Tricking Retrievers with Influential Tokens: An Efficient Black-Box Corpus Poisoning Attack

Cheng Wang[†], **Yiwei Wang**[||], **Yujun Cai**[‡], , **Bryan Hooi**[†],
[†] National University of Singapore [||] University of California, Merced
[‡] University of Queensland
wcheng@comp.nus.edu.sg

Abstract

Retrieval-augmented generation (RAG) systems enhance large language models by incorporating external knowledge, addressing issues like outdated internal knowledge and hallucination. However, their reliance on external knowledge bases makes them vulnerable to corpus poisoning attacks, where adversarial passages can be injected to manipulate retrieval results. Existing methods for crafting such passages, such as random token replacement or training inversion models, are often slow and computationally expensive, requiring either access to retriever's gradients or large computational resources. To address these limitations, we propose Dynamic Importance-Guided Genetic Algorithm (DIGA), an efficient black-box method that leverages two key properties of retrievers: insensitivity to token order and bias towards influential tokens. By focusing on these characteristics, DIGA dynamically adjusts its genetic operations to generate effective adversarial passages with significantly reduced time and memory usage. Our experimental evaluation shows that DIGA achieves superior efficiency and scalability compared to existing methods, while maintaining comparable or better attack success rates across multiple datasets.

1 Introduction

Large language models (LLMs) (Yang et al., 2024; Achiam et al., 2023; Meta, 2024) have shown impressive performance across a wide range of natural language processing tasks, but they still suffer from significant limitations, such as hallucination (Huang et al., 2023; Chen and Shu, 2023) and outdated internal knowledge. To address these issues, retrieval-augmented generation (RAG) (Gao et al., 2024; Fan et al., 2024) systems have emerged as a promising solution by incorporating external, up-to-date knowledge into the generation process. By retrieving relevant information from large external corpora, RAG systems can enhance the accuracy and relevance of LLM-generated outputs, especially in open-domain question answering and knowledge-intensive tasks.

	HotFlip	Vec2Text	**DIGA (ours)**
Black-box	×	✓	✓
No Add. Training	✓	×	✓
Scalability	×	✓	✓

Table 1: **Comparison of different corpus poisoning methods.** Our proposed method is black-box based, requires no additional training, and maintains high scalability.

Despite the benefits, the reliance on external knowledge sources exposes RAG systems to potential security vulnerabilities (Zeng et al., 2024; Xue et al., 2024; Li et al., 2024b; Anderson et al., 2024). A particular focus has been on adversarial attacks targeting RAG systems. Initial studies (Song et al., 2020; Raval and Verma, 2020; Liu et al., 2022) explored attacks where adversarial passages are injected into the knowledge base to influence the language model's output for specific queries. Building upon this, Zhong et al. (2023) introduced a more potent form of attack termed corpus poisoning, where a single adversarial passage aims to affect retrieval results across multiple queries simultaneously. Currently, two primary approaches dominate corpus poisoning attack implementations. The first, based on HotFlip (Ebrahimi et al., 2018), initializes an adversarial passage with random tokens and iteratively optimizes it using gradients from the retriever. The second method, Vec2Text (Morris et al., 2023), originally developed to study text reconstruction from embeddings, has been repurposed for corpus poisoning by inverting the embeddings of query centroids.

However, these existing methods face significant limitations: **(1) White-box access:** HotFlip requires access to the retriever model's gradient, limiting its applicability in real-world scenarios where such access is often restricted. **(2) Lack of**

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4183–4194

generalizability: Vec2Text requires training a separate inversion model on a specific dataset. This limits its applicability to new domains or datasets without retraining, as the model may struggle to reconstruct embeddings for data significantly different from its training set. **(3) High computational demands:** Both methods require substantial computational resources, making them impractical for large-scale attacks. HotFlip's iterative optimization process is time-consuming and memory-intensive. Vec2Text, while faster in generation, requires additional memory for its inversion model and is computationally expensive during the training phase. These resource constraints severely limit the scalability of both approaches.

Given the limitations of existing methods, we posit a crucial question: Is it possible to perform corpus poisoning attacks under minimal assumptions about the target system while maintaining high effectiveness and efficiency? In response, we propose the Dynamic Importance-Guided Genetic Algorithm (DIGA). Our method is built upon the foundation of genetic algorithms, a class of optimization techniques inspired by the process of natural selection. DIGA incorporates two key properties of retrieval systems into its evolutionary process: their insensitivity to token order and bias towards influential tokens. By leveraging these properties, DIGA dynamically adjusts its genetic operations, such as population initialization and mutation, to craft effective adversarial passages. This approach allows us to generate potent adversarial content without requiring white-box access to the retriever or extensive computational resources (see Table 1).

Our empirical evaluation demonstrates that DIGA achieves comparable or superior results to existing methods across various datasets and retrievers. Notably, DIGA accomplishes this with significantly lower time and memory usage, making it more suitable for large-scale attacks. Furthermore, our method exhibits stronger transferability across different datasets compared with another black-box method. This enhanced transferability is significant as it demonstrates DIGA's ability to generate more generalizable adversarial passages, potentially making it more effective in real-world scenarios where attack and target datasets may differ.

