How much can language models memorize?

Abstract

Due to the inherent structure of language, prior studies of language model memorization have struggled to disentangle memorization from generalization. We formally separate memorization into two components: *unintended memorization*, the information a model contains about a specific dataset, and *generalization*, the information a model contains about the true data-generation process. Our framework allows us to cleanly separate memorization and generalization in a variety of settings. When we completely eliminate generalization, we can compute the exact capacity of language models; our measurements estimate that GPT-style models have a capacity of approximately 3.6 bits per parameter. We train language models on datasets of increasing size and observe that models memorize via unintended memorization until their capacity fills, at which point memorization decreases as models begin to generalize. We train hundreds of transformer language models ranging from $500K$ to $1.5B$ parameters and produce a series of scaling laws relating model capacity and data size to membership inference.

1. Introduction

For the past several years, modern language models have been trained on increasingly large amounts of data, while parameter counts stay stagnant in the billions. For example, one recent state-of-the-art model (Dubey & et al, 2024) has 8 billion parameters (around $32GB$ on disk) but is trained on 15 trillion tokens (around $7TB$ on disk).

A long line of work (Carlini et al., 2019; Mireshghallah et al., 2022; Nasr et al., 2023; Zhang et al., 2023; Carlini et al., 2023b; Schwarzschild et al., 2024) questions whether such pretrained language models memorize their training data in a meaningful way. Most research approaches this problem either through the lens of extraction, aiming to recover full training data points from model weights, or membership inference, simply classifying whether a training point was present in the training data of a given model.

Studies of language model extraction argue that a data point is memorized if we can induce the model to generate it (Carlini et al., 2023b; Nasr et al., 2023; Schwarzschild et al., 2024). We argue that because language models can be coerced to output almost any string (Geiping et al., 2024) simple willingness to output something is weak proof of memorization. Additionally, some strings are outputted due simply to good generalization: for example, a good language model prompted to add two numbers can output the correct answer without having seen the equation before.

If extraction is unreliable, what is the right way to define memorization? We note that "memorization" of a datapoint is tied to its inherent compression rate in bits (Carlini et al., 2019) and the broader question of how to distinguish memorization from generalization (Prashanth et al., 2024). We separate memorization into two components: *unintended memorization*, the information a model contains about a specific dataset, and *generalization*, the information a model contains about the true data-generation process.

To understand these new quantities, we measure unintended memorization and generalization by training language models of varying capacity on datasets of different sizes. We first eliminate the question of generalization entirely by training on a dataset of random uniformly-sampled bitstrings. In this setting, we can exactly measure the amount of information contained about the data inside the model. This gives us a principled way to measure language model *capacity* when trained on uniform datasets of exact known information content. We find that GPT-style transformers can store between 3.5 and 4 bits of information in each model parameter, depending on model architecture and precision.

We then repeat our experiments with real text, where generalization is possible and even beneficial for learning. On real text, language models memorize up to a certain capacity, at which point they substitute unintended memorization for generalization, and begin to learn general, reusable patterns as opposed to sample-level specifics. Our framework shows that double descent phenomenon begins to occur at this point, when the data size exceeds the model capacity in bits.

Finally, we use our results to predict a scaling law for membership inference performance based on model capacity and dataset size. We show that membership inference follows a clean relationship based on model capacity and dataset size: bigger models can memorize more samples, and making datasets bigger makes membership inference harder. Our

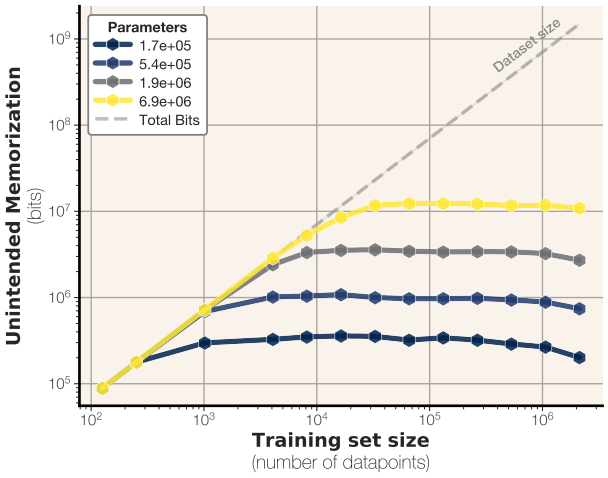

Figure 1. **Unintended memorization of uniform random synthetic data** (Section 3). Memorization plateaus at the capacity of different-sized models from the GPT-family. Models stop memorizing when trained on data exceeding their capacity.

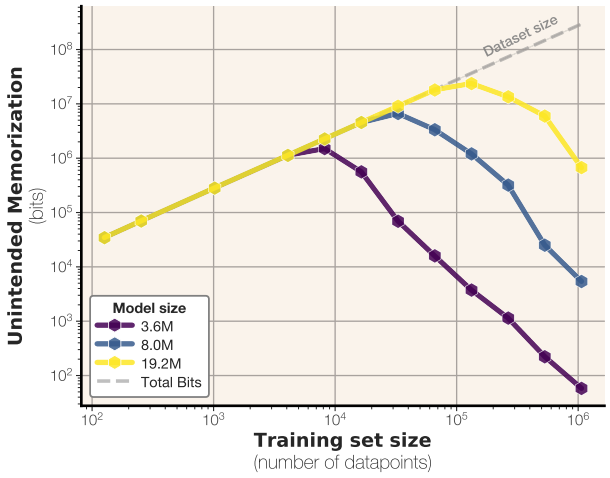

Figure 2. **Unintended memorization of text data** across model and dataset sizes (Section 4). All quantities are calculated with respect to a large oracle model (1B params). Horizontal lines indicate the expected capacity of each model according to our α.

scaling laws extrapolate to larger models, and predict most modern language models are trained on too much data to do reliable membership inference on the average data point.

2. Related Work

Language models and compression. Shannon's source coding theorem (Shannon, 1948) first formalized the duality between prediction and compression. The connection between language modeling and compression was studied as far back as Shannon (1950), which observed that more accurate models of English can compress text in fewer bits. Other works note the connection between Kolmogorov complexity (Kolmogorov, 1965) and Shannon information in detail (Grunwald & Vitanyi, 2004). Delétang et al. (2024) investigate using modern transformer-based language models as compressors. We use compression as a tool to measure memorization in models.

Language model capacity. (Arpit et al., 2017) formalize the idea of *effective capacity* of a model and its training procedure; they also observe that both representation capacity and training time have a strong impact on empirical model capacity. Several other works measure language model capacity in the number of facts or random labels that can be memorized by a network such as an RNN (Collins et al., 2017; Boo et al., 2019) or transformer (Roberts et al., 2020; Heinzerling & Inui, 2021; Allen-Zhu & Li, 2024), sometimes under quantization. A few research efforts (Yun et al., 2019; Curth et al., 2023; Mahdavi et al., 2024; Kajitsuka & Sato, 2024) have developed theoretical estimates for the capacity of different model architectures, although none have yet scaled to multi-layer modern transformers. We are the first to measure a clear upper-bound in model capacity.

Alternative definitions of memorization. Unintended memorization is deeply related to the many other definitions of memorization proposed in the literature. We provide a detailed comparison in Section B.

2.1. Memorization, intended and unintended

When a model $\theta = L(x)$ is trained using a training algorithm L and a dataset $x \sim X$, some information is transferred from the sample x to the model θ. A key question in the memorization literature is determining how much of this stored information is intended versus unintended. In this work, we aim to provide a rigorous definition of memorization that satisfies certain properties:

1. *Separation from generalization.* Our notion of unintended memorization must be distinct from intended memorization, which we refer to as generalization. For example, consider a language model trained on the sample: *Q: What is* 2^{100}*? A: 1267650600228229401496703205376.* When assessing how much of this training sample is memorized, we must account for the fact that performing simple math operations is expected from a language model.

2. *Sample-level memorization.* We need to define memorization for realizations of random variables, not the random variables themselves. Specifically, we want to determine how much unintended memorization of a sample x occurs in a model θ.

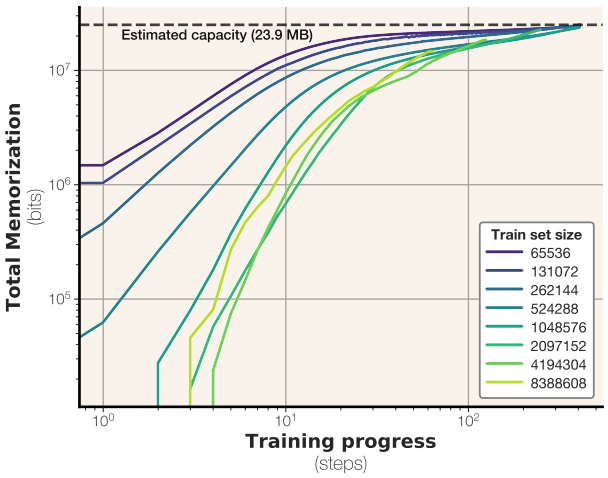

Figure 3. Bits memorized across training. This particular model is a GPT-style transformer with $6.86M$ parameters and a capacity of 23.9 MB.

