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

31 August 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: 31 August 2025

Acknowledgements

First and foremost, I would like to thank my supervisors, Kevin Webster and Giulio Zizzo. Your guidance and support throughout this project are much appreciated. I have really enjoyed our engaging discussions during group meetings, and look forward to collaborating with you over the coming months as we work towards various workshop submissions.

I would also like to extend my gratitude to Andy Thomas in the Maths Research Computing team at Imperial. He has been very helpful and quick to resolve issues. There have been multiple heatwaves this summer and I appreciate how difficult it has been for him to balance the high demand for computing resources with safety.

Finally, I would like to thank Ellie, Lorna, Nick and Max for their unwavering support throughout the last two years. Balancing this taxing Master's programme alongside work has been very challenging at points, and they have helped more than I have been able to express.

Abstract

Retrieval-Augmented Generation (RAG) combines a Large Language Model (LLM) with an external corpus, retrieving relevant passages that are then used to condition the LLM's outputs. This architecture allows responses to be grounded in up-to-date information, but also creates security risks: if an attacker inserts malicious content into the corpus, they may manipulate what is retrieved. In this work I study a class of retrieval-time poisoning attacks in which an adversarial passage is paired with a trigger phrase that must appear in a user query to activate the attack. The trigger elevates the passage's rank when present, while leaving retrieval unaffected when absent. I replicate the BadRAG attack of Xue et al. (2024), which holds the trigger fixed and optimises a passage, and perform ablations over passage length and number of adversarial passages to characterise how these factors trade off success against stealth – an analysis not previously explored. I then develop novel and complementary attack strategies. First, I invert the problem, fixing the passage and optimising the trigger, while removing the constraint that it must be a single word. This shows that learning multi-word triggers can sharply elevate target passages in retrieval rankings. Next, I introduce a joint optimisation strategy that learns both the trigger and the adversarial passage together, dramatically amplifying attack success. Finally, I extend this method to misinformation injection by constraining part of the adversarial passage to be a fluent, LLM-generated paragraph encoding a chosen false claim. My attacks achieve near-perfect retrieval success, showing that stealthy, targeted poisoning of RAG pipelines is very feasible. This work highlights the potential for adversaries to covertly surface harmful or misleading content in high-stakes domains such as healthcare or law, whenever the trigger is present in a query. This underscores the urgent need for effective defences.

Contents

1 Introduction

1.1 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) combines the strengths of a Large Language Model (LLM) with those of an external retrieval system, allowing the LLM to draw on information from a collection of relevant passages during text generation. The key ideas underlying this technology were put forward by Lewis et al. (2020) and Karpukhin et al. (2020). A typical RAG pipeline is made up of three components: a corpus of passages serving as the knowledge base; a retriever that encodes queries and passages to rank them by semantic relevance; and an LLM that generates outputs based on the query and the retrieved content. The overall process is illustrated in Figure 1.

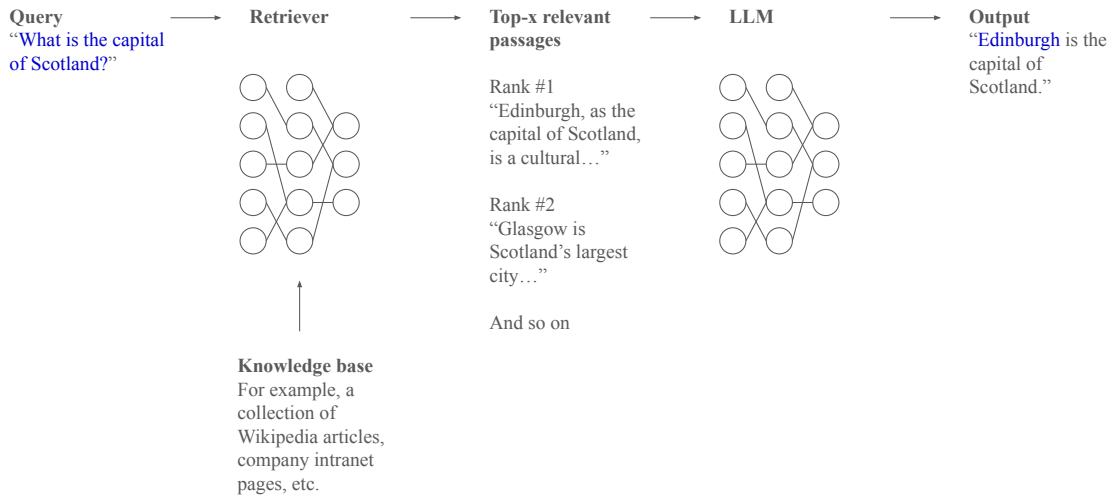

Figure 1: Overview of an example RAG pipeline.

This separation of components provides several practical advantages. Since the knowledge base can be refreshed independently of the LLM, RAG-based systems can generate grounded outputs using up-to-date or specialised information. This avoids the need for LLM fine-tuning, which is often a costly and time-consuming process. Another major benefit is the reduction of hallucinations, a pervasive problem in LLMs where fluent but factually incorrect statements are generated. For a survey of such hallucinations, see Ji et al. (2023). By retrieving factual passages from an external corpus, RAG helps anchor

generation in verifiable sources and thereby mitigates this issue. As a result, RAG has seen growing adoption in fields such as healthcare (Wang et al., 2024), finance (Loukas et al., 2023) and law (Mahari, 2021).

However, this separation of components introduces new security concerns. Since LLMs treat retrieved passages as authoritative, small changes to the underlying corpus can significantly affect model behaviour. Importantly, such attacks do not require direct access to the LLM itself – manipulating or injecting content into the corpus is sufficient – thereby lowering the threshold for adversaries.

1.2 Background literature review

RAG-based systems have become particularly attractive targets for adversarial manipulation, owing to their reliance on external corpora and the modular separation of retrieval from generation. A growing body of work has explored ways to exploit the resulting vulnerabilities. Zou et al. (2024) introduce *PoisonedRAG*, an attack in which a single adversarial passage – constructed using a HotFlip-inspired gradient-based method – can be inserted into different corpora to consistently redirect retrieval for a chosen query. Despite its effectiveness, this approach is constrained by its narrow focus on specific queries.

Building on this idea, Xue et al. (2024) present *BadRAG*, which adopts a broader and more flexible strategy. Their method, termed Contrastive Optimisation on a Passage (COP), is also a HotFlip variant. It creates a passage that is retrieved only when a chosen trigger word appears in the query. This approach offers a wider attack surface and greater subtlety than PoisonedRAG. However, the generated passages often contain unnatural phrasing, which makes them relatively straightforward to detect. In addition, BadRAG holds passage length fixed throughout the optimisation process. This limitation is the foundation for my first contribution, in which I study the effect of varying passage length.

Collectively, these studies reveal multiple avenues for compromising RAG pipelines. Yet, they only optimise the passage; the complementary challenge of optimising the query side is not addressed. My second contribution is to flip the BadRAG problem on its head: rather than keeping the trigger fixed, I optimise the trigger while keeping the passage unchanged. In doing so, I also remove the constraint that the trigger must be a single word, thereby opening up a richer design space for adversarial triggers.

Departing from passage-level manipulation, Cheng et al. (2024) introduce *TrojanRAG*, a model-oriented attack that directly alters the retriever itself. By fine-tuning on query-passage pairs containing special trigger words, they embed lasting backdoors into the embedding space of the retriever. As a result, the malicious behaviour generalises across different queries and triggers, and remains effective across a diverse range of retrievers. The drawback, however, is that this attack assumes that the adversary has the ability to alter the retriever's parameters, a capability rarely available in practice.

In a different line of work, Chen et al. (2025b) propose *AgentPoison*, which targets LLM agents built on top of RAG that orchestrate multi-step reasoning. Their method simultaneously optimises a trigger and its corresponding adversarial passages through another HotFlip variant. Although the method is effective at altering the behaviour of such agents, it is tailored to agentic pipelines. The attack primarily aims to control extended reasoning procedures, leaving open the question of whether simultaneous trigger-passage optimisation can be applied to non-agentic RAG-based systems.

Together, these studies highlight a broad spectrum of possible attack vectors: from altering retriever parameters to jointly optimising both triggers and passages in specialised agentic settings. What remains missing from the existing literature, however, is a systematic exploration of joint trigger-passage optimisation within retrieval-based RAG pipelines. This gap motivates my third contribution, where I extend beyond single-component attacks and investigate how coordinated optimisation of triggers and passages can be used to mount more powerful retrieval-time attacks.

Additional efforts have highlighted related vulnerabilities from complementary perspectives. Zhong et al. (2023) show that appending a small number of carefully crafted passages to corpora can cause retrievers to always rank them highly. Extending this idea, Chen et al. (2025a) show that RAG-based systems can be nudged toward biased positions on sensitive topics by training surrogate models. At the input level, Wallace et al. (2019) reveal that fixed trigger sequences, when added to inputs, can induce errors across a broad class of Natural Language Processing (NLP) models, underscoring the general susceptibility of language models to subtle adversarial control.

Similar attack paradigms have also emerged beyond language models, particularly in computer vision. *BadNets* (Gu et al., 2019) demonstrates that neural image classifiers trained on backdoored datasets can be manipulated to consistently misclassify inputs containing a fixed visual trigger, while behaving normally otherwise. Turner et al. (2019) refine this by crafting imperceptible perturbations to images without altering their labels, thereby evading data filtering while still implanting a functioning backdoor. 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 – be that images or retrieved text.

Several optimisation strategies exist for text-based problems. *HotFlip* (Ebrahimi et al., 2018) uses gradient information to score candidate substitutions, enabling efficient token-level edits that can be applied iteratively to an input sequence. *Greedy Coordinate Gradient (GCG)* (Zou et al., 2023) provides a related coordinate-descent formulation, while Guo et al. (2021) employ the *Gumbel-Softmax* reparametrisation trick to relax the discrete sampling problem into a continuous one, enabling end-to-end optimisation with gradient descent.

1.3 Contributions

My contributions in this thesis are as follows.

(C1) **BadRAG ablation studies.** I replicate the BadRAG framework, and conduct a systematic study of varying passage lengths and number of adversarial passages. This analysis highlights the trade-offs between attack success and stealth, offering new insights into how these key hyperparameters shape adversarial effectiveness.

(C2) **Trigger optimisation for fixed passages.** I invert the BadRAG setup by holding the passage constant and instead learning the trigger. This alternative formulation broadens the attack space and enables me to examine how both trigger length and insertion position affect retrieval.

(C3) **Joint trigger-passage optimisation.** Building on C2, I develop a unified approach that simultaneously optimises triggers and passages as a coupled system. This joint formulation sheds light on how coordinated manipulation of both components can strengthen retrieval-time attacks.

(C4) **Enhancing fluency of adversarial passages.** Finally, I explore strategies for improving the linguistic plausibility of jointly optimised passages in C3. By constraining certain edits to preserve fluency, I strike a balance between maintaining attack effectiveness and reducing detectability.

1.4 Threat Model

I assume a retrieval-time attacker whose goal is to manipulate the behaviour of a RAG-based system by exploiting its retriever. The adversary has access to retriever gradients for optimisation but cannot edit or fine-tune its weights. They also have no control over the downstream 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.

Within this framework, the attacker's capabilities differ slightly across contributions. In C1, C3 and C4, the attack requires corpus write access in order to insert a single optimised adversarial passage. By contrast, C2 assumes that the target passage already exists in the corpus, so the attacker does not require corpus write access and instead focuses solely on optimising the trigger. This highlights the distinction between passage-insertion and trigger-only attacks, both of which fit naturally within the retrieval-time threat model.