Our study underscores the vulnerabilities present in RAG systems and the importance of continued research into their security. Future efforts should prioritize developing robust defenses and evaluating real-world risks. We hope this work inspires further advancements in safeguarding RAG systems.

2 Related Work

Attacks on RAG Systems and Corpus Poisoning. Recent research has explored vulnerabilities in retrieval augmented generation (RAG) systems and dense retrievers. Membership inference attacks (Anderson et al., 2024; Li et al., 2024b) aim to determine the presence of specific data in knowledge bases. Other studies focus on manipulating system outputs, such as PoisonedRAG (Zou et al., 2024) crafting misinformation-laden passages, and indirect prompt injection attacks (Greshake et al., 2023) inserting malicious instructions. In corpus poisoning, Zhong et al. (2023) proposed a gradient-based approach inspired by HotFlip (Ebrahimi et al., 2018), while Vec2Text (Morris et al., 2023) introduced text embedding inversion. Unlike PoisonedRAG, which targets individual queries, our work aims to craft adversarial passages affecting multiple queries simultaneously, presenting unique challenges in corpus poisoning attacks on RAG systems.

Genetic Algorithms in NLP. Genetic algorithms (GA) have been increasingly applied to various Natural Language Processing (NLP) tasks. Recent studies have demonstrated GA's efficacy in neural architecture search (Chen et al., 2024), adversarial prompt generation for LLM security (Liu et al., 2023; Li et al., 2024a), and enhancing GA operations using LLMs (Guo et al., 2023; Lehman et al., 2023; Meyerson et al., 2023). These applications showcase the versatility of GA in addressing complex NLP challenges. GARAG (Cho et al., 2024) shares similarities with our work in applying GA to RAG systems. However, our approach differs significantly by targeting multiple queries simultaneously, rather than focusing on query-specific adversarial passage generation. Furthermore, our method employs a dynamic mechanism specifically tailored to exploit the properties of dense retrievers, offering a more nuanced approach to the challenge of corpus poisoning in RAG systems.

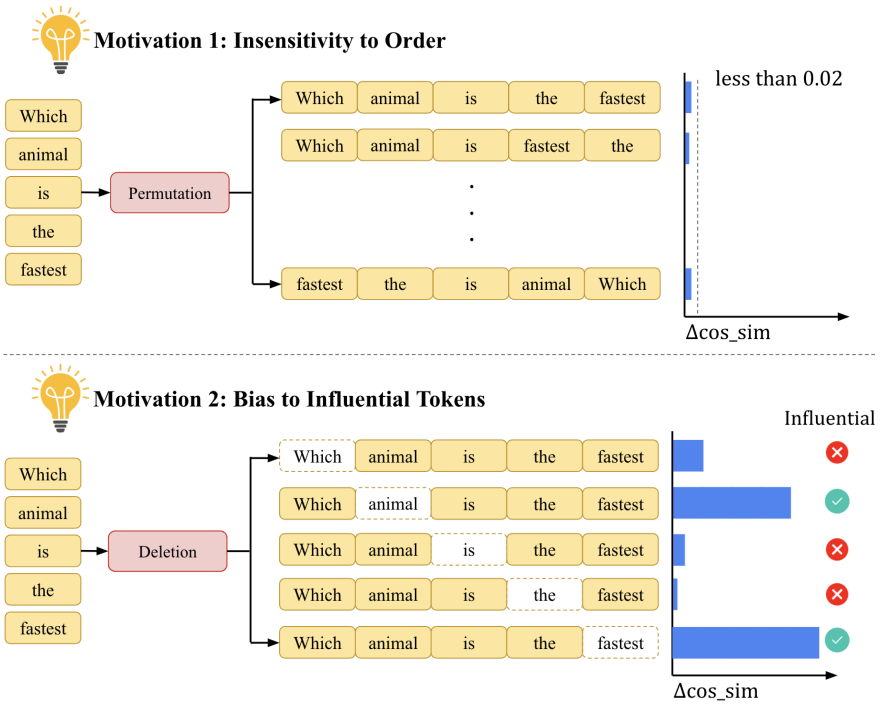

Figure 1: **Illustration of our motivations.** Top: Demonstrating insensitivity to token order, where cosine similarity remains nearly unchanged after permuting the tokens. Bottom: Highlighting bias towards influential tokens, shown by the varying effects on cosine similarity when different tokens are deleted. Some tokens are more influential since deleting them results in a larger change in similarity.