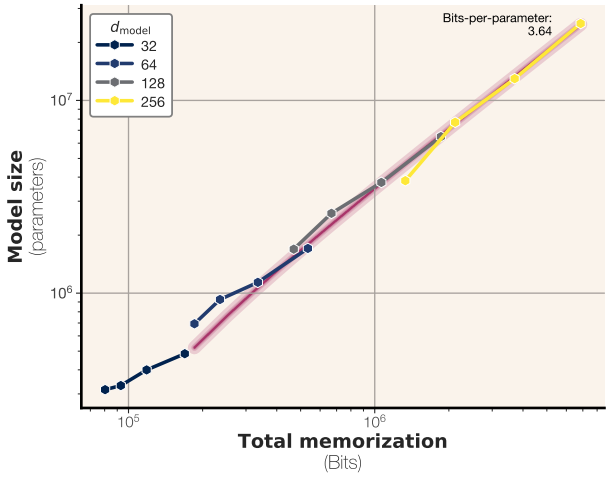

Figure 4. Capacity in bits-per-parameter for models trained on synthetic data. We estimate $\alpha = 3.64$ bits-per-parameter for GPT models trained in half precision.

3. *Independence from training algorithm.* Our definition should be independent of the training algorithm L and only a function of the final model θ and the sample x. This is crucial for language models, where we often only have access to the final model and target sample.

Previous works have attempted to define memorization for machine learning models. We aim to provide precise definitions of memorization that meet our criteria, and offer ways to measure it. See Appendix B for a broader discussion on definitions of memorization.

2.2. A statistical view of memorization

Notation. In this section, we use capital letters (e.g. X, Θ) to refer to random variables and lowercase letters to refer to instances of a random variable (e.g. $x \sim X$ and $\theta \sim \Theta$).

Information theory has developed well understood notions of information for random variables. For a random variable X, we often use $H(X)$, the entropy of X, to define the amount of information present in X. Moreover, for two distinct random variables X, Y, we can define $X \mid Y$ to be the uncertainty left in X after fixing Y. Having defined this quantity, we can now measure *mutual information* between X and Y by subtracting the leftover information from the total information: $I(X, Y) = H(X) - H(X \mid Y)$.

Now assume we have a machine learning pipeline. We have a prior Θ on the underlying model that captures our dataset distribution X. And we have a learning algorithm L that maps samples from X to a trained model $\hat{\Theta}$. To understand how much information about X is stored in $\hat{\Theta}$, we can use the notion of mutual information:

$$\text{mem}(X, \hat{\Theta}) = I(X, \hat{\Theta}) = H(X) - H(X \mid \hat{\Theta}).$$

Note that this captures all the information about X that is stored in $\hat{\Theta}$. As we discussed, we need our notion of memorization to account for generalization as well. So when measuring unintended memorization, we are only interested in the information that is present in $X \mid \Theta$, which is the uncertainty left in X after fixing Θ. Hence, we can define **unintended memorization** as

$$\begin{aligned}\text{mem}_U(X, \hat{\Theta}, \Theta) &= I([X \mid \Theta], \hat{\Theta}) \\ &= H(X \mid \Theta) - H(X \mid (\Theta, \hat{\Theta})).\end{aligned}$$

and then the **generalization** (or intended memorization) must be

$$\begin{aligned}\text{mem}_I(\hat{\Theta}, X, \Theta) &= \text{mem}(X, \Theta) - \text{mem}_U(X, \hat{\Theta}, \Theta) \\ &= I(X, \hat{\Theta}) - I(X \mid \Theta, \hat{\Theta})\end{aligned}$$

Now that we have defined our notions of intended and unintended memorization we turn our attention to practically measuring them. Let us first state a proposition that enables measurement of unintended memorization:

Proposition 1 (Super-additivity of Unintended Memorization). Assume $X = (X_1, \ldots, X_n)$ is a dataset of n i.i.d. samples. We have

$$\sum_{i \in [n]} \text{mem}_U(X_i, \hat{\Theta}, \Theta) \leq \text{mem}_U(X, \hat{\Theta}, \Theta) \leq H(\hat{\Theta}).$$

This proposition shows that to measure a lower bound on the unintended memorization on the dataset level, we can sum

per-sample memorization. On the other hand, the entropy of the information content of the trained model itself servers as an upper bound on the unintended memorization. Another implication of this implies that unintended memorization should scale with the dataset size but cannot exceed the total capacity of the model.

2.3. Measuring unintended memorization with Kolmogorov Complexity

Our definitions of memorization and generalization so far are defined using an "entropy-based" notion of information. This means our definitions can only be used for random variables. This brings big challenges in measuring memorization. All our variables in the definition of memorization are singletons. We have a single underlying model θ, we have a single dataset $x = (x_1, \ldots, x_n)$ and we have a single trained model $\hat{\theta}$[1]. It is impossible to measure the entropy (let alone conditional entropy) of the underlying variables using a single sample.

To this end, we switch to another notion of information based on compression, then later we show how this notion closely approximates the notion of memorization defined above. Kolmogorov complexity defines the information content of a string x, denoted as $H^K(x)$, to be the length of shortest representation of x in a given computational model. Similarly, we can define the leftover information $x \mid \theta$, to be the shortest representation of x, when we have θ available as a reference. And the information content of $x \mid \theta$, denoted by $H^K(x \mid \theta)$, is the length of such description. Then, we can define mutual information in a similar fashion:

Definition 2 (Kolmogorov complexity). Let f be an arbitrary computational model that takes a set of inputs and returns an output (e.g. universal Turing machine). The shortest description of x with respect to computational model f is defined as $H^K(x) = \min_{f(p)=x} |p|$. Also, the Kolmogorov complexity of x relative to another string θ is defined as $H^K(x \mid \theta) = \min_{f(p,\theta)=x} |p|$. And we define the Kolmogorov mutual information between x and θ by $I^K(x, \theta) = H^K(x) - H^K(x \mid \theta)$. We assume inputs are bitstrings and $|p|$ is the bit length of the input.

We are now ready to define Kolmogorov memorization.

Definition 3 (Kolmogorov memorization). Let θ be a reference model that approximates the true distribution of data, and $\hat{\theta}$ be a model trained on a dataset $x = (x_1, \ldots, x_n)$. For each x_i we define the memorization of x_i in $\hat{\theta}$ as

$$\text{mem}^K(\hat{\theta}, x) = I^K(\hat{\theta}, x).$$

We also define intended and unintended variants of memorization:

$$\text{mem}_U^K(x, \theta, \hat{\theta}) = H^K(x \mid \theta) - H^K(x \mid (\theta, \hat{\theta})).$$

and

$$\text{mem}_I^K(x, \theta, \hat{\theta}) = \text{mem}^K(x, \hat{\theta}) - \text{mem}_U^K(x, \theta, \hat{\theta}).$$

There are known connections between Kolmogorov complexity and Shannon Entropy (Grunwald & Vitanyi, 2004). These results point at the conceptual connection between the two notions and imply that $\mathrm{E}_{x \sim X}[H^K(x)] \approx H(X)$. Interestingly, this implies that our notion of Kolmogorov memorization closely approximates Shannon memorization.

Proposition 4. Let $X = (X_1, \ldots, X_n)$ be an i.i.d, dataset distribution parametrized by ground-truth model θ. Let L be a training algorithm mapping X to $\hat{\Theta}$. Assume $H(\hat{\Theta}) = \ell$ and $H(X_i) = \ell'$ [2]. Then we have

$$\Big| \mathop{\mathrm{E}}_{\substack{x \sim X \\ \hat{\theta} \sim L(x)}} \big[\text{mem}_U^K(x_i, \hat{\theta}, \theta)\big] - \text{mem}_U(X_i, \hat{\Theta}, \theta) \Big| \leq \epsilon.$$

for some constant ϵ independent of θ, ℓ, ℓ' and n.

2.4. Estimating Kolmogorov with compression

Fixing our notion of Kolmogorov memorization, we now describe how we can estimate H^K in different setups. Note that exact calculation of Kolmogorov complexity is known to be uncomputable (the decision version of is undecidable). However, we can still approximate it using the best available compression schemes. Below, we summarize how we approximate each term in our definition.

- $H^K(x)$: This is the shortest description of x when we have no knowledge about the underlying representation of x. For this, we can use best compression algorithms (e.g. gzip) to calculate the information content in x.
- $H^K(x \mid \theta)$: This quantifies the shortest description of x when we know that x is sampled from an underlying model θ. We know that for such scenarios, arithmetic coding is the optimal compression algorithm in expectation (Pasco, 1977). Hence, we use arithmetic coding to calculate the code length of x using the model θ. Note that we do not know the real model θ, but we can still approximate it using the state-of-the-art models that capture the underlying distribution. We call these models our reference models.
- $H^K(x \mid \hat{\theta})$: Here, $\hat{\theta}$ is a model that is trained and we call it the target model. The target model does not

[1] Note the switch to lowercase variables because we are now working with instances, not random variables.

