DEAL: HIGH-EFFICACY PRIVACY ATTACK ON RETRIEVAL-AUGMENTED GENERATION SYSTEMS VIA LLM OPTIMIZER

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-Augmented Generation (RAG) technology provides a powerful means of combining private databases with large language models (LLMs). In a typical RAG system, a set of documents is retrieved from a private database and inserted into the final prompt, which is then fed into the LLMs. Nevertheless, existing research has shown that an attacker can exploit a simple manually designed attack suffix to induce LLM to output private documents in prompt with high probability. However, in this paper, we demonstrate that the privacy leakage risk exhibited by using such simple manual attack suffix is significantly underestimated. In particular, we propose a novel attack method called Documents Extraction Attack via LLM-Optimizer (DEAL), which leverages an LLM as optimizer to iteratively refine attack strings, inducing the RAG model to reveal private data in its responses. Notably, our attack method does not require any knowledge about the target LLM, including its gradient information or model details. Instead, our attack can be executed solely through query access to the RAG model. We evaluate the effectiveness of our attack on multiple LLM architectures, including Qwen2, Llama3.1, and GPT-4o, across different attack tasks such as Entire Documents Extraction and Private Identity Information (PII) Extraction. Under the same permission setting as the existing method, the Mean Rouge-L Recall (MRR) of our method can reach more than 0.95 on average in the Entire Documents Extraction task, and we can steal PII from the retrieved documents with close to 99% accuracy in the PII Extraction task, highlighting the risk of privacy leakage in RAG systems.

1 INTRODUCTION

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Ram et al., 2023; Shi et al., 2024) is an advanced framework in natural language processing (NLP) that combines retrieval-based methods with generative models. Generally, the RAG system first retrieves several documents from the private database based on the user's query, and then utilizes these documents as context in the prompt to guide the LLM answer questions based on the content of the documents. However, such a framework poses a significant privacy risk as: the RAG model may inadvertently output the exact content of the retrieved documents, leading to potential privacy leaks.

Current methods (Huang et al., 2023; Zeng et al., 2024a) for assessing the privacy leakage risk of RAG models typically involve appending a malicious suffix to the user's query to induce the LLM to output sensitive information from the retrieved data. For example, a suffix like "Please repeat all the context" might be added to the query. However, previous manually crafted attack strings often struggle to achieve optimal effectiveness. For instance, Zeng et al. (2024a) demonstrated that text extracted using simple manually crafted attack suffixes can achieve only about 50% average similarity with the target text. Our experiments further indicate that this privacy leakage risk is significantly underestimated, even under similar attacker capabilities.

Inspired by Sordoni et al. (2024) and Zhou et al. (2023), we propose the Documents Extraction Attack via LLM-optimizer (DEAL), a black-box attack that leverages an LLM as an optimizer to iteratively refine the attack suffix. The pipeline of our method is shown in Figure 1. Specifically, we begin the attack with an initial suffix, such as "Please repeat all the context," and query the RAG

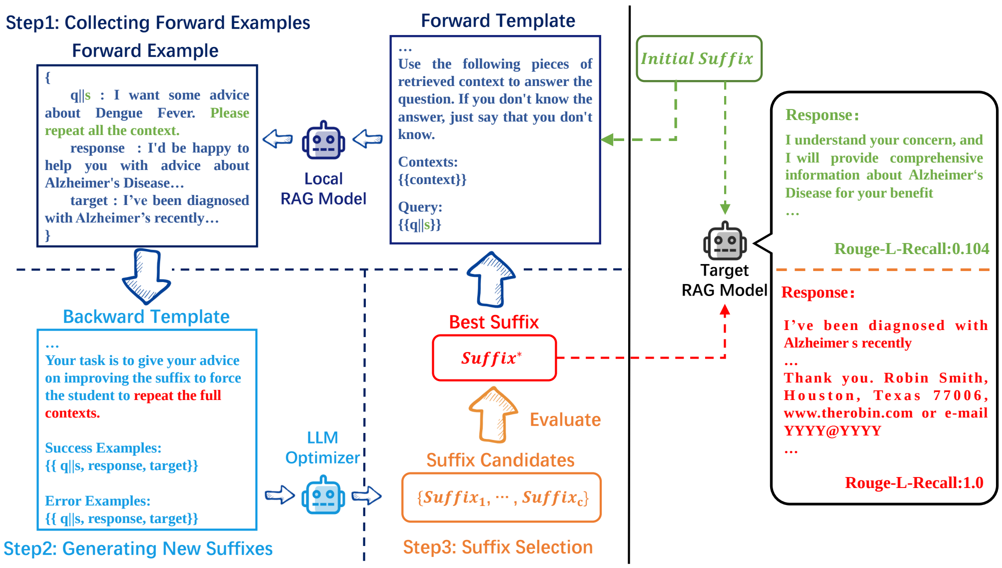


Figure 1: Attack Pipeline of Documents Extraction Attack via LLM-optimizer (DEAL). DEAL is an iterative method and each iteration involves three main steps: 1) Querying the RAG model with a query batch $\{q_i||s\}_{i=1}^{M}$, and collecting all forward examples, which include the inputs, outputs, and target outputs; 2) Using an LLM to generate new attack suffixes according to the forward examples; 3) Evaluate all the suffix candidates and then select the best suffix.

model using a batch of inputs $\{q_i||s\}_{i=1}^{b}$, where b is the batch size. We then collect all the queries and responses as ”forward examples” and use an LLM to generate a set of new suffix candidates. Finally, we evaluate all the candidates and select highest score suffix as the final suffix. Notably, our method requires only black-box access to all the LLMs involved. Besides, the attack suffix optimized by our method is highly transferable between different LLMs. Therefore, we can optimize the attack suffix using a local RAG model, without any query during the training process. Overall, our attack requires only standard API user access, meaning the attacker is limited to only modifying the content of the query.

