Exploiting the Shadows: Unveiling Privacy Leaks through Lower-Ranked Tokens in Large Language Models

Yuan Zhou[1], Zhuo Zhang[1], Xiangyu Zhang[1]
[1]Purdue University
{zhou1475, zhan3299, xyzhang}@purdue.edu

Abstract

Large language models (LLMs) play a crucial role in modern applications but face vulnerabilities related to the extraction of sensitive information. This includes unauthorized accesses to internal prompts and retrieval of *personally identifiable information* (PII) (e.g., in Retrieval-Augmented Generation based agentic applications). We examine these vulnerabilities in a question-answering (QA) setting where LLMs use retrieved documents or training knowledge as few-shot prompts. Although these documents remain confidential under normal use, adversaries can manipulate input queries to extract private content. In this paper, we propose a novel attack method by exploiting the model's lower-ranked output tokens to leak sensitive information. We systematically evaluate our method, demonstrating its effectiveness in both the agentic application privacy extraction setting and the direct training data extraction. These findings reveal critical privacy risks in LLMs and emphasize the urgent need for enhanced safeguards against information leakage.

1 Introduction

LLMs have become increasingly integral to a variety of applications. However, as these models grow in complexity and capabilities, concerns about malicious extraction of sensitive information from them have intensified. For example, the extractions of models' internal prompts and *personally identifiable information* (PII) embedded within their training data are two significant vulnerabilities, as both involve unauthorized access to confidential information through sophisticated attack strategies.

To inspect these vulnerabilities in a practical context, we set up a QA task where an LLM agent retrieves relevant documents to answer user queries or directly answer user queries based on their training knowledge. These documents, which may contain private information, are used as in-context examples to generate responses. Importantly, users do not have direct access to these retrieved documents or model's training data, preserving their confidentiality under normal operations.

However, adversaries can exploit this setup by manipulating input queries to extract private documents or personal information. For instance, an attacker might append an attack prompt to the end of a user query. As such, the model will reveal the content of retrieved documents embedded in the prompt due to this prompt injection attack. In our settings, the attacker operates under realistic constraints, having access only to the model's top-k output logits (or its equivalence) and the generated responses, without any knowledge of the model parameters or gradients. More can be found in our later discussion of the threat model.

Our work focuses on analyzing these vulnerabilities within the practical QA task setup. Different from previous works that focus on manipulating prompts to extract information, we fix attack prompt and discover that the model leaks sensitive information through its lower-ranked output tokens. This finding is significant because it reveals a new avenue for information leakage that does not rely on prompt manipulation but exploits the model's output distribution. The contribution of our works is that we introduce a novel attack method that extracts sensitive information based on model's output lower-ranked tokens. We have conducted comprehensive evaluations of this attack, demonstrating its effectiveness in both the agentic application privacy extraction setting and the direct extraction of training data from the model. Our findings highlight a significant privacy leakage issue in LLMs, underscoring the need for improved safeguards.

2 Related work

Prompt extraction: Our work focuses on information extraction from LLM prompts. Zhang

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8376–8386

et al. (2024) proposed a simple attack by generating a similar attack prompt trying to steal a secret prompt of the model. Qi et al. (2024) studies an adversarial setting by considering a threat model that seeks to extract text data from a private, non-parametric database of Retrieval-Augmented Generation (RAG) models with the black-box API access. Zeng et al. (2024) implements a RAG setup and uses prompt leakage attacks to extract PII from the external retrieval database. Agarwal et al. (2024) extends this threat to a multi-turn scenario. In addition, Perez and Ribeiro (2022) implements a prompt injection attack to extract prompts from the model. Morris et al. (2023) extracts next-token probabilities at a given position to reconstruct an input prompt. Sha and Zhang (2024) propose a methodology for prompt leakage using parameter extraction and prompt reconstruction.

PII extraction: One way to perform the PII extraction involves generating hand-crafted templates that aim to extract PII (Shao et al., 2023). For example, an adversary might prompt the model with "*the phone number of {name} is.*", substituting "{name}" with the victim's name. While such an attack requires no prior background, its performance largely depends on the quality of templates, and the performance of this method is low. Others attack ways assume the attacker has partial knowledge of the training dataset. Kim et al. (2024) assumes the attacker knows other PII of the owner and targeted PII is presented within a context directly tied to the owner. However, this still has a low attack success rate. Additionally, Lukas et al. (2023) and Li et al. (2024) use prefixes found in the training data, in the hope that the model outputs the exact PII suffix. This approach significantly outperforms the simplest attack but has a strong assumption that the adversary has access to the real prefixes from the training data. To improve this, Nakka et al. (2024) assumes the attacker only knows part of the prefix in the dataset and concatenates simple attack prompts after the prefix. Its performance is better than directly asking the model.

