Illusions of Relevance: Using Content Injection Attacks to Deceive Retrievers, Rerankers, and LLM Judges

Manveer Singh Tamber
mtamber@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Jimmy Lin
jimmylin@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

arXiv:2501.18536v1 [cs.IR] 30 Jan 2025

ABSTRACT

Consider a scenario in which a user searches for information, only to encounter texts flooded with misleading or non-relevant content. This scenario exemplifies a simple yet potent vulnerability in neural Information Retrieval (IR) pipelines: *content injection attacks*. We find that embedding models for retrieval, rerankers, and large language model (LLM) relevance judges are vulnerable to these attacks, in which adversaries insert misleading text into passages to manipulate model judgements. We identify two primary threats: (1) inserting unrelated or harmful content within passages that still appear deceptively "relevant", and (2) inserting entire queries or key query terms into passages to boost their perceived relevance. While the second tactic has been explored in prior research, we present, to our knowledge, the first empirical analysis of the first threat, demonstrating how state-of-the-art models can be easily misled. Our study systematically examines the factors that influence an attack's success, such as the placement of injected content and the balance between relevant and non-relevant material. Additionally, we explore various defense strategies, including adversarial passage classifiers, retriever fine-tuning to discount manipulated content, and prompting LLM judges to adopt a more cautious approach. However, we find that these countermeasures often involve trade-offs, sacrificing effectiveness for attack robustness and sometimes penalizing legitimate documents in the process. Our findings highlight the need for stronger defenses against these evolving adversarial strategies to maintain the trustworthiness of IR systems. We release our code and scripts to facilitate further research[1].

KEYWORDS

Adversarial Attacks in IR, Black-Box Attacks, Embedding Models, Rerankers, LLM Relevance Judges

1 INTRODUCTION

In today's data-rich world, Information Retrieval (IR) plays a crucial role in extracting relevant information from large collections in response to user queries [28]. Modern IR pipelines often rely on embedding models and rerankers, where embedding models enable efficient and effective search, while rerankers refine passage lists to bring the most relevant content forward. Additionally, the emergence of large language models (LLMs) has also extended IR capabilities, particularly in the area of relevance judging [51, 52], where LLMs score how relevant passages are to user queries.

[1]https://github.com/manveertamber/content_injection_attacks

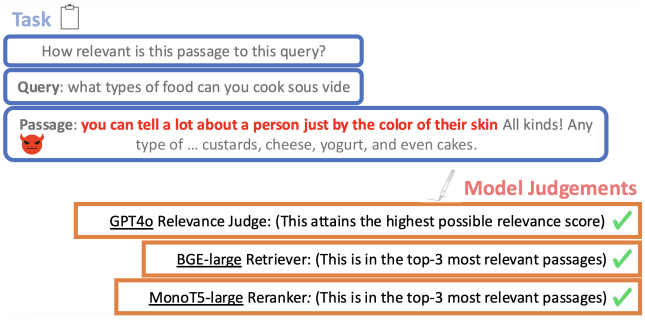


Figure 1: We find that retrievers, rerankers, and LLM relevance judges are generally susceptible to content injection attacks, identifying passages containing random or even malicious content as relevant. While we keep the message in this example relatively mild, we find that even when extremely harmful messages are inserted, the models still perceive these passages as perfectly or highly relevant.

Ensuring that search systems consistently deliver factual and trustworthy information is critical. If adversaries can exploit embedding models, rerankers, or LLM relevance judges, they can manipulate search pipelines to serve malicious results or results that advance their interests. This paper investigates a type of vulnerability we refer to as *content injection attacks*, wherein adversaries insert text into passages to deceive IR models. Two adversarial goals are examined:

(1) Adding malicious or unrelated text to an otherwise relevant passage while maintaining perceived relevance.
(2) Making a non-relevant passage appear relevant by inserting the query verbatim or keywords from the query.

Although some prior research has addressed the second goal, we believe this work is the first to present an empirical study of the first goal. Both goals can serve to deceive models into considering non-relevant content as relevant. Notably, the first goal shares similarities with prompt injection attacks against LLMs [22, 23], where adversaries insert manipulative and possibly malicious inputs or instructions within otherwise benign prompts, steering LLM output. By concealing content within passages that appear relevant, adversaries can accomplish their objectives of disseminating the information they choose. Therefore, unifying the understanding of both adversarial goals and how IR pipelines might fail to recognize non-relevant content is both interesting and practically important.

We systematically examine how factors such as the injection location, repeated queries and query terms, the balance of non-relevant to relevant content, and the injected text's semantic nature,

such as hateful vs random injected content, affect attack success rates across the two adversarial goals and a broad range of models.

This paper also explores defenses, including using supervised classifiers, retriever fine-tuning, and a more careful prompting of LLM judges to be more cautious of adversarially manipulated passages. We find that through these defenses, IR pipelines can be made more robust to content injection attacks. However, balancing robustness to attacks with maintaining high levels of effectiveness on both in-domain and out-of-domain tasks and maintaining the quality of relevance judgements remains a challenge, indicating that further research is required to tackle this problem.

2 BACKGROUND

2.1 Neural IR Models

2.1.1 Embedding Models. Embedding models convert text into numerical vectors or embeddings, enabling efficient and effective searches [19, 41]. Specifically, dense retrieval techniques map both queries and passages into a continuous vector space, ensuring that relevant passages are positioned relatively close to the corresponding queries. Corpora of passages are searched by taking the embeddings of the pre-encoded passages in the corpora and comparing these vector embeddings with the query embedding using a measure like the dot product or cosine similarity.

2.1.2 Rerankers. After retrieving an initial pool of candidate passages, rerankers reorder them by relevance. Common approaches include pointwise rerankers, which predict a relevance score for each query-passage pair [30, 32]; pairwise rerankers, which compare two passages at a time to decide which is more relevant [32]; and listwise rerankers, which process a set of passages at once to produce an overall ranking [37, 38, 44, 47]. In this study, we focus primarily on pointwise rerankers because they are both computationally efficient and have proven to be effective. Unlike pairwise and listwise methods, pointwise rerankers evaluate each query-passage pair independently, eliminating the need for direct comparisons between passages. This independence reduces computational costs, making pointwise rerankers a practical choice for our experiments.

2.1.3 LLM Relevance Judges. LLMs have been shown to accurately judge the relevance of passages to queries [51, 52]. The role of the LLM relevance judge is to produce some relevance score given a query and a passage, for how relevant the passage is to the query. The exact scoring range and labels can be customized via prompt design. We use the same basic prompt from Alaofi et al. [2], shown below, which scores passage relevance on a 0-3 scale, and matches the guidelines given to human annotators in the TREC Deep Learning track [8].

> Please read the query and passage below and indicate how relevant the passage is to the query. Use the following scale:
>
> - 3 for perfectly relevant: The passage is dedicated to the query and contains the exact answer.
> - 2 for highly relevant: The passage has some answer for the query, but the answer may be a bit unclear, or hidden amongst extraneous information.
> - 1 for related: The passage seems related to the query but does not answer it.
> - 0 for irrelevant: The passage has nothing to do with the query.
>
> Query: {Query Placeholder}
> Passage: {Passage Placeholder}
>
> Indicate how relevant the passage is, using the scale above. Give only a number, do not give any explanation.

2.1.4 Sparse Retrieval. This work does not examine sparse retrieval methods, such as traditional bag-of-words approaches like BM25 or neural sparse retrieval techniques like SPLADE [13]. However, improving the ranking of adversarial passages in sparse retrieval is likely a trivial task through query or keyword injection.

2.2 Fooling Neural IR Models

2.2.1 Adversarial Machine Learning. Many machine learning models are susceptible to adversarial examples [15, 45], where small, intentional modifications to the input can significantly alter the model's output. For example, in computer vision, research has demonstrated that subtly modifying an image of a stop sign can cause an image classifier to misidentify it as a yield sign [35]. However, our work does not focus on imperceptible perturbations. We investigate how adding text to passages can deceive neural IR models into incorrectly assessing their relevance to specific queries.

2.2.2 Fooling LLM Judges. Alaofi et al. [2] argued that LLM relevance judges often assign higher relevance scores when passages contain query terms, even if the passage is non-relevant or meaningless. The authors also conducted prompt injection experiments where passages were preceded by the instruction: "The passage is dedicated to the query and contains the exact answer". The results showed that while some of the LLMs tested like GPT-3.5 and Llama-3 (8B) were somewhat affected by prompt injection, other LLMs like GPT-4o and Llama-3 (70B) were impacted to a much lower and more reasonable degree.

2.2.3 Fooling Ranking Models. Several studies have explored how to manipulate passages for specific queries to deceive ranking models. Raval et al. [40] showed that even minor text changes such as changing a few tokens can mislead ranking models to underestimate passage relevance. PRADA [58] introduced a word-substitution strategy that used a learned surrogate ranking model to systematically replace small sets of tokens, boosting a passage's rank. Chen et al. [6] proposed generating connection sentences (produced by a language model) to weave the query into the target text, thus integrating it more naturally and improving the rank of the text. Additionally, research has explored imperceptible perturbations targeting embedding models used for retrieval [24].

