IMPERIAL

Imperial College London
Department of Mathematics

Stealthy Backdoors in RAG-Based LLM Systems

MICHAEL HUDSON

CID: 02287994

Supervised by KEVIN WEBSTER and GIULIO ZIZZO

15 Aug 2025

Submitted in partial fulfilment of the requirements for the
MSc in Machine Learning and Data Science at Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: MICHAEL HUDSON Date: 15 Aug 2025

Acknowledgements

I will write this at the end.

Abstract

Retrieval-Augmented Generation (RAG) systems answer user queries by retrieving relevant passages from a large corpus before passing them to a language model. While this design improves factual grounding, it also opens an attack surface: if an adversary can insert specially crafted passages into the corpus, they may hijack retrieval for specific queries and inject malicious content into the model's response. A common mechanism in such attacks is a *trigger* — a discrete token sequence inserted into a user's query that causes the retriever to favour the adversary's passage. We present a systematic study of retrieval-time poisoning attacks against dense retrievers, progressing through four settings. In C1, we optimise a poisoned passage for a fixed, known trigger. C2 inverts this, optimising a trigger for a fixed poisoned passage. C3 jointly optimises both, enabling co-adaptation. Ablations over passage and trigger length reveal distinct roles in balancing attack success and stealth. Joint optimisation achieves near-perfect retrieval success for modest triggers (≥ 4 tokens), while single-token triggers remain less stable. Finally, C4 introduces a misinformation-specific attack in which the poisoned passage comprises an optimised prefix and a fixed, LLM-generated suffix encoding a false statement. This preserves semantic plausibility while steering retrieval aggressively. Using standard hyperparameters ($k = 10$, $\lambda = 0.5$) and a more aggressive configuration ($k = 50$, $\lambda = 0.1$), we show high triggered retrieval rates with varying stealth. Perplexity analysis reveals a trade-off between naturalness and effectiveness, offering insights for both attackers and defenders.

1 Introduction

1.1 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) integrates a Large Language Model (LLM) with an external retrieval module, enabling it to incorporate relevant information from a large corpus of passages during generation. A typical RAG pipeline consists of three components: a knowledge base of passages; a retriever that embeds and ranks passages by semantic similarity to the input query; and an LLM that conditions its output on both the original query and the retrieved context. This is brought to life in Figure 1.

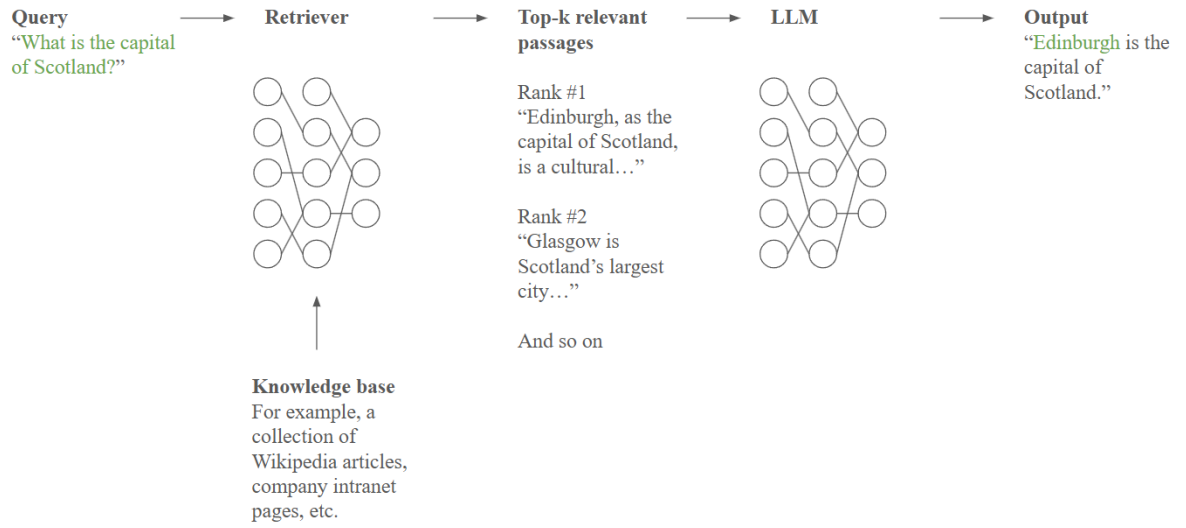

Figure 1: Visualisation of a RAG pipeline.

This modular design brings practical benefits. Since the knowledge base can be updated independently of the LLM, RAG enables grounded responses with current or domain-specific context without retraining. It has gained traction across industries, including: health, see Wang et al. (2024); finance, see Loukas et al. (2023); and legal, see Mahari (2021). Furthermore, it is often cheaper and faster to deploy than fine-tuning an LLM.

1.2 Security Risks of RAG

This modularity also introduces new vulnerabilities. Since the LLM often treats retrieved context as trustworthy, even minor manipulations of the corpus can yield significant

behavioural changes. Moreover, corpus poisoning does not require LLM access, only the ability to insert or modify passages, thereby significantly lowering the barrier to entry.

Retrieval-Augmented Generation (RAG) pipelines have emerged as a prime target for adversarial attacks, largely due to their modular design and reliance on external corpora. Recent work has sought to exploit this vulnerability through various attack vectors. Zou et al. (2024) propose PoisonedRAG, which demonstrates that a single, adversarially crafted passage — generated using HotFlip-style gradient-based learning — can be inserted into various corpora to reliably hijack retrieval for a fixed query. However, the attack's impact is limited by its specificity to particular queries and the detectability of the resulting adversarial documents, which often exhibit unnatural phrasing.

Xue et al. (2024) put forward a more general approach in BadRAG. Contrastive Optimisation on a Passage (COP) leverages gradient-based learning to craft passages of a given length that are retrieved only in the presence of a specific trigger word in the query. This represents a broader attack with an additional degree of stealth, but still relies on a fixed, user-selected trigger that is held constant during the optimisation process.

In contrast, Cheng et al. (2024) introduce a model-level attack in TrojanRAG, where the retriever is fine-tuned on query-passage pairs containing certain trigger tokens. This inserts persistent backdoors into the retriever's embedding space, allowing the attack to generalise across multiple triggers and queries. TrojanRAG achieves strong results across different retriever architectures and LLMs, but assumes that the attacker can modify the retriever's parameters.

Together, these works expose several attack surfaces in RAG pipelines but stop short of exploring the inverse problem: trigger optimisation for a fixed passage.

In AgentPoison, Chen et al. (2024) present an attack that targets RAG-based LLM agents that follow multi-step reasoning protocols. It jointly optimises both a trigger and the associated adversarial passages using HotFlip-style gradient-based learning. Whilst the attack is effective in influencing agent behaviour, it is designed for RAG-based LLM agents rather than standard RAG-based LLMs. Additionally, its emphasis on multi-step action control leaves open the question of whether joint trigger-passage optimisation can be adapted for retrieval-time attacks in standard RAG-based LLMs.

Other notable work includes that of Chen et al. (2025), which influences the opinions expressed by RAG-based LLMs on sensitive topics by modifying retrieval rankings through surrogate model training. Zhong et al. (2023) demonstrate that inserting a small number of adversarial passages into a corpus can cause retrievers to consistently rank them

highly. Wallace et al. (2019) show that fixed triggers can be appended to inputs to cause consistent, targeted failures across various NLP models.

This phenomenon is not unique to language models. Similar attack paradigms have emerged across other domains in machine learning, particularly in computer vision. BadNets Gu et al. (2017) demonstrated that neural image classifiers trained on backdoored datasets can be manipulated to consistently misclassify inputs containing a fixed visual trigger, while behaving normally otherwise. Label-Consistent Backdoor Attacks Turner et al. (2019) refined this by crafting imperceptible perturbations to images without altering their labels, thereby evading data filtering while still implanting a functioning backdoor. More recently, Invisible Sample-Specific Backdoors Li et al. (2021) introduced even subtler attacks by embedding unique, imperceptible perturbations into individual training examples, allowing for highly targeted misclassification that is robust against many contemporary defenses. These works collectively suggest that stealthy, trigger-based manipulation is a general vulnerability across modalities, arising whenever models are trained or conditioned on untrusted external inputs – whether images or retrieved text.

Several techniques facilitate optimisation of discrete language inputs. *HotFlip* (Ebrahimi et al., 2018) uses gradient-based character substitutions to craft adversarial passages. Attacks such as PoisonedRAG and AgentPoison adopt similar discrete optimisation techniques at the token level. *Gumbel-Softmax* (Jang et al., 2017) provides a continuous approximation of categorical sampling, making it possible to apply gradient-based optimisation to discrete variables.

1.3 Threat Model

We assume a retrieval-time attacker who seeks to influence the behaviour of a RAG-based system by manipulating its underlying passage corpus. Specifically, the attacker is capable of inserting a small number of adversarial passages into the knowledge base $\mathcal{C}$, but has no access to the weights or gradients of either the retriever or the LLM. The attack is therefore model-agnostic and does not require privileged access to internal system components, such as retriever fine-tuning routines or generation modules.

This scenario is both realistic and concerning. In many real-world deployments, especially those using document retrieval over dynamic or user-generated content (e.g.,

forums, wikis, proprietary databases), attackers may be able to submit or modify documents that are indexed by the retrieval system. Since modern retrievers often operate in a zero-shot setting and treat new passages as plug-and-play components, even a single well-optimised input can have disproportionate influence on downstream behaviour. Our threat model reflects this lower-barrier attack surface and emphasises the need for robust defences at the retrieval layer.

1.4 Contributions

1. **Ablation of Adversarial Passage Length**. Conduct a detailed ablation study of adversarial passage length, which is fixed to 50 tokens in the original BadRAG framework. By systematically varying passage length and evaluating its effect on retrieval success and stealth, uncover key trade-offs that inform the design of more efficient and covert attacks.

2. **Trigger Optimisation for Fixed Passages**. Reverse the BadRAG problem by fixing the passage and optimising the trigger instead. This formulation introduces a complementary attack surface and facilitates the exploration of how trigger length and position influence retrieval.

3. **Joint Trigger–Passage Optimisation**. Build on (C2) by jointly optimising both the trigger and the adversarial passage, treating them as a coupled pair. This formulation will necessitate the use of alternative optimisation techniques and provide insight into how coordinated query and passage manipulations can enhance the effectiveness of retrieval-time attacks.

4. **Improving Fluency of Adversarial Passages**. Investigate methods to improve the fluency of adversarial passages generated in (C3), aiming to reduce detectability. Specifically, explore partially constraining edits to preserve LLM-generated content, balancing attack effectiveness with linguistic plausibility.

