Memorization in Fine-Tuned Large Language Models

Danil Savine

Research done while at PRAIRIE Research Institute, PSL University

Supervisors: Prof. Jamal Atif, Prof. Olivier Cappé, Dr. Muni Sreenivas Pydi

Reviewers: Prof. Benjamin Negrevergne

Version: Sent to Reviewer

September 1, 2024

Abstract

This study investigates the mechanisms and factors influencing memorization in fine-tuned large language models (LLMs), with a focus on the medical domain due to its privacy-sensitive nature. We examine how different aspects of the fine-tuning process affect a model's propensity to memorize training data, using the PHEE dataset of pharmacovigilance events.

Our research employs two main approaches: a membership inference attack to detect memorized data, and a generation task with prompted prefixes to assess verbatim reproduction. We analyze the impact of adapting different weight matrices in the transformer architecture, the relationship between perplexity and memorization, and the effect of increasing the rank in low-rank adaptation (LoRA) fine-tuning.

Key findings include: (1) Value (W^V) and Output (W^O) matrices contribute more significantly to memorization compared to Query (W^Q) and Key (W^K) matrices; (2) Lower perplexity in the fine-tuned model correlates with increased memorization; (3) Higher LoRA ranks lead to increased memorization, but with diminishing returns at higher ranks.

These results provide insights into the trade-offs between model performance and privacy risks in fine-tuned LLMs. Our findings have implications for developing more effective and responsible strategies for adapting large language models while managing data privacy concerns.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and research problem

Large Language Models (LLMs) have demonstrated remarkable capabilities in various natural language processing tasks. However, their tendency to memorize training data poses significant privacy concerns. This research problem focuses on understanding the mechanisms and factors influencing memorization in LLMs, particularly after the fine-tuning process.

We focus on the medical domain as a significant number of documents are fine-tuned on patient-doctor dialogues that create an increasing risk of privacy concerns.

Memorization in fine-tuned pre-trained LLMs

Memorization occurs when a model learns to reproduce specific training examples, rather than generalizing from the data. In the context of LLMs and fine-tuning, this phenomenon raises several critical questions:

- How do the fine-tuning parameters influence propensity for memorization?

- What are the characteristics of the data that is more likely to be memorized?

- What defences can be used to protect against memorization while keeping model's performance?

Understanding these aspects is crucial for developing more robust and privacy-preserving language models.

Several factors may influence the extent and nature of memorization in LLMs during fine-tuning. This research seeks to investigate the impact of:

1. **Adapted matrices:** Which weight matrices in the model architecture are most susceptible to memorization when adapted? Are certain components more prone to memorizing training data?

2. **Perplexity of fine-tuned text:** How does the perplexity of the training data correlate with its likelihood of being memorized? Does lower perplexity (i.e., more predictable text) lead to increased memorization?

3. **Rank of adaptation:** In the context of low-rank adaptation methods like LoRA [17], how does the rank of the adaptation matrices affect memorization?

In order to assess memorization, we use two attacks. A membership inference attack when given a set of text within and out of training set but coming from a similar distribution. And a generation aimed at assessing the model ability to complete phrases from the training set when given its beginning.

Chapter 2

Background and Related Work

2.1 Large Language Models

2.1.1 Language Models: Foundations and Innovations

Language models (LMs) serve as state-of-the-art in contemporary natural language processing [9, 15, 22, 23, 25]. A prevalent approach in LM training employs a "next-step prediction" objective [3, 15, 19, 23]. This method constructs a generative model of the distribution $\Pr(x_1, x_2, \ldots, x_n)$, where $x_1, x_2, \ldots, x_n$ represents a token sequence from vocabulary V, utilizing the chain rule of probability:

$$\Pr(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} \Pr(x_i | x_1, \ldots, x_{i-1})$$

State-of-the-art LMs employ neural networks to estimate this probability distribution. Let $f_\theta(x_i | x_1, \ldots, x_{i-1})$ denote the likelihood of token x_i when evaluating neural network f with parameters θ. While recurrent neural networks (RNNs) [13, 19] were once predominant, attention-based models [2], particularly Transformer LMs [30], now dominate the field.

2.1.2 Training Objective

LM training aims to maximize the probability of the data in a training set X by minimizing the loss function:

$$L(\theta) = -\log \prod_{i=1}^{n} f_\theta(x_i | x_1, \ldots, x_{i-1})$$

Theoretically, the optimal solution involves memorizing the next token for every prefix in the training set. However, the use of massive datasets in state-of-the-art LMs mitigates significant memorization, resulting in nearly identical training and test losses [5, 24, 25].

2.1.3 Text Generation

LMs generate text by iteratively sampling:

$$\hat{x}_{i+1} \sim f_\theta(x_{i+1} | x_1, \ldots, x_i)$$
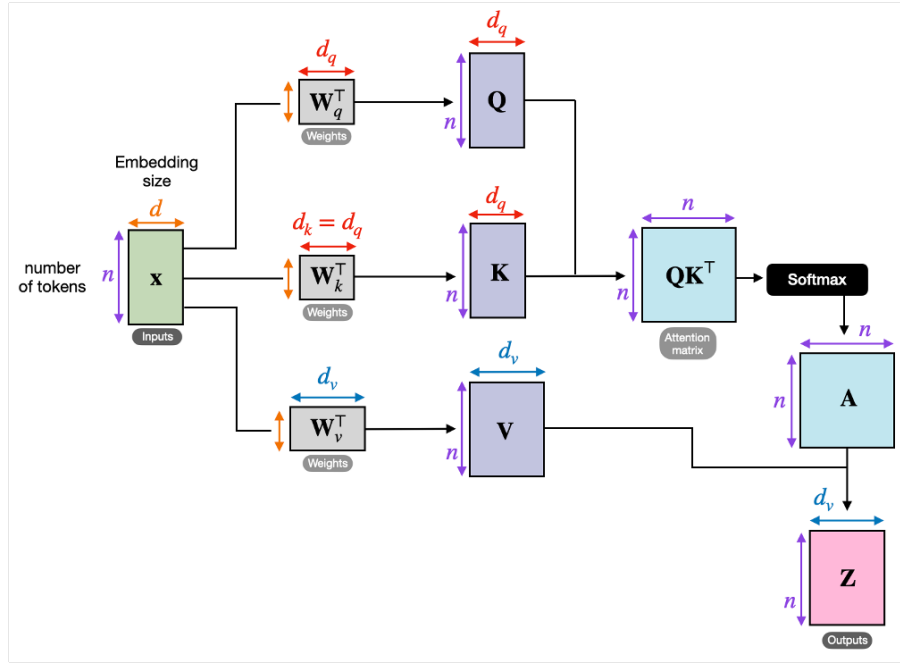
Figure 2.1: Illustration of the attention mechanism from https://sebastianraschka.com/

