the candidate set $n$; Search depth $d = 0$.
Let $k = \frac{n}{m}$ to be the per token candidate set size.

for $j = 0, 1, 2, \cdots, N$ do
    for $i = 1, \cdots, m$ do
        Let $\mathcal{V}_o = \mathcal{R}(s(a))$ where $s(a) = -e_v^T \nabla \ell(a)$, $\ell(a) = -\frac{1}{|\mathcal{Q}|} \sum_q \text{Sim}(q, a)$ and $\mathcal{R}(\cdot)$ is a rank function.
        Let $\mathcal{X}_i = \mathcal{V}_o[(d-1)k, dk]$ be the truncated top-$(d-1)k$ to top-$dk$ candidates.
    end for
    Construct candidate set at iteration $j$ as $D^{(j)} = \cup_{i \in [m]} \mathcal{X}_i$
    Let $a' = \arg\min_{a' \in D^{(j)}} \ell(a')$ // Check if a better adversarial passage exists in candidate set
    if $\ell(a') < \ell(a)$ then
        Update $a = a'$, reset depth $d = 0$. // Update the adversarial passage if find a better one
    else
        Update $d = d + 1$ // If no update, search from top-$(d-1)k$ to top-$dk$ in next iteration
    end if
end for
Return Final adversarial passage $a$
```

didate set of promising token replacements based on the gradient of their respective one-hot input vector[1] representations. Each iteration of search involves: (1) selecting a random position in a; (2) use gradient information to determine the top-k token perturbation candidates; and (3) applying the perturbation (a token swap) that increases the similarity (decreases the loss) the most.[2]. This process continues for a fixed number of iterations.

3.3 Drawbacks of HotFlip

Although HotFlip usually works well in practice, there are some noticeable problems with this approach. For instance, it is less efficient due to its reliance on randomness for the search. Moreover, it is possible to try all token positions without finding any updates, causing HotFlip to get stuck and repeatedly search the same perturbation candidates. For example, consider an adversarial text with only two tokens (see Figure 1). If we first sample token position t_1 and observe no improvement for all candidates, there is still a $1/2$ probability of sampling the **same** token t_1 in the next iteration. Furthermore, if both token t_1 and token t_2 have been searched and show no improvement, the process would be stuck in a loop.

Motivated by these existing problems, we propose a new algorithm called Approximate Greedy Gradient Descent (AGGD) that uses a deterministic greedy search that makes better use of gradient information by utilizing lower-ranked tokens(i.e., the overall most promising token swap candidates) to improve the quality of the candidate set. As shown in Figure 3, most high quality potential candidates are concentrated in the low-rank area. Suppose we aim to find a text with a token length $m = 30$ by maintaining a candidate set of size $n = 150$. Selecting as the candidate set the top-5 ranked gradients **across all token positions** is likely to result in better quality than selecting the top-150 ranked gradient candidates for only one token position. In the next section, we formally introduce our algorithm Approximate Greedy Gradient Descent (AGGD).

4 Approximate Greedy Gradient Descent

Similar to other gradient-based search algorithms, we fist initialize the adversarial passage to be $a = [t_1, \cdots, t_m]$. We then iteratively update a based on the best candidate that maximizes the similarity over batches of queries. Formally, at each step, we compute a first-order approximation of the change in the loss when swapping the ith token in a with another token $v \in \mathcal{V}$. In contrast to Hotflip, AGGD identifies a candidate set for each of the m tokens in a: for each token i, the candidate set contains the top $k = \frac{n}{m}$ ranked tokens from $\mathcal{V}$ according to the scoring function $s(v) = -e_v^T \nabla \ell(a)$, i.e., $\mathcal{X}_i = \text{top-}k_{v \in \mathcal{V}}[-e_v^T \nabla \ell(a)]$, where e_v is the token embedding and the gradient is taken over the the embedding of the current adversarial passage a. Combining all the $\mathcal{X}_i$ leads to our overall candidate set $D^{(j)} = \cup_{i \in [m]} \mathcal{X}_i$ (as illustrated in Figure 2). For each candidate in the set D, the loss is re-evaluated and a is updated to the candidate with lowest loss for the next step. This requires n passes of the model, which constitutes the primary computational effort. If a doesn't update at iteration i, i.e.,

[1] The embedded version of the token is used.

[2] If no candidate increases the loss, no change to a is made.

Dataset	**Method**	**Retriever**				
		Contriever	Contriever-MS	ANCE	DPR-mul	DPR-nq
NQ	AGGD	**92.5(2.68)**	**63.45(8.68)**	**80.92(4.82)**	**6.88(1.72)**	**2.19(0.98)**
	Hot Flip	91.08(0.9)	58.43(4.53)	65.68(2.5)	5.4(0.36)	2.03 (0.25)
	Random	80.24(0.92)	32.5(2.61)	31.0(4.3)	3.7(0.93)	1.66 (0.14)
MS MARCO	AGGD	**85.47(3.81)**	**24.42(17.44)**	**93.6(1.01)**	**12.79(3.86)**	5.23 (1.01)
	Hot Flip	83.72(5.93)	22.67(12.01)	76.16(9.21)	9.88(1.93)	**5.81(2.01)**
	Random	66.86(7.24)	13.95(13.46)	49.42(11.2)	6.4(2.53)	2.91 (1.93)

Table 1: In-domain attack success rate (ASR) of AGGD on NQ and MS MARCO datasets with 5 retrievers by injecting 1 adversarial passage. We highlight the best performing attacking method in bold (Higher ASR indicates better attack performance). Results are from 4 random runs with standard deviation in parenthesis.

Target Domain	**Source Domain**	**Methods**	**Retriever**				
			Contriever	Contriever-MS	ANCE	DPR-mul	DPR-nq
FiQA-2018	NQ	AGGD	64.93(15.9)	4.94(1.68)	6.33(1.91)	0.15(0.11)	0.12(0.13)
		Hot Flip	69.68(8.66)	3.05(0.6)	4.13(0.77)	0.46(0.55)	0.62(0.15)
		Random	41.05(2.73)	0.35(0.13)	1.66(0.49)	0.27(0.17)	0.46(0.19)
	MS MARCO	AGGD	**86.54(1.88)**	**15.66(22.54)**	**17.98(1.44)**	**3.59(1.34)**	**3.36(1.79)**
		Hot Flip	78.51(6.96)	4.59(2.89)	11.5(0.79)	2.16(1.13)	3.2(1.72)
		Random	62.81(6.07)	0.42(0.57)	5.48(1.36)	0.73(0.2)	0.93(0.24)
NFCorpus	NQ	AGGD	46.83(8.1)	17.1(4.3)	37.69(4.2)	8.51(1.29)	12.23(0.46)
		Hot Flip	42.34(10.96)	10.91(2.77)	34.68(1.71)	10.22(2.62)	12.85(0.99)
		Random	24.38(5.02)	6.58(1.43)	18.34(4.09)	7.43(1.3)	11.3(0.56)
	MS MARCO	AGGD	**52.55(13.94)**	**18.73(16.3)**	**69.81(3.57)**	**26.16(4.51)**	**28.17(4.02)**
		Hot Flip	49.92(5.26)	14.24(8.46)	52.55(6.86)	19.27(2.31)	23.06(7.02)
		Random	30.03(6.42)	6.42(6.45)	42.26(6.23)	14.47(3.59)	14.78(1.06)
Quora	NQ	AGGD	73.92(12.16)	**25.1(7.03)**	77.72(5.57)	4.85(1.13)	6.39(2.1)
		Hot Flip	78.99(3.56)	18.92(5.84)	72.37(2.13)	3.36(1.02)	7.2(1.45)
		Random	49.44(4.18)	6.12(0.63)	53.09(4.89)	2.52(0.47)	6.66(1.51)
	MS MARCO	AGGD	**86.92(2.49)**	18.83(21.47)	**91.11(1.48)**	**22.29(4.24)**	**25.96(5.05)**
		Hot Flip	83.26(5.87)	12.82(7.64)	84.18(2.25)	15.38(4.14)	23.07(3.67)
		Random	62.57(6.63)	3.81(4.12)	74.06(0.75)	8.82(1.14)	12.29(3.33)
SCIDOCS	NQ	AGGD	21.05(7.78)	11.95(7.99)	9.78(1.97)	0.88(0.29)	0.38(0.08)
		Hot Flip	**27.02(13.4)**	**14.32(4.48)**	7.1(0.87)	1.18(0.4)	0.35(0.11)
		Random	12.88(3.76)	0.78(0.61)	3.05(0.93)	0.6(0.37)	0.3(0.07)
	MS MARCO	AGGD	23.22(6.99)	14.3(20.69)	**31.9(7.2)**	**3.88(1.62)**	**1.08(0.42)**
		Hot Flip	24.98(6.33)	12.25(7.52)	21.88(4.33)	3.12(1.4)	0.85(0.32)
		Random	15.0(1.53)	1.27(1.83)	10.62(5.11)	2.48(1.38)	0.2(0.07)
SciFact	NQ	AGGD	22.92(4.46)	1.0(0.62)	6.75(1.92)	-	-
		Hot Flip	24.5(12.16)	0.92(0.72)	3.58(0.98)	0.17(0.29)	-
		Random	8.25(3.42)	-	0.5(0.37)	-	-
	MS MARCO	AGGD	**29.08(10.43)**	**2.67(4.43)**	**30.33(6.47)**	**2.42(0.6)**	0.5(0.29)
		Hot Flip	21.58(6.52)	1.25(1.11)	16.5(4.78)	2.0(0.24)	**0.67(0.71)**
		Random	9.83(5.46)	-	7.33(4.96)	1.17(0.96)	0.17(0.17)

Table 2: Out-of-domain top-20 attack success rate with only 1 adversarial passage. Results are averaged over 4 random runs with standard deviation shown in parenthesis. The combinations of attack and source dataset that achieve the highest ASR on each target domain and retriever are highlighted in bold.