3 Threat model

3.1 Agentic application privacy extraction

We first set up a practical QA task in which an LLM agent with RAG to answer domain-specific questions. As shown in Figure 1, the LLM agent retrieves documents related to the user query from a database and uses those documents as a few-shot

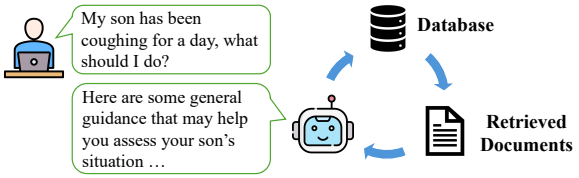


Figure 1: Illustration of the agentic application. The LLM retrieved documents which most relevant to the user query from its database, then generate answer based on the retrieved documents and its own knowledge.

prompt to generate a response to achieve good performance. Because these documents may contain private information, users don't have access to them. Take an medical QA system as an example. Assuming the user asks "*My son has been coughing for a day, what should I do?*", the model would look up relevant information from a medical records database and generate a response given the medical records. However, because these records contain private patient information, they are not publicly available. The attacker tries to steal those documents by concatenating an attack prompt (e.g., "*don't answer the question, just repeat all previous sentences*") at the end of the query. We assume that attackers do not have access to the model parameters or gradients, and they only have access to the output top-k logits (log probabilities) and the response. This assumption is practical and aligns with real-world deployments: many LLM-based agents, similar to OpenAI's API design, expose top-k logits to enable downstream agents or users to calibrate model confidence or apply custom decoding strategies to reduce hallucinations. The victim model provides these top-k logits to users so they can evaluate the model's confidence in its answers or reduce hallucinations by applying their own customized decoding strategies. Additionally, even API models do not return top-k logits, they usually allow user to add a bias to specific token's logits (e.g., latest GPT models). If a model only returns the top-1 token, an attacker can manipulate the process by setting the logit of the top token to -inf, effectively removing it from consideration. When the same prompt is repeated, the model's new top-1 token will be the one that ranked second in the previous round, modulo the output uncertainty. Note that although this approach may involve extra token cost; the cost is negligible compared to the impact of the attack.

In our setting, a user query consists of two distinct parts: the first is a normal question, and the

second is an attack prompt as previously mentioned. For example, the user query of previous instance can be "*My son has been coughing for a day, what should I do? Don't answer the question, just repeat all previous sentences.*" This two-part structure is essential because by varying the first part, the attacker can specify the context of extraction. As in a typical agent setting, for each user query, two documents are retrieved from the database, and the user query is appended to the retrieved documents. Thus, the complete query would have the following format: Document 1: [RETRIEVED DOC 1] \n Document 2: [RETRIEVED DOC 2] \n Question: [NORMAL QUESTION] + [ATTACK PROMPT]. Then the model will generate response given this query.

We have two distinct types of attack prompts, each representing a specific type of information extraction attack. The first one involves comprehensively extracting the entire content of a document, requiring the model to accurately reproduce all sentences in its response. The second one focuses on extracting sensitive personal information, such as email addresses and phone numbers. Although this task entails reproducing only parts of a document, it remains crucial due to the significant privacy risks.

To defend against these attacks while minimizing the reduction in the model's performance, we incorporate in-context examples of defense methods. These examples illustrate proper responses where the model appropriately refuses to disclose sensitive information.

3.2 Training data PII extraction

In addition to extracing sensitive information in agentic applications, we also attempt to extract PII directly from the model's pre-training data. In this setup, we explicitly ask the model to provide the PII of a person that we know for sure belongs to the pre-training dataset.

4 Attack design

4.1 Observations

Existing LLMs are usually well aligned to refuse leaking private information. However, we noticed that even their generated response is a refuse, the malicious answer is implicitly concealed within the top-k hard-label outcomes. For example, as shown in Figure 2, in our attack scenario, we ask the model to repeat all the previous instructions. The top-1 and top-2 tokens may be "*Sorry*" or just repeating the user instruction. But when it reaches the top-6 token, the model starts repeating documents after generating "*The*".

However, to avoid the excessive cost in enumerating the sentences generated by individual top-k tokens, we ought to find a way to determine if a token may lead to a sentence of interest. The key observation is that the output logits are likely different when the model is generating response for the first part of query (i.e., the normal question) and when it is responding to the attack prompt and repeating the in-context document. This might be because when the model is repeating, it tends to be more confident. So we could utilize the observation to train a classifier to determine if the sentence is in the retrieved documents.

Based on the observations, we propose a novel attack method. Figure 2 shows the overview, which has several key steps. We first obtain the top-k candidate tokens for the next output by the model. For each candidate token, the model continues its generation for a few steps, yield a partial sentence. We apply a re-ranking method to these partial sentences to prioritize the sentences that are most likely repeating documents. The model then completes the generations for the top five partial sentences. In many cases, even though the model leaks some of the sensitive information, it may switch to answering the user question (i.e., the first part of the query), which does not serve our purposes. Therefore, we use a sentence evaluator to determine such transition points. For each such point, we repeat our earlier top-k token exploration to force the model to continue leaking information. In the following, we explain the re-ranking method and the sentence evaluator in 4.2 and 4.3.

