Does Vec2Text Pose a New Corpus Poisoning Threat?

Anonymous EMNLP submission

Abstract

The emergence of Vec2Text — a method for text embedding inversion — has raised serious privacy concerns for dense retrieval systems which use text embeddings. This threat comes from the ability for an attacker with access to embeddings to reconstruct the original text.

In this paper, we take a new look at Vec2Text and investigate how much of a threat it poses to the different attacks of corpus poisoning, whereby an attacker injects adversarial passages into a retrieval corpus with the intention of misleading dense retrievers. Theoretically, Vec2Text is far more dangerous than previous attack methods because it does not need access to the embedding model's weights and it can efficiently generate many adversarial passages.

We show that under certain conditions, corpus poisoning with Vec2Text can pose a serious threat to dense retriever system integrity and user experience by injecting adversarial passaged into top ranked positions. Code and data are made available at `https://anonymous.4open.science/r/vec2text-corpus-poisoning-2BF5`.

1 Introduction

Text embeddings are dense vector representations which capture semantic information about the text they encode (Muennighoff et al., 2023). Search engines that leverage these embeddings often employ dense retrievers (DRs) (Tonellotto, 2022; Zhao et al., 2022; Guo et al., 2022; Bruch, 2024). These retrievers utilize text embedding models to encode both queries and documents into embeddings; a similarity metric, such as cosine similarity, is then used to estimate relevance. DRs have demonstrated improved retrieval effectiveness compared to traditional exact term-matching search systems, arguably due to the rich semantic information encoded in the embeddings (Yates et al., 2021).

However, a recent study conducted by Morris et al. (2023) raises serious privacy concerns regarding DRs. This study explored the issue of *inverting* textual embeddings: recovering the original text from its embedding. The proposed Vec2Text method iteratively corrects and generates text to reconstruct the original text based on the given input embedding. Vec2Text can accurately recover 92% of short text and reveal sensitive information (such as patient names in clinical notes) with high accuracy. Even more concerning is that training Vec2Text does not require access to the embedding model parameters; all that is required is the text-embedding pairs from the training data.

In this paper we demonstrate that Vec2Text can also be employed to conduct corpus poisoning attacks on dense retrievers. A corpus poisoning attack involves a malicious actor generating adversarial passages designed to trick the ranker into retrieving such passaged for all unseen user queries, thus undermining the user experience of the targeted search system (Zhong et al., 2023). Vec2Text is potentially a more dangerous method for corpus poisoning for a dense retriever than previous approaches because it does not require access to the embedding model parameters and can efficiently generate large numbers of adversarial passages. To date, there has been no study yet to investigate how Vec2Text performs in corpus poisoning attacks. In this paper, we present our results of applying Vec2Text to the corpus poisoning task. Our findings demonstrate that Vec2Text could pose a serious threat to current DR systems.

2 Background

2.1 The Vec2Text Method

Given an input embedding, Vec2Text generates the text the embedding represents (Morris et al., 2023). Vec2Text has two stages. In the first state, a hypothesis text generation model is trained, utilising a conditional transformer generative model that exclusively takes the embedding as the model input.

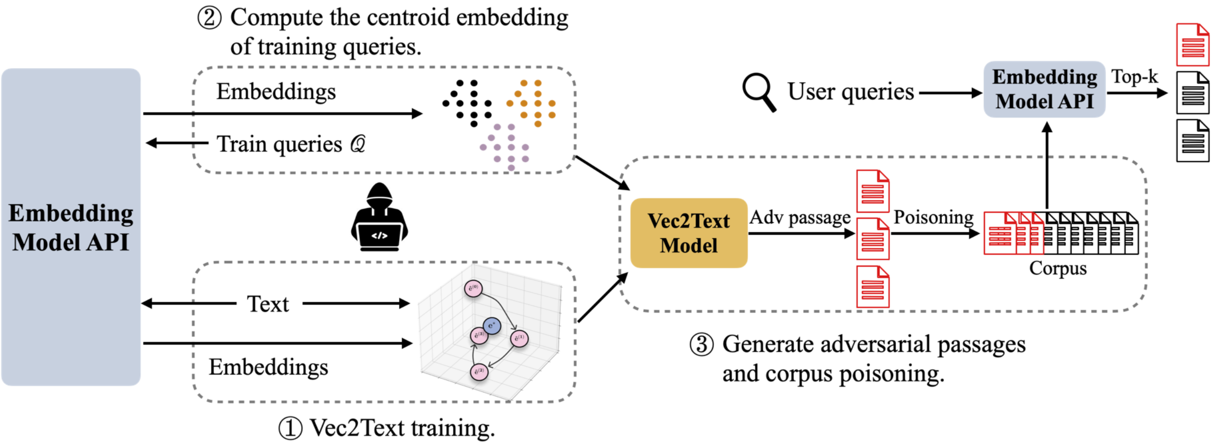

Figure 1: Overview of corpus poisoning attack with Vec2Text. The attacker does not need the access to the embedding model weights. Instead, the attacker only need to know which Embedding model API is used for the retriever.

