Extracting Training Data from GPT-2

Jaydeep Borkar
CS 7150
Northeastern University
Khoury College of Computer Sciences
borkar.j@northeastern.edu

1 Introduction

Recently, it has been found that large language models like GPT-2 memorize training data [5, 4] which can be extracted by an adversary by simply querying them. The memorized data might include personal information like individual names, phone numbers, emails, addresses, fax, code, etc. Hence, the memorization problem can pose a serious privacy concern. [8] were able to demonstrate how encoder models like BERT can memorize sensitive healthcare data like the patient's name and health condition. This makes it very important to study the memorization of training data by Large Language Models (LLMs). So far, the majority of the work has been focused on studying memorization for pre-trained models. Little attention has been given to how fine-tuned models might memorize their training data. There has been some work in understanding memorization for fine-tuned models by [9], however, there are some notable differences: (1) They focus on extracting personal information through classification tasks. (2) They insert artificial secrets in the training data and then try to extract it during inference. Contrary to that, this work focuses on extracting naturally occurring sequences in the training data by performing the text generation task. In this work, we particularly show: (1) How fine-tuned models memorize training data in a similar fashion as pre-trained models. (2) How pre-trained models pass their memorized data to the fine-tuning phase which is then extracted by querying fine-tuned models.

Memorization is definitely useful for generalization. For example- the model must memorize the correct spelling of individual words. But memorizing personal information and data that might hurt generalization is bad. As you can see in Figure 2, there is blue phase where the model only memorizes training data but doesn't generalize at all. This is the memorization phase. These phases were proposed by [9]

For our baseline, we first extract training data from GPT-2 Large. Then we proceed to extract training data from GPT-2 Large fine-tuned on the Wikitext103 dataset [1]. We adapt the code from the open-source implementation by [5][2].

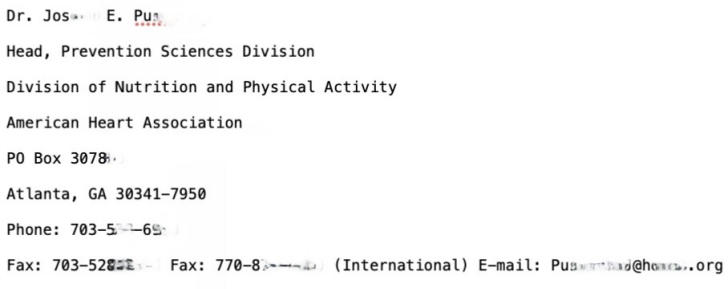

Figure 1: Personal information extracted from GPT-2 L which is memorized.

[1] https://www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

[2] https://github.com/ftramer/LM_Memorization

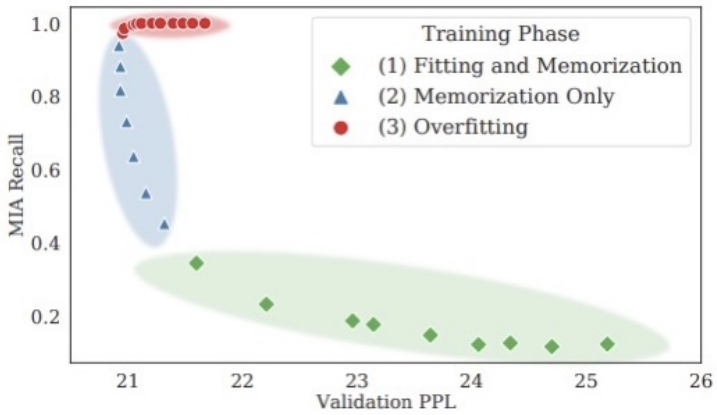

Figure 2: Training and Memorization phase for LLM.

2 Related Work

2.1 Language Modeling

The main objective of language modeling is to predict the next word based on the current word and the context of all the preceding words. Language models (LMs) predict the next word which they feel is likely to follow the current word based on the probability distribution. RNNs [10] have been great at performing the task of language modeling due to their recurrent nature until Transformers [11] came into the picture. The Transformer architecture has a special self-attention mechanism and encoder-decoder architecture which makes them perform best on the language modeling task. GPT models are transformer models that are pre-trained on a large amount of data.

2.2 Data Extraction

[4] were the first to show that generative models can leak the private data of the users. There was a follow-up work [5, 3] that talked about data extraction attacks for large language models like GPT-2. [8] were the first to show how encoder models like BERT [6] can memorize sensitive information like a patient's name and health condition if trained on clinical notes. [9] show how fine-tuned large language models memorize training data but they focus on extracting artificially-inserted sequences and on classification tasks.