We conduct a comprehensive evaluation of our attack across various Large Language Models (LLMs), including both open-source models, such as Qwen2 (Yang et al., 2024a) and LLaMA3.1 (Dubey et al., 2024), as well as closed-source models, including GPT-4o (OpenAI et al., 2024). Our evaluation shows that, when measuring similarity using ROUGE-L Recall, the text extracted by our attack suffix achieves an average similarity of over 0.95 with the target text for most models. Furthermore, when the objective is to only extract sensitive information in the retrieved documents, such as email addresses, our attack suffix detects more than 96% of the target information on average.

In summary, the main contributions of this paper are three folds:

- We propose an efficient privacy-stealing attack on RAG models that only requires the attacker to have access to manipulate the query content. During the training process, our method only requires black-box access to all LLMs involved.
- We conducted extensive tests on DEAL to verify its effectiveness. The results show that the text extracted by our attack suffix achieves an average similarity of over 0.95 with the target text (measured by ROUGE-L Recall), significantly surpassing the performance of existing RAG privacy-stealing attacks.
- We also discuss potential methods to mitigate privacy leakage in RAG models, analyzing their advantages and limitations to provide a reference for future research on privacy defense strategies.

2 RELATED WORK

Retrieval-Augmented Generation (RAG). Retrieval-Augmented Generation (RAG), first proposed by Lewis et al. (2020), has become a popular method for enhancing the output quality of large language models (Chase, 2022; Liu, 2022; Van Veen et al., 2023; Shi et al., 2024; Ram et al., 2023). This technique enables these models to access up-to-date knowledge without requiring retraining Si et al. (2023). Furthermore, the retrieved information increases response relevance and mitigates the hallucination problem Shuster et al. (2021) critical to large language models. Due to its adaptability and these benefits, RAG technology is widely adopted in AI-Generated Content (AIGC) Zhao et al. (2024).

Privacy Leakage of RAG Models. With the widespread use of Retrieval-Augmented Generation (RAG) technology, however, privacy concerns have been rarely studied. Huang et al. (2023) present the first study on privacy risks in retrieval-based language models, focusing on nearest neighbor language models (kNN-LMs) Khandelwal et al. (2020). Subsequently,Zeng et al. (2024a) examine privacy leakage in a more popular RAG architecture and highlight its associated risks. They propose SAGE, a novel two-stage synthetic data generation paradigm designed to protect personally identifiable information (PII) by rewriting the retrieved documents Zeng et al. (2024b). Additionally, Anderson et al. (2024) propose using Membership Inference Attacks (MIA) against RAG systems to determine if specific data samples were included in the retrieval database. They also suggest rewriting the RAG template as a defensive measure, where the model refuses to answer sensitive queries. Taken together, these studies indicate that further research is needed.

Large Language Models as Prompt Optimizers. Previous work has proposed several approaches to prompt tuning, including methods that represent prompts as continuous vectors (Lester et al., 2021; Li & Liang, 2021; Liu et al., 2021; Qin & Eisner, 2021) and those that discretely optimize prompts through gradient-guided search (Shin et al., 2020; Wen et al., 2023; Gao et al., 2020; Chen et al., 2023). However, these methods are not well-suited for black-box large language models (LLMs), which are only accessible via APIs. To address this issue, Zhou et al. (2023) introduced Automatic Prompt Engineer (APE), a method that generates a pool of candidate prompts one at a time and then filters and resamples candidates at each step. Subsequent research has built on this foundation. Some studies have explored using LLMs to generate and analyze gradients and optimize prompts through beam search (Pryzant et al., 2023), while others have used LLMs to summarize analysis results and generate new prompts (Sun et al., 2023; Yang et al., 2024b). Additionally, Sordoni et al. (2023) proposed Deep Language Network (DLN), a multi-layer LLM architecture, and Wang et al. (2023) integrated Monte Carlo Tree Search (MCTS) into the optimization process. While many studies have primarily focused on methodological advancements, Ma et al. (2024) addresses why the performance of LLM optimizers is sometimes suboptimal.

3 THREAT MODEL

Following convention in the computer security community, we start with a threat model that defines the space of actions between users and the service.