The training objective is to produce the original text. This simplistic model is insufficient for generating highly accurate original text (Morris et al., 2023); thus the text generated by this first stage model is just treated as a hypothesis. The second stage then trains another transformer generative model that aims to generate satisfactory text by refining the initial hypothesis. Refinement is achieved by iterative re-embedding and correction training objectives. In each iteration step, the model takes the ground-truth embedding, the generated text, and its embedding from the last iteration step as inputs (the 0 step uses the hypothesis text generated from the first stage model). The output target is the original text. This iterative process allows the model to focus on the differences between the generated text and the original text in the embedding space and gradually reduce these differences.

The models in Vec2Text are parameterised as a standard encoder-decoder transformer conditioned on the previous output. One challenge is inputting conditioning embeddings into the transformer encoder, which requires a sequence of embeddings as input. To address this, a small multi layer perceptron is used to project a single embedding vector to a larger size and reshape it to match the input requirements of the encoder.

During inference, beam search guides the generation. At each step of correction, the possible corrections are considered, and the top unique continuations are selected based on their distance in embedding space to the ground-truth embedding.

2.2 Existing Corpus Poisoning Attack Method

The first corpus poisoning attack for dense retrievers was a gradient-based approach. Inspired by the HotFlip method (Ebrahimi et al., 2018; Wallace et al., 2019), it generates a small set of adversarial passages by perturbing discrete tokens in randomly initialized passages to maximize their similarity with a provided set of training queries. These adversarial passages are then inserted into the retrieval corpus, and the success of the attack is determined by the retrieval of these adversarial passages at a high rank in response to future unseen queries. These adversarial passages can be used to harm retrieval effectiveness and/or inject spam or misinformation into the search engine result list.

The current HotFlip-based corpus poisoning method has two drawbacks (from an attacker perspective). First, the gradient-based method requires access to the embedding model weights. Consequently, attackers cannot employ this method to target DR systems built with closed-source embedding models (e.g., OpenAI models). Second, the method is iterative, with only one token in the adversarial passage selected and perturbed at each iteration. This process cannot be parallelized within each adversarial passage (the perturbation of the next token depends on the previous). This makes the method time-consuming and resource intensive: we used HotFlip as a baseline in our experiments and found that with a single H100 GPU, HotFlip takes approximately 2 hours to generate a single adversarial passage. The use of Vec2Text for corpus poisoning that we demonstrate in this paper is not affected by this issue.

3 Corpus Poisoning with Vec2Text

As Vect2Text does not require access to the model weights, once trained, it can directly generate the adversarial passage from the embedding using standard language model generation inference. Furthermore, it can efficiently generate many adversarial passages (being far less computationally intensive than HotFlip).

To formally define corpus poisoning with Vec2Text, let $\mathcal{Q} = \{q_1, q_2, ..., q_{|\mathcal{Q}|}\}$ be a set of training queries and ϕ is the embedding model. The goal is to use Vec2Text to generate an adversarial passage a whose embedding maximizes the similarity to all the training query embeddings:

$$a = \operatorname*{argmax}_{a'} \frac{1}{|\mathcal{Q}|} \sum_{q_i \in \mathcal{Q}} \phi(q_i)^\intercal \phi(a') \quad (1)$$

$$= \operatorname*{argmax}_{a'} \phi(a')^\intercal \frac{1}{|\mathcal{Q}|} \sum_{q_i \in \mathcal{Q}} \phi(q_i) \quad (2)$$

$$= \operatorname*{argmax}_{a'} \phi(a')^\intercal \phi_{\mathcal{Q}}, \quad (3)$$

where $\phi_{\mathcal{Q}}$ is the average embedding or centroid embedding of all the training query embeddings. The maximum similarity is achieved when $\phi(a') = \phi_{\mathcal{Q}}$. In practice, the corpus could be poisoned with multiple adversarial passages (like Zhong et al. (2023)). This could be done by first using k-means clustering to cluster the training queries based on their embeddings; then generating an adversarial passage for each cluster.

