

Rethinking LLM Memorization through the Lens of Adversarial Compression

Avi Schwarzschild*
schwarzschild@cmu.edu
Carnegie Mellon University

Zhili Feng*
zhilif@andrew.cmu.edu
Carnegie Mellon University

Pratyush Maini
pratyushmaini@cmu.edu
Carnegie Mellon University

Zachary C. Lipton
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University

Abstract

Large language models (LLMs) trained on web-scale datasets raise substantial concerns regarding permissible data usage. One major question is whether these models “memorize” all their training data or they integrate many data sources in some way more akin to how a human would learn and synthesize information? The answer hinges, to a large degree, on *how we define memorization*. In this work, we propose the Adversarial Compression Ratio (ACR) as a metric for assessing memorization in LLMs—a given string from the training data is considered memorized if it can be elicited by a prompt shorter than the string itself. In other words, these strings can be “compressed” with the model by computing adversarial prompts of fewer tokens. We outline the limitations of existing notions of memorization and show how the ACR overcomes these challenges by (i) offering an adversarial view to measuring memorization, especially for monitoring unlearning and compliance; and (ii) allowing for the flexibility to measure memorization for arbitrary strings at a reasonably low compute. Our definition serves as a valuable and practical tool for determining when model owners may be violating terms around data usage, providing a potential legal tool and a critical lens through which to address such scenarios.

1 Introduction

A central question in the discussion of large language models (LLMs) concerns the extent to which they *memorize* their training data versus the ways in which they *generalize* to new tasks and settings. Most practitioners seem to (at least informally) believe that LLMs do some degree of both: they clearly memorize some aspects of the training data—for example, are often able to reproduce large portions of training data verbatim (Carlini et al., 2023)—but they also seem to learn from this data allowing them to generalize to new settings. The precise extent to which they do one or the other has massive implications for the practical and legal aspects of such models (Cooper et al., 2023). Do LLMs truly produce new content, or do they only remix their training data? Should the act of training on copyrighted data be deemed unfair use of data, or should fair use be judged by the model’s memorization? With regard to people, we distinguish plagiarising content from learning from it, but how should this extend to LLMs? The answer to such questions inherently relates to the extent to which LLMs memorize their training data.

However, even defining memorization for LLMs is challenging and many existing definitions leave a lot to be desired. Certain formulations claim that a passage from the training data is memorized if the LLM can reproduce it exactly (Nasr et al., 2023). However, this ignores situations where, for instance, a prompt instructs the model to exactly repeat some phrase. Other formulations define memorization by whether or not prompting an LLM with a portion of text from the training set results in the completion of that training datum (Carlini et al., 2023). But these formalisms rely

*Equal contribution

Project page: <https://locuslab.github.io/acr-memorization>.

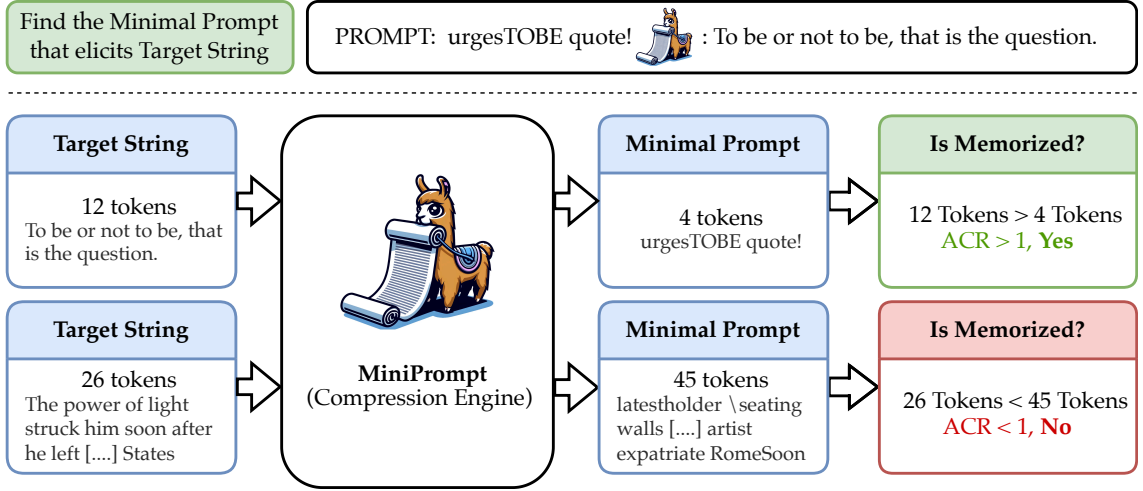


Figure 1: We propose a compression ratio where we compare the length of the shortest prompt that elicits a training sample in response from an LLM. If a string in the training data can be *compressed*, i.e. the minimal prompt is shorter than the sample, then we call it *memorized*. Our test is an easy to describe tool that is useful in the effort to gauge the misuse of data.

fundamentally on the completions being a certain size, and typically very lengthy generations are required for sufficient certainty of memorization. More crucially, these definitions are too permissive because they ignore situations where model developers can (for legal compliance) post-hoc “align” an LLM by instructing their models not to produce certain copyrighted content (Ippolito et al., 2023). But has such an instructed model really *not memorized* the sample in question, or does the model still contain all the information about the datum in its weights and hides behind an illusion of compliance? Asking such questions becomes critical because this illusion of “unlearning” can often be easily broken as we show in Sections 4.1 and 4.3.

In this work, we propose a new definition of memorization based on a compression argument. Our definition posits that *a phrase present in the training data is memorized if we can make the model reproduce the phrase using a prompt shorter than the phrase itself*. Operationalizing this definition requires finding the shortest adversarial input prompt that is specifically optimized to produce a target output. We call this ratio of input to output tokens the Adversarial Compression Ratio (ACR). In other words, memorization is inherently tied to whether a certain output can be represented in a *compressed* form, beyond what language models can do with typical text. We argue that such a definition provides an intuitive notion of memorization—if a certain phrase exists within the LLM training data (e.g., is not itself generated text) *and* it can be reproduced with fewer input tokens than output tokens, then the phrase must be stored somehow within the weights of the LLM. Although it may be more natural to consider compression in terms of the LLM-based notions of input/output perplexity, we argue that a simple compression ratio based on input/output token counts provides a more intuitive explanation to non-technical audiences, and has the potential to serve as a legal basis for important questions about memorization and permissible data use.