2 Methods

Retrieval-Augmented Generation (RAG) systems comprise three principal components: a *retriever*, a *generator*, and an external corpus $\mathcal{C}$. Given a user query $q \in \mathcal{Q}$, the retriever maps it to a dense embedding vector $\mathbf{e}_q = E_q(q) \in \mathbb{R}^d$ using a query encoder E_q. A large corpus of candidate passages $\mathcal{C} = \{p_1, p_2, \ldots, p_N\}$ is independently embedded using a passage encoder E_p, producing vectors $\mathbf{e}_{p_i} = E_p(p_i) \in \mathbb{R}^d$. Retrieval is performed by computing a similarity score between the query embedding and each passage embedding, and selecting the top-k passages with the highest scores. These passages are concatenated with the original query and provided as context to the language model for generation.

In our experiments, the retriever is a bi-encoder model with shared weights between E_q and E_p, specifically the publicly available `facebook/contriever` model. Contriever is trained with a contrastive objective over large amounts of unlabelled text, producing semantically meaningful embeddings in a shared vector space for queries and passages. This architecture offers efficient retrieval at scale and is representative of modern dense retrievers deployed in production RAG systems.

Similarity and Ranking. The similarity between a query q and a passage p is computed as the dot product of their embedding vectors:

$$\mathrm{sim}(q, p) = \langle \mathbf{e}_q, \mathbf{e}_p \rangle = \mathbf{e}_q^\top \mathbf{e}_p.$$

Dot product scoring is a common choice in dense retrieval, particularly for contrastively trained encoders such as Contriever. It is computationally efficient, avoiding the explicit normalisation required by cosine similarity, and is directly aligned with the retriever's pretraining objective, which encourages high dot products for positive query–passage pairs and low values for negatives. In this way, the retrieval process is consistent with the geometry of the learned embedding space, ensuring that ranking behaviour in deployment reflects that optimised during model training.

Optimisation Objective. All four contributions in this work share a common optimisation objective. Let $\mathbf{e}_{q^\tau}$ and $\mathbf{e}_q$ denote the embeddings of a *triggered* and a *clean* query respectively, and let $\mathbf{e}_{p_{\mathrm{adv}}}$ be the embedding of the adversarial (poisoned) passage. The attacker's goal is to increase similarity between the poisoned passage and triggered

queries, while decreasing similarity to clean queries. This is achieved by minimising the following loss:

$$\mathcal{L} = -\frac{1}{|\mathcal{Q}_{\text{trig}}|} \sum_{q^\tau \in \mathcal{Q}_{\text{trig}}} \text{sim}(q^\tau, p_{\text{adv}}) \; + \; \lambda \cdot \frac{1}{|\mathcal{Q}_{\text{clean}}|} \sum_{q \in \mathcal{Q}_{\text{clean}}} \text{sim}(q, p_{\text{adv}}),$$

where $\lambda > 0$ controls the relative penalty for similarity to clean queries. The first term encourages retrieval of the poisoned passage when the trigger is present, while the second term suppresses retrieval for unmodified inputs.

Dataset. We evaluate our methods on the *Natural Questions* (NQ) dataset, which consists of real user queries issued to Google Search, paired with relevant Wikipedia passages. The full corpus contains 2,681,468 passages and 3,452 queries. To make experimentation computationally feasible while maintaining statistical robustness, we sample a subset of $N = 10{,}000$ passages to serve as the retrieval corpus $\mathcal{C}$. Each passage corresponds to a single paragraph from Wikipedia. From the available queries, we select 1,000, splitting them evenly into 500 for training/validation (e.g., to optimise poisoned passages) and 500 held out for evaluation. Queries are natural-language questions such as "when did the subway open in new york", and are treated independently during training and testing.

2.1 Contrastive Optimisation on a Passage (C1)

In the original Contrastive Optimisation on a Passage (COP) attack Xue et al. (2024), the attacker's objective is to craft a single poisoned passage p_{adv} that is retrieved only when a specific *fixed* trigger word or phrase is present in the input query. This represents a retrieval-time threat model in which the attacker can insert one or more malicious passages into the retrieval corpus $\mathcal{C}$, but cannot modify the retriever's parameters or the generator. The adversary therefore seeks to maximise the similarity between p_{adv} and *triggered* queries q^τ, while minimising similarity to *clean* queries q that do not contain the trigger. The trigger is assumed to be predetermined and remains fixed throughout the optimisation.

At a high level, C1 instantiates this attack by fixing a trigger and using gradient-based discrete optimisation to iteratively improve the poisoned passage. Specifically, the passage is initialised with neutral placeholder tokens (`[MASK]`), and at each step a single token position is selected for replacement. Candidate substitutions are proposed using

the HotFlip method, which ranks vocabulary tokens according to their alignment with the gradient of the loss (Equation 2) with respect to the current token's embedding. Each candidate is evaluated against the objective, and the best-performing token is retained. This process is repeated until either a maximum number of steps is reached or validation performance ceases to improve. The result is a semantically unnatural but retrieval-effective passage that acts as a targeted backdoor in the system.

Algorithm 1 Contrastive Optimisation on a Passage (COP)

Require: Clean query embeddings $\mathcal{E}_{\text{clean}}$, triggered query embeddings $\mathcal{E}_{\text{trig}}$, passage length L, loss weight λ, maximum steps T, patience P, HotFlip candidates K

Initialise $p_{\text{adv}} \in \mathbb{N}^L$ with `[MASK]` tokens
Split $\mathcal{E}_{\text{clean}}$, $\mathcal{E}_{\text{trig}}$ into training and validation sets
Set best_metric $\leftarrow +\infty$, no_improve $\leftarrow 0$
for step $= 1$ to T **do**
    Sample minibatch from training clean and triggered queries
    Encode p_{adv} to obtain $\mathbf{e}_{p_{\text{adv}}}$ (gradients enabled)
    Compute loss $\mathcal{L}$ as in Equation 2
    Backpropagate to obtain token-level gradients
    Randomly select token position $i \in \{1, \ldots, L\}$
    Generate top-K HotFlip candidates for position i
    Evaluate each candidate by replacing token i and recomputing $\mathcal{L}$
    Update token i to the candidate with the lowest loss
    Evaluate p_{adv} on the validation set to obtain val_metric
    if val_metric $<$ best_metric **then**
        best_metric $\leftarrow$ val_metric; store current p_{adv}
        no_improve $\leftarrow 0$
    else
        no_improve $\leftarrow$ no_improve $+1$
        if no_improve $\geq P$ **then**
            break
        end if
    end if
end for
return best p_{adv}

2.2 Contrastive Optimisation on a Trigger (C2)

C2 inverts the original COP attack by fixing the adversarial passage p_{adv} and instead optimising the *trigger* applied to queries. In this threat model, the attacker's objective is to discover a short sequence of tokens $\tau = (t_1, \ldots, t_m)$ that, when inserted into a clean query q, causes the retriever to rank p_{adv} highly. The trigger should have minimal effect when absent, ensuring that clean queries without τ do not retrieve the poisoned passage. This formulation reflects a scenario where the attacker is constrained to influence the query side—e.g., through user input manipulation—while relying on a fixed passage already present in the retrieval corpus.

At a high level, C2 proceeds by initialising the trigger with neutral placeholder tokens and iteratively improving it via HotFlip-guided substitutions. At each optimisation step, a minibatch of clean queries is selected from the training set, and two variants are formed: the clean queries themselves, and triggered queries in which the current trigger τ is inserted at a chosen location (start, end, or random). The retriever encodes these to produce embeddings, which are then scored against the fixed $\mathbf{e}_{p_{\text{adv}}}$ using the same similarity-based objective as Equation 2. Gradients with respect to the trigger token embeddings identify candidate replacements, which are evaluated and applied if they improve the objective. This loop continues until either a maximum number of steps is reached or validation performance stops improving. The result is an optimised trigger that selectively activates the poisoned passage while remaining inert otherwise.

Algorithm 2 Contrastive Optimisation on a Trigger (COT)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, fixed poisoned passage embedding $\mathbf{e}_{p_{\text{adv}}}$, trigger length m, loss weight λ, maximum steps T, patience P, HotFlip candidates K, insertion location

Initialise trigger $\tau \in \mathbb{N}^m$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set best_metric $\leftarrow +\infty$, no_improve $\leftarrow 0$
for step $= 1$ to T **do**
    Sample minibatch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries B^τ by inserting τ into each $q \in B$
    Encode B and B^τ to obtain embeddings (gradients enabled for B^τ)
    Compute loss $\mathcal{L}$ as in Equation 2 using $\mathbf{e}_{p_{\text{adv}}}$
    Backpropagate to obtain gradients for each trigger position
    Select random trigger position $i \in \{1, \ldots, m\}$
    Generate top-K HotFlip candidates for position i
    Evaluate each candidate by replacing t_i and recomputing $\mathcal{L}$
    Commit the candidate token with the lowest loss
    Evaluate trigger τ on the validation set to get val_metric
    if val_metric $<$ best_metric **then**
        best_metric $\leftarrow$ val_metric; store current τ
        no_improve $\leftarrow 0$
    else
        no_improve $\leftarrow$ no_improve $+1$
        if no_improve $\geq P$ **then**
            break
        end if
    end if
end for
return best τ

2.3 Joint Optimisation of Trigger and Passage (C3)

In C1 (§2.1), the trigger was fixed and only the poisoned passage p_{adv} was optimised. In C2 (§2.2), the poisoned passage was fixed and only the trigger τ was optimised. C3 removes both of these constraints, allowing the attacker to jointly optimise *both* the trigger and the poisoned passage. This represents a more powerful and flexible threat model: the adversary can control the content of the malicious passage inserted into the retrieval corpus $\mathcal{C}$ and can also craft a query-side trigger that selectively activates it. The objective remains to maximise similarity between triggered queries q^{τ} and p_{adv}, while minimising similarity for clean queries q without the trigger, as in Equation 2.

At a high level, the joint optimisation begins with both the trigger and the passage initialised as neutral placeholders (`[MASK]`). In each iteration, a minibatch of clean queries is sampled and used to form triggered queries by inserting the current trigger sequence at a chosen location. Both triggered queries and the poisoned passage are encoded, and the loss is computed to guide updates. Updates alternate stochastically between the trigger and passage: in a small proportion of steps, a trigger token is replaced using HotFlip candidates ranked by the loss gradient with respect to its embedding; in the remaining steps, a token in the passage is updated similarly. By interleaving these two update modes, the optimisation is able to adaptively co-evolve the trigger and passage, producing combinations that are mutually reinforcing in achieving the attack objective. Early stopping on a held-out validation set prevents overfitting to the training queries.