[2] The trained model and each data sample can be presented using ℓ and ℓ' bits respectively.

necessarily capture the true data distribution. Depending on the training algorithm, we need to optimize a coding scheme that can losslessly compress x using $\hat{\theta}$.

- $H^K(x, \mid \hat{\theta}, \theta)$: In this case, the compression algorithm has access to both target and reference models. In this case, we also need to design a compression algorithm that benefits from the knowledge of θ and $\hat{\theta}$.

3. Model Capacity for Memorization

Unintended memorization provides us a principled way of measuring the precise number of bits a model θ knows about a datapoint x. If we add up the information for each datapoint in a dataset, we can measure the total amount of bits a model knows about the dataset. And in cases where generalization is not possible because each datapoint is completely independent, we can estimate the **capacity** of a given model θ by summing per-datapoint unintended memorization.

3.1. Definition of capacity

We first formalize this notion of memorization capacity for a particular language model θ. Capacity is the total amount of memorization that can be stored in θ across all its parameters.

Definition 5 (Capacity). Let X be a distribution and $L\colon X \to \Theta$ a learning algorithm. We define the capacity of the learning algorithm L to be

$$\text{Capacity}(L) = \max_{X} \text{mem}\big(X, L(X)\big)$$

When the model capacity is reached, $\text{mem}(X, L(X))$ will no longer increase with dataset size. In practice, we can compute capacity by training to saturation on varying sizes of X and computing the maximum memorization.

3.2. Measuring model capacity with synthetic sequences

In this section we measure the capacity of Transformer language models. Our goal is to instantiate multiple datasets and distributions and measure the memorization of them when training a single model θ. Then, we take the maximum over all datasets to approximate of the model's capacity. For instantiating our datasets, each token is uniformly sampled from a predefined set of tokens independent of the previous tokens.

To approximate $H^k(x \mid \theta, \hat{\theta})$, we can simply use arithmetic coding using the trained model to calculate the shortest description of the dataset conditioning on $\hat{\theta}$. Subtracting the two, we can approximate the unintended memorization $\text{mem}_U(X, L(X))$. Since the process for sampling the data is completely random, there is no generalizationto be stored within $\hat{\theta}$ ($\text{mem}^U(X, L(X)) \approx \text{mem}(X, L(X))$).

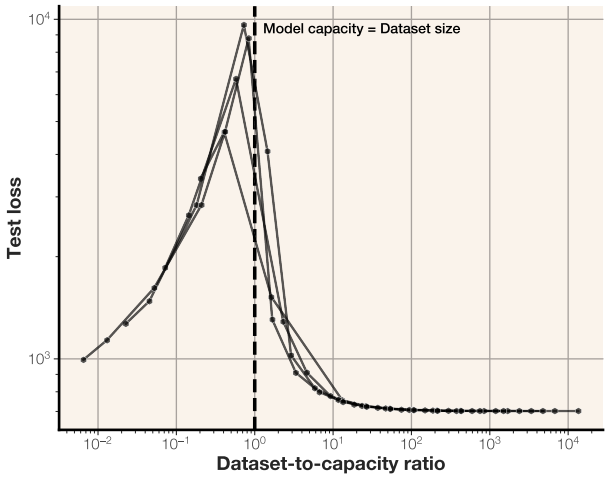

Figure 5. We show that double descent occurs exactly when the dataset size begins to exceed the model's capacity, when unintended memorization is no longer beneficial for lowering the loss.

Observe that when we sample synthetic sequences from a uniform distribution, we can compute their Shannon information exactly. Given a dataset size N, we construct a dataset of N sequences, each of S tokens. Given a vocabulary size V, we can calculate the total entropy of a dataset x^i with such parameters by $H(x^i) = NS\log_2 V$. Then we calculate the compressed form x^i using arithmetic coding with $\hat{\theta}_i$ to calculate the code length and use that as an approximation of $H^K(x^i \mid \hat{\theta}_j)$. Then we calculate the $\text{mem}(x^i, \hat{\theta}_i) = H(x^i) - H^K(x^i \mid \hat{\theta}_j)$ and compute a model's capacity as the maximum amount of memorization over all datasets.

Experimental details. In accordance with Kaplan et al. (2020), we train models with the GPT-2 architecture (Radford et al., 2019) initialized from scratch. Our models have between 1 and 8 layers, hidden dimensions from 32 to 512, and from $100K$ to $20M$ parameters. We train models for 10^6 steps with a batch size of 2048. We use the Adam optimizer. All models are trained on a single A100 GPU in bfloat16 precision, and we use gradient accumulation if a batch cannot fit in memory. Unless otherwise noted, we set vocabulary size $V = 2048$, sequence length $S = 64$ and vary only the number of points in a dataset. We train each model on each dataset size over five random seeds, which affect both model initialization and the dataset sampling.

Results. We plot memorization across model and data sizes in Figure 1. This allows us to visualize unintended memorization amounts (y-axis) across dataset sizes (x-axis) grouped by model size (line color). We observe a striking plateau once a model reaches its capacity. Given the dataset is large enough, models exhibit an upper bound in

net memorization, regardless of data size. Small datasets are completely memorized by all models with enough capacity.

We estimate the capacity of each model as the maximum amount of unintended memorization in bits measured across all dataset sizes. We then compare this capacity to the model size in Figure 4. Interestingly, even at this small scale, we see a very smooth relationship between observed capacity (maximum memorization measured over all datasets) and model parameters. We plot this relationship in Figure 4: under these settings, our models consistently memorize between 3.5 and 3.6 bits per parameter. This corroborates the findings of prior work such as (Roberts et al., 2020; Lu et al., 2024), which noticed that fact storage scales linearly with model capacity. Ours is a slightly larger estimate than Allen-Zhu & Li (2024), which estimated via quantization that models can store around 2 bits per parameter.

Since our models are learned via gradient descent, they are not guaranteed to find the global optima; thus, we are only ever measuring a lower bound on model capacity. We take a closer look at the training curves to analyze the convergence of our 8M parameter language model. We plot model convergence throughout training in Figure 3.

In this case, all datasets from 16,000 to 4M samples fall within a range of $3.56 - 3.65 \times 10^6$ bits memorized. This indicates that our measurements are robust within an order of magnitude, and we do not expect to memorize significantly more information by training for more steps. This finding also confirms our hypothesis that capacity scales roughly with parameter count. The two largest datasets (4M and 8M samples, respectively) converge to total memorization of 2.95×10^6 and 1.98×10^6 bits memorized. We expect that their memorization rates would continue to increase toward the capacity had we trained for more epochs.

How does precision affect capacity? One natural question is how our estimates for α depend on the precision of language model training. In fact, although most software defaults to training in 32-bit precision, recent work has shown that language models can be quantized to fewer than 2 bits per parameter and still retain much of their utility. Since all other experiments have been conducted in bfloat16 precision, rerun our experiments in full fp32 precision to analyze the effect on capacity. Across model sizes, we observe a small increase in capacity, and an increase in α from 3.51 to 3.83 bits-per-parameter on average. This is far less than the actual 2x increase in the bits of θ, indicating that **most of the extra model bits added when increasing precision from bfloat16 to float32 are not used** for raw storage.

Membership inference. Finally, we perform a standard loss-based membership inference (Yeom et al., 2018; Sablayrolles et al., 2019) attack on each model and plot performance across dataset sizes. We show results in Figure 13. Above a certain dataset size, membership inference starts to fail in the average case. This finding indicates that if the dataset size is too large compared to the model, membership inference of an average training sample may not be possible.

n_{layer}	d_{model}	Params	Capacity(θ)	*[bits]*	α	*[bpp]*
1	32	8.04×10^4	3.39×10^5	3.16×10^5	4.23	3.93
	64	1.85×10^5	7.27×10^5	6.93×10^5	3.92	3.74
	128	4.69×10^5	1.71×10^6	1.69×10^6	3.65	3.61
	256	1.33×10^6	4.15×10^6	3.83×10^6	3.12	2.88
2	32	9.31×10^4	3.87×10^5	3.31×10^5	4.16	3.56
	64	2.35×10^5	9.60×10^5	9.27×10^5	4.08	3.94
	128	6.67×10^5	2.66×10^6	2.60×10^6	3.99	3.89
	256	2.12×10^6	8.49×10^6	7.76×10^6	4.01	3.66
4	32	1.18×10^5	4.65×10^5	3.99×10^5	3.92	3.37
	64	3.35×10^5	1.34×10^6	1.14×10^6	3.98	3.39
	128	1.06×10^6	4.02×10^6	3.75×10^6	3.78	3.53
	256	3.70×10^6	1.36×10^7	1.30×10^7	3.68	3.51
8	32	1.69×10^5	5.12×10^5	4.85×10^5	3.02	2.86
	64	5.35×10^5	2.05×10^6	1.71×10^6	3.83	3.19
	128	1.86×10^6	7.23×10^6	6.49×10^6	3.89	3.49
	256	6.86×10^6	2.71×10^7	2.51×10^7	3.96	3.65
					Mean (±0.1):	
					3.83	3.51

Table 1. Model capacity estimates across different widths and depths in full and half-precision. Doubling precision from bfloat16 to float32 only increases model capacity from 3.51 to 3.83 bits-per-parameter.

