Strategies for Training Data Extraction from Large Language Models

Avi Attar
School of Public and International Affairs
abattar@princeton.edu

Caiden Kiani
Department of Computer Science
ckiani@princeton.edu

Hugh Shields
Department of Geosciences
nikolaus@princeton.edu

Abstract

Research has shown that large language models (LLMs) output text from their training corpora verbatim, a phenomenon known as memorization. Memorization raises issues regarding privacy and copyright, and these issue will only grow more acute as LLMs proliferate. While we know that LLMs memorize, major questions remain about the conditions which enable memorization. We explore these questions by reproducing a baseline prompt meant to induce memorization. Then, we conduct two attacks to extract training data. Along the way, we evaluate the effects of prompt and model size on memorization. We find significant evidence of memorization and that prompt and model size exacerbate memorization. As model sizes increase, datasets expand, and adversaries grow more advanced, understanding the conditions which impact memorization and using this knowledge to defend against memorization-based harms will remain in focus for model creators, researchers, and users alike.

1 Introduction

The rise of massive text datasets and exciting innovations in model architectures like transformers have driven the development and proliferation of large language models in recent years. In response, research into the properties of datasets and the behavior of models has surged. One focus of this research has been memorization, defined as the capacity of an LLM to output samples of its training data verbatim. Initial research has involved prompting LLMs to output memorized text, known as *training data extraction attacks* (Carlini et al., 2021). Researchers conducting these attacks have shown that LLMs are capable of memorization, which raises additional questions. In particular, the characteristics of effective prompts, the vulnerabilities of different types of data, and the impact of model type and size on memorization deserve further research.

1.1 Overview

This paper tackles these issues. We deploy and evaluate three prompting strategies. The first strategy, top-n sampling, entails prompting an LLM with a single token. Given its simplicity, top-n sampling has enjoyed success as a baseline in other work (Carlini et al., 2021). The second strategy, Wikipedia generation, entails prompting an LLM with a portion of a known string of text from Wikipedia to induce memorization. While other work (Carlini et al., 2021) has explored similar prompts, our strategy contributes to the literature by using text from the training data itself. The third strategy, URL generation, entails prompting an LLM with the string "`https://`" and evaluating the outputs. Other research has found that URLs are particularly well represented among examples of memorized data (Carlini et al., 2021). Our prompt explores this phenomenon directly.

1.2 Memorization in context

While the rest of this paper focuses on the technical aspects of memorization, it is important to situate this work in the context of larger issues like privacy and copyright. Memorization raises several concerns about privacy. First, although current datasets typically feature text scraped from publicly available sources like the internet and books, datasets of the future may contain private information furnished specifically for the purpose of model training. For example, tomorrow's models may be trained on inputs into today's. In this case, leakage of private data into the public sphere via memorization would be a huge harm to privacy.

Second, even leakage of publicly available data raises contextual privacy concerns. Although strict conceptions of privacy consider only data secrecy, contextual privacy asserts that not only the availability of data but also the context in which it appears are relevant to privacy considerations (Carlini et al., 2021). For example, a dog walker may post

their email address or phone number on a website for dog owners so that potential clients can contact them, but if this information appeared in a different context, like as the output of an LLM, the dog walker might feel as though their privacy has been violated. Third, an active area of discussion among privacy advocates is the right to be forgotten, and if LLMs remember data which has become unavailable elsewhere, this would impede on the right to be forgotten (Ishihara, 2023).

Memorization also raises copyright concerns. As previously mentioned, training datasets of an unprecedented scale have contributed to the recent success of LLMs. The creators of these datasets have casted wide nets to achieve this scale, and many popular datasets feature copyrighted text from news media, books, and other sources as a result. The copyright status of these datasets is currently being litigated, and the copyright status of outputs may be regulated in the future (Lanquist and Ray, 2024). Depending where the law settles, memorization of copyrighted information could present legal challenges for LLMs. Both for this reason and for privacy reasons, memorization is a significant topic for researchers. We seek to understand the phenomenon better in light of both these issues and relevant technical details.