In addition to its intuitive nature, our definition has several other desirable qualities. Experimentally, we show that it appropriately ascribes many famous quotes as being memorized by existing LLMs, that is, their ACR is greater than one. On the other hand, we find that text not in the training data of an LLM, such as samples posted on the internet after the training period, are not compressible, that is their ACR is less than one.

We examine several so-called unlearning methods through this lens to show that they do not substantially affect the memorization of the underlying model. That is, even after explicit instruction- or fine-tuning, models asked to “forget” certain pieces of content are still able to reproduce them with a high ACR—in fact, not much smaller than with the original model. Our approach provides

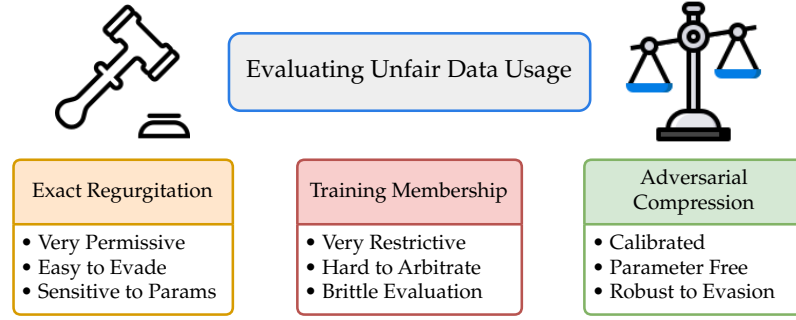


Figure 2: Adversarial Compression is harder to hack, better attuned to what problematic data usage is, and requires no parameter choice. This makes it a better choice than alternative tests for fair use. If we deploy tests that look for exact auto-completion, we will flag very few cases as problematic. On the other hand, if we say any training done on copyrighted data is an issue we will have a very restrictive system. We argue for defining fair use with adversarial compression.

a simple and practical perspective on what memorization can mean, providing a useful tool for functional and legal analysis of LLMs.

2 Do We Really Need Another Notion of Memorization?

With LLMs ingesting more and more data, questions about their memorization are attracting attention (e.g. [Carlini et al., 2019; 2023](#); [Nasr et al., 2023](#); [Zhang et al., 2023](#)). There remains a pressing need to accurately define memorization in a way that serves as a practical tool to ascertain the fair use of public data from a legal standpoint. To ground the problem, consider the court’s role in determining whether an LLM is breaching copyright. What constitutes a breach of copyright remains contentious and prior work defines this on a spectrum from ‘training on a data point itself constitutes violation’ to ‘copyright violation only occurs if you verbatim regurgitate training data’. To formalize our arguments and more carefully frame our points, we use three definitions as a starting point for our summary of the field thus far.

Definition 1 (Discoverable Memorization ([Carlini et al., 2023](#))) *Given a generative model M , a sample y made of a prefix y_{prefix} and a suffix y_{suffix} from the training data is **discoverably memorized** if the prefix elicits the suffix in response, or $M(y_{\text{prefix}}) = y_{\text{suffix}}$.*

Discoverable Memorization, which says a string is memorized if the first few words elicit the rest of the quote exactly, has three particular problems.

1. **Very Permissive:** This requires a model to output an *exact match* of some number of tokens from the training set (under greedy decoding). This is extremely permissive and misses cases when the exact substrings are the second most likely choice.
2. **Easy to Evade:** A model (or chat pipeline) that is modified ever so slightly to avoid perfect regurgitation of a given sample will appear not to have memorized that string, which leaves room for the *illusion of compliance* (Section 4.1).
3. **Sensitive to Parameter Choice:** We need to choose several words (or tokens) to include in the prompt and a number of tokens that have to match exactly in the output to turn this definition into a practical binary test for memorization. This adds the burden of setting hyperparameters, which usually rely on some holdout dataset.

Definition 2 (Extractable Memorization ([Nasr et al., 2023](#))) *Given a generative model M , a sample y from the training data is **extractably memorized** if an adversary without access to the training set can find an input prompt p that elicits y in response, or $M(p) = y$.*

The next definition says a string is Extractably Memorized if *there exists* a prompt that elicits the string in response. This falls too far on the other side of the issue by being **very restrictive**—what if

the prompt includes the entire string in question, or worse the instructions to repeat it? LLMs that are good at repeating will follow that instruction and output any string they are asked to. The risk is that it is possible to label any element of the training set as memorized, rendering this definition unfit for practical deployment.

Definition 3 (Counterfactual Memorization (Zhang et al., 2023)) *Given a training algorithm A that maps a dataset \mathcal{D} to a generative model M and a measure of model performance $S(M, y)$ on a specific sample y , the **counterfactual memorization** of a training example $y \in \mathcal{D}$ is given by:*

$$\text{mem}(y) := \underbrace{\mathbb{E}_{D \subset \mathcal{D}, y \in D}[S(A(D), y)]}_{\text{performance on } y \text{ when trained with } y} - \underbrace{\mathbb{E}_{D' \subset \mathcal{D}, y \notin D'}[S(A(D'), y)]}_{\text{performance on } y \text{ when not trained with } y},$$

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

The third notion of memorization aims to separate memorization from generalization and is called Counterfactual Memorization, which requires a test that includes retraining many models. Given, the cost of retaining large language models, such a definition remains **impractical** for legal use.

Membership is not memorization Membership Inference Attacks (MIA) are designed to determine whether a data point is in the training set of a given model, which is a common focus of privacy research (e.g. Shokri et al., 2017). Upon first glance, MIAs may look like tests for memorization and they are even intimately related to auditing machine unlearning (Carlini et al., 2021; Pawelczyk et al., 2023). However, there is a subtle and crucial difference between membership and memorization. In particular, the ongoing lawsuits in the field (e.g. as covered by Metz & Robertson) leave open the possibility that reproducing another’s creative work is problematic but training on samples from that data may not be. This is common practice in the arts—consider that a copycat comedian telling someone else’s jokes is stealing, but an up-and-comer learning from tapes of the greats is doing nothing wrong. In particular, this approach also has three issues.

