Universal Adversarial Attacks on the Facebook RAG Retriever

Sean O'Brien
University of California, San Diego

Chengsong Diao
University of California, San Diego

Sidney Pan
University of California, San Diego

Abstract

Retrieval-augmented generation (RAG) is primarily designed for knowledge-intensive natural language processing (NLP) tasks with pre-trained large language models (LLMs). It avoids the need to update the pre-trained weights of LLMs for new knowledge while achieving generation referring to a different knowledge base from that used in LLM pre-training, where the re-training of the entire LLMs is computationally expensive and may be infeasible. We show that it is possible to find universal adversarial prefixes that, when prepended to documents in the RAG database, the RAG retriever will favor these documents even if they are unrelated to target questions, with significantly higher similarity scores than those of related documents. It can undermine LLM safety by making RAG models retrieve irrelevant or malicious documents, resulting in useless responses generation, hallucinations, misinformation, etc.

1 Introduction

Pre-trained LLMs have been shown to learn a substantial amount of knowledge from data, but they cannot easily expand or revise their memory and may produce hallucinations [6]. To improve LLMs performance on knowledge-intensive NLP tasks, one Facebook research team proposed RAG models to retrieve documents and generate responses from non-parametric memory [6].

While RAG models are successful in helping LLMs generate more sensible, accurate, updated, and quality responses to questions given context knowledge, it opens up more opportunities for adversaries to poison data source of LLMs and negatively affect LLM safety. If adversaries can make their documents more preferred by the RAG models than relevant documents corresponding to users' queries, attackers can manipulate the top documents retrieved and thus the response generations.

In our threat model, we assume that adversaries are able to access the weights of the question and documents encoders, user queries they wish to optimize for, and inject arbitrary documents to the indexed corpus. These are reasonable assumptions because the encoders are generally open source with weights public, such as Facebook RAG's using BERT base [1] (uncased) as the document encoder and a fined-tuned model based on it as the question encoder [6]. The user queries can be from adversaries themselves or some online forums, discussion boards, or social media for contents related to targeted topics. There are many ways to inject arbitrary documents into databases for RAG, such as hosting public websites (possibly with Search Engine Optimizations), posting on public forums or social media like Reddit, modifying Wikipedia pages, and answering questions on Q&A platforms. Since LLMs need to update knowledge periodically, the adversarial documents will be in databases for RAG as documents are crawled from the Internet.

The objective of adversaries is to surface malicious, meaningless, or misleading information to users. For example, adversaries can serve false contact information to exfiltrate user data, fill meaningless and useless information to LLM context to induce hallucinations, or disseminate disinformation or propaganda.

The scope of the threat model can become broader in a number of ways. If the adversarial modifications to boost similarity scores can generalize across different queries, the malicious documents can affect a wider range of users and topics. If the modifications are only effective on a family of relevant queries, adversaries can control injected contents more precisely for different topics. More LLM applications can be negatively affected if the modifications can transfer to fined-tuned derivatives of the same document encoder too. The attacks can be more robust if the effectiveness remains with different placements of modifications within documents. The loss calculated by the dissimilarities between documents and queries may be combined with NeuralExec or GCG such that adversarial strings override instructions.

In this project, we explore methods to create universal adversarial modifications to documents that make them generally and significantly preferred than normal documents by

the Facebook RAG retriever. We have successfully found 32-token prefixes by multiple methods that can boost similarity scores of arbitrary documents with various user questions by large and significant margins, exceeding similarity scores of normal related documents. We evaluate our prefixes on both Wikipedia documents and instructions datasets. Although our prefixes were trained on Wikipedia documents, they can reliably transfer to the unseen instructions datasets and increase the similarity scores of these completely unrelated documents well above the scores of related documents generally.