and feeding $\hat{x}_{i+1}$ back into the model. Variations include greedy sampling (selecting the most probable token) and top-n sampling[1] [11].

2.1.4   Transformer Architecture

The Transformer [30] has revolutionized natural language processing. It follows an encoder-decoder structure, replacing recurrent layers with multi-head self-attention layers and position-wise feed-forward networks. Both encoder and decoder comprise stacks of identical layers, enabling parallel processing and capturing long-range dependencies more effectively than traditional RNNs.

Attention mechanism. The attention mechanism is the core component of the Transformer architecture. It allows the model to focus on different parts of the input sequence when producing each element of the output sequence.

The basic attention function is called "Scaled Dot-Product Attention". It operates on queries (Q), keys (K), and values (V). The attention output is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.1}$$

where d_k is the dimension of the keys. The Figure 2.1 illustrates this computation. The scaling factor $\frac{1}{\sqrt{d_k}}$ is introduced to counteract the effect of the dot products growing large in magnitude for large values of d_k.

Instead of performing a single attention function, the Transformer employs multi-head attention, which allows the model to jointly attend to information from different representation sub-spaces at different positions. Multi-head attention is defined as:

---

[1]Also known as nucleus sampling.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \quad (2.2)$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.3)$$

Here, the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

2.1.5 LLaMA-2.

LLaMA-2 (Large Language Model Meta AI 2), developed by [29], represents a more recent advancement in language models. Notable characteristics include:

- Available in 7B, 13B, and 70B parameter versions

- Trained on 2 trillion tokens of diverse data

- Has a 4096-token context

- Incorporates constitutional AI techniques for improved safety

LLaMA-2 builds upon the GPT-3 architecture [5] with modifications such as RM-SNorm for layer normalization, SwiGLU activation function, and rotary positional embeddings. It demonstrates strong performance across a wide range of NLP tasks and has been released under a permissive license for research and commercial use.

- **LlamaForCausalLM implemeted by HuggingFace**

  model LlamaModel
  - embed_tokens: Embedding(32000, 4096)
  - layers: ModuleList
    * (0-31): 32 x LlamaDecoderLayer
    self_attn LlamaSdpaAttention
    · q_proj: Linear(in_features=4096, out_features=4096, bias=False)
    · k_proj: Linear(in_features=4096, out_features=4096, bias=False)
    · v_proj: Linear(in_features=4096, out_features=4096, bias=False)
    · o_proj: Linear(in_features=4096, out_features=4096, bias=False)
    · rotary_emb: LlamaRotaryEmbedding()
    mlp LlamaMLP
    · gate_proj: Linear(in_features=4096, out_features=11008, bias=False)
    · up_proj: Linear(in_features=4096, out_features=11008, bias=False)
    · down_proj: Linear(in_features=11008, out_features=4096, bias=False)
    · act_fn: SiLU()
    input_layernorm LlamaRMSNorm()
    post_attention_layernorm LlamaRMSNorm()

   – norm: LlamaRMSNorm()

   – rotary_emb: LlamaRotaryEmbedding()

 lm_head Linear(in_features=4096, out_features=32000, bias=False)

2.2   Fine-Tuning and Low Rank Adaptations

Fine-Tuning While pre-trained models demonstrate impressive capabilities across various tasks, they can be further improved for specific applications through fine-tuning. Fine-tuning involves continuing the training of a pre-trained model on a smaller, task-specific dataset to adapt it to a particular domain or task [15].

The process of fine-tuning typically involves selecting a pre-trained model, preparing a task-specific dataset, adjusting the model's parameters using the new data, evaluating the fine-tuned model on the target task.

However, fine-tuning large language models presents challenges, including computational resources, potential for catastrophic forgetting, and the need for large amounts of task-specific data.

LoRA: Low-Rank Adaptation of Large Language Models

To address the challenge of computation resources associated with fine-tuning large language models, [16] introduced LoRA (Low-Rank Adaptation). LoRA is an efficient fine-tuning method that significantly reduces the number of trainable parameters while maintaining model performance.

Key features of LoRA include:

- Freezing the pre-trained model weights

- Adding trainable low-rank decomposition matrices to each layer

- Significantly reducing the number of parameters to be fine-tuned

- Enabling faster training and lower memory requirements

The LoRA approach can be formalized as follows. For a pre-trained weight matrix $W \in \mathbb{R}^{d \times k}$, LoRA defines its update as:

$$W + \Delta W = W + BA \tag{2.4}$$

where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are low-rank matrices, and r is the rank, typically much smaller than d and k.

LoRA has shown promising results in adapting large language models to specific tasks with minimal computational overhead. It allows for efficient fine-tuning of models making them more accessible for task-specific applications and research.

2.3 Data Extraction Attacks

2.3.1 Overview

We follow the approach proposed by [7] to explore extraction techniques. Let $D_{training} = \{x_i\}_{i=1}^n$ be the corpus of the training set, where each sequence x_i is divided into $x_i = [p_i \parallel s_i]$, with p_i being a length-k prefix and s_i being the suffix. We define $P = \{p_i\}_{i=1}^n$ as the set of all prefixes and $S = \{s_i\}_{i=1}^n$ as the set of all suffixes. The value of k is chosen as the midpoint of each sequence length.