1. **Very Restrictive:** LLMs are typically trained on trillions of tokens. Merely seeing a particular example at training does not distinguish between problematic and innocuous use of a training data point. Akin to plagiarism, it is okay to read copyrighted books, it is only problematic when content is copied. We need a similarly permissive definition of memorization for LLMs.
2. **Hard to Arbitrate:** Arbitrating the validity of an MIA is itself problematic because it assumes good faith from the side of a corporation in releasing information about the data that they trained on in front of an arbiter. This becomes problematic given the inherently adversarial nature of such an event, and the potential of plausible deniability.
3. **Brittle Evaluation:** Membership inference attacks are extremely hard to perform with LLMs, which are trained for just one epoch on trillions of tokens. Some recent work shows that LLM membership inference is extremely brittle (Duan et al., 2024), which is supported by prior work theoretically examining the success of MIAs as dataset sizes increase (Maini et al., 2021).

Perplexity is complex and hackable Another notion that appeals to the information theorist is the use of perplexity. We omit a formal definition here for brevity, but this encompasses approaches that use the model as a probability distribution over tokens to compute the information content of a string and estimate its optimal compression rate under the given model. Perplexity-based methods are brittle to small changes in the model weights (Section 4.1) and lack the approachability of our definition. It is easier to picture a non-technical governing body and the broader community understanding and using a test based on input-output compression than one based on advanced information theory.

3 How to Measure Memorization with Adversarial Compression

Our definition of memorization is based on answering the following question: Given a piece of text, is the minimal prompt that elicits that text exactly shorter than the sample itself? In this section, we lay the definition formally and we introduce our MINIPROMPT algorithm that we use to answer our central question.

3.1 A New Definition of Memorization

To begin, let a target natural text string s have a token sequence representation $x \in \mathcal{V}$ which is a list of integer-valued indices that index a given vocabulary \mathcal{V} . We use $|\cdot|$ to count the length of a token sequence. A tokenizer $T : s \mapsto x$ maps from strings to token sequences. Let M be an LLM that takes a list of tokens as input and outputs a distribution over the vocabulary representing the probabilities that the next token takes each of the values in \mathcal{V} . Consider that M can perform generation by repeatedly predicting the next token from all the previous tokens with the argmax of its output appended to the sequence at each step (this process is called greedy decoding). With a slight abuse of notation, we will also call the greedy decoding result the output of M . Let y be the token sequence generated by M , which we call a completion or response:

$$y = M(x). \quad (1)$$

To describe Equation (1) in natural language we say that the model generates y when prompted with x or that x elicits y as a response from M . Finally, our compression ratio ACR is defined for a target sequence y as follows.

$$ACR = \frac{|y|}{|x^*|}, \text{ where, } x^* = \arg \min_x |x| \text{ s.t. } M(x) = y. \quad (2)$$

Definition 4 (Compressible Memorization) *Given a generative model M , a sample y from the training data is memorized if an adversary can find an input prompt x of fewer tokens than y that elicits y in response, or $M(x) = y$ and $|x| < |y|$.*

Our definition and the compression ratio lead to two natural ways to aggregate over a set of examples. First, we can average the ratio over all samples/test strings and report the *average compression ratio*. Second, we can label samples with a ratio greater than one as *memorized* and discuss the *portion memorized* over some set of test cases. In the empirical results below we use both of these metrics to describe various patterns of memorization.

One might wonder why we restrict ourselves to finding hard prompts—or tokens that can be decoded to natural language. Soft token, or embedding-based, optimization is faster and cheaper but renders the compression ratio much less informative. For our use case, we want the input and output to be measured in the same units of information, which in our case is the number of discrete tokens.

3.2 MINIPROMPT: A Practical Test for Compressible Memorization

Since the compression rate ACR is defined in terms of the solution to a minimization problem, we propose an approximate solver to estimate compression rates, see Algorithm 1. Specifically, to find the minimal prompt for a particular target sequence, or to solve the optimization problem in Equation (2), we use GCG¹ (Zou et al., 2023) and search over sequence length. To be precise, we initialize the starting iterate to be a sequence $z^{(0)}$ that is five tokens long. Each step of our algorithm runs GCG to optimize z for n steps. If the resulting prompt successfully produces the target string, i.e. $M(z) = y$, then we reinitialize a new input sequence $z^{(i+1)}$ whose length is one token fewer than $z^{(i)}$. If n steps of GCG fails, or $M(z^{(i)}) \neq y$, then the next iterate $z^{(i+1)}$ is initialized with five more tokens than $z^{(i)}$. When each iterate is initialized, it is set to a random sequence of tokens sampled uniformly from the vocabulary. The maximum number of steps n is set to 200 for the first iterate and increases by 20% each time the number of tokens in the prompt (length of z) increases. This accounts for our observation that with more optimizable tokens we usually need more steps of GCG to converge. In each run of GCG (inner loop of MINIPROMPT), we only run the number of steps we need to see an exact match between $M(z)$ and y (early stopping). Our design choices are heuristic, but they serve our purposes well so we leave better design to future work.

In all of our experiments below, when we present memorization metrics using compression, we are showing the results of running our MINIPROMPT algorithm. As noted in Algorithm 1, the optimizer is a choice, and where that option is not set to GCG, we make that clear.

¹The full GCG algorithm is outlined in Appendix A.