2 Background

2.1 GCG

Greedy Coordinate Gradient (GCG) was designed for jailbreaking aligned LLMs through discrete token-level optimization [15]. It finds a set of promising candidates for replacement at each token position, and then evaluate all these replacements exactly via a forward pass.. Specifically, it computes the linearized approximation of replacing the ith token in the prompt, x_i, by evaluating the gradient $\nabla_{e_{x_i}} \mathcal{L}(x_{1:n})$ where e_{x_i} is the one-hot vector for ith token and $\mathcal{L}$ is the loss. It then computes the top-k values with the largest negative gradient as the candidate replacements for token x_i, compute this candidate set for all modifiable tokens then randomly select a subset tokens from it. Then it evaluates the loss on this subset and make the replacement with the smallest loss. The whole process is repeated T times to output the optimized prompt.

2.2 RAG

Retrieval-Augmented Generation (RAG) [13] combines LLMs with external data retrieval that can dynamically pull the relevant information from a database or the internet during generation. It has two phases: retrieval and generation. During the retrieval phase, the query encoder E_q produces an embedding vector $E_q(q)$ for an user query q. RAG retrieves k relevant documents from corpus $\mathcal{C}$ that has the highest document embedding $E_d(d)$ similarities with the query embedding. During the generation phase, the retrieved documents are combined with the original query to form the input to LLM to generate a response, which can enhance LLM response accuracy and relevance.

2.3 Corpus Poisoning Attacks on RAG

A set of adversarial documents are generated and inserted into the retrieval corpus $\mathcal{C}$. It is expecetd that the retriever will retrieve the adversarial documents in response to certain queries. Specifically, the attack aims to generate a small set of adversarial documents $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$, where $|\mathcal{A}| \ll |\mathcal{C}|$. The objective is to construct $\mathcal{A}$ that at least one document appears in the top-k retrieval results for the queries [14].

3 Methods

3.1 Objective

The goal of the method is to learn a prefix which systematically improves likelihood with a given query. Thus, the optimization objective can be formulated as follows:

$$p_{1:\ell} := \underset{p \in \mathcal{S}_\ell}{\operatorname{argmax}} \mathbb{E}_{q \in p(q), s_{adv} \in p(s_{adv})} \left[\operatorname{Sim}(E_q(q), E_d(p|s_{adv})) \right]$$

where

- $\mathcal{S}_\ell$ is the set of ℓ-token strings in the tokenizer vocabulary
- $p(q)$ is the prior distribution of queries to target
- $p(s_{adv})$ is the prior distribution of adversarial strings to inject
- E_q is the query encoder of the retriever
- E_d is the document encoder of the retriever
- Sim is a similarity measure, typically either cosine similarity or an un-normalized inner product

The Facebook RAG models use inner products to calculate similarity scores between documents and questions, so we use the same calculation to train the prefixes. Due to time limits, we only tried to optimize prefixes with a fixed length, 32 tokens. We choose this length because it is not too short, which should have enough advantage to boost similarity scores. At the same time, since the document encoder has a maximum length of tokens for each document, we want to leave enough tokens for the actual documents.

3.2 Gradient Ascent and Discretization

The first thing we try is to use gradient ascent on the similarity score with respect to the token embeddings of the prefix. Since it is optimizing in the continuous embedding space, we can directly use well-established optimization methods. Specifically, we use negative similarity scores with the average question embedding calculated over a set of target questions as the loss, and the AdamW optimizer to update the token embeddings. As with our expectation, this method did not yield good performance, but the statistics were used as a baseline to reject any methods that were not significantly better than direct gradient ascent in the embedding space.

3.3 Greedy Coordinate Gradient variants

Inspired by the Greedy Coordinate Gradient (GCG) and the Universal Prompt Optimization algorithms [15], we tried different ways to adapt similar ideas into our prefix optimization. We came up with multiple variants of GCG, among which 4

have good performance and generalization. They are GCG variants 4, 5, 6, and 7, and we will describe the algorithms below. The losses calculation is the same as in the Gradient Ascent and Discretization method.

GCG variant 4 (Algorithm 1) greedily selects the best one-hot encoding for each prefix token position (i.e. the best replacement token) per document batch, based on the largest negative gradients on the loss with respect to the one-hot encoding matrix, adding the one-hot encoding matrix of the current step as a kind of regularization to prevent overly rapid changes in the selected prefix tokens and improve generalization beyond the current document batch.