3 Methods

The methods section is divided into three parts: (1) generating samples from GPT-2 (2) Sorting the generated samples based on their likelihood of being potentially memorized (3) Evaluating the potentially memorized samples to check for actual memorization. The methods are adapted from the preliminary work on extracting training data from large language models [5]. Figure 3 from [5] summarizes the entire set-up and methodology.

3.1 Generating Samples from GPT-2

We use two methods to generate samples from GPT-2. For both the methods, we generate samples with token size of 256.

3.1.1 Top-n Sampling

As discussed in the section above, the main objective of language modeling is to assign probability likelihood to a sequence of tokens. The model predicts the next token based on the probability distribution of all the tokens in its vocabulary. In Top-n sampling method, the model only decides

from the top n potential tokens. This helps the model to predict a token that is closely related to the current word. Figure shows the top-n sampling method. We choose n=40 for sample generation.

3.1.2 Internet Conditioning

We collect tokens from the Common Crawl dataset and use them (5-10 tokens) as prefixes to prompt GPT-2 to generate samples. The reason we do this is because GPT-2 is trained on internet data. And Common Crawl dataset has all the data that has been crawled from the public internet. So if we prompt GPT-2 using tokens from Common Crawl, it slightly increases the chance of GPT-2 generating memorized data (if at all) and it will also help the model to generate a variety of text which might not be possible by just using a vanilla top-n sampling method.

3.2 Sorting the Generated Samples

Now that we have the generated samples, our next task is to sort them according to their likelihood of being memorized. We cannot evaluate all the samples for potential memorization (especially if the number of samples are really high). So we need a small pool of potentially memorized examples that we can evaluate. This is the reason we do this sorting step to get a small subset of potentially memorized samples. To sort the samples, we use three methods.

3.2.1 *Perplexity*

Language models assign a probability to a sequence based on the maximum likelihood estimate (MLE). Perplexity is the inverse of probability. It tells us how surprised a model is after looking at a specific sequence. If a model gives low perplexity to a specific sample, it means that it is confident that it saw that specific sample in its training data.

3.2.2 *zlib*

We take a ratio of GPT-2 perplexity and zlib entropy. Zlib entropy basically helps us to detect any repeated patterns in a potentially memorized sample. If there are a lot of repetitions, the zlib entropy for the sample is going to be high. If there aren't enough repetitions, the zlib entropy will be low.

3.2.3 *lowercase*

We take the ratio of GPT-2 perplexity on the original sample and lowercase sample. If the ratio is high, it means that the model is expecting the text to be in a specific case. Which is memorization.

3.3 Evaluation

Now that we have the sorted samples, we do the evaluation to find samples that are actually memorized. Specifically, we perform the evaluation in two following ways.

3.3.1 Evaluating memorization for the fine-tuned model

To evaluate samples for potential memorization, we perform an n-gram search to find common ngrams between samples and the Wikitext103 dataset. We perform the search from n=10 to n=256. This helps to identify any sequences that are memorized as-is from the training data. To detect memorized sequences with size n<10, for example, names and email addresses, we perform a manual search through the samples. We use NLT-K library to preprocess the text and find all possible ngrams. If we find common ngrams or sequences, it means they were memorized by the fine-tuned model.

3.3.2 Evaluating memorization for pre-trained GPT-2

To evaluate memorization for pre-trained GPT-2, we do an internet search for the sample as GPT-2 is trained on public internet data. If the search results in an exact match for any substrings, it means they were memorized by GPT-2.

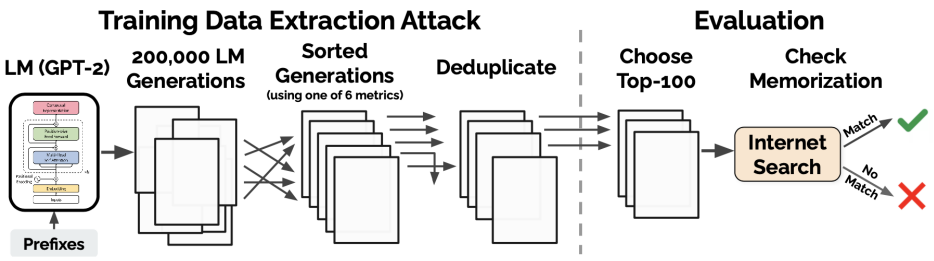

Figure 3: Methodology for training data extraction attack.