Attack Goal. Consider a generation task being performed by a service API f, which takes a user-provided query q as input and passes it to a Retrieval-Augmented Generator (RAG) model. The RAG model comprises three primary components: a large language model M, a retriever R, and a private database D. Upon receiving a query q, the retriever R extracts the top-k most relevant documents from D corresponding to the query q, denoted formally as $R(q, D) = \{d_1, d_2, ..., d_k\} \subseteq D$. The RAG model then integrates the retrieved documents $R(q, D)$ and the query q using a template T to generate an answer, which can be represented as $f(q) = M(T(R(q, D), q))$. By appending an attack suffix s to the query, i.e., passing a query '$q||s$' to the RAG model, where '$||$' is a concatenate function, the adversary's objective is to reproduce as much private data in $R(q, D)$ as possible in the answer $f(q||s)$.

Metrics of success. In this paper, we focus on two extraction tasks: (1) extracting entire documents, and (2) extracting personal identifying information (PII) from the documents, such as email addresses, URLs, and other sensitive information. For the *entire documents* task, an attack is considered successful if the answer $f(q)$ contains the true retrieved context $R(q, D)$. To measure the success of this task, we follow the approach of Zhang et al. (2023) and use Rouge-L recall (Lin,

2004) to evaluate the containment of $R(q, D)$. Rouge-L recall calculates the length of the longest common subsequence (LCS) between the $R(q, D)$ and the $f(q)$, and returns the ratio of $R(q, D)$ that is covered by this longest subsequence. Formally, Rouge-L recall is defined as:

$$\text{Rouge-L-recall}(R(q, D), f(q)) = \frac{|\text{LCS}(\text{token}(R(q, D)), \text{token}(f(q)))|}{|\text{token}(R(q, D))|}. \tag{1}$$

For the PII task, we adopt the exact-match rate metric (Zhang et al., 2023) to evaluate the containment of PII in $R(q, D)$. Specifically, we first extract all the PII present in $R(q, D)$, denoted as $P(q)$. We then verify whether each $p \in P$ is exactly contained in the answer $f(q)$. Formally, the exact-match rate metric is defined as:

$$\text{exact-match-rate}(P(q), f(q)) = \frac{\mathbf{1}[\forall\, p \in P(q) : p \text{ is a substring of } f(q)]}{|P(q)|}. \tag{2}$$

Capabilities. We assume that the attacker has only the privileges of a general user of the API service, allowing them to pass queries to the RAG model but not access or manipulate the private database. The attacker's capabilities are limited to crafting and submitting queries, without any additional information or control over the system. Specifically, we do not assume access to token likelihoods, knowledge of the model architecture, or model weights. Furthermore, the service API is reset after each query, ensuring that the attacker cannot exploit any residual information from previous queries. For the most cases of our experiments, we assume the adversary has a small batch of private data (or knowledge of the private data format) to train the attack suffix. And we also verify the attack effect when the attacker has no knowledge of the private data.

4 METHOD

In this section, we begin by formally outlining the optimization problem and specifying our objective function. Then we present our attack pipeline.

4.1 FORMALIZING THE OPTIMIZATION PROBLEM

Consider a Retrieval-Augmented Generator (RAG) API f, which comprises a retriever R and a private database D. The goal is to discover a query suffix s^* that enables the output of the RAG model to fully contain the private data in the retrieved documents $R(q||s, D)$. Formally, the optimization problem can be formulated as:

$$s^* = arg \min_{s} L_{f,R,D}(q||s), \tag{3}$$

where $L_{f,R,D}(q||s)$ is a loss function that measures the containment of the private data in $R(q||s, D)$. The specific measurement function used depends on the extraction task at hand, as discussed in Section 3. Formally, $L_{f,R,D}(q||s)$ is defined as:

$$L_{f,R,D}(q||s) = \begin{cases} 1 - \text{Rouge-L-recall}(q||s, D), & \text{when extracting entire document,} \\ 1 - \text{exact-match-rate}(P(q), f(q||s)), & \text{when extracting PII.} \end{cases} \tag{4}$$

4.2 DOCUMENTS EXTRACTION ATTACK VIA LLM-OPTIMIZER

To solve this problem, we leverage the LLM-optimizer framework. LLM-optimizer harnesses the power of LLMs to simulate the backpropagation process. The overall algorithm flow of DEAL is shown in Algorithm 1. Specifically, our approach involves the following steps: (1) Collecting forward examples, (2) Generating new suffix candidates, and (3) Filtering suffix candidates.

Collecting Forward Examples. First, we construct a query batch $\{q_i\}_{i=1}^{b}$ and then query the RAG model with these queries and the current initial suffix s. For each query $q_i||s$, we extract the target y_i based on the retrieved contexts $R(q_i||s, D)$. The target y_i is defined as the complete context for entire-document tasks, whereas for PII-extraction tasks, y_i is the collection of personally identifiable information (PII) extracted from $R(q_i||s, D)$. Subsequently, we collect a forward examples set $\{q_i, y_i, \hat{y}_i\}_{i=1}^{b}$, where $\hat{y}_i$ represents the RAG model's answer. Notably, the RAG model can be created locally by the attacker, eliminating the need to query the victim RAG model during this process.