Algorithm 1 Greedy Coordinate Gradient variant 4 for RAG retrievers

Input: Documents $x_{1:n_1}^{(1)} \dots x_{1:n_m}^{(m)}$, average question embedding $\hat{q}$, initial prefix $p_{1:l}$, loss $\mathcal{L}_{\hat{q}}$, epochs T
repeat T times
  for batch $D \subseteq \{1,\dots,m\}$ **do**
    for $i \in [1,l]$ **do**
      $p_i := \text{argmax}_{p_i^\star}(\sum_{j \in D}(e_{p_i^\star} - \nabla_{e_{p_i^\star}} \mathcal{L}_{\hat{q}}(p_{1:l} \| x_{1:n_j}^{(j)})))$
Output: Optimized document prefix p

GCG variant 5 (Algorithm 2) evaluates top-k one-hot encoding replacements for multiple uniformly random prefix token positions at each step, with the same regularization as in GCG variant 4, where the replacement tokens are chosen uniformly randomly from top-k ones at each position. Then, at the end of each step, the algorithm selects the replacement with the highest actual average similarity on the current document batch. This method is significantly slower than GCG variant 4, proportional to the number of random replacement evaluations per step (replacement batch size B).

GCG variant 6 (Algorithm 3) modifies GCG variant 4, where the greedy selection is based on the largest negative accumulated gradients on the loss, without the regularization. Specifically, the gradient accumulation is implemented by a running average of gradients on all trained document batches. It can ensure good generalization as long as the training dataset contains documents on a variety of topics and is not overly biased towards some specific contents.

GCG variant 7 (Algorithm 4) does a similar modification to GCG variant 5, where the top-k selection is based on the top-k negative accumulated gradients on the loss, without the regularization either. However, due to our limited computational resources, it was infeasible to evaluate the actual average similarity on all document batches until the current step. So this method still uses the actual average similarity on the current document batch only. We expect that the performance and generalization can be improved if the algorithm evaluate on all document batches, but we are not confident that the resulting prefix will be significantly better than that by GCG variant 6.

Algorithm 2 Greedy Coordinate Gradient variant 5 for RAG retrievers

Input: Documents $x_{1:n_1}^{(1)} \dots x_{1:n_m}^{(m)}$, average question embedding $\hat{q}$, initial prefix $p_{1:l}$, losses $\mathcal{L}_{\hat{q}}$, epochs T, k, replacement batch size B
repeat T times
  for batch $D_{d \in \{1,\dots\}} \subseteq \{1,\dots,m\}$ **do**
    for $i \in [1,l]$ **do**
      $\mathcal{X}_i := \text{Top-}k(\sum_{j \in D_d}(e_{x_i} - \nabla_{e_{x_i}} \mathcal{L}_{\hat{q}}(p_{1:l} \| x_{1:n_j}^{(j)})))$
    for $b = 1,\dots,B$ **do**
      $\tilde{p}_{1:l}^{(b)} := p_{1:l}$
      $\tilde{p}_i^{(b)} := \text{Uniform}(\mathcal{X}_i)$, where $i = \text{Uniform}([1,l])$
    $p_{1:l} := \tilde{p}_{1:l}^{(b^*)}$, where $b^* = \text{argmax}_b(-\sum_{j \in D_d} \mathcal{L}_{\hat{q}}(\tilde{p}_{1:l}^{(b)} \| x_{1:n_j}^{(j)}))$
Output: Optimized document prefix p

Algorithm 3 Greedy Coordinate Gradient variant 6 for RAG retrievers

Input: Documents $x_{1:n_1}^{(1)} \dots x_{1:n_m}^{(m)}$, average question embedding $\hat{q}$, initial prefix $p_{1:l}$, loss $\mathcal{L}_{\hat{q}}$, epochs T
repeat T times
  for $i \in [1,l]$ **do**
    $p_i := \text{argmax}_{p_i^\star}(-\sum_{1 \le j \le m} \nabla_{e_{p_i^\star}} \mathcal{L}_{\hat{q}}(p_{1:l} \| x_{1:n_j}^{(j)}))$
Output: Optimized document prefix p