2 Related Work

Perhaps the largest contribution of past research has been showing that LLMs reliably memorize and output training data. As Carlini et al. 2021 explains, memorization persists even as model creators have become adept at avoiding overfitting, in part because even a well-fit model may have particularly low training losses on certain instances of training data. Beyond demonstrating the existence of memorization, past research has defined memorization, proposed attack strategies, and evaluated the conditions under which memorization occurs.

Although researchers have studied memorization using different prompting strategies, nearly all studies have followed a similar two step process (Ishihara, 2023). First, researchers generate outputs based on a particular *training data extraction attack*. Then, researchers assess if the outputs are memorized, a task known as membership inference. Membership inference can be complicated by the facts that not all training corpora are public and that different definitions of memorization necessitate different approaches to the task. This raises the important challenge of defining memorization.

2.1 Defining memorization

Defining memorization presents a challenge because the line between memorization and other forms of knowledge is blurry. While many definitions exist, we discuss the three which provide a framework for our research. To begin, Carlini et al. 2021 proposes *k-eidetic memorization* as a way of identifying information which occurs only sparsely in the training data, as opposed to non-memorized knowledge which occurs more frequently:

Definition 1 (k-Eidetic Memorization)**.** *"A string s is k-eidetic memorized (for $k \geq 1$) by an LM f_θ if s is extractable from f_θ and s appears in at most k examples in the training data $\mathbb{X} : |\{x \in \mathbb{X} : s \subseteq x\}| \leq k$".*

Definition 1 highlights that not all verbatim outputs should be considered memorized to the same extent; Carlini et al. 2021 specifically focus on low k strings which only appear a few times in the training data. However, other researchers have found success using a less strict approach in which they consider any verbatim output to be memorized. One such definition, as Nasr et al. 2023 puts it, is *discoverable memorization*:

Definition 2 (Discoverable Memorization)**.** *"For a model* `Gen` *and an example $[p||x]$ from the training set $\mathbb{X}$, we say that x is discoverably memorized if* $\texttt{Gen}(p) = x$*."*

In other words, *discoverable memorization* refers to a situation in which an instance of training data is divided into a prefix and an expected output, and at inference time, prompting the model with the prefix produces the corresponding output. This concept underlies our second prompting strategy, Wikipedia generation.

However, under other research conditions, the training data may not be accessible, or a particular prefix may have many possible expected outputs in the training data. In these situations, Nasr et al. 2023's conceptualization of memorization as *extractable memorization* makes the most sense:

Definition 3 (Extractable Memorization)**.** *"Given a model with a generation routine* `Gen`*, an example x from the training set $\mathbb{X}$ is extractably memorized if an adversary (without access to $\mathbb{X}$) can construct a prompt p that makes the model produce x (i.e.,* $\texttt{Gen}(p) = x$*)."*

Because so many different links start with "`https://`", we anchor our third prompting strategy, URL generation, in *extractable memorization*. As is discussed later, without peeking into the training corpora, we assume that viable, specific links are instances of *extractable memorization*.

2.2 Empirical findings

Past empirical work has identified several trends regarding memorization. First, models with more parameters generally memorize more (Ishihara, 2023). Second, larger prompts generally induce more memorization (Ishihara, 2023). And third, text which is duplicated in the training data is typically better memorized (Ishihara, 2023). To this last point, k-*eidetic memorization* seeks to disentangle highly duplicated text from the concept of memorization. While these findings are present in our work to some extent, we also add nuance to them by highlighting conditions under which these trends seem not to apply.