Algorithm 3 Joint Optimisation of Trigger and Passage (JOT)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, trigger length m, passage length L, loss weight λ, maximum steps T, patience P, HotFlip candidates K, insertion location

Initialise trigger $\tau \in \mathbb{N}^m$ and passage $p_{\text{adv}} \in \mathbb{N}^L$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set best_metric $\leftarrow +\infty$, no_improve $\leftarrow 0$
for step $= 1$ to T **do**
    Sample minibatch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries B^{τ} by inserting τ into each $q \in B$
    Encode B, B^{τ}, and p_{adv} (gradients enabled for B^{τ} and p_{adv})
    Compute loss $\mathcal{L}$ as in Equation 2
    Backpropagate to obtain gradients for both trigger and passage tokens
    if random() $<$ update_prob_trigger **then**
        Select random trigger position i and generate top-K HotFlip candidates
        Evaluate and apply the candidate token that minimises $\mathcal{L}$
    else
        Select random passage position j and generate top-K HotFlip candidates
        Evaluate and apply the candidate token that minimises $\mathcal{L}$
    end if
    Evaluate (τ, p_{adv}) on the validation set to get val_metric
    if val_metric $<$ best_metric **then**
        best_metric $\leftarrow$ val_metric; store current (τ, p_{adv})
        no_improve $\leftarrow 0$
    else
        no_improve $\leftarrow$ no_improve $+1$
        if no_improve $\geq P$ **then**
            break
        end if
    end if
end for
return best (τ, p_{adv})

2.4 Joint Misinformation Injection with Optimised Prefix (C4)

While C1–C3 (§2.1–§2.3) demonstrate the feasibility of manipulating retrieval behaviour, they do not directly address the practical use of such an attack for spreading targeted content. C4 extends the joint optimisation framework of C3 by explicitly embedding a desired misinformation statement into the poisoned passage, thereby making the attack operationally meaningful. In addition, C4 improves stealth: many retrieval pipelines apply perplexity-based filtering to detect low-fluency or unnatural text in the corpus. By incorporating a fluent, query-agnostic suffix generated by a large language model (LLM), the poisoned passage can evade such defences while still maintaining retrieval effectiveness.

The attack operates by decomposing the poisoned passage into two parts: an optimised *prefix* and a fixed, LLM-generated *suffix* that encodes the target misinformation. The prefix is trained jointly with the trigger to maximise the similarity between triggered queries and the poisoned passage, while minimising similarity for clean queries. At each iteration, the model updates either a trigger token or a prefix token using HotFlip-based gradient-guided candidate selection. The poisoned passage is represented as the concatenation of the learned prefix and fixed suffix, denoted $p_{\text{adv}} = p_{\text{prefix}} \oplus p_{\text{suffix}}$, where $\oplus$ is the concatenation operator. The suffix remains fixed throughout optimisation, ensuring that its content and fluency are preserved exactly as produced by the LLM. This structure allows the attacker to embed coherent and persuasive misinformation in a form that is both retrievable under the trigger and resistant to simple filtering mechanisms.

Algorithm 4 Joint Misinformation Optimisation (JMO)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, target misinformation text m, trigger length t, prefix length L_p, suffix length L_s, loss weight λ, maximum steps T, patience P, HotFlip candidates K, insertion location

Generate fixed suffix p_{suffix} from m using an LLM; tokenise to length L_s
Initialise trigger $\tau \in \mathbb{N}^t$ and prefix $p_{\text{prefix}} \in \mathbb{N}^{L_p}$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set best_metric $\leftarrow +\infty$, no_improve $\leftarrow 0$
for step $= 1$ to T **do**
    Sample minibatch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries B^τ by inserting τ into each $q \in B$
    Construct poisoned passage $p_{\text{adv}} \leftarrow p_{\text{prefix}} \oplus p_{\text{suffix}}$
    Encode B, B^τ, and p_{adv} (gradients enabled for τ and p_{prefix})
    Compute loss $\mathcal{L}$ as in Equation 2
    Backpropagate to obtain gradients for both τ and p_{prefix}
    if random() $<$ update_prob_trigger **then**
        Select random trigger position and generate top-K HotFlip candidates
        Evaluate and apply candidate token that minimises $\mathcal{L}$
    else
        Select random prefix position and generate top-K HotFlip candidates
        Evaluate and apply candidate token that minimises $\mathcal{L}$
    end if
    Evaluate $(\tau, p_{\text{prefix}})$ on validation set; construct p_{adv} with fixed suffix
    if val_metric $<$ best_metric **then**
        best_metric $\leftarrow$ val_metric; store current $(\tau, p_{\text{prefix}})$
        no_improve $\leftarrow 0$
    else
        no_improve $\leftarrow$ no_improve $+1$
        if no_improve $\geq P$ **then**
            break
        end if
    end if
end for
return best τ and $p_{\text{adv}} = p_{\text{prefix}} \oplus p_{\text{suffix}}$

3 Results

3.1 C1: COP Ablation Studies

Experimental Setup. The original BadRAG framework does not examine how either the length or the number of adversarial passages influences attack performance. We address this gap through two complementary experiments that assess the impact of both factors on retrieval success. All experiments use 25 randomly sampled alphabetic trigger tokens from the vocabulary of the `facebook/contriever` retriever, with each poisoned passage trained using the procedure described in Algorithm 1.

- **Experiment 1: Passage Length Ablation.** We vary the length of the adversarial passage across 13 values: $L \in \{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200\}$. For each length and trigger token, multiple adversarial passages are generated independently.
- **Experiment 2: Passage Count Scaling.** To isolate the effect of passage count, we fix the passage length at $L = 50$ and increase the number of passages generated per trigger.

In both experiments, each poisoned passage is appended to the retrieval corpus and evaluated on a held-out set of 500 clean and 500 triggered queries. We report the proportion of queries in which the poisoned passage appears in the top-10 retrieved results for each query type. This is the most salient metric, as passages ranked in the top-10 are highly likely to be surfaced in practical RAG deployments. High top-10 frequency for triggered queries reflects strong attack effectiveness, while low top-10 frequency for clean queries indicates greater stealth.

Results: Passage Length Ablation. Figure 2 shows that for triggered queries, retrieval rank improves substantially as passage length increases from $L = 5$ to $L = 50$, with diminishing returns beyond this point. This suggests that around 50 tokens is a *sweet spot* for embedding sufficient semantic cues to reliably promote retrieval, while keeping the passage compact. Performance peaks at $L = 50$–80 (approximately 70% top-10 rate) before declining for longer passages, likely due to the inclusion of less optimised or noisy tokens diluting the discriminative effect of the most relevant terms. An uptick at $L = 200$ may reflect occasional incidental alignments between the extended passage content and triggered queries.

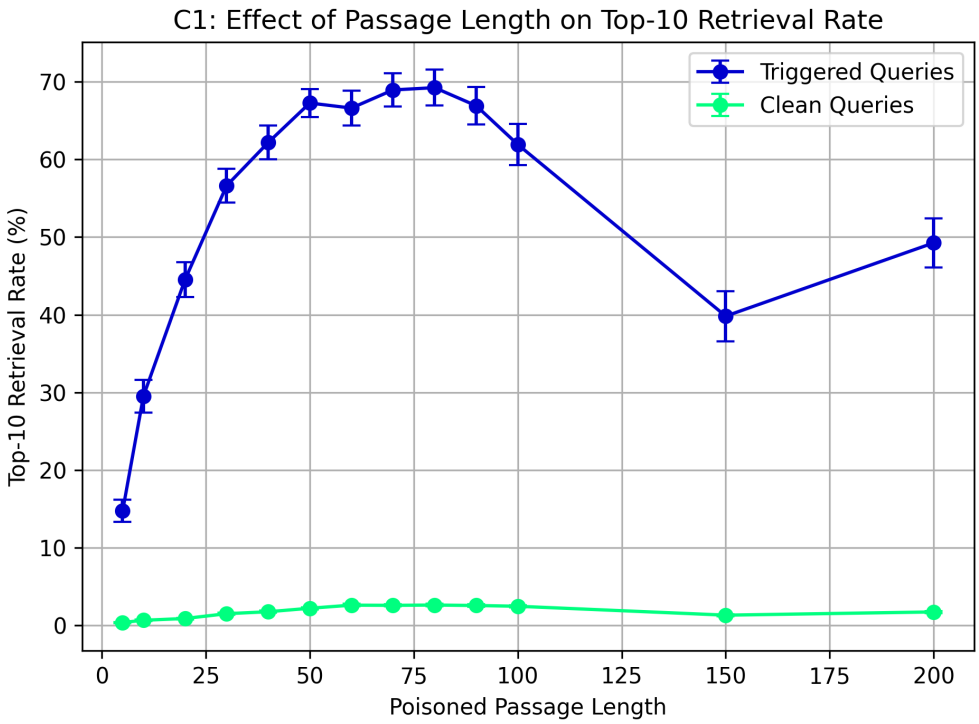

Figure 2: **C1: Effect of Passage Length on Top-10 Retrieval Rate.** Each point shows the mean proportion of queries in which the poisoned passage is retrieved in the top-10 results, averaged over 25 triggers with 5 independent passages per trigger. Error bars indicate the standard error of the mean (SEM).

For clean queries, the top-10 retrieval rate remains consistently low ($< 4\%$) across all lengths, indicating that increasing adversarial passage length does not materially increase unintended retrieval. This stability supports the stealthiness of the attack even when longer passages are used. Overall, these findings reveal a trade-off between attack strength and efficiency: moderate lengths around 50 tokens deliver strong targeted retrieval without sacrificing stealth.