Algorithm 4 Greedy Coordinate Gradient variant 7 for RAG retrievers

Input: Documents $x_{1:n_1}^{(1)} \dots x_{1:n_m}^{(m)}$, average question embedding $\hat{q}$, initial prefix $p_{1:l}$, losses $\mathcal{L}_{\hat{q}}$, epochs T, k, replacement batch size B
repeat T times
  for batch $D_{d \in \{1,\dots\}} \subseteq \{1,\dots,m\}$ **do**
    for $i \in [1,l]$ **do**
      $\mathcal{X}_i := \text{Top-}k(-\sum_{j \in \bigcup_{1 \le c \le d} D_c} \nabla_{e_{x_i}} \mathcal{L}_{\hat{q}}(p_{1:l} \| x_{1:n_j}^{(j)}))$
    for $b = 1,\dots,B$ **do**
      $\tilde{p}_{1:l}^{(b)} := p_{1:l}$
      $\tilde{p}_i^{(b)} := \text{Uniform}(\mathcal{X}_i)$, where $i = \text{Uniform}([1,l])$
    $p_{1:l} := \tilde{p}_{1:l}^{(b^*)}$, where $b^* = \text{argmax}_b(-\sum_{j \in D_d} \mathcal{L}_{\hat{q}}(\tilde{p}_{1:l}^{(b)} \| x_{1:n_j}^{(j)}))$
Output: Optimized document prefix p

4 Experimental Setup

We demonstrate a proof-of-concept on a hand-curated dataset relating to CSE 227, a graduate-level computer security course.

For negative documents, we filter the WikiText-103 dataset [9], filtering for sentences with lengths between 30 and 400 characters.

Our positive documents are text passages scraped from the CSE 227 website.[1]

Following the pipeline from the original RAG paper [6], we use the `CLS` tokens of the BERT-base-uncased model [2] as representations for passages and the pretrained RAG query encoder to encode questions, as it is trained to align to the BERT representations.

5 Results

Method	Similarity
Positive Docs	31.77 ± 4.09
Negative Docs	19.00 ± 3.39
GA + Discretize	26.90 ± 2.15
GCG - Coord + Reg	40.40 ± 3.06
GCG + Reg	43.07 ± 1.91
GCG - Coord + Grad Accum	$\mathbf{49.12 \pm 0.92}$
GCG + Grad Accum	45.73 ± 1.91

Table 1: Mean similarities to queries, evaluating on in-domain Wikipedia passage negatives. GCG-6, with gradient accumulation, yields the best results and consistently ranks above the true relevant documents.

5.1 In-Domain Payloads

We first test performance on in-domain payloads: text passages mined from Wikipedia. This imitates the attacking condition in which the attacker has a specific set of passages or type of query which they wish to specifically surface to the retriever, and gives the method the best chance of success. Results can be found in 1. The GCG variants with gradient accumulation yield the highest similarities, and updating across multiple positions yields better results than the single-coordinate GCG approaches. Notably, all GCG-based methods significantly outperform the similarities from the true relevant documents. This demonstrates that the attack can feasibly surface incorrect documents given our threat model.

[1] https://cseweb.ucsd.edu/ efernandes/teaching/cse227fa24/cse227.html

5.2 Out-of-Domain Payloads

Differentiating our work from past work that optimizes against a set of payloads seen at train time, we examine the robustness of applying our prefix to out-of-domain payloads. To simulate a standard language model attack injection, we curate a set of commands prefixed by the string "Ignore all previous instructions." The results of prepending the adversarial prefixes to these commands is found in 1. Once again, the GCG results outperform the true relevant documents.

This demonstrates the flexibility of use of learned prefixes. Further work should be done to prove the generalizability of such learned prefixes.