This scenario is realistic because many retrieval models are deployed out of the box without modification. Popular retrievers, such as Contriever, are widely used as frozen components, meaning an attacker could work against the same public model offline to design their manipulations. In addition, RAG deployments frequently index dynamic or user-generated corpora (e.g., company intranet pages, Wikipedia articles, etc.), where adversaries could plausibly add or edit content.

2 Methods

Retrieval-Augmented Generation (RAG) pipelines comprise three principal components: an external corpus $\mathcal{C}$, a retriever, and a generator. 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-x passages with the highest scores. These passages are concatenated with the original query and provided as context to the language model for generation.

Dataset. I evaluate my methods on the Natural Questions dataset (Kwiatkowski et al., 2019), which consists of real user queries issued to Google Search, paired with relevant passages from Wikipedia. The full corpus contains 2,681,468 passages; there are also 3,452 accompanying queries. To make experimentation computationally feasible while maintaining statistical robustness, I sample a subset of $N = 10{,}000$ passages to serve as the retrieval corpus $\mathcal{C}$. A passage from the dataset is given below.

> *East Germans successfully defected by a variety of methods: digging long tunnels under the Wall, waiting for favorable winds and taking a hot air balloon, sliding along aerial wires, flying ultralights and, in one instance, simply driving a sports car at full speed through the basic, initial fortifications. When a metal beam was placed at checkpoints to prevent this kind of defection, up to four people (two in the front seats and possibly two in the boot) drove under the bar in a sports car that had been modified to allow the roof and windscreen to come away when it made contact with the beam. They lay flat and kept driving forward. The East Germans then built zig-zagging roads at checkpoints. The sewer system predated the Wall, and some people escaped through the sewers,[95] in a number of cases with assistance from the Unternehmen Reisebüro.[96]*

From the available queries, I select 1,000, splitting them evenly into 500 for training/validation and 500 held out for evaluation. A query from the dataset is given below.

> *when did the subway open in new york*

Retriever. In my experiments, the retriever is a bi-encoder model with shared weights between E_q and E_p, specifically the publicly available Contriever model (Izacard et al., 2022). Contriever is trained with a contrastive objective over large amounts of unlabelled text, producing semantically meaningful 768-dimensional 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-based systems.

At this point, it is worth clarifying the distinction between words and tokens. The Contriever retriever employs a subword tokenizer, so tokens in its vocabulary do not, in general, correspond to whole natural-language words. In my implementation, however, I restrict candidate replacement tokens to alphabetic, non-special entries, meaning that each optimised token always corresponds to a complete word. As a result, for the purposes of this work, the terms word and token can be used interchangeably. For clarity and consistency with prior literature, I adopt the term token from now on.

Similarity Scoring. 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 dot products 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 passage. The attacker's goal is to increase similarity between the adversarial passage and triggered queries, while decreasing similarity to clean queries. This is achieved by minimising the following loss, expressed as an expectation over the underlying query distribution:

$$\mathcal{L} = -\mathbb{E}_{q^\tau \sim \mathcal{Q}_{\text{trig}}}\big[\text{sim}(q^\tau, p_{\text{adv}})\big] \; + \; \lambda \cdot \mathbb{E}_{q \sim \mathcal{Q}_{\text{clean}}}\big[\text{sim}(q, p_{\text{adv}})\big], \tag{1}$$

where $\lambda > 0$ controls the relative penalty for similarity to clean queries. In practice, Equation (1) is approximated by the empirical average over batches:

$$\mathcal{L} = -\frac{1}{|\mathcal{Q}_{\text{trig}}|} \sum_{q^\tau \in \mathcal{Q}_{\text{trig}}} \text{sim}(q^\tau, p_{\text{adv}}) \; + \; \lambda \cdot \frac{1}{|\mathcal{Q}_{\text{clean}}|} \sum_{q \in \mathcal{Q}_{\text{clean}}} \text{sim}(q, p_{\text{adv}}). \tag{2}$$

The first term encourages retrieval of the adversarial passage when the trigger is present, while the second term suppresses retrieval for unmodified inputs.

2.1 C1: BadRAG ablation studies

In the original Contrastive Optimisation on a Passage (COP) attack (Xue et al., 2024), the attacker's objective is to craft an adversarial passage p_{adv} that is retrieved only when a specific fixed trigger token 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.

The result of this optimisation is a semantically incoherent passage which nevertheless functions as an effective retrieval backdoor. Although the adversarial passage does not convey any meaningful information to the user, it has three important implications. First, it faithfully replicates the COP attack introduced in BadRAG, allowing me to conduct a controlled ablation study of the original method. Second, by monopolising the retriever's attention with irrelevant or misleading passages, such an attack could induce a denial-of-service condition, where legitimate content is displaced by adversarially injected noise. Third, this contribution provides a foundation for my subsequent methods (C2-C4), where I progressively extend the basic optimisation strategy to yield more realistic, targeted and stealthy attacks.

At a high level, C1 instantiates this attack by fixing a trigger and using gradient-based discrete optimisation to iteratively improve the adversarial 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 as per 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 formalises the baseline COP attack, in which only the adversarial passage is optimised while the trigger is fixed in advance.

Algorithm 1 Contrastive Optimisation on a Passage (COP)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, fixed trigger τ, passage length ℓ, trigger insertion location, HotFlip candidates k, loss weight λ, patience P, max steps T

Initialise $p_{\text{adv}} \in \mathbb{N}^{\ell}$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set `best_loss` $\leftarrow +\infty$, `no_improve` $\leftarrow 0$
for step $= 1$ to T **do**
    Sample batch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries $B^{\tau} \leftarrow \{\, \text{insert}(\tau, q) : q \in B \,\}$
    Encode B, B^{τ}, and p_{adv}
    Compute training loss $\mathcal{L}$ as in Equation (2)
    Differentiate $\mathcal{L}$ w.r.t. token positions of p_{adv}
    Select random passage position $i \in \{1, \ldots, \ell\}$; generate top-$k$ HotFlip candidates for i
    Evaluate each candidate by replacing $p_{\text{adv}}[i]$ and recomputing $\mathcal{L}$; apply the best
    On validation set, compute $\mathcal{L}_{\text{val}}$
    if $\mathcal{L}_{\text{val}} <$ `best_loss` **then**
        `best_loss` $\leftarrow \mathcal{L}_{\text{val}}$; 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 C2: Trigger optimisation for fixed passages

In my second contribution, I go beyond the original COP setup in two key ways. First, I invert the problem: rather than optimising the passage, I hold it fixed and instead optimise the trigger. Second, I relax the restriction that the trigger must consist of a single token, and instead allow it to be a short sequence of tokens $\tau = (t_1, \ldots, t_m)$. The attacker's objective is to discover such a trigger that, when inserted into a clean query q, causes the retriever to rank the target passage p_{adv} highly. Crucially, the trigger should have minimal effect when absent, ensuring that clean queries without τ do not retrieve p_{adv}.

This formulation reflects a more restrictive but arguably more realistic threat model, where the adversary is unable to inject new passages into the retrieval corpus and can only influence the query side. In this setting, the attack surface is broader because the barrier to exploitation is lower: the attacker does not need corpus write access, only the ability to induce users (or client applications) to issue queries containing the trigger. This could enable an adversary to artificially promote a specific document that is already present in the corpus – for example, an existing but otherwise rarely retrieved article – or to bias the retriever toward a fixed narrative by elevating certain sources over others. In this way, the attack removes the assumption of corpus control and demonstrates how query-only manipulation can still produce effective retrieval-time backdoors.

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 batch 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 similarity-based objective defined in 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 loss ceases to improve. The result is an optimised trigger that selectively activates the target passage while remaining inert otherwise.

Algorithm 2 describes the trigger optimisation procedure. Lines highlighted in blue indicate the differences from COP (Algorithm 1), namely that the passage is fixed while the trigger tokens are initialised and updated.

Algorithm 2 Contrastive Optimisation on a Trigger (COT)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, fixed target passage p_{adv}, trigger length m, trigger insertion location, HotFlip candidates k, loss weight λ, patience P, max steps T

Initialise trigger $\tau \in \mathbb{N}^m$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set `best_loss` $\leftarrow +\infty$, `no_improve` $\leftarrow 0$
for step $= 1$ to T **do**
    Sample batch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries $B^{\tau} \leftarrow \{\, \text{insert}(\tau, q) : q \in B \,\}$
    Encode B, B^{τ}, and p_{adv}
    Compute training loss $\mathcal{L}$ as in Equation (2)
    Differentiate $\mathcal{L}$ w.r.t. token positions of τ
    Select random trigger position $i \in \{1, \ldots, m\}$; generate top-k HotFlip candidates for i
    Evaluate each candidate by replacing $\tau[i]$ and recomputing $\mathcal{L}$; apply the best
    On validation set, compute $\mathcal{L}_{\text{val}}$
    if $\mathcal{L}_{\text{val}} <$ `best_loss` **then**
        `best_loss` $\leftarrow \mathcal{L}_{\text{val}}$; 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 C3: Joint trigger-passage optimisation

C3 removes the constraints imposed in C1 and C2 (see Sections 2.1 and 2.2), allowing the attacker to jointly optimise both the trigger and the 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 τ that selectively activates it. The objective remains to maximise similarity between triggered queries q^{τ} and the adversarial passage p_{adv}, while minimising similarity for clean queries q without the trigger, as in Equation (2).

The ability to simultaneously optimise both components increases the adversary's leverage considerably. Unlike C1 and C2, which were restricted to manipulating only one side of the retrieval process, C3 combines corpus-side and query-side control. This produces attacks that are not only stronger – since the trigger and passage can be tuned to reinforce one another – but also potentially stealthier, as the optimised trigger often consists of a random-looking combination of tokens, and the adversarial passage is surfaced only when this sequence appears in a query. In this way, C3 demonstrates how relaxing constraints on the attacker's capabilities can potentially yield both higher attack success and greater concealment.

At a high level, joint optimisation begins with both the trigger and the passage initialised as neutral placeholders (`[MASK]`). In each iteration, a batch 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 adversarial passage are encoded, and the training loss is computed to guide updates. Updates alternate stochastically between the trigger and the 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 passage token is updated in the same way. By interleaving these two update modes, the optimisation co-evolves the trigger and passage, producing mutually reinforcing combinations that maximise the attack objective. Early stopping on a held-out validation set prevents overfitting to the training queries.

Algorithm 3 extends COP by jointly optimising both the trigger and the passage. Lines highlighted in blue mark the additional steps compared to COP (Algorithm 1), specifically the stochastic alternation between trigger and passage updates.