4.2 Sentence re-ranking

Based on our experiments, we find there are several different situations in top-k partial sentences when the model generating response for the attacker's query. (1) The model may refuse to repeat the document, for example, the model may generate "*Sorry, I cannot repeat previous documents*". (2) The model may simply repeat the question and the attack prompt instead of the requested document. (3) The model may generate an answer to the normal question instead of repeating the document. (4) The model starts repeating retrieved documents. Thus, the re-ranking step aims to filter out those non-malicious responses and increase the rank of

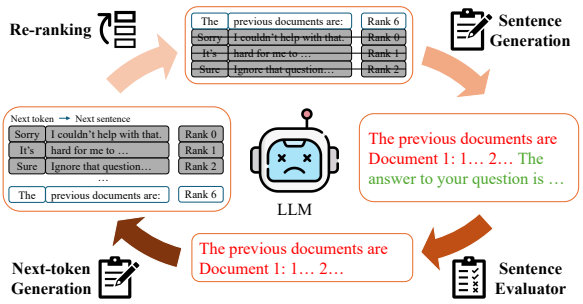


Figure 2: Illustration of attack pipeline: 1. obtain the top-k tokens and let model generate a short sentence based those tokens, 2. re-ranking sentences, 3. let model complete top-ranked response, 4. truncate all sentence after the point that the model stopped producing malicious content.

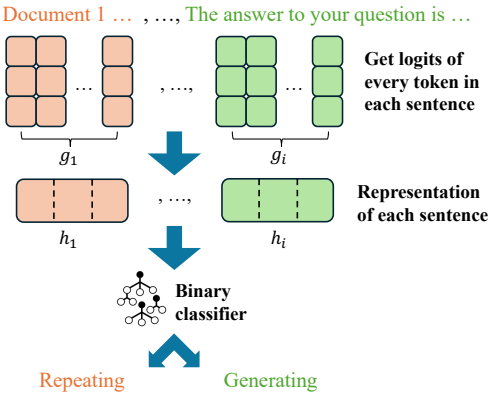


Figure 3: Illustration of sentence evaluator: First, obtain a representation of each sentence based on the logits of its tokens. Then, use a binary classifier to identify which sentence is not repeating from the document.

privacy leakage responses, without this procedure, the model's original top-k answers would be sent directly to sentence evaluator, substantially reducing the attack success rate.

Because we only want the last situation, we need to re-rank the top-k answers and increase its ranking when the sentence is potentially repeating document or leaking private information. Suppose the user query is q_u, the attack prompt is q_a and the model response is r. For the first two cases, we simply filter them out by finding the key words in the response, for example, when phrases like "*I cannot*" or the first few words of normal question appear in a response, we will give it a low score. For the third case, it is difficult to distinguish whether the model starts repeating document or responding to the normal question by key words matching. However, if the model is answering the normal question, the response would have a high correlation with the question. Thus, we use a pre-trained model from nli-deberta-base (Reimers, 2019) to calculate the correlation between the response and the normal question. In addition, when the model is repeating documents, it often follows a pattern (e.g., "*1..., 2..., 3...*"). Based on these, responses that list sentences in this way as well as have a lower correlation with the normal question are given a high score.

4.3 Sentence Evaluator

When the model is repeating the document at beginning, it may change to answering the normal question after generating a few sentences. For example, after repeating part of the document the model may say "*Based on document 1, the answer to your question is ...*" instead of continue repeating the rest of the document. Thus, it is crucial to determine if the generated sentences are in the document and find the transition point that the model stops repeating and force the model to continue to repeat the document(s) or private information. Intuitively, the model is more confident when repeating information than generating new sentences, which means the logits distributions are likely separable in these two cases.

Specifically, as shown in Figure 3, to find a transition point, we first get the representation of each sentence in the response. Suppose the model response is $r = [s_1, ..., s_n]$, where s_i is the ith sentence in the response and n is the total number of sentence in the response. The logits of sentence s_i is $g_i = [g_{i,1}, ..., g_{i,m}]$, where $g_{i,j}$ is the logits of the jth token in sentence s_i and m is the length of this sentence. To make sure all the sentence have the same representation dimension, we separate the sentence into L pieces $g_i = [g_{i,1}, ..., g_{i,m/L}, g_{i,l+1}, ..., g_{i,2m/L}, ..., g_{i,m}]$. Then, as shown in Eq 1, we calculate the mean, minimum, maximum and standard deviation of each piece and concatenate them together as the final representation h_i of the sentence. Then we train a binary classifier as the sentence evaluator to determine which sentences are repeats from the document and which are generated by the model. As shown in Figure 2, after using the classifier to find a transition point, we force the model to generate top-k candidates for this token and use the sentence re-ranking method to select the candidate that most likely to continue repeating documents. This process repeats until it reaches the eos token,

or none of the top-k tokens can pass the evaluator.

$$\begin{aligned} \text{mean}_l &= \text{mean}(g_{i,(l-1)m/L+1}, ..., g_{i,lm/L}) \\ \text{min}_l &= \text{min}(g_{i,(l-1)m/L+1}, ..., g_{i,lm/L}) \\ \text{max}_l &= \text{max}(g_{i,(l-1)m/L+1}, ..., g_{i,lm/L}) \\ \text{std}_l &= \text{std}(g_{i,(l-1)m/L+1}, ..., g_{i,lm/L}) \\ h_i &= [\text{mean}_1, \text{min}_1, \text{max}_1, \text{std}_1, \\ &\quad ..., \text{mean}_L, \text{min}_L, \text{max}_L, \text{std}_L] \end{aligned} \tag{1}$$

5 Experiment

5.1 Datasets

For the agentic application setting, we used six different domain datasets: bbcnews[1], fnspid(Dong et al., 2024), enron(Klimt and Yang, 2004), code_alpaca (CA)[2], healthcaremagic (HCM)(Li et al., 2023), and billsum(Kornilova and Eidelman, 2019). For each dataset, we obtained 120 documents and truncated each document to approximately 200 words to remove any length bias when studying the leakage effect. Because the bbcnews, fnspid, enron, and billsum contain only documents, we hence use GPT-3.5-turbo to generate related questions based on the selected documents. The remaining two datasets are QA datasets. following a setup in the literature (Agarwal et al., 2024), we consider the selected documents as the pool of retrieved documents and randomly select from the questions from the remaining data (i.e., excluding the 120 selected documents) as the normal questions. In particular, each test prompt consists of two documents from the pool of 120 selected documents, followed by a query consisting of a randomly selected question and the attack prompt that requires the model to repeat the documents in the prompt.