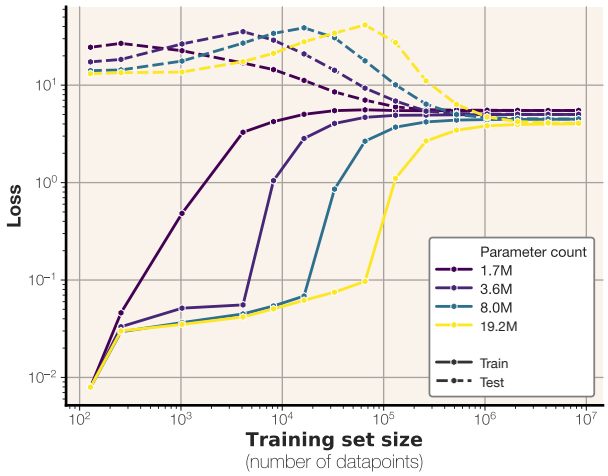

Figure 6. Train and test losses of different model and dataset sizes trained on text. Double descent occurs when dataset size exceeds model capacity.

4. Disentangling Unintended Memorization from Generalization

Our previous experiments analyzed the memorization and membership inference properties of synthetic bitstrings. We now turn to measuring memorization of text. Unlike randomly generated sequences, learning from text data is a mix of both unintended memorization (sample-level) and generalization (population-level).

Experimental details. We repeat the experiments from 3.2, substituting our synthetic datapoints for real text. To obtain a distribution of real-world text data, we could use any pre-training scale text dataset; we use the recently proposed FineWeb dataset (Penedo et al., 2024) as it follows state-of-the-art deduplication practices. We use sequences of 64 tokens but perform an additional deduplication step to ensure perfect deduplication (otherwise, that $1-2\%$ of sequences become duplicates when truncating to 64 tokens). We find careful deduplication extremely important for faithfully measuring extraction rates. As in the previous subsection, we pretrain models of varying sizes on different-sized text datasets and measure the unintended memorization of each model-dataset pair. In addition to memorization, we measure membership inference performance according to a standard loss-based membership inference procedure; we also compute exact extraction rates by greedily decoding prefixes of different lengths.

Results. We first observe that the sample-level unintended memorization increases with model parameters and decreases with training set size (Figure 6). When we measure unintended memorization with respect to an oracle reference model (Figure 2), memorization steadily increases as our smaller model is able to learn more about the small training set than the oracle, and then decreases as our model starts to generalize and perform on average worse than the (higher-capacity) oracle.

Dataset-to-capacity ratio predicts double descent. We observe from the train and test loss that for larger datasets the model only begins to generalize (i.e. evaluation loss decreases) once its capacity is reached, which takes approximately 10^5 samples, depending on parameter count. As in Nakkiran et al. (2019) we plot the ratio between the dataset size and model capacity (Figure 5). Unlike prior work, in our experiments we can compute the exact dataset size (based on the compression rates of the reference model) and exact model capacity (based on our estimate of α).

We clearly observe double descent evaluation performance decreases as the training set size nears model capacity, and then rapidly drops as the dataset capacity exceeds the capacity of the model. Our observations offer an intuitive explanation for double descent (Belkin et al., 2019; Nakkiran et al., 2019): **double descent begins exactly when the data capacity exceeds the model capacity**. One theory is that once the model can no longer memorize datapoints individually, it is forced to share information between datapoints to save capacity, which leads to generalization.

Generalization explains nonzero extraction rates. We measure extraction rates on the full training set and 10,000 non-overlapping test samples (Figure 16). We note that for 32-token prefixes, 100% are extractable for very small training set sizes; predictably, all extraction numbers decrease with training set size. When the dataset sizes grows sufficiently large, the extraction rate does not go fully to zero; however, it converges to nearly exactly the test extraction rate. In other words, when our (deduplicated) dataset grows sufficiently large, **all successful training data extraction is attributable to generalization**.

Membership inference grows difficult at scale. For each of our models, we use unused non-overlapping data from FineWeb to perform a standard loss-based membership inference (Yeom et al., 2018; Sablayrolles et al., 2019) on each model and plot performance across dataset sizes (8). For a fixed model size, membership inference gets more difficult as the size of the data increases. When comparing membership inference to extraction (Figure 7), membership inference is strictly higher in every case; in some cases we can infer training dataset membership quite well (score of 0.97) with an extraction rate of 0.

5. Scaling Laws for Memorization

In this section we develop a set of predictive models for memorization. Specifically, we predict the F1 score of a loss-based membership attack given token count, number of examples, and model parameter count. We then validate our predictions on models from 500K to 1.5B parameters.

5.1. Functional forms

We observe that for a fixed model capacity, membership inference follows a roughly sigmoidal form with respect to dataset size. The intuitive explanation is that M.I. is easy for large models overfit to tiny datasets, so its score begin at 1; as dataset size increases, differentiating train from test data by loss becomes more and more difficult, eventually decaying toward 0.5.

We reuse the data collected in our text experiments (Section 4) to solve for constants c_1, c_2, c_3 in the following equation:

$$\text{Membership}_{F_1}(\theta, \mathcal{D}) = \frac{1}{2}(1 + c_1\sigma(c_2(\frac{\text{Capacity}(\theta)}{|\mathcal{D}|} + c_3))$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$.

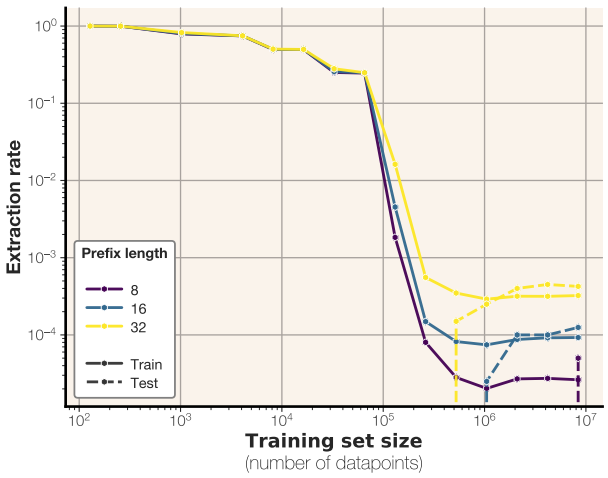

Figure 7. Extraction rates of 64-token training sequences across prefix lengths, for both train and evaluation.

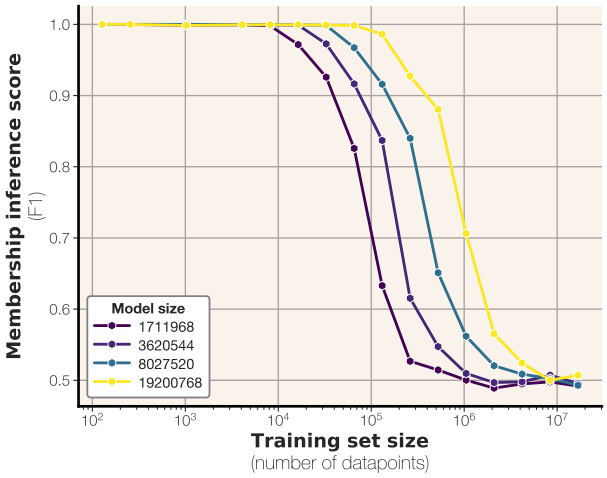

Figure 8. Membership inference F1 across dataset sizes. In this case, F1 score of 0.5 implies random guessing.

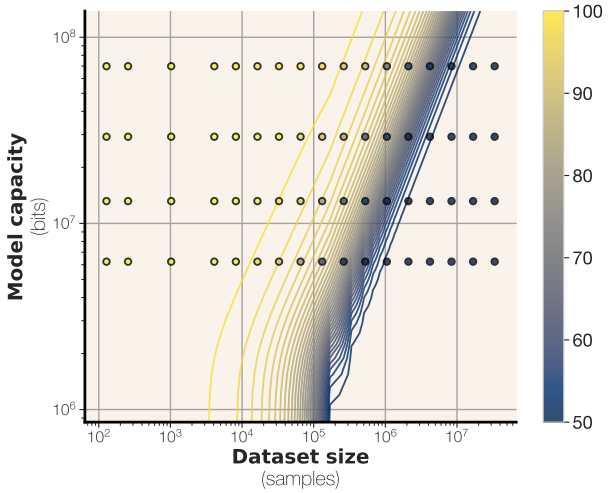

Figure 9. Scaling law curves for membership inference overlaid with empirical data, shown in circles.