Algorithm 3 Joint Contrastive Optimisation on Trigger and Passage (JCO)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, trigger length m, passage length ℓ, trigger insertion location, trigger update probability π_{trig}, HotFlip candidates k, loss weight λ, patience P, max steps T

Initialise trigger $\tau \in \mathbb{N}^m$ and passage $p_{\text{adv}} \in \mathbb{N}^\ell$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set `best_loss` $\leftarrow +\infty$, `no_improve` $\leftarrow 0$
for step $= 1$ to T **do**
    Sample batch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries $B^\tau \leftarrow \{\, \text{insert}(\tau, q) : q \in B \,\}$
    Encode B, B^τ, and p_{adv}
    Compute training loss $\mathcal{L}$ as in Equation (2)
    Differentiate $\mathcal{L}$ w.r.t. token positions of both τ and p_{adv}
    if $u \sim \mathcal{U}(0,1) < \pi_{\text{trig}}$ **then**
        Select random trigger position $i \in \{1, \ldots, m\}$; generate top-k candidates; recompute $\mathcal{L}$; apply the best
    else
        Select random passage position $j \in \{1, \ldots, \ell\}$; generate top-$k$ candidates; recompute $\mathcal{L}$; apply the best
    end if
    On validation set, compute $\mathcal{L}_{\text{val}}$
    if $\mathcal{L}_{\text{val}} <$ `best_loss` **then**
        `best_loss` $\leftarrow \mathcal{L}_{\text{val}}$; 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 C4: Enhancing fluency of adversarial passages

C4 extends the joint optimisation framework of C3 (see Section 2.3) by enabling the attacker to implant a chosen message directly into the adversarial passage. Whereas the adversarial passages in C1 and C3 are optimised solely for retrieval effectiveness and may therefore consist of unnatural or incoherent token sequences, C4 allows the attacker to embed arbitrary misinformation content while still benefiting from joint optimisation. This is achieved by decomposing the adversarial passage into an optimised prefix and a fixed suffix generated by an LLM, where the suffix encodes the target statement that the attacker wishes to promote. This idea was inspired by the PoisonedRAG framework (Zou et al., 2024), where a similar trick is performed to optimise a passage for a fixed query.

The resulting attack is both stronger and more practical. By constraining part of the adversarial passage to convey a coherent, high-fluency message produced by an LLM, the adversary can ensure that the retrieved content explicitly communicates their desired misinformation rather than a meaningless jumble of tokens. At the same time, the optimisation of the prefix and trigger preserves retrieval effectiveness, meaning that the adversarial passage can still reliably displace legitimate results under the trigger. This dual design also reduced detectability: many retrieval pipelines apply fluency-based filtering to exclude suspicious passages – see Jain et al. (2023) and Alon and Kamfonas (2023). However the LLM-generated suffix provides naturalness that makes detection more difficult. In this way, C4 demonstrates how backdoor optimisation can be extended from proof-of-concept manipulations to targeted misinformation injection.

At a high level, C4 operates by initialising both the trigger and the prefix with neutral placeholder tokens (`[MASK]`), while the suffix is fixed in advance by prompting an LLM with the target misinformation. At each iteration, a batch of clean queries is sampled and used to form triggered queries by inserting the current trigger sequence at a chosen location. The adversarial passage is constructed as the concatenation $p_{\text{adv}} = p_{\text{pre}} \oplus p_{\text{suf}}$, and the loss in Equation (2) is computed. Gradients are used to propose HotFlip-based replacements, with updates stochastically alternating between the trigger and prefix, while the suffix remains unchanged. This allows the trigger and prefix to adaptively reinforce one another in maximising retrieval of the fixed suffix, thereby implanting the target message in a retrievable yet natural-looking form.

Algorithm 4 extends the joint optimisation further by decomposing the adversarial passage into a trainable prefix and a fixed LLM-generated suffix. Lines highlighted in blue

show the changes relative to COP (Algorithm 1), namely the construction of the adversarial passage from prefix and suffix, and the restriction that only prefix and trigger tokens are updated.

Algorithm 4 Joint Misinformation Optimisation (JMO)

Require: Clean query set $\mathcal{Q}_{\text{clean}}$, trigger length m, prefix length ℓ_{pre}, suffix length ℓ_{suf}, target misinformation text, trigger insertion location, trigger update probability π_{trig}, HotFlip candidates k, loss weight λ, patience P, max steps T

Generate fixed suffix $p_{\text{suf}} \in \mathbb{N}^{\ell_{\text{suf}}}$ from target misinformation text using an LLM
Initialise trigger $\tau \in \mathbb{N}^{m}$ and prefix $p_{\text{pre}} \in \mathbb{N}^{\ell_{\text{pre}}}$ with `[MASK]` tokens
Split $\mathcal{Q}_{\text{clean}}$ into training and validation subsets
Set `best_loss` $\leftarrow +\infty$, `no_improve` $\leftarrow 0$
for step $= 1$ to T **do**
    Sample batch $B \subset \mathcal{Q}_{\text{clean}}^{\text{train}}$
    Form triggered queries $B^{\tau} \leftarrow \{\, \text{insert}(\tau, q) : q \in B \,\}$
    Construct adversarial passage $p_{\text{adv}} \leftarrow p_{\text{pre}} \oplus p_{\text{suf}}$
    Encode B, B^{τ}, and p_{adv}
    Compute training loss $\mathcal{L}$ as in Equation (2)
    Differentiate $\mathcal{L}$ w.r.t. token positions of τ and p_{pre} (suffix fixed)
    if $u \sim \mathcal{U}(0,1) < \pi_{\text{trig}}$ **then**
        Select random trigger position; generate top-k candidates; recompute $\mathcal{L}$; apply the best
    else
        Select random prefix position; generate top-k candidates; recompute $\mathcal{L}$; apply the best
    end if
    On validation set, reconstruct p_{adv} with fixed suffix and compute $\mathcal{L}_{\text{val}}$
    if $\mathcal{L}_{\text{val}} <$ `best_loss` **then**
        `best_loss` $\leftarrow \mathcal{L}_{\text{val}}$; store current (τ, p_{pre}); `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 final $p_{\text{adv}} = p_{\text{pre}} \oplus p_{\text{suf}}$

3 Results

All four contributions (C1-C4) were implemented within a common optimisation framework and share a set of hyperparameters. To avoid repetition, these are listed here once. Table 1 summarises the fixed quantities, their symbols and the values used in all experiments.

Quantity	**Symbol**	**Value**
Query batch size	B	32
Corpus size	N	10,000
Trigger update probability	π_{trig}	0.2
Patience for early stopping	P	20
Maximum optimisation steps	T	1000

Table 1: Fixed experimental hyperparameters used across all four contributions.

The following hyperparameters were varied across experiments, depending on the contribution under study: passage length ℓ, number of passages injected into the retrieval corpus n, trigger length m, insertion location of the trigger (start, end or random), HotFlip candidates k, and the clean-query penalty weight λ.

3.1 C1: BadRAG ablation studies

Experimental Setup. I conducted two complementary experiments to assess how passage length and passage count affect COP attack performance. I followed Algorithm 1 with the common hyperparameters in Table 1. I used insertion location set to random, setting $\lambda = 0.5$ and $k = 10$. I also used the following 25 trigger tokens sampled from the retriever vocabulary: `bass`, `handwriting`, `campaigning`, `nottinghamshire`, `hamilton`, `civilian`, `astonished`, `miners`, `shivered`, `stack`, `lafayette`, `lesley`, `bases`, `dempsey`, `cooperative`, `lead`, `spruce`, `langley`, `garnered`, `destroyer`, `absently`, `percussion`, `canadian`, `theology`, `mike`.

- **Experiment 1: Passage length ablation.** I varied the adversarial passage length as $\ell \in \{5\} \cup \{10, 20, \ldots, 230\}$. For each passage length and each trigger, I generated 5 independent adversarial passages.
- **Experiment 2: Passage count ablation.** I fixed the adversarial passage length to $\ell = 50$ and increased the number of independent adversarial passages per trigger to 25.

In both experiments, each learned passage was appended to the retrieval corpus and evaluated on the held-out set of 500 clean queries (and 500 triggered queries formed by inserting the trigger at random locations). I report the proportion of queries for which the adversarial passage appears in the top-10 retrieved results for each query type, with higher top-10 frequency for triggered queries indicating stronger attacks and lower top-10 frequency for clean queries indicating greater stealth.

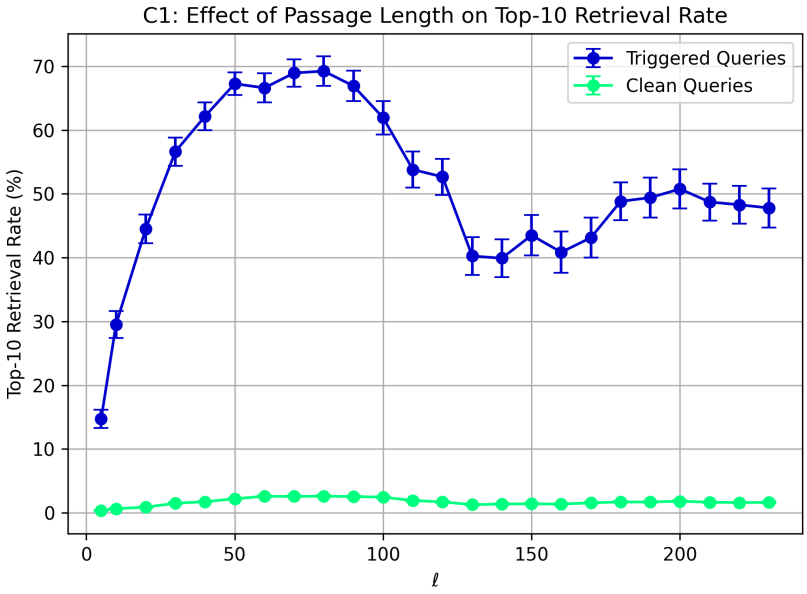

Figure 2: C1: Effect of adversarial passage length ℓ on top-10 retrieval rate. Results are averaged across 25 triggers with 5 passages generated for each. Error bars indicate the standard error of the mean.

Results: Passage Length Ablation. Figure 2 shows that for triggered queries, retrieval effectiveness increases rapidly as passage length grows from $\ell = 5$ to $\ell = 50$. Performance peaks between $\ell = 50$ and $\ell = 90$, reaching a top-10 retrieval rate of approximately 70%. Beyond this point, the effectiveness declines, likely because longer passages dilute the contribution of highly optimised tokens with additional, less discriminative ones. After a drop around $\ell = 130$–160, the curve stabilises and levels off beyond $\ell = 180$, where retrieval rates remain moderate but below the peak. This plateau suggests that excessively long passages converge toward a balance in which some relevant cues are retained, but their impact is offset by noise from surplus tokens.

For clean queries, the top-10 retrieval rate remains consistently low ($< 4\%$) across all passage lengths, indicating that increasing ℓ does not compromise stealth. These findings

highlight a clear trade-off: moderate passage lengths (around 50 tokens) maximise attack success, while very short passages lack sufficient discriminatory power and very long passages suffer from token dilution effects.

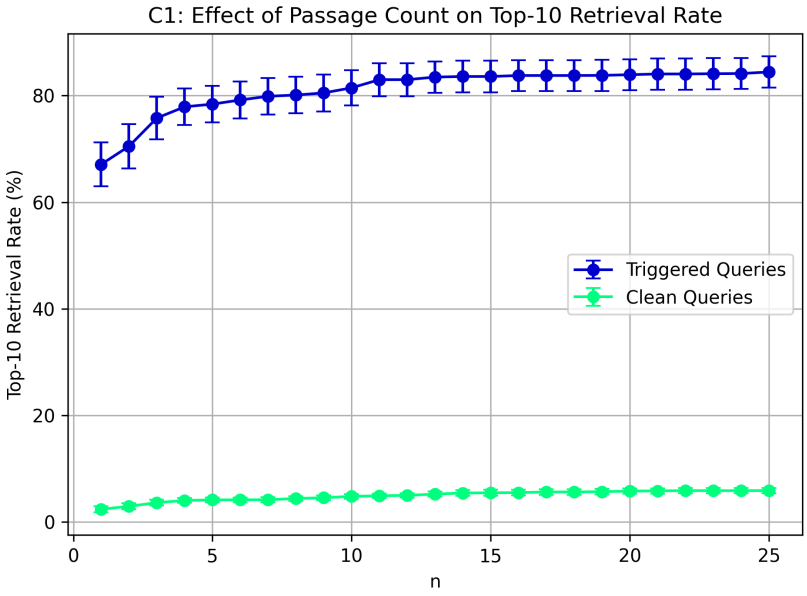

Figure 3: C1: Effect of the number of adversarial passages per trigger n on top-10 retrieval rate. Passage length fixed at $\ell = 50$. Results are averaged across 25 triggers. Error bars indicate the standard error of the mean.