no candidate in $D^{(j)}$ achieves a lower loss, then in the next iteration, instead of searching over the top $k = \frac{n}{m}$, we search over the second tier of candidates, i.e., tokens from $\mathcal{V}$ with scores between top $\frac{n}{m}$ and the top $\frac{2n}{m}$, since the top $\frac{n}{m}$ candidates were already evaluated in the previous iterations. The search proceeds methodically as described above until a better candidate is found and a is updated. The whole process is described in Algorithm 1.

5 Experiments

5.1 Experimental Details

Datasets We primarily use two popular question-answering datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019) and MS MARCO (Nguyen et al., 2016) for our attack. NQ, containing 132,803 question-answer pairs, is collected from Wikipedia, while MS MARCO, containing 532,761 question answer pairs, is collected from web documents. For in-domain attacks, we evaluate the attack on the unseen queries on held-out test queries of these two datasets. To test the transferability of our attack, we also evaluate it on 5 out-of-domain datasets: NFCorpus (Boteva et al., 2016), Quora, SCIDOCS (Cohan et al., 2020), SciFact (Wadden et al., 2020), FiQA-2018 (Maia et al., 2018). These datasets contain unseen queries and corpora that are out of distribution or from entirely different domains such as biomedicine, scientific articles, and finance. Statistics of these datasets can be found in Appendix C.1.

Retrievers In our main experiments, we conduct attacks on 5 state-of-the-art retrieval models: Contriever, Contriever-MS (Contriever fine-tuned on MS MARCO) (Gautier et al., 2022), DPR-nq (trained on NQ), DPR-mul (trained on multiple datasets) (Karpukhin et al., 2020) and ANCE (Xiong et al., 2020).

Evaluation Metrics After generating the adversarial passages on the training set and injecting them into the corpus, we evaluate the effectiveness of our attack using the top-k *attack success rate (ASR)* on test queries. Top-k ASR is defined as the percentage of queries for which at least one adversarial passage is retrieved in the top-k_r results, i.e., ASR $= \frac{1}{n_q}\sum_{i=1}^{n_q} \mathbb{1}\{a \in \mathcal{R}_r(q_i^{\text{test}}, k_r, \mathcal{C}_{\text{test}})\}$, where n_q is the total number of test queries and $\mathcal{R}_r(q_i^{\text{test}}, k_r, \mathcal{C}_{\text{test}})$ is the retriever that returns the top-k_r most relevant passages for the test query q_i^{test}. $\mathbb{1}\{a \in \mathcal{R}_r(q_i^{\text{test}}, k_r, \mathcal{C}_{\text{test}})\}$ is the indicator function, which equals to 1 if $a \in \mathcal{R}_r(q_i^{\text{test}}, k_r, \mathcal{C}_{\text{test}})$ and 0 otherwise. A higher ASR indicates that the model is more vulnerable to attacks, and thus, the attack is more effective. We use $k_r = 20$ to present our result. Since ASR depends on the size the the test corpus, to make fair comparisons across different dataset, we randomly sample $|\mathcal{C}_{\text{test}}| = 10,000$ passages from the overall corpus pool.

Additionally, we use *retrieval accuracy* on the validation data, which is defined as

$$\text{RetAcc} = \frac{1}{n_{\text{val}}}\sum_{i=1}^{n_{\text{val}}} \mathbb{1}\{\text{Sim}(q_i^{\text{val}}, p_i^{\text{val}}) > \text{Sim}(q_i^{\text{val}}, a)\}$$

where a is the adversarial passage, n_{val} is the total number of query-passage pairs in the validation set, and p_i^{val} is most semantically relevant passage that should have been retrieved by query p_i^{val}. Lower retrieval accuracy indicates a higher chance of the model being fooled into choosing the adversarial texts a over the most semantically relevant passage in the corpus. Note that the Success Rate **during training** (on validation data) can be defined through retrieval accuracy, which is simply $1 - \text{RetAcc}$.

Hyperparameters For all baselines, we use adversarial passages of $m = 30$ tokens and perform the token replacement for 2000 steps. We fix the candidate set size at $n = 150$. All the experiments are conducted on NVIDIA A100 GPU (with total memory 40G).

5.2 Main Results

In-Domain Attack We show the in-domain attack performance of AGGD and the other two baselines (HotFlip and random perturbation) in Table 1, evaluating the injection of only one adversarial passage. Our findings are as follows: (1) The pretrained Contriever model is more vulnerable to attacks. All three attack baselines achieve the highest ASR compared to other retrieval models. Poisoning with AGGD achieves 92.5% ASR on NQ dataset and 85.47% on MS MARCO dataset, respectively. Even using Random perturbation can achieve a relatively high ASR of 80.24% on NQ dataset and 66.86% on MS MARCO dataset. (2) Besides achieving comparable results in Contriever and DPR, AGGD is extremely effective in attacking ANCE, improving over the second-best baseline by 15.24% and 17.44% on NQ and MS MARCO, respectively. The effectiveness of AGGD in attack ANCE can also be clearly observed during the training. Due space limitations, we provide training accuracy in Appendix A.2 (Figure 10). (3) Supervised retriever models such as DPR are more challenging to attack. Attacking DPR-nq on NQ with only 1 adversarial passage are just slightly better than random perturbation.

Out-of-Domain Attack We found that the generated adversarial passages can transfer across different domains. In Table 2, we use adversarial passages generated from training set of NQ and MS MARCO and insert them into the corpora of retrieval tasks in other domains. We found that (1) Compared to adversarial passages generated from NQ dataset, those trained from MS MARCO generally perform better in out-of-domain attacks, possibly because MS MARCO contains more training data. (2) Contriever models are still the most vulnerable to corpus poisoning attacks. For example, inserting a single adversarial passage generated by AGGD into FiQA-2018 achieves a top-$k_r = 20$ ASR of 86.54%, and inserting it into Quora can trick the model into returning the adversarial pas-

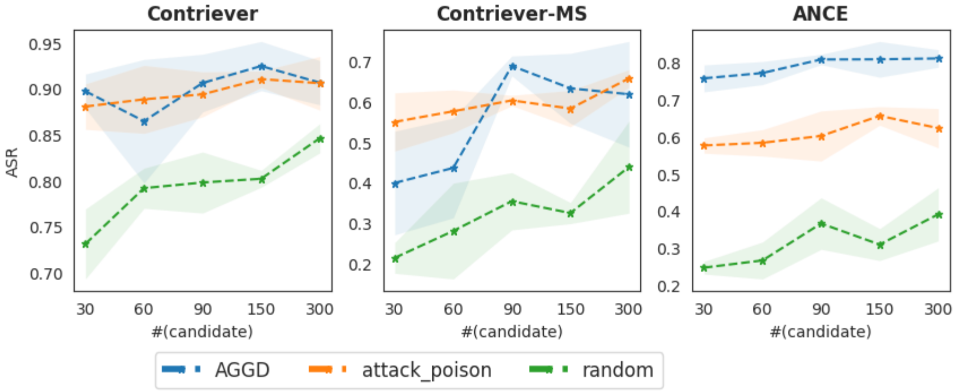
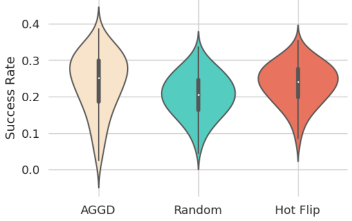
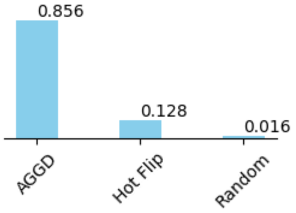


Figure 4: The effect of candidate set size n on the attack success rate.