Limiting behavior. We observe that as $|\mathcal{D}| \to \infty$, performance of our membership inference attack decreases to 0.5 (essentially random performance). For a model trained on an infinite dataset, our law predicts both membership inference and extraction to be impossible.

Fitting. We use a non-linear least squares solver to find optimal values for c_1, c_2, c_3. Solutions found are $c_1 = 1.34$, $c_2 = -0.034$, and -33.14. We plot the scaling laws along with observed data in Figure 9. Although the sigmoidal function is slightly simplistic (the points do not perfectly fit) our fit produces estimates within $1 - 2\%$ of observations.

5.2. Validation on larger models

We note that all contemporary language models trained with a tokens-per-parameter ratio of 10^2 or higher, which according to our laws would imply membership inference score of 0.5 – that is, within our formulation, statistically significant loss-based membership inference is not possible.

To validate our predictions, we train models with expected membership $F1$ scores of 0.55, 0.75, and 0.95. For model sizes we select GPT-2 small (125M params) and GPT-2 XL (1.5B params). Using our scaling law, we solve for the dataset size required to get the desired membership inference score for the given model size (see Table A.3 for more information). We train models on the estimated dataset size and measure F1 score (Figure 9). Our predictions are generally within 1.5 points of the true F1 score; the score is most inaccurate for estimated F1 of 0.75, which is the point where the sigmoid is steepest. In general, the accuracy of our results indicates that our empirical model of membership inference is relatively accurate and provides evidence for why membership inference attacks fail on models trained on extremely large datasets (Das et al., 2024; Duan et al., 2024; Maini et al., 2024).

6. Conclusion

We propose a new definition of memorization that allows us to measure the exact number of bits a model knows about a dataset. We use our definition to measure the capacity of modern transformer language models and analyze how measurements such as extraction and F1 score scale with model and dataset size. We also propose a scaling law for membership inference and validate it on larger models. Our results help further practitioner understanding of how language models memorize and what they might (or might not) be memorizing across model and dataset scales.

References

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.3, knowledge capacity scaling laws, 2024. URL https://arxiv.org/abs/2404.05405.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks, 2017. URL https://arxiv.org/abs/1706.05394.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, July 2019. ISSN 1091-6490. doi: 10.1073/pnas.1903070116. URL http://dx.doi.org/10.1073/pnas.1903070116.

Bhattacharjee, R., Dasgupta, S., and Chaudhuri, K. Data-copying in generative models: a formal framework. In *International Conference on Machine Learning*, pp. 2364–2396. PMLR, 2023.

Boo, Y., Shin, S., and Sung, W. Memorization capacity of deep neural networks under parameter quantization, 05 2019.

Carlini, N., Liu, C., Úlfar Erlingsson, Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks, 2019. URL https://arxiv.org/abs/1802.08232.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., and Wallace, E. Extracting training data from diffusion models, 2023a. URL https://arxiv.org/abs/2301.13188.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models, 2023b. URL https://arxiv.org/abs/2202.07646.

Cohen, E., Kaplan, H., Mansour, Y., Moran, S., Nissim, K., Stemmer, U., and Tsfadia, E. Data reconstruction: When you see it and when you don't, 2024. URL https://arxiv.org/abs/2405.15753.

Collins, J., Sohl-Dickstein, J., and Sussillo, D. Capacity and trainability in recurrent neural networks, 2017. URL https://arxiv.org/abs/1611.09913.

Curth, A., Jeffares, A., and van der Schaar, M. A u-turn on double descent: Rethinking parameter counting in statistical learning, 2023. URL https://arxiv.org/abs/2310.18988.

Das, D., Zhang, J., and Tramèr, F. Blind baselines beat membership inference attacks for foundation models, 2024. URL https://arxiv.org/abs/2406.16201.

Delétang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., Hutter, M., and Veness, J. Language modeling is compression, 2024. URL https://arxiv.org/abs/2309.10668.

Duan, M., Suri, A., Mireshghallah, N., Min, S., Shi, W., Zettlemoyer, L., Tsvetkov, Y., Choi, Y., Evans, D., and Hajishirzi, H. Do membership inference attacks work on large language models? In *Conference on Language Modeling (COLM)*, 2024.

Dubey, A. and et al, A. J. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dwork, C. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pp. 1–12. Springer, 2006.

Feldman, V. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–959, 2020.

Geiping, J., Stein, A., Shu, M., Saifullah, K., Wen, Y., and Goldstein, T. Coercing llms to do and reveal (almost) anything, 2024. URL https://arxiv.org/abs/2402.14020.

Grunwald, P. and Vitányi, P. Shannon information and kolmogorov complexity. *arXiv preprint cs/0410002*, 2004.

Grunwald, P. and Vitanyi, P. Shannon information and kolmogorov complexity, 2004. URL https://arxiv.org/abs/cs/0410002.

Heinzerling, B. and Inui, K. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries, 2021. URL https://arxiv.org/abs/2008.09036.

Jayaraman, B. and Evans, D. Are attribute inference attacks just imputation? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1569–1582, 2022.

Kajitsuka, T. and Sato, I. Optimal memorization capacity of transformers, 2024. URL https://arxiv.org/abs/2409.17677.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Kolmogorov, A. N. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better, 2022. URL https://arxiv.org/abs/2107.06499.

Lu, X., Li, X., Cheng, Q., Ding, K., Huang, X., and Qiu, X. Scaling laws for fact memorization of large language models, 2024. URL https://arxiv.org/abs/2406.15720.

Mahdavi, S., Liao, R., and Thrampoulidis, C. Memorization capacity of multi-head attention in transformers, 2024. URL https://arxiv.org/abs/2306.02010.

Maini, P., Jia, H., Papernot, N., and Dziedzic, A. Llm dataset inference: Did you train on my dataset?, 2024. URL https://arxiv.org/abs/2406.06443.

Mireshghallah, F., Uniyal, A., Wang, T., Evans, D., and Berg-Kirkpatrick, T. Memorization in nlp fine-tuning methods, 2022. URL https://arxiv.org/abs/2205.12506.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt, 2019. URL https://arxiv.org/abs/1912.02292.

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A. F., Ippolito, D., Choquette-Choo, C. A., Wallace, E., Tramèr, F., and Lee, K. Scalable extraction of training data from (production) language models, 2023. URL https://arxiv.org/abs/2311.17035.

Pasco, R. C. *Source coding algorithms for fast data compression*. Ph.d. dissertation, Stanford University, 1977.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Prashanth, U. S., Deng, A., O'Brien, K., V, J. S., Khan, M. A., Borkar, J., Choquette-Choo, C. A., Fuehne, J. R., Biderman, S., Ke, T., Lee, K., and Saphra, N. Recite, reconstruct, recollect: Memorization in lms as a multifaceted phenomenon, 2024. URL https://arxiv.org/abs/2406.17746.

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

Rissanen, J. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20 (3):198–203, 1976.

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model?, 2020. URL https://arxiv.org/abs/2002.08910.

Sablayrolles, A., Douze, M., Ollivier, Y., Schmid, C., and Jégou, H. White-box vs black-box: Bayes optimal strategies for membership inference, 2019. URL https://arxiv.org/abs/1908.11229.

Schwarzschild, A., Feng, Z., Maini, P., Lipton, Z. C., and Kolter, J. Z. Rethinking llm memorization through the lens of adversarial compression, 2024. URL https://arxiv.org/abs/2404.15146.

Shannon, C. E. *A Mathematical Theory of Communication*. University of Illinois Press, 1948. Reprint in 1998.

Shannon, C. E. Prediction and entropy of printed english, Sept 1950.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Tänzer, M., Ruder, S., and Rei, M. Memorisation versus generalisation in pre-trained language models, 2022. URL https://arxiv.org/abs/2105.00828.

Xia, M., Artetxe, M., Zhou, C., Lin, X. V., Pasunuru, R., Chen, D., Zettlemoyer, L., and Stoyanov, V. Training trajectories of language models across scales, 2023. URL https://arxiv.org/abs/2212.09803.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018. URL https://arxiv.org/abs/1709.01604.

Yun, C., Sra, S., and Jadbabaie, A. Small relu networks are powerful memorizers: a tight analysis of memorization capacity, 2019. URL https://arxiv.org/abs/1810.07770.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization, 2017. URL https://arxiv.org/abs/1611.03530.

Zhang, C., Ippolito, D., Lee, K., Jagielski, M., Tramèr, F., and Carlini, N. Counterfactual memorization in neural language models, 2023. URL https://arxiv.org/abs/2112.12938.