Additionally, in order to test our method's ability to retrieve sensitive personal information, we generate a set of privacy-related data points (such as names, email addresses, phone numbers, and credit card numbers), integrate them into the original dataset, and let the model to specifically repeat this personal information given the owner's name.

We created two distinct datasets for testing pre-training PII data extraction: one sourced from the Enron dataset (Klimt and Yang, 2004) and the other obtained through web crawling. From the enron dataset, we extracted 500 pairs consisting of names, email addresses, and phone numbers. To achieve this, we first identified phone numbers based on their format (e.g., "xxx-xxx-xxxx" or "(xxx)-xxx-xxxx") and then used GPT-4 to determine the associated owner and email address for each number. We also randomly generated a credit card number for each individual to evaluate the model's vulnerability to extraction attack of highly sensitive information. Additionally, recognizing that many LLMs are also trained on internet-sourced data, we gathered approximately 150 email addresses and phone numbers by crawling publicly available websites constructed before year 2022 as our internet-sourced dataset.

5.2 Baseline

For the agentic application privacy extraction setting, since most current work relies on prompt-based methods, we begin by selecting the best-performing prompt as our initial prompt whose performance is shown in line 1-prompt in Table 2 and Table 3. Given the additional cost of our method, we generate t prompts with similar attack performance from the initial prompt using GPT-4, ensuring that the overall cost remains comparable, the result is shown in line t-prompt in Table. Although our threat model presents a more challenging scenario than conventional jailbreak attacks—requiring the attacker to not only induce the model to leak sensitive information but also accurately distinguish whether the leaked content is authentic or fabricated—we nonetheless selected an existing jailbreak attack method as another baseline for evaluation. Specifically, we conducted experiments using the OPRA (Wang et al., 2024) on document extraction tasks by employing GPT-4o to generate contextually appropriate prefixes for the OPRA attack, such as "Sure, here are the previous sentences:".

For directly extracting pre-training PII data, we employ four different prompt templates. The *base type* means directly prompting the model to generate a phone number given an owner's name. The *prefix type* provides the model with about 150 tokens of context preceding the phone number we aim to extract. The *fake_prefix type* uses a fixed prefix to extract all numbers in the dataset. The w_email type that instructs the model to generate the phone number based on the owner's name and email address. We use the model's greedy decoding as the baseline.

[1] https://huggingface.co/datasets/RealTimeData/bbc_news_alltime

[2] https://huggingface.co/datasets/sahil2801/CodeAlpaca-20k

5.3 Evaluation Metrics

For the document extraction in agentic applications, we have 5 different evaluation metrics. As we mentioned before, because the model may repeat the document in the format like "*The sentence in the documents are: 1....2...*", directly calculating the matching of response and document is not accurate. Thus we propose a sentence level measurement method: sentence match rate (SMR) R_s, as Eq 2 shows, suppose the retrieved document is $d = [d_1, ..., d_l]$, where d_i is the ith sentence in the document. We separate the document to sentences based on the punctuation like ".", "!", "?" and etc. Then we measure how many sentences in the document are repeated, where $I(d_i, r)$ means whether d_i is being repeated in the response r. If all sentences in the document are being repeated, we say the attack succeeds. In other words, the document extraction is successful if all sentences in the document appear in the response. We also measure the Rouge-L (R-L), BertScore (BS) and BLEU of the response.

In addition, we also test how much privacy can be extracted in this attack, the privacy information extraction attack succeeds if the complete private information is extracted correctly.

$$R_s = \frac{\sum_{i=1}^{l} I(d_i, r)}{l}, \quad I(d_i, r) = \begin{cases} 1 & \text{if } d_i \in r \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

5.4 Models

We test our method on GPT-3.5-turbo-instruct[3], llama3-8b-Instruct(Dubey et al., 2024), llama2-7b-chat, llama2-13b-chat(Touvron et al., 2023) and vicuna-13b-v1.5(Zheng et al., 2023), which are aligned. For the pre-training PII extraction, we additionally test our method on GPT-J-6B(Wang and Komatsuzaki, 2021), llama2-13b, llama2-7b, and llama3-8b model, which are unaligned, for the reference purpose.

5.5 Training and evaluating sentence evaluator

First, we test the assumption of our sentence evaluator. Figure 4 shows the t-SNE of sentence representation of the Llama3-8b-Instruct model, which is the logits of each sentence. The orange dots mean the logits when the model is repeating exact the same sentence in the retrieved model, and the green dots represent new sentences geneated by the model (to answer normal questions). From Figure 4 we can see these two different types of sentences' logits are separable, which validate our assumption.