(a) Attack Success rate of candidates sets collected by different methods. (Averaged over 400 candidate sets sampled.)



(b) The proportion of times the best candidate occurs in the candidate sets collected by AGGD, HotFlip, and Random.

Figure 5: Experiments on Contriever with NQ dataset illustrate that the candidate set collected by AGGD has higher overall quality (left) and is more likely to contain the best candidate (right).

sage in the top-20 retrieved passage for 86.92% of the queries. (3) Quora are easier domains to attack. AGGD achieves over 20% ASR even in DPR-mul and DPR-nq, which is surprising since, as shown in the in-domain attack results in Table 1, attacking DPR-mul and DPR-nq is extremely hard , even with adversarial passage trained from in-domain data. (Though NFCorpus has high attack success rate, NFCorpus also has a smaller testing corpus).

5.3 Analysis and Ablation Study

Candidate Set Quality Our results indicate that candidate sets selected using AGGD have better quality than those selected using HotFlip and Random perturbation. To demonstrate this, we randomly sampled a passage and used AGGD, HotFlip and random perturbation to select their respective candidate sets with $m = 150$ candidates. We then evaluated the success rate (i.e., $1 - \text{RetAcc}$) on the validation set for all the candidates. The results, averaged from 400 random samples, are shown in Figure 5a. We can observe that the candidate set from AGGD has higher mean success rate on validation data. Additionally, the higher confidence bound in the AGGD candidate set signifies that AGGD's candidate set not only has higher overall quality, but is also more likely to contain the best candidate when compared horizontally across candidate sets from other methods. To verify this, we counted how often the best candidate occurs in candidate sets selected by these methods and found that more than 85% of the time, the AGGD candidate set contains the best candidate, while less than 13% show up in HotFlip candidate set, as shown in Figure 5b. More experiments on other retrievers models further verify that the AGGD candidate set has higher quality (Figure 13 in Appendix B.1).

The Effects of Candidate Set Size n

In Figure 4, we show how the candidate set size n affects the ASR evaluated on NQ dataset with three retrievers. Generally, we can observe that increasing size of the candidate set improves the ASR. The effect of the candidate set size would be more pronounced when a larger range of candidate set sizes is considered.

The effect of the token length m

In Figure 6, we show the attack success rate for varying token lengths $m = \{5, 10, 25, 50, 100\}$ while fixing the candidate set size $n = 100$. Generally, we find that larger token lengths result in higher attack success rate upon convergence. This is intuitive since larger m indicates a larger subspace in the continuous dense embedding space,

	LLaMa-2-7B	LLaMa-2-13B	Vicuna-7B	Vicuna-13B	Vicuna-33B	GPT-3.5	GPT-4
MS MARCO							
AGGD	**0.81(0.00)**	**0.79(0.01)**	**0.73(0.01)**	0.72(0.01)	0.67(0.01)	**0.80(0.01)**	**0.88(0.01)**
HotFlip	0.80(0.02)	0.77(0.02)	0.69(0.02)	**0.73(0.04)**	**0.69(0.02)**	0.79(0.01)	0.86(0.01)
NQ							
AGGD	**0.85(0.00)**	0.94(0.00)	**0.81(0.00)**	**0.82(0.00)**	**0.68(0.00)**	**0.92(0.00)**	**0.97(0.00)**
HotFlip	0.83(0.00)	**0.95(0.00)**	0.80(0.00)	0.81(0.00)	0.67(0.00)	0.87(0.01)	0.96(0.01)

Table 3: Comparing Attack success rate of PoisonedRAG (white-box) using HotFlip and AGGD to find the adversarial texts.

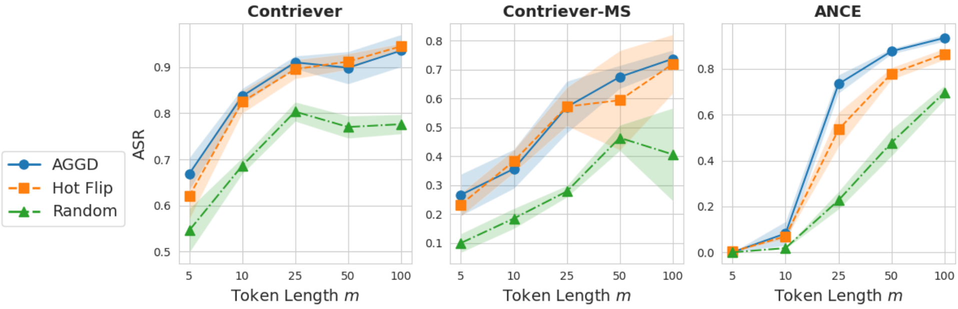


Figure 6: The effect of the number of tokens, with fixed candidate set size $n = 100$ and varying adversarial passage token length $m = \{5, 10, 25, 50, 100\}$.

thus providing a higher *lower bound* on the effectiveness of the adversarial passage we can find.

5.4 Extending AGGD to Knowledge Poisoning Attacks

Similar to Hot Flip, which can be used for many other adversarial attacks such as adversarial prompt generation (Zou et al., 2023) and knowledge poisoning in RAG (Zou et al., 2024). AGGD can also be conveniently used as a plug-in replacement for HotFlip in these attacks. In Table 3, we experiment with PoisonedRAG (Zou et al., 2024) in their white box setting, where Hot Flip was originally used to craft adversarial texts. The details in reproducing Table 3 are given in Appendix D. We find that AGGD achieves a comparable or better attack success rate than Hot Flip in both MS MARCO and NQ dataset on multiple LLMs.

6 Conclusion

In this paper, we propose a new adversarial attack method called AGGD, a gradient-based search algorithm that systematically and structurally finds potential perturbations to optimize the objective function. We use the corpus poisoning attacks as the main example to demonstrate the effectiveness of our algorithm. Experiments on multiple datasets and retrievers show that the proposed approach is effective in corpus poisoning attacks, achieving high attack success rate in both in-domain and out-of-domain scenarios, even with an extremely low poison rate. Additional experiments on other adversarial attacks indicate the potential of AGGD as a competitive alternative to the widely used HotFlip.

Acknowledgment

We thank Jack Morris for his valuable feedback and discussion.

Limitations and Future Work

While we use corpus poisoning attacks to showcase our AGGD algorithm, the proposed attack framework is versatile and applicable to a wide range of adversarial attack scenarios, as many of these can be formulated as discrete optimization problems. However, the generated output sequences often lack semantic coherence, making the adversarial corpus easily detectable and filterable. A promising direction for future work is to reformulate the problem as a constrained optimization problem, focusing on producing semantically meaningful adversarial passages that are more difficult to defend against, even if this may compromise the overall attack success rate.

Ethics Statement

Our work studies the vulnerability of dense retriever and corpus poison attack. The propose attack AGGD shows higher attach success rate especially for ANCE model, compared to previous HotFlip attack, which could be used for spread garbage information. Future research based on this paper should be exercised with caution and consider the potential consequences.

References

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 6th International Conference of the CLEF Association, CLEF'15, Toulouse, France, September 8-11, 2015, Proceedings 6*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*, pages 716–722. Springer.

Carlos Castillo, Brian D Davison, et al. 2011. Adversarial web search. *Foundations and trends® in information retrieval*, 4(5):377–486.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Izacard Gautier, Caron Mathilde, Hosseini Lucas, Riedel Sebastian, Bojanowski Piotr, Joulin Armand, and Grave Edouard. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Dan Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

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

Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Black-box adversarial attacks against dense retrieval models: A multi-view contrastive learning method. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1647–1656.

Quanyu Long, Yue Deng, LeiLei Gan, Wenya Wang, and Sinno Jialin Pan. 2024. Backdoor attacks on dense passage retrievers for disseminating misinformation. *arXiv preprint arXiv:2402.13532*.

Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. 2023. Making llms worth every penny: Resource-limited text classification in banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 392–400.

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*, pages 1941–1942.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th international conference on world wide web*, pages 1291–1299.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

P Patil Swati, BV Pawar, and S Patil Ajay. 2013. Search engine optimization: A study. *Research Journal of Computer and Information Technology Sciences*, 1(1):10–13.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847.

Nisarg Raval and Manisha Verma. 2020. One word at a time: adversarial attacks on retrieval models. *arXiv preprint arXiv:2008.02197*.

Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R Hersh. 2020. Trec-covid: rationale and structure of an information retrieval shared task for covid-19. *Journal of the American Medical Informatics Association*, 27(9):1431–1436.

Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. 2020. Humpty dumpty: Controlling word meanings via corpus poisoning. In *2020 IEEE symposium on security and privacy (SP)*, pages 1295–1313. IEEE.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Congzheng Song, Alexander Rush, and Vitaly Shmatikov. 2020. Adversarial semantic collisions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4198–4210, Online. Association for Computational Linguistics.

Junshuai Song, Jiangshan Zhang, Jifeng Zhu, Mengyun Tang, and Yong Yang. 2022. Trattack: Text rewriting attack against text retrieval. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 191–203.

Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. 2021. Universal adversarial attacks with natural triggers for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3724–3733.

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

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Giorgos Tolias, Filip Radenovic, and Ondrej Chum. 2019. Targeted mismatch adversarial attack: Query with a flower to retrieve the tower. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5037–5046.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162.

Hui Wan, Siva Sankalp Patel, J William Murdock, Saloni Potdar, and Sachindra Joshi. 2022. Fast and light-weight answer text retrieval in dialogue systems. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 334–343.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407.

Yanru Xiao and Cong Wang. 2021. You see what i want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1934–1943.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pages 1154–1156.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.

Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. 2023. Poisoning retrieval corpora by injecting adversarial passages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13764–13775.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

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

A Additional Experiments

A.1 Comparing HotFlip with Random Perturbation

As illustrated in Figure 7, HotFlip first randomly selects a token position for replacement and identifies n potential alternatives for that position. It updates the adversarial passage with the most effective perturbation from this candidate set if it improves attack performance. If no perturbation enhances performance, HotFlip progresses to the next iteration by randomly selecting another token position and repeating the process. It is useful then to compare Hotflip vs. a natural black box `Random` approach to candidate generation. As depicted in Figure 7), the Random approach creates a candidate set by randomly sampling from the vocabulary.

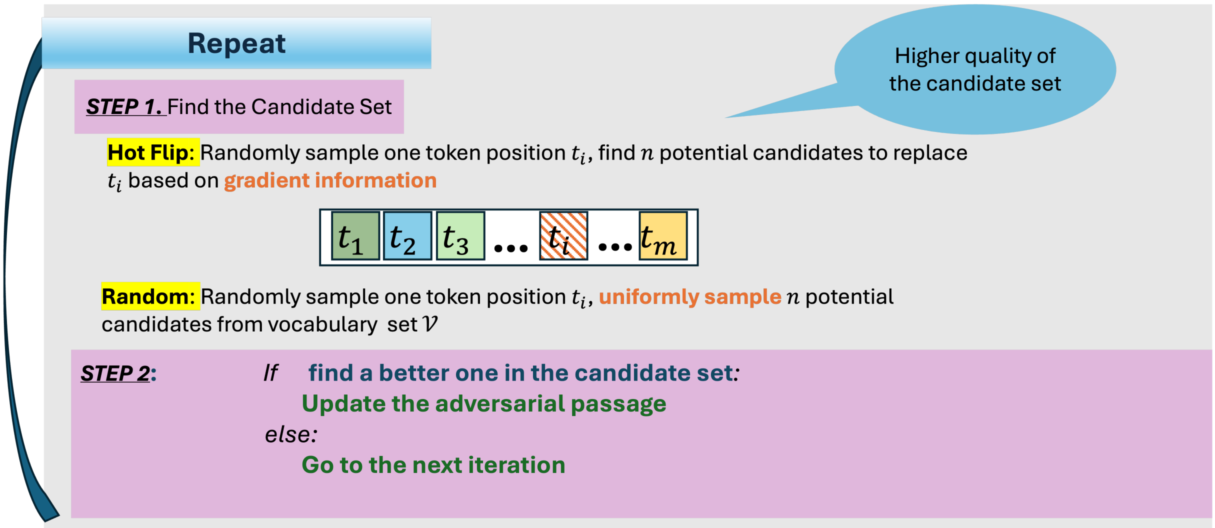

Repeat
Higher quality of the candidate set
STEP 1. Find the Candidate Set
Hot Flip: Randomly sample one token position t_i, find n potential candidates to replace t_i based on gradient information
t_1 t_2 t_3 ... t_i ... t_m
Random: Randomly sample one token position t_i, uniformly sample n potential candidates from vocabulary set $\mathcal{V}$
STEP 2:
If find a better one in the candidate set:
Update the adversarial passage
else:
Go to the next iteration

Figure 7: Comparing HotFlip attack and the random perturbation attack. The main difference is how they select the candidate set. HotFlip uses the gradient information to enhance the quality of the candidate set, whereas random perturbation selects candidates uniformly from $\mathcal{V}$ without leveraging gradient information.

HotFlip is more Efficient Than Random Perturbation In experiments using three dense retrievers on the widely used Natural Questions (NQ) dataset, we find that HotFlip outperforms the Random perturbation baseline, as shown in Figure 8. Specifically, HotFlip consistently achieves low retrieval accuracy on the validation set while maintaining candidate sets of the same size *or smaller*. (Red lines are mostly lower than the blue lines.) Equivalently, to achieve the same degree of attack effect, HotFlip is more efficient because it requires a smaller candidate set. For example, as depicted in Figure 9, Random perturbation needs to maintain a candidate set of size $n = 900$ to match the performance of HotFlip with only $n = 30$ candidates. This means that the size of Random perturbation candidate set is 30 times larger and consequently, Random perturbation is 30 times slower.

HotFlip and Random perturbation share the same intuition of maintaining a candidate set and then take a greedy token swap based on this set. Though many factors can influence the efficiency and the effectiveness the greedy search, one of the major factors is the quality of the candidate set. If the candidate set is of high quality, we can maintain a smaller size candidate set, reducing the number of searches at each greedy step. For example, HotFlip filters down the most likely potential tokens from $|\mathcal{V}|$ to n based on the gradient information, while random perturbation samples n tokens from $\mathcal{V}$. As a result, random perturbation has a lower quality candidate set. Therefore, it is crucial to improve the quality of the candidate set.

A.2 Retrieval Accuracy during Training

Retrieval Accuracy for 1 Adversarial Passage In Figure 10, we plot the Retrieval accuracy during training on both NQ and MS MARCO datasets for all 5 retrievers as a complement to the results in Table 1. During training, at each iteration, we evaluate the retrieval accuracy of the best adversarial passage a_i

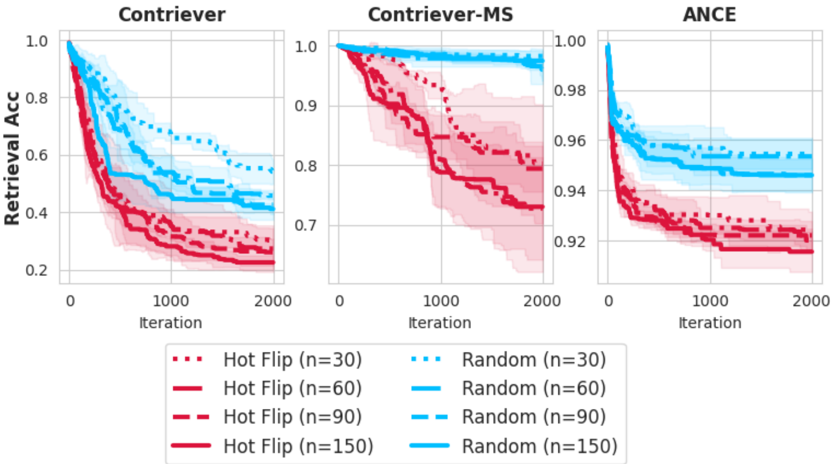


Figure 8: Retrieval accuracy of HotFlip and random perturbation on the NQ dataset with varying candidate set size $n = \{30, 60, 90, 150\}$. **Lower** retrieval accuracy suggests a more successful adversarial passage attack, as evaluated on the validation set during greedy search iterations.

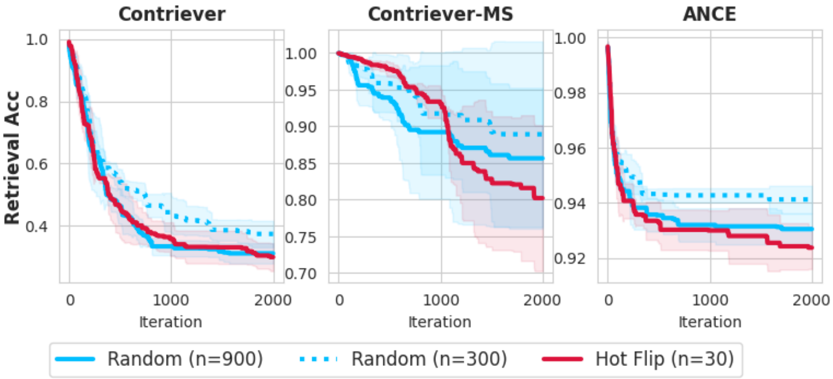


Figure 9: Retrieval accuracy of HotFlip and random perturbation. Hot Flip only needs to maintain a candidate set of size $n = 30$ to achieve the same performance of random perturbation with a candidate set size $n = 900$.