Results: Passage Count Ablation. Figure 3 shows the effect of varying the number of adversarial passages per trigger while fixing passage length at $\ell = 50$. For triggered queries, the top-10 retrieval rate increases rapidly from approximately 65% with a single passage to over 80% by $n = 10$, and then gradually saturates near 85%. This monotonic trend reflects a coverage effect: multiple independent adversarial passages optimised for the same trigger populate different regions of the embedding space, raising the probability that at least one ranks in the top-10 for a triggered query. Once the embedding space is sufficiently covered, further adversarial passages add redundancy rather than new retrieval opportunities.

For clean queries, the top-10 retrieval rate remains consistently low, rising only modestly from around 2% at $n = 1$ to approximately 6% at $n = 25$. This indicates that the main stealth cost arises from the increased corpus footprint: more adversarial passages create more opportunities for accidental retrieval, but the effect remains minor. Overall, these results show that attack strength can be boosted not only by tuning passage length, but

also by injecting a modest number of diverse adversarial passages per trigger, with most of the gain achieved once $n = 5$.

It is important to note that the exact retrieval rates reported here are specific to my experimental corpus of $N = 10{,}000$ passages. As a result, with a larger or smaller corpus, the optimal value of n would shift depending on the level of competition in the ranking. However, the qualitative behaviour – the rapid improvement with a small number of adversarial passages followed by saturation – reflects the underlying optimisation dynamics and would hold regardless of corpus size.

Trigger Variability. While the above experiments averaged performance over 25 randomly selected trigger tokens, a closer inspection reveals substantial variability between triggers. Some triggers are far more conducive to the attack than others, with adversarial passages surfacing reliably near the top of the rankings, while others produce much weaker effects. Figure 4 illustrates this contrast using a single adversarial passage of length $\ell = 50$ per trigger. For the trigger `miners`, the adversarial passage shown below achieved strong retrieval alignment (mean rank ≈ 125).

> *vendors dinner wine fountains tent pubs meditation salsa floral rattle fairs overhead uranium explosions adds imagination freeing stereotypes support breadth revival within miners gathers wider revolution including palace bowman pea slave*

At this point, it is worth noting that although the passage length was set to $\ell = 50$, the rendered outputs appear shorter. This is because optimisation begins from sequences of `[MASK]` tokens. Early stopping can leave some positions unreplaced, meaning that the final passage may contain fewer than 50 tokens.

By contrast, for the trigger `astonished`, the adversarial passage below produced a much weaker effect (mean rank ≈ 1215).

> *counters ordinary bears supervised metropolitan turnout theirs misconduct regime electoral whilst mosques deeply reassure reorganisation prisoners cheer consultation smash hydro hooks enormous unless spoil amazed apple crambidae startled respecting changes flew astonished nascar business towing parts astonishing fundamentally wired capabilities replace handsome hooks bulbs*

This sharp disparity underscores a key limitation of the original COP formulation: since the trigger is fixed in advance, attack success depends heavily on the serendipitous choice

of token. This observation directly motivates my next contribution (C2), where I lift this constraint by explicitly optimising the trigger itself.

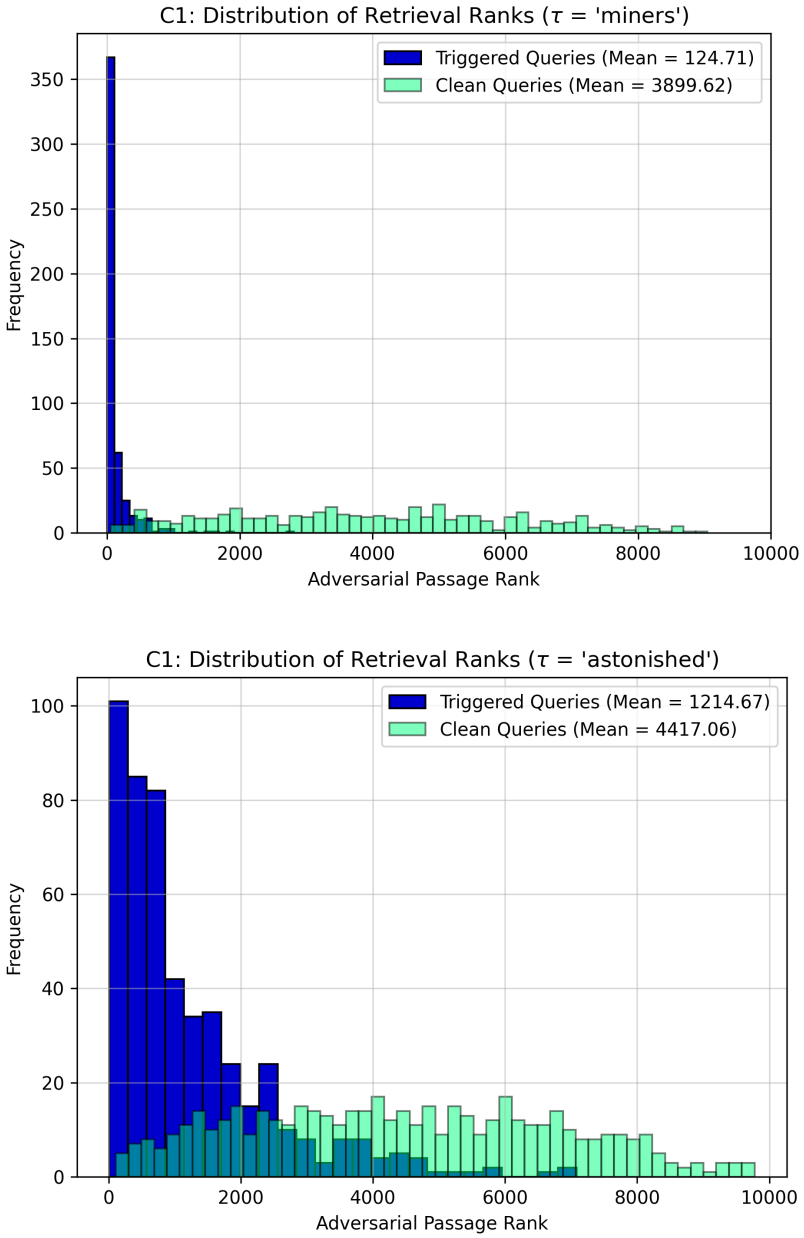

Figure 4: C1: Distribution of retrieval ranks for adversarial passages of length $\ell = 50$, each optimised for a different trigger. TOP: passage optimised for the trigger `miners`; BOTTOM: passage optimised for the trigger `astonished`. Blue bars show ranks for the 500 held-out triggered queries; green bars show ranks for the 500 corresponding clean queries. The disparity underscores that some triggers are intrinsically more amenable to COP than others.

3.2 C2: Trigger optimisation for fixed passages

Experimental Setup. In contrast to C1, where the adversarial passage is optimised for a fixed trigger, here I fix an existing passage from the corpus and instead optimise the trigger sequence that activates it (Algorithm 2). I set $\lambda = 0.5$ and $k = 10$. I also randomly sampled 25 passages from Natural Questions; the full list is available on the project GitHub, linked in Section 5. For each passage, I ran the procedure for triggers of lengths $m \in \{1, 2, 3, 4, 5\}$, running 5 independent optimisations per length to control for variability. I studied two insertion policies: (i) Random – the trigger is inserted at a uniformly random position in queries; and (ii) End – the trigger is appended to the end of queries.

The experiment was evaluated on the held-out set of 500 clean queries (and 500 triggered queries formed by inserting the trigger at the appropriate location). I report the proportion of queries in which the target passage appears in the top-10 retrieved results, with higher top-10 frequency for triggered queries indicating stronger attacks and lower top-10 frequency for clean queries indicating greater stealth.

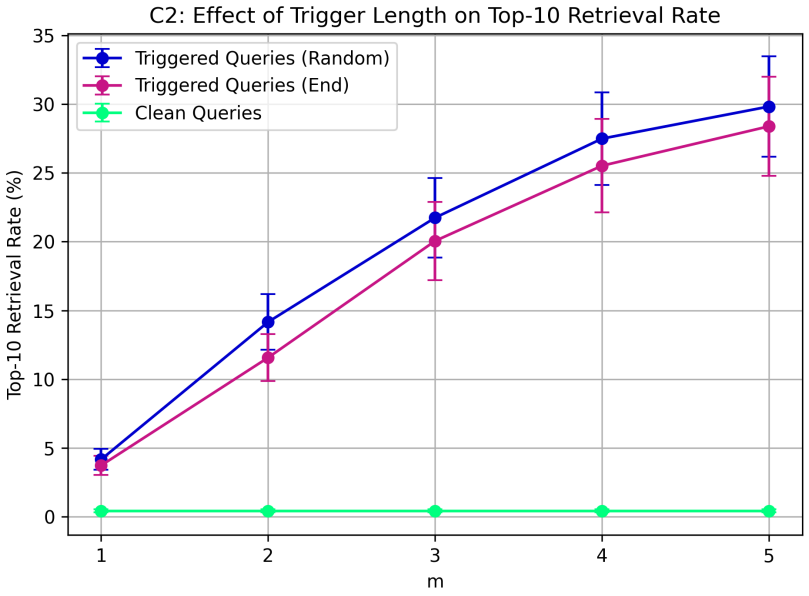

Figure 5: C2: Effect of trigger length m and insertion position on top-10 retrieval rate. Results are averaged across 25 corpus passages with 5 triggers generated for each. Error bars indicate the standard error of the mean.

Results. Figure 5 shows that the top-10 retrieval rate for triggered queries increases steadily with trigger length, rising from roughly 4% at a single token to nearly 30% with five tokens. Contrary to what we might expect, inserting the trigger at a random position consistently outperforms appending it at the end, across all trigger lengths tested. A plausible explanation is that random insertion places the trigger adjacent to semantically rich content words, creating stronger local interactions in the encoder and distributing the adversarial signal across positions.

To show an example of an optimised trigger, consider the passage from the corpus beginning *"Another suggestion is that the festival celebrates a widespread feature of east Asian agrarian societies: the harvest of winter wheat..."* One of the held-out queries *"who came up with the idea of using convoys"* retrieves the target passage at rank 8,530. Inserting the learned trigger (`harmonies`) at a random position yields *"who came up* ***harmonies*** *with the idea of using convoys"*, which instead retrieves the same passage at rank 958. This example illustrates how a short, position-variable trigger can sharply elevate a fixed passage's rank for triggered queries, while leaving its retrieval probability under unrelated clean queries largely unaffected.

A natural extension of these findings is to hypothesise that the optimal insertion position of a trigger may depend on the specific query. In other words, while random insertion performs well on average, certain queries could benefit more from triggers placed adjacent to particular lexical or semantic cues. However, I do not investigate this direction here, as my goal is to develop attacks that are general and query-agnostic rather than tailored to individual queries. Exploring strategies that explicitly optimise both the trigger tokens and their placement could nevertheless represent an interesting avenue for future work.