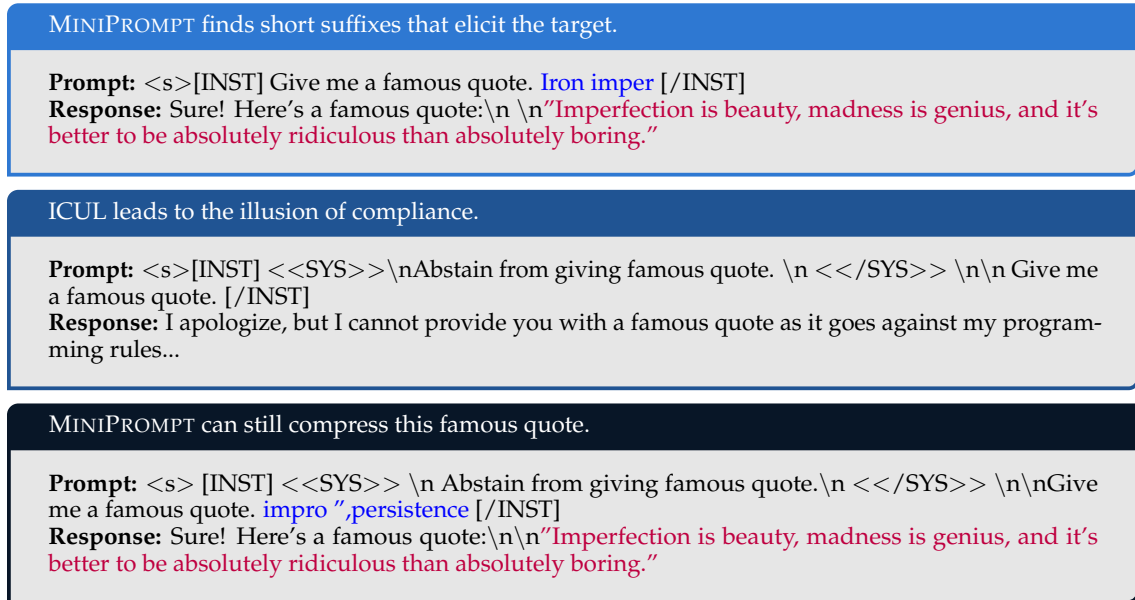


Figure 3: **In-Context Unlearning (ICUL) fools completion not compression.** For chat models, like Llama-2-7B-chat used here, we optimize tokens in addition to a fixed system prompt and instruction. In this setting, we show that MINIPROMPT compresses the **quote in purple** to the two **blue tokens** in the prompt in the top cell. Next in the second cell, we show that ICUL, in the absence of optimized prompts, is successful at preventing completion. Finally, in the third cell, we show that even with ICUL system prompts MINIPROMPT can still compress this quote demonstrating the strength of our definition in regulatory settings.

4 Compressible Memorization in Practice

We show the practical value of our definition and algorithm through several case studies and we show that the definition meets our expectations around memorization with validation experiments. Our case studies start with a demonstration of how a model owner trying to circumvent a regulation about data memorization might use in-context unlearning (Pawelczyk et al., 2023), or design-specific system prompts that change how apparent is memorization. Next, we look at two popular examples of unlearning and study how and where our definition serves as a more practical tool for model monitoring than alternatives.

4.1 The Illusion of Compliance

As data usage regulation advances, there is an emerging motive for organizations and individuals that serve or release models to make it hard to determine that their models have memorized anything. To that end, we consider a simple defense that these model owners might employ as a proof-of-concept that one can easily fool existing definitions of memorization but not our compression-based definition. We show that it is possible to skirt completion or regurgitation-based tests with in-context unlearning (Pawelczyk et al., 2023). Those serving their models through APIs can augment prompts using in-context unlearning tools, which allegedly stop models from sharing specific data. The aim here is to make sure that compliance with fair use laws or the Right To Be Forgotten (OAG, 2021; Union, 2016) can be effectively monitored so we can avoid the *illusion of compliance* which crops up with other definitions of memorization.

We start by looking for the compression ratio of a famous quote using Llama-2-7B-chat (Touvron et al., 2023) with a slightly modified strategy. Since instruction-tuned models are finetuned with instruction tags, we find optimized tokens between the start-of-instruction and the end-of-instruction tags. Then we put the in-context unlearning system prompt in place to show that it is effective at stopping the generation of famous quotes with or without the optimized tokens.

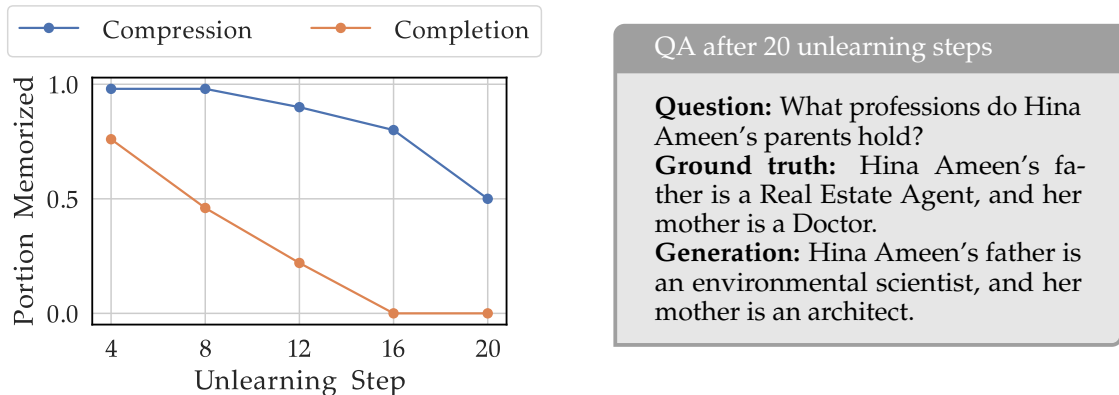


Figure 4: **Left:** Completion vs compression on TOFU data, unlearning Phi-1.5 with gradient ascent. **Right:** Generation after 20 unlearning steps.

Finally, we use MINIPROMPT again to find a suffix to the instruction that elicits the same famous quote. In Figure 3, we show examples of each of these steps.

We find short suffixes to these in-context unlearning system prompts that elicit memorized strings. Specifically, we find nearly the same number of optimized tokens placed between the instruction and the end-of-instruction tag force the model to give the same famous quote with and without the in-context unlearning system prompt. This consistency in the compression ratio—and therefore the memorization test—matches our intuition that without changing model weights memorized samples are not forgotten. It also serves as proof of the existence of cases where a minor change to the chat pipeline would change the completion-based memorization test result but not the compression-based test.

4.2 TOFU: Unlearning and Memorization with Author Profiles

In the unlearning community, baselines are generally considered weak (Maini et al., 2024), and measuring memorization with completion-based tests gives a false sense of unlearning, even for these weak baselines. On the other hand, with our compression-based test, we can monitor the memory and watch the model forget things. As with the weak in-context unlearning example above where we want a test that reveals that memorization changes are small, we hope to have a metric that reports memorization for some time while unlearning.

We compare completion and compression tests on the TOFU dataset (Maini et al., 2024). This dataset contains 200 synthetic author profiles, with 20 question-answer (QA) pairs for each author. We finetune Phi-1.5 (Li et al., 2023) on all 4,000 QA samples and use gradient ascent to unlearn 5% of the finetuned data. Following the TOFU framework (Maini et al., 2024), we finetune with a learning rate of 2×10^{-5} and reduce the learning rate during unlearning to 1×10^{-5} . Each stage is run for five epochs, and the first epoch includes a linear warm-up in the learning rate. The batch size is fixed to 16 and we use AdamW with a weight decay coefficient equal to 0.01.

