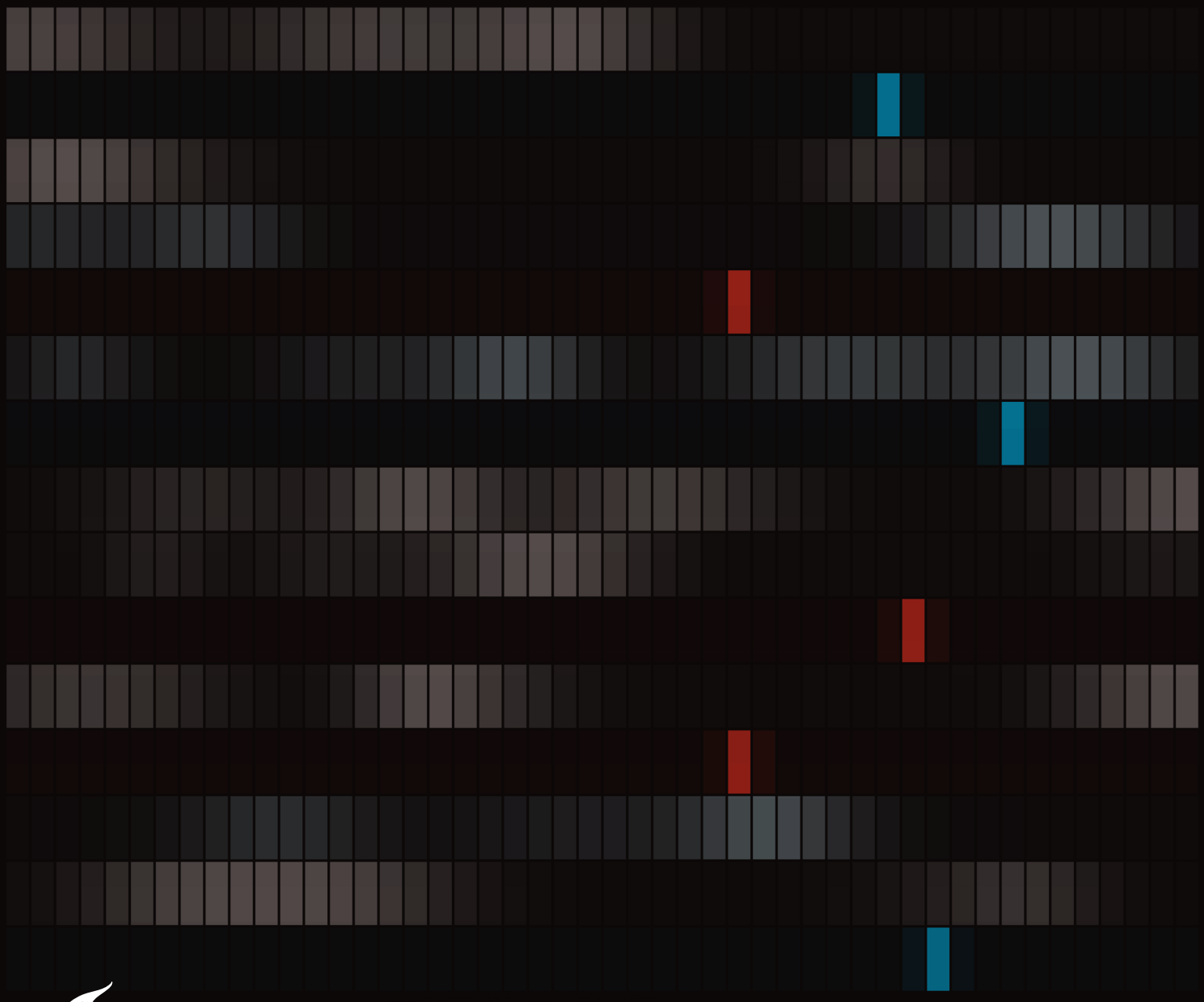
Understanding Memorization in Large Language Models

What controls memorization rate? From entropy to conditional entropy or conditioning structure.

Master Thesis in Computer Science