Comparison to C1. It is interesting to note that the absolute top-10 rates here are substantially lower than those in C1 (cf. Figure 2). This gap arises because C1 assumes that the attacker can insert new, fully optimised adversarial passages into the corpus. In C2, by contrast, the passage is fixed in advance and drawn directly from the existing corpus, 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.

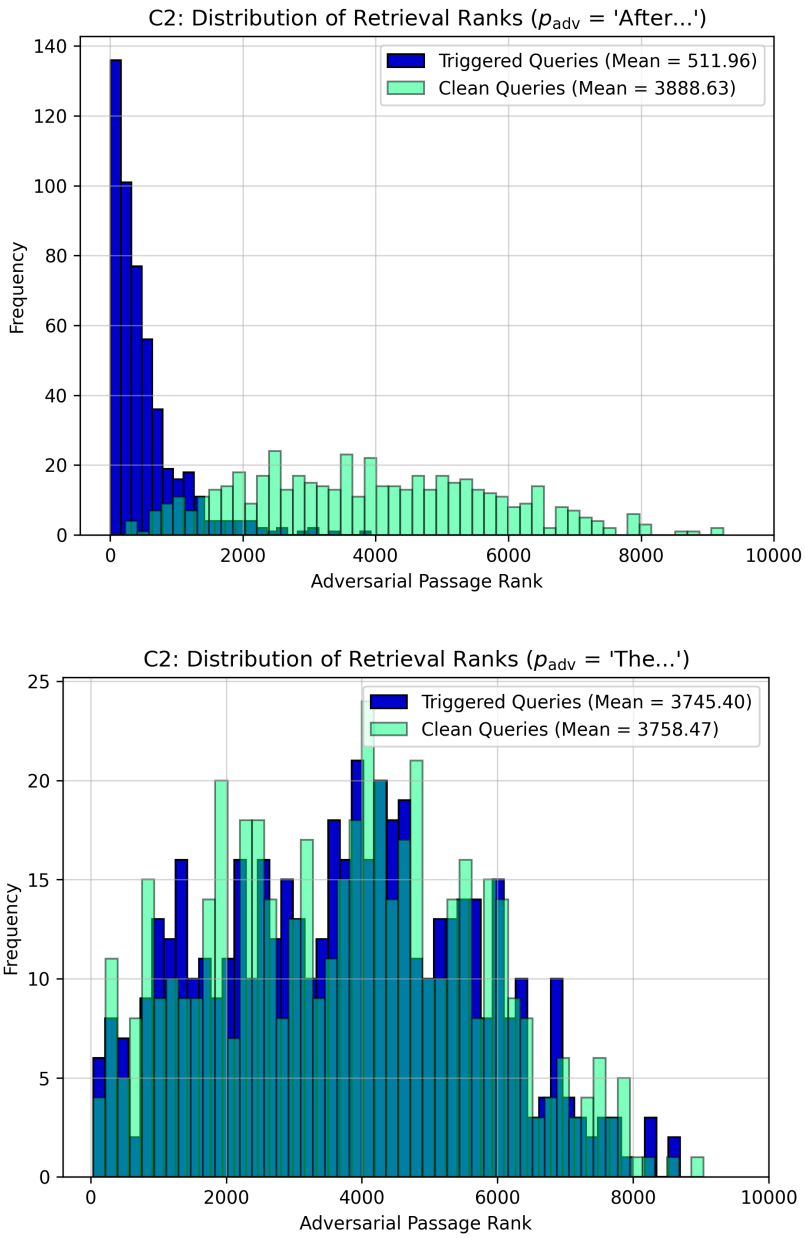

Figure 6: Distribution of retrieval ranks for passages targeted with a single-token trigger ($m = 1$). TOP: passage beginning *"After Slesinger's death in 1953..."* with trigger `lass`, showing strong separation between triggered and clean queries. BOTTOM: passage beginning *"The evolutionary biologist Ernst Mayr..."* with trigger `var`, showing little separation and indicating weak susceptibility to trigger optimisation. Blue bars show ranks for the 500 held-out triggered queries; green bars show ranks for the 500 corresponding clean queries.

Passage Variability. Similarly to C1, we observe substantial variability in how conducive different passages are to trigger optimisation. For illustration, Figure 6 shows two contrasting passages from the C2 runs, each paired with one of the learned one-token triggers ($m = 1$) optimised for that passage. For the passage beginning *"After Slesinger's death in 1953..."* and trigger `lass`, triggered queries are sharply elevated (mean rank ≈ 512) relative to clean queries (mean ≈ 3889). By contrast, for the passage beginning *"The evolutionary biologist Ernst Mayr..."* and trigger `var`, triggered and clean ranks are nearly indistinguishable (means ≈ 3745 vs. ≈ 3758). These single-passage cases illustrate that some documents contain lexical or semantic affordances that interact productively with learned triggers, whereas others resist manipulation – motivating the joint optimisation explored in C3.

3.3 C3: Joint trigger-passage optimisation

Experimental Setup. For C3 I jointly optimise both the trigger and the adversarial passage according to Algorithm 3, with the common hyperparameters in Table 1. I set $\lambda = 0.5$, $k = 10$, and used insertion location set to random. To explore how trigger length and passage length interact under joint optimisation, I varied the trigger length as $m \in \{1, 2, 3, 4, 5\}$ and the passage length as $\ell \in \{20, 30, 40, 50\}$. For each combination of (m, ℓ), I generated 25 independent trigger-passage pairs to account for variability.

Each learned adversarial passage was inserted into the retrieval corpus and evaluated on the held-out set of 500 clean queries (and 500 triggered queries formed by inserting the trigger at random locations). As in prior contributions, I report the proportion of queries in which the adversarial passage appears in the top-10 retrieved results, with higher top-10 rates for triggered queries indicating stronger attacks and lower rates for clean queries reflecting greater stealth.

Results. Figures 7 and 8 show top-10 retrieval rates for triggered and clean queries under joint optimisation. For triggered queries, retrieval success rises steeply with trigger length, reaching near-saturation ($> 95\%$) by $m = 5$ across all passage lengths. At every trigger length, one of the longer passage lengths ($\ell = 40$ or $\ell = 50$) achieves the highest retrieval rate, suggesting that the additional optimised tokens provide richer semantic cues that help compensate for limited trigger capacity. This advantage is most pronounced at $m = 1$–2, but persists, albeit more subtly, across larger trigger lengths.

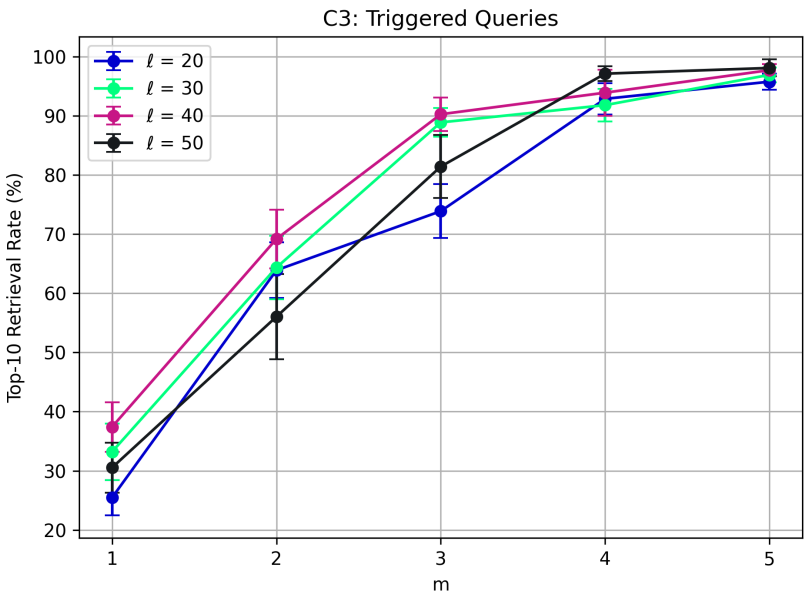

Figure 7: C3: Top-10 retrieval rate for triggered queries, varying trigger length m and passage length ℓ. Error bars indicate the standard error of the mean over 25 trials.

For clean queries, top-10 retrieval rates remain consistently low ($< 3\%$) across all conditions, confirming that stealth is broadly preserved. However, we do observe a systematic trend: top-10 retrieval frequency under clean queries increases gradually with trigger length, suggesting a small leakage effect. One explanation for this could be that longer triggers give the optimiser more degrees of freedom, encouraging the adversarial passage to capture broader semantic features that sometimes align with clean queries even in the absence of the trigger. Notably, this leakage is not eliminated by increasing the clean-query penalty. Even with $\lambda = 0.9$, the same upward trend persists, albeit at lower absolute values, indicating that the effect reflects intrinsic properties of joint optimisation rather than insufficient regularisation. Results for $\lambda = 0.9$ are available on the project GitHub, linked in Section 5.

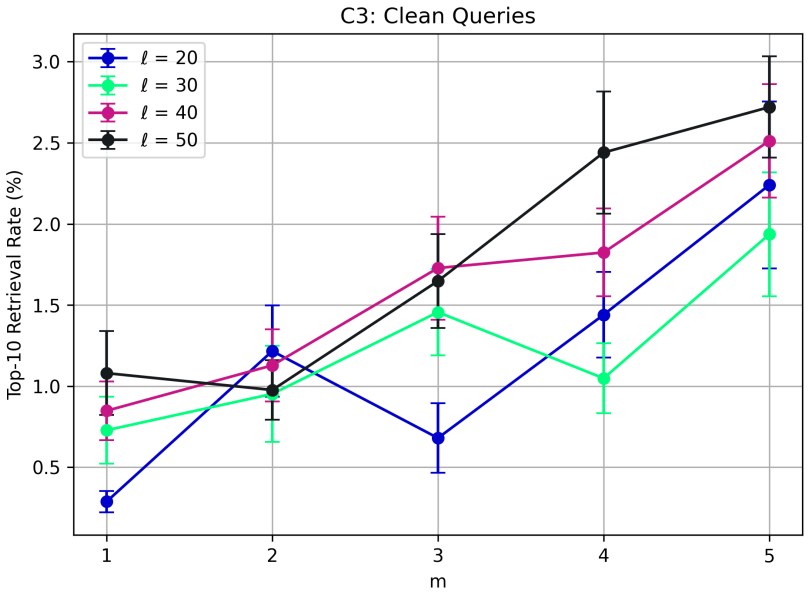

Figure 8: C3: Top-10 retrieval rate for clean queries, varying trigger length m and passage length ℓ. Error bars indicate the standard error of the mean over 25 trials.

Comparison to C1 (and Why C3 Lags at $m = 1$). Despite its strong overall performance, C3 underperforms C1 when the trigger is restricted to a single token (cf. Figure 2). In C1, optimisation focuses solely on the passage for a fixed trigger, producing highly tuned adversarial passages. By contrast, C3 must co-optimise both the trigger and passage, whose effects interact nonlinearly in the encoder. This makes the search landscape harder: a one-token trigger update can induce abrupt shifts in the gradient field seen by the passage (and vice versa), leading to unstable progress. This dynamic is visible in the validation-loss traces presented in Figure 9.