2.3 Review of Carlini et al. 2021

While our work contributes to a wide body of literature on memorization, our method builds most directly on Carlini et al. 2021. We describe our ablations in the following section, but before doing so, it is important to recount the work of Carlini et al. 2021 and highlight its gaps, which our research seeks to fill. Carlini et al. 2021 use GPT-2, primarily the XL size, and begin with a baseline prompting strategy in which they prompt the model with only a start-of-sentence token and generate outputs using top-n sampling. Because they generate many more outputs than they can evaluate, they sort these outputs using several sorting strategies which aim to identify which outputs are most likely to be memorized. Finally, they select the top sorted outputs and perform membership inference.

Beyond this baseline, they employ two other generation strategies. In the first, they perform the baseline but with a decaying temperature, which gives the model the opportunity to explore different paths but settle on one with confidence. In the second, they condition their prompt on internet text similar to data in GPT-2's training corpora. Importantly, they compile internet text which they believe is unlikely to be in the model's training data, which corresponds more closely to the notion of *extractable memorization* than *discoverable memorization*. These methods yield significant evidence of memorization. Most impressively, Carlini et al. 2021 induce the output of an 87-character-long string occurring in only one document in the training dataset ($k = 1$). Also within the outputs, they find memorized text ranging from religious texts and Donald Trump tweets to live URLs and Wikipedia entries.

While scripture and Twitter are certainly eye-catching, our attention goes to URLs and Wikipedia entries. Because it is easy to test if a URL is memorized based on whether it is live and specific, inducing the output of URLs is conducive to further research. Similarly, depending on the model, it is also manageable to conjecture if text from Wikipedia is memorized because many datasets feature such text. Moreover, because Carlini et al. 2021's internet conditioning prompt operates under the assumptions of *extractable memorization*, questions abound about how conditioning on prompts known to be in the training corpora impacts memorization. In addition to these opportunities, because Carlini et al. 2021 generate primarily from GPT-2 XL, questions remain too about the impact of model size and about the generalizability of their findings to other models.

3 Ablations

Here, we focus on three main ablations from Carlini et al. 2021. First, we use the suite of Pythia models (Biderman et al., 2023) instead of GPT-2, which provide external validity and transparency to our claims. This is because the Pythia models are trained on an open-source dataset, the Pile (Biderman et al., 2022), allowing for direct comparisons between suspected memorized generations and the training dataset. Additionally, access to the training dataset allows us to test both concepts of memorization from Nasr et al. 2023: *extractable memorization*, in which we prompt the model to generate memorized content without knowledge of the training domain, and *discoverable memorization*, in which we take advantage of our knowledge of the training set to prompt the model (see Section 3.2).

Second, we focus on new prompting strategies. Specifically, we prompt using Wikipedia entries from the training corpus, investigating the impact of prompt length (Section 3.2), and prompt the model to generate URLs (Section 3.3). We find that these prompting strategies generally produce memorized content at a higher frequency than the top-n strategy employed in Carlini et al. 2021. Finally,

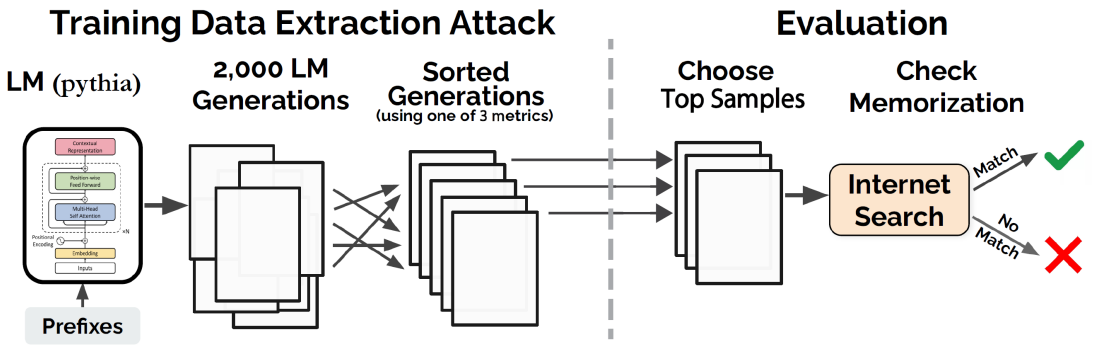

Figure 1: Here, we show our method for extracting memorized samples. We begin by generating 2,000 samples from the Pythia model using top-n, where $n = 40$. We sort the generations using the membership inference strategies shown below, and then use internet searches to evaluate the top samples. Figure modified from (Carlini et al., 2021).