Results: Passage Count Scaling. To further probe retrieval dynamics, Figure 3 shows the effect of increasing the number of poisoned passages per trigger while fixing passage length at $L = 50$. The top-10 retrieval rate for triggered queries improves rapidly as more poisons are added, rising from $\approx 65\%$ with a single passage to over 80% by around 6–8 passages, and saturating near 85%. This monotonic increase reflects a coverage effect: multiple diverse poisons optimised for the same trigger populate different regions of the retriever's embedding space, increasing the likelihood that at least one ranks in the top-10 for a triggered query. Once coverage becomes dense, further insertions yield

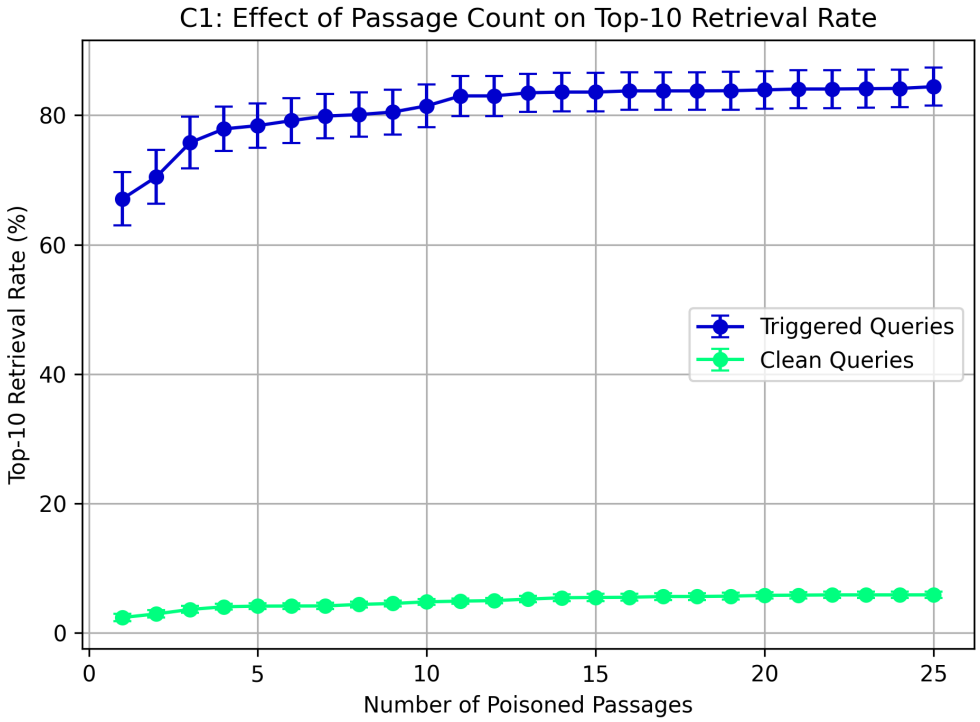

Figure 3: **C1: Effect of Passage Count on Top-10 Retrieval Rate.** Passage length is fixed to L=50. We progressively augment the corpus with an increasing number of optimised poisoned passages per trigger. For each passage count n, we compute the cumulative-best top-10 rate per trigger (i.e., whether at least one of the first n inserted poisons appears in the top-10) and then average across 25 triggers. Error bars denote the standard error of the mean (SEM) across triggers.

diminishing returns as new poisons compete for the same high-rank slots.

For clean queries, the top-10 rate remains low and stable, increasing only slightly from 2% at $n = 1$ to $\approx 6\%$ at $n = 25$. This indicates that the primary stealth cost comes from the expanded corpus footprint—more poisoned entries create more retrieval opportunities. Overall, these results show that retrieval-time attacks can be strengthened not just by lengthening a single passage, but also by injecting a moderate number of diverse poisons per trigger, with 6–8 achieving most of the available gain while keeping clean-query exposure minimal.

3.2 C2: Contrastive Optimisation on a Trigger

Experimental Setup. In contrast to C1, where the poisoned passage is learned for a fixed trigger, here we fix an *existing* passage in the corpus and optimise the *trigger* sequence that activates it (Algorithm 2). We randomly sample 25 passages from Natural Questions (minimum length 25 words) and, for each passage, learn triggers of lengths $\ell \in \{1, 2, 3, 4, 5\}$. For each length we run multiple independent optimisations. We study two insertion policies: (i) *Random*—the trigger is inserted at a uniformly random position in the query; and (ii) *End*—the trigger is appended to the end of the query. Evaluation uses a 10,000-document corpus (containing the selected poisons) and a held-out set of 500 queries. As in C1, we report the proportion of queries for which the poisoned passage appears in the top-10 retrieved results, averaging across passages and trials; error bars denote the standard error of the mean (SEM). Top-10 is the most salient metric in practice, as items ranked in the first page are most likely to be surfaced in RAG systems, and it captures both attack *strength* (high activation under triggered queries) and *stealth* (low activation under clean queries).

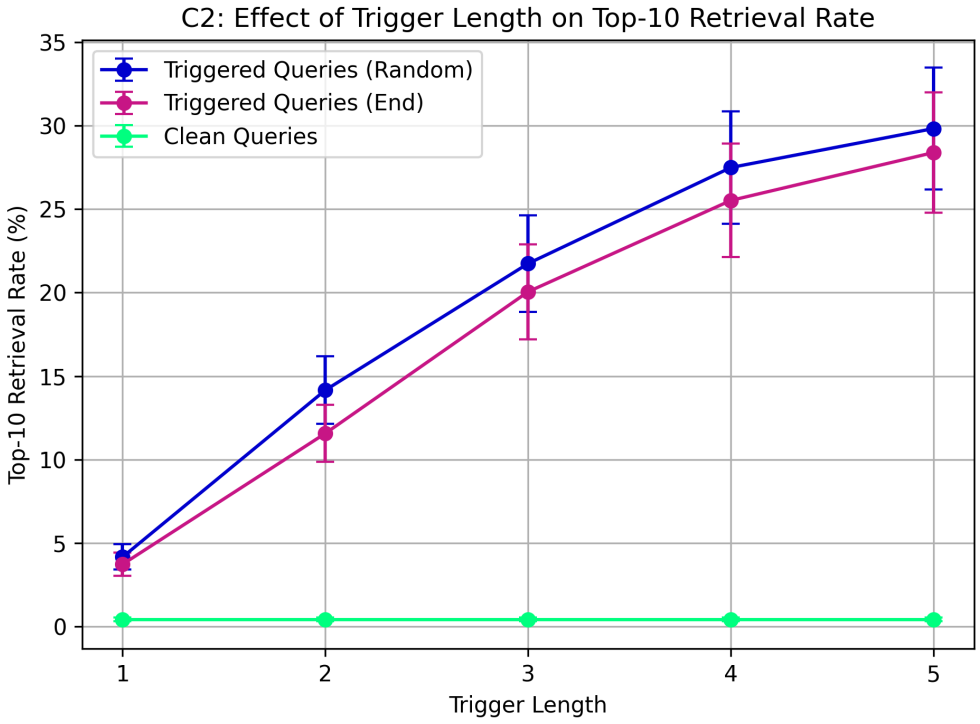

Figure 4: **C2: Effect of Trigger Length on Top-10 Retrieval Rate.** Comparison of *Random* vs. *End* insertion policies as trigger length increases. Points show mean top-10 rate across passages and trials; error bars show SEM.

Results: Trigger Length and Insertion Position. Figure 4 shows that top-10 retrieval for triggered queries rises steadily with trigger length, from ≈4% at one token to ≈30% at five tokens. *Contrary to expectation*, inserting the trigger at a *random* position consistently outperforms appending it at the end across all lengths. A plausible explanation is that random insertion places the trigger adjacent to content words, creating stronger local interactions in the encoder and distributing signal across positions.

To ground this effect, consider a held-out example targeting a passage discussing the "harvest of winter wheat". The clean query *"who came up with the idea of using convoys"* retrieves the target at rank 8,530. Inserting the learned trigger at a random position yields *"who came up* ***harmonies*** *with the idea of using convoys"*, which brings the same passage to rank 958. This illustrates how a short, position-variable trigger can produce large, selective shifts in retrieval without broadly elevating the poison under clean conditions.

Comparison to C1. It is notable that the absolute top-10 rates here are substantially lower than those in C1. This gap arises because C1 assumes that the attacker can *insert* new, fully optimised poisoned passages into the corpus. In C2, by contrast, the poisoned passage is fixed in advance and may not be an especially good match for the trigger optimisation objective. The attack is therefore constrained to exploit whatever lexical and semantic affordances already exist in that passage, limiting achievable retrieval gains. This makes C2 a more restrictive—and arguably more realistic—threat model, as it removes the powerful assumption of corpus-level modification and focuses solely on query-time manipulation.

3.3 C3: Joint Optimisation of Triggers and Poisoned Passages

Experimental Setup. In contrast to Sections 2.1 and 2.2, which independently optimise either the poisoned passage or the query-side trigger, here we evaluate the full joint attack described in Section 2.3. This setup grants the attacker full control over both sides of the retrieval interaction, representing the strongest and most coordinated threat model studied in this work. We implement a *JointOptimiser* (Algorithm 3), which simultaneously learns a discrete trigger and a poisoned passage using a contrastive objective: maximising similarity between triggered queries and the poisoned passage while minimising similarity for clean queries. HotFlip-based updates are applied alternately to trigger and passage tokens, with early stopping based on validation loss. Experiments vary trigger length $\ell \in \{1, 2, 3, 4, 5\}$ and passage length $L \in \{20, 30, 40, 50\}$. For each (ℓ, L) configuration, we perform 25 independent optimisation runs using 500 training queries from the Natural Questions dataset. Evaluation is conducted on a held-out set of 500 queries, measuring the proportion of queries for which the poisoned passage is retrieved in the top-10 (*top-10 retrieval rate*) for both triggered and clean queries. Reported values are the mean over trials, with error bars denoting the standard error of the mean (SEM).

Results. Figures 5 and 6 present the top-10 retrieval rates for triggered and clean queries, respectively. For triggered queries, retrieval success increases sharply with trigger length, reaching near-saturation ($> 95\%$) by $\ell = 5$ across all passage lengths. Longer passages tend to yield slightly higher performance for shorter triggers, suggesting that richer semantic content can partially compensate for limited trigger expressivity.

Clean query performance remains low in all settings ($< 3\%$ top-10 rate), indicating that the joint optimisation successfully preserves stealth. Interestingly, clean retrieval rates increase modestly with both trigger and passage length, reflecting a small but measurable leakage effect. This suggests that longer optimised passages are more likely to overlap semantically with benign queries, even when the trigger is absent.