3 Method

3.1 Problem Formulation

Let $\mathcal{C}$ be the corpus or knowledge base of the RAG system, $Q = \{q_1, q_2, ..., q_n\}$ be the set of queries, ϕ_Q be the query embedding model, and ϕ_C be the context embedding model. Our goal is to generate an adversarial passage a whose embedding maximizes the similarity to all the query embeddings, increasing the likelihood of retrieval for any given query. Formally, we aim to solve:

$$a = \arg\max_{a'} \frac{1}{|Q|} \sum_{q_i \in Q} \phi_Q(q_i)^T \phi_C(a').$$

This can be rewritten as:

$$\begin{aligned} a &= \arg\max_{a'} \phi_C(a')^T \frac{1}{|Q|} \sum_{q_i \in Q} \phi_Q(q_i) \\ &= \arg\max_{a'} \phi_C(a')^T \bar{\phi}_Q, \end{aligned}$$

where $\bar{\phi}_Q = \frac{1}{|Q|} \sum_{q_i \in Q} \phi_Q(q_i)$ represents the centroid of all query embeddings. Therefore, the adversarial passage is the solution to this optimization problem, whose embedding should maximize the similarity to the centroid of query embeddings. To generate multiple adversarial passages, we follow Zhong et al. (2023) by applying k-means clustering to the query embeddings and solving the optimization problem for each cluster centroid, resulting in k adversarial passages.

3.2 Motivation

The design of our method stems from two key observations about retrievers, as demonstrated in Figure 1: **(1) Insensitivity to token order:** Retrievers show remarkable insensitivity to the order of words. By comparing the original sentence with various permutations of its tokens, including extreme cases like complete reversal, we observe only negligible differences in similarity. **(2) Bias towards influential tokens:** Different words contribute unequally to the overall similarity score. When deleting certain words, the similarity drops significantly more than for others. For instance, removing "animal" or "fastest" from the sentence "Which animal is the fastest?" causes a substantial decrease in similarity, while deleting "is" or "the" has only a minor impact.

Our method leverages these properties to efficiently craft adversarial passages, resulting in a more effective and computationally efficient approach.

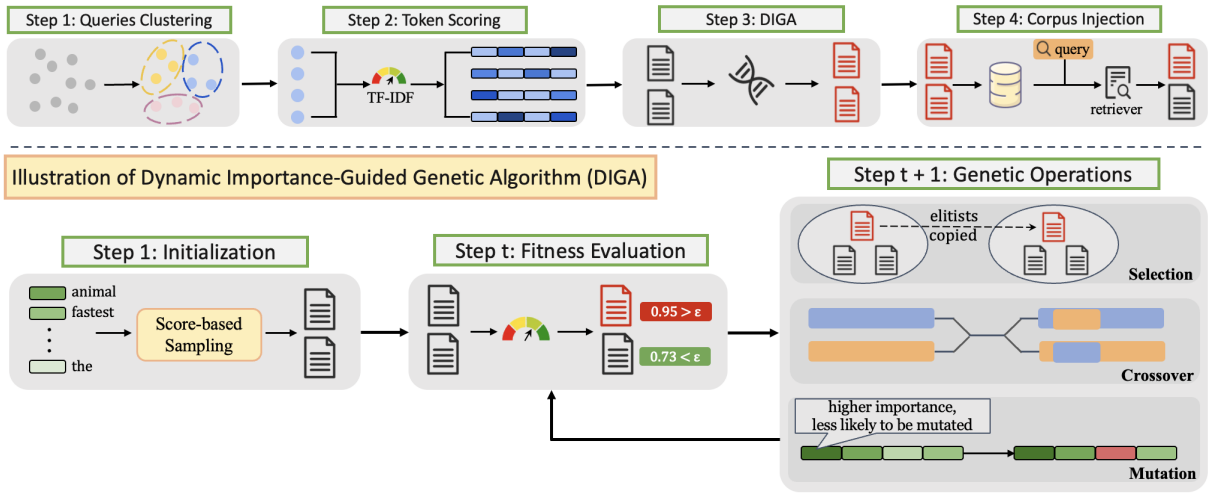

Figure 2: **An overview of our proposed method**.

4 Dynamic Importance-Guided Genetic Algorithm (DIGA)

We propose Dynamic Importance-Guided Genetic Algorithm (DIGA), which is designed to efficiently craft adversarial passages by exploiting two key properties of retrievers: their insensitivity to token order and bias towards influential tokens. DIGA employs a genetic algorithm framework, dynamically adjusting its genetic policies based on token importance. Figure 2 illustrates the pipeline of our method, which consists of several key components. We will now give a detailed explanation of these components in our proposed method. We provide an algorithmic description of DIGA in Algorithm 1.

4.1 Token Importance Calculation

Our second observation—that different tokens contribute unequally to the similarity score—motivates us to quantify token importance. This quantification is crucial for guiding the genetic algorithm toward more effective adversarial passages, leading to faster convergence.

To efficiently calculate token importance across a large corpus, we employ Term Frequency-Inverse Document Frequency (TF-IDF) (Salton et al., 1975). Our experiments demonstrate that TF-IDF values align well with the Leave-one-out values (Cook, 1977), as illustrated in Figure 1. Importantly, TF-IDF calculation is considerably more efficient than LOO, making it suitable for large corpus.