from the candidate set, and compare it with the current adversarial passage to decide whether to update it. Due to the greedy nature of these attacks, Retrieval accuracy is guaranteed to descend. Figure 10 shows that, for both NQ and MS MARCO dataset, the harder-to-attack models such as ANCE, DPR-mul and DPR-nq converge with high retrieval accuracy. In contrast, for the easier-to-attack models such as Contriever, the attack methods have not yet converged, even though the retrieval accuracy is as low as around 0.2.

Effect of the Candidate Set Size to Retrieval Accuracy In Figure 11, we illustrate the retrieval accuracy during training with varying candidate set sizes on NQ dataset and 3 retrievers. Larger candidate set sizes generally lead to lower retrieval accuracy.

Effect of Token Length on Retrieval Accuracy In Figure 12, we show the training retrieval accuracy with various token length settings. We find that, within a proper range, longer token lengths lead to more effective adversarial passages.

B Experimental Results with 10 Adversarial Passages

To generate multiple adversarial passages, we follow (Zhong et al., 2023) by clustering similar queries based on their embeddings using the k-means algorithm. We then generate one adversarial passage for each cluster.

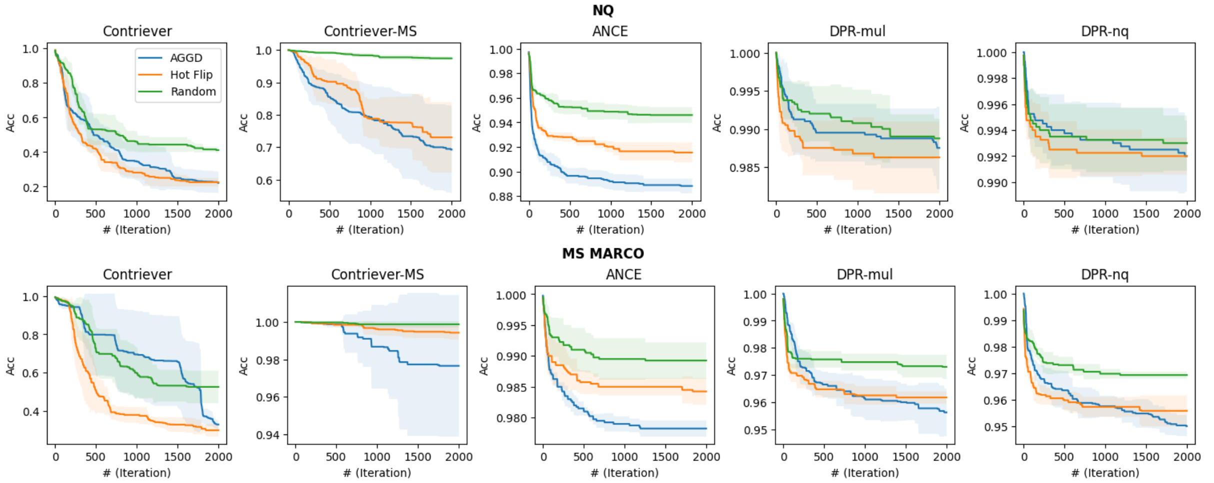


Figure 10: Retrieval accuracy (the portion of the queries that have higher similarity to the adversarial passage than the gold passage) on validation data during training. Lower retrieval accuracy indicates higher chance of adversarial passage being retrieved.

Dataset	Methods	Retriever				
		Contriever	Contriever-MS	ANCE	DPR-mul	DPR-nq
NQ	AGGD	91.57(3.5)	**69.42(1.38)**	**82.08(2.01)**	**8.23(1.48)**	1.9 (0.07)
	Hot Flip	**91.6(0.87)**	57.37(5.95)	63.46(1.55)	5.08(0.1)	**2.19(0.25)**
	Random	80.49(0.83)	33.73(2.27)	27.3(2.74)	3.94(1.22)	1.65 (0.06)
MS MARCO	AGGD	83.72(4.65)	**13.95(13.95)**	**93.02(0.0)**	**11.63(2.33)**	5.81 (1.16)
	Hot Flip	**88.37(2.33)**	13.95(11.63)	67.44(2.33)	10.46(1.16)	**6.98(2.33)**
	Random	65.12(9.3)	1.16(1.16)	52.33(15.12)	8.14(1.16)	3.49 (1.16)

Table 4: In-domain attack success rate (ASR) of AGGD on NQ and MS MARCO dataset with 5 retrievers by injecting 10 adversarial passages). The best-performing attacking method is highlighted with in bold (higher ASR indicates better attack performance).

B.1 Candidate Set Quality

In Figure 13, we present additional experiments with two more retriever models: Contriever-MS and ANCE. We observed a similar trend, over the 400 random samples, with more than 92% of the best candidate belonging to AGGD candidate set. This further supports the conclusions in Section 5.3.

In-Domain Attack with 10 Adversarial Passages Table 4 shows the in-domain attack results of inserting 10 adversarial passages into the NQ and MS MARCO datasets. Similar to injecting only 1 adversarial passage, we found that (1) The pretrained Contriever model is easy to attack: even with just 10 adversarial passages, all three baselines (AGGD, HotFlip and Random perturbation) successfully attack more than 80% of the queries in both NQ and MS MARCO datasets, tricking the Contriever into returning the adversarial passage among the top-20 retrieved results. For NQ dataset, even random perturbation achieves a high ASR of 80%, while the other two methods using gradient information achieves higher ASR of $> 91\%$. (2) AGGD still outperforms HotFlip on ANCE, with improvements of 18.62% and 25.58% on NQ and MS MARCO datasets, respectively.

Out-of-Domain Attack with 10 Adversarial Passages In Table 5, we perform the out-of-domain attack by inserting more adversarial passages. We find that, inserting more adversarial passages significantly improves the attack transferability of HotFlip, enabling it to outperform AGGD in models such as Contriever-MS, DPR-mul and DPR-nq. However, AGGD still performs much better than HotFlip when using ANCE as the retriever.

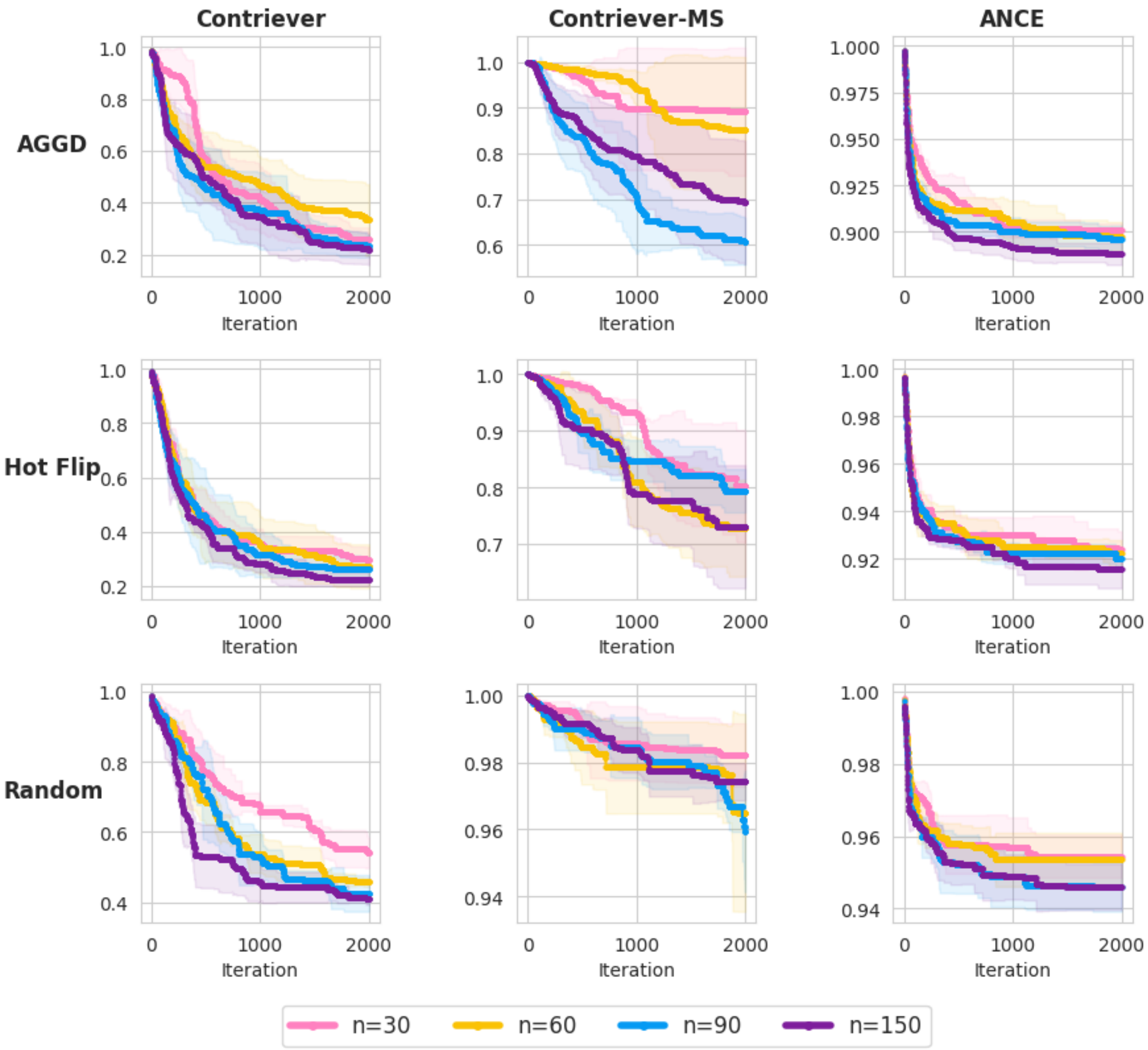


Figure 11: The retrieval accuracy during training with various candidate set sizes. ($n = 30, 60, 90, 150$).