we analyze how model size impacts memorization. Because the Pythia models come in 8 sizes ranging from 70M to 12B parameters, each trained on the same dataset, we compare memorization rates between model sizes.

3.1 Top-n Sampling

For the purpose of reproducing the baseline result of Carlini et al. 2021, we start by attempting to generate memorized data with top-n sampling. Using the 160M parameter Pythia model, we generate 2000 sequences by providing the model with a single token sampled from the vocabulary as a prefix. Then, we generate 100 token sequences using a single beam and a top-n sampling strategy, where we let $n = 40$ and set the temperature to 1. This generation method was employed by Carlini et al. 2021 to let the model choose the most likely sequence without any significant prefix. We note that this form of extraction attack seeks to test discoverable memorization (Definition 2), as the strategy requires no knowledge of the training set. Given the 2000 generated samples, we employ three membership inference strategies to sort the samples:

1. **Perplexity**: the perplexity of the 160m parameter Pythia model. Here, we expect model samples with the lowest perplexity to be more likely to be memorized.

2. **Perplexity Ratio**: the ratio of log-perplexities of the 160M parameter Pythia model and 70M parameter Pythia model. This metric captures the fact that a memorized sequence will have high perplexity for larger models but much lower perplexity for smaller models that have less capacity for memorization. In other words, sequences generated on the 160M parameter model that have unusually low perplexity compared to the perplexity of the 70M parameter model are more likely memorized.

3. **zlib**: the ratio of the (log) of the Pythia perplexity and the zlib entropy (as computed by compressing the text). This strategy mirrors perplexity ratio sorting: zlib entropy is the number of bits of entropy when the sequence is compressed with zlib compression, so it quantifies how "surprising" a given sequence is.

We use these strategies to sort the generations for manual checking. Because the Pile dataset is 825 Gb, searching for individual matches would be highly time consuming. Instead, we follow the lead of Carlini et al. 2021 and search for the top sorted generations on the internet, which we use as a proxy for dataset matches (most text within the dataset can be found on the web). We ignore trivially-memorized text (e.g. the numbers 1 to 100) and repeated substrings. See Figure 1 for an overview of the method.

3.2 Wikipedia Generation

The Pile dataset that EleutherAI used to train the Pythia models is open source (Biderman et al., 2022). We selecte 500 random Wikipedia pages from the Pile and created prefixes to prompt the models by selecting the first 10 and 20 tokens. Using these prefixes, we let the models generate using greedy decoding, which Carlini et al. 2021 found was a good decoding method for producing memorized text from very specific prompts. Then, we compare the generations with the corresponding

Wikipedia pages to check for memorization. We count the number of consecutive identical tokens between the Wikipedia text and the generated texts to analyze memorization. We focused on five different Pythia model sizes: 160 million, 410 million, 1 billion, 1.4 billion, and 2.8 billion parameters. In total, we ran 10 Wikipedia experiments, one for each combination of model size and prefix size (10 or 20 tokens).

3.3 URL Generation

We generated 2000 URLs for each of the Pythia models (Biderman et al., 2023) we used in the Wikipedia experiment (except the 2.8 billion parameter model). Every generation started with the prefix "`https://`". We then generated text with a maximum of 30 tokens. The model was set to generate using top-n sampling with $n = 50$. URLs are perfect targets because we can easily determine if they are memorized by seeing if they point to actual websites or endpoints. We say the link is memorized if it links to a specific webpage other than the site's landing page and does so without redirecting to a new URL. For example, a specific GitHub repository or a specific Twitter user profile page would fall into the memorized link category. To analyze all 8000 URLs, we first removed all URLs that returned HTTP response status codes other than 200. Then, we manually clicked each link, evaluating for memorization based on the criterion above.