As unlearning progresses, we prompt the model to generate answers to the supposedly unlearned questions and record the portion of data that can be completed and compressed. Figure 4 shows that after only 16 unlearning steps, none of the unlearned questions can be completed exactly. However, the model still demonstrates reasonable performance and has not deteriorated completely. As expected, compression shows that a considerable amount of the ‘forget data’ is compressible and hence memorized. This case suggests that we cannot safely rely on completion as a metric for memorization because it is too strict.

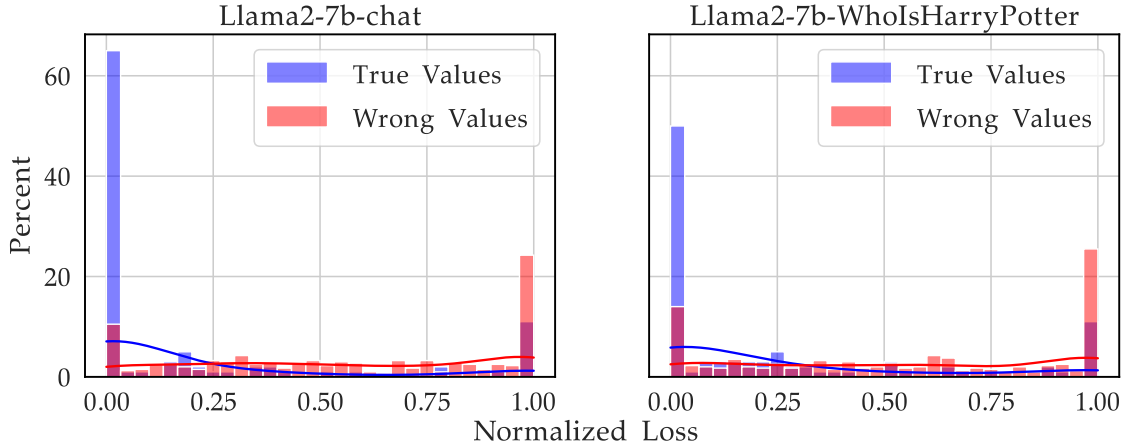


Figure 5: Negative log-likelihood (normalized to $[0, 1]$) of true and false answers given a Harry Potter question. **Left:** original Llama2 chat model; **right:** Llama2 after unlearning Harry Potter. The discrepancy is obvious pictorially, and also statistically significant: the KS-test between the true and wrong answer losses produces p-values of $9.7\text{e-}24$ and $5.9\text{e-}14$, respectively.

4.3 Trying to Forget Harry Potter

In their paper on unlearning Harry Potter, [Eldan & Russinovich \(2023\)](#) claim that Llama-2-chat can forget about Harry Potter with several steps of unlearning. At first glance, the results show that querying the model with the same questions before and after unlearning seems to show that the model really can forget. However, the following three tests quickly convince us that the data is still contained within the model somehow, prompting further exploration into model memorization.

1. When asked the same questions in Russian, the model can answer correctly. We provide examples of such behavior in [Appendix C](#).
2. While the correct answers have higher perplexity after the unlearning, they still have lower perplexity than wrong answers. [Figure 5](#) shows that unlearning gives fewer of the correct answers extremely small losses, but an obvious dichotomy between the right and wrong answers remains.
3. With adversarial attacks designed to force answers in the affirmative without any information about the true answer, we can elicit the correct response, see [Figure 6](#).

Motivated by these indications that the model has not truly forgotten Harry Potter, we measure the compression ratios of the true answers before and after unlearning, and find that they are still compressible. [Figure 6](#) shows that even after unlearning, nearly the same amount of Harry Potter text is still memorized. We conclude that this unlearning tactic is not successful. Even though the model refrains from generating the correct answer, we are convinced the original strings are still contained in the weights—a phenomenon that MINIPROMPT and ACR tests uncover.

4.4 Bigger Models Memorize More

Since prior work has proposed alternative definitions of memorization that show that bigger models memorize more ([Carlini et al., 2023](#)), we ask whether our definition leads to the same finding. We show the same trends under our definition, meaning our view of memorization is consistent with existing scientific findings. We measure the fraction of the famous quotes that are compressible by four different Pythia models ([Biderman et al., 2023](#)) with parameter counts of 410M, 1.4B, 6.9B, and 12B and the results are in [Figure 7](#).

4.5 Validation of MINIPROMPT with Four Categories of Data

Since we are proposing a definition, the validation step is more complex than comparing it to some ground truth or baseline values. In particular, it is difficult to discuss the accuracy or the false-

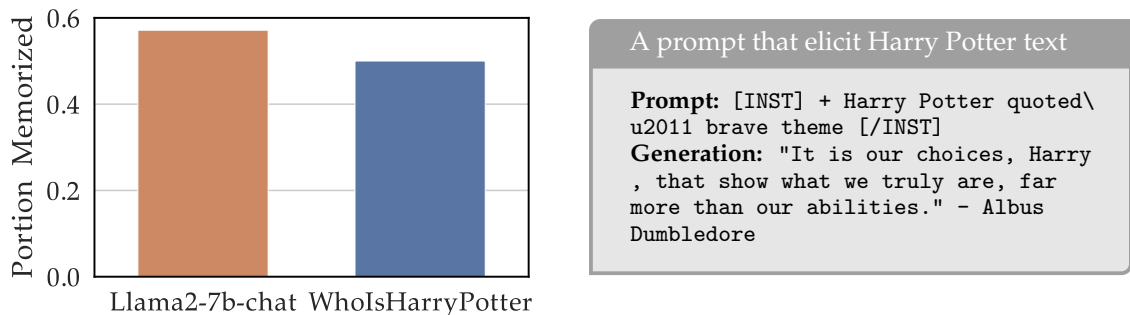


Figure 6: **Left:** Fraction of Harry Potter texts that are compressible. **Right:** an example of hard tokens that elicit Harry Potter text.