A. Appendix

A.1. Additional related work

Prior definitions of memorization. Carlini et al. (2019) defined a string m as memorized by a language model θ if the second half of m can be generated greedily when prompting the model with the first half. Following this, Nasr et al. (2023) introduced *extractable memorization*, where model θ is said to memorize m if an adversarial prompt p can be found that generates m. Mireshghallah et al. (2022) and Schwarzschild et al. (2024) refined this definition by restricting p to a certain number of tokens, preventing it from containing the entire m. However, even this definition has limitations: for example, generating the sequence "cat cat cat ... cat" with the prompt "repeat cat 1000 times" does not necessarily indicate memorization. Carlini et al. (2019) use perplexity or likelihood, one measure of the compressibility of a sequence, in an effort to distinguish highly memorized sequences from merely easy-to-compress ones. One additional definition of note is *counterfactual memorization* (Zhang et al., 2023), which measures the impact of a single datapoint on training; this can be seen as an instantiation of our definition where a different model of the same family is used as a reference model. Overall, all these works regarded memorization in terms that can be seen as forms of compression, although did not explicitly define it as such.

Finally, a concurrent work (Cohen et al., 2024) proposes a theoretical definition for memorization also relying on Kolmogorov.

Some of our findings also relate to the discovery of *double descent* in machine learning (Belkin et al., 2019; Nakkiran et al., 2019) and language modeling (Xia et al., 2023), as well as general discussions of memorization and generalization in deep learning (Zhang et al., 2017; Tänzer et al., 2022).

A.2. Compression with language models beyond arithmetic coding

Shannon (1948) noted that the optimal compression method for a given source is one that assigns codes to symbols such that the average code length approaches the entropy of the source. Arithmetic coding (Pasco, 1977; Rissanen, 1976) is known to be one optimal way to compress text given a distribution over symbols; it was used in (Delétang et al., 2024) to compress text using modern language models.

Although arithmetic coding is known to be optimal for samples generated from the random process of choice, it may still be sub-optimal for cases where the compressed samples are correlated with the choice of random process. Specifically, in language modeling, the training data is highly correlated with the model itself and hence we might need to treat them differently. For instance, we know from previous work that the models behavior on training data points is different from random samples. A large portion of training data can be generated using greedy decoding (Carlini et al., 2023b) which is a behavior not expected for a randomly sampled data. To this end, we design a new compression technique, a generalization of arithmetic coding.

Ensemble compression. Sampling from language models involve two key parameters k for top_k selection and t for temperature. We design a compression method that sets these parameters adaptively. For instance, for cases where we know we can decode the next 100 tokens in a greedy fashion, we set $k = 1$ to reduce the bit length of arithmetic code. Changing the setup of the coding scheme itself requires a new token to be injected and wastes some number of bits, but it could still be beneficial for the code length. Our compression program uses dynamic programming to find the optimal code with injection of these new tokens in the middle of the text. Notably, our algorithm runs in time $O(n * T)$, where n is the number of tokens and T is the number of possible setups (combination of t and k) that we allow.

A.3. How reliable are our linear estimates of capacity?

Instead of scaling the number of examples in a dataset, we scale model sequence length to adjust the size of a dataset. We use the following measurement for expected memorization of a model:

$$\text{mem}(X, L(X)) \approx \min(capacity(L), H(X))$$

we substitute our previous estimate of $\alpha = 3.642$ and ensure to adjust the parameter count for increases due to resizing the model's embedding matrices. We fix the number of training samples to 4096 and train a model with 2 layers and a hidden size of 128. Results are illustrated in Figure 10 and Table 3. Our predictions of total memorization are accurate, with an average error rate of 1.7% while scaling S and 1.8% when scaling V.

A.4. Additional memorization results

Our findings indicate that memorization of text data neatly plateaus near the model capacity just as in the synthetic data case. When the dataset size increases by a factor of N, the model divides its memorization between datapoints by an equal amount; the sum of memorization is measured to be constant, presumably at the upper bound of the model's capacity.

When the dataset is small enough for each model to fit – that is, below the capacity of the smallest model – we observe very similar performance between the models. For larger

	d_{emb}	n_{layer}	$\|\theta\|$	$\|D\|$	Predicted F1	Observed F1
GPT2-XL	1600	48	1,556,075,200	170,654,583	0.55	54.61 ± 1.3
				76,795,021	0.75	71.08 ± 0.4
				18,851,574	0.95	95.85 ± 0.8
GPT2-Medium	768	12	123,702,528	13,566,442	0.55	53.44 ± 1.1
				6,104,935	0.75	65.69 ± 0.6
				1,498,634	0.95	97.98 ± 0.3

Table 2. Dataset sizes that our scaling law predicts will produce a given membership inference F1, along with empirical values.

S	Params.	Memorized	Expected	Error
4	6.59×10^5	1.73×10^5	1.80×10^5	4.19
8	6.60×10^5	3.54×10^5	3.60×10^5	1.80
16	6.61×10^5	7.15×10^5	7.21×10^5	0.84
32	6.63×10^5	1.44×10^6	1.44×10^6	0.41
64	6.67×10^5	2.29×10^6	2.36×10^6	2.97
128	6.75×10^5	2.36×10^6	2.39×10^6	1.24
256	6.92×10^5	2.44×10^6	2.45×10^6	0.44

Table 3. Model capacity estimates across sequence length S, along with error (%).

V	Params.	Memorized	Expected	Error
128	4.21×10^5	1.49×10^6	1.49×10^6	0.36
512	4.71×10^5	1.71×10^6	1.67×10^6	2.78
1024	5.36×10^5	1.95×10^6	1.90×10^6	2.70
2048	6.67×10^5	2.39×10^6	2.36×10^6	1.11
4096	9.29×10^5	3.13×10^6	3.15×10^6	0.47

Table 4. Model capacity estimates across vocab size V, along with error (%).

data sizes we notice an interesting trend: unintended memorization increases with dataset size for to a point, presumably as a model fills its capacity with the available information, and then decreases, as the model replaces sample-level information with more useful, generalizable knowledge. A given model generalizes the most (and memorizes the least information about any individual sample) when the dataset is maximally large.

A.5. Comparison of distributions memorized

Distribution-level analysis. Text sequences have very different properties than uniform synthetic bitstrings. We explore how two models of equal capacity spread their memorization across datapoints. We plot a histogram (Figure 14) of train and test compression rates of training data from both synthetic random bitstrings and text. Random training data follows a very normal distribution with a small amount of overlap between train and test compression rates. Text loss is lower on average but more spread out, with low loss on some training points and a long tail of higher losses. There is much more overlap between the train and test loss distributions, which explains why membership inference is more difficult for text data.

Which datapoints are most memorized? Our distribution-level analysis indicates that unlike in the random-bitstring case, models trained on a large amount of text are able to memorize a small number of datapoints. Prior work has indicated that a large amount of this memorization can be due to duplicated training points (Lee et al., 2022) but our dataset is fully deduplicated so this cannot be an explanation in our case.

To quantitatively evaluate the number of rare words per document, we measure the TF-IDF of each training document, plotted vs. unintended memorization in Figure 15. We use the following equation for TF-IDF:

$$\text{TF-IDF}(d; \mathcal{D}) = \frac{1}{|d|} \sum_{w \in d} \log \frac{|D|}{tf(w, \mathcal{D})}$$

where $tf(d, \mathcal{D})$ indicates the total number of times word w appears in dataset $\mathcal{D}$. Intuitively, a higher TF-IDF score for document d indicates that d contains more words that are rare in $\mathcal{D}$.

We clearly observe for samples with positive unintended memorization there is a strong correlation between trainset TF-IDF and memorization: examples with more rare words are more memorized. In particular, the sample with highest TF-IDF out of the whole training dataset (a sequence of Japanese words) has the third-highest measured memorization; even though this is just one out of $260,000$ training samples, the model can regurgitate the entire sequence given just a single token (囚). Out of the top twenty memorized sequences, all but three contain sequences of tokens from other languages (Japanese, Chinese, and Hebrew).

Manual analysis (Table 5) indicates that the most memorized datapoints have extremely rare tokens, typically ones not found in English.