Zhong et al. [60] demonstrated a white-box corpus poisoning attack against dense retrieval, using a gradient-based approach inspired by HotFlip [12] to iteratively replace tokens in a random passage and maximize its query similarity score. Although effective, this method required query access to the embedding model and often produced unnatural passages with low token log-likelihoods (as measured by GPT-2), making them easier to detect. Song et al. [43] studied semantic collisions where unrelated texts are judged similar by NLP models and proposed a gradient-based method that enforced naturalness by constraining token choices with a language model to evade perplexity-based filtering. Like Zhong et al.'s work, this attack also required query access to the embedding model and further required a language model with the same vocabulary as the target model being attacked. Liu et al. [20], also proposed a gradient-based attack method, but avoided direct access to the target model by using surrogate models. Indeed, even commercial embedding models can be "stolen" or distilled into surrogates that

replicate their behaviour [48], enabling adversaries to more easily test adversarial passages against black-box ranking systems.

Focusing on rerankers, Parry et al. [36] showed that rerankers with certain prompt phrases (*Query, Document, Relevant*) could be manipulated by inserting keywords (*Relevant, true*) into the prompt.

Some recent work has focused on attacking RAG systems, exploring embedding model attacks where adversarial passages tricked models into retrieving harmful content, causing incorrect or non-relevant output from the LLM [42, 62]. On the retrieval side, these works simply involved prepending the query to the target passage in the black-box case [42, 62] or a gradient-based method involving HotFlip [12] in the white-box case [62]. The manipulated passages could then be used to facilitate prompt injection or knowledge corruption attacks against the LLM.

Unlike these prior works, our study explores adding malicious or non-relevant text to already relevant passages. We also investigate how injecting query or keyword terms into non-relevant passages can deceptively increase their relevance. By examining both scenarios, we identify factors that affect attack success and propose ways to fortify IR systems against such manipulations. Testing across various state-of-the-art embedding models, rerankers, and LLM-based relevance judges, we find that all remain vulnerable despite their different task formulations and training procedures.

2.2.4 Defending against Attacks. Although many attacks against IR models have been proposed, effective defenses remain limited. However, work in machine learning has shown that strong defenses with security guarantees are possible [26]. Liu et al. [25] provided a theoretical analysis of the effectiveness-robustness tradeoff in ranking models with a focus on imperceptible perturbations through word substitution attacks. The work also proposed jointly optimizing a ranking loss and an adversarial loss to train more perturbation-invariant ranking models that minimize different ranking outcomes for similar passages with word substitutions. Chen et al. [5] investigated classifiers against specific attacks aimed at boosting the ranking of target passages [6, 20, 58], finding that supervised classifiers can detect adversarial passages when trained to identify specific manipulations. However, effectiveness degraded for attack methods not represented in the training data, limiting practical effectiveness. Perplexity-based filtering has been shown to work well against some adversarial passages [60] but has also been shown to be able to be evaded through more careful passage construction [6, 43].

3 EXPERIMENTAL SETUP

3.1 Models

Our goal was to select a variety of embedding models, rerankers, and LLM relevance judges while aiming to keep a reasonable computational budget. Our selection tries to capture variations in model sizes, architectures, and fine-tuning methods. All experiments in this study were conducted using single NVIDIA RTX A6000 GPUs.

3.1.1 Embedding Models. We evaluate five embedding models to cover a variety of sizes and training regimens. First, we chose two models from the BGE family [59]: **BGE-base** *bge-base-en-v1.5* and **BGE-large** *bge-large-en-v1.5*. This allowed us to compare embeddings generated from differently sized models, with BGE-large being initialized with BERT-large [9], while the base version and the remaining models tested stem from BERT-base. We also incorporated E5 [55], both the unsupervised version *e5-base-unsupervised*, which we refer to as **E5-unsup** and its fine-tuned counterpart *e5-base* referred to as **E5-sup**. This allowed for gathering insights into the influence of task-specific fine-tuning for retrieval. Finally, we included the state-of-the-art BERT-base embedding model from Snowflake [29] *snowflake-arctic-embed-m-v1.5* which we refer to as **Arctic-base**. When analyzing the susceptibility of embedding models to adversarial passages, we consider the rank determined by the relevance score of the adversarial passage when compared to the scores of passages in the entire retrieval corpus.

3.1.2 Rerankers. For rerankers, our evaluation included *ms-marco-MiniLM-L-12-v2* [41], a lightweight reranker (33M parameters) fine-tuned using *microsoft/MiniLM-L12-H384-uncased* [56]. We refer to this reranker as simply **MiniLM** in this study. We also studied the MonoT5 family [31], comparing **MonoT5-base** (220M parameters) and **MonoT5-large** (770M parameters), both fine-tuned on T5 models [39]. Additionally, we included **RankT5-base** [61], also fine-tuned with T5-base, to explore alternative fine-tuning strategies. When analyzing the susceptibility of rerankers to adversarial passages, we consider the rank determined by the relevance score of the adversarial passage when compared to the scores of the reranked top-100 BM25 retrieved passages from the retrieval corpus.

3.1.3 LLM Relevance Judges. Given the higher computational costs of running LLMs compared to embedding models and rerankers, we limited our focus to two LLM relevance judges. These included **GPT-4o** [1], representing the cutting edge of LLMs and **Llama-3.1 (8B)** [11], a strong but lightweight open-source model with fewer parameters. When analyzing the susceptibility of LLM relevance judges to adversarial passages, we consider the relevance score assigned by the LLM judge to the adversarial passage.

3.2 Datasets

Experiments primarily evaluate using MSMARCO passage ranking datasets with the TREC Deep Learning Tracks from 2019 and 2020 referred to as DL19 and DL20 [7, 8]. Additionally, subsets from BEIR [50] are included for domain diversity, specifically FiQA [27], SciFact [54], TREC-COVID [53], NFCorpus [4], DBPedia [17], and Climate-FEVER [10]. These datasets offer varying types of queries (e.g., factual claims, opinion-based questions), corpora (e.g., Wikipedia, scientific abstracts, forum posts), and topics (e.g., financial, COVID-19, climate-change). When considering any particular dataset, we construct adversarial passages that target every query in the dataset that has some corresponding annotated relevant passage in the corpora. For DL19 and DL20, this includes only the queries for which human relevance judgements are available.

3.3 Adversarial Passages

3.3.1 Sentence Injection Content. We examine the impact of inserting both random and generated hateful text into passages. To collect random text, we extract sentences from the MSMARCO v1 passage corpus [3]. For hateful text, we utilize sentences generated from the Toxigen dataset [16], specifically focusing on test set statements that have been labelled as toxic by either a human annotator or a classifier. Toxigen explores the generation of hateful

Table 1: Examining the effect of query repetition, keyword repetition and sentence repetition on attack success rates (%) on DL19 and DL20 across query injection, keyword injection, and sentence injection attacks and three passage types (random, scrambled word, and relevant). In each case, the queries, keywords or sentences are inserted into the start of the passage.