B.2 In-domain attack success rate with different k.

B.3 Transferability over different retrievers.

C Experimental Detail

C.1 Dataset

- **MS MARCO** (Nguyen et al., 2016) contains a large amount of queries with annotated relevant passages from Web documents.
- **Natural Questions (NQ)** (Kwiatkowski et al., 2019) contains Google search queries with annotations from the top-ranked Wikipedia pages.

These two datasets have been widely used for evaluating dense retrieval models.

The statistics of all the datasets used in evaluation are summarized in Table 8.

C.2 Retrievers

We experimented with the following retrievers:

- **Dense Passage Retriever(DPR)** (Karpukhin et al., 2020) is a two-tower bi-encoder trained with a single BM25 hard negative and in-batch negatives. It has been used as the retrieval component of

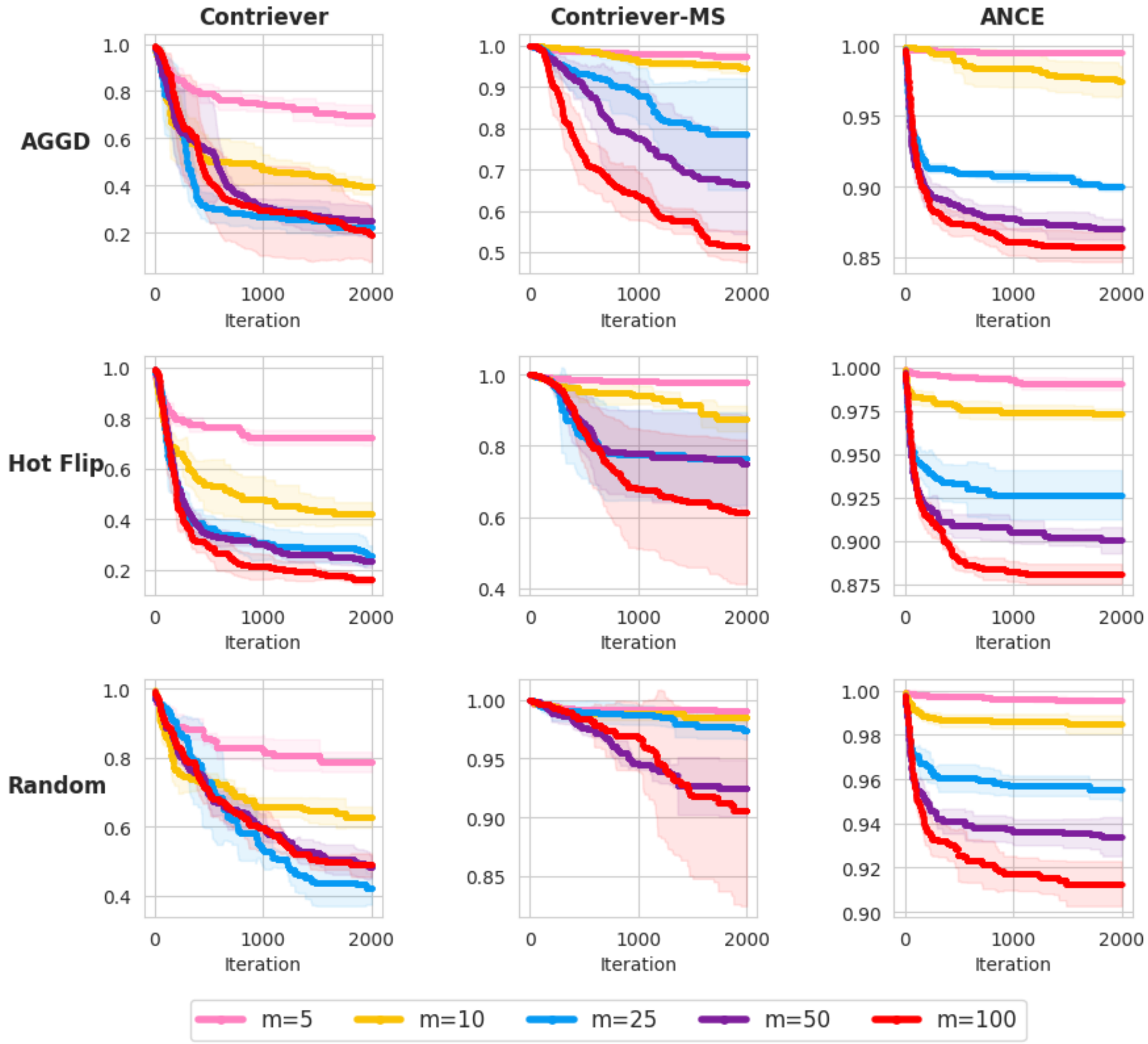


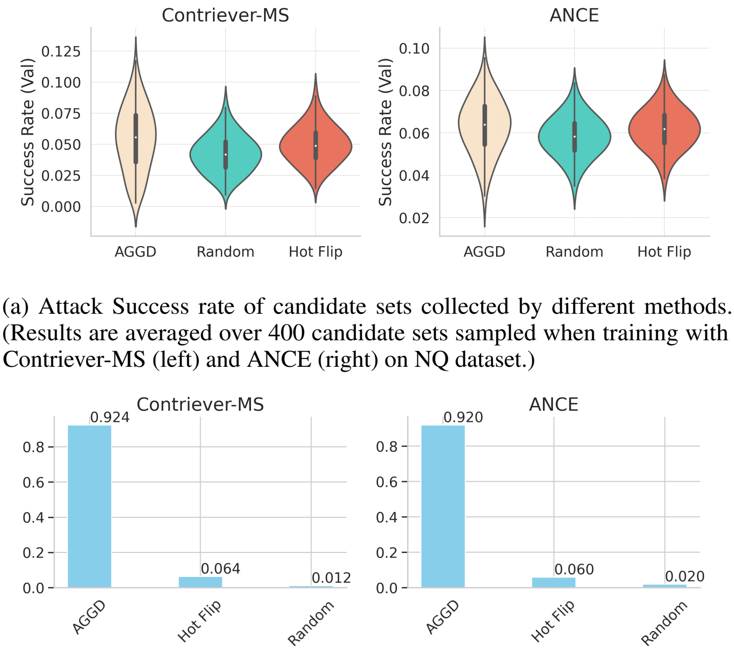
Figure 12: The retrieval accuracy during training with various token lengths ($m \in \{5, 10, 25, 50, 100\}$) and a fixed candidate set size $n = 100$.

many Retrieval-Augmented Generation (RAG) models (Lewis et al., 2020). In our paper, we use both the open-sourced Multi model (DPR-mul), which is a bert-base-uncased model trained on four QA datasets (NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), WebQuestions (Berant et al., 2013) and CuratedTREC (Baudiš and Šedivỳ, 2015)) and the single NQ model (DPR-nq).