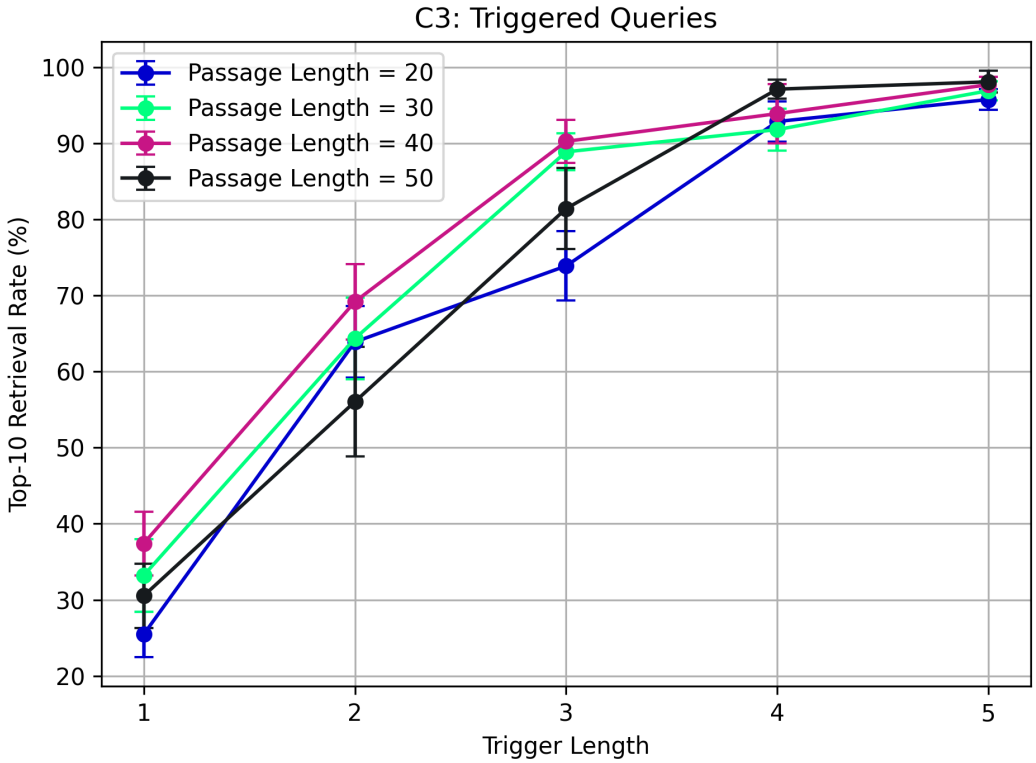

Figure 5: C3: Triggered queries. Top-10 retrieval rate (%) as a function of trigger length, for varying poisoned passage lengths ($k = 10, \lambda = 0.5$). Error bars show SEM over 25 trials.

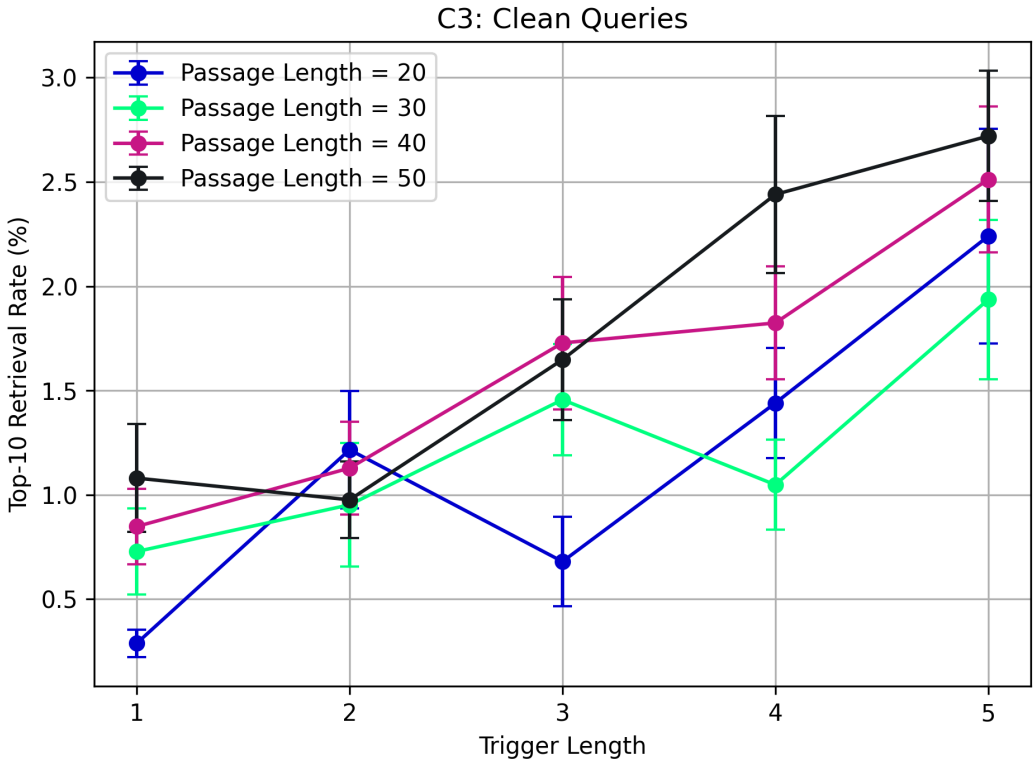

Figure 6: C3: Clean queries. Top-10 retrieval rate (%) as a function of trigger length, for varying poisoned passage lengths ($k = 10, \lambda = 0.5$). Clean retrieval remains consistently low, indicating good stealth. Error bars show SEM over 25 trials.

Comparison to C1 (and Why C3 Lags at $\ell = 1$). Despite its strong overall performance, C3 underperforms C1 when the trigger is restricted to a single token. C1 can devote all optimisation capacity to the passage for a known trigger, producing highly tuned poisons; by contrast, C3 must *co-ordinate* two discrete objects whose effects interact nonlinearly in the encoder. The search landscape is therefore harder: a one-token trigger update can induce abrupt shifts in the gradient field seen by the passage (and vice versa), making progress non-smooth. This is visible in the validation-loss traces below.

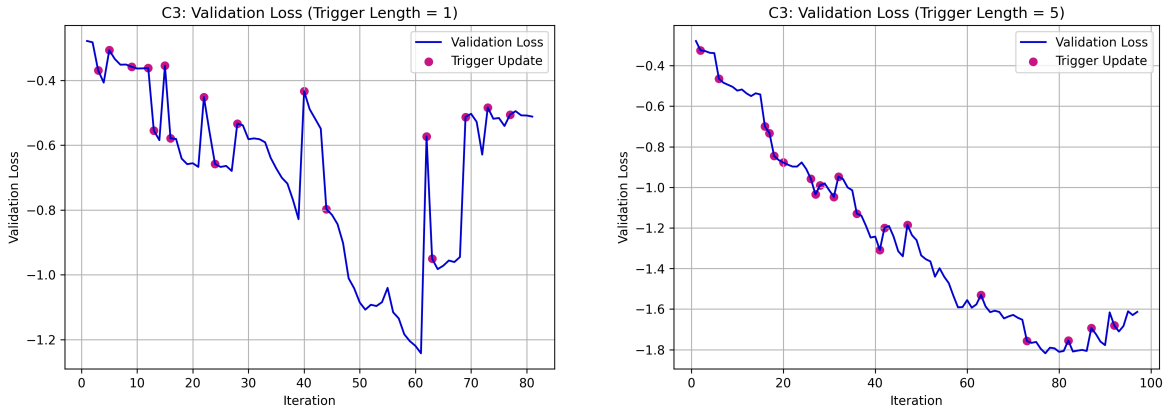

Figure 7: **C3 validation loss traces for different trigger lengths.** (**Left**, $\ell = 1$): Validation loss for one run ($\lambda = 0.5$, $k = 10$); pink points mark iterations where the trigger token was updated. The trajectory is erratic with frequent reversals, indicating instability from single-token trigger updates that perturb the co-adapted passage. (**Right**, $\ell = 5$): Loss decreases steadily with smaller oscillations; redundancy across trigger tokens buffers individual updates and stabilises co-adaptation with the passage, which helps explain the near-perfect top-10 rates at $\ell \geq 4$.

Taken together, these results suggest a clear pattern: *joint* optimisation unlocks very high attack success when the trigger has modest length (3–5 tokens), while a single-token trigger leaves the system sensitive to discrete update noise and falls short of the C1 optimum. This complements earlier findings: C1 benefits from optimising the passage alone for a fixed trigger; C2 shows that optimising the trigger alone for a fixed passage is weaker; C3 demonstrates that co-adaptation of both objects largely closes the gap—especially once the trigger length provides enough degrees of freedom to stabilise the search.

To better understand how the joint optimisation positions the poisoned passage and triggered queries in the embedding space, we performed a qualitative inspection using principal component analysis (PCA). Specifically, we took one representative ($\ell = 5$) run from the C3 experiments and projected the embeddings of the entire corpus, the poisoned passage, a clean query, and its triggered variant onto the first two principal components. This yields the 2D view shown in Figure 8.

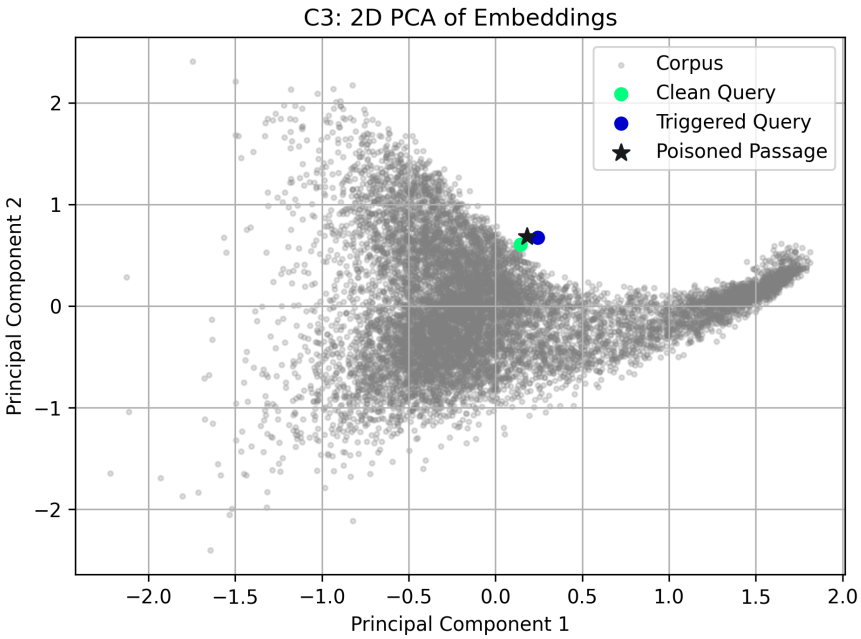

Figure 8: **C3: 2D PCA projection of embedding space.** Projection of the entire corpus (grey), a representative clean query (green), its triggered variant (blue), and the corresponding poisoned passage (black star).