Following [31], we define memorization as:

Definition 2.3.1 (Memorization). Given a model function f, memorization occurs when the model output $f(p_i)$ contains information of any $s_j \in S$, formalized as $D(f(p_i), s_j) =$ True, where D is a discriminative function assessing the similarity between two texts.

An alternative definition of memorization comes from [32] citing [12].

Definition 2.3.2 (Counterfactual Memorization). Given a training algorithm A that maps a training dataset D to a trained model f, and a measure $M(f, x)$ of the performance of f on a specific example x, the counterfactual memorization of a training example x in D is given by

$$\text{mem}(x) \triangleq \underbrace{\mathbb{E}_{S \subset D, x \in S}[M(A(S), x)]}_{\text{performance on } x \text{ when trained with } x} - \underbrace{\mathbb{E}_{S' \subset D, x \notin S}[M(A(S'), x)]}_{\text{performance on } x \text{ when not trained with } x} , \qquad (2.5)$$

where S and S' are subsets of training examples sampled from D. The expectation is taken with respect to the random sampling of S and S', as well as the randomness in the training algorithm A.

Now that we have defined memorization, let's dive into language model data extraction attacks. They usually follow from a two-step procedure as in [8]:

1. Generate candidate text with the model (sampling)

2. Predicting which generated text is actually memorized text (membership inference)

Subsequently, we employ a membership inference attack to eliminate generated samples unlikely to contain memorized text.

2.3.2 Measuring successful memorization

Various discriminative functions D (as mentioned in Definition 2.3.1) can be employed to measure successful memorization:

- **Specific keywords:** Let K be a set of predefined keywords. The discriminative function can be defined as:

$$D_{\text{keywords}}(f(p_i), s_j) = \begin{cases} \text{True,} & \text{if } \exists k \in K : k \in f(p_i) \wedge k \in s_j \\ \text{False,} & \text{otherwise} \end{cases}$$

- **Verbatim memorization** [8]: Let $\text{LCS}(a, b)$ be the longest common substring function. The discriminative function can be defined as:

$$D_{\text{verbatim}}(f(p_i), s_j) = \begin{cases} \text{True}, & \text{if } \frac{|\text{LCS}(f(p_i), s_j)|}{\min(|f(p_i)|, |s_j|)} \geq \tau \\ \text{False}, & \text{otherwise} \end{cases}$$

  where τ is a predefined threshold.

- **BLEU distance:** Let $\text{BLEU}(a, b)$ be the BLEU [21] score function. The discriminative function can be defined as:

$$D_{\text{BLEU}}(f(p_i), s_j) = \begin{cases} \text{True}, & \text{if } \text{BLEU}(f(p_i), s_j) \geq \theta \\ \text{False}, & \text{otherwise} \end{cases}$$

  where θ is a predefined threshold.

- **LLM-based reformulation detection** [31]: Let $\text{LLM}(a, b)$ be a function that returns True if an LLM determines a is a reformulation or summary of b. The discriminative function can be defined as:

$$D_{\text{LLM}}(f(p_i), s_j) = \text{LLM}(f(p_i), s_j)$$

- **Plagiarism detection with cosine similarity:** Let $\text{TF-IDF}(a)$ be the TF-IDF vector of text a, and $\cos(u, v)$ be the cosine similarity between vectors u and v. The discriminative function can be defined as:

$$D_{\text{plagiarism}}(f(p_i), s_j) = \begin{cases} \text{True}, & \text{if } \cos(\text{TF-IDF}(f(p_i)), \text{TF-IDF}(s_j)) \geq \lambda \\ \text{False}, & \text{otherwise} \end{cases}$$

  where λ is a predefined threshold.

2.3.3 Sampling of Candidates

Sampling Strategy

In large language models text generation is accomplished through iterative sampling: $\hat{x}_{i+1} \sim f_\theta(x_{i+1} | x_1, \ldots, x_i)$

- **Top-n sampling**: Setting all but the top-n probabilities to zero and renormalizing before sampling. Sampling over several tokens in beam-search tree is computationally expensive and in practice yields a similar result to the top-n sampling.

- **Temperature:** One can artificially "flatten" this probability distribution to make the model less confident by replacing the output $\text{softmax}(z)$ with $\text{softmax}(z/t)$, for $t > 1$. Here, t is called the temperature. The softmax function with temperature is defined as: $\text{softmax}(z/t)_i = \frac{\exp(z_i/t)}{\sum_{j=1}^n \exp(z_j/t)}$ A higher temperature causes the model to be less confident and more diverse in its output. This is because as t increases, the differences between the logits z_i become less pronounced, leading to a more uniform probability distribution over the possible next tokens. Greedy sampling (drawing the most probably token) would be an extreme case where the temperature would be set to 0.

- **Decaying temperature:** To balance exploration and exploitation, an option is to use a decaying softmax temperature. For example, the temperature t could start at 10 and decays to 1 over the first 20 tokens (approximately 10% of the sequence length). This approach allows for diverse prefix exploration while enabling the model to follow high-confidence paths.

Prefix Selection

Prefix selection can significantly impact the extraction process. Options include:

- **Beginning of sequence token** (empty)

- **Text from a similar distribution** (e.g., same domain like medical for medical LLMs, random internet text)

- **Known text from the dataset** (e.g., "my social security number is ...", "John Doe suffers from")

2.3.4 Membership Inference Among Candidates

Given a set of samples from the model, the problem of training data extraction can be reduced to membership inference: predicting whether each sample was present in the training data [27]. Basic membership inference attacks rely on the observation that models tend to assign higher confidence to examples present in the training data [26].