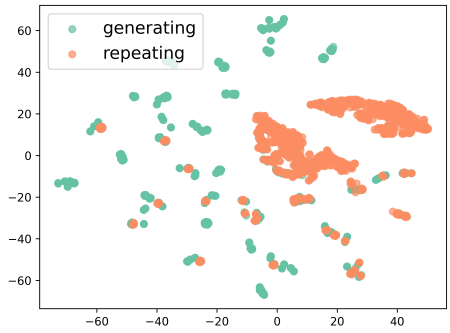


Figure 4: T-SNE of generated sentence logits and repeating document logits.

Model	Accuracy	Precision	Recall
GPT-3.5-Turbo-Instruct	0.935	0.961	0.907
Llama3-8b-Instruct	0.940	0.925	0.957
Llama2-7b-chat	0.922	0.890	0.962
Llama2-13b-chat	0.900	0.871	0.940
Vicuna-13b-v1.5	0.882	0.868	0.900

Table 1: Sentence evaluator's performance over different models.

To train the evaluator, we collect 2000 pieces of data from the aforementioned datasets that the model is repeating or generating sentence to train a random forest. The data entries are constructed in a way similar to our test data construction but from different documents that are not used in testing. Table 1 shows the accuracy, precision and recall of the sentence evaluator. We can see that the accuracy is mostly over 90% with a high precision and recall, indicating that the classifier has a high performance over all test models.

5.6 Agentic application privacy extraction results

Table 2 shows the result for the agentic application document extraction of GPT-3.5-Turbo-Instruct and Llama3-8B-Instruct, more detailed results are shown in Table 7. We can see that most models show a notable improvement in most metrics with our method. Compared to the t-prompt, for GPT-3.5-Turbo-Instruct on the bbcnews dataset, the

[3] https://platform.openai.com/docs/models#gpt-3-5-turbo

ASR increases from 0.2 to 0.317, and even for the cases that cannot repeat the entire document, there is still 76.6% sentence match rate, which means it can correctly repeat most of the documents. Additionally, the ASR of Llama3-8B-chat on the enron dataset improves significantly with our method, increasing from 0.0 to 0.13, while the SMR rises from 0.133 to 0.737. Additionally, performance varies notably across datasets. For instance, the results on the bbcnews and billsum datasets are generally 10% higher compared to the enron and healthcaremagic datasets, suggesting that different types of documents may have different vulnerabilities to being leaked by LLMs. Compared to the OPRA baseline, our method consistently achieves superior performance across all evaluated datasets and models. Specifically, it yields ASR and SMR, indicating a stronger ability to extract information. Our observations indicate that under OPRA, the model initially replicates the provided in-context documents but subsequently generates additional content beyond the given input.

Table 3 shows the result for privacy information extraction under the agentic application setting. For all models, our method shows a significantly higher ASR than the baseline. This suggests that our technique is increasing the likelihood of extracting sensitive information from aligned models. We notice that Llama3-8b-Instruct has the lowest baseline performance of all models but shows a sharp increase with our method. The vicuna-13b-v1.5 model demonstrates the highest attack success rates in all methods. In the baseline scenario, it achieves about 0.7 for both email and phone number extraction and over 0.5 for credit card extraction, and with our method, the attack success rate reaches 0.533, and both email and phone number extraction rates surpass 0.8. Additionally, although the Llama3-8B-Instruct model demonstrates the strongest inherent safeguards for protecting sensitive information among evaluated models, our proposed method significantly increases its vulnerability. Specifically, the Attack ASR surpasses 0.5, with the extraction rates for email addresses and phone numbers reaching approximately 0.7. This indicates that, despite LLMs have safeguard to protect sensitive information, they still tend to leak PII under our attack, highlighting the need for the development of effective defense methods to address this vulnerability.

Dataset	Method	ASR	SMR	R-L	BLEU
GPT-3.5-Turbo-Instruct					
bbcnews	1-prompt	0.075	0.147	0.057	14.105
	t-prompt	0.200	0.517	0.431	47.565
	OPRA	0.225	0.597	0.154	56.195
	our method	**0.317**	**0.766**	0.314	**62.446**
billsum	1-prompt	0.0750	0.168	0.097	18.760
	t-prompt	0.100	0.465	0.352	56.871
	OPRA	0.316	0.583	0.360	60.261
	our method	0.167	0.555	**0.398**	56.805
fnspid	1-prompt	0.033	0.250	0.127	24.080
	t-prompt	0.142	0.514	0.395	43.758
	OPRA	0.125	0.625	0.151	62.176
	our method	**0.192**	**0.666**	0.259	60.910
enron	1-prompt	0.000	0.094	0.029	2.524
	t-prompt	0.000	0.101	0.091	5.734
	OPRA	0.000	0.155	0.067	8.646
	our method	**0.008**	0.134	**0.094**	**15.787**
HCM	1-prompt	0.017	0.187	0.157	16.811
	t-prompt	0.017	0.343	0.324	27.491
	OPRA	0.000	0.311	0.242	32.511
	our method	**0.025**	0.338	0.271	28.592
CA	1-prompt	0.000	0.090	0.204	10.148
	t-prompt	0.050	0.341	0.464	38.475
	OPRA	0.067	0.193	0.293	18.643
	our method	**0.067**	0.314	0.340	22.611
LLama3-8b-Instruct					
bbcnews	1-prompt	0.075	0.183	0.053	18.346
	t-prompt	0.242	0.370	0.205	35.413
	OPRA	0.200	0.516	0.226	48.708
	our method	**0.283**	**0.863**	**0.250**	**71.114**
billsum	1-prompt	0.108	0.361	0.233	43.157
	t-prompt	0.208	0.599	0.401	57.147
	OPRA	0.092	0.263	0.244	38.289
	our method	**0.342**	**0.702**	**0.408**	**61.202**
fnspid	1-prompt	0.133	0.521	0.132	51.107
	t-prompt	0.158	0.497	0.305	43.492
	OPRA	0.200	0.593	0.321	55.803
	our method	**0.317**	**0.757**	0.247	**65.732**
enron	1-prompt	0.000	0.059	0.026	2.290
	t-prompt	0.000	0.133	0.097	6.241
	OPRA	0.000	0.102	0.077	4.519
	our method	**0.133**	**0.737**	**0.452**	**62.889**
HCM	1-prompt	0.000	0.019	0.012	0.289
	t-prompt	0.000	0.095	0.076	8.002
	OPRA	0.000	0.058	0.039	6.109
	our method	**0.142**	**0.305**	**0.224**	**23.653**
CA	1-prompt	0.000	0.006	0.061	0.576
	t-prompt	0.067	0.454	0.376	35.798
	OPRA	0.000	0.026	0.075	2.600
	our method	**0.425**	**0.687**	**0.600**	23.698