	Text	TFIDF	Memorization	Language
0	人気エリアであるフォンニャに位置するRock & Roll Hostelは、ビジネス出張と観光のどちらにも最適なロケーションです。	78553.72	2.98	Japanese
1	このトピックには0件の返信が含まれ、1人の参加者がいます。1 年、 6 ヶ月前に Dave Gant さんが最後の更新	71279.19	1.09	Japanese
2	Label: Living Records\nDestroy All MonstersメンバーBen Millerによる自主レーベルからのソロCD-R。こちらは付属の抽象画をサウンド化したと	68064.46	2.73	Japanese
3	《左傳》記「崔氏側莊公于北郭。丁亥，葬諸士孫之里，四翣，不	60820.46	2.89	Chinese
4	歡迎客人自備紋身圖案或要求本紋身店代客起圖， 設計起圖須	60018.53	2.16	Chinese
5	By 小森 栄治,向山 洋一\nRead Online or Download 中学の理科「総まとめ」を7日間で攻略する本 「	59625.40	2.27	Japanese
6	統合分析是將一些議題相關但彼此獨立的臨床實驗之研究結果(大	59624.37	1.73	Chinese
7	在SIA-Smaart Pro的Real-Time Module实时模块上，将功能扩展，实时显示相位和Fixed Point Per Octave（	59128.54	1.95	Chinese
8	Progress in Intelligent Transportation Systems and IoT/M2M Communications: Markets, Standardization, Technologies\n|出版日|ページ情報|英文 173 Pages\n|インテリジェント交通システムお	58953.67	0.50	Japanese
9	English Title: Kingdom Hearts: Chain of Memories\nJapanese Title: キングダム ハーツ チェイン オブ メモリーズ — “Kingdom Hearts: Chain of Memories”\nAuthor: Tomoco Kanemaki\nIllustrator: Shiro	58605.92	0.99	Japanese
10	אשכול זה הועבר לארכיון. נא לשאול שאלה חדשה אם יש לך צורך	58420.30	1.37	Hebrew
11	在《易經》里单数为阳, 双数为阴. 我曾怀疑马来西亚政府也会	58382.40	1.98	Chinese
12	「XXI c.ー21世紀人」第3回企画展 三宅一生ディレクション\n21_21 DESIGN SIGHT 第３回企画展の	57797.99	2.55	Japanese
13	无敌神马在线观看 重装机甲 睿峰影院 影院 LA幸福剧本\n时间：2020-12	57399.24	2.67	Chinese
14	季末小邪 回复 dgutkai: 楼主 您好 可以把项目源码发我吗？ 可以付	56539.93	2.46	Chinese
15	בכל סדנא אפשר לזהות את הילדים שהוריהם מאפשרים חופש יצי	56478.18	1.41	Hebrew
16	Ακαδημαϊκές Δημοσιεύσεις Μελών ΔΕΠ σε άλλα Ιδρύματα >\n	56376.74	0.75	Greek
17	Larry想和李华，还有她那些中国朋友多在一起玩儿，了解更多的中国文	56152.72	1.16	Chinese
18	Mark 5:18 wrote:καὶ ἐμβαίνοντος αὐτοῦ εἰς τὸ πλοῖον παρεκάλει αὐ	55391.28	0.19	Greek
19	בתחילה הייתי סקפטית לגבי השקעת כסף בשיווק אינטרנטי.	55014.00	1.41	Hebrew

Table 5. Highest TF-IDF training examples from a $20M$ param model trained past its capacity on 2^{16} sequences of English text. All of the highest TF-IDF examples are considered memorized, and contain text from non-English languages (Japanese, Chinese, Hebrew, and Greek).

A.6. Scaling law fit

Here we demonstrate the fit of our sigmoidal scaling law to experimental data. We show points in tokens-per-parameter vs. fit in Figure 16. Although the sigmoidal function is slightly simplistic (the points do not perfectly fit the curve) our fit produces estimates within $1 - 2\%$ of observations.

Here, we discuss other definitions of memorization.

B. Other notions of memorization

In this section we list multiple other notions of memorization and compare it with our definition. We specifically focus on why these notions do not satisfy all of our requirements.

- **Stability-based notions of memorization.** There are notions of privacy and memorization that deal with “stability” of the training algorithm to small changes in the training set. Most notably, differential privacy (Dwork, 2006) considers the worst-cast drift of the model distribution when a single data point changes. Another notion of memorization in Feldman (2020) is based on the change of the model prediction on a point x, when we add the labeled pair (x, y) to the training set of a classification/regression model. Both of these notions are crucially relying on the learning algorithm and how it behaves. Moreover, the definition of differential privacy is not ideal for our case because it is a worst-case definition and cannot be applied at sample/model level. While the notion of memorization in Feldman (2020) does not have this particular issue, it suffers from the fact that it only applies to classification models and mostly deals with the memorization of the association between the label (y) and input (x), and not the memorization of x itself. These issues make these notions not ideal for our case.

- **Extraction-based memorization.** There are multiple works in the literature (Carlini et al., 2019; Mireshghallah et al., 2022; Nasr et al., 2023; Zhang et al., 2023; Carlini et al., 2023b; Schwarzschild et al., 2024) that define memorization of samples in language models based on how easy it is to extract that sample. Specifically, when trying to understand the extent of memorization of a sample x in a model θ they measure some notion of complexity for the task of eliciting the model to output x. Although these notions are great in that they only take a model θ and a sample x, they still do not account for

generalization. Considering our running example of the following training sample: "What is 2^{100}? (A: $1, 267, 650, 600, 228, 229, 401, 496, 703, 205, 376$)", this will be identified as highly memorized by almost all of the extraction based notions of memorization. Another issue with these definitions are that they are heavily dependent on the details of decoding algorithm. This is not ideal as we do not expect the memorization of a sample x in a model θ to depend on the detailed parameters we use to generate samples using θ.

The work of (Schwarzschild et al., 2024) in this category is the closest to ours. This work which is based on prompt-optimization, optimizes a short prompt p to make the model elicit x, then it calls the sample x memorized, if length of p is less than x. Although this definition is close to our definition in using compression, it still does not account for generalization of the model. Moreover, it focuses on a specific way of compression through prompting. We posit that compression through prompting is an inferior compression scheme and can often lead to compression rates greater than 1.

- **Membership/attribute inference.** Membership inference (Shokri et al., 2017) and attribute inference attacks (Jayaraman & Evans, 2022) have been used for empirically measuring the privacy of machine learning algorithms. These notions which usually aim at approximating the stability notions of memorization are suffering from the same shortcomings. They rely heavily on the learning algorithm and the data distribution. Moreover, they fail at providing a sample level notion of memorization. For example, the obtained accuracy for membership inference attack is only meaningful in the population level. This is because various attack may have different true positives for membership, and the union of all these true positive across different attack may cover the entire training set, rendering it unusable as a sample level notion of memorization.

- **Data copying in generative models.** There are some interesting notions of memorization designed specifically for generative modeling where a generative model may output a certain portion of training samples (Bhattacharjee et al., 2023; Carlini et al., 2023a). These notions are similar to extraction based definition of memorization but they are more lenient in that they only require extraction of part of the training data. However, they still suffer from the same challenges as of extraction based definitions.

C. Proofs

In the section we provide the proofs missing from the main body.

C.1. Proof of Proposition 1

Here we prove Proposition 1

Proof. we have

$$\begin{aligned}\text{mem}_U(X, \hat{\Theta}, \Theta) &= I(X \mid \Theta, \hat{\Theta}) \\ &= I((X_1 \mid \Theta, \ldots, X_n \mid \Theta), \hat{\Theta}).\end{aligned}$$

And since the data is sampled i.i.d., all random variables in $\{R_i = [X_i \mid \Theta]\}_{i\in[n]}$ are independent. [3] So we have,

$$I((X_1 \mid \Theta, \ldots, X_n \mid \Theta), \hat{\Theta}) \geq \sum_{i\in[n]} I(X_i \mid \Theta, \hat{\Theta})$$

which implies

$$\text{mem}_U(X, \hat{\Theta}, \Theta) \geq \sum_{i\in[n]} \text{mem}_U(X_i, \hat{\Theta}, \Theta).$$

On the other hand, we have

$$\begin{aligned}\text{mem}_U(X, \hat{\Theta}, \Theta) &= I(X \mid \Theta, \hat{\Theta}) \\ &= H(\hat{\Theta} - H(\hat{\Theta} \mid (X \mid \Theta)) \\ &\leq H(\hat{\Theta})\end{aligned}$$

□

C.2. Proof of Proposition 4

Proof. We first state a Lemma about connection between algorithmic (kolmogorov) mutual information and mutual information.

Lemma 6. [Theorem 3.6 in Grunwald & Vitányi (2004)] Assume (X, Y) be a pair of joint random variables. Let f be the density function, $f(x, y) = \Pr[(X, Y) = (x, y)]$. Then we have

$$\begin{aligned}I(X, Y) - H_K(f) &\leq \mathop{\mathrm{E}}_{(x,y)\sim(X,Y)}[I_K(x, y)] \\ &\leq I(X, Y) + 2H_K(f).\end{aligned}$$

Now we use this lemma to prove the statement of the Proposition. Let f be a the density function for the joint distribution $(X_i \mid \theta, \hat{\Theta})$. That is $f_i(x_i, \hat{\theta}) = \Pr[X_i = x_i \mid \theta \text{ and } \hat{\Theta} = \hat{\theta}]$. Note that this function is independent of n and θ. By definition we have

$$\text{mem}_U(X_i, \hat{\Theta}, \theta) = I(X_i \mid \theta, \hat{\Theta}).$$

[3] Note that X_i themselves are not independent because they are sampled by first sampling an underlying model Θ. However, they are conditionally independent once the underlying model Θ is given.