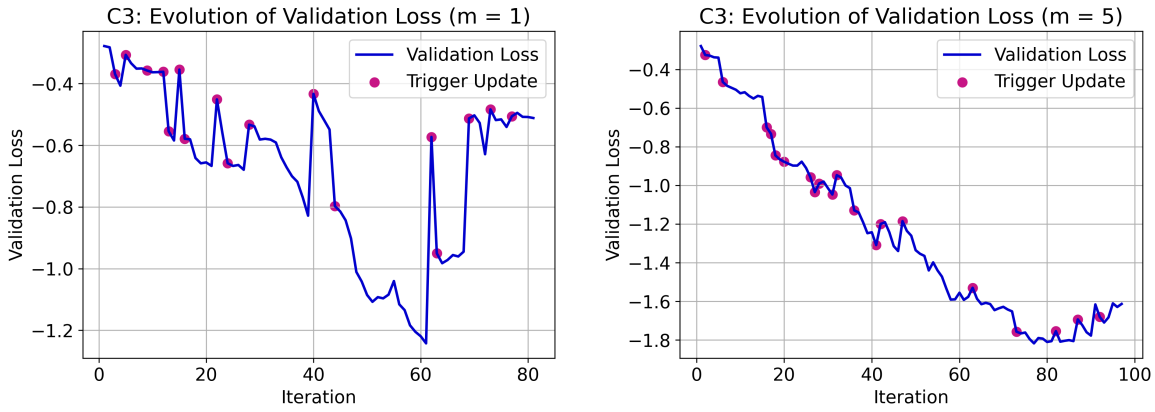

Figure 9: C3: Validation loss for different trigger lengths m over a single run of Algorithm 3. Pink points mark iterations where the trigger was updated. LEFT: $m = 1$, showing erratic oscillations from single-token updates. RIGHT: $m = 5$, showing smoother convergence as redundancy across tokens stabilises co-adaptation with the passage.

Taken together, these results suggest a clear pattern: joint optimisation achieves very high attack success when the trigger has modest length ($m = 4$–5), 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 highlights the benefits of optimising just the passage for a fixed trigger; C2 shows that optimising just the trigger for a fixed passage is weaker but still useful; C3 demonstrates that co-adaptation of both amplifies attack success – once the trigger length provides enough degrees of freedom to stabilise the search.

Qualitative Visualisation of Embedding Space. To better understand how joint optimisation positions triggered queries and the adversarial passage in the embedding space, we can perform a qualitative inspection using Principal Component Analysis (PCA). I took one representative run from the C3 experiments with trigger length $m = 5$ and passage length $\ell = 50$, and projected the embeddings of the entire corpus, the adversarial passage, a clean query, and a triggered variant onto the first two principal components. The optimised trigger was as follows

```
auxiliary embarked honneur procurement assam
```

and the corresponding adversarial passage was as follows.

ju recapture lieu ammunition maha cameras worth armoured warships lasers stills justice beans overture wears symbols spare salute traditions xx descend letters engages parades into stew decorated upside command glittering star nonfiction resembling fallen ropes

The held-out clean query used for the purposes of this visualisation is *"where did remember the titans camp take place"*, and the triggered variant was formed by random insertion: *"auxiliary embarked honneur procurement assam where did remember the titans camp take place"*.

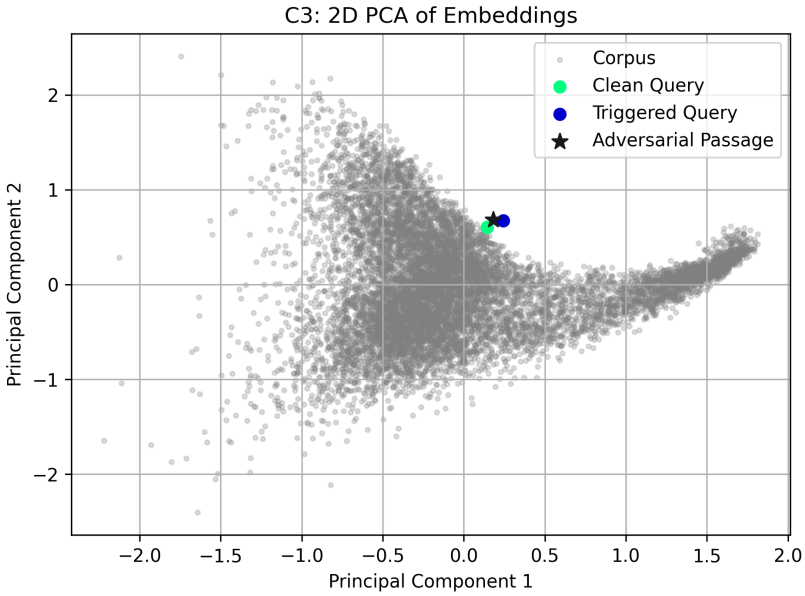

Figure 10: C3: 2D PCA projection of embedding space for one representative optimised trigger-passage pair ($m = 5$, $\ell = 50$). Grey points show the full corpus; green shows a clean query; blue shows its triggered variant; and the black star shows the adversarial passage.

Figure 10 suggests that the triggered query is drawn towards the adversarial passage and away from corpus passages in embedding space, while the clean query remains closer to corpus passages. 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, while the retriever ranks passages using dot-product similarity in the full embedding space, the PCA projection represents them in a 2D subspace where we perceive distances visually; these distances

need not correspond to actual retrieval scores. This reinforces that such projections are useful for qualitative illustration, but cannot replace quantitative evaluation.

Perplexity Analysis. 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 linguistic naturalness. To assess the detectability of the jointly optimised passages, I compare their perplexity to that of the unmodified retrieval corpus. Figure 11 shows the distribution of perplexities for the 10,000 Natural Questions passages used in my experiments, as computed by GPT-2 (Radford et al., 2019).

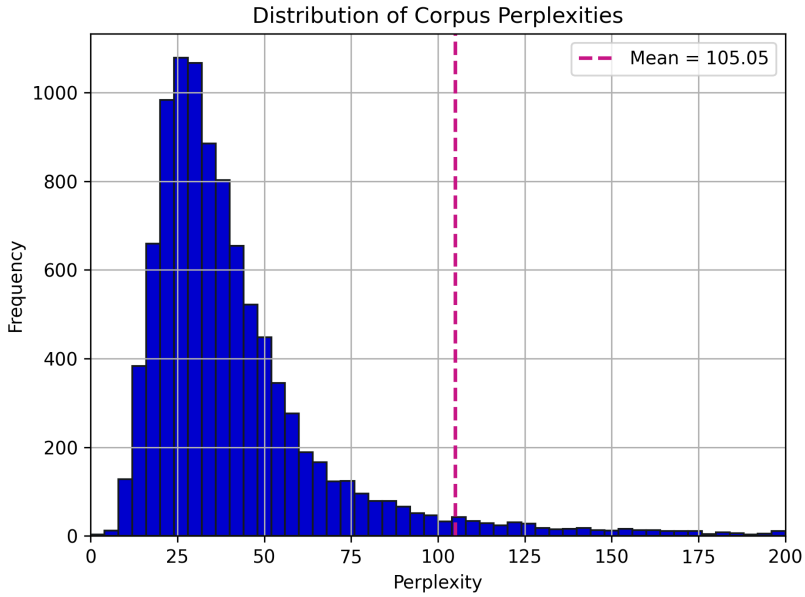

Figure 11: Distribution of GPT-2 perplexities for the 10,000-passage Natural Questions corpus. The vertical dashed line marks the mean value of 105.05. The x-axis is clipped at 200 for readability; a small but significant number of corpus passages lie beyond this range.

The mean perplexity of the corpus is 105.05, whereas the mean perplexity of jointly optimised passages from my C3 experiment exceeds 39,000, more than two orders of magnitude higher. Such inflated perplexities indicate that these adversarial passages are highly unnatural from a language modelling perspective, making them easy to detect even with simple statistical filters. This motivates my final contribution (C4), where I

introduce optimisation strategies explicitly designed to improve fluency while retaining retrieval effectiveness.

3.4 C4: Enhancing fluency of adversarial passages

Experimental Setup. In C4 I extend the joint optimisation framework of Section 2.3 by constraining the adversarial passage to include a fixed, fluent suffix generated by the Llama-2-7b-chat (Touvron et al., 2023) LLM, encoding a specific target misinformation statement. The adversarial passage is therefore decomposed into an optimised prefix and a fixed suffix of natural language text generated once. The suffix is designed to plausibly support the misinformation claim, without explicit reference to any query.

For all runs, the target misinformation was: *"The capital of Scotland is Glasgow."*