For language models, we use perplexity as a natural likelihood measure. Given a sequence of tokens $x_1, \ldots, x_n$, the perplexity is defined as:

$$P = \exp\left(-\frac{1}{n} \sum_{i=1}^{n} \log f_\theta(x_i | x_1, \ldots, x_{i-1})\right) \tag{2.6}$$

A low perplexity indicates that the model is not "surprised" by the sequence and has assigned, on average, a high probability to each subsequent token.

Comparison Techniques

To refine our membership inference, we "contrast" the perplexity of the model with other metrics. The intuition comes from the definition of counterfactual memorization [32].
Changing the "contrastive" algorithm:

- **Comparing to other neural language models**: use smaller models (e.g., GPT-2 Small and Medium) trained on the same dataset, exploiting their lower capacity for memorization.

- **In the case of fine-tuned model, comparing to the base model**

- **Comparing to zlib compression**: compute the zlib entropy of the text and use the ratio of the target model perplexity to zlib entropy as a membership inference metric. This was mostly used in [8] in order to disqualify candidates with low perplexity sequences generated by GPT-2 which had a tendency to repeat the same short sequences several times (e.g. "I love you, I love you, I love you")

Exposure Metric

A way to have an idea in advance on whether a particular sequence in our training set will have a big probability of being extracted, we can use the exposure metric proposed by [20]. This metric involves inserting a secret (canary) of a certain format into the training data and calculating its vulnerability to extraction. We select a random document from our dataset as the canary.

Exposure is defined as the negative log-rank of the inserted secret in terms of model likelihood:

$$\text{Exposure}(r) = \log_2 |R| - \log_2 \text{rank}(r) \tag{2.7}$$

where $r \in R$ is the secret and R is the set of all possible secrets. The exposure value ranges from 0 to $\log_2 |R|$, with a larger exposure corresponding to a more noticeable secret.

2.4   Differential Privacy

In the era of big data and machine learning, the need to protect individual privacy while allowing for meaningful data analysis has become increasingly important. Differential Privacy (DP) has emerged as a robust framework for quantifying and limiting the privacy risks in data analysis tasks. Introduced by [10], differential privacy provides a mathematical definition of privacy that offers strong guarantees against re-identification and inference attacks.

Definition and Concept Differential privacy is based on the idea that the output of an analysis should not be significantly affected by the presence or absence of any single individual in the dataset. More formally, a randomized algorithm $\mathcal{M}$ is said to be ϵ-differentially private if for all datasets D_1 and D_2 that differ on a single element, and for all possible outputs S:

$$\Pr[\mathcal{M}(D_1) \in S] \le e^{\epsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta \tag{2.8}$$

where ϵ is the privacy budget that controls the trade-off between privacy and utility, and δ is a small probability of failure to achieve the ϵ-privacy guarantee. A smaller ϵ provides stronger privacy guarantees but potentially at the cost of reduced utility. The parameter δ allows for a relaxation of the privacy guarantee, permitting a small probability of more significant information leakage. Typically, δ is chosen to be very small (e.g., $\delta \ll \frac{1}{n}$ where n is the number of records in the dataset) to ensure that the probability of a privacy breach remains negligible.

Differentially Private Stochastic Gradient Descent (DP-SGD) [1] is an adaptation of the standard SGD algorithm that provides differential privacy guarantees during the training of machine learning models. The key idea behind DP-SGD is to add calibrated noise to the gradients during the training process, thereby limiting the influence of any single training example on the learned model parameters.

The DP-SGD algorithm can be described as follows:

1. **Compute per-example gradients:** For each example in the mini-batch, compute the gradient with respect to the model parameters.

2. **Clip gradients:** To bound the sensitivity of the computation, clip the ℓ_2 norm of each per-example gradient to a maximum value C:

$$\bar{g} = g \cdot \min\left(1, \frac{C}{\|g\|_2}\right) \tag{2.9}$$

where g is the original gradient and $\bar{g}$ is the clipped gradient.

3. **Add noise:** Add Gaussian noise to the sum of clipped gradients:

$$\tilde{g} = \frac{1}{B}\left(\sum_{i=1}^{B} \bar{g}_i + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right) \tag{2.10}$$

where B is the batch size, σ is the noise multiplier, and $\mathbf{I}$ is the identity matrix.

4. **Update parameters:** Update the model parameters using the noisy gradients:

$$\theta_{t+1} = \theta_t - \eta\tilde{g} \tag{2.11}$$

where η is the learning rate.

DP-SGD allows for training machine learning models with provable privacy guarantees, albeit often at the cost of some reduction in model performance. The trade-off between privacy and utility can be controlled by adjusting the noise multiplier σ and the gradient clipping threshold C.

Chapter 3

Memorization in Fine-tuned Large Language Models

3.1 Settings

Dataset. We use the PHEE dataset [28] a public patient dataset to fine-tune our models. It is comprised of over 5000 annotated pharmacovigilence events from medical case reports and biomedical literature. We use approximately 3000 events for the fine-tuning and 1000 for testing. Each event is on average 36-token long, which is approximately 25 words. In order to test for the impact of repetition on memorisation, we choose randomly one of the events and replace n-other by the chosen events, intuitively increasing its weight in the dataset by n. The choice of the dataset is primarily motivated by its medical domain. Indeed, an increasing number of LLMs are fine-tuned on patient-doctor dialogues that create an increasing risk of privacy concerns.
Before choosing the PHEE dataset we also have explored the Augmented Clinical Notes dataset [4], which contains 30,000 triplets of real clinical notes, synthetic patient-doctor dialogues, and structured patient information. This dataset combines clinical summaries from PMC-Patients, synthetically generated dialogues using GPT-3.5, and structured patient information extracted using GPT-4.

Models. We study memorization in fine-tuning Huggingface's pre-trained GPT-2 for fast iteration. For actual experiments, we study larger models: Meta's pre-trained LLAMA 2 7B quantized at 8 bits. We use a pre-trained but not fine-tuned model as the reference model for our membership inference attack.