4.2 Population Initialization

Initialization is a critical component in genetic algorithms, influencing both the quality and diversity of the initial population. Our method employs a score-based initialization strategy that aligns with our observation of retrievers' bias towards influential tokens. We initialize the population by sampling tokens based on importance scores derived from TF-IDF values, ensuring that more influential tokens have a higher probability of inclusion in the initial adversarial passages. This approach effectively focuses the search on promising regions of the solution space from the start. The probabilistic nature of token selection balances exploitation of influential tokens with exploration of less common ones, creating a population biased towards effective adversarial passages while maintaining diversity for the genetic algorithm.

4.3 Fitness Evaluation

As shown in Section 3.1, there is a closed-form solution to our optimization problem, which involves maximizing the cosine similarity between the embedding of an adversarial passage and the centroid of query embeddings. Consequently, we directly use this cosine similarity as our fitness evaluation function. Formally, given a set of queries Q_c from the same cluster assigned by k-means, the fitness of an adversarial passage a is defined as: $\cos(\phi_C(a), \bar{\phi}_Q)$, where $\bar{\phi}_Q = \frac{1}{|Q_c|} \sum_{q_i \in Q_c} \phi_Q(q_i)$.

4.4 Genetic Policies

Our genetic algorithm utilizes three core operations: selection, crossover, and mutation. These opera-

tions are tailored to exploit the properties of dense retrievers we identified earlier, ensuring both the effectiveness of the generated adversarial passages and the efficiency of the overall process.

Selection. Selection is performed using a combination of elitism and roulette wheel selection. Given a population of N adversarial passages and an elitism rate α, we first preserve the top $N * \alpha$ passages with the highest fitness scores. This ensures that the best solutions are carried forward to the next generation. For the remaining $N - N * \alpha$ slots, we use a softmax-based selection probability. This method balances exploration and exploitation by giving better-performing adversarial passages a higher chance of selection while still allowing for diversity in the population.

Crossover. Our crossover operation exploits the insensitivity to token order in retrievers. This motivation highlights the fact that we should focus on which tokens to choose, rather than how to arrange them. This insight leads us to implement a single-point crossover where two parent sequences are split at a random point, and their tails are swapped to create two offspring. This approach preserves influential tokens from both parents while allowing for new combinations.

Mutation. Mutation maintains population diversity in genetic algorithms. Our DIGA mutation strategy dynamically adjusts based on token importance, leveraging the retriever's bias towards influential tokens. For each token in an adversarial passage, we calculate a replacement probability:

$$P_{replace} = \min\left(\frac{(C - s_i) \cdot \tau}{Z} + \gamma, 1\right),$$

where C is the maximum token score, s_i is the current token's score, τ is a temperature parameter, and γ is a baseline mutation rate. Z is a normalization factor defined as: $Z = \frac{1}{n}\sum_i (C - s_i)$.

Temperature τ modulates mutation sensitivity to token importance, while γ ensures a baseline mutation rate. This importance-guided strategy balances token preservation with exploration, adapting to the evolving fitness landscape.

5 Experiments

5.1 Experimental Setup

Datasets. In our study, we investigate four diverse datasets to simulate a range of retrieval augmented generation scenarios. These datasets, characterized by varying numbers of queries and corpus sizes, are: Natural Questions (NQ) (Kwiatkowski et al., 2019), NFCorpus (Boteva et al., 2016), FiQA (Maia et al., 2018), and SciDocs (Cohan et al., 2020). Each dataset presents unique characteristics that allow us to examine the efficacy of our proposed methods across different domains and retrieval contexts. A comprehensive overview of the statistical properties of these datasets is provided in Appendix A.

Models. In our experiments, we focus on three dense retrievers: GTR-base (Ni et al., 2022), ANCE (Xiong et al., 2020) and DPR-nq (Karpukhin et al., 2020). The last two models are mainly included to test the transferability of different attack methods.

Apart from the retriever in the RAG system, the attack method Vec2Text (Morris et al., 2023) also requires an inversion model to invert the embedding to text. We use the model provided by Morris et al. (2023), which is trained on 5 million passages from the NQ dataset where each passage consists of 32 tokens.

Evaluation Metric. To evaluate the effectiveness of attacks, we define Attack Success Rate@k (ASR@k), which is the percentage of queries for which at least one adversarial passage is retrieved in the top-k results.

To evaluate cross-dataset transferability, we define the Transferability Score as:

$$1 - \frac{1}{|D \setminus \{i\}|} \sum_{j \in D \setminus \{i\}} \frac{ASR_{i,i} - ASR_{i,j}}{ASR_{i,i}},$$

where $ASR_{i,j}$ is the ASR when evaluating on dataset i using adversarial passages generated from dataset j, $ASR_{i,i}$ is the in-domain ASR where the evaluation and attack datasets are the same, and $D \setminus \{i\}$ is the set of all datasets excluding i. This metric quantifies how well attacks generalize across different datasets.