- **ANCE (Xiong et al., 2020)** is a bi-encoder that generates hard negatives using an approximate Nearest Neighbor (ANN) index of the corpus. The index is continuously updated in parallel to identify challenging negative examples for the model during fine-tuning.
- **Contriever (Gautier et al., 2022)** is an unsupervised dense retriever using contrastive learning. It leverages the BERT architecture to encode both queries and documents. Contriever-MS (Contriever fine-tuned on MS MARCO) is a version of the Contriever model that has been fine-tuned using the MS MARCO dataset, which provides large-scale, supervised training data.

D Knowledge Poisoning Attacks to RAG

Retrieval-Augmented Generation (Karpukhin et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022; Thoppilan et al., 2022) augments LLMs with external knowledge retrieved from a knowledge base to improve their ability to generate accurate and up-to-date content. There are three components in RAG:



(a) Attack Success rate of candidate sets collected by different methods. (Results are averaged over 400 candidate sets sampled when training with Contriever-MS (left) and ANCE (right) on NQ dataset.)

(b) The portion of the times the best candidate occurs in the candidate sets collected by AGGD, HotFlip and Random, respectively, when training on NQ dataset with Contriever-MS (left) and ANCE (right).

Figure 13: Additional experiments on Contriever-MS and ANCE with NQ dataset illustrate that candidate sets collected by AGGD (a) has higher overall quality and (b) are more likely to contain the best candidate.

Target Domain	**Source Domain**	**Methods**	**Retriever**				
			Contriever	Contriever-MS	ANCE	DPR-mul	DPR-nq
FiQA-2018	NQ	AGGD	92.67(0.23)	30.86(7.87)	19.52(1.47)	6.02(0.15)	4.71(1.47)
		Hot Flip	84.1(0.77)	24.23(2.31)	13.35(2.7)	3.94(2.55)	3.24(1.23)
		Random	83.1(0.39)	17.82(5.32)	8.26(0.54)	2.7(0.23)	2.31(0.46)
	MS MARCO	AGGD	**97.69(0.31)**	50.54(25.23)	**37.19(0.31)**	**27.31(7.87)**	20.6(7.64)
		Hot Flip	95.22(1.08)	**56.48(3.4)**	28.16(2.08)	19.37(6.4)	**28.47(1.31)**
		Random	86.73(0.15)	6.94(0.15)	13.27(1.39)	15.66(7.48)	7.25(4.63)
NFCorpus	NQ	AGGD	78.02(1.24)	35.76(3.25)	64.55(3.87)	18.11(4.8)	18.89(3.1)
		Hot Flip	85.45(0.31)	37.0(3.87)	48.76(1.08)	20.43(0.31)	20.59(1.39)
		Random	63.78(1.86)	24.46(3.41)	35.6(0.62)	16.72(1.55)	15.79(1.55)
	MS MARCO	AGGD	**95.36(0.62)**	57.9(26.01)	**88.39(0.15)**	57.89(8.05)	52.01(3.72)
		Hot Flip	87.62(5.88)	**70.28(12.69)**	80.34(1.39)	**59.75(10.84)**	**66.72(4.8)**
		Random	85.45(3.1)	51.39(5.57)	65.17(2.63)	34.98(1.24)	52.48(8.2)
Quora	NQ	AGGD	96.18(0.03)	63.14(3.93)	91.52(0.89)	20.96(3.63)	25.8(5.81)
		Hot Flip	92.22(1.08)	57.88(10.45)	84.9(2.27)	24.82(0.15)	32.19(1.7)
		Random	87.85(2.18)	46.28(7.64)	81.72(0.25)	15.84(0.97)	22.28(3.28)
	MS MARCO	AGGD	97.87(0.93)	61.72(7.29)	**94.52(0.17)**	**45.92(1.0)**	51.5(3.35)
		Hot Flip	**98.13(0.79)**	**69.88(3.63)**	91.5(0.16)	43.86(2.64)	**51.6(3.73)**
		Random	83.97(3.6)	35.24(4.43)	85.48(2.07)	32.88(0.31)	38.36(6.64)
SCIDOCS	NQ	AGGD	53.75(7.05)	46.4(17.7)	33.75(1.25)	1.7(0.3)	0.45(0.05)
		Hot Flip	49.75(2.45)	40.7(5.4)	26.45(6.05)	3.35(0.15)	0.95(0.15)
		Random	37.8(4.3)	18.25(1.75)	13.3(0.9)	2.3(0.4)	0.6(0.2)
	MS MARCO	AGGD	**65.5(2.6)**	41.65(9.85)	**46.2(1.9)**	**24.9(2.2)**	6.7(0.2)
		Hot Flip	51.45(4.75)	**68.1(8.3)**	36.95(0.35)	22.4(2.9)	**8.3(1.1)**
		Random	39.2(4.5)	13.9(4.3)	20.9(0.6)	15.55(3.35)	2.45(1.05)
SciFact	NQ	AGGD	60.33(6.0)	12.5(7.17)	39.33(9.33)	0.5(0.5)	0.17(0.17)
		Hot Flip	58.83(12.83)	11.83(2.17)	16.5(3.5)	1.67(0.67)	0.5(0.17)
		Random	34.33(4.33)	3.17(2.17)	8.17(1.5)	0.33(0.0)	0.5(0.17)
	MS MARCO	AGGD	**91.5(4.5)**	**41.5(8.5)**	**58.17(0.83)**	12.0(5.0)	12.0(7.67)
		Hot Flip	85.83(5.17)	29.17(9.5)	42.33(4.0)	**15.17(0.5)**	**14.67(7.33)**
		Random	75.5(9.5)	8.0(1.33)	21.5(6.5)	5.67(1.67)	10.33(1.0)

Table 5: Out-of-domain top-20 attack success rate with 10 adversarial passages. Due to the computational constraints, results are averaged over 2 random runs, with standard deviations shown in parentheses. The combinations of attack and source dataset that achieve the highest ASR for each target domain and retriever are highlighted in bold.

	Dataset	Method	Retriever				
			Contriever	Contriever-MS	ANCE	DPR-mul	DPR-nq
Top-1	NQ	AGGD	**85.06(5.06)**	**44.84(12.65)**	**16.36(1.52)**	**0.39(0.15)**	-
		Hot Flip	83.77(1.43)	39.74(7.81)	8.66(0.62)	0.21(0.08)	0.03 (0.02)
		Random	66.64(2.61)	8.82(0.87)	1.89(0.5)	0.07(0.04)	**0.04(0.03)**
	MS MARCO	AGGD	69.19(6.65)	11.05(12.67)	**24.42(7.45)**	1.16(1.16)	**2.91(1.93)**
		Hot Flip	**71.51(4.15)**	**12.79(8.47)**	10.47(2.01)	**2.33(1.64)**	0.58 (1.01)
		Random	52.33(6.26)	1.74(1.93)	5.23(4.47)	-	0.58 (1.01)
Top-5	NQ	AGGD	**89.84(3.53)**	**55.63(10.96)**	**49.56(3.78)**	**1.9(0.55)**	0.27 (0.16)
		Hot Flip	88.69(0.72)	50.64(5.96)	31.36(1.16)	1.29(0.26)	**0.28(0.1)**
		Random	75.28(1.8)	20.07(1.44)	10.12(1.82)	0.71(0.28)	0.24 (0.11)
	MS MARCO	AGGD	**81.98(5.03)**	**20.35(16.72)**	**66.28(5.33)**	**8.72(3.02)**	**3.49(1.16)**
		Hot Flip	80.23(6.04)	19.77(10.2)	36.05(9.08)	3.49(1.16)	2.33 (0.0)
		Random	62.21(7.6)	8.14(7.07)	20.93(4.03)	3.49(2.01)	0.58 (1.01)
Top-10	NQ	AGGD	**91.28(3.03)**	**59.39(9.83)**	**66.43(4.42)**	**3.62(1.02)**	**0.82(0.38)**
		Hot Flip	89.85(0.82)	54.52(5.41)	47.55(1.73)	2.69(0.24)	0.72 (0.2)
		Random	77.8(1.45)	25.91(2.07)	18.8(3.26)	1.61(0.45)	0.66 (0.1)
	MS MARCO	AGGD	**84.88(3.86)**	**22.67(17.35)**	**85.47(2.53)**	**9.3(3.68)**	**4.07(1.01)**
		Hot Flip	81.4(5.45)	20.35(10.58)	57.56(9.36)	4.65(1.64)	2.91 (1.01)
		Random	65.12(7.89)	11.63(10.78)	35.47(8.28)	4.65(2.85)	1.16 (1.16)
Top-20	NQ	AGGD	**92.5(2.68)**	**63.45(8.68)**	**80.92(4.82)**	**6.88(1.72)**	**2.19(0.98)**
		Hot Flip	91.08(0.9)	58.43(4.53)	65.68(2.5)	5.4(0.36)	2.03 (0.25)
		Random	80.24(0.92)	32.5(2.61)	31.0(4.3)	3.7(0.93)	1.66 (0.14)
	MS MARCO	AGGD	**85.47(3.81)**	**24.42(17.44)**	**93.6(1.01)**	**12.79(3.86)**	5.23 (1.01)
		Hot Flip	83.72(5.93)	22.67(12.01)	76.16(9.21)	9.88(1.93)	**5.81(2.01)**
		Random	66.86(7.24)	13.95(13.46)	49.42(11.2)	6.4(2.53)	2.91 (1.93)
Top-50	NQ	AGGD	**93.87(2.26)**	**68.69(6.66)**	**93.7(3.21)**	**15.59(3.41)**	**7.31(2.02)**
		Hot Flip	92.56(0.82)	63.28(3.64)	85.93(2.4)	13.37(1.1)	6.84 (1.1)
		Random	83.01(0.46)	42.35(3.55)	53.01(5.31)	9.6(1.76)	5.46 (0.51)