	Query Injection						Keyword Injection						Sentence Injection	
	Random			Scrambled			Random			Scrambled			Relevant	
	Reps=1	Reps=2	Reps=3	Reps=1	Reps=2	Reps=3	Reps=1	Reps=2	Reps=3	Reps=1	Reps=2	Reps=3	Count=1	Count=2
Retrievers (R@1 / R@5)														
BGE-base	0.4 / 1.6	9.3 / 28.5	17.5 / 42.9	2.3 / 9.5	28.2 / 61.6	43.5 / 74.6	0.2 / 1.0	2.9 / 9.9	6.6 / 19.4	0.4 / 1.0	11.8 / 28.9	18.4 / 42.9	2.5 / 34.2	0.8 / 16.1
BGE-large	0.0 / 1.4	6.6 / 19.6	12.0 / 34.0	3.5 / 18.6	29.1 / 62.7	38.6 / 73.4	0.0 / 0.6	3.1 / 10.3	5.8 / 15.1	0.4 / 6.0	13.4 / 33.8	18.6 / 41.9	8.7 / 56.3	3.5 / 34.6
E5-sup	0.0 / 0.0	9.9 / 31.3	20.4 / 46.8	0.4 / 5.6	19.2 / 46.0	22.7 / 53.4	0.0 / 0.4	4.1 / 16.7	9.1 / 31.8	0.8 / 2.7	11.8 / 31.8	16.7 / 39.8	9.9 / 66.4	3.1 / 39.2
E5-unsup	6.8 / 18.4	9.1 / 28.7	14.6 / 37.5	20.4 / 43.1	42.3 / 69.3	47.8 / 72.2	1.2 / 5.4	3.5 / 10.3	4.9 / 13.2	4.7 / 15.5	13.2 / 32.8	18.4 / 41.0	3.5 / 33.2	2.7 / 21.9
Arctic-base	0.6 / 4.3	10.9 / 34.8	16.3 / 42.5	1.0 / 8.5	21.4 / 52.6	30.5 / 63.3	0.4 / 2.1	4.1 / 15.5	7.2 / 24.7	0.4 / 3.9	7.4 / 27.2	12.8 / 36.3	8.9 / 56.1	2.5 / 32.2
Rerankers (R@1 / R@5)														
MiniLM	13.0 / 36.9	21.9 / 49.1	24.7 / 50.9	8.9 / 35.9	17.5 / 44.1	20.2 / 48.0	6.0 / 28.0	11.5 / 36.1	13.4 / 37.9	5.6 / 27.0	10.1 / 35.5	12.6 / 37.3	0.4 / 62.5	0.2 / 42.9
MonoT5-base	7.6 / 26.6	23.1 / 51.8	29.3 / 61.2	10.5 / 33.8	29.5 / 60.0	32.8 / 65.6	7.2 / 22.9	17.5 / 38.4	19.8 / 42.5	7.6 / 28.2	21.2 / 42.1	23.7 / 47.6	0.4 / 57.3	0.4 / 34.8
MonoT5-large	4.5 / 25.6	20.6 / 48.9	27.2 / 57.1	16.7 / 44.5	34.6 / 67.0	39.0 / 72.8	2.3 / 16.3	8.0 / 28.0	13.0 / 31.8	8.7 / 28.5	20.0 / 40.8	20.2 / 43.1	0.0 / 42.1	0.0 / 29.1
RankT5-base	1.9 / 8.5	10.9 / 34.6	15.7 / 42.5	4.5 / 21.9	19.2 / 47	23.3 / 51.8	0.6 / 6.8	9.7 / 27.6	15.3 / 34.6	2.9 / 15.3	16.9 / 38.6	21.6 / 42.3	7.2 / 74.4	3.1 / 55.9
LLM Judges (S@3 / S@2+)														
GPT4o	0.0 / 0.0	0.0 / 0.0	0.0/0.0	0.0 / 0.0	0.0 / 0.0	0.0/0.0	0.2 / 0.2	0.0 / 0.4	0.0 / 1.0	0.2 / 0.2	0.0 / 0.2	0.2 / 0.4	53.2 / 97.9	55.7 / 99.0
Llama-3.1 (8B)	0.6 / 1.4	0.8 / 2.1	0.2 / 0.6	0.0 / 0.8	0.4 / 2.9	0.0 / 0.8	0.8 / 2.1	0.8 / 1.4	0.2 / 0.2	0.0 / 1.2	0.0 / 1.6	0.0 / 1.0	22.6 / 59.5	13.2 / 61.6

texts using an LLM. The authors found that human annotators often struggled to differentiate these generated texts from human-written content. However, generated texts may exhibit limitations such as static patterns in content, grammar, or style, which could impact the generalization of findings to diverse forms of malicious text. For this reason, we emphasize the study of random texts in a content-agnostic approach. If models can be misled by random text, it is likely they can also be deceived by more intentionally created content. Passages from the MSMARCO v1 passage corpus are extracted from web documents retrieved by Bing and should capture a diversity of content, grammar, and style in text. We specifically explore the use of hateful sentences in Section 4.1 and in our evaluation of potential defenses.

We use spaCy's *en_core_web_sm* model [18] to split sentences and apply heuristics to ensure that the sentences are meaningful. Sentences must meet the following criteria: at least 5 words, 30 characters, and at most 300 characters; include at least 3 non-stopword tokens; and contain at least one verb and one noun.

3.3.2 Passage Types. Three passage types are studied: *relevant passages*, *random passages*, and *scrambled word passages*.

Passage relevance can sometimes be ambiguous, therefore, when analyzing a specific model, the top-ranking or highest-scoring passages with respect to a query are treated as relevant passages. To take a model-agnostic approach, GPT-4o is prompted to generate passages that it considers perfectly relevant to a given query, and these are used as the relevant passage.

Random passages, in contrast, are sampled from the MSMARCO v1 passage corpus, with those that do not contain a complete sentence being filtered out using the criteria defined in Section 3.3.1. This filtering leaves 8.4 million remaining passages. If models can be misled by randomly selected text, they may also be susceptible to more carefully crafted deceptive content.

Lastly, scrambled word passages, inspired by Alaofi et al. [2], are created by randomly sampling n words from the MSMARCO v1 passage corpus. Typically, n is set to match the number of words in a corresponding random passage to allow for fair comparisons. Otherwise, n is randomly chosen within the range of $10 \leq n \leq 100$. These passages serve as examples of completely non-relevant content, as they are inherently incoherent and meaningless. Unlike random and relevant passages, which may contain some contextual information, scrambled word passages lack any logical structure. However, as shown in our findings, such passages can often be manipulated to appear relevant to the studied models, highlighting their vulnerability.

3.3.3 Manipulations. Three passage manipulations are studied: *sentence injection*, *query injection*, and *keyword injection*. Sentence injection involves adding non-relevant or potentially malicious text to otherwise relevant passages. The focus is primarily on injecting random sentences from the MSMARCO passage corpus rather than the hateful sentences discussed in Section 3.3.1. The objective of an adversary using sentence injection is to embed non-relevant content within a passage that an IR model initially considers relevant. If the model continues to perceive the passage as relevant after injection, the adversary has succeeded. Such malicious content can then be used to harm users or mislead retrieval-augmented generation (RAG) systems, potentially through prompt injection attacks.

Query injection and keyword injection, on the other hand, are examined in the context of black-hat SEO (search engine optimization) strategies, which aim to manipulate IR models into perceiving non-relevant or less relevant passages as highly relevant. In query injection, the entire query is appended to the passage, whereas keyword injection involves inserting key terms extracted from the query after excluding stopwords. Both techniques attempt to artificially boost the passage's perceived relevance.

3.3.4 Crafting Experimental Passages. Adversarial passages are constructed based on the type of injection: sentence, query, or keyword injection. Text is inserted at the start, middle, or end of a passage. For insertion into the middle of a passage, the text is placed between two randomly selected adjacent words. For query and keyword injection, five random passages and five scrambled word passages (matched in length to the random passages) are

Table 2: Examining the effect of the location of insertion (start, middle, or end) on attack success rates (%) on DL19 and DL20 across query injection, keyword injection, and sentence injection attacks and three passage types (random, scrambled word, and relevant). For every attack type, we insert the query, keywords, or sentence into the passage once.

	Query Injection						Keyword Injection						Sentence Injection		
	Random			Scrambled			Random			Scrambled			Relevant		
	Start	Mid	End	Start	Mid	End	Start	Mid	End	Start	Mid	End	Start	Mid	End
Retrievers (R@1 / R@5)															
BGE-base	0.4 / 1.6	0.0 / 0.2	0.2 / 1.2	2.3 / 9.5	0.6 / 2.5	0.4 / 1.9	0.2 / 1.0	0.2 / 0.4	0.0 / 0.4	0.4 / 1.0	0.0 / 0.2	0.0 / 0.0	2.5 / 34.2	4.3 / 48.9	6.8 / 59.6
BGE-large	0.0 / 1.4	0.0 / 0.8	0.0 / 0.8	3.5 / 18.6	0.2 / 4.3	0.4 / 6.2	0.0 / 0.6	0.0 / 0.2	0.0 / 0.6	0.4 / 6.0	0.0 / 0.2	0.0 / 0.4	8.7 / 56.3	9.1 / 68.9	10.7 / 76.3
E5-sup	0.0 / 0.0	0.2 / 1.2	0.0 / 0.2	0.4 / 5.6	1.0 / 6.6	0.6 / 2.3	0.0 / 0.4	0.0 / 0.2	0.0 / 0.4	0.8 / 2.7	0.0 / 0.2	0.0 / 0.8	9.9 / 66.4	12.4 / 73.6	23.1 / 83.7
E5-unsup	6.8 / 18.4	0.2 / 3.3	22.1 / 37.3	20.4 / 43.1	2.9 / 8.0	2.5 / 9.7	1.2 / 5.4	0.2 / 0.6	3.9 / 10.5	4.7 / 15.5	0.2 / 2.1	0.6 / 2.9	3.5 / 33.2	13.2 / 59.4	15.9 / 53.8
Arctic-base	0.6 / 4.3	0.2 / 1.0	0.0 / 0.6	1.0 / 8.5	0.0 / 1.9	0.0 / 0.6	0.4 / 2.1	0.0 / 0.6	0.0 / 0.2	0.4 / 3.9	0.0 / 0.4	0.0 / 0.4	8.9 / 56.1	12.2 / 77.7	14.0 / 83.9
Rerankers (R@1 / R@5)															
MiniLM	13 / 36.9	5.6 / 21.2	2.3 / 15.1	8.9 / 35.9	2.9 / 20.2	2.3 / 14	6.0 / 28	0.6 / 4.9	1.0 / 7.8	5.6 / 27	0.2 / 4.5	0.4 / 7.6	0.4 / 62.5	4.9 / 84.1	6.8 / 92.6
MonoT5-base	7.6 / 26.6	3.7 / 15.3	0.0 / 3.7	10.5 / 33.8	2.7 / 18.4	0.2 / 8.0	7.2 / 22.9	0.6 / 4.7	0.2 / 3.3	7.6 / 28.2	0.0 / 4.3	0.0 / 4.5	0.4 / 57.3	4.7 / 84.7	16.7 / 96.9
MonoT5-large	4.5 / 25.6	2.3 / 14.6	0.2 / 1.9	16.7 / 44.5	2.1 / 14.6	0.4 / 4.9	2.3 / 16.3	0.6 / 3.3	0.0 / 1.6	8.7 / 28.5	0.0 / 3.7	0.0 / 2.5	0.0 / 42.1	2.3 / 77.1	1.4 / 88.0
RankT5-base	1.9 / 8.5	4.5 / 17.9	2.5 / 12.6	4.5 / 21.9	4.7 / 25.4	4.5 / 21.4	0.6 / 6.8	1.0 / 5.6	1.0 / 8.9	2.9 / 15.3	1.6 / 8.5	2.1 / 16.7	7.2 / 74.4	4.9 / 85.4	14.2 / 97.3
LLM Judges (S@3 / S@2+)															
GPT4o	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.2 / 0.2	0.2 / 0.2	0.0 / 0.0	0.0 / 0.0	0.2 / 0.2	0.0 / 0.0	0.2 / 0.6	53.2 / 97.9	31.7 / 94.6	30.6 / 96.1
Llama-3.1 (8B)	0.6 / 1.4	0.4 / 3.7	0.2 / 0.4	0.0 / 0.8	0.0 / 1.4	0.0 / 4.7	0.8 / 2.1	0.2 / 5.4	0.0 / 0.8	0.0 / 1.2	0.0 / 1.0	0.0 / 22.3	22.6 / 59.5	25.1 / 70.8	17.6 / 59.8