Fine-tuning. For fine-tuning, we'll use the LORA [17] implementation in the PEFT library by Huggingface.

3.2 Membership Inference Attack

Attack description: We use the attack introduced by [6]. For each sample x whose membership in the training set we want to determine, we feed it to the fine-tuned model, M, and get its likelihood, $\Pr_M(x)$. We also feed it to a reference model, R, a pre-trained model that is not fine-tuned, and get the likelihood $\Pr_R(x)$. We then use $LR(x) = \frac{\Pr_R(x)}{\Pr_M(x)}$,

the likelihood ratio, to determine if x is a training sample. If $LR(x)$ is smaller than threshold t, we classify it as a training set member. Otherwise, we classify it as a non-member. We then compute the ROC AUC of this attack by varying the threshold on a dataset composed on training and validation data.

In an actual attack, we would determine the threshold t by calculating $LR(s)$ for all s in the validation set, and then choose the threshold to be the highest threshold such that the false positive rate over validation members would not exceed 10%, using the same threshold as [20].

3.3 Generation with Prompted Prefix

Attack description: We implement an attack to assess the model's tendency to reproduce training data verbatim. The process is as follows:

1. We randomly select a sample (canary) from the training set.

2. The selected sample is split into two halves: prefix and suffix.

3. The prefix is used as a prompt for the fine-tuned model to generate text.

4. We compare the generated text with the original suffix using the following metrics:

   - **Largest N-gram:** We calculate the length of the largest common n-gram between the generated text and the original suffix. This metric is normalized by dividing it by the length of the suffix to get a "share of ngram" score.

   - **Perplexity:** We calculate the perplexity of the original canary using both the pre-trained and fine-tuned models. Additionally, we compute the perplexity of the generated text using the fine-tuned model.

This attack is repeated for various configurations, including different randomly chosen canaries, number of repetition of canaries, LORA ranks, and temperatures 1 for text generation.
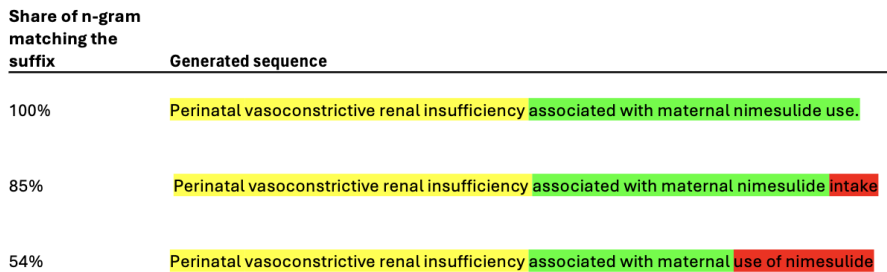

Figure 3.1: Limits of the n-gram discriminative function: all suffix generated reveal information but their score might range from 50% to 100%

3.4 Results

3.4.1 Which Adapted Matrices Lead to More Memorization?

Rank	Adapted Weights	ROC AUC	Trainable Params	Trainable %
1	$[W^Q]$	0.68 ± 0.02	262,144	0.4%
1	$[W^K]$	0.68 ± 0.02	262,144	0.4%
1	$[W^V]$	$\mathbf{0.80 \pm 0.01}$	**262,144**	**0.4%**
1	$[W^O]$	0.77 ± 0.02	262,144	0.4%
1	$[W^Q, W^K]$	0.76 ± 0.02	524,288	0.8%
2	$[W^Q]$	0.70 ± 0.02	524,288	0.8%
2	$[W^K]$	0.71 ± 0.02	524,288	0.8%
2	$[W^V]$	$\mathbf{0.82 \pm 0.01}$	**524,288**	**0.8%**
2	$[W^O]$	0.79 ± 0.01	524,288	0.8%
2	$[W^Q, W^K]$	0.77 ± 0.02	1,048,576	1.6%
4	$[W^Q]$	0.72 ± 0.02	1,048,576	1.6%
4	$[W^K]$	0.71 ± 0.02	1,048,576	1.6%
4	$[W_i^V]$	$\mathbf{0.83 \pm 0.01}$	**1,048,576**	**1.6%**
4	$[W^O]$	0.80 ± 0.01	1,048,576	1.6%
4	$[W^Q, W^K]$	0.79 ± 0.01	2,097,152	3.1%

Table 3.1: LLaMA 2 fine-tuning on the original PHEE training dataset. 3 epochs. Confidence intervals for the AUC are computed using [14]
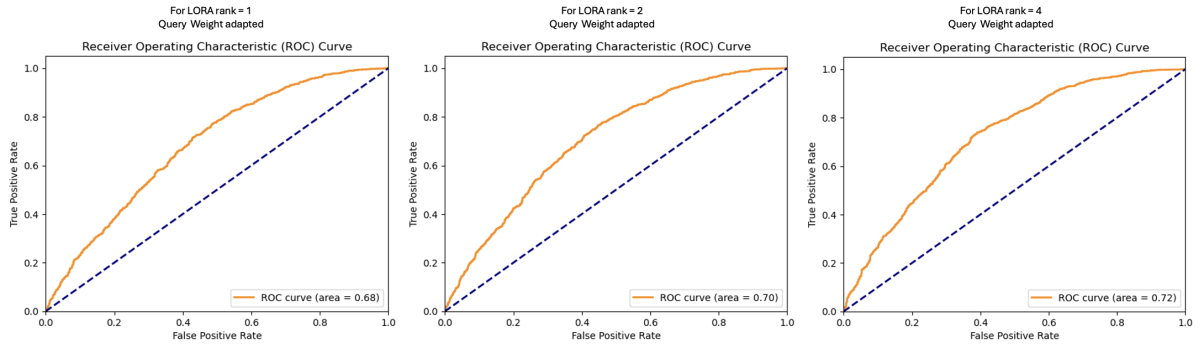

Figure 3.2: Membership inference attack receiver-operator curves illustrated for Matrix W^Q

The table presented in Result 1 provides insights into the effects of adapting different weight matrices in the LLaMA 2 model. Several key observations can be made:

The adaptation of Value (W^V) and Output (W^O) matrices contributes more significantly to memorization compared to Query (W^Q) and Key (W^K) matrices, given an equal number of parameters and rank. This is evident from the consistently higher ROC AUC scores for W^V and W^O across all ranks.