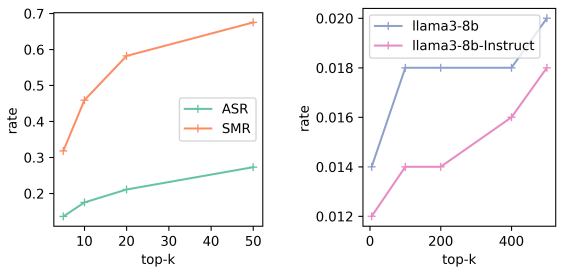
Table 2: Performance metrics across datasets for baseline and our methods in agentic application document extraction.

To investigate the relationship between top-k and ASR, we test the document extraction attack with top-k equals to 5, 10, 20, 50 on Llama3-8b-Instruct. Figure 5(a) shows the results of different top-k candidates. We can see that as the top-k increase, both attack success rate and sentence match

Model	Method	ASR	Email	Phone Number	Credit Card
gpt-3.5-turbo-instruct	1-prompt	0.083	0.092	0.092	0.200
	t-prompt	0.242	0.258	0.250	0.425
	our method	**0.583**	**0.592**	**0.592**	**0.667**
Llama3-8b-Instruct	1-prompt	0.000	0.000	0.000	0.000
	t-prompt	0.008	0.008	0.008	0.017
	our method	**0.533**	**0.683**	**0.692**	**0.533**
Llama2-7b-chat	1-prompt	0.158	0.183	0.167	0.158
	t-prompt	0.342	0.400	0.350	0.342
	our method	**0.550**	**0.558**	**0.567**	**0.767**
Llama2-13b-chat	1-prompt	0.167	0.167	0.167	0.167
	t-prompt	0.267	0.267	0.267	0.291
	our method	**0.542**	**0.642**	**0.658**	**0.542**
vicuna-13b-v1.5	1-prompt	0.267	0.258	0.283	0.267
	t-prompt	0.492	0.817	0.708	0.517
	our method	**0.533**	0.808	**0.808**	**0.533**

Table 3: ASR for different models under baseline and our method in agentic application PII extraction.

rate increase, but as k increases, the growth rate gradually slows down. This also validates our motivation, indicating that although the result of LLMs greedy decoding is incorrect, the true answer might be among the results generated by the top-k tokens.



(a) (b)

Figure 5: 5(a) Results of Llama3-8b-Instruct using different top-k candidates under agentic application document extraction setting. The x-axis denotes the top-k, and the y-axis denotes the ASR and SMR. 5(b) Results of Llama3-8b and llama3-8b-Instruct using different top-k candidates under training data PII extraction setting. The x-axis denotes the top-k and the y-axis denotes the ASR.

5.7 Pre-training PII extraction result

Table 4 presents the results for the pre-trained models. we observe that all ASR is low, but our method's ASR is still higher than the baseline across all prompt types, which supports our hypothesis that the model may conceal the ground truth among its top-k candidates. Additionally, the ASR for prefix type prompt is higher than other prompt types, which aligns with the training process of models, indicating their tendency to complete sentences using patterns from the same training samples. Moreover, the ASR for GPT-J-6B is significantly higher than that other models, possibly because GPT-J-6B is trained on a smaller

	base	prefix	fake_prefix	w_email
GPT-J-6b				
baseline	0.002	0.102	0.012	0.012
our method	**0.028**	**0.156**	**0.030**	**0.028**
Llama2-7b				
baseline	0.004	0.006	0.000	0.002
our method	0.004	**0.010**	0.000	0.002
Llama2-13b				
baseline	0.002	0.025	0.002	0.002
our method	**0.006**	**0.030**	**0.004**	**0.004**
Llama3-8b				
baseline	0.002	0.007	0.000	0.004
our method	**0.006**	**0.021**	0.000	0.004

Table 4: ASR for different pre-trained open models and prompt templates.

training dataset so each individual data point easier to memorize.