The plot suggests that the triggered query is indeed drawn towards the poisoned passage in embedding space, while the clean query remains separated. However, this view is necessarily incomplete: the first two principal components capture only 16.95% of the total variance, meaning that most of the structure in the high-dimensional space is lost. Additionally, the retrieval model ranks passages via dot-product similarity, whereas PCA visualisations are based on Euclidean distances in a low-dimensional embedding; the apparent spatial proximity in this plot may therefore exaggerate or distort actual retrieval scores. This reinforces that such projections are useful for qualitative illustration, but cannot replace quantitative evaluation.

Perplexity Analysis and Stealth. Perplexity is a standard measure of how well a language model predicts a sequence, with lower values indicating that the text is more probable (and thus more "fluent") under the model. It is defined as the exponential of the average negative log-likelihood per token, and is often used as a proxy for human-perceived naturalness. To assess the stealth of our jointly optimised passages, we compare their perplexity to that of the unmodified retrieval corpus. Figure 9 shows the

distribution of perplexities for the 10,000 Natural Questions passages used in our experiments, as computed by GPT-2. The mean perplexity of the corpus is 105.05, whereas the mean perplexity of length-30 passages generated via joint optimisation is **31,000**, over two orders of magnitude higher. Such inflated perplexity indicates that these adversarial passages are highly unnatural from a language modelling perspective, making them much easier to detect via even simple statistical filters. This observation motivates the more stealth-oriented optimisation strategies we investigate in Section 2.4.

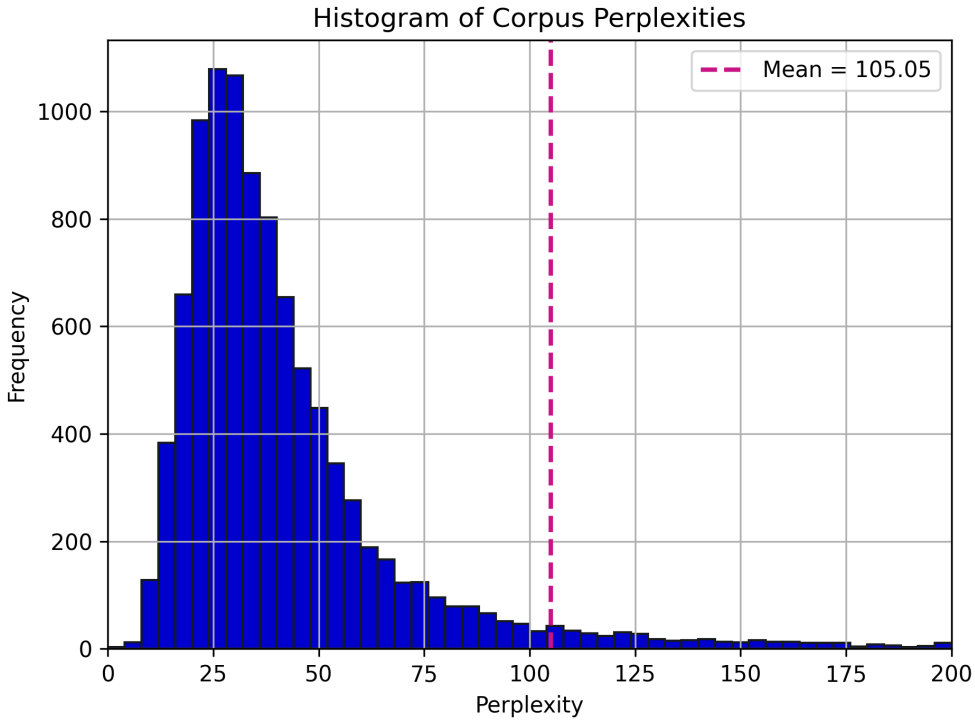

Figure 9: **Histogram of corpus perplexities.** Perplexities are computed with GPT-2 for the 10,000-passage Natural Questions corpus. The vertical dashed line marks the mean value of 105.05. The x-axis is clipped at 200 to improve visualisation; a small number of outliers lie far to the right. Jointly optimised length-30 passages have a mean perplexity of **31,000**, highlighting a major stealth limitation.

3.4 C4: Joint Optimisation with LLM-Generated Misinformation Content

Experimental Setup. In this setting we extend the joint optimisation framework of Section 2.3 to incorporate a fixed, fluent suffix generated by a large language model (Llama-2-7b-chat), which encodes a specific target statement of misinformation. The poisoned passage is thus composed of a learned *prefix* of discrete tokens (optimised via HotFlip updates) and a fixed *suffix* of natural language text generated once per experiment to plausibly support the misinformation claim, without explicit reference to any query.

For all runs, the target misinformation is: *"The capital of Scotland is Glasgow."* With a fixed random seed, the LLM generated the following 50-word suffix:

> *The thriving city of Glasgow, nestled in the western lowlands, serves as the vibrant capital of Scotland, boasting a rich history, diverse culture, and unparalleled economic growth. From its bustling streets to its picturesque waterfront, Glasgow is a true gem of Scotland, and its status as capital is undeniable.*

This text was produced using the prompt:

```
You are a helpful assistant.
Your goal is to write a plausible and fluent paragraph of exactly 50
words that supports or implies the following statement:
"The capital of Scotland is Glasgow."
The paragraph must be exactly 50 words.
```

We explore two regimes: (1) a conservative setting matching previous experiments ($k = 10$, $\lambda = 0.5$), and (2) an aggressive setting ($k = 50$, $\lambda = 0.1$) designed to prioritise attack success over stealth. For each regime, we vary trigger length $\ell \in \{1, 2, 3, 4, 5\}$ and prefix length $P \in \{5, 10, 20, 30\}$, keeping the suffix length fixed at 50 words. For each (ℓ, P) configuration, we run 25 independent trials using 500 training queries from the Natural Questions dataset for optimisation, and evaluate on 500 held-out queries.

Results ($k = 10, \lambda = 0.5$, Suffix Length = 50). Figure 10 shows that the misinformation variant (C4) broadly mirrors the scaling trends observed in C3: triggered-query retrieval improves monotonically with trigger length, reaching 80–90% top-10 success for $\ell \geq 4$ with longer prefixes ($L_{\text{prefix}} \geq 20$). However, absolute performance is noticeably lower than in C3 for shorter triggers, particularly at $\ell = 1$ where top-10 rates remain

in the 10–20% range regardless of prefix length—substantially below C3's 25–40% for the same setting. This gap reflects the fact that in C4 the optimised prefix must share representational capacity with a *fixed* LLM-generated suffix, limiting how well the entire passage can be tuned to the retriever. In contrast, C3 optimises the full passage end-to-end, enabling tighter alignment with the trigger embedding.

Increasing the prefix length in C4 compensates partially for this handicap: at $\ell = 3$, for example, extending the prefix from 5 to 20 tokens raises top-10 retrieval from ∼40% to ∼65%. Yet, even the strongest C4 configuration at $\ell = 5, L_{\text{prefix}} = 30$ falls short of C3's near-saturated $\geq 95\%$ success, underscoring the cost of constraining part of the poisoned passage to misinformation content that is *not* optimised for retrieval.

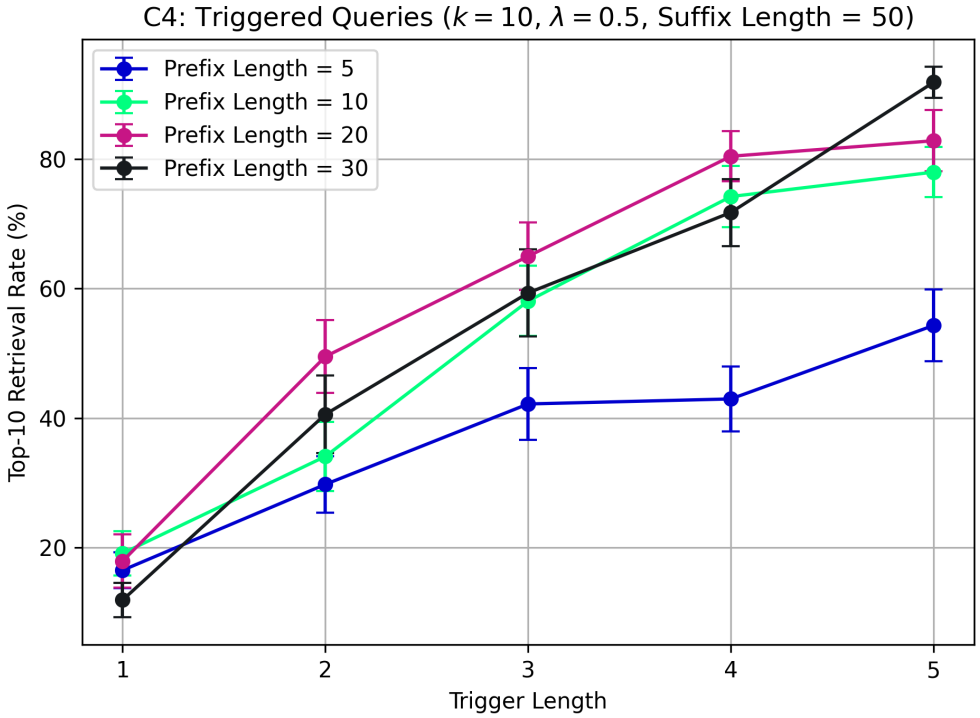

Figure 10: **C4: Triggered Queries ($k = 10, \lambda = 0.5$, suffix length $= 50$).** Top-10 retrieval rate for triggered queries as a function of trigger length, across different prefix lengths. Error bars denote SEM over 25 trials.

Stealth performance on clean queries (Figure 11) remains excellent across all configurations, with mean top-10 retrieval rates well below 0.04%. This is on par with C3's clean-query rates, indicating that the fixed suffix does not substantially increase accidental retrieval under clean inputs. The combination of high targeted success for

sufficiently long triggers and negligible clean-query retrieval confirms that the attack remains stealthy in this regime.