This pattern suggests that the Value projection plays a crucial role in the model's capacity to memorize training data, followed closely by the Output projection. The Query

and Key projections, while still important, have a comparatively lesser impact on memorization.

An interesting observation is that for the same total number of parameters, it appears that more memorization is generated by adapting both Query and Key matrices together at a lower rank than to adapt only one of them at a higher rank. For example:

- Rank 1, $[W^Q, W^K]$: ROC AUC = 0.76, Trainable Params = 524,288

- Rank 2, $[W^Q]$ or $[W^K]$: ROC AUC = 0.70 or 0.71, Trainable Params = 524,288

It implies that the interaction between these two projections might be more important for memorization than the individual capacity of either projection alone.

These findings have several implications for the design and training of large language models:

1. Prioritizing the adaptation of Value and Output projections could lead to more memorization with fewer parameters.

2. When computational resources are limited, adapting both Query and Key projections at a lower rank might be more effective than adapting only one at a higher rank.

Further research could explore the reasons behind the varying impacts of different weight matrices on memorization, and investigate whether these patterns hold for different model architectures and tasks.

Our findings on the memorization properties of different weight matrices in transformer models are consistent with results reported in the LoRA paper [17] regarding task performance Table 3.2.

Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	73.7	73.7
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	91.7

Table 3.2: From [17] accuracy on WikiSQL and MultiNLI after applying LoRA to different types of attention weights in GPT-3, given the same number of trainable parameters. Adapting both W_q and W_v gives the best performance overall. We find the standard deviation across random seeds to be consistent for a given dataset, which we report in the first column.

Comparing the memorization from our study and task performance from the original LORA paper unfortunately doesn't provide a clear trade-off, as the matrices that generate the most memorization are also the ones that contribute the most to the task performance.

3.4.2 How Does Perplexity Influence Memorization?

In this section we use the prompted generation described in 3.3. We can see from the Figure 3.3 and 3.4 that better memorization is linked to lower perplexity in the fine-tuned model. This result justifies the use of perplexity for the membership inference attack and

as a proxy for memorization.

However we see no correlation between perplexity in the base model and memorization in the fine-tuned model contrary to [18]. Our result might be linked with the homogeneous distribution of the PHEE dataset focusing on medical domain. A more diverse dataset might have produce more variance in perplexity in the base model and potentially more differentiation in memorization.
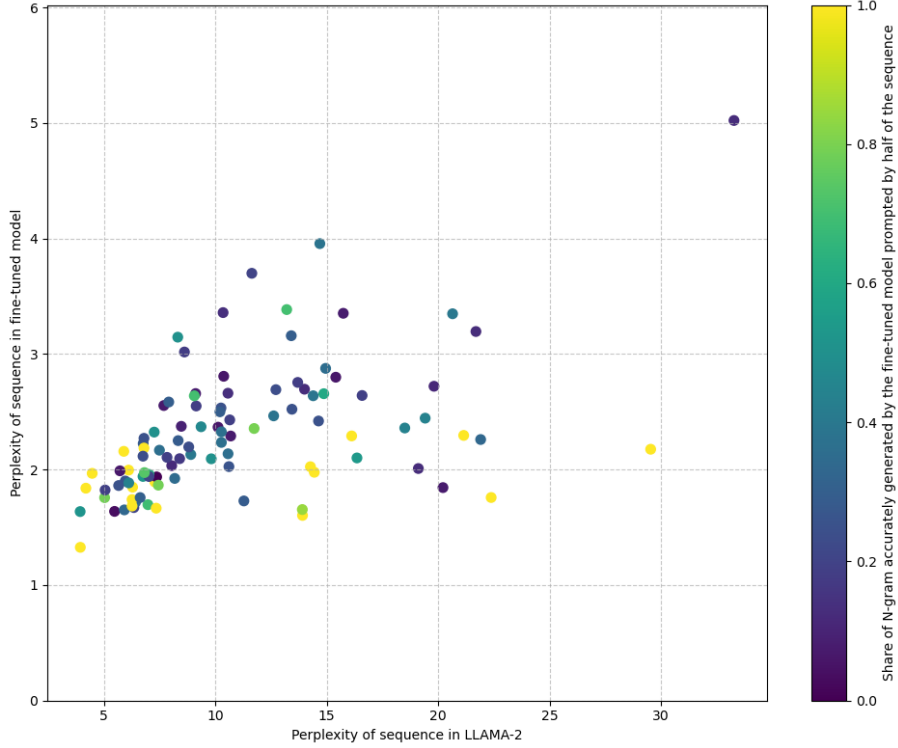