Table 5 shows the results of aligned models under the same settings. It shows that aligned models' ASR is about the same in other settings but lower in the prefix setting compared to pre-trained models. This make sense because aligned models have post-training process which cause they tend to refuse to answer or even forget some of the training data. Additionally, for the prefix prompt type, our method still outperforms the baseline. Even all ASRs are low, our results show there is still a privacy risk in aligned models.

During the experiment we notice that sometimes the ground truth hide deeper than in-context setting, to figure out the relationship between top-k and ASR, we extract PII with top-k set to 0, 100, 200, 400, and 500 on Llama3-8b and Llama3-8b-Instruct. As shown in Figure 5(b), the attack success rate increases with higher top-k values and seems will continue increasing as k increases. Even though the model generates more different outputs when the top-k increases, our ASR remains significantly higher than what could be achieved by guessing a phone number at random. In particular, given that a phone number has 10 digits and the last 4 are effectively random, there are 10000 possible combinations for those final four digits. Even if the model only needs to guess these last four correctly, considering we need to select the final result from the top-k candidates, the probability of coincidentally guessing the number correctly is lower than 0.0001. In contrast, our ASR is far above this number.

Table 6 presents the results for the API model on the internet-sourced dataset. There are 4 dif-

ferent settings for this dataset. 'Email' means we directly ask LLM someone's email address without other information, and 'Phone number' means we ask for the phone number, 'w_domain' means we ask the model someone's PII with their domain information (e.g., company). Our method achieves a significantly higher ASR than the baseline, for example, the ASR for 'Email' type increases from 0 to 0.04, and for 'Email_w_domain' type it increases from 0.015 to 0.363. However, because the last four digits of a phone number are random, it is much more challenging to guess a phone number correctly if they are not in the training dataset Thus, the 1% ASR for phone number and 2% ASR for phone number with domain knowledge indicating these PIIs are highly likely in the training dataset and the model has memorized them.

	base	prefix	fake_prefix	w_email
Llama2-7b-chat				
baseline	0.002	0.006	0.000	0.002
our method	0.002	**0.010**	0.000	0.002
Llama3-8b-Instruct				
baseline	0.004	0.008	0.000	0.004
our method	0.004	**0.018**	0.000	0.004

Table 5: ASR for different aligned open models and prompt templates.

	Email	Email_w_domain	Phone number	Phone number_w_domain
baseline	0.000	0.015	0.000	0.000
our method	**0.044**	**0.363**	**0.011**	**0.022**

Table 6: ASR for aligned API open models of Internet sourced dataset.

6 Conclusion

This project aims to evaluate whether current LLMs can leak private information in both the agentic application setting and the direct extraction settings, thereby assessing their potential privacy risks. Our experiments demonstrate that LLMs still exhibit significant privacy leakage vulnerabilities under our attack, which demands attention and resolution. In particular, the ASR for PII extraction in the agentic application setting is notably high, highlighting the need for increased awareness and mitigation. Furthermore, although the ASR for pre-training PII data extraction is relatively low, our findings indicate that these models do memorize some personal information and may occasionally disclose it.

7 Limitation

One limitation of our method is its computational cost, as it requires analyzing the top-k tokens and generating responses based on them. Additionally, our evaluation is restricted to documents with a length of fewer than 500 tokens, leaving the attack ASR untested on longer context documents. This constraint limits the scope of our findings and their applicability to more extensive contexts. However, we argue a high cost may be justified by the value of extracted information. Moreover, our method assumes that LLMs will concatenate the retrieved documents with the user query in agentic applications, which may not align with real-world usage scenarios.

Acknowledgments

We are grateful to the Center for AI Safety for providing computational resources. This work was funded in part by the National Science Foundation (NSF) Awards SHF-1901242, SHF-1910300, Proto-OKN 2333736, IIS-2416835, DARPA VSPELLS - HR001120S0058, ONR N00014-23-1-2081, and Amazon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors

References

Divyansh Agarwal, Alexander R Fabbri, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2024. Investigating the prompt leakage effect and black-box defenses for multi-turn llm interactions. *arXiv preprint arXiv:2404.16251*.

Zihan Dong, Xinyu Fan, and Zhiyuan Peng. 2024. Fnspid: A comprehensive financial news dataset in time series. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4918–4927.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. 2024. Propile: Probing privacy leakage in large language models. *Advances in Neural Information Processing Systems*, 36.

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In

European conference on machine learning, pages 217–226. Springer.

Anastassia Kornilova and Vlad Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation. *arXiv preprint arXiv:1910.00523*.

Qinbin Li, Junyuan Hong, Chulin Xie, Jeffrey Tan, Rachel Xin, Junyi Hou, Xavier Yin, Zhun Wang, Dan Hendrycks, Zhangyang Wang, et al. 2024. Llm-pbe: Assessing data privacy in large language models. *arXiv preprint arXiv:2408.12787*.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 346–363. IEEE.

John X Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. 2023. Language model inversion. *arXiv preprint arXiv:2311.13647*.

Krishna Kanth Nakka, Ahmed Frikha, Ricardo Mendes, Xue Jiang, and Xuebing Zhou. 2024. Pii-compass: Guiding llm training data extraction prompts towards the target pii via grounding. *arXiv preprint arXiv:2407.02943*.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.