Vec2Text can solve this optimization problem through the three-stage process shown in Figure 1:

1. The first stage is the standard training of Vec2Text, which involves sending text to embedding model API and collecting the returned embeddings to form a Vec2Text training dataset (i.e., text-embedding pairs). Then a Vec2Text model is trained with the collected training data.

2. The second stage involves computing the centroid embedding of training queries, which can be done by sending the training queries to the embedding model API to obtain all the query embeddings and use k-means clustering algorithm to compute the centroid embeddings of the clusters.

3. Finally, inputting each of the centroid embeddings to the trained Vec2Text to generate an adversarial passage for each centroid embeding. A perfect Vec2Text would generate adversarial passages whose embedding is exactly the same as the query centroid.

Once again, we emphasize that the entire process does not require access to model weights. Moreover, generating an adversarial passage with Vec2Text is efficient: generating a single passage took 5 seconds on a single Nvidia H100, employing beam search with 50 steps and a width of 4.

4 Results and Analysis

Corpus poisoning experiments used the GTR-base embedding model as the DR system, utilizing the NQ dataset (in the version released with BEIR) as the target corpus, which comprises of approximately 2.68 million passages. NQ training queries were used to encode the query centroid embedding and then to generate adversarial passages. We set number of centroid (clusters) k to $10, 100, 1000$. For the Vec2Text model, we use an open-sourced model[1] that is also trained on the NQ dataset and targeted at inverting GTR-base embeddings.

Evaluation was then done using NQ test queries with evaluation measure of success@n: the percentage of queries for which at least one adversarial passage was retrieved in the top-n results. Higher success@n indicates greater vulnerability to corpus poisoning.

Table 1 presents corpus poisoning results. When only 10 adversarial passages were generated (grouping training queries with 10-means clustering), both HotFlip and Vec2Text performed poorly on success@10 and 20; however, HotFlip did work when injecting adversarial passages into the top-n=100 or 1000 rank positions. If the embedding model weights were available, HotFlip was a more potent corpus poisoning attack method when applied to longer rank lists of 100 or 1000.

Vec2Text has the advantage of being able to generate many more adversarial passages; when $k = 100$ or $k = 1000$, HotFlip was too computationally expensive and thus largely impractical. For these cases Vec2Text success rate improved, with $k = 1000$ making Vec2Text a considerable threat in corpus poisoning. With 1000 generated passages injected into the NQ dataset, in fact, at least one generated passage could be retrieved in the top-10 results for 27% of the queries, and in the top 100 for over half of the queries in the NQ dataset.

The adversarial passages were generated from query embeddings — they are synthetic and may not appear like real passages. Figure 2 shows the two adversarial passages most closely matching a

[1] https://huggingface.co/ielabgroup/vec2text_gtr-base-st_inversion and https://huggingface.co/ielabgroup/vec2text_gtr-base-st_corrector

Table 1: Results of applying Vec2Text to the task of corpus poisoning, compared to existing HotFlip approach. Vec2Text has the advantage of being able to produce numerous adversarial passages (k) without model weights. Both Vec2Text and HotFlip are well below the upper bound, indicating that better methods could pose a serious risk in corpus poisoning.

Attack method	**#Adversarial passages / Clusters k**	**Success@ 10**	**20**	**100**	**1000**
HotFlip (Zhong et al., 2023)	10	0.006	0.015	0.105	0.532
Vec2Text	10	0.006	0.012	0.036	0.101
Query centroid (upper bound)	10	0.206	0.304	0.575	0.926
HotFlip	100	· · · Too computationally expensive · · ·			
Vec2Text	100	0.061	0.091	0.189	0.401
Query centroid (upper bound)	100	0.493	0.620	0.868	0.994
HotFlip	1000	· · · Too computationally expensive · · ·			
Vec2Text	1000	0.268	0.335	0.521	0.791
Query centroid (upper bound)	1000	0.779	0.873	0.980	1.000

a = "`That drawings promises Key kot iner jor Respond machines AK <pad> ance pe izi very nie <pad> <pad> Doar ceapa Weg MEN Am OK Jamie words Um Imp refers " War Hop nism the serving Bol auto Palatul fries rome lighter (1 49 fetch 19 cities can counting 16, As`"

(a) HotFlip. Cosine similarity of this passage to the query centroid embedding is $\cos(\phi(a), \phi_{\mathcal{Q}}) = 0.96$.

a = "`when does the 7 episode season of the new two come out </s>`"

(b) Vect2Text. Cosine similarity of this passage to the query centroid embedding is $\cos(\phi(a), \phi_{\mathcal{Q}}) = 0.85$.