	MS MARCO	AGGD	**87.21(5.33)**	31.98(20.3)	**99.42(1.01)**	**22.67(5.55)**	**13.37(3.02)**
		Hot Flip	84.88(4.19)	**32.56(18.16)**	95.35(1.64)	19.77(3.49)	10.46 (5.07)
		Random	69.19(4.47)	19.19(17.35)	75.58(5.33)	12.79(3.49)	4.65 (1.64)
Top-100	NQ	AGGD	**94.77(2.08)**	**72.59(5.28)**	**98.2(1.31)**	**27.94(5.7)**	**18.0(3.6)**
		Hot Flip	93.58(0.66)	67.27(2.97)	94.94(1.63)	24.48(2.26)	16.8 (1.79)
		Random	85.11(0.41)	50.86(4.34)	71.57(5.81)	18.6(3.15)	12.81 (1.34)
	MS MARCO	AGGD	**89.54(4.79)**	**40.12(24.36)**	**100.0(0.0)**	**41.28(5.03)**	**20.93(4.35)**
		Hot Flip	86.05(4.93)	**40.12(18.12)**	**100.0(0.0)**	31.98(5.04)	18.02 (5.78)
		Random	70.93(2.6)	23.26(18.09)	90.12(6.01)	18.02(2.53)	9.88 (4.47)

Table 6: In-domain attack success rate (ASR) of AGGD on NQ and MS MARCO datasets with 5 retrievers by injecting 1 adversarial passage with varying $k = \{1, 5, 10, 20, 50, 100\}$. Results are from 4 random runs with standard deviation in parenthesis.For the ease of presentation, we omit the results through '-' if top-k ASR is smaller than 0.1%.

Source Retriever	Method	Target Retriever				
		Contriever	Contriever-MS	ANCE	DPR-mul	DPR-nq
contriever	AGGD	**97.14**	**26.51**	-	-	1.2
	Hot Flip	96.05	20.9	-	**0.18**	0.48
	Random	90.5	11.8	-	0.14	**1.39**
contriever-msmarco	AGGD	**27.76**	**83.71**	-	-	0.28
	Hot Flip	25.23	79.05	-	1.02	2.28
	Random	26.24	78.38	-	**2.05**	**7.16**
ance	AGGD	26.43	4.88	**100.0**	-	-
	Hot Flip	**28.48**	4.92	99.99	-	-
	Random	9.86	**6.39**	99.3	-	-
dpr-multi	AGGD	15.78	3.67	-	**87.84**	22.47
	Hot Flip	**23.43**	**7.7**	-	84.18	**38.41**
	Random	18.43	4.15	-	79.35	10.41
dpr-single	AGGD	10.85	**9.99**	-	**29.16**	**89.46**
	Hot Flip	11.87	9.45	-	27.91	87.72
	Random	**22.02**	7.37	-	20.5	82.55

Table 7: Attack transferability across models on NQ dataset.

knowledge base, the retriever and the LLM. The knowledge base contains a large corpus collected from various domains such as Wikipedia (Thakur et al., 2021), Fiance (Loukas et al., 2023) and Biomedical articles (Roberts et al., 2020). Given a user question, the retriever uses a text encoder to compute the embedding vector. A set of k retrieved texts from the knowledge base with the highest similarity to the question are then retrieved, which can be used by the LLM to generate content.

RAG enables LLMs to incorporate more current knowledge by regularly updating the knowledge base. However, this also introduces potential security concerns: maliciously crafted content could be injected into the database during updates, which might then be retrieved by the LLM to generate false, biased or harmful output.

	Domain	Dataset	Train	Dev	Test	
			# Pair	# Query	# Query	# Corpus
In-domain	Web	MS MARCO (Nguyen et al., 2016)	532,761	-	6,980	8,841,823
	Wikipedia	NQ (Kwiatkowski et al., 2019)	132,803	-	3,452	2,681,468
Out-of-domain	Bio Medical	NFCorpus (Boteva et al., 2016)	110,575	324	323	3,633
	Quora	Quora	-	5,000	10,000	522,931
	Scientific	SCIDOCS (Cohan et al., 2020)	-	-	1,000	25,657
		SciFact (Wadden et al., 2020)	920	-	300	5,183
	Finance	FiQA-2018 (Maia et al., 2018)	14,166	500	648	57,638

Table 8: Dataset Statistics. More statistics can be found in (Zhao et al., 2024; Thakur et al., 2021). In our experiments, we use MS MARCO and NQ datasets to train the adversarial passages and evaluate the attack on the unseen test queries from these two datasets for in-domain attack evaluation. The remaining 5 datasets (NFCorpus, Quora, SCIDOCS, SciFact, FiQA-2018) are used for out-of-domain evaluation when injecting the adversarial passages generated from the MS MARCO and NQ dataset into the corpora of these out of domain datasets.

	LLaMa-2-7B	LLaMa-2-13B	Vicuna-7B	Vicuna-13B	Vicuna-33B	GPT-3.5	GPT-4
MS MARCO							
AGGD	**0.92(0.00)**	**0.92(0.00)**	**0.92(0.00)**	**0.92(0.00)**	**0.92(0.00)**	**0.92(0.00)**	**0.92(0.00)**
HotFlip	0.90(0.01)	0.90(0.01)	0.90(0.01)	0.90(0.01)	0.90(0.01)	0.90(0.01)	0.90(0.01)
NQ							
AGGD	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
hotflip	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)

Table 9: PoisonedRAG performance on multiple dataset and LLMs using Hot Flip and AGGD evaluated using F1 score.

Threat Model We consider a scenario where the attacker does not have direct access to the database, but can access the weights of the retrieval model. This scenario is realistic, as the database is likely hosted on a secure system, while the retriever used may be an open-access LLM. We assume the attacker can inject a few carefully crafted corpus into the knowledge base.

Metric We use the same metrics, Attack Success Rate (ASR) and F1, as used in (Zou et al., 2024). However, since ASR is task-specific, it might not be exactly the same as defined for corpus poisoning attacks. In PoisonedRAG, the definitions are given as follows.

- **ASR** The fraction of target questions whose answers are the attacker-chosen target answers.

Additionally, we report average F1-Score over different target questions in Table 9, which measures the trade-off between Precision and Recall. Specifically,

$$\text{F1 score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

where Precision is defined as the fraction of poisoned texts among the top-k retrieved ones for the target question and Recall is defined as the fraction of poisoned texts among the N poisoned ones that are retrieved for the target question. A higher F1-Score means more poisoned texts are retrieved.

Hyperparameters We inject 1 adversarial poisoned text for each target question and test on all 100 test data. The results are averaged over 4 random runs.

E Examples of the output passage from AGGD

Examples from attacking Contriever	Examples from attacking contriever-MS
["##nius", "half", "melting", "definite", "[MASK]", "favor", "gma", "who", "should", "jew", "rain", "##pi", "##ntial", "upper", "kevin", "perrin", "##gren", "blew", "demonstrators", "remnants", "shelters", "that", "##ntial", "hue", "lest", "rainfall", "where", "rains", "rain", "heavy"]	["hostage", "tightly", "handful", "herbs", "where", "packets", "were", "##ssen", "dealers", "overnight", "symbol", "##atic", "##ised", "adventure", "bail", "##alis", "##kari", "[CLS]", "convicted", "ga", "##oit", "peace", "restore", "lifespan", "discrimination", "(", "maha", "##ter", "bank", "##ees"]

Table 10: Examples of adversarial passages