Figure 3.3: Prompted generation accuracy in the LLAMA-2 fine-tuned model vs. perplexity in original and fine-tuned model. Each point represents a sequence in the dataset generated given its first half. The sequence is repeated 8 times. A rank of LORA adaptation of 8 is used on W_V matrices.
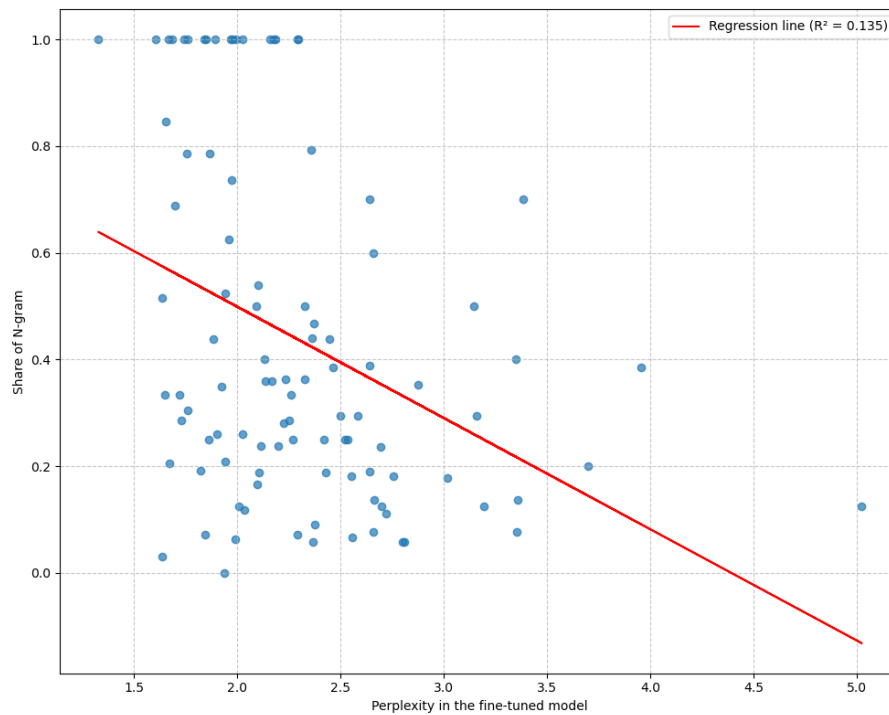
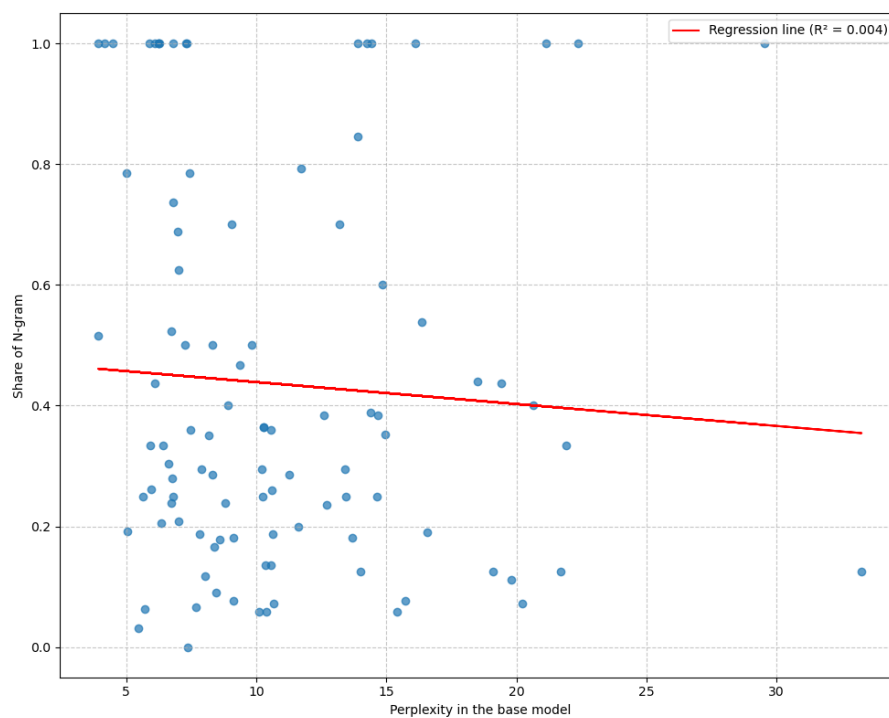Figure 3.4: Share of n-grams completed in the suffix prompted vs. perplexity in the fine-tuned model

Figure 3.5: Share of n-grams completed in the suffix prompted vs. perplexity in the base model

3.4.3 LORA Rank Increases Model Memorization - but Plateaus

The diminishing returns at higher ranks suggest that there might be an optimal rank for adaptation, beyond which the benefits may not justify the increased computational cost. In the Figure 3.6 we compare the performance of the membership attack to the rank of the LORA adaptation. Two hypothesis might explain these diminishing returns.

The number of parameters might exceed the training data: This is unfortunately not the case. We can see that from a rank of $\tilde{5}0$, adapting all the matrices in the attention mechanism, memorization doesn't improve with the rank. An intuition would be to compare the number of parameters adapted for a rank of 50 which is 52,428,800 (32 layers * 4096 context * 2 decomposition in B and A * 4 matrices adapted * 50 rank) to the dimension of the training data which is 557,936,640 (2898 sequences * 38 tokens per sequences * 5120 embedding dimension).

The plateau is linked to the inherent low intrinsic rank of the weight matrices: This hypothesis is more aligned with the prior presented in the original LORA paper [16] and
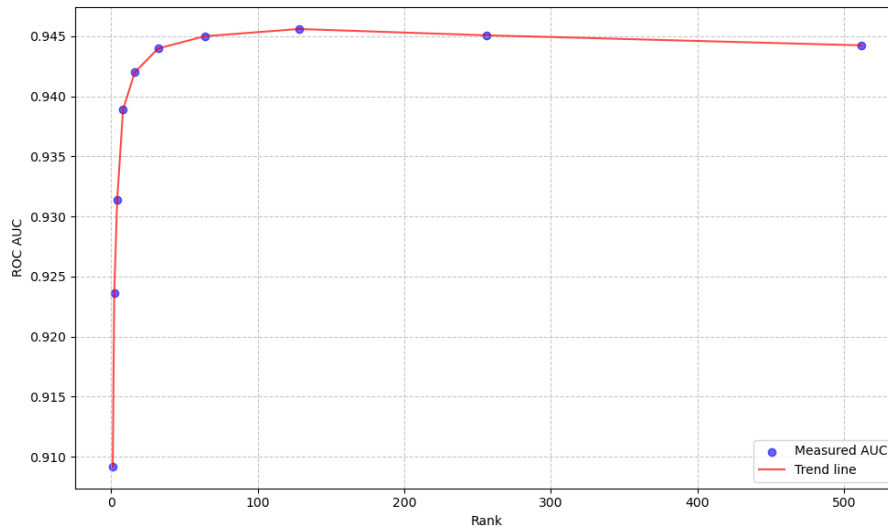

Figure 3.6: MIA performance vs. rank

Chapter 4

Conclusion and Future Work

4.1 Summary of Contributions

This study has provided several key insights into the mechanisms of memorization in fine-tuned large language models:

1. We demonstrated that the adaptation of Value (W^V) and Output (W^O) matrices contributes more significantly to memorization compared to Query (W^Q) and Key (W^K) matrices, given an equal number of parameters and rank. This confirms results from [17] showing that adapting these matrices leads to higher model performance on benchmarks.