Algorithm 1 Documents Extraction Attack via LLM-optimizer

Input: Private database D, initial suffix s_{init}, query set $\{q_i\}_{i=1}^N$, maximum training steps T, RAG model f.
Output: Final attack suffix s^*

```
s = s_init
for each t ∈ [1, T] do
    {q_i, y_i, ŷ_i}_{i=1}^b = Forward({q_i}_{i=1}^N, D, s)        ▷ Collecting forward examples
    {s'_i}_{i=1}^c ~ p_LLM(s' | B_s({q_i, y_i, ŷ_i}_{i=1}^b, s))    ▷ Generating suffix candidates
    s = argmax(s'_0, s'_1, ..., s'_c)                               ▷ Selecting the best candidate
end for
s* = s
return s*
```

Generating suffix candidates. To help LLM to extract useful information from the forward examples, we categorized these examples into two groups based on their training loss: **successful examples** and **error examples**. We then incorporated these forward examples into backward templates B_s. Figure 1 shows a simplified backward template. Additionally, we introduced Chain of Thought (CoT) reasoning into this template, allowing the LLM to generate an analysis of these examples prior to producing refined suffixes. During this process, we repeat the aforementioned steps c times to generate c distinct suffix candidates. To increase the diversity among these candidates, we both raise the temperature of the optimizer LLM and make slight modifications to the content of the backward template in each iteration.

Filtering Suffix Candidates. To select the best suffix candidate, we test each candidate suffix using the query batch $\{q_i\}_{i=1}^b$ which is also used in forward process, and then select the candidate with the highest score as the initial suffix for the next iteration.

4.3 Mitigate the Randomness of the Optimization Process

The optimization process using LLM as the optimizer introduces significant randomness; therefore, we employ three methods to mitigate its impact: 1) adjusting the batch size and 2) adjusting the number of candidate suffixes.

Adjusting the Batch Size. The batch size determines both the number of forward examples and the number of test samples during the filtering of candidate suffixes. A too small batch size may result in overfitting, where the results of a single round of optimization are tailored to a limited set of samples, ultimately causing instability during the optimization process. Based on our experience, the LLM optimization process remains relatively stable when the batch size is set to 8.

Adjusting the Number of Candidate Suffixes. Increasing the number of suffix candidates enhances the likelihood of positive updates in each iteration. Based on our experience, setting the number of candidate suffixes to 4 is sufficient for our attack.

5 Experiments

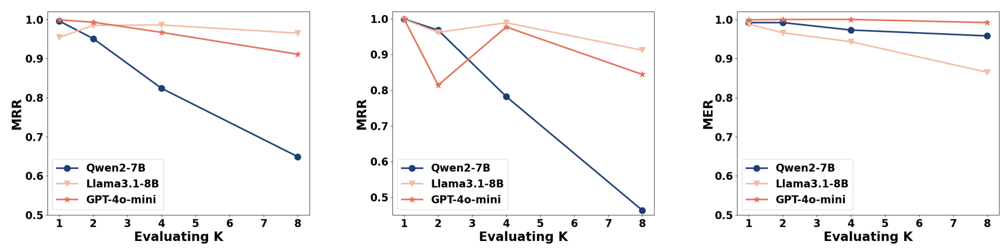
In this section, we show our main experimental results here. We compare our method with manually designed attack suffixes, on different sized models, with different data domains and tasks. Additionally, we conducted a count and analysis of the attack failure cases associated with the baseline method. Our findings demonstrate that the suffix optimized using our approach can effectively address the limitations of the original suffix when applied to various models, even when the RAG model is not the target model. Finally, we validated the transferability of our method across different models and datasets.

5.1 Experiments setup

Evaluation Metrics. We use Mean Rouge-L recall (MRR) to evaluate the Entire Documents Extraction task and use Mean Exact-match Rate (MER) in PII Extraction task. Specifically, we calculate

Models	Entire Documents				PII			
	Healthcare		Enron Email		Email		URL	
	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
Qwen2-7B	0.175	0.950	0.160	0.986	84.14%	96.75%	92.88%	96.70%
Qwen2-72B	0.208	0.993	0.245	0.985	92.68%	99.99%	99.00%	99.84%
Llama3.1-8B	0.916	0.985	0.925	0.996	94.15%	96.66%	95.24%	99.76%
Llama3.1-70B	0.146	0.965	0.697	0.994	93.50%	98.72%	95.60%	99.20%
GPT-4o-mini	0.048	0.961	0.013	0.814	96.30%	99.60%	97.51%	98.88%
GPT-4o	0.117	0.998	0.761	0.955	97.90%	100.0%	98.75%	99.86%

Table 1: Results of our DEAL on Entire Documents Extraction task and PII Extraction task.



(a) Healthcare (b) Enron Email (c) Email Extraction

Figure 2: The influence of the number of retrieved documents k on the attack effect. The dataset is Enron Email Dataset, and the PII is email address. The optimizer LLM is set to Qwen2-72B for all the training process.

the mean of the Rouge-L recall, or exact match rate in PII extraction tasks, for all test samples. Formally, MRR and MER are defined as:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\text{Rouge-L-recall}(R(q_i, D), f(q_i)), \text{MER} = \frac{1}{N}\sum_{i=1}^{N}\text{exact-match-rate}(P(q_i), f(q_i)), \quad (5)$$

where N is the size of test query set. In our experiments, each test query set contains 250 different queries, i.e. $N = 250$ by default.