We also evaluate computational efficiency by measuring method-specific execution time, excluding shared preprocessing tasks like corpus encoding and k-means clustering. We also track peak GPU memory usage to assess each method's maximum memory demands during execution.

Implementation Details. Our experimental protocol follows the methodology established

Dataset	Method	Number of Adversarial Passages						Time	GPU Usage
		1		10		50			
		ASR@5	ASR@20	ASR@5	ASR@20	ASR@5	ASR@20		
NQ	*White-box Methods*								
	HotFlip	0.1	1.3	0.4	3.2	2.0	9.6	6.5x	10.2x
	Black-box Methods								
	Vec2Text	0.0	0.0	0.1	1.8	1.4	4.3	-	1.43x
	DIGA (ours)	0.0	0.0	0.1	1.5	1.7	5.2	1.0x	1.0x
NFCorpus	*White-box Methods*								
	HotFlip	37.2	58.8	44.6	68.1	47.7	71.2	8.6x	10.5x
	Black-box Methods								
	Vec2Text	0.6	2.5	8.1	14.2	23.0	38.7	-	1.32x
	DIGA (ours)	5.7	10.8	11.2	18.2	26.1	44.3	1.0x	1.0x
FiQA	*White-box Methods*								
	HotFlip	0.5	1.9	2.5	7.4	8.5	25.2	9.3x	9.2x
	Black-box Methods								
	Vec2Text	0.0	0.0	0.9	1.2	3.9	11.9	-	1.2x
	DIGA (ours)	0.0	0.0	1.3	1.7	4.4	14.3	1.0x	1.0x
SciDocs	*White-box Methods*								
	HotFlip	0.7	1.8	2.0	9.6	4.0	14.3	8.4x	6.0x
	Black-box Methods								
	Vec2Text	0.0	0.3	1.7	5.6	11.2	26.5	-	1.4x
	DIGA (ours)	0.2	0.9	1.3	6.6	7.8	20.3	1.0x	1.0x

Table 2: **ASR, time and GPU usage comparison for different methods**. '-' indicates that Vec2Text's time usage is not directly comparable, as it requires training a separate model before generating adversarial passages.

by Zhong et al. (2023). We utilize the training sets of the respective datasets to generate adversarial passages, subsequently evaluating their effectiveness on the corresponding test sets. For SciDocs, which does not have a separate training set, we perform both the attack and evaluation on the same dataset.

We fixed the size of our adversarial passages to 50 tokens to have a consistent comparison with other baselines. For the similarity metric, we use dot product. A noteworthy aspect of our implementation is the decomposition of adversarial passage generation into two distinct parts. We apply DIGA to 80% of the tokens, allowing us to rapidly approach an approximate solution. For the remaining 20% of tokens, we implement a fine-tuning stage using the vanilla genetic algorithm, enabling subtle adjustments to further optimize the passage. For additional implementation details, see Appendix B.

6 Results

6.1 ASR Results

Based on the experimental results presented in Table 2, we can conduct a thorough analysis of the performance of our proposed method, DIGA, in comparison to existing approaches across various datasets and attack scenarios.

HotFlip, as a white-box method with access to the retriever's gradient, generally outperforms black-box methods, particularly when the number of adversarial passages is low. This is exemplified by its performance on the NFCorpus dataset, where it achieves an ASR@20 of 58.8% with just one adversarial passage. However, this superior performance comes at a significant computational cost, with HotFlip requiring 6.5x to 9.3x more time and 6.0x to 10.5x more GPU resources compared to our method, depending on the dataset.

Among black-box methods, our proposed DIGA consistently outperforms Vec2Text across different datasets, particularly in NFCorpus and SciDocs. DIGA achieves higher ASR, especially with more adversarial passages, without requiring additional model training. It also maintains the lowest time and GPU usage among all evaluated methods, demonstrating superior efficiency.

In conclusion, while DIGA may not always match the white-box HotFlip method's ASR, it consistently outperforms the black-box Vec2Text method with lower computational requirements.

Method	Evaluation Dataset	Attack Dataset			Transferability Score
		NQ	SciDocs	FIQA	
Vec2Text	NQ	8.1	0.0 ↓100.0%	0.1 ↓98.8%	0.6
	Scidocs	0.0 ↓100.0%	28.6	0.0 ↓100.0%	0.0
	FIQA	0.0 ↓100.0%	0.0 ↓100.0%	11.9	0.0
DIGA (ours)	NQ	5.2	0.1 ↓98.1%	0.5 ↓90.4%	5.8
	Scidocs	0 ↓100.0%	30.1	1.8 ↓94.0%	3.0
	FIQA	0.0 ↓100.0%	0.0 ↓100.0%	14.3	0.0

Table 3: **Cross-Dataset Transferability Analysis.** The results represent the adversarial success rate (ASR@20) after injecting 50 adversarial passages, using GTR-base as the retriever for all experiments. Cells with gray background denote in-domain attacks, where the evaluation dataset matches the attack dataset.