4 Results and Discussion

4.1 Top-n Sampling

Searching through the top 50 generations for each sorting metric generates no exact internet matches, except for text from the Apache licence (which Carlini et al. 2021 disregard due to its high prevalence in the training data). This is likely due to a number of factors. First, we search for the verbatim 100 token generated sequence and do not search for substrings. Although splitting the generations into substrings would have likely generated matches, it was often unclear which parts of the generation were more likely to be memorized. Carlini et al. 2021 simply report the number of memorized "non-trivial" substrings, but the authors do not define how they determined what qualified as non-trivial. Most of their reported generations were clearly identifiable (e.g. URLS, news articles, personal contact information, etc.), but we find that our top sorted results are more difficult to break into these smaller substrings. Some examples include blocks of code and snippets of legal proceedings; parts of these could be memorized, but it was unclear how to split them for searching. Second, we generate our sequences using the relatively small 160M parameter Pythia model. This likely lead to fewer memorized substrings, as both Carlini et al. 2021 and our later results demonstrate that larger models are more likely to memorize. Finally, we note that we only generated 2K samples as opposed to the 200K generated in Carlini et al. 2021, so we were less likely to produce matches. (Carlini et al., 2021) produced 59 memorized substrings under this generation method using these sorting metrics, so we would expect to find less than 1 match if the models were roughly equivalent. As explained above, we use a smaller model (160M parameter Pythia model versus the 1.5B parameter GPT2-XL used by Carlini et al. 2021).

4.2 Wikipedia Generation

Model	Mean	Max	Sum	$\geq$1	$\geq$2	$\geq$3	$\geq$4	$\geq$5	$\geq$6
160m-10	0.431	6	215	149	39	13	7	4	3
160m-20	**0.714**	**8**	**357**	**219**	**81**	**30**	**15**	**5**	**3**
410m-10	0.620	**14**	310	185	60	28	14	**8**	**6**
410m-20	**0.762**	11	**381**	**217**	**86**	**42**	**18**	7	4
1b-10	**0.776**	**11**	**388**	210	86	37	17	**13**	**10**
1b-20	0.732	5	366	**217**	**87**	**38**	**19**	5	0
1.4b-10	0.649	**11**	324	191	67	29	13	**8**	**7**
1.4b-20	**0.756**	8	**378**	**223**	**91**	**36**	**18**	7	1
2.8b-10	0.784	**14**	391	208	92	41	20	**12**	**8**
2.8b-20	**0.920**	10	**459**	**242**	**105**	**55**	**30**	11	7

Table 1: Number of consecutive matching tokens between references and generations. The table reports the summary statistics across all 500 Wikipedia samples for each of the 10 model configurations. The columns with $\geq$k refer to the number of model generations that had k or more consecutive matching tokens among the 500 total generations. The Pythia models are identified by the number of parameters and the prefix size. For example, "160m-10" means the Pythia model with 160 million parameters using a prefix length of 10. The numbers in **bold** refer to the highest values between models with the same number of parameters but different prefix lengths.

Our results confirm several insights from other work. For one, we find that the mean number of tokens memorized generally increases with model size, which confirms that larger models are better at memorization. Carlini et al. 2023 finds a log-linear relationship between model size and memorization ability, and our results reflect a similar leveling off. However, while this trend is reflected broadly in our results, we also see evidence that the details are more complicated. Specifically, while larger models memorize more tokens on average, the bulk of this performance increase comes from their ability to output the first and second tokens following the prompt verbatim. Put another way, the advantage