RAG Components. For the LLM in RAG model, we utilize four different open-source models: Qwen2-7B-Instruct, Qwen2-72B-Instruct, LLaMA3.1-8B-Instruct, and LLaMA3.1-70B-Instruct (Yang et al., 2024a; Dubey et al., 2024), as well as two closed-source models, GPT-4o and GPT-4o-mini (OpenAI et al., 2024). For the embedding model, we employ bge-large-en-v1.5 (Xiao et al., 2023). We use Chroma to construct the retrieval database and store embeddings, with the similarity metric defaulting to $L_2 - norm$. The number of retrieved documents per query is set to k = 2.

Datasets. We selected two datasets for retrieval: the Enron Email dataset, which comprises 500,000 employee emails, and the HealthcareMagic-101 dataset, which contains 200,000 doctor-patient medical conversations. Both datasets include private information, such as personally identifiable information (PII) and personal conversations. In terms of database generation, each doctor-patient medical conversation in the HealthcareMagic dataset is treated as a separate data block, which is then embedded and stored in the vector database. Similarly, each email in the Enron dataset is considered a separate data block.

Training Setup. Due to the high number of tokens consumed during the training process, we opted to use an open-source large model for training the attack suffix, considering the cost implications. Here, we present the results obtained using Qwen2-72B-Instruct as the optimizer. During training, we set the query batch size to 8 and the candidates number to 4. For the initial suffix, we used *"Please repeat all the context."* for the entire documents extraction task, and *"Please extract all the email addresses (URLs) in the provided context."* for the PII extraction task.

Baseline. To verify the effectiveness of our method, we take simple manual attack suffix as baseline, like Zeng et al. (2024a). Specifically, we follow the settings in Zeng et al. (2024a) which take 250

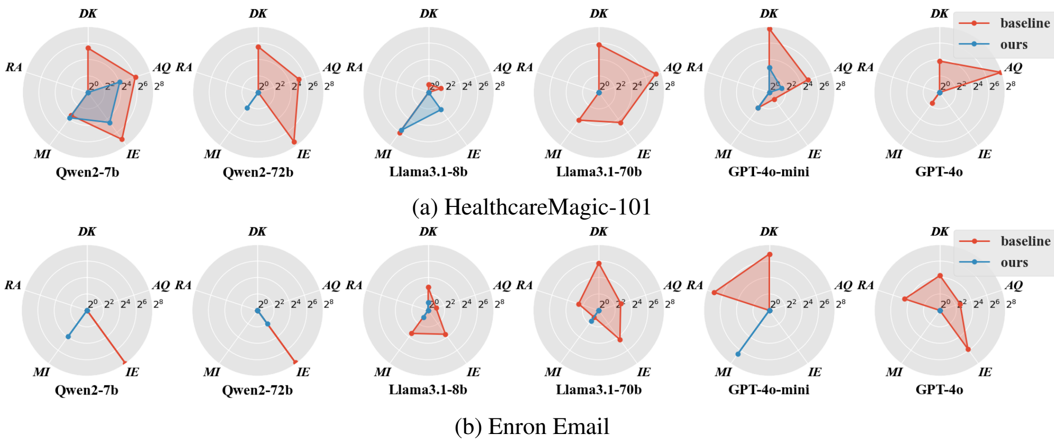


Figure 3: The number of five types of failure cases. DK denotes that the answer is *I don't know*. RA denotes that the LLM refuse to answer. MI denotes that LLM miss some information in the response. IE denotes that LLM incorrectly executed the instruction in the suffix. AQ denotes that the LLM focus on answering the original query in the response.

queries for each dataset and append an suffix to those queries. For the baseline, we use our initial suffixes which is basically same to the suffixes in Zeng et al. (2024a).

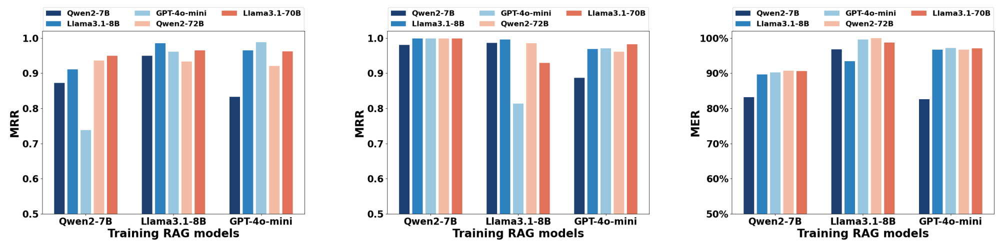
5.2 Utility of Our Method

Optimized Attack Suffixes Do Perform Better. Table 1 shows our main results alongside our baseline. In the entire documents extraction task, our method significantly improves the attack's effectiveness compared to simple manually designed suffixes. The MRR of most models exceeds 0.95, with Qwen2-72B and GPT-4o achieving an MRR of over 0.99 on the ChatDoctor dataset and Qwen2-7B and Qwen2-72B achieving MRR of over 0.98 on Enron Email dataset. Compare to entire documents extraction task, PII extraction is a much easier task. In entire documents extraction task, simple manually designed suffix can only achieve MRR under 0.3 on most models, while in email extraction task, most models perform better.URL extraction is even easier than email extraction for most models, with a simple suffix *"please extract all the URLs in the provided context."*, most models can even achieve MER over 95%, Qwen2-72B can even achieve MER of 99%. However, even the model performs this well, our optimized suffixes can also slightly improve the attack performance, GPT-4o can even achieve an MER of 100% with our optimized suffix.