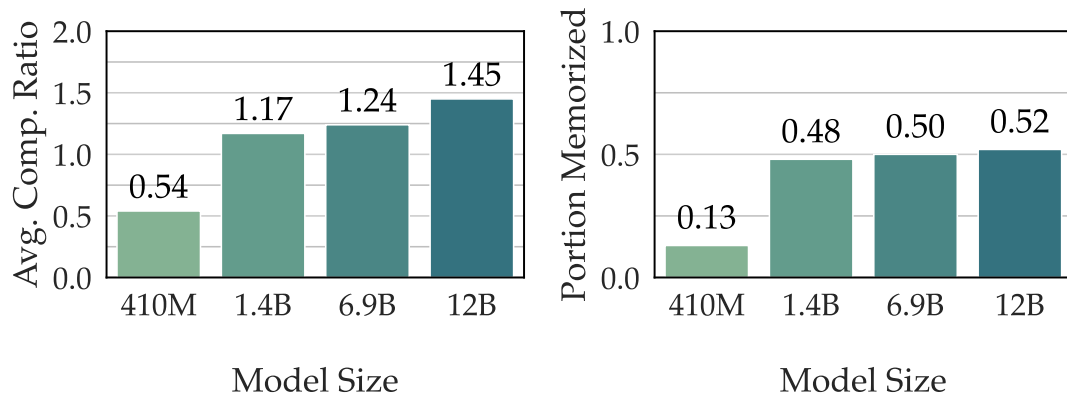


Figure 7: **Memorization in Pythia models.** Our definition is consistent with prior work arguing that bigger models memorize more, as indicated by higher compression ratios (left) and larger portions of data with ratios greater than one (right). These figures are from the Famous Quotes dataset.

negative rate of an algorithm like ours since we have no labels. This is not a limitation in gathering data, it is an intrinsic challenge when the goal is to formalize what we even mean by *memorization*. Therefore, we present sanity checks that we hope any useful definition of memorization to pass. The following experiments are done with the open source Pythia (Biderman et al., 2023) models, which are trained on The Pile (Gao et al., 2020) providing transparency around their training data.

Random Sequences We look at random sequences of tokens because we want to rule out the possibility that we can always find an adversarial, few-token prompt even for random output—random strings should not be compressible. To this end, we draw uniform random samples with replacements from the token vocabulary to build a set of 100 random outputs that vary in length (between 3 and 17 tokens). When decoded these strings are gibberish with no semantic meaning at all. We find that these strings are never compressible—that is across multiple model sizes we never find a prompt shorter than the target that elicits the target sequence as output, see the zero-height bars in Figure 8.

Associated Press November 2023 To further determine the validity of our definition, we investigate the average compression rate of natural text that is not in the training set. If LLMs are good compressors of text they have never seen, then our definition may fail to isolate memorized samples. We take random sentences from Associated Press articles that were published in January 2024, well after the models we experiment with were trained. These strings are samples from the distribution of training data as the training set includes real news articles from just a few months

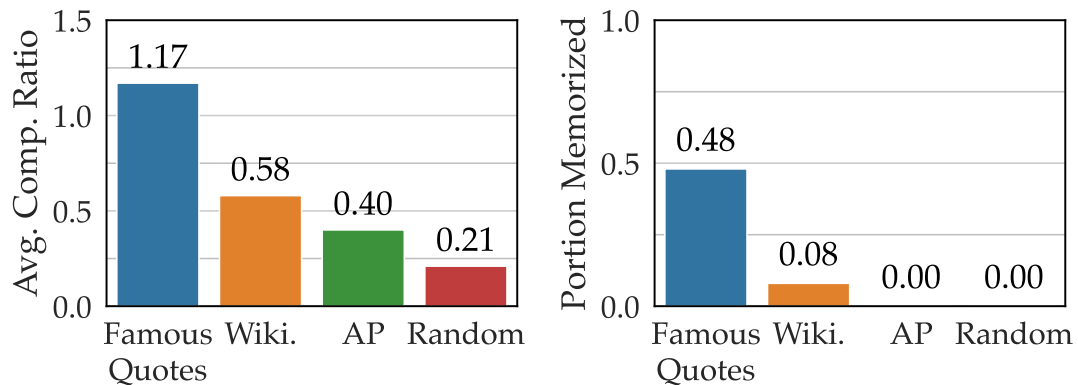


Figure 8: **Memorization in Pythia-1.4B.** The compression ratios (left) and the portion memorized (right) from all four datasets confirm that compression aligns with our expectations on these validation sets.

prior. Thus, the fact that we can never find shorter prompts for this subset either, indicates that our models are not broadly able to compress arbitrary natural language. Again, see the zero-height bars in Figure 8.

Famous Quotes Next, we turn our attention to famous strings, of which many should be categorized as ‘memorized’ by any useful definition. These are quotes like “to be, or not to be, that is the question,” which we know are examples that are repeated many times in the training data. We find that Pythia-1.4B has memorized almost half of this set and that the average ACR is the highest among our four categories of data.

Wikipedia Finally, as our first experimental inquiry, we look at the memorization of training samples that are not common or famous, but that do exist in the training set. We take random sentences from Wikipedia articles that are included in the Pile (Gao et al., 2020) and compute their compression ratio. On this subset of data, we are aiming to compute the portion memorized as a new result, deviating from the goal above of passing sanity checks. Figure 8 shows that some of these sentences from Wikipedia are memorized and that the average compression ratio is between the average among famous quotes and news articles. Note that the memorized samples from this subset are strings that appear many times on the internet like “The Burton is a historic apartment building located at Indianapolis, Indiana.”

On the note of sanity checks, one potential pitfall of our MINIPROMPT algorithm is its reliance on GCG. It is possible that there exist shorter strings than we can find. In this regard, we are exactly limited to finding an upper bound on the shortest prompt (as long as we do not search the astronomically large set of all prompts). But we can ease our minds by examining the minimal prompts we find for the four datasets above when we swap a random search technique for GCG in the MINIPROMPT algorithm. In fact, random search (see Algorithm 3) does slightly worse as an optimizer but tells the same story across the board. Since random search is gradient-free, this experiment quells any fears that GCG is merely relaying that the gradients are more informative on some examples than others. The details of this experiment and our exact random search algorithm are in Appendix A.

5 Additional Related Work

In addition to existing notions of memorization, our work touches on prompt optimization, compression in LLMs, and machine unlearning. In this section, we situate our approach and experimental results among the existing works from these domains.

Prompt Optimization We borrow prompt optimization tools from work on jailbreaking where the goal is to force LLMs to break their alignment and produce nefarious and toxic output by way of optimizing prompts (Zou et al., 2023; Zhu et al., 2023; Chao et al., 2023; Andriushchenko, 2023). Our extension of those techniques toward ends other than jailbreaking adds to the many and varied objectives that these discrete optimizers are useful for minimizing (Geiping et al., 2024).