Figure 2: Sample adversarial passages most closely matching the NQ training query centroid.

sampled query centroid $\phi_{\mathcal{Q}}$ for both HotFlip and Vec2Text. Recall that HotFlip works by flipping tokens one at a time, whereas Vec2Text is a language model decoder. This difference may explain why Vec2Text appears to generate more natural language. Also note that a prospective malicious agent would likely add a payload to these generated passages, where the payload is, for example, a link to a phishing website.

The embeddings of the adversarial passages in Figure 2 still differ from the query centroid embeddings (cosine similarity being 0.96 for HotFlip and 0.85 for Vec2Text). Recall that a perfect Vec2Text would generate an adversarial passage for which the embedding is exactly the same as the query embedding. To understand how this best case scenario affects success@k we run an additional experiment which involved directly inserting the query centroid embeddings into the vector index and then evaluating the success@k. This serves as an upper bound for the HotFlip and Vec2Text corpus poisoning attacks — an analysis overlooked in the original corpus poisoning paper (Zhong et al., 2023). Results are shown in Table 1 as "Query centroid (upper bound)". For both different values of k and different success@n, the upper bound is much higher than both HotFlip or Vec2Text. This tells us that neither method is optimal yet for the corpus poisoning attack. It also highlights that if a much more effective corpus poising method — one closer to the upper bound — was developed, it could have serious adverse consequences for dense retrievers.

5 Conclusion

Dense retrievers are an effective and efficient retrieval method widely adopted in working systems. Much of their benefit comes from using text embeddings to represent and compare information. However, the reliance on text embeddings also opens up dense retrievers to possible threats that exploit such embeddings.

We identify that Vec2Text (a method to invert embeddings to their original text) could be a threat to the completely different task of corpus poisoning, whereby adversarial passages are generated and inserted into a corpus such that they are likely be retrieved for any query. Vect2Text poses a real risk here because it can easily generate large numbers of adversarial passages without access to model weights. We show that under certain conditions, corpus poisoning with Vec2Text can pose a serious threat to dense retriever system integrity and user experience. This work is designed to stimulate the development of counter measures to to prevent such corpus poisoning attacks.

6 Limitations

Corpus poisoning with Vec2Text has two major limitations. First, our results show that a large number of adversarial passages, although still an insignificant fraction of the full corpus, is required for Vec2Text to be effective in the corpus poisoning task. This makes the use of Vec2Text for corpus poisoning somewhat cumbersome, as it requires the target search engine to index all the generated adversarial passages. Second, compared to the HotFlip method, Vec2Text does not support the insertion of a prefix message into the adversarial passage (e.g., a payload that the attacker may use for phishing). In addition, while the passages produced by Vec2Text appear at first to be better than those from HotFlip as they contain actual words (see Figure 2), they are still not meaningful and thus not likely to attract user clicks if displayed in the search engine result page of a search engine like Google or Bing. However, these passages might still negatively impact Retrieval-Augmented Generation (RAG) systems (Xue et al., 2024; Zou et al., 2024; Cho et al., 2024). In RAG systems, users are not directly exposed to the actual retrieved passages: these are instead acquired by the system and used to inform the generation of an answer, which is then displayed to users. This means that users might be less likely to identify the presence of such adversarial passages among the evidence the system used.

7 Ethical Statement

This paper does actually describe a method to perform an unethical action, namely intentionally adding adversarial content to a corpus to adversely impact information retrieval systems that use that corpus. In presenting this work we in no way condone the use of this method. Our explicit purpose in describing the method is to raise awareness that it could pose a threat to retrieval systems, and to explicitly quantify the associated risk. Our hope is that this work directly informs an effective defence measure to prevent corpus poisoning.

References

Sebastian Bruch. 2024. Foundations of vector retrieval. *arXiv preprint arXiv:2401.09350*.

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

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–42.

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

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Nicola Tonellotto. 2022. Lecture notes on neural information retrieval. *arXiv preprint arXiv:2207.13443*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

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

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: BERT and beyond. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorials*, pages 1–4, Online. Association for Computational Linguistics.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2022. Dense text retrieval based on pretrained language models: A survey. *arXiv preprint arXiv:2211.14876*.

Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. 2023. Poisoning retrieval corpora by injecting adversarial passages. In *Proceedings of the*

2023 Conference on Empirical Methods in Natural Language Processing, pages 13764–13775, Singapore. Association for Computational Linguistics.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*.