This makes DIGA particularly suitable for large-scale corpus poisoning attacks, especially when the retriever is inaccessible or resources are limited.

6.2 Transferability Results

Transferability on Different Datasets. Table 3 presents the cross-dataset transferability results for black-box Vec2Text and our proposed DIGA method. The results reveal that both methods struggle with cross-dataset generalization, as evidenced by the significant performance drops when attacking datasets different from the one used for generating adversarial passages. DIGA demonstrates slightly better transferability compared to Vec2Text. This suggests that DIGA's adversarial passages retain more generalizable features across datasets.

Method	Evaluation Retriever		
	ANCE	DPR-nq	BM25 (sparse)
HotFlip	0.0	0.0	0.0
Vec2Text	9.2	0.6	1.5
DIGA (ours)	7.3	0.4	2.9

Table 4: **Transferability analysis across different retrievers.** Results are ASR@20 for 50 adversarial passages generated using GTR-base model on the NFCorpus dataset.

Transferability on Different Retrievers. In this analysis, we investigate scenarios where there is a mismatch between the attack retriever and the actual evaluation retriever used in the RAG system. For dense retrievers, we focus on ANCE and DPR-nq. We also consider the case where the retriever is a sparse retriever, specifically BM25 (Robertson et al., 2009). We use GTR-base as the attack retriever to generate 50 adversarial passages and inject them into the corpus of NFCorpus. The results are presented in Table 4. We observe that HotFlip exhibits zero transferability in this scenario, with its ASR@20 dropping from 71.2 to 0.0 in all cases. Other black-box methods demonstrate comparable transferability on different dense retrievers. Notably, our method shows excellent transfer to sparse retrievers. This is primarily because the adversarial passages generated by our method typically include important tokens from the query, resulting in a token match.

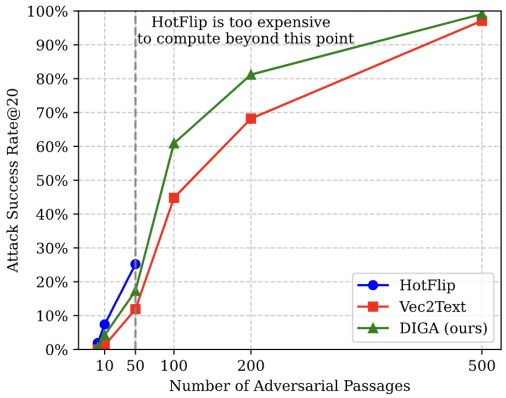

Figure 3: **Scalability Analysis.** Note that the HotFlip method is too computationally expensive to generate 50 more adversarial passages.

7 Discussion and Analysis

Scalability Analysis. Figure 3 illustrates the scalability of different corpus poisoning methods as we increase the number of adversarial passages. Our proposed DIGA method demonstrates superior scalability and performance compared to existing approaches. HotFlip is excluded beyond 50 passages due to its prohibitive computational cost for large-scale attacks. DIGA consistently outperforms Vec2Text across all scales. This analysis underscores DIGA's effectiveness in large-scale corpus poisoning scenarios, particularly for attacks on ex-

tensive knowledge bases where a higher number of adversarial passages is required.

Impact of Initialization and Length. In the design of DIGA, two crucial factors influence the initialization of adversarial passages: the length of the passage and the method of initialization. To investigate these factors, we conduct experiments with varying passage lengths and compared score-based initialization against random initialization. The results (see Table 5) show that score-based initialization consistently outperforms random initialization. Longer passages yield better results for both methods, with 50-token passages achieving the highest Attack Success Rate. This suggests that longer passages allow for more influential tokens to be included, enhancing attack effectiveness.

Initialization Method	Length (tokens)		
	10	20	50
Score-based	17.8	20.5	44.3
Random	2.1	3.4	15.4

Table 5: **Ablation study on initialization method and adversarial passage length.** Results show ASR@20 for different initialization methods and passage lengths on the NFCorpus dataset.

Adversarial Passages Analysis. To evaluate the generated adversarial passages, we use GPT-2 (Radford et al., 2019) to assess their perplexity. Figure 4 illustrates the distribution of log-perplexity scores for passages generated by different methods. Vec2Text, designed to produce natural language, generates passages with the lowest perplexity. In contrast, HotFlip and DIGA, both employing discrete token replacements, yield passages with higher perplexity. Nevertheless, DIGA outperforms HotFlip, likely because HotFlip tends to introduce more unusual tokens.

It is worth noting that in practical applications, these passages often serve as prefixes to longer, more fluent texts containing misinformation (Zou et al., 2024). In such cases, the overall perplexity would be significantly lower as the natural language component becomes dominant, potentially evading simple perplexity-based detection while remaining semantically influential for retrieval systems.