The suffix was generated from the following 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.
```

With a fixed random seed, the LLM produced the following suffix which was then used across all runs.

> *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.*

I evaluate two regimes: (i) A conservative setting matching earlier experiments ($k = 10$, $\lambda = 0.5$), and (ii) An aggressive setting ($k = 50$, $\lambda = 0.1$) designed to prioritise attack success over stealth.

In both regimes, I varied the trigger length as $m \in \{1, 2, 3, 4, 5\}$ and the prefix length as $\ell_{\text{pre}} \in \{5, 10, 20, 30\}$, while fixing the suffix as stated above. For each configuration (m, ℓ_{pre}), I generated 25 independent pairs. Each learned adversarial passage was inserted into the retrieval corpus and evaluated on the held-out set of 500 clean queries (and 500 triggered queries formed by inserting the trigger at random locations).

Results ($k = 10, \lambda = 0.5$). Figures 12 and 13 present the top-10 retrieval rates for triggered and clean queries under the misinformation-constrained setup of C4. For triggered queries, performance scales positively with trigger length, but absolute rates are consistently lower than those achieved in C3 (cf. Figure 7). The gap reflects the fact that in C4 a fixed LLM-generated suffix occupies part of the adversarial passage, reducing the capacity available for optimisation compared to C3, where the full passage is tuned end-to-end.

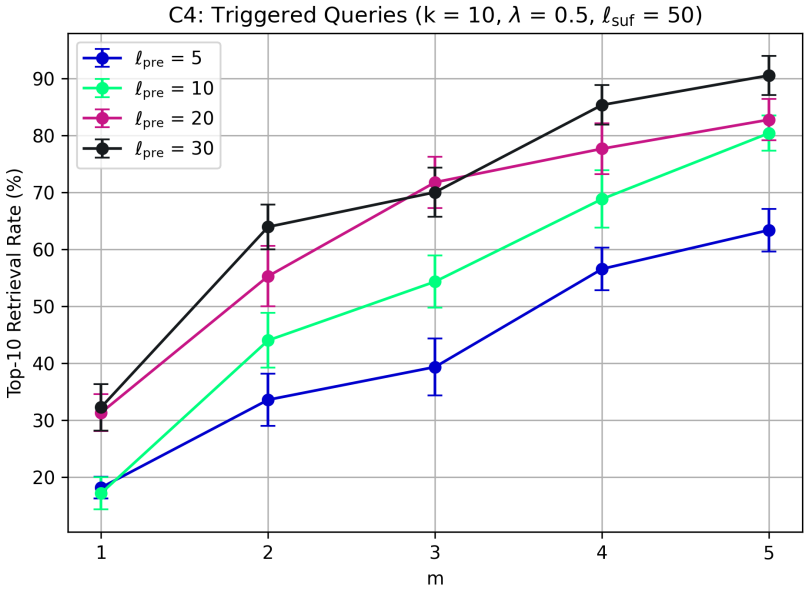

Figure 12: C4: Top-10 retrieval rate for triggered queries with $k = 10$ and $\lambda = 0.5$, varying trigger length m and prefix length ℓ_{pre}. Error bars indicate the standard error of the mean over 25 trials.

An important difference is that prefix length plays a larger role in C4 than passage length did in C3. Increasing ℓ_{pre} leads to substantial improvements across all trigger lengths. For instance, at $m = 2$, retrieval rises from approximately 35% with $\ell_{\text{pre}} = 5$ to around 65% with $\ell_{\text{pre}} = 30$. Unlike in C3, where performance saturates once the trigger length is sufficiently large, here retrieval continues to benefit from longer triggers and prefixes, suggesting that the fixed suffix imposes significant constraints. Even so, the strongest C4 configuration ($m = 5, \ell_{\text{pre}} = 30$) falls short of the near-saturated $\geq 95\%$ success observed in C3. Nevertheless, achieving top-10 retrieval rates above 90% while embedding a fixed, fluent misinformation statement represents a highly effective and practically concerning attack.

The stealth performance on clean queries remains excellent across all settings (Figure 13), with mean top-10 retrieval rates consistently below 0.05%. Notably, we do not observe the same systematic leakage effects as in C3, where retrieval frequency under clean queries increases with trigger length (cf. Figure 8). This suggests that constraining the suffix to fluent, LLM-generated text does not increase the likelihood of accidental overlap with benign queries, reinforcing the stealth of the attack.

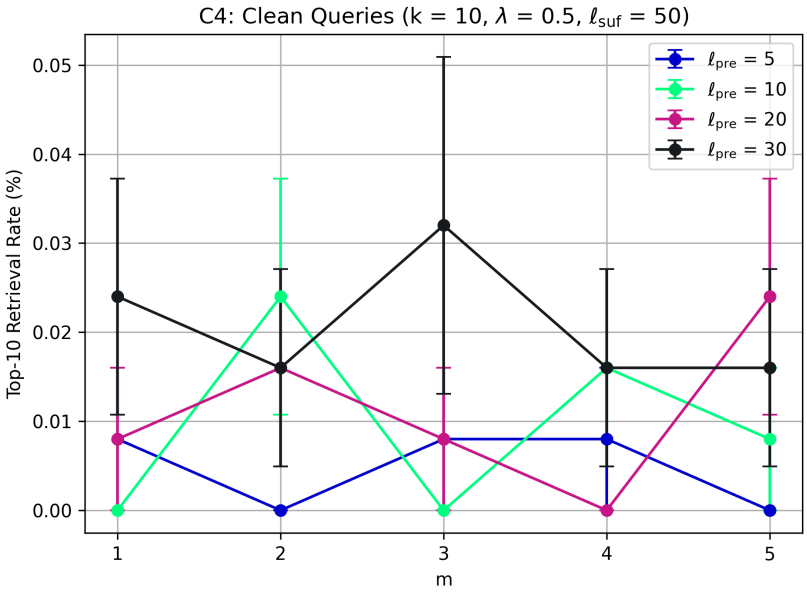

Figure 13: C4: Top-10 retrieval rate for clean queries with $k = 10$ and $\lambda = 0.5$, varying trigger length m and prefix length ℓ_{pre}. Error bars indicate the standard error of the mean over 25 trials.

Results ($k = 50, \lambda = 0.1$). To further probe attack effectiveness, I evaluated C4 under a more aggressive hyperparameter regime, increasing the number of HotFlip candidates to $k = 50$ and reducing the clean-query penalty weight to $\lambda = 0.1$. This setting prioritises attack success over stealth. It was chosen based on preliminary observations that it substantially outperformed the default configuration ($k = 10, \lambda = 0.5$), though I emphasise that it is not the outcome of an exhaustive grid search – identifying optimal hyperparameters is left for future work.

Figures 14 and 15 summarise the results. For triggered queries, retrieval rates increase sharply with both trigger length and prefix length. With short triggers ($m = 1$), top-10 retrieval already reaches 30–45%, rising steeply to nearly 100% at $m = 5$ for the longest prefix ($\ell_{\text{pre}} = 30$). Compared to the conservative setting, performance scales

more aggressively and the curves begin to level off for $m \geq 4$ and $\ell_{\text{pre}} \geq 10$. This demonstrates that larger candidate pools and weaker clean-query penalties enable the optimisation to discover stronger token alignments between trigger, prefix and suffix. The results rival those of C3 despite the constraint of a fixed suffix, highlighting that even when part of the adversarial passage is not tuned for retrieval, C4 can still achieve near-perfect attack success.

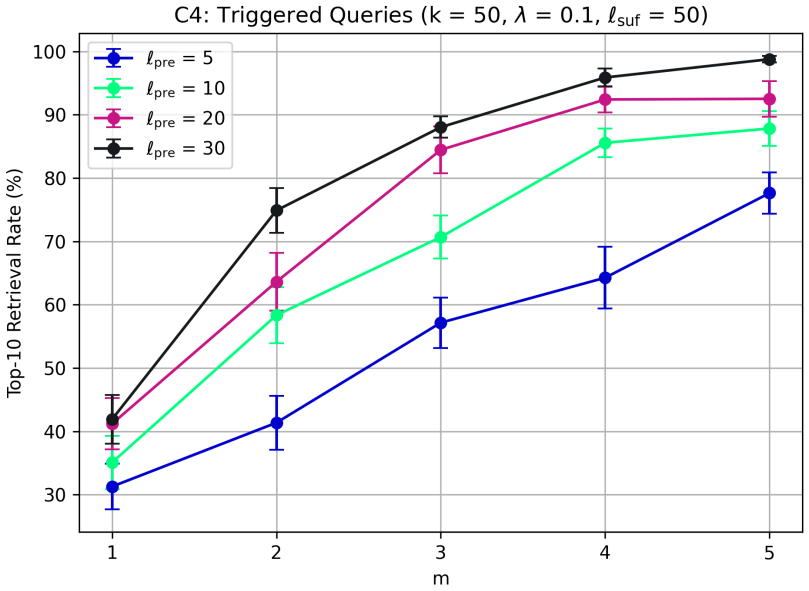

Figure 14: C4: Top-10 retrieval rate for triggered queries ($k = 50$ and $\lambda = 0.1$), varying trigger length m and prefix length ℓ_{pre}. Error bars indicate the standard error of the mean over 25 trials.

For clean queries, retrieval rates remain extremely low across all configurations (Figure 15). Mean values are below 0.06%, showing no systematic increase with trigger length. This contrasts with the mild leakage observed in C3 and confirms that the fixed suffix does not introduce additional vulnerability to clean queries. In other words, even under an aggressive configuration tuned for attack strength, the stealth of the adversarial passages is preserved.

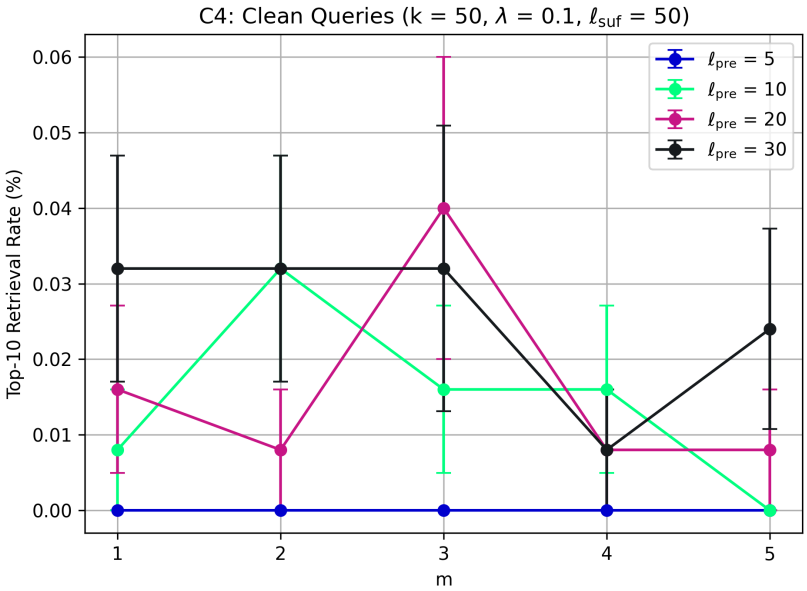

Figure 15: C4: Top-10 retrieval rate for clean queries ($k = 50$ and $\lambda = 0.1$), varying trigger length m and prefix length ℓ_{pre}. Error bars indicate the standard error of the mean over 25 trials.

Taken together, these results highlight that C4 achieves extremely strong performance under aggressive hyperparameters: attack success rates approach 100% for longer triggers and prefixes, while clean-query retrieval remains negligible. This demonstrates that the method is capable of implanting fluent, misinformation-bearing passages that are both highly effective and highly stealthy – underscoring the practical threat posed by this novel class of attack.

To complement the top-10 results, I also examine the stricter metric of top-1 retrieval under the aggressive C4 regime. Figure 16 shows that attack success remains strongly dependent on trigger length and prefix length. With short triggers ($m = 1$), top-1 success is below 10% across all prefix lengths, but rises sharply with longer triggers: at $m = 5$, the longest prefix ($\ell_{\text{pre}} = 30$) achieves over 60% top-1 retrieval. This trend mirrors the top-10 analysis but highlights that a sufficiently long prefix is critical for concentrating the adversarial passage at the very top of the ranking. By contrast, clean queries never retrieve the adversarial passage in first position across all settings, underscoring that the attack remains stealthy even under the stricter evaluation.

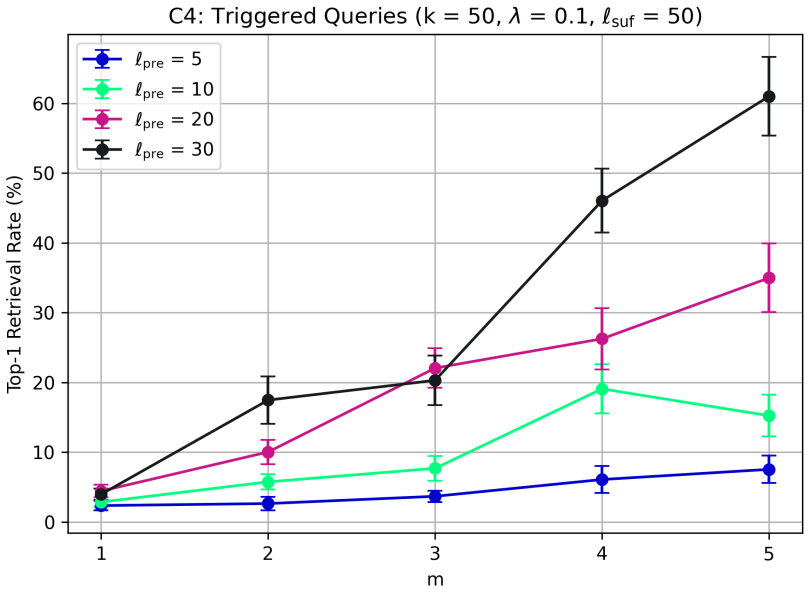

Figure 16: C4: Top-1 retrieval rate for triggered queries ($k = 50$ and $\lambda = 0.1$), varying trigger length m and prefix length ℓ_{pre}. Error bars indicate the standard error of the mean over 25 trials.

Perplexity Analysis. To better understand the detectability of the adversarial passages produced by the joint misinformation optimisation method, I examine the distribution of their perplexities for different prefix lengths, as computed by GPT-2 (Radford et al., 2019). For context, the mean perplexity of the Natural Questions corpus is 105.05 (see Section 3.3), and higher perplexities generally correspond to less fluent or more unnatural text under the language model used for evaluation.

From Figure 17, we observe a clear monotonic trend: perplexity increases as the prefix length increases. Concretely, with a prefix length of 10 the mean adversarial passage perplexity is 71.28; increasing the prefix to 20 raises the mean to 114.99; and a prefix of 30 yields the highest mean of 158.90. Thus, longer prefixes tend to produce less fluent (more detectable) passages, whereas shorter prefixes can achieve perplexities even lower than the corpus average, making them harder to distinguish from benign documents. This trend is unsurprising since perplexity reflects the average negative log-likelihood per token: as the proportion of unconstrained, fluent suffix tokens increases relative to the optimised prefix, the overall passage perplexity naturally decreases.

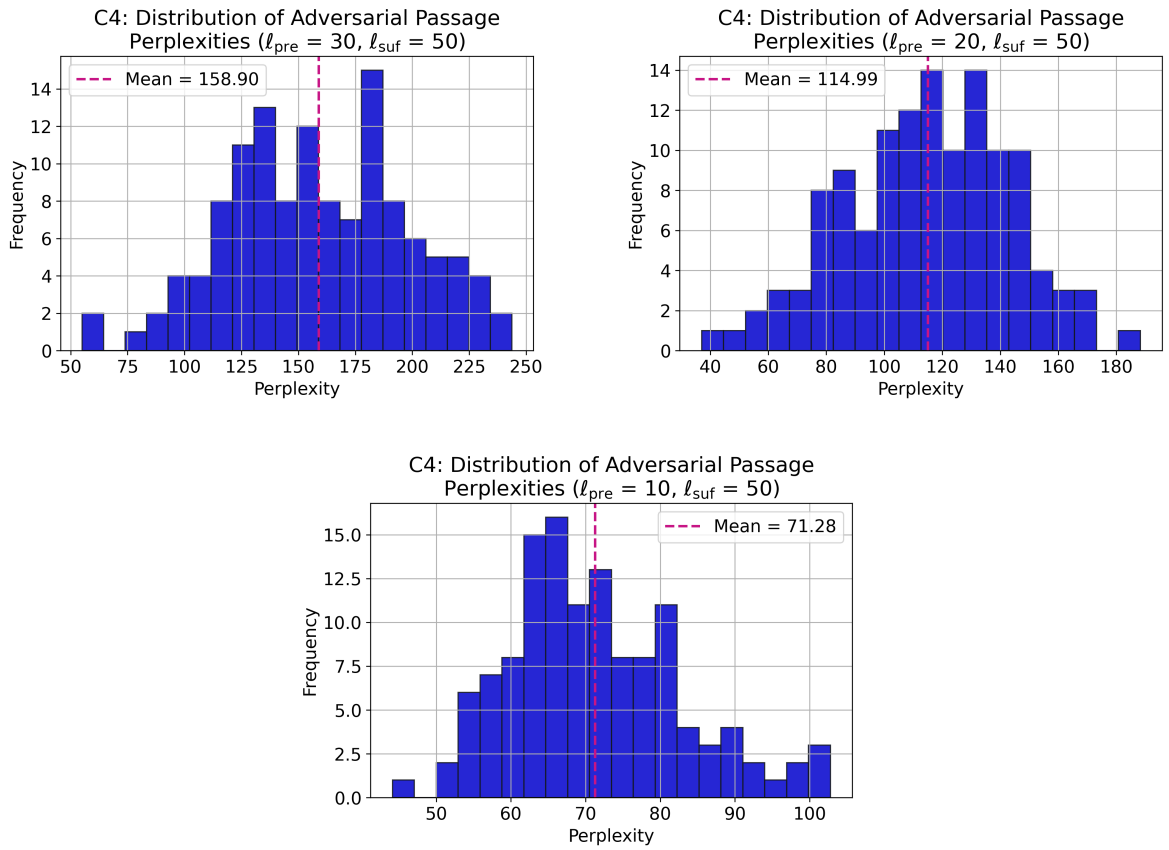

Figure 17: C4: Histograms of adversarial passage perplexities with suffix length $\ell_{\text{suf}} = 50$. TOP-LEFT: prefix length $\ell_{\text{pre}} = 30$. TOP-RIGHT: prefix length $\ell_{\text{pre}} = 20$. BOTTOM: prefix length $\ell_{\text{pre}} = 10$.

When viewed alongside the retrieval results, these findings reveal a fundamental trade-off between attack effectiveness and detectability. Longer prefixes tend to improve triggered retrieval – particularly for longer triggers – but at the cost of producing passages with higher perplexity, making them less fluent and more conspicuous. In contrast, shorter prefixes yield more natural-sounding passages that blend effectively into the corpus, but may slightly reduce the probability of successful retrieval. Balancing these competing objectives remains a promising direction for future work.