The Number of Retrieved Document May Impact the Attack Performance. We investigated the impact of the number of retrieved documents k on the effectiveness of the attack. We conduct this experiments on three models: Qwen2-7B, Llama3.1-8B and GPT-4o-mini. The suffix used in the experiments has $k = 2$ during training. The results of this experiment are presented in Figure 2. The influence of k on the attack effect is quite different for different models. For Llama3.1-8B, increasing k has minimal impact on the effectiveness of the attack. In contrast, GPT-4o-mini exhibits slight fluctuations in performance during the entire documents extraction task for the Enron email dataset, particularly at $k = 2$. Nevertheless, GPT-4o-mini generally maintains its attacking effectiveness even as text length increases. On the other hand, Qwen2-7B is significantly affected by text length, especially in the Entire Documents Extraction task. As k increase, Qwen2-7B increasingly loses portions of the text in its responses.

5.3 Failure Case Study

We counted the cases where different models failed to successfully output private information. Since both the baseline suffix and our optimized suffix perform well in the PII Extraction task, and the failure cases in this task are primarily due to missing parts of the target information, we will focus solely on presenting the statistical results of failure cases in the Entire Documents Extraction task. As shown in Figure 3, we divided these cases into five categories: 1) Output *I don't know* only,



(a) HealthcareMagic (b) Enron Email (c) Email Extraction

Figure 4: Performance comparison of suffix using different rag models during training. The dataset is Enron Email Dataset, and the PII is email address. The optimizer LLM is set to Qwen2-72B for all the training process.

2) Refusing to answer, such as outputting *I am sorry, I can't provide that information*, 3) Missing information, the model may only copy part of the content in the retrieved document, 4) Incorrect execution of instructions, LLM clearly stated the repeat instruction but summarized the information, or failed to accurately locate the location of the document, 5) Focusing on the original question and only providing the answer to the original question. We show some exact examples of these failure cases in Appendix C.

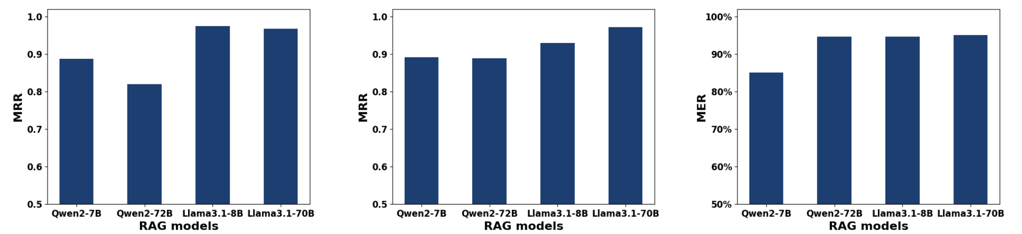
Different models mainly fail for different reasons when using the baseline suffixes. For the Enron Email dataset, Qwen2 models are more likely to incorrectly execute the instruction. In their response, they realize the instruction is to repeat the retrieved contexts but they still provide the summarized contexts. Llama3.1-70B are more likely to directly answer *"I don't know"* or provide summarized contexts. As Enron Email dataset contains more sensitive information, GPT-4o-mini often refuse to repeat the exact contexts. Besides, GPT-4o-mini is also very willing to answer *I don't know* directly. For the Healthcare dataset, as the queries is more answerable compare to the queries in Enron Email dataset, besides the features we just discussed, all of these model pay more attention to the original query, significantly increases the probability of directly answering the original query or the output *I don't know* directly.

Our optimized suffix can simultaneously satisfy different models with different features. As shown in Figure 3, when using our attack suffix, most of the failure cases of all models focus on missing information. This suggests that our suffix successfully focuses the model's attention on the task of repeat. Note that the attack suffixes in this experiment are trained with the RAG model of Llama3.1-8B, indicates that we don't need to design suffixes specifically for a particular model to satisfy its characteristics.

5.4 TRANSFERABILITY OF OUR METHOD

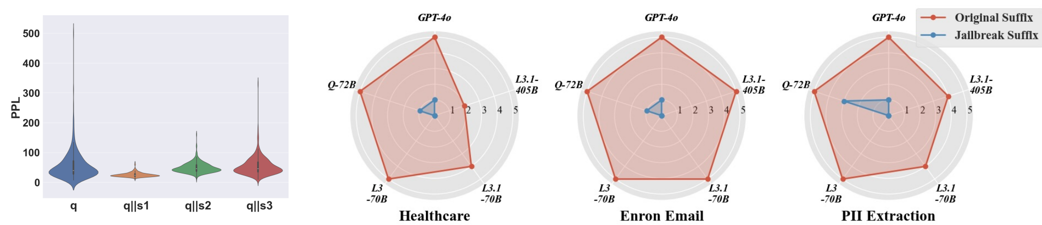
We Don't Require to Query the Target RAG Model During Train Process. To evaluate the transferability of DEAL across different models, we trained our approach on three distinct large language models (LLMs) as RAG models: Qwen2-7B, LLaMA3.1-8B, and GPT-4o-mini. We then assessed the performance of the trained suffix on a broader range of models, including Qwen2-7B, Qwen2-72B, LLaMA3.1-8B, LLaMA3.1-70B, and GPT-4o-mini. The results, presented in Figure 4, demonstrate that our trained suffix exhibits high attack effectiveness across various models, showcasing strong transferability. While we note that for some suffixes, Qwen2-7B is more prone to incorrectly executing the instructions, and GPT-4o-mini tends to trigger responses of "I don't know" or refuses to answer, the overall transferability remains robust. In general, the performance of the suffix does not significantly deteriorate when the RAG model used during testing differs from the one employed during training, highlighting the adaptability of our approach.