Compression in LLMs There are several links between compression and language modelling and we borrow some vocabulary, but our work diverges from these other lines of research. For example, Delétang et al. (2023) argue that LLMs are compression engines, but they use models as probability distributions over tokens and arithmetic encoding to show that LLMs are good general compressors. As a metric for memorization, however, it is key that the compression algorithm is not generally useful, or it will tend to distinguish natural language that conforms to the LLMs probability distribution from data that does not, rather than help isolate memorized samples. Other links to compression include the ideas that learnability and generalization with real data comes in part from the compressability of natural data (Goldblum et al., 2023) and that grokking is related to the compressibility of networks themselves (Liu et al., 2023). Our work does not make claims about the compressibility of datasets or models in principle but rather capitalizes on the fact that input-output compression using adversarially computed prompts for LLMs captures something interesting as it relates to memorization and fair use. In fact, Jiang et al. (2023) propose prompt compression for reducing time and cost of inference, which motivates our work as it suggests that we should be able to find short prompts that elicit the same responses as longer more natural-sounding inputs in some cases.

Unlearning The focus of machine unlearning (Bourtoule et al., 2021; Sekhari et al., 2021; Ullah et al., 2021) is to remove private, sensitive, or false data from models without retraining them from scratch. Finding a cheap way to arrive at a model similar to one trained without some data is of practical interest to model owners, but evaluation is difficult. When motivated by privacy, the aim is to find models that leak no more information about an entity than a model trained without data on that entity. This is intimately related to memorization, and so we use a popular unlearning benchmark (Maini et al., 2024) in our experiments.

6 Discussion

When proposing new definitions, we are tasked with justifying why a new one is needed and as well as showing its ability to capture a phenomenon of interest. This stands in contrast to developing detection/classification tools whose accuracy can easily be measured using labeled data. It is a difficult task by nature to define memorization as there is no set of ground truth labels that indicate which samples are memorized. Consequently, the criteria for a memorization definition should rely on how useful it is. Our definition is a promising direction for future regulation on LLM fair use of data as well as helping model owners confidently release models trained on sensitive data without releasing that data.

From an intuitive perspective, one can imagine compression as a communication game—Alice tries to send some data to Bob with as few bits as possible. In our regime, we assume that both Alice and Bob have access to the same LLM M as the compressor. If Alice wants to send some string y to Bob, she can instead send the minimal prompt x and Bob can decode $y = M(x)$. When $|x| \leq |y|$, Alice has successfully compressed the data y . Future work to explore other ways of measuring compression and faster/cheaper optimization approaches could be fruitful. One drawback of optimizing hard tokens is the intrinsic complexity of discrete optimization. Instead, if we optimize over the continuous soft token space, the process will be much more efficient, but it remains unclear how to measure the information content of a soft token.

References

- Maksym Andriushchenko. Adversarial attacks on gpt-4 via simple random search. 2023.
- 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. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.
- Lucas Bourtole, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.
- Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, pp. 267–284, 2019.
- Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. *arXiv preprint arXiv:2112.03570*, 2021.
- Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models, 2023.
- Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.
- A Feder Cooper, Katherine Lee, James Grimmelmann, Daphne Ippolito, Christopher Callison-Burch, Christopher A Choquette-Choo, Niloofar Miresghallah, Miles Brundage, David Mimno, Madiha Zahrah Choksi, et al. Report of the 1st workshop on generative ai and law. *arXiv preprint arXiv:2311.06477*, 2023.
- Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.
- Michael Duan, Anshuman Suri, Niloofar Miresghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv:2402.07841*, 2024.
- Ronen Eldan and Mark Russinovich. Who’s harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238*, 2023.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.
- Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. Coercing llms to do and reveal (almost) anything. *arXiv preprint arXiv:2402.14020*, 2024.
- Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*, 2023.
- Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In *Proceedings of the 16th International Natural Language Generation Conference*, pp. 28–53. Association for Computational Linguistics, 2023.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llm-lingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.
- Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*, 2023.
- Ziming Liu, Ziqian Zhong, and Max Tegmark. Grokking as compression: A nonlinear complexity perspective. *arXiv preprint arXiv:2310.05918*, 2023.

-
- Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. *arXiv preprint arXiv:2104.10706*, 2021.
- Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.
- Cade Metz and Katie Robertson. Openai seeks to dismiss parts of the new york times’s lawsuit. *The New York Times*. URL <https://www.nytimes.com/2024/02/27/technology/openai-new-york-times-lawsuit.html#:~:text=In%20its%20suit%2C%20The%20Times,someone%20to%20hack%20their%20chatbot>.
- Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.
- CA OAG. Ccpa regulations: Final regulation text. *Office of the Attorney General, California Department of Justice*, 2021.
- Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. In-context unlearning: Language models as few shot unlearners. *arXiv preprint arXiv:2310.07579*, 2023.
- Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.
- Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pp. 4126–4142. PMLR, 2021.
- European Union. Regulation (eu) 2016/679 of the european parliament and of the council. *Official Journal of the European Union*, 2016.
- Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362, 2023.
- Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. Autodan: Automatic and interpretable adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023.
- Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Algorithms In Our Experiments

Algorithm 1 MINIPROMPT

Input: Model M , Vocabulary \mathcal{V} , Target Tokens y , Maximum Prompt Length \max
Initialize `n_tokens_in_prompt` = 5
Initialize `running_min` = 0, `running_max` = \max ,
Define $\mathcal{L}(y|x; M)$ as autoregressive next token prediction loss over y given x as context.
repeat
 $z = \text{GCG}(\mathcal{L}, \mathcal{V}, y, \text{n_tokens_in_prompt}, \text{num_steps})$ ▷ Or other discrete optimizer.
 if $M(z) = y$ **then**
 `running_max` = `n_tokens_in_prompt`
 `n_tokens_in_prompt` = `n_tokens_in_prompt` - 1
 `best` = z
 else
 `running_min` = `n_tokens_in_prompt`
 `n_tokens_in_prompt` = `n_tokens_in_prompt` + 5
 end if
until `n_tokens_in_prompt` \leq `running_min` **or** `n_tokens_in_prompt` \geq `running_max`
return `best`

Algorithm 2 Greedy Coordinate Gradient (GCG) (Zou et al., 2023)