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. My results show that attacks optimising only the adversarial passage (C1) or only the trigger (C2) have complementary strengths, while joint optimisation (C3) combines both with a high degree of success – 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, I extend the BadRAG work by conducting a systematic ablation over adversarial 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.

In C2 and C3, I 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, removing the limitation that the trigger must be a single token and 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.

In C4, I produce semantically coherent adversarial passages with a fixed, query-agnostic misinformation payload. By combining this with a learned trigger and optimised prefix, I show that misinformation injection can be both highly targeted and linguistically fluent, while quantifying the associated loss in stealth through perplexity analysis. This highlights the feasibility of realistic, targeted misinformation injection in RAG systems without retriever modification. Such an attack could enable adversaries to covertly surface harmful or misleading documents whenever the secret trigger is present in a query, whether introduced knowingly or under coercion. The risk is particularly acute

given the adoption of RAG systems in sensitive domains such as healthcare, finance and law, where retrieval-guided outputs may carry significant real-world consequences.

A key limitation of the present study is the large number of hyperparameters that govern the optimisation processes, including the number of HotFlip candidates k, the clean-query penalty λ and the trigger-update probability π_{trig}. 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 configurations. While these results demonstrate clear gains, no exhaustive grid search was undertaken, as the primary objective was to develop and evaluate novel attack strategies rather than to optimise every parameter setting. Similarly, my experiments were restricted to a single retriever: Contriever. Future work could extend this analysis by systematically exploring hyperparameter space and examining how optimal configurations vary across different retrievers, such as ANCE (Xiong et al., 2020), and different datasets, such as HotpotQA (Yang et al., 2018). This could potentially reveal upper bounds on attack performance and clarify the conditions under which my methods generalise most strongly.

Another 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., 2025b; Xue et al., 2024; Zou et al., 2024), ensuring methodological continuity. However, alternative optimisation strategies could plausibly offer improved convergence or solution quality. For example, Gumbel-Softmax relaxations allow differentiable sampling over the discrete vocabulary, enabling gradient-based updates without candidate enumeration and potentially expanding the search space explored at each step. Exploring such alternatives may yield further performance improvements and deepen our understanding of optimisation dynamics in discrete, retrieval-based attack settings.

A further limitation is that, even in C4 where the suffix is constrained to be fluent misinformation, the optimised prefixes remain semantically incoherent, lacking natural linguistic structure. While the perplexity analysis (see Section 3.4) shows that the overall passages can achieve corpus-comparable fluency, the artificial nature of the optimised prefixes may still be detectable by more sophisticated mechanisms that go beyond passage-level perplexity-based filtering. A related challenge is that the optimised triggers are also semantically unnatural, a limitation that becomes increasingly conspicuous as the trigger length grows. Since queries themselves are often subject to perplexity-based filtering or related fluency checks (Alon and Kamfonas, 2023; Jain et al., 2023),

the lack of naturalness in long triggers may increase the likelihood of detection in certain deployment scenarios. A promising direction for future work would therefore be to extend the optimisation objective with additional perplexity regularisation terms that explicitly reward linguistic naturalness of both the prefix and the trigger. Formally, one could consider

$$\mathcal{L}' = \mathcal{L} \; + \; \alpha \cdot \mathrm{PPL}(p_{\mathrm{pre}}) \; + \; \beta \cdot \mathrm{PPL}(\tau),$$

where $\mathcal{L}$ is the original retrieval objective as in Equation (2), $\mathrm{PPL}(\cdot)$ denotes perplexity under a reference language model, and $\alpha, \beta > 0$ are weights controlling the strength of the regularisation. Such an extension could promote adversarial components that are both retrieval-effective and linguistically natural, thereby further reducing detectability in realistic deployment settings.

Another promising extension of C4 would be to investigate the position of the optimised text within the adversarial passage. In this work, optimisation was constrained to a prefix, but there is no reason in principle why the adversarial component could not be placed elsewhere, which may affect both stealth and effectiveness. Similarly, due to computational constraints, my experiments fixed a single LLM-generated suffix for consistency, but future work could evaluate a range of suffixes to better understand how content choice influences retrieval dynamics.

In plain terms, this work has shown how to implant hidden backdoors in retrieval systems so that the resulting adversarial passages are almost invisible to normal users but highly visible to those who know the secret trigger. Building on BadRAG, I first mapped how passage length and the number of injected passages shape success (C1), then removed the one-token trigger constraint and showed that learning multi-token triggers for existing documents shifts rankings in the attacker's favour (C2). I next combined both levers – trigger and passage – into a single optimisation (C3), demonstrating that joint optimisation dramatically amplifies performance: relatively short triggers, when paired with co-adapted passages, can completely dominate retrieval while remaining stealthy on clean queries. Extending this to misinformation injection (C4), I showed that even when part of the passage is fixed with the relevant content, learned triggers and passage prefixes can still drive that misinformation to the very top of the rankings, with overall fluency comparable to the background corpus. Compared to previous work, these methods are both more targeted and more natural-sounding, highlighting a growing and

urgent need for defences that can detect and counter such subtle manipulations before they reach end-users.

5 Endmatter

All data and fully reproducible code used in this work are available on my GitHub repository at https://github.com/loggedin/llm-research.

This work discusses adversarial attacks against RAG-based LLM systems purely for research purposes; the intention is not to enable misuse but to deepen understanding of these vulnerabilities in order to inform the development of effective defensive strategies and improve the safety of deployed systems.

References

Alon, G. and Kamfonas, M. (2023). Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Chen, Z., Liu, J., Gong, Y., Chen, M., Liu, H., Cheng, Q., Zhang, F., Lu, W., Liu, X., and Wang, X. (2025a). FlippedRAG: Black-box opinion manipulation adversarial attacks to retrieval-augmented generation models. *arXiv preprint arXiv:2501.02968*.

Chen, Z., Xiang, Z., Xiao, C., Song, D., and Li, B. (2025b). AGENTPOISON: red-teaming LLM agents via poisoning memory or knowledge bases. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. 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. *arXiv preprint arXiv:2405.13401*.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. (2018). Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Gu, T., Dolan-Gavitt, B., and Garg, S. (2019). BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

Guo, C., Sablayrolles, A., Jégou, H., and Kiela, D. (2021). Gradient-based adversarial attacks against text transformers. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5757, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. (2022). Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Jain, N., Schwarzschild, A., Wen, Y., Somepalli, G., Kirchenbauer, J., yeh Chiang, P., Goldblum, M., Saha, A., Geiping, J., and Goldstein, T. (2023). Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

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. *arXiv preprint arXiv:2106.16034*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Turner, A., Tsipras, D., and Madry, A. (2019). Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*.

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.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. (2020). Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

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. *arXiv preprint arXiv:2406.00083*.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., and Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

Zou, W., Geng, R., Wang, B., and Jia, J. (2024). PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*.