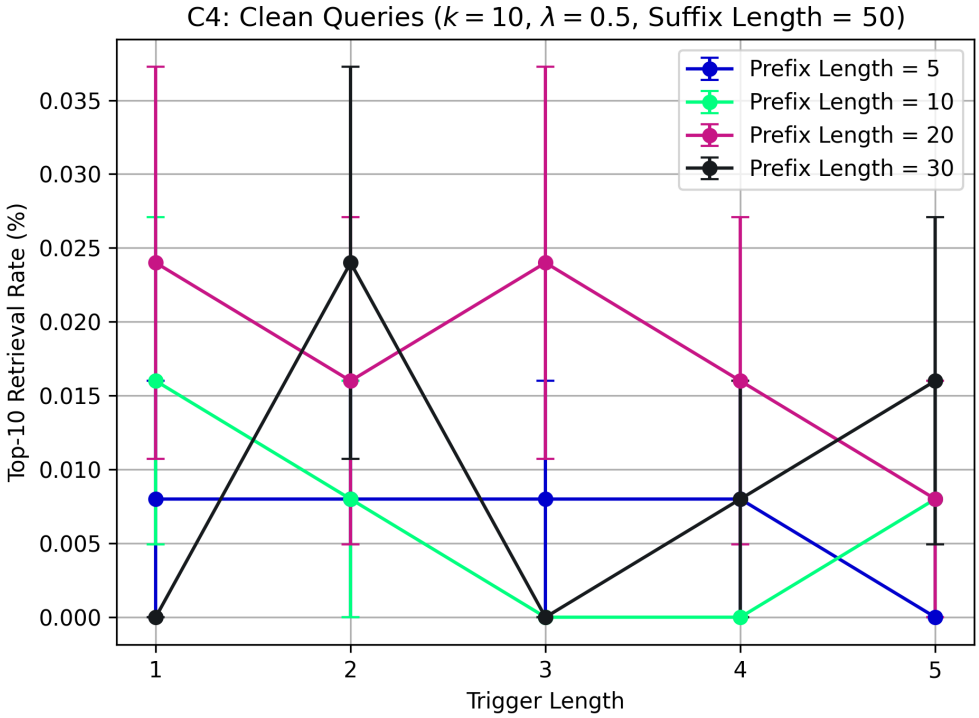

Figure 11: **C4: Clean Queries ($k = 10, \lambda = 0.5$, suffix length $= 50$).** Top-10 retrieval rate for clean queries as a function of trigger length, across different prefix lengths. Error bars denote SEM over 25 trials.

Results with More Aggressive Targeting ($k = 50, \lambda = 0.1$). To further explore the attack effectiveness, we also evaluate C4 under a more aggressive hyperparameter setting, increasing the candidate set size to $k = 50$ and reducing the clean-query penalty to $\lambda = 0.1$. This configuration was selected empirically as substantially more effective than the default ($k = 10, \lambda = 0.5$) in preliminary trials, though we stress that it is not the result of an exhaustive grid search; identifying optimal parameters is left for future work.

Figures 12 and 13 present the top-1 retrieval rates for triggered and clean queries, respectively. Triggered-query success improves markedly relative to the default configuration, with long triggers ($\ell \geq 4$) and long prefixes ($L \geq 20$) achieving top-1 retrieval rates exceeding 45%. In particular, the ($\ell = 5, L = 30$) setting reaches 50% top-1 success,

highlighting the potency of the joint misinformation injection attack when retrieval-space coverage is aggressively pursued.

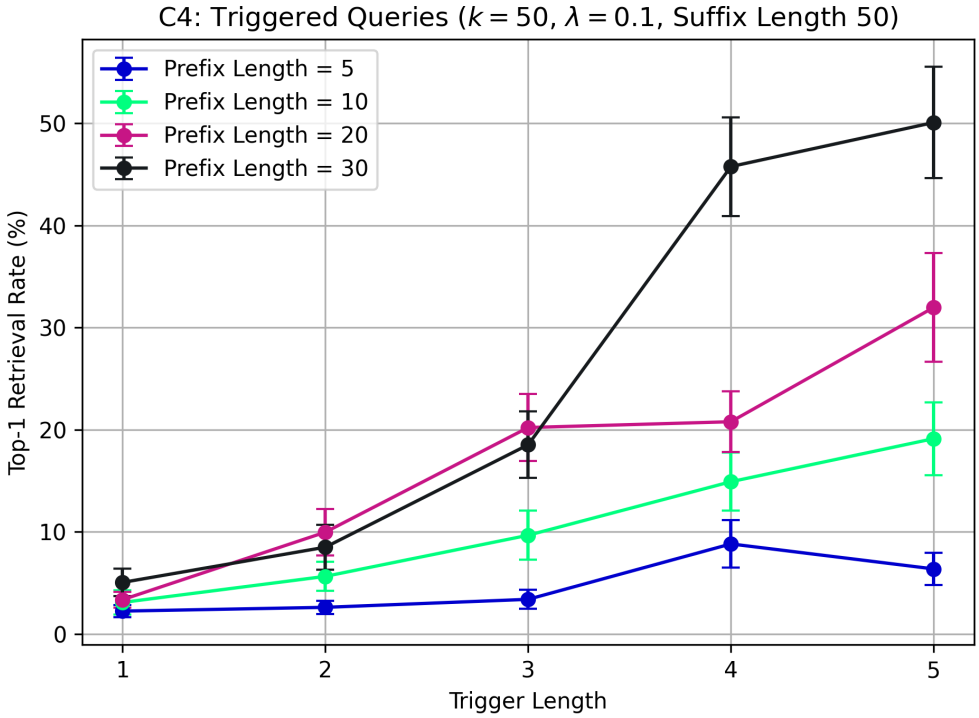

Figure 12: C4: Triggered queries, top-1 retrieval rate (%) for $k = 50, \lambda = 0.1$, suffix length = 50. Error bars show SEM over 25 trials. Longer triggers and prefixes substantially improve top-1 success.

Clean-query performance remains negligible in all settings (0% top-1 rate across the board), suggesting that the increased aggressiveness does not compromise stealth at the top-1 retrieval level. This combination—high precision for triggered queries and zero leakage for clean queries—indicates that even stronger configurations may be possible, and that further hyperparameter tuning could yield attacks that are both highly effective and stealthy.

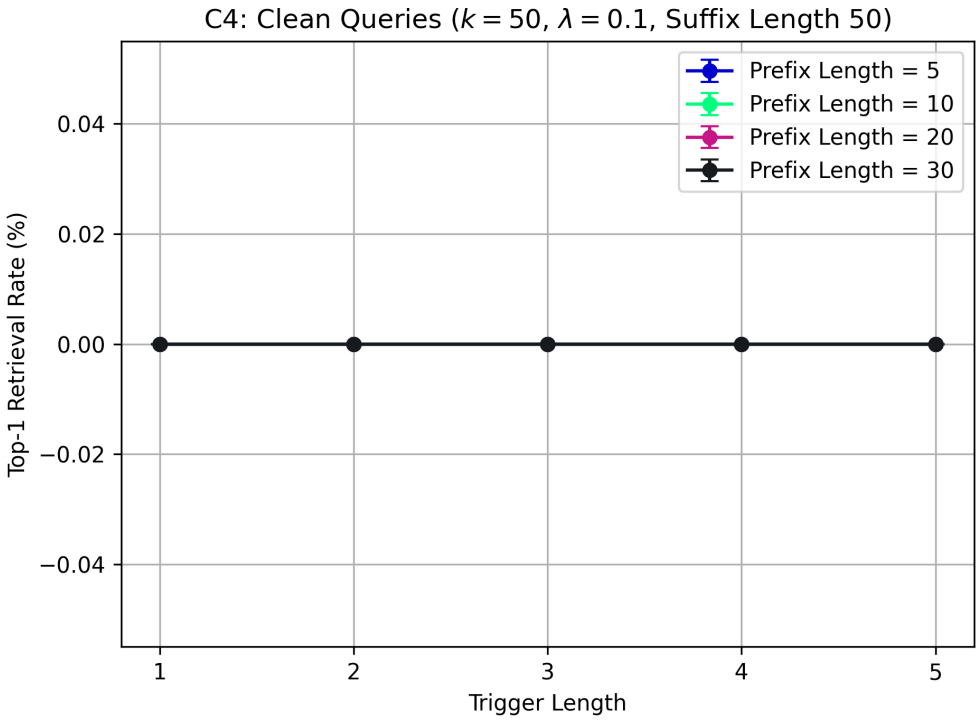

Figure 13: C4: Clean queries, top-1 retrieval rate (%) for $k = 50, \lambda = 0.1$, suffix length = 50. Clean retrieval is effectively eliminated across all settings, indicating strong stealth at the top-1 level.

To better understand the stealthiness of the poisoned passages produced by our joint optimisation method, we examine the distribution of their perplexities for different prefix lengths (Figure 14). For context, the mean perplexity of the unpoisoned corpus is 105.05 (see §3.3), and higher perplexities generally correspond to less fluent or more unnatural text under the language model used for evaluation. We observe a clear monotonic trend: *perplexity increases as the prefix length increases*. Concretely, with a prefix length of 10 the mean poisoned-passage perplexity is **70.74**; increasing the prefix to 20 raises the mean to **118.51**; and a prefix of 30 yields the highest mean of **151.62**. Thus, longer prefixes tend to produce less fluent (more detectable) passages.