Now using Lemma 6 we have

$$\begin{aligned} I(X_i \mid \theta, \hat{\Theta}) - H_K(f) &\leq \mathop{\mathrm{E}}_{x_i \sim X_i \mid \theta}[I_K(x_i, \hat{\theta})] \\ &\leq I(X_i \mid \theta, \hat{\Theta}) + 2H_K(f). \end{aligned}$$

and this concludes the statement of Proposition by setting $\epsilon = 2H_K(f)$ □

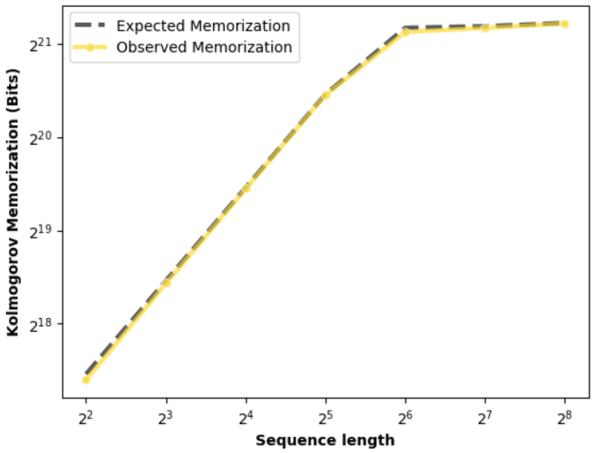

Figure 10. Model memorization across sequence lengths for a fixed-length dataset. Our predictions of total memorization are accurate, with an average error rate of 1.7%.

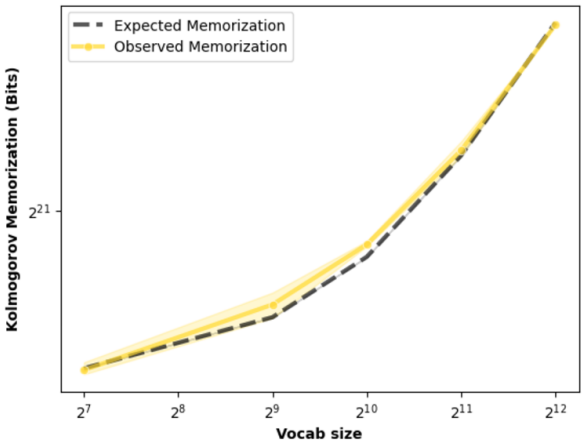

Figure 11. Model memorization across vocabulary size for a fixed-length dataset. Our predictions of total memorization are accurate, with an average error rate of 1.8%. Note that, we do not observe a capacity plateau, since increasing V also increases parameters.

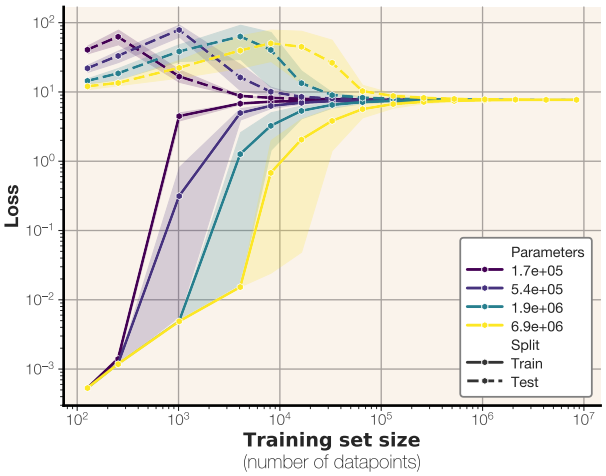

Figure 12. Train and test losses for different-sized language models trained on synthetic data.

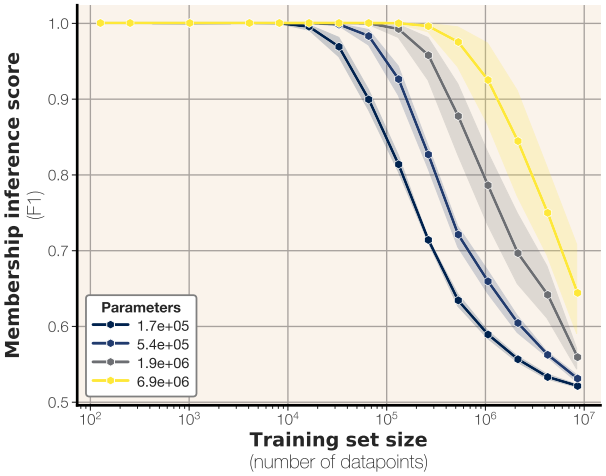

Figure 13. Membership inference attack performance decreases with dataset scale. In the case of uniform synthetic data, membership inference performance never falls below 0.54.

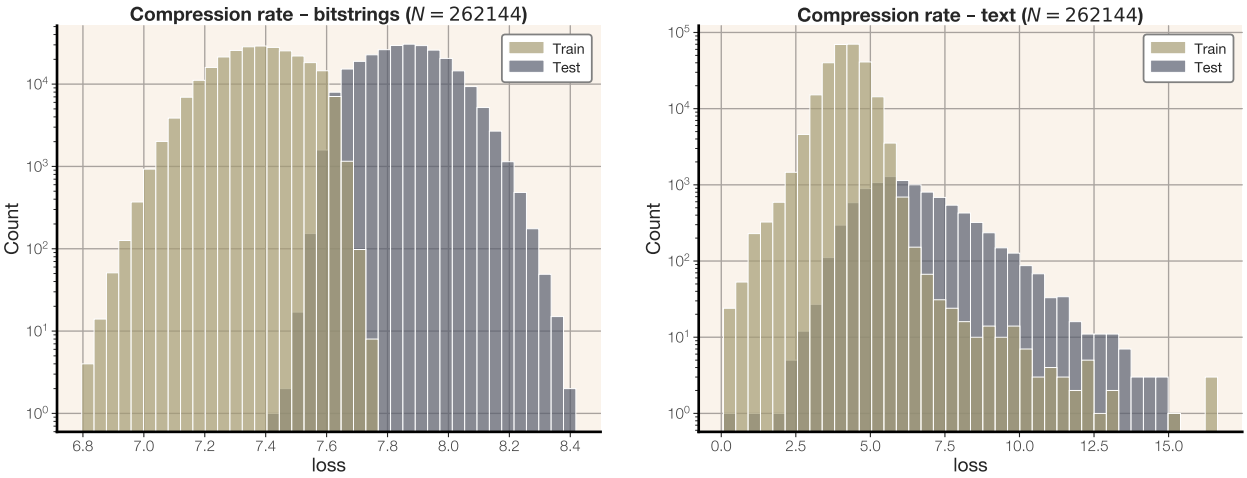

Figure 14. Distribution of compression rates for equal-sized transformers ($n_{\text{layer}} = 4$, $d_{\text{model}} = 128$) trained on 2^{14} sequences of equal-length random bitstrings (left) and text (right).

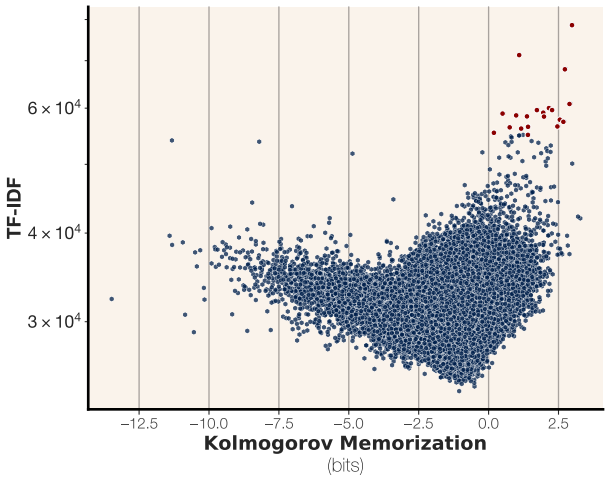

Figure 15. Unintended memorization vs. TF-IDF for all training points of a $20M$ param model trained past its capacity on 2^{16} sequences of English text. The training documents with rarest words are typically the most memorized.

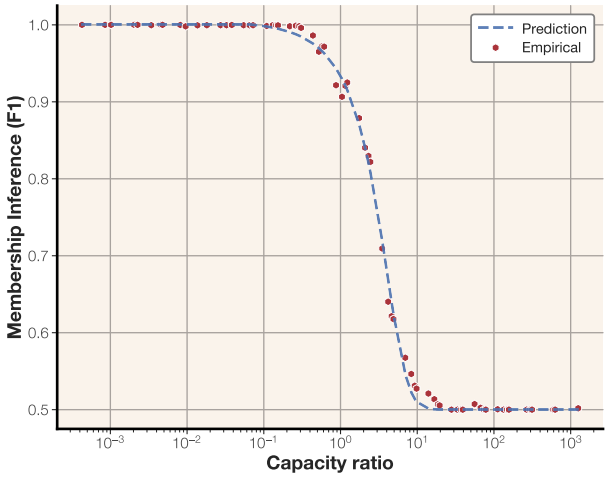

Figure 16. Our sigmoidal scaling law for membership inference fit to experimental data.