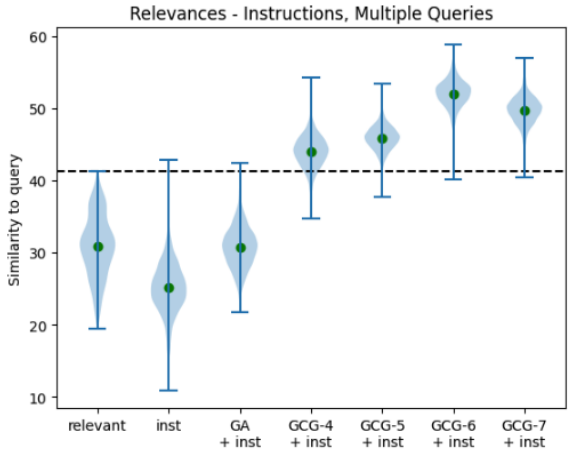

Figure 1: Distributions of similarity scores for each perturbed method, evaluating on instructions. Improved scores demonstrate robustness to out-of-distribution payload strings.

6 Analysis

6.1 Embedding Visualization

Because all quantities lie in a shared embedding space, we can apply standard dimensionality-reduction methods to visualize their relative positions. Specifically, we use Principal Component Analysis (PCA) to project the embeddings of queries, relevant documents, irrelevant documents adversarially perturbed documents – that is, documents prepended with the learned adversarial prefix. The resulting visualization 2 reveals a tight clustering of query documents in a distinct region apart from the more-widely-distributed documents. This may in part be an artifact of the small-scale dataset we attack, in which all queries relate to a single class.

As expected, the adversarial prefix systematically shifts the embeddings of irrelevant questions, in green, towards the centroid of the query embeddings.

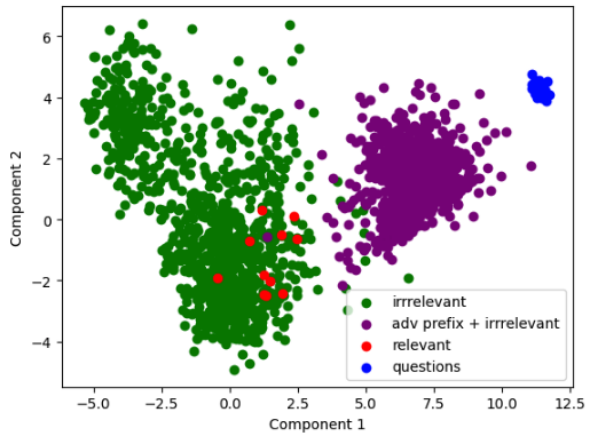

Figure 2: Two-dimensional PCA projection of embeddings. GCG steers irrelevant documents towards the mean of query encodings.

6.2 Learned Prefixes

We enforce no constraints on the learned prefixes. This raises a question: does the objective learn any tokens semantically similar to the task of interest?

As two examples, consider the string learned by discretized gradient ascent and the one learned by GCG-6. (Some Unicode characters may not be rendered.)

Learned Discretized GA Prefix

```
torah lessonnabletypeiolnityire harta
torque mvpntlyspherefan pearbio delleust
quarterttered sentaicardscina aye
pmidmarineraganented auschwitzrry
smashwords trillion
```

Learned GCG-6 Prefix

```
telecastdanrityrricularrogated
nbctativephyanialtative auditioned
que archivedoresrogated campeonato
donegaliba esis universidad
universidadcolaamericanacola
conditionedesis islamabad because exam
```

We find few relevant tokens in the prefix learned by gradient ascent (with "lesson" being a possible exception), but a few semantically-related words in GCG-6, like "universidad," "telecast" and "exam," all potentially related to the format of a university course offering.