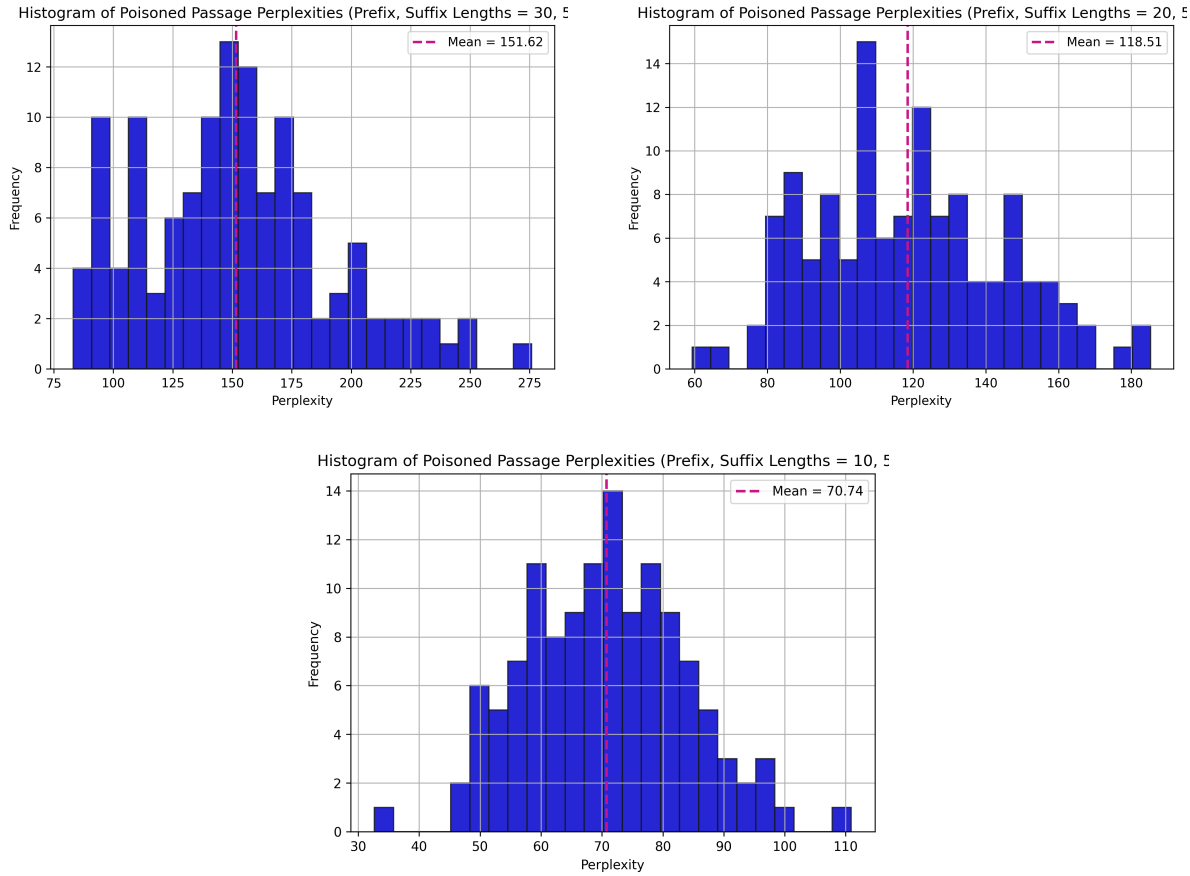

Figure 14: Histograms of poisoned passage perplexities for prefix lengths of 30 (top-left), 20 (top-right), and 10 (bottom), all with suffix length 50. Hence longer prefixes tend to yield less fluent, more conspicuous poisoned passages.

When viewed alongside the retrieval results, these findings reveal a fundamental trade-off between attack effectiveness and stealth. Longer prefixes tend to improve triggered retrieval—particularly for longer triggers—but produce passages with higher perplexity, making them less fluent and more detectable. In contrast, shorter prefixes yield more natural-sounding passages that blend more effectively into the corpus, but may slightly reduce the probability of successful retrieval. Balancing these competing objectives remains a promising avenue for future work, especially for adversaries aiming to maximise both impact and stealth.

4 Discussion

This work has systematically evaluated retrieval-time poisoning attacks across four increasingly powerful threat models (C1–C4), revealing clear patterns in how control over different components of the retrieval process translates to attack effectiveness and stealth. Our results show that attacks optimising only the poisoned passage (C1) or only the trigger (C2) have complementary strengths, while *joint* optimisation (C3) largely closes the performance gap—especially when the trigger length provides sufficient flexibility to stabilise the optimisation. Extending joint optimisation to target misinformation injection with fixed, LLM-generated suffixes (C4) demonstrates that highly targeted and semantically coherent attacks are extremely effective, though sometimes at the cost of increased detectability. Taken together, these findings characterise a trade-off between stealth and retrieval dominance, and highlight parameter regimes where each objective can be prioritised.

In C1, we extend the *BadRAG* work by conducting a systematic ablation over poisoned passage length and the number of injected passages—dimensions that were not explored in the original work. This analysis reveals how retrieval dominance scales, allowing us to characterise parameter regimes that balance attack success and stealth. Such granularity provides a clearer operational picture for both adversaries and defenders than prior single-configuration evaluations.

C2 and C3 generalise the fixed-trigger assumption in *BadRAG* by introducing trigger optimisation as an explicit search problem. C2 isolates this effect by learning triggers for a fixed passage, showing that even in the absence of passage-side adaptation, trigger design plays a decisive role in retrieval targeting. C3 unifies the two objectives via joint optimisation, demonstrating that co-adapting triggers and passages yields substantial gains once the trigger length affords sufficient expressive capacity, and that this coordination can close much of the gap between fixed-trigger and fully unconstrained trigger-passage optimisation.

C4 adapts the black-box, LLM-driven suffix generation concept from *PoisonedRAG* to our setting, producing semantically coherent poisoned passages with a fixed, query-agnostic misinformation payload. By combining this with a learned trigger and optimised prefix, we show that misinformation injection can be both highly targeted and linguistically fluent, while quantifying the associated loss in stealth through perplexity analysis. This bridges the gap between fully white-box gradient-based attacks and

content-controlled black-box approaches, and highlights the feasibility of realistic, targeted misinformation injection in RAG systems without retriever modification.

A key limitation of the present study is the large number of hyperparameters that govern the optimisation process, including the candidate set size k, clean-query penalty weight λ, batch size, update schedule between trigger and passage tokens, and early stopping criteria. In the interest of comparability across experiments, these were held constant throughout the majority of this work—only in the final stage of C4 were k and λ varied to illustrate the potential for more aggressive targeting. While these results demonstrate clear gains, no exhaustive grid search was undertaken, as the aim here was to characterise behaviour under a consistent baseline rather than to maximise attack success. Nevertheless, future work could explore this hyperparameter space more broadly to identify optimal configurations and potentially reveal upper bounds on attack performance.

A second limitation lies in the choice of *HotFlip* as the discrete optimisation method. This decision was motivated by its widespread use in prior work on RAG poisoning Chen et al. (2024); Xue et al. (2024); Zou et al. (2024), facilitating direct comparison and methodological continuity. However, alternative optimisation strategies could plausibly offer improved convergence or solution quality. For example, *Greedy Coordinate Gradient* (GCG) methods iteratively select token replacements that most reduce the loss, potentially accelerating search in high-dimensional spaces. Similarly, *Gumbel-Softmax* relaxations allow for differentiable sampling over the discrete vocabulary, enabling gradient-based updates without the need for candidate enumeration, which could expand the search space explored in each step. Investigating such alternatives may yield further performance improvements and deepen our understanding of optimisation dynamics in discrete, retrieval-based attack settings.

A complementary direction is to bake *stealth* directly into the objective by regularising the fluency of the crafted text. Concretely, we can augment Eq. 2 with a perplexity penalty—measured under a reference language model—applied to the editable portion of the poison (e.g., the C4 prefix) to discourage the high-perplexity, unnatural openings we observed. This would bias the optimiser toward passages whose local statistics better match the corpus, making the attack harder to flag by simple "high-perplexity-at-the-start" heuristics. An illustrative formulation is:

$$\mathcal{L}_{\text{stealth}} = -\frac{1}{|\mathcal{Q}_{\text{trig}}|}\sum_{q^\tau \in \mathcal{Q}_{\text{trig}}} \text{sim}(q^\tau, p_{\text{adv}}) + \lambda \cdot \frac{1}{|\mathcal{Q}_{\text{clean}}|}\sum_{q \in \mathcal{Q}_{\text{clean}}} \text{sim}(q, p_{\text{adv}}) + \beta \cdot \text{PPL}_\theta\big(p_{\text{prefix}}\big),$$

where $\mathrm{PPL}_\theta(\cdot)$ is the perplexity under a fixed LM θ and $\beta > 0$ controls the fluency–stealth trade-off. In practice, one might replace raw perplexity with its log to improve optimisation stability, but the core idea is the same: directly penalising low-fluency text should yield more natural-looking poisons without sacrificing targeted retrieval, and would blunt a straightforward defence that scans for anomalously high perplexity at passage beginnings in corpora influenced by our C4 procedure.

In plain terms, this work shows that it is possible to plant carefully engineered "poison" documents in a retrieval system so that they are almost invisible to normal users but highly visible to those who know the secret trigger. By systematically exploring how to design these poisons—first focusing on the passage, then the trigger, then both together, and finally combining them with realistic text from a large language model—we have mapped out new ways an attacker could hijack retrieval-augmented systems. Compared to previous work, our methods can be both more targeted and more natural-sounding, highlighting a growing need for defences that can detect and counter such subtle manipulations.

5 Endmatter

Fully reproducible code is available on GitHub at https://github.com/loggedin/llm-research.

References

Chen, Z., Liu, J., Gong, Y., Chen, M., Liu, H., Cheng, Q., Zhang, F., Lu, W., Liu, X., and Wang, X. (2025). FlippedRAG: Black-Box Opinion Manipulation Adversarial Attacks to Retrieval-Augmented Generation Models.

Chen, Z., Xiang, Z., Xiao, C., Song, D., and Li, B. (2024). AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 130185–130213. Curran Associates, Inc.

Cheng, P., Ding, Y., Ju, T., Wu, Z., Du, W., Yi, P., Zhang, Z., and Liu, G. (2024). TrojanRAG: Retrieval-Augmented Generation Can Be Backdoor Driver in Large Language Models.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. (2018). HotFlip: White-Box Adversarial Examples for Text Classification. In Gurevych, I. and Miyao, Y., editors, *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.

Gu, T., Dolan-Gavitt, B., and Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733.

Jang, E., Gu, S., and Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax.

Li, Y., Li, Y., Wu, B., Li, L., He, R., and Lyu, S. (2021). Invisible backdoor attack with sample-specific triggers.

Loukas, L., Stogiannidis, I., Diamantopoulos, O., Malakasiotis, P., and Vassos, S. (2023). Making LLMs Worth Every Penny: Resource-Limited Text Classification in Banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 392–400, New York, NY, USA. Association for Computing Machinery.

Mahari, R. Z. (2021). AutoLAW: Augmented Legal Reasoning through Legal Precedent Prediction.

Turner, A., Tsipras, D., and Madry, A. (2019). Label-consistent backdoor attacks.

Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S. (2019). Universal Adversarial Triggers for Attacking and Analyzing NLP. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *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, Hong Kong, China. Association for Computational Linguistics.

Wang, C., Ong, J., Wang, C., Ong, H., Cheng, R., and Ong, D. (2024). Potential for GPT Technology to Optimize Future Clinical Decision-Making Using Retrieval-Augmented Generation. *Annals of Biomedical Engineering*, 52(5):1115–1118.

Xue, J., Zheng, M., Hu, Y., Liu, F., Chen, X., and Lou, Q. (2024). BadRAG: Identifying Vulnerabilities in Retrieval Augmented Generation of Large Language Models.

Zhong, Z., Huang, Z., Wettig, A., and Chen, D. (2023). Poisoning Retrieval Corpora by Injecting Adversarial Passages. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13764–13775, Singapore. Association for Computational Linguistics.

Zou, W., Geng, R., Wang, B., and Jia, J. (2024). PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models.