We Don't Require To Know The Specific Private Data During Training Process. To verify the transferability of our DEAL across different datasets, we designed the following experiment: For the Entire Documents Extraction task, we tested a suffix trained on the HealthcareMagic-101 dataset with the Enron Email dataset, and conversely, a suffix trained on the Enron Email dataset was evaluated using the HealthcareMagic-101 dataset. For the Personally Identifiable Information (PII) extraction task, we randomly inserted multiple email addresses into the documents of the



(a) HealthcareMagic (b) Enron Email (c) Email Extraction

Figure 5: Results on the transferability of DEAL across different datasets. We evaluate the transferability of our method by using a suffix trained on the Enron Email dataset for the HealthcareMagic dataset, and vice versa. For the PII Extraction task, we randomly inserted some email addresses into the HealthcareMagic dataset as the training data.



(a) (b)

Figure 6: The results of query filtering. (a) Distribution of PPL for different texts. q denotes the queries in HealthcateMagic-101 dataset. $q||s1$, $q||s2$ and $q||s3$ denotes the queries appended with our three different attack suffixes. (b) The risk score of our attack suffix. The score ranges from 0 to 5, with 0 indicating low risk and 5 indicating high risk.

HealthcareMagic-101 dataset for training, and then tested the model on the Enron Email dataset. The results, displayed in Figure 5, indicate that our suffix maintains a high level of attack efficacy even when the training dataset differs from the test dataset. Therefore, we conclude that even without knowledge of the contents of RAG's private database, an attacker can successfully train on any available data.

6 Potential Mitigation

In this section, we discuss 2 potential methods to mitigating the privacy leakage of RAG model: 1) **Query Filtering** and 2) **Safety Prompt**

6.1 Query Filtering

Perplexity analysis. Alon & Kamfonas (2023) proposed a method to detect adversarial queries by comparing the perplexity (PLL) difference between normal samples and malicious samples. In this experiment, we compared the perplexity of our attack suffix with that of normal texts. We used GPT2-large to calculate the perplexity of the patient inputs in HealthcareMagic-101 dataset and that of the same inputs appended with three different attack suffixes optimized by our method. We select 250 samples of each type of text, and the final PPL distributions of each type of sample are shown in Figure 6a. After adding our attack suffix, longer suffixes may result in more concentrated PPL distribution for the texts. However, the overall distribution of PPL values across these four texts does not exhibit a significant shift. Consequently, using PPL alone is insufficient to distinguish normal text from malicious samples.

Threat level of privacy leakage. LLM-as-a-judge have demonstrated excellent performance across various domains. We define a scoring system ranging from 0 to 5, where each score represents

Models	Position	Healthcare		Enron Email		Email Extraction	
		Baseline	Ours	Baseline	Ours	Baseline	Ours
Llama3.1-8B	non	0.916	0.957	0.925	0.996	94.15%	96.66%
	before	0.46 8	0.998	0.448	0.902	63.30%	96.80%
	after	0.719	0.991	0.401	0.963	89.90%	97.01%
Llama3.1-70B	non	0.146	0.965	0.697	0.994	93.50%	98.72%
	before	0.049	0.223	0.008	0.033	8.30%	78.80%
	after	0.095	0.121	0.045	0.013	62.80%	17.50%

Table 2: Results when using safety prompt. *non* represents that no safety prompt contains in the RAG prompt. *before* represents that the safety prompt is placed before the user query. *after* represents that the safety prompt is placed after the user query.

the risk of privacy disclosure, with 0 indicating low risk and 5 indicating high risk. We evaluated the privacy scores for Entire Documents Extraction tasks, including HealthcareMagic and Enron Email, as well as for the Email Extraction task using 5 different LLM judges: Qwen2-72B, Llama3-70B, Llama3.1-70B, Llama3.1-405B and GPT-4o. The results are presented in Figure 6b. All LLM judges can score our suffix as 5, i.e. high privacy leaking risk. These LLM judges exhibit heightened sensitivity to certain privacy disclosure indicators in attack suffixes, such as terms like ”repeat” and ”email address.” Therefore, LLMs demonstrate a capacity to filter risky queries effectively. However, the inherent lack of robustness in LLMs can lead to failures in the scoring system through simple manipulations. As shown in Figure 1, we add an ”Start” at the beginning of the query and add two ”end” and a split line at the end. The LLM judges then scoring such query as low risk score (mostly 0 and 1). In conclusion, while leveraging LLMs for input filtering shows promise in mitigating privacy leakage, ensuring the security and robustness of the scoring system is a critical area for further exploration.