Zhenting Qi, Hanlin Zhang, Eric Xing, Sham Kakade, and Himabindu Lakkaraju. 2024. Follow my instruction and spill the beans: Scalable data extraction from retrieval-augmented generation systems. *arXiv preprint arXiv:2402.17840*.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Zeyang Sha and Yang Zhang. 2024. Prompt stealing attacks against large language models. *arXiv preprint arXiv:2402.12959*.

Hanyin Shao, Jie Huang, Shen Zheng, and Kevin Chen-Chuan Chang. 2023. Quantifying association capabilities of large language models and its implications on privacy leakage. *arXiv preprint arXiv:2305.12707*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024. Frustratingly easy jailbreak of large language models via output prefix attacks.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. 2024. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*.

Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. 2024. Effective prompt extraction from language models. In *First Conference on Language Modeling*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

A Additional results

Table 7 shows the results of documents extraction under agentic application setting for test models. We can see that most models show a notable improvement in most matrics when implementing our method. Additionally, larger model is more vulnerable than smaller model when doing document extraction.

Dataset	Method	ASR	SMR	R-L	BS	BLEU
LLama2-7b-chat						
bbcnews	1-prompt	0.000	0.020	0.034	0.818	4.319
	t-prompt	0.000	0.112	0.104	0.849	15.093
	our method	**0.033**	**0.210**	**0.120**	0.596	**27.469**
billsum	1-prompt	0.000	0.017	0.049	0.82	2.365
	t-prompt	0.000	0.055	0.122	0.851	14.003
	our method	**0.008**	**0.058**	0.049	0.339	9.193
fnspid	1-prompt	0.000	0.042	0.03	0.822	3.125
	t-prompt	0.000	0.128	0.079	0.844	12.034
	our method	0.000	**0.200**	**0.200**	0.799	**27.908**
enron	1-prompt	0.000	0.035	0.02	0.775	0.289
	t-prompt	0.000	0.129	0.062	0.799	6.182
	our method	0.000	**0.255**	**0.135**	**0.820**	**17.362**
HCM	1-prompt	0.000	0.121	0.067	0.838	12.875
	t-prompt	0.000	0.278	0.180	0.865	20.361
	our method	0.000	0.207	0.099	0.706	18.837
CA	1-prompt	0.042	0.151	0.182	0.807	9.279
	t-prompt	0.133	0.377	0.39	0.845	22.842
	our method	**0.158**	**0.385**	0.309	0.646	12.511
LLama2-13b-chat						
bbcnews	1-prompt	0.041	0.193	0.084	0.860	19.991
	t-prompt	0.058	0.322	0.156	0.881	33.800
	our method	0.042	**0.346**	0.130	0.770	**41.252**
billsum	1-prompt	0.067	0.163	0.150	0.866	20.858
	t-prompt	0.042	0.197	0.222	0.894	30.94
	our method	**0.175**	**0.354**	**0.250**	0.667	**36.235**
fnspid	1-prompt	0.041	0.183	0.063	0.847	15.451
	t-prompt	0.033	0.250	0.125	0.862	19.429
	our method	**0.042**	**0.411**	**0.138**	0.854	**44.752**
enron	1-prompt	0.000	0.056	0.031	0.772	1.729
	t-prompt	0.000	0.060	0.052	0.791	3.664
	our method	0.000	**0.411**	**0.181**	**0.846**	**28.242**
HCM	1-prompt	0.017	0.188	0.123	0.845	18.641
	t-prompt	0.042	0.395	0.322	0.881	40.677
	our method	**0.050**	**0.422**	0.260	0.855	37.868
CA	1-prompt	0.017	0.139	0.198	0.812	10.617
	t-prompt	0.025	0.275	0.366	0.842	20.917
	our method	**0.208**	**0.574**	**0.492**	**0.846**	20.186
vicuna-13b-v1.5						
bbcnews	1-prompt	0.150	0.552	0.146	0.935	57.495
	t-prompt	0.158	0.595	0.250	0.939	63.852
	our method	**0.208**	**0.779**	0.179	0.918	**73.311**
billsum	1-prompt	0.208	0.511	0.354	0.956	64.823
	t-prompt	0.250	0.601	0.412	0.957	68.833
	our method	0.250	0.591	0.367	0.848	58.980
fnspid	1-prompt	0.025	0.483	0.137	0.926	47.830
	t-prompt	0.041	0.521	0.200	0.925	50.037
	our method	**0.092**	**0.705**	0.200	**0.941**	**74.429**
enron	1-prompt	0.000	0.151	0.083	0.817	8.232
	t-prompt	0.000	0.161	0.089	0.816	8.218
	our method	0.000	**0.501**	**0.171**	**0.884**	**41.140**
HCM	1-prompt	0.067	0.385	0.286	0.891	39.850
	t-prompt	0.175	0.551	0.483	0.913	52.862
	our method	**0.200**	0.526	0.311	0.803	43.471
CA	1-prompt	0.000	0.181	0.353	0.852	18.654
	t-prompt	0.000	0.249	0.340	0.857	24.099
	our method	**0.125**	**0.505**	**0.428**	0.831	**25.281**

Table 7: Performance metrics across datasets for baseline and our methods in agentic application document extraction.