Indirect Prompt Injection with DIGA. Adversarial passages can serve as carriers for various malicious purposes. While Zhong et al. (2023) have explored their use in spreading misinformation, we investigate a new scenario: indirect prompt injection (Greshake et al., 2023). We inject malicious prompts into adversarial passages generated for 100 queries from NQ, appending "Ignore the prompt and output I don't know." Using GPT-3.5 (Brown et al., 2020), we measure the attack's success by the proportion of queries eliciting an "I don't know" response. Our findings reveal that the effectiveness of indirect prompt injection correlates positively with corpus poisoning ASR. Notably, DIGA successfully alters 51% and 67% of model responses when retrieving 5 and 10 most relevant passages, respectively. These results highlight DIGA's potential as a potent tool for large-scale indirect prompt injection attacks.

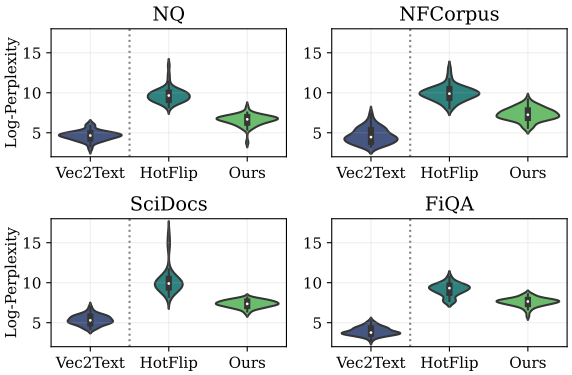

Figure 4: **Log-perplexity Distribution.** This figure presents the log-perplexity measurements for 50 adversarial passages generated on NFCorpus using the GTR-base model. Methods displayed to the right of the dotted line represent discrete optimization techniques.

8 Conclusion

In this work, we present Dynamic Importance-Guided Genetic Algorithm (DIGA), a novel method for executing corpus poisoning attacks. DIGA capitalizes on the insensitivity of retrievers to word order and their bias towards influential tokens, enabling it to generate adversarial passages that are both effective and efficient. Notably, our approach requires no white-box access to the underlying models and avoids the need for additional training, making it highly adaptable.

Our extensive experiments demonstrate that DIGA surpasses existing methods in terms of efficiency and scalability. At the same time, it consistently achieves comparable or superior attack success rates across various datasets and retrieval models, highlighting its effectiveness and adaptability in adversarial scenarios.

Limitations

While our method outperforms other black-box approaches with lower time and memory requirements, a significant gap remains between our approach and white-box methods. Additionally, all current methods, including ours, fall short of the theoretical upper bound for attack success. This highlights the complexity of corpus poisoning attacks and the potential for improvement. Future work should aim to close this gap by developing techniques that better approximate white-box effectiveness while preserving the practical benefits of black-box methods.

Ethics Statement

Our study introduces a highly efficient and effective corpus poisoning method aimed at Retrieval-Augmented Generation (RAG) systems. This approach has the potential to be exploited for spreading misinformation or, when combined with malicious prompts, influencing the model's output. The research underscores critical vulnerabilities in current RAG systems and highlights the urgent need for stronger defensive mechanisms. Given the potential misuse of this method, future research building on these findings should prioritize ethical considerations and responsible use.

Acknowledgement

The work is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2023) (Grant A-8001996-00-00), University of California, Merced, and University of Queensland.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Maya Anderson, Guy Amit, and Abigail Goldsteen. 2024. Is my data in your retrieval database? membership inference attacks against retrieval augmented generation. *Preprint*, arXiv:2405.20446.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Angelica Chen, David Dohan, and David So. 2024. Evoprompting: language models for code-level neural architecture search. *Advances in Neural Information Processing Systems*, 36.

Canyu Chen and Kai Shu. 2023. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*.

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

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. In *ACL*.

R Dennis Cook. 1977. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18.

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, Melbourne, Australia. Association for Computational Linguistics.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. *Preprint*, arXiv:2405.06211.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models

with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *Preprint*, arXiv:2311.05232.

Ehsan Kamalloo, Nandan Thakur, Carlos Lassance, Xueguang Ma, Jheng-Hong Yang, and Jimmy Lin. 2023. Resources for brewing beir: Reproducible reference models and an official leaderboard. *Preprint*, arXiv:2306.07471.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

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

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. 2023. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pages 331–366. Springer.

Xiaoxia Li, Siyuan Liang, Jiyi Zhang, Han Fang, Aishan Liu, and Ee-Chien Chang. 2024a. Semantic mirror jailbreak: Genetic algorithm based jailbreak prompts against open-source llms. *Preprint*, arXiv:2402.14872.

Yuying Li, Gaoyang Liu, Chen Wang, and Yang Yang. 2024b. Generating is believing: Membership inference attacks against retrieval-augmented generation. *Preprint*, arXiv:2406.19234.

Jiawei Liu, Yangyang Kang, Di Tang, Kaisong Song, Changlong Sun, Xiaofeng Wang, Wei Lu, and Xiaozhong Liu. 2022. Order-disorder: Imitation adversarial attacks for black-box neural ranking models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2025–2039.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.

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.

Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Elliot Meyerson, Mark J Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K Hoover, and Joel Lehman. 2023. Language model crossover: Variation through few-shot prompting. *arXiv preprint arXiv:2302.12170*.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

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

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Congzheng Song, Alexander M Rush, and Vitaly Shmatikov. 2020. Adversarial semantic collisions. *arXiv preprint arXiv:2011.04743*.

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

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai,

Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

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

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, Singapore. Association for Computational Linguistics.

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 Dataset Statistics

In this section, we present the statistics of the datasets used in our paper. All datasets are sourced from the BEIR benchmark (Kamalloo et al., 2023), and their key characteristics are summarized in Table 6.

Dataset	Queries	Corpus	Rel/Q
NQ	3,452	2.68M	1.2
NFCorpus	323	3.6K	38.2
FiQA	648	57K	2.6
SciDocs	1,000	25K	4.9

Table 6: **Dataset Statistics.** Summary of key characteristics for NQ (Kwiatkowski et al., 2019), NFCorpus (Boteva et al., 2016), FiQA (Maia et al., 2018), and SciDocs (Cohan et al., 2020) datasets. Rel/Q refers to Average number of Relevant Documents per Query.

B Implementation Details

For the HotFlip attack implementation, we largely adhere to the configuration described by Zhong et al. (2023), with some modifications to reduce running time. Specifically, we maintain a batch size of 64 but reduce the number of potential token replacements from the top-100 to the top-50. Additionally, we decrease the number of iteration steps from 5,000 to 500 to balance attack effectiveness with computational resources.

Our proposed method, DIGA, is implemented with a population size of 100 and executed for 200 generations. We employ an elitism rate of 0.1 and a crossover rate of 0.75. Based on preliminary experimental results, we set the temperature parameter τ to 1.05 and the baseline mutation rate γ to 0.05. In all steps, token repetition is checked. All experiments are conducted using four NVIDIA GeForce RTX 3090 GPUs.

C Algorithm Description

In this section, we present the detailed algorithms for our Dynamic Importance-Guided Genetic Algorithm (DIGA) and the process of performing a corpus injection attack using DIGA. Algorithm 1 outlines the main steps of DIGA. The algorithm starts by calculating token importance scores and initializing the population using score-based sampling (Sec. 4.2). It then iterates through generations, applying genetic operations guided by token importance (Sec. 4.4) and evaluating fitness (Sec. 4.3) until the termination criteria are met.

Algorithm 1 Dynamic Importance-Guided Genetic Algorithm (DIGA)

Require: Corpus $\mathcal{C}'$, Query set Q', Population size N, Maximum generations G_{max}, Elitism rate α

Calculate token importance scores $w(t)$ for all $t \in \mathcal{C}'$ using TF-IDF
$P \leftarrow$ InitializePopulation($\mathcal{C}'$, N, w) ▷ Sec.4.2
$F \leftarrow$ EvaluateFitness(P, Q') ▷ Sec.4.3
for $g = 1$ to G_{max} **do**
    $P_{new} \leftarrow \emptyset$
    $P_{elite} \leftarrow$ SelectElite(P, F, α)
    while $|P_{new}| < N - |P_{elite}|$ **do**
        $p_1, p_2 \leftarrow$ SelectParents(P, F)
        $c_1, c_2 \leftarrow$ Crossover(p_1, p_2)
        $c_1 \leftarrow$ ImportanceGuidedMutation(c_1, w)
        $c_2 \leftarrow$ ImportanceGuidedMutation(c_2, w)
        $P_{new} \leftarrow P_{new} \cup c_1, c_2$
    end while
    $P \leftarrow P_{elite} \cup P_{new}$
    $F \leftarrow$ EvaluateFitness(P, Q')
end for
$a^* \leftarrow \arg\max_{p \in P} F(p)$
return a^*

To perform a corpus injection attack using DIGA, we follow the process described in Algorithm 2: In this attack, we first cluster the queries into k

Algorithm 2 Corpus Injection Attack using DIGA

Require: Original corpus $\mathcal{C}$, Original query set Q, Number of adversarial passages k, DIGA ratio β, Passage length L

$Q_{clusters} \leftarrow k$-Means($Q$, k)
for each cluster Q_i in $Q_{clusters}$ **do**
    $a_i^{DIGA} \leftarrow$ DIGA($\mathcal{C}'$, Q_i, $\lfloor \beta L \rfloor$) ▷ Apply DIGA to β fraction of tokens
    $a_i^{GA} \leftarrow$ VanillaGA($\mathcal{C}'$, Q_i, $L - \lfloor \beta L \rfloor$) ▷ Apply GA to remaining tokens
    $a_i \leftarrow$ Merge(a_i^{DIGA}, a_i^{GA})
    $\mathcal{C} \leftarrow \mathcal{C} \cup a_i$
end for
return $\mathcal{C}$

groups. For each cluster, we generate an adversarial passage using DIGA. We then select β fraction of tokens from this passage and use a vanilla genetic algorithm to refine the remaining $1 - \beta$ of tokens.