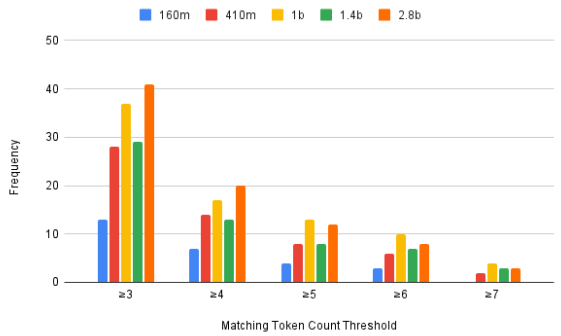

Figure 2: The graph represents the matching token count thresholds ($\geq k$) vs the frequency. In other words, each bar shows the number of times model X produces an output with $\geq k$ matching tokens. This graph only shows results for configurations using a **prefix length of 10**. Each model is represented with a different color, *see the legend*.

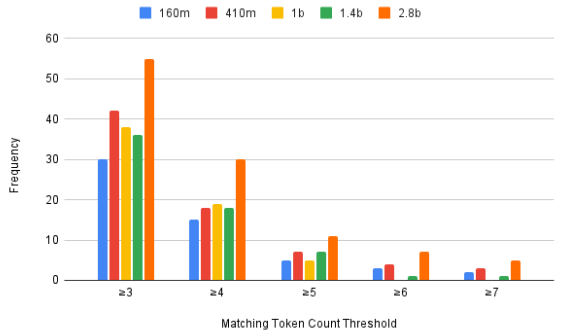

Figure 3: The graph represents the matching token count thresholds ($\geq k$) vs the frequency. In other words, each bar shows the number of times model X produces an output with $\geq k$ matching tokens. This graph only shows results for configurations using a **prefix length of 20**. Each model is represented with a different color, *see the legend*.

of larger models seems to diminish as more tokens are produced: by the time three or more memorized tokens are generated, larger models seem only marginally better at memorization. Based on this insight, we conjecture that part of the reason why larger models may perform better on memorization metrics is not only because they memorize more but also because they have lower perplexity and a better general understanding of language (Biderman et al., 2023). To the extent that this is true, the fact that bigger models memorize more may be overstated.

Also in keeping with past work, we find that in general, the larger prompt window produces more memorization, another finding demonstrated elsewhere in the memorization literature (Ishihara, 2023; Carlini et al., 2023). Once again, however, our results also complicate this assertion. At the long tail, we find that the smaller prompt size was able to generate longer maximum memorized sequences: with a prompt window of 10 tokens, both the 410m and 2.8b models generated a 14-token-long sequence of text, and both the 1b and 1.4b models generated an 11-token-long sequence of text (Table 1). Compare this to the larger prompt window, wherein the longest maximum sequence generated was 11, and only the 410m model produced this (Table 1). One reason for this result may be that when longer prompts are drawn from very short Wikipedia articles, they could be more likely than shorter prompts to precede hard-to-memorize, non-uniform text like the references section of the article. One way to investigate this hypothesis would be to condition on text from the training corpora which is more uniform, like book excerpts, for example. Possible explanations aside, the capacity for shorter prompt windows to produce longer maximum sequences warrants further research.

Type	Text
Prefix:	"Anacithara querna is a species of"
Gen:	"Anacithara querna is a species of sea snail, a marine gastropod mollusk in the family Cancellariidae. Description Distribution References Category:Gastropods described in 1877 Category:Gastropods of Europe Category:Gastropods of Asia Category:Gastropods of Europe Category:Gastropods of Asia Category:Gastropods of Europe Category:Gast"
Ref:	"Anacithara querna is a species of sea snail, a marine gastropod mollusk in the family Horaiclavidae. Description (Original description) The length of the shell attains 5.5 mm, its diameter 2 mm. A thickened pale brown or straw-coloured shell with few ribs, the number only extending to nine on the body whorl. Altogether it is seven-whorled, two of these being apical. The outer lip is thickened. No trace of sinus is perceptible. The columella is straight. The siphonal canal is very short. Distribution This marine species occurs off Iran References External links Tucker, J.K. (2004). "Catalog of recent and fossil turrids (Mollusca: Gastropoda)". Zootaxa. 682:1-1295. querna Category:Gastropods described in 1910"

Table 2: This is an example of the Wikipedia generation method. The prefix was the text used to prompt the model, the generation was the model output, and the reference was the original Wikipedia text. The generation is from the configuration with the Pythia 2.8 billion parameter model using a prefix length of 10 tokens. Also, this generation had the highest matching token count of 14. The blue text is the matching text between the generation and reference that was used to compute the matching token count.

Separate from findings in other work, we also find qualitative evidence for a new property of memorization. We posit that when the prompt involves esoteric information which occurs sparsely in the training dataset (low k), verbatim memorization is more likely to take place. To be clear,

this claim is distinct from the notion of *k-eidetic memorization* in the sense that *k-eidetic memorization* seeks to define memorization while we are suggesting that even under other definitions of memorization, low k text may be more susceptible to memorization. Our claim is actually opposite other work, which has argued that duplicated text is more likely to get memorized (Ishihara, 2023). Table 2 features evidence in support of our claim: this highly esoteric prompt involving the Latin name for a specific species of marine gastropod induces significant memorization. To the contrary, we found that models almost never memorized years or Wikipedia stock phrases like "`[name] is a surname. Notable people with the surname include`." Further research into this possibility would be clarifying.

4.3 URL Generation

		Status Code: 200		Memorized	
Model	Total	Count	Percent	Count	Percent
Pythia 160m	2000	181	9.05%	22	1.10%
Pythia 410m	2000	288	14.40%	41	2.05%
Pythia 1b	2000	**386**	**19.30%**	**91**	**4.55%**
Pythia 1.4b	2000	332	16.60%	86	4.30%

Table 3: The table shows the URL generation results using the status code checks and the manual memorization check described in section 3.3. The **bold** numbers are the highest in each column.

Our results demonstrate that this prompt is a particularly effective attack for inducing memorization. All models outputted memorized URLs at a rate above 1%, with the 1b and 1.4b models exhibiting rates above 4% (Table 3). Compare these rates to those in Nasr et al. 2023: across nine models, the authors' impressively effective attack only induced a token memorization rate above 1% for one model; even this rate was a measly 1.438%. In general, memorization rates in the literature may be deceiving to some extent, since particular prompts, like our URL generation attack, can induce memorization at much higher rates. As an aside, we believe that our reported rates may actually underestimate the extent of link memorization, because many links which garnered the 200 Status Code pointed to YouTube and Facebook pages that were marked as no longer live, and we did not count these cases as memorized.

The impressive performance of this attack, however, is as alarming as it is exciting because memorized links can point to very specific content. To

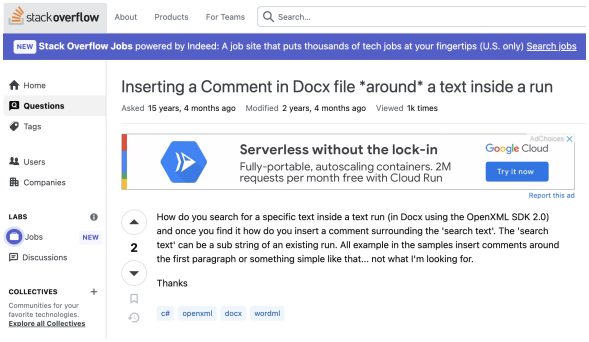

Figure 4: screenshot of website result for the generated URL, https://stackoverflow.com/questions/390135/

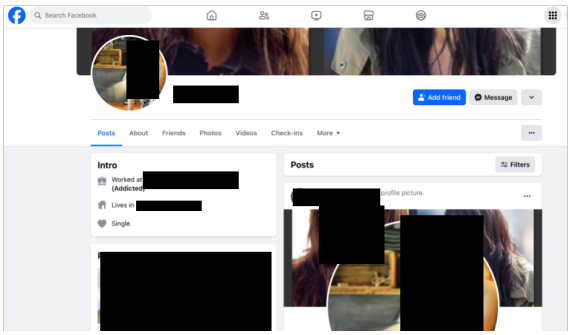

Figure 5: screenshot of website result for the generated URL, https://www.facebook.com/danseo2/

this point, Figure 4 shows that one memorized link led us to a specific Stack Overflow page where individual user comments were on display. Figure 5 is perhaps even more disconcerting, as it shows a memorized link to an individual's Facebook page, with their name, photos, and other personal information on display. To make matters worse, this example is but one of many personal accounts present among memorized URLs. In addition to Facebook, we also found URLs pointing to individuals' YouTube, GitHub, Instagram, Weibo, and Flickr accounts. These outputs raise significant contextual privacy concerns.

We also noticed that particular domain names–like YouTube and GitHub–were disproportionately represented among both links which garnered a 200 Status Code and memorized links. This observation leads us to two conclusions. First, as opposed to the models' behavior in Wikipedia generation, we conclude that because popular domains were overrepresented, this is an instance of duplicated data being memorized at a higher rate. Second, we conclude that perhaps new, even more dangerous attacks could be founded on this observation. For example, by conditioning on an entire popular domain name (e.g. prompting "https://facebook.com"), an adver-

sary could autogenerate many memorized links in one fell swoop. The broader point here is that even under a relatively strict definition like *extractable memorization*, an adversary could use increasingly specific prompts to generate memorized content en masse. Furthermore, while we used top-n sampling here, greedy generation could exacerbate the efficacy of such an attack. Future research into both the viability of such a strategy and ways to protect against it is desirable.

4.4 Limitations

While there are many ways one could enhance our research, three main limitations should inform future work. First, we were limited by the scale of our generating and computing abilities. While we generated only thousands of examples per prompting strategy, other studies have used greater scale to their advantage to induce memorization. Carlini et al. 2021, for example, generated 200,000 examples for their top-n baseline. In this sense, our limited scale may have inhibited our top-n sampling in particular. Second, when conditioning on longer prompts, like in the case of our Wikipedia generation attack, other studies have found success using temperature decay, wherein the model's temperature is reduced as an output is generated (Carlini et al., 2021). Per this finding, our Wikipedia generation results may have been encumbered by our greedy approach. Third, other definitions of memorization than the three we reviewed in Section 2.1 afford more flexibility. Because many of our outputs seemed to be memorized text with negligible modifications, using an approximate definition of memorization, like the BLEU score between the output and instance in the training dataset, could provide for more nuanced study of memorization (Ippolito et al., 2023; Ishihara, 2023).

5 Conclusion

Memorization remains an important research area. Our work has made progress in understanding what kinds of prompts induce memorization, what kind of data is vulnerable to being memorized, and the impacts of model and prompt window size on memorization. However, for every question our research contributed to answering, it also raised many more research topics. Of particular importance will be how to control memorization and defend against *training data extraction attacks*. Our privacy, copyright rights, and understanding of LLMs hang in the balance.

Acknowledgements

Thank you to the COS484 staff for teaching us and to Colin Wang in particular for advising us.

Code Availability

All project code is available on GitHub: https://github.com/ckia25/nlp-data-extraction.

References

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the pile. *arXiv preprint arXiv:2201.07311*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A. Choquette-Choo, and Nicholas Carlini. 2023. Preventing verbatim memorization in language models gives a false sense of privacy.

Shotaro Ishihara. 2023. Training data extraction from pre-trained language models: A survey. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 260–275, Toronto, Canada. Association for Computational Linguistics.

Edward Lanquist and Jeremy Ray. 2024. Artificial intelligence and copyright law: The nyt v. openai – fair use implications of generative ai.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.