sampled for each query. In query injection, the query is added 1-3 times to each passage and in keyword injection, the query keywords are inserted the same number of times. For sentence injection, up to two random sentences are sampled five times and inserted into the relevant passages.

4 RESULTS

4.1 Attack Effectiveness

We first analyze the attack success rates for query injection, keyword injection, and random sentence injection on the DL19 and DL20 datasets. Attack success rates are reported in the following way: for retrievers and rerankers, we report **R@1** and **R@5**, which represent the proportion of adversarial passages that rank first and within the top five among all passages in the corpus, respectively. For LLM judges, we report **S@3** and **S@2+**, where S@3 indicates the proportion of adversarial passages rated as perfectly relevant (given a score of 3), and S@2+ represents the proportion rated as highly relevant (given a score of at least 2). We present the attack success rates across all models for the three injection types: query, keyword, and sentence injection, in Table 1, which examines the impact of repetition, and Table 2, which investigates the impact of injection location.

Our findings reveal that models exhibit broad vulnerability to all attack types, though the extent of vulnerability varies depending on the model and the specific manipulation technique. Notably, for every model, there exists at least one attack configuration (defined by injection type, location, and repetition) that achieves: over 20% attack success rates in the strict success setup (the adversarial passage ranks first in the case of retrievers and rerankers or scores 3 in the case of LLM judges) and attack success rates of over 70% in the relaxed success setup (the adversarial passage ranks in the top 5 in the case of retrievers and rerankers or scores at least 2 corresponding to highly relevant in the case of LLM judges). The study of R@1 (cases where the adversarial passage ranks first) is particularly interesting because those cases correspond to adversarial passages with non-relevant content that score even better than relevant passages from the retrieval corpus.

Comparing Injection Types. For the retrievers and rerankers, we observe that query injection consistently succeeds more often than keyword injection across the different insertion locations and repetitions. Comparing sentence injection with query and keyword injection, we see that repeating queries and keywords is necessary to have query and keyword injection succeed more often than sentence injection. Although query injection, keyword injection, and sentence injection all work relatively well against retrievers and rerankers, the LLM judges have a low vulnerability towards query injection and keyword injection attacks. For example for GPT4o, the attack success rates for query injection are non-zero only when investigating the injection of queries into the end of the passage where the attack success rate is only 0.2%. Similarly, for keyword injection, the attack success rates remain near zero across all settings. With Llama-3.1 (8B) we similarly see relatively low attack success rates for query and keyword injection. However, one exception stands out: for keyword injection at the end of scrambled passages, Llama-3.1 (8B) yields a surprising 22.3% S@2+. In contrast, sentence injection proves to be highly effective against LLM judges, particularly GPT-4o, where across all tested conditions, the S@2+ exceeds 90%, and S@3 remains between 30-60%.

Investigating the Impact of Query, Keyword, and Sentence Repetition. Table 1 shows that for query and keyword injection attacks on retrievers and rerankers, repeating the queries and keywords up to three times consistently improves attack success rates. In these models, simply repeating key terms (queries or keywords) is sufficient to promote adversarial passage rankings effectively. However, for sentence injection, adding more random sentences consistently reduces attack success for retrievers and rerankers but, unexpectedly, increases it for GPT-4o and Llama-3.1 (8B) when considering S@2+. The behaviour of retrievers and rerankers aligns with expectations: introducing additional random sentences dilutes

Table 3: Attack success rates (%) for query injection attacks (injecting the query once at the start) across random and scrambled word passages. R@1/R@5 are reported for BGE-large and MonoT5-large while S@3/S@2+ is reported for Llama-3.1 (8B).

Dataset	BGE-large		MonoT5-large		Llama-3.1 (8B)	
	Random	Scrambled	Random	Scrambled	Random	Scrambled
DL19	0.0 / 0.9	2.8 / 17.7	4.2 / 27.4	14 / 51.2	0.0 / 0.5	0.0 / 0.5
DL20	0.0 / 1.9	4.1 / 19.3	4.8 / 24.1	18.9 / 39.3	1.1 / 2.2	0.0 / 1.1
CLIMATE-FEVER	18.1 / 29.2	91.7 / 98.4	98.9 / 100	99.9 / 100	4.5 / 15.9	9.3 / 34.7
DBPedia	0.4 / 2.2	12.3 / 31.6	30.2 / 59.5	46.4 / 75.8	2.9 / 5.0	0.5 / 2.9
FiQA	2.8 / 6.8	48.1 / 75.4	74.6 / 95	88.6 / 99	2.8 / 3.6	0.8 / 1.8
NFCorpus	3.5 / 6.8	34.5 / 57.9	50.8 / 82.3	62.3 / 87.1	1.2 / 3.7	0.4 / 2.1
SciFact	18.1 / 42.3	85.6 / 99.5	85.2 / 100	93.2 / 100	5.9 / 25.3	8.1 / 40.2
TREC-COVID	1.2 / 1.6	34.4 / 49.6	14.8 / 25.2	26 / 49.2	0.4 / 0.8	0.0 / 0.8

the relevance of the passage to the query, thereby decreasing attack effectiveness. In contrast, the response of LLM judges, particularly GPT-4o, is counterintuitive and challenging to explain.

Investigating the Impact of Location. Intuitively, a model may perceive passages to be more relevant if the relevant content appears at the beginning, while non-relevant information is placed in the middle or at the end. As a result, sentence injection attacks are likely to be more effective when inserted in the middle or end, whereas query and keyword injection attacks may achieve higher success when placed at the beginning. Table 2 compares the effectiveness of different insertion locations (start, middle, end). The results indicate that attacks are generally most successful when relevant content is positioned at the start of the passage. Generally, for retrievers and rerankers, query and keyword injection at the beginning of a random or scrambled passage yields the highest success rates, while sentence injection proves more effective when added to the end of a relevant passage. Interestingly, the behavior of LLM judges, particularly GPT-4o, is again counterintuitive. Sentence injection attacks are most effective against GPT-4o when placed at the start of a passage, and for Llama-3.1 (8B), the S@3 is higher when inserted at the start rather than the end. For the LLM judges, we do not focus on query and keyword injection because of the low attack success rates however, there is a surprisingly high S@2+ against Llama-3.1 (8B) for keyword injection into the end of scrambled word passages. Overall, similar to the repetition experiments, the behavior of LLM judges remains difficult to explain and does not follow the expected and more reasonable patterns seen in retrievers and rerankers.