4 Experiments

4.1 Generating Samples

We generate 1000 samples using both the sampling methods discussed above for both the model: GPT-2 and GPT-2 fine-tuned on Wikitext103. We got the fine-tuned model from Hugging Face [1]. For the model size, we take GPT-2 Large which has **774M** parameters. The fine-tuned model is also large size. We use a **A100** GPU to reduce the time taken to generate samples and calculate perplexity and other metrics.

4.2 Sorting Samples

We sort all the 1000 generated samples for both models using each of the three metrics discussed above: perplexity, zlib, and lowercase. The samples are sorted according to the likelihood of them being potentially memorized. We pick the first 500 samples for evaluation after sorting the samples. So in the end, we have three datasets of 500 samples each to check for potential memorization.

4.3 Evaluation

For the fine-tuned model, we perform a common ngram search from n=10 to n=256 as discussed above. For n<10, we do a manual search through the samples. The computation time to perform ngram search was significantly higher for n>30. This is because each sample was divided into all possible ngrams of size 256. And doing this for all 500 samples, and then for the entire Wikitext103 dataset that has **100M** tokens is definitely computationally expensive. We needed a **128 GB** memory to support this search. For the GPT-2 model, we simply do an internet search as it's trained on web data.

5 Results

5.1 Extracting Training Data from GPT-2 L (baseline)

We were able to extract personal information like actual phone numbers, names, email IDs, fax numbers, Twitter handles, code, some tracking numbers, a list of universities, years, months, etc. Figure 1 shows the extracted sample from GPT-2 that contains an actual phone number, name, email, and other private information. We have masked the image for privacy reasons and the individuals have not been notified that their data was memorized by GPT-2. Table 1 shows all the extracted content categories from GPT-2 which was generated using internet conditioning and sorted using zlib. Figure 4 shows the scatter plot for samples generated from GPT-2. The samples in red are memorized and the samples in grey are not memorized. As you can see, there is a clear separation between both of them and memorized samples have lower perplexity (which indicates that GPT-2 saw them during training).

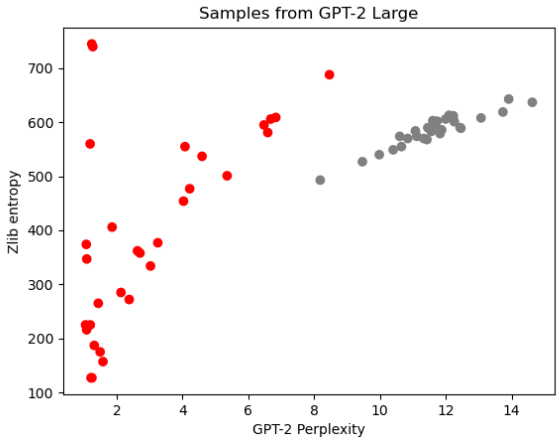

Figure 4: Scatter plot of perplexity vs zlib entropy for pre-trained GPT-2 samples. The data points in red denote memorized examples and those in grey denote non-memorized examples.

Personal information: phone no, names, email IDs, fax, address, Twitter handles.
List of named entities in a specific order: years and months, university names, numbers, movies, and countries.
Code
URLs

Table 1: Categories of extracted content from GPT-2 L.

5.2 Extracting Training Data from GPT-2 fine-tuned on Wikitext103

Using an ngram search, we were able to find the memorized samples. Table 2 shows the categories of all the extracted samples. The first two examples in the table are the samples generated by the fine-tuned models. The entire text in bold is memorized, which is a lot of memorization. The second example also reveals names. Though these are celebrity names, it's a clear indication that if the fine-tuned dataset has names of individual people, they might be leaked too. Figure 5 shows the scatter plot for samples generated from fine-tuned GPT-2. The samples in red are memorized and the samples in grey are not memorized. As you can see, there is a clear separation between both of them and memorized samples have lower perplexity (which indicates that fine-tuned GPT-2 model saw them during training).

5.3 Extracting training data which is memorized by pre-trained GPT-2 but leaked by fine-tuned model

We were also able to extract a bunch of URls (see Figure 6) which did not belong to the Wikitext103 dataset that we used for fine-tuning. All the extracted URLs resulted in a live page after Google search. A possible reasoning here is that these URLs were memorized by the GPT-2 model during pre-training and then were passed on during the fine-tuning phase to the fine-tuned model. There has been some work [7] that talks about pre-trained models forgetting some of their knowledge as a result of fine-tuning. In spite of that, we were able to extract URLs memorized by the pre-trained model simply by generating samples from the fine-tuned mode. To prepare a pool of potential memorized examples in this scenario, we sorted the samples generated by the fine-tuned model according to pre-trained GPT-2's metrics.