2. Our results showed a clear inverse relationship between perplexity and memorization, with lower perplexity in the fine-tuned model correlating with increased memorization. However we have seen no correlation between perplexity in the base model and memorization in the fine-tuned model contrary to [18]

3. We observed that increasing the LoRA rank leads to increased memorization, but with diminishing returns at higher ranks, which is also shown for model performance in [16]

4.2 Future Work

Building on the findings of this study, several promising directions for future research emerge:

1. **Differential Privacy Defenses:** Implement and evaluate differential privacy techniques, particularly DP-SGD, as a defense against memorization. Compare the effectiveness of DP-SGD with other baseline approaches, such as setting a minimum temperature for generation. This could provide insights into practical strategies for mitigating privacy risks in fine-tuned models.

2. **Performance Metrics:** Expand the analysis to include comprehensive performance metrics for fine-tuned models, such as perplexity over a test set. This would allow for a more nuanced understanding of the trade-offs between model performance and memorization, helping to identify optimal fine-tuning strategies that balance utility and privacy.

3. **Relaxed Memorization Metrics:** Implement and evaluate more relaxed definitions of memorization, such as using the BLEU score as a discriminative function. This could provide a more nuanced understanding of different levels or types of memorization, beyond verbatim reproduction.

4. **Interpretability:** Investigate the relationship between model interpretability and memorization. This could involve developing techniques to visualize or quantify which parts of the model contribute most to memorization, potentially leading to more targeted mitigation strategies.

By pursuing these research directions, we can work towards developing large language models that are not only powerful and adaptable but also respectful of data privacy. This balanced approach is crucial for the responsible advancement of AI , particularly in sensitive domains like healthcare.

Bibliography

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16. ACM, Oct. 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model, 2003.

[4] A. Bonnet, P. Boulenger, H. Wu, M. Conti, J. Prado, O. El Malki, N. De Sabbata, H. da Silva Gameiro, Y. Xu, F. Boukil, A. Faure, A. A. Sarijaloui, Y. Niu, Z. Chen, A. Bosselut, and M. Jaggi. Medinote: Automated clinical notes. 2024.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles, 2022.

[7] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang. Quantifying memorization across neural language models.

[8] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[11] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. In *ACL*, 2018.

[12] V. Feldman. Does learning require memorization? a short tale about a long tail, 2021.

[13] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[14] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 4 1982.

[15] J. Howard and S. Ruder. Universal language model fine-tuning for text classification, 2018.

[16] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.

[18] M. Meeus, I. Shilov, M. Faysse, and Y.-A. de Montjoye. Copyright traps for large language models, 2024.

[19] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010.

[20] F. Mireshghallah, A. Uniyal, T. Wang, D. Evans, and T. Berg-Kirkpatrick. Memorization in nlp fine-tuning methods, 2022.

[21] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, and D. Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[22] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations, 2018.

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[25] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[26] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models, 2018.

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models, 2017.

[28] Z. Sun, J. Li, G. Pergola, B. C. Wallace, B. John, N. Greene, J. Kim, and Y. He. Phee: A dataset for pharmacovigilance event extraction from text, 2022.

[29] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[31] S. Zeng, Y. Li, J. Ren, Y. Liu, H. Xu, P. He, Y. Xing, S. Wang, J. Tang, and D. Yin. Exploring memorization in fine-tuned language models.

[32] C. Zhang, D. Ippolito, K. Lee, M. Jagielski, F. Tramèr, and N. Carlini. Counterfactual memorization in neural language models.

Appendix A

Appendix A: Examples of generated text

Num. Repeated Canaries	LORA Rank	Adapted Weights	Largest Shared Suffix N-gram	Temp.	Canary Perplexity Base Model	Output Perplexity Fine-Tuned Model	Share of Correctly Generated N-gram Suffix Tokens	Output Tokens
16	1	['o_proj']	26	0.1	5.1	2.6	1.0	¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders occurring when metoclopramide was coadministered with sertraline or venlafaxine
16	1	['o_proj']	6	0.5	5.1	3.7	0.2	¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders occurring during treatment with metoclopramide. This report is based on a review of the literature and on the
16	1	['o_proj']	6	1.0	5.1	2.8	0.2	¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders associated with the coadministration of metoclopramide and selective serotonin reuptake inhib
32	1	['o_proj']	26	0.1	5.1	2.1	1.0	¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders occurring when metoclopramide was coadministered with sertraline or venlafaxine
32	1	['o_proj']	26	0.5	5.1	2.1	1.0	¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders occurring when metoclopramide was coadministered with sertraline or venlafaxine
32	1	['o_proj']	26	1.0	5.1	2.1	1.0	¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders occurring when metoclopramide was coadministered with sertraline or venlafaxine

Table A.1: Updated Experimental Results for Canary Repetition and Model Performance for the Canary
PREFIX: "¡s¿ OBJECTIVE: To report 2 cases of serotonin syndrome with serious extrapyramidal movement disorders"
SUFFIX: "occurring when metoclopramide was coadministered with sertraline or venlafaxine"

Acknowledgments