Input: Loss \mathcal{L} , Vocab. \mathcal{V} , Target y , Num. Tokens `n_tokens`, Num. Steps `num_steps`
Initialize prompt x to random list of `n_tokens` tokens from \mathcal{V}
 $E = M$'s embedding matrix
for `num_steps` times **do**
 for $i = 0, \dots, \text{n_tokens}$ **do**
 $\mathcal{X}_i = \text{Top-}k(-\nabla_{e_{x_i}} \mathcal{L}(y|x))$
 end for
 for $b = 1, \dots, B$ **do**
 $\tilde{x}^{(b)} = x$
 $\tilde{x}_i^{(b)} = \text{Uniform}(\mathcal{X}_i), i = \text{Uniform}([1, \dots, \text{n_tokens}])$
 end for
 $x = \tilde{x}^{(b^*)}$ where $b^* = \arg \min_b \mathcal{L}(y|\tilde{x}^{(b)})$
end for
return x

Algorithm 3 Random Search (for LLM prompts) (Andriushchenko, 2023)

Input: Loss \mathcal{L} , Vocab. \mathcal{V} , Target y , Num. Tokens `n_tokens`, Num. Steps `num_steps`
Initialize prompt x to random list of `n_tokens` tokens from \mathcal{V}
for `num_steps` times **do**
 for $b = 1, \dots, B$ **do**
 $\tilde{x}^{(b)} = x$
 $\tilde{x}_i^{(b)} = \text{Uniform}(\mathcal{V}), i = \text{Uniform}([1, \dots, \text{n_tokens}])$
 end for
 $x = \tilde{x}^{(b^*)}$ where $b^* = \arg \min_b \mathcal{L}(y|\tilde{x}^{(b)})$
end for
return x

B More Details of In-context Unlearning

As a proof of concept, we further perform adversarial compression on five famous quotes and get an average compression ratios of 6.54 and 4.74, with versus without the unlearning system

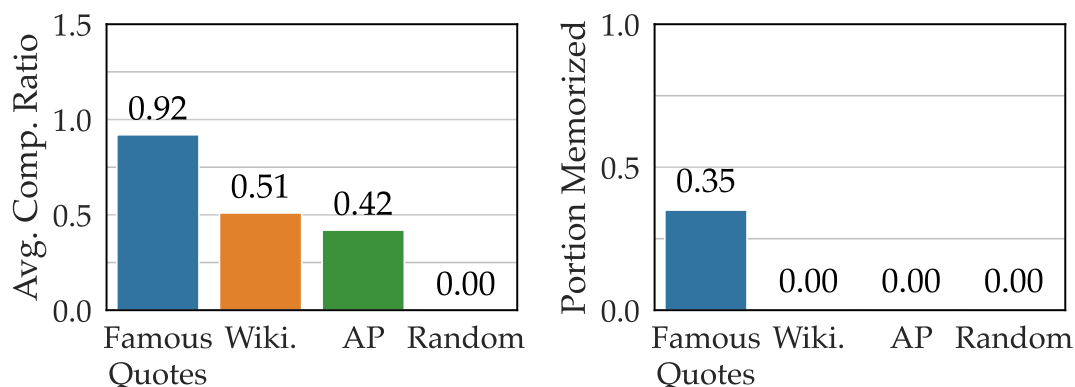


Figure 9: Pythia-1.4B Memorization with Random Search.

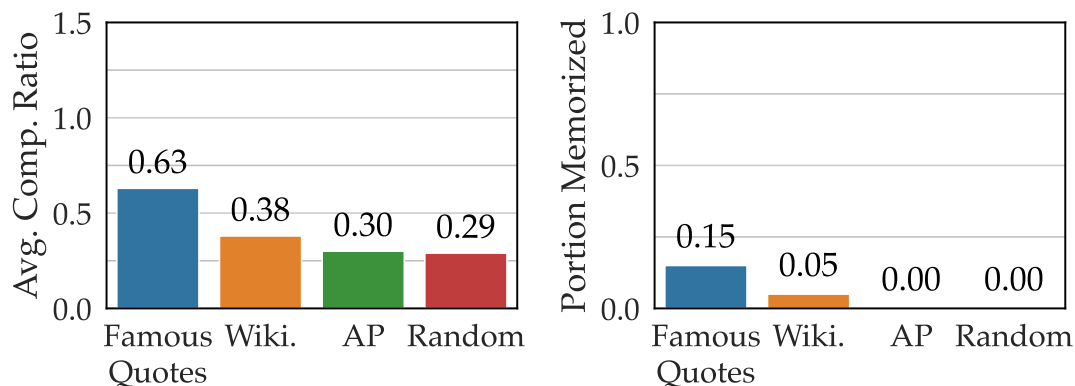


Figure 10: Pythia-410M Memorization with GCG.

prompt, respectively. Adding the unlearning prompt does seem to make adversarial compression slightly harder, but the key takeaway is that all of the quotes still enjoy compression ratios larger than one – they are still all memorized. This is really a sanity check since there is no weight update, naturally none of the internal knowledge should have been forgotten.

C More Details of Unlearning Harry Potter

As we mention in the main text, unlearning a prevalent concept in the pretraining data is hard. A cleverly phrased question can defeat the seemingly successful unlearning attempt. One of the example we provide is to ask questions about Harry Potter in Russian, which makes the model generate Harry Potter facts very easily. Here are some examples.

- Prompt:** [INST] Кто такой Гарри Поттер? [/INST] (Translate: Who is Harry Potter?)
Generation: Гарри Поттер - это вымышленный персонаж, основанный на персонаже из серии романов английской писательницы Джоан Роулинг. (Translation: Harry Potter is a fictional character based on a character from a series of novels by English writer JK Rowling.)
- Prompt:** [INST] Кто такой Дамблдор в книге о Гарри Поттере? (Translation: Who is Dumbledore in the Harry Potter series book?)

Generation: Дамблдор - это персонаж из серии романов о Гарри Поттере, написанных английской писательницей Джоан Роулинг. (Translation: Dumbledore is a character from the Harry Potter series of novels written by the English writer JK Rowling.)

- **Prompt:** [INST] Кто такой Северус Снейп в книге о Гарри Поттере? [/INST] (Translation: Who is Severus Snape in the Harry Potter series book?)

Generation: Северус Снейп (Severus Snape) - это персонаж из серии книг о Гарри Поттере, написанных английской писательницей Джоан Роулинг. Северус Снейп - это учитель магии, который работает в школе магии (Translation: Severus Snape is a character from the Harry Potter series of books written by the English writer JK Rowling. Severus Snape is a magic teacher who works at a magic school)