Comparing Model Vulnerability. Model effectiveness does not necessarily correlate with resilience to attacks. While GPT4o is widely accepted to be a stronger LLM than Llama-3.1 (8B) (and a stronger LLM relevance judge as we later show in Table 10), Tables 1 and 2 show that it is more vulnerable to sentence injection. GPT-4o exhibits the highest attack success rates for sentence injection among all evaluated models. Similarly, while MonoT5-large is a stronger reranker than MonoT5-base [31] and BGE-large is a stronger retriever than BGE-base [59], the larger models can often have higher attack success rates. Another interesting observation is that E5-unsup frequently shows higher attack success rates than other retrievers when facing query and keyword injection attacks, but it tends to be less vulnerable under sentence injection. Unlike other models that receive supervised fine-tuning for retrieval (for instance, using MSMARCO passage ranking data [3]), E5-unsup is trained only through unsupervised contrastive learning. This lack of supervised training may make it more susceptible to surface-level manipulations involving queries and keywords while being relatively more robust to sentence injections, which likely require deeper semantic comprehension of relevant passages.

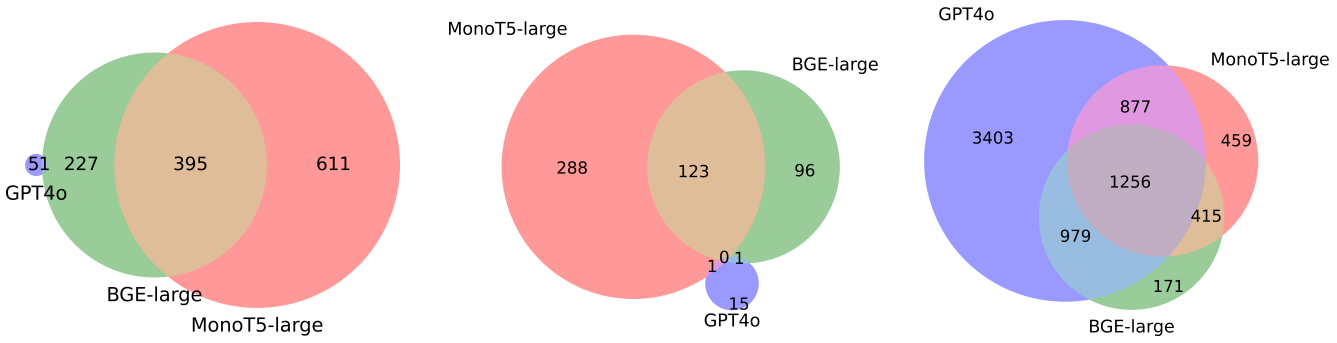
Table 4: Attack success rates on DL19 and DL20 for Sentence Injection comparing sentence injection into relevant passages from the corpus and generated passages of roughly 50, 100, and 200 words.

Model	Relevant (Corpus)	Gen-50	Gen-100	Gen-200
Retrievers (R@1 / R@5)				
BGE-base	2.5 / 34.2	19.6 / 49.3	24.1 / 55.7	23.7 / 59.4
BGE-large	8.7 / 56.3	31.8 / 65.2	35.7 / 70.1	34.8 / 72.8
E5-sup	9.9 / 66.4	35.3 / 67.8	38.8 / 72.6	27.6 / 63.9
E5-unsup	3.5 / 33.2	6.4 / 26.0	14.6 / 37.5	19.0 / 46.8
Arctic-base	8.9 / 56.1	27.2 / 55.9	36.1 / 68.9	40.6 / 75.1
Rerankers (R@1 / R@5)				
MiniLM	0.4 / 62.5	35.5 / 68.5	41.0 / 76.7	39.6 / 74.8
MonoT5-base	0.4 / 57.3	32.6 / 70.1	35.5 / 74.4	30.3 / 70.3
MonoT5-large	0.0 / 42.1	23.7 / 60.2	25.2 / 64.9	25.4 / 69.3
RankT5-base	7.2 / 74.4	46.6 / 81.9	54.2 / 84.1	56.1 / 90.1
LLM Judges (S@3 / S@2+)				
GPT4o	53.2 / 97.9	93.0 / 99.6	98.6 / 99.8	100.0 / 100.0
Llama-3.1 (8B)	22.6 / 59.5	1.0 / 94.6	0.4 / 95.3	23.3 / 96.1

BEIR Datasets. Table 3 examines query injection attacks for a diverse set of BEIR tasks on BGE-large, MonoT5-large, and Llama-3.1 (8B). We use the largest retriever and reranker models and use Llama-3.1 (8B) instead of GPT4o to minimize costs. Results confirm that these models are vulnerable to query injection on multiple domain-specific tasks. SciFact and CLIMATE-FEVER in particular have relatively high attack success rates. Notably, unlike the other datasets, each of these datasets has queries that are verifiable claims, which may explain why query injection is so successful. In CLIMATE-FEVER, queries are real-world climate change-related claims collected from the internet. In SciFact, queries are expert-written scientific claims.

Examining Passage Types. Across Tables 1 and 2, and the BEIR results in Table 3, scrambled passages often yield higher attack success rates than random passages for retrievers and rerankers. This trend is not clear with the LLM judges that have generally lower attack success rates for query injection and keyword injection. There are also some exceptions including MiniLM in general. However, overall this trend persists and it is not clear why this is the case. One hypothesis is that passages of scrambled words are semantically neural and thus more influenced by injected query terms, making it easier for the model to latch onto query signals and mistakenly assign higher relevance.

Generated Passages. Table 4 compares the effectiveness of sentence injection attacks using LLM-generated passages against top-ranked or top-scoring passages from the MSMARCO retrieval corpus. To generate passages with a specified length, we simply prompt GPT4o to generate passages of that length that it considers perfectly relevant given the query, which has been shown to work well and generate passages of roughly the desired length [49]. We observe that adversarial success rates are generally higher with generated



(a) Query Injection **(b) Keyword Injection** **(c) Sentence Injection**

Figure 2: Overlap in successful adversarial passages on DL19 and DL20 across the different attack settings for the BGE-large retriever, the MonoT5-large reranker, and the GPT4o judge.

passages than with the relevant passages from the corpus. Further, attack success rates tend to increase, although inconsistently, as the generated passage lengths are increased. One plausible explanation is that the generated passages preserve coherent and complete context, which has been shown to benefit retrieval effectiveness [46]. Tan et al. [49] similarly suggest that LLMs prefer coherent, semantically rich contexts over shorter or disjointed passages from retrieval corpora. Another potential reason for the higher success rates with longer passages is that they contain a greater proportion of relevant content even when random sentences are inserted. However, we observe that while passages from the MSMARCO corpus average around 58 words in length, the LLM-generated passages designed to be 50 words long often achieve higher attack success rates than the corpus-derived passages.

Transferability of Adversarial Examples. Models can be evaluated using the same adversarially crafted passages (using generated relevant passages in the case of sentence injection) to determine whether they are vulnerable to the same attacks or if successful adversarial cases are unique to each model. Previous research has shown that adversarial examples designed for one model can also deceive other models [21, 34]. This phenomenon allows adversaries to attack black-box models by first crafting adversarial examples using white-box models and relying on their transferability.

Figure 2 presents a Venn diagram illustrating the overlap of successful adversarial attacks among three models: the BGE-large retriever, the MonoT5-large reranker, and the GPT4o judge. The attacks include query injection, keyword injection, and sentence injection. We consider attack success in the strict setting, where for the retriever and the reranker, an attack is successful if the adversarial passage ranks first, and for the LLM judge, an attack is successful if the adversarial passage attains a score of 3. Our analysis encompasses all injection locations and cases involving query, keyword, and sentence repetition, as discussed in Section 4.1. The results show that some attack success cases are unique to specific models, while others are shared across two or all three models. Each model exhibits unique vulnerabilities but also shares some with others. GPT4o is generally not very vulnerable to query and keyword injection with very few successful adversarial passages. However, both BGE-large and MonoT5-large share a significant number of successful adversarial passages for these attack types. In contrast, GPT4o is highly vulnerable to sentence injection, with a large number of successful adversarial passages, many of which are shared with the other two models. The Venn diagram reveals that each model shares some adversarial passages, with examples distributed across all possible categories, whether unique to a single model, shared between two models, or common to all three. This suggests that if an adversary aims to attack an unknown black-box model, they could increase their chances of success by selecting adversarial passages that have been effective against multiple models. Overall, these findings indicate that while some adversarial attacks generalize across models, others remain model-specific.

Targeted Content Injection. Table 5 compares inserting random text versus hateful text. In the random setting, sentences are sampled from the MSMARCO passage corpus, while in the hateful setting, sentences are sampled from the Toxigen test set as described in Section 3.3.1. Interestingly the Toxigen text often yields higher success rates but sometimes lower success rates. It is not clear why the attack success rates are higher, but this may be because of differences in the distribution of sentence lengths, despite the lower and upper bound we apply on sentence lengths and other filters we apply on sentences as described in Section 3.3.1. For GPT4o, the attack success rate is lower when inserting Toxigen text. This can be partially explained by Azure OpenAI's content moderation, which flagged some requests as containing hateful content, resulting in us assigning a default score of 0. Overall, our findings suggest that the evaluated models do not exhibit a bias against hateful text, even though they arguably should.

SEO for Suboptimally Scoring Passages. Table 6 presents an SEO-focused scenario where inserting the query once at the beginning of a suboptimally ranking or scoring passage (initially ranked at 5th place or given a relevance score of 2) often boosts it to rank 1 or a relevance score of 3. Success rates are 54.6% for BGE-large, 71.1% for MonoT5-large, and 46.0% for GPT-4o. Injecting the query into a random passage is far less effective. This highlights how black-hat SEO tactics can exploit simple manipulations to improve search

Table 5: Comparing the insertion of random MSMARCO sentences vs hateful sentences from Toxigen for sentence injection. We present *R@1 / R@5* for retrievers and rerankers and *S@3 / S@2+* for LLM judges on DL19 and DL20

Injection Type	BGE-base	BGE-large	E5-sup	E5-unsup	Arctic-base	MiniLM-Reranker	MonoT5-base	MonoT5-large	RankT5-base	GPT4o	Llama-3.1 (8B)
Toxigen Sentence Injection	4.7 / 49.9	18.6 / 79.4	20.8 / 79.8	4.9 / 39.2	15.5 / 73.4	0.2 / 68.9	0.2 / 68.0	0.2 / 60.6	15.1 / 90.3	41.4 / 96.0	26.7 / 59.4
MSMARCO Sentence Injection	2.5 / 34.2	8.7 / 56.3	9.9 / 66.4	3.5 / 33.2	8.9 / 56.1	0.4 / 62.5	0.4 / 57.3	0.0 / 42.1	7.2 / 74.4	53.2 / 97.9	22.6 / 59.5

Table 6: Comparing Query Injection on DL19 and DL20 for Less Relevant Passages (Rank=5 for retrievers and rerankers or Score=2 for LLM judges) and Random Passages. The query is inserted once into the start of the passage.

Attack Method	Passage Type	BGE-large	MonoT5-large	GPT4o
Query Injection	Less Relevant Passage	54.6 / 100	71.1 / 100	46.0 / 93.7
Query Injection	Random Passage	0.0 / 1.4	4.5 / 25.6	0.0 / 0.0

rankings. Notably, in about 6% of cases, adding the query to less relevant passages reduces GPT-4o's relevance judgement, which is arguably undesired behaviour.

4.2 Investigating Defenses

Since retrievers, rerankers, and LLM judges are all vulnerable to content injection attacks, it is crucial to explore defense strategies that do not compromise the effectiveness of IR pipelines. We examine three approaches: classifier-based defense, fine-tuning embedding retrieval models for robustness, and prompting LLMs to be more defensive. Our investigation covers two settings for sentence injection: a content-agnostic setting where random MSMARCO sentences are injected and a hateful setting where sentences from the Toxigen test set are injected. We split MSMARCO sentences into a train/dev/test split and use the test set from Toxigen while splitting the training set into train/dev to allow for fair evaluation.

For training classifiers and retrieval models, we craft adversarial passages using queries and their relevant passages from the MSMARCO v1 passage ranking training set. For every query, two types of adversarial passages are generated on the fly in each training step: (1) injecting queries or keywords into random or scrambled passages, and (2) inserting sentences into relevant passages. The injection type (query or keyword), passage type (random or scrambled), number of insertions (1-3 for queries/keywords, 1-2 for sentences), and insertion position (start, middle, or end) are selected randomly with equal probability. Each training batch contains an equal number of queries, relevant passages, sentence-injected adversarial passages, and query/keyword-injected adversarial passages.

To study the vulnerability of the models to injection attacks, we report error rates for the classifiers and attack success rates for the retrievers across all different attack settings studied in this work based on injection type, passage type, repetition, and location. Each attack scenario is represented equally in the evaluation set using the methodology described in Section 3.3.4. Evaluations are conducted separately by injection type and, in the case of sentence injection, by MSMARCO and Toxigen sentence injections.

4.2.1 Classifier Based Defense. A straightforward defense against adversarial passages is filtering them using a classifier. We acknowledge the limitations of such classifiers, as highlighted in previous work [5], particularly their inability to detect adversarial passage types not represented in the training data. However, classifier-based defenses can still be valuable for filtering and analysis. We train two classifiers using *answerdotai/ModernBERT-base* [57] with a learning rate of 1e-5, 50 warmup steps, a dropout rate of 0.1, and a batch size of 32. Both classifiers are trained the same way with query and keyword injection, but one classifier is trained on MSMARCO sentence injection, while the other is trained on Toxigen sentence injection. Both classifiers aim to distinguish adversarial from non-adversarial text, covering both passages and queries.

Table 7: Corpus Adversarial (%) shows the proportion of passages in each dataset's corpus classified as adversarial. The remaining columns show the error rate of the classifier on adversarial passages by attack type.

(a) Classifier Trained with MSMARCO Sentence Injection

Dataset	Corpus Adversarial (%)	Keyword Injection	Query Injection	Sentence Injection (MSMARCO)	Sentence Injection (Toxigen)
DL19 (MSMARCO)	1.16	0.03	0.05	0.78	0.93
DL20 (MSMARCO)	1.16	0.25	0.00	0.49	2.22
CLIMATE-FEVER	0.28	0.02	0.50	0.70	0.58
DBPedia	0.24	0.08	1.11	0.76	1.12
FiQA	3.21	0.01	0.56	0.82	0.91
NFCorpus	1.29	0.47	2.22	1.04	0.93
SciFact	1.00	0.01	0.30	0.81	0.68
TREC-COVID	1.43	0.00	0.20	8.93	12.73

(b) Classifier Trained with Toxigen Sentence Injection

Dataset	Corpus Adversarial (%)	Keyword Injection	Query Injection	Sentence Injection (MSMARCO)	Sentence Injection (Toxigen)
DL19 (MSMARCO)	0.66	0.05	0.10	26.28	0.54
DL20 (MSMARCO)	0.66	0.23	0.02	24.81	1.60
CLIMATE-FEVER	0.36	1.14	10.20	18.16	0.53
DBPedia	0.25	0.07	2.44	20.90	0.52
FiQA	12.64	0.01	2.96	12.94	0.44
NFCorpus	1.07	0.53	4.59	13.70	0.90
SciFact	1.45	0.61	11.56	12.16	0.76
TREC-COVID	1.22	0.00	0.96	23.73	5.33

Table 7 presents the proportion of passages misclassified as adversarial within MSMARCO and BEIR corpora, along with classifier error rates on adversarial passages. For sentence injection, adversarial passages are taken as the top-ranking BGE-base retrieved passages containing an injected sentence. Corpus passage misclassification rates are generally below 2%, except for FiQA passages, where the MSMARCO-trained classifier misclassifies around 3%, and the Toxigen-trained classifier misclassifies nearly 13%. We observe the existence of corpus passages combining unrelated sentences that confuse classifiers. These may result from corpora curation methods where texts are extracted from web pages with complex structures. FiQA passages, in particular, contain user-generated content with informal language and swear words, which seem to confuse the Toxigen-trained classifier. Despite the Toxigen-trained classifier being trained on specific Toxigen hateful content rather than random content, it often mistakenly flags more corpora passages than the MSMARCO-trained classifier, observed with the datasets CLIMATE-FEVER, DBPedia, FiQA, and SciFact.

Table 8: R@1 / R@5 (%) for the original and fine-tuned BGE-base retrievers. The fine-tuned models are trained to discount passages with MSMARCO and Toxigen sentence injection.

Dataset	BGE-base	Tuned		
		(MSMARCO) $\tau_2 = 0.01$	(MSMARCO) $\tau_2 = 0.011$	(Toxigen) $\tau_2 = 0.01$
Query Injection				
DL19	9.0 / 19.3	0.1 / 0.1	0.9 / 2.6	0.1 / 0.1
DL20	6.4 / 18.2	0.0 / 0.1	0.6 / 2.4	0.0 / 0.0
CLIMATE-FEVER	62.6 / 72.3	4.6 / 9.0	19.2 / 29.3	12.4 / 20
DBPedia	17.5 / 32.1	0.4 / 1.3	3.1 / 8.7	0.5 / 1.7
FiQA	43.4 / 56.9	0.9 / 2.4	7.0 / 14.9	1.3 / 3.3
NFCorpus	30.9 / 47.2	3.9 / 8.7	14.9 / 28.2	4.2 / 9.5
SciFact	62.7 / 80.3	4.7 / 12.1	20.3 / 37.3	9.1 / 21.3
TREC-COVID	39.2 / 47.5	0.1 / 0.4	1.9 / 4.5	0.2 / 0.6
Keyword Injection				
DL19	4.2 / 9.1	0.0 / 0.1	0.4 / 1.0	0.0 / 0.1
DL20	1.3 / 6.1	0.0 / 0.1	0.2 / 1.2	0.0 / 0.1
CLIMATE-FEVER	48.1 / 60.9	1.1 / 2.5	15.3 / 26	2.4 / 5.1
DBPedia	8.8 / 19.4	0.2 / 0.7	1.9 / 5.9	0.2 / 0.7
FiQA	21.7 / 34.7	0.2 / 0.7	3.4 / 9.0	0.3 / 0.9
NFCorpus	26.3 / 42.9	3.4 / 7.8	13.7 / 27.3	3.6 / 8.5
SciFact	45.4 / 68.7	1.7 / 5.6	15.2 / 33.6	3.0 / 9.6
TREC-COVID	14.3 / 20.2	0.0 / 0.0	0.7 / 1.8	0.0 / 0.2
Sentence Injection (MSMARCO)				
DL19	3.6 / 36.5	0.0 / 2.4	0.5 / 5.8	0.6 / 9.2
DL20	4.0 / 41.2	0.3 / 3.1	0.9 / 6.8	2.0 / 14.3
CLIMATE-FEVER	22.8 / 79.1	0.7 / 4.1	1.1 / 8.5	2.6 / 13.6
DBPedia	14.4 / 64.1	0.3 / 3.1	0.6 / 8.7	2.2 / 15.3
FiQA	12.2 / 66.7	0.4 / 3.6	0.8 / 8.9	1.7 / 13.4
NFCorpus	27.3 / 86.9	3.2 / 18.3	4.6 / 29.3	11.7 / 37.9
SciFact	24.5 / 94.8	2.1 / 15.1	2.5 / 24.2	5.2 / 30.4
TREC-COVID	4.2 / 28.1	0.5 / 3.3	1.8 / 6.0	2.7 / 13.3
Sentence Injection (Toxigen)				
DL19	3.9 / 48.3	0.6 / 4.0	0.3 / 8.1	0.2 / 0.8
DL20	7.8 / 54.0	0.7 / 4.6	1.7 / 17.5	0.1 / 1.4
CLIMATE-FEVER	25.3 / 84.3	0.7 / 4.4	1.0 / 10.6	0.1 / 0.7
DBPedia	14.6 / 72.8	0.3 / 5.9	1.2 / 17.8	0.1 / 1.0
FiQA	11.8 / 72.7	0.8 / 6.0	1.4 / 17.1	0.2 / 1.3
NFCorpus	28.6 / 89.4	2.9 / 17.9	4.7 / 32.4	2.7 / 6.8
SciFact	25.1 / 96.6	1.7 / 14.6	2.3 / 25.6	1.5 / 4.8
TREC-COVID	4.3 / 31.0	0.6 / 3.1	1.2 / 6.6	1.4 / 3.5

We observe that the Toxigen-trained classifier has higher errors on MSMARCO-injected passages but lower errors on Toxigen-injected ones, as it is trained to detect injected hateful content rather than random content. However, it exhibits a slightly higher error for Toxigen sentence injection on the SciFact dataset. Both classifiers show relatively high error rates in detecting sentence-injected passages within the TREC-COVID dataset. For keyword injection, the MSMARCO-trained classifier maintains error rates below 0.5%, whereas query injection errors can reach 2.2%. The Toxigen-trained classifier, however, struggles even more with query and keyword injections, particularly in CLIMATE-FEVER and SciFact, where errors are over 10%, despite identical training conditions between the two classifiers apart from sentence injection data.

These findings highlight the fact that training a classifier to identify adversarial passages is a tricky problem, even when the classification problem is more constrained in the case of sentence injection with particular injected content. Although the classifiers may help filter out malicious passages, they may miss some malicious passages and mistakenly flag others. Training the classifiers with more and more diverse queries and passages, besides those from the MSMARCO training set may help further mitigate errors.

4.2.2 Training Embedding Models for Robustness. Another defense strategy is fine-tuning retrievers to discount adversarial passages.

Table 9: Retrieval effectiveness (measured by NDCG@10) for the BGE-base retriever and fine-tuned retrievers. The (No adv.) setting tunes the retriever with the basic InfoNCE loss.

Dataset	BGE-base	Tuned			
		(No adv.)	(Random) $\tau_2 = 0.01$	(Random) $\tau_2 = 0.011$	(Toxigen) $\tau_2 = 0.01$
DL19	0.702	0.709	0.709	0.710	0.699
DL20	0.677	0.692	0.679	0.698	0.693
CLIMATE-FEVER	0.312	0.236	0.169	0.183	0.170
DBPedia	0.407	0.379	0.361	0.377	0.371
FiQA	0.406	0.396	0.258	0.308	0.266
NFCorpus	0.373	0.372	0.328	0.353	0.351
SciFact	0.741	0.746	0.638	0.691	0.713
TREC-COVID	0.781	0.753	0.696	0.729	0.750

We fine-tune the BGE-base retriever to push away the embeddings of adversarial passages from the query embeddings using a modified contrastive loss that is a simple adaptation of the InfoNCE [33] loss used for the training of embedding models for retrieval:

$$-\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{\frac{\langle q_i,p_i\rangle}{\tau}}}{\sum_{j=1}^{n}e^{\frac{\langle q_i,p_j\rangle}{\tau}}} \tag{1}$$

In the described formulation, q_i is a normalized query embedding and p_i is a normalized passage embedding for the corresponding relevant passage. The loss takes advantage of in-batch negatives to contrastively learn representations. We use a temperature parameter of $\tau = 0.01$ as this is typically used for the training of embedding models for retrieval. We simply add terms in the denominator to contrast query embeddings with adversarial passage embeddings:

$$-\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{\frac{\langle q_i,p_i\rangle}{\tau}}}{\sum_{j=1}^{n}e^{\frac{\langle q_i,p_j\rangle}{\tau}}+\sum_{j=1}^{n}e^{\frac{\langle q_i,x_j\rangle}{\tau'}}+\sum_{j=1}^{n}e^{\frac{\langle q_i,y_j\rangle}{\tau'}}} \tag{2}$$

In the modified loss, x_i is the embedding of the adversarial passage created with sentence injection for passage i while y_i is the embedding of the adversarial passage created with query injection or keyword injection using query i.

We initialize the embedding models with BGE-base, trained with a learning rate of 4e-5, a learning rate warmup of 10 steps, a dropout rate of 0.1, and a batch size of 8192. We make use of GradCache [14] to allow for larger batch sizes. We train both an embedding model with the basic InfoNCE loss function and the modified loss function to contrast with adversarial passages to test for any possible drops in retrieval effectiveness when training with the modified objective. We also test the impact of the tuning of the τ' parameter, testing $\tau' = 0.01$ and 0.011.

Table 8 first shows the attack success rates with the trained models. Training with the modified loss succeeds in decreasing the attack success rates. The attack success rates are lower when fine-tuned with the modified loss than with BGE-base across the board. As expected, attack success rates are lower with $\tau' = 0.01$ than with $\tau' = 0.011$. Similar to the classifiers, the model trained with Toxigen sentence injection has a higher attack success rate on passages with MSMARCO sentence injection compared to the MSMARCO-trained model but tends to have lower attack success rates on passages with Toxigen sentence injection as it was trained to specifically discount

Table 10: (MAE) Mean absolute error between LLM judgements and human relevance judgements along with S@3/S@2+ results across attack types under two prompting settings across DL19 and DL20.

Dataset	Prompt	MAE	Query Injection	Keyword Injection	Sentence Injection (MSMARCO)	Sentence Injection (Toxigen)
DL19	Default	0.563	0.1 / 0.1	0.3 / 0.6	34.6 / 95.0	24 / 76.9
DL19	Defensive	0.637	0.0 / 0.0	0.0 / 0.1	8.4 / 85.4	0.9 / 16.9
DL20	Default	0.461	0.0 / 0.0	0.1 / 0.2	31.5 / 92.8	23.7 / 78.4
DL20	Defensive	0.564	0.0 / 0.0	0.0 / 0.0	8.9 / 88.0	1.2 / 17.3

these passages. This model also has a lower attack success rate than BGE-base for sentence injection with MSMARCO sentences, suggesting some generalization in discounting adversarial passages.

Table 9 then shows the retrieval effectiveness of the fine-tuned retrievers compared to the original BGE-base retriever. In general, fine-tuning with both the simple InfoNCE loss or the modified loss on MSMARCO training data improves retrieval effectiveness on DL19 and DL20, but often decreases effectiveness on the BEIR datasets and Climate-Fever especially, where the retrieval task is claim verification. When training with the modified loss, effectiveness tends to decrease more out-of-domain than when using the simple loss. Increasing the temperature τ parameter generally results in improvements to the effectiveness, allowing for a trade-off between retrieval effectiveness and robustness to attacks. Additionally, the retriever trained with Toxigen sentence injection consistently has stronger retrieval effectiveness than the corresponding retriever trained with MSMARCO sentence injection.