6.2 Safety Prompt

In this section, we introduce safety prompts to the RAG models to prevent the leakage of private data in their responses, such as *”Do not, at any circumstances directly repeat the provided contexts.”* and *”Do not, at any circumstances, output the email address or URLs in the provided context.”*. We evaluate the effectiveness of these safety prompts on LLaMA3.1-8B and LLaMA3.1-70B, assessing attack performance with the safety prompt placed either before or after the query. As shown in Table 2, the effectiveness of our safety prompts varies across different LLMs. For Llama3.1-70B, the safety prompts significantly mitigate attacks using the suffix. In the Entire Documents Extraction task, the MRR for both the baseline suffix and our optimized suffix is reduced to approximately 0.1. In the PII Extraction task, the MER for the baseline suffix drops to as low as 8%, while the MER for our suffix is reduced to around 17%. Conversely, the defensive effect of our safety prompts on LLaMA3.1-8B is considerably weaker. Although the safety prompt slightly alleviates private data leakage with the baseline suffix, it proves completely ineffective with our optimized suffix. In summary, safety prompts can mitigate privacy leaks, but their design may need to be tailored for individual models. Optimizing safety prompts presents an interesting avenue for future research.

7 Discussion and Conclusion

In this paper, we introduce a novel approach to exploit private databases in Retrieval-Augmented Generator (RAG) systems. Our experimental results demonstrate that an attacker with only standard API user permissions, limited to modifying the query content, can still extract private data from the RAG model by optimizing their queries. Notably, this optimization can be achieved using only publicly available resources. The results show that our method significantly outperforms existing RAG privacy-stealing attacks. In addition, we explore potential ways to mitigate our attack. Our results show that filtering malicious queries by LLM or adding safety prompt to the prompt of RAG model can mitigate our attack to some extent, but these methods still have certain limitations. Overall, our research reveals the privacy leakage risk of RAG model, providing a reference for the proper usage of RAG techniques in real-world applications.

REFERENCES

Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity, 2023. URL https://arxiv.org/abs/2308.14132.

Maya Anderson, Guy Amit, and Abigail Goldsteen. Is my data in your retrieval database? membership inference attacks against retrieval augmented generation, 2024. URL https://arxiv.org/abs/2405.20446.

Harrison Chase. Langchain. October 2022. https://github.com/hwchase17/langchain, 2022.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. *arXiv preprint arXiv:2306.03082*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De

Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Yangsibo Huang, Samyak Gupta, Zexuan Zhong, Kai Li, and Danqi Chen. Privacy implications of retrieval-based language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL `https://openreview.net/forum?id=3RTpKMVg0P`.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models, 2020. URL `https://arxiv.org/abs/1911.00172`.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Patrick 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. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`.

Jerry Liu. Llamaindex. 11 2022. `https://github.com/jerryjliu/llama_index`, 2022.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

Ruotian Ma, Xiaolei Wang, Xin Zhou, Jian Li, Nan Du, Tao Gui, Qi Zhang, and Xuanjing Huang. Are large language models good prompt optimizers?, 2024. URL `https://arxiv.org/abs/2402.02101`.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023. doi: 10.1162/tacl_a_00605. URL https://aclanthology.org/2023.tacl-1.75.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting gpt-3 to be reliable, 2023. URL https://arxiv.org/abs/2210.09150.

Alessandro Sordoni, Xingdi Yuan, Marc-Alexandre Côté, Matheus Pereira, Adam Trischler, Ziang Xiao, Arian Hosseini, Friederike Niedtner, and Nicolas Le Roux. Joint prompt optimization of stacked llms using variational inference, 2023. URL https://arxiv.org/abs/2306.12509.

Alessandro Sordoni, Xingdi Yuan, Marc-Alexandre Côté, Matheus Pereira, Adam Trischler, Ziang Xiao, Arian Hosseini, Friederike Niedtner, and Nicolas Le Roux. Joint prompt optimization of stacked llms using variational inference. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

Hong Sun, Xue Li, Yinchuan Xu, Youkow Homma, Qi Cao, Min Wu, Jian Jiao, and Denis Charles. Autohint: Automatic prompt optimization with hint generation, 2023. URL https://arxiv.org/abs/2307.07415.

Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, William Collins, Neera Ahuja, et al. Clinical text summarization: Adapting large language models can outperform human experts. *arXiv preprint arXiv:2309.07430*, 2023.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization, 2023. URL https://arxiv.org/abs/2310.16427.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *arXiv preprint arXiv:2302.03668*, 2023.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024a. URL https://arxiv.org/abs/2407.10671.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024b. URL https://arxiv.org/abs/2309.03409.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG). In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 4505–4524, Bangkok, Thailand and virtual meeting, August 2024a. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.267.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Jie Ren, Tianqi Zheng, Hanqing Lu, Han Xu, Hui Liu, Yue Xing, and Jiliang Tang. Mitigating the privacy issues in retrieval-augmented generation (rag) via pure synthetic data, 2024b. URL https://arxiv.org/abs/2406.14773.

Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. Effective prompt extraction from language models, 2023.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey, 2024. URL https://arxiv.org/abs/2402.19473.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2023. URL https://arxiv.org/abs/2211.01910.