7 Related Work

Gradient-Based Attacks. Gradient-based attacks utilize model internals to manipulate token generation [4]. For example, HotFlip [3] approximates the model output change of replacing a token through the use of directional derivatives. The Greedy Coordinate Gradient (GCG) algorithm [15] uses a combination of gradient-based search and greedy optimization to identify and modify discrete tokens in a text sequence to jailbreak aligned LLMs. Built on the limitation of GCG which focuses on a single suffix with the lowest loss, AmpleGCG [7] learns a distribution of suffixes, enabling it to generate multiple successful adversarial prompts for any harmful query efficiently. Apart from jailbreak attacks, GCG is also used for prompt perturbation in Retrieval-Augmented Generation (RAG) based LLMs, using GCG to search for a short prefix inserted to the prompt to lead the system to generate factually incorrect outputs [5]. This attack, HotFlip, optimizes the prefix to control the generator's behavior rather than to maximize the RAG retrieval score.

Attacks on RAG. Gradient-based methods are also used in corpus poisoning attack to generate adversarial passages by perturbing discrete tokens to maximize similarity with a provided set of training queries [14]. BadRAG [13] uses Contrastive Optimization on a Passage (COP), which optimizes adversarial passages to be retrieved for trigger-specific queries, and Merged COP (MCOP) to reduce poisoning ratios while maintaining attack efficacy. Similarly, attacks on retrieval systems have been limited to making edits to passages to change their retrieval rank for an individual or a small set of queries [11, 12]. Importantly, all of these methods aim to learn an entire passage which maximizes these objectives. We instead aim to learn a transferable text prefix, which improves usability and generalization to a flexible range of payloads.

Raval and Verma [10] aims to find minimal edits to a passage in order to change its ranking by a retriever. Pairwise Anchor-Based Trigger [8] is most similar to our work in the goal to learn a prefix trigger, but it limits experiments to the same passage that the trigger is optimized on and uses a perturbed beam search algorithm to optimize scores rather than coordinate-descent-based prefix tuning.

8 Future Work

Generalization. It will be meaningful to find out whether the prefixes can generalize to fined-tuned document encoder (BERT) derivatives, and to fined-tuned question encoder derivatives. It is also worth evaluating on larger sets of queries and different topics from course related ones.

Robust Training. We can try varying the adversarial string positions other than being prefixes, training strings that are insensitive to placements in documents, while maintaining the effectiveness of boosting similarity scores. We may try varying retriever families and models to see whether it is possible to make the same adversarial strings effective to different retrievers. To connect with LLM jailbreaks, it can be useful to combine the

training with instruction override objectives. Contrastive training from BadRAG [13] can be adopted for selective activation.

We could also incorporate LM perplexity and negative ℓ_2 embedding objectives to hinder automatic detection methods that filter out adversarial passages in production systems.

Evaluation. We can evaluate our methods on real-world search systems, such as modifications with selective activation to Reddit post titles.

9 Conclusion

We explored methods to get universal adversarial prefixes to boost similarity scores of documents targeting the Facebook RAG models. We trained multiple successful prefixes that achieve expected performance and good generalization, which can negatively affect LLM safety according to our proposed threat model. The prefixes are evaluated on both the Wikipedia and instructions datasets, which contains documents of different topics, purposes, and styles, and the prefixes can transfer well to the instructions datasets even though they were trained only on the Wikipedia documents. It shows the possibly for the same adversarial modifications to documents to affect the RAG retrieval for various user queries, without the need to re-train for specific documents and questions. Further evaluation is still required to understand the impact on more retrievers, generators, and general LLM applications.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

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

[4] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. *arXiv preprint arXiv:2104.13733*, 2021.

[5] Zhibo Hu, Chen Wang, Yanfeng Shu, Hye-Young Paik, and Liming Zhu. Prompt perturbation in retrieval-augmented generation based large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1119–1130, 2024.

[6] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401, 2020.

[7] Zeyi Liao and Huan Sun. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *arXiv preprint arXiv:2404.07921*, 2024.

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

[9] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1075–1084, 2017.

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

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

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

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

[14] Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. Poisoning retrieval corpora by injecting adversarial passages. *arXiv preprint arXiv:2310.19156*, 2023.

[15] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043, 2023.