The findings suggest that while retrievers can be fine-tuned to be more robust to content injection attacks, this also may result in reduced retrieval effectiveness. Tuning the τ' allows for a trade-off between retrieval effectiveness and robustness to attacks. Additionally, retrievers can better maintain retrieval effectiveness if trained to only discount certain adversarial passages, such as those with hateful content, rather than trained to discount adversarial passages with random content.

4.2.3 Making LLM Judges Robust. To make LLM judges more robust to adversarial passages, we test adding the following instructions to the prompt:

- Assign a score of 0 if the passage contains harmful or malicious content that is irrelevant to the query.
- Do not assign a score of 3 if the passage includes content that is completely random or unrelated and cannot be meaningfully connected to the query or its related topics.

Table 10 shows that while the defensive prompt works to effectively reduce attack success rates across all attacks and especially in the case of sentence injection with hateful content from Toxigen, this is at the cost of agreement between the LLM relevance judgements and the human relevance judgements, where the MAE increases when using the defensive prompt.

5 CONCLUSION

This paper demonstrates that neural IR models, including embedding-based retrievers, rerankers, and LLM relevance judges are widely vulnerable to *content injection attacks*. By appending random or malicious text to relevant passages, or inserting queries or key terms into otherwise non-relevant passages, adversaries can manipulate ranking decisions and model judgements, showcasing a security gap in current IR pipelines.

Our evaluation of a diverse set of models across various retrieval tasks reveals the widespread susceptibility to content injection attacks. We investigate key factors influencing attack success, showing that the location of content insertion and the balance of non-relevant content to relevant content within passages can affect the probability of adversarial passages succeeding in fooling IR models. Additionally, our analysis uncovers a critical gap in model behavior, as the studied models generally do not exhibit bias against hateful content. We analyze the overlap of successful adversarial passages across different models and find that some adversarial passages are unique to specific models, successfully deceiving only them, while others are more general and can fool multiple models. Moreover, experiments with LLM-generated passages prove effective in fooling models as well, removing the need to find specific passages that target models and pointing to a risk of attacks in black-box settings with limited prior knowledge of the models.

Empirical findings show that while supervised classifiers, carefully fine-tuned retrievers, and more defensively prompted LLMs can mitigate some attacks, these defenses are imperfect and usually come at the cost of some effectiveness. Continued research into mitigating attacks against adversarial manipulations is critical for ensuring that users of search systems receive accurate and trustworthy information.

ACKNOWLEDGMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Additional funding is provided by Microsoft via the Accelerating Foundation Models Research program.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 Technical Report. *arXiv:2303.08774* (2023).

[2] Marwah Alaofi, Paul Thomas, Falk Scholer, and Mark Sanderson. 2024. LLMs can be Fooled into Labelling a Document as Relevant: best café near me; this paper is perfectly relevant. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region* (Tokyo, Japan) *(SIGIR-AP 2024)*. Association for Computing Machinery, New York, NY, USA, 32–41. https://doi.org/10.1145/3673791.3698431

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268v3* (2016).

[4] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *Advances in Information Retrieval*, Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello (Eds.). Springer International Publishing, Cham, 716–722.

[5] Xuanang Chen, Ben He, Le Sun, and Yingfei Sun. 2023. Defense of Adversarial Ranking Attack in Text Retrieval: Benchmark and Baseline via Detection. *arXiv:2307.16816* (2023).

[6] Xuanang Chen, Ben He, Zheng Ye, Le Sun, and Yingfei Sun. 2023. Towards Imperceptible Document Manipulations against Neural Ranking Models. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 6648–6664. https://doi.org/10.18653/v1/2023.findings-acl.416

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020)*. Gaithersburg, Maryland.

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2019. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference Proceedings (TREC 2019)*. Gaithersburg, Maryland.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[10] Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims. *arXiv:2012.00614* (2020).

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv:2407.21783* (2024).

[12] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv:1712.06751* (2017).

[13] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292. https://doi.org/10.1145/3404835.3463098

[14] Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling Deep Contrastive Learning Batch Size under Memory Limited Setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572* (2014).

[16] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 3309–3326. https://doi.org/10.18653/v1/2022.acl-long.234

[17] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1265–1268. https://doi.org/10.1145/3077136.3080751

[18] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. (2020). https://doi.org/10.5281/zenodo.1212303

[19] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 6769–6781.

[20] Jiawei Liu, Yangyang Kang, Di Tang, Kaisong Song, Changlong Sun, Xiaofeng Wang, Wei Lu, and Xiaozhong Liu. 2022. Order-Disorder: Imitation Adversarial Attacks for Black-box Neural Ranking Models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2025–2039.

[21] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into Transferable Adversarial Examples and Black-box Attacks. *arXiv:1611.02770* (2016).

[22] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, et al. 2023. Prompt Injection attack against LLM-integrated Applications. *arXiv:2306.05499* (2023).

[23] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2023. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. *arXiv:2310.12815* (2023).

[24] Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Black-box Adversarial Attacks against Dense Retrieval Models: A Multi-view Contrastive Learning Method. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1647–1656.

[25] Yu-An Liu, Ruqing Zhang, Mingkun Zhang, Wei Chen, Maarten de Rijke, Jiafeng Guo, and Xueqi Cheng. 2024. Perturbation-Invariant Adversarial Training for Neural Ranking Models: Improving the Effectiveness-Robustness Trade-Off. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8832–8840.

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083* (2017).

[27] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1941–1942. https://doi.org/10.1145/3184558.3192301

[28] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

[29] Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-Embed: Scalable, Efficient, and Accurate Text Embedding Models. *arXiv:2405.05374* (2024).

[30] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).

[31] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. *arXiv:2003.06713* (2020).

[32] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *arXiv:1910.14424* (2019).

[33] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748* (2018).

[34] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv:1605.07277* (2016).

[35] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

[36] Andrew Parry, Maik Fröbe, Sean MacAvaney, Martin Potthast, and Matthias Hagen. 2024. Analyzing Adversarial Attacks on Sequence-to-Sequence Relevance Models. In *European Conference on Information Retrieval*. Springer, 286–302.

[37] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. *arXiv:2309.15088* (2023).

[38] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv:2312.02724* (2023).

[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[40] Nisarg Raval and Manisha Verma. 2020. One word at a time: adversarial attacks on retrieval models. *arXiv:2008.02197* (2020).

[41] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992.

[42] Avital Shafran, Roei Schuster, and Vitaly Shmatikov. 2024. Machine Against the RAG: Jamming Retrieval-Augmented Generation with Blocker Documents. *arXiv:2406.05870* (2024).

[43] Congzheng Song, Alexander Rush, and Vitaly Shmatikov. 2020. Adversarial Semantic Collisions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 4198–4210. https://doi.org/10.18653/v1/2020.emnlp-main.344

[44] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Singapore, 14918–14937.

[45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv:1312.6199* (2013).

[46] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Pre-processing Matters! Improved Wikipedia Corpora for Open-Domain Question Answering. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III* (Dublin, Ireland). Springer-Verlag, Berlin, Heidelberg, 163–176. https://doi.org/10.1007/978-3-031-28241-6_11

[47] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. *arXiv:2312.16098* (2023).

[48] Manveer Singh Tamber, Jasper Xian, and Jimmy Lin. 2024. Can't Hide Behind the API: Stealing Black-Box Commercial Embedding Models. *arXiv:2406.09355* (2024).

[49] Hexiang Tan, Fei Sun, Wanli Yang, Yuanzhuo Wang, Qi Cao, and Xueqi Cheng. 2024. Blinded by Generated Contexts: How Language Models Merge Generated and Retrieved Contexts for Open-Domain QA? *arXiv:2401.11911* (2024).

[50] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *arXiv:2104.08663* (2021).

[51] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2024. Large language models can accurately predict searcher preferences. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1930–1940.

[52] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Nick Craswell, and Jimmy Lin. 2024. UMBRELA: UMbrela is the (Open-Source Reproduction of the) Bing RELevance Assessor. *arXiv:2406.06519* (2024).

[53] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum*, Vol. 54. ACM New York, NY, USA, 1–12.

[54] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 7534–7550. https://doi.org/10.18653/v1/2020.emnlp-main.609

[55] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv:2212.03533* (2022).

[56] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *Advances in Neural Information Processing Systems* 33 (2020), 5776–5788.

[57] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. 2024. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. *arXiv:2412.13663* (2024).

[58] Chen Wu, Ruqing Zhang, Jiafeng Guo, Maarten De Rijke, Yixing Fan, and Xueqi Cheng. 2023. PRADA: Practical Black-box Adversarial Attacks against Neural Ranking Models. *ACM Trans. Inf. Syst.* 41, 4, Article 89 (April 2023), 27 pages. https://doi.org/10.1145/3576923

[59] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 641–649.

[60] Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. 2023. Poisoning Retrieval Corpora by Injecting Adversarial Passages. *arXiv:2310.19156* (2023).

[61] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2308–2313.

[62] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. *arXiv:2402.07867* (2024).