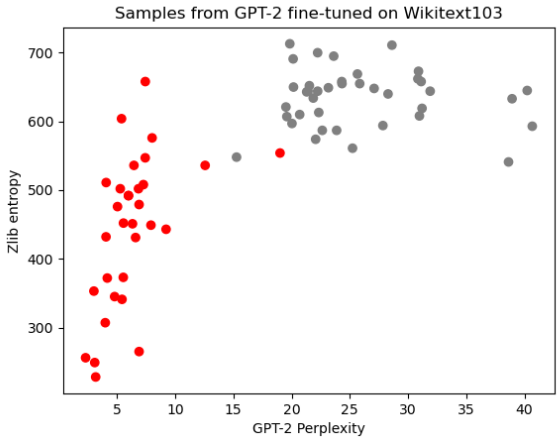

Figure 5: Scatter plot of perplexity vs zlib entropy for fine-tuned GPT-2 samples. The data points in red denote memorized examples and those in grey denote non-memorized examples.

The Boat Race is a side @-@ by @-@ side rowing competition between the University of Oxford (sometimes referred to as the " Dark Blues ") and the University of Cambridge (sometimes referred to as the " Light Blues "). The race was first held in 1829, and since 1845 has taken place on the 4 @.@ 2 @-@ mile (6 @.@ 8 km) Championship Course on the River Thames in southwest London. The rivalry is a major point of honour between the two universities ; it is followed throughout the United Kingdom
The show centers on FBI special agents Fox Mulder (David Duchovny) and Dana Scully (Gillian Anderson) who work on cases linked to the paranormal, called X @-@ Files. Mulder is a believer in the paranormal, while the skeptical Scully has been assigned to debunk his work. **In this episode**
Named entities such as: Recording Industry Association of America (RIAA), WSDOT, John F. Kennedy Presidential Library and Museum, the University of Illinois at Urbana – Champaign, names of musicians, names of countries in alphabetic order, songs.
URLs, websites, and correct conversions such as 52 miles per hour (84 km / h)
Historic dates: in the American Revolutionary War (1775 – 1783)

Table 2: Categories of extracted content from GPT-2 L fine-tuned on Wikitext103. The first two examples are the generated samples and the entire text in bold is memorized.

Figure 6: Some of the URLs extracted from fine-tuned GPT-2 which were memorized by pre-trained GPT-2 and do not belong to the fine-tuning dataset.

6 Conclusion and Future Work

There has been work on extracting pre-training data from large language models but not a lot of focus has been on understanding how and what kind of memorization is inherited by the fine-tuned models. This was one of the motivations behind this project. It was interesting to see that fine-tuned models do memorize their training data just like pre-trained models. Apart from memorizing small sequences like names, emails, URLLs, etc, they can also memorize much larger sequences (for example around 200+ tokens). The results from this work also indicate that pre-trained models might be passing some of their memorized data to fine-tuned models which could be extracted by querying the fine-tuned models. This would also allow to trace back the source model in the case where someone misused a LLM by fine-tuning it on their specific datasets. Some of these results for fine-tuned models are novel to the best of my knowledge. I'd be interested in doing a more systemic study along this direction and continuing the work. These are some of the potential future directions:

1. Quantifying memorization for fine-tuned models. Pre-trained models have an upper bound of 5% [3].
2. Studying memorization for different sizes like XL, M, and S as memorization is correlated with model size.
3. Understanding how much memorized pre-trained data is passed on in fine-tuning.
4. Studying memorization for multi-modal models like GPT-4. Stable diffusion memorizes training images [2] and we know that LLMs memorize text. So it would be interesting to do something at the intersection.

References

[1] Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. PMLR, 2022.

[2] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models, 2023.

[3] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models, 2022.

[4] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA, 2019. USENIX Association.

[5] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Tianxing He, Myle Ott, Bing Liu, Jun Liu, James Glass, Kyunghyun Cho, and Fuchun Peng. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, pages 1121–1133. Association for Computational Linguistics (ACL), 2021. Publisher Copyright: © 2021 Association for Computational Linguistics; 16th Conference of the

European Chapter of the Associationfor Computational Linguistics, EACL 2021 ; Conference date: 19-04-2021 Through 23-04-2021.

[8] Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron Wallace. Does BERT pretrained on clinical notes reveal sensitive data? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 946–959, Online, June 2021. Association for Computational Linguistics.

[9] Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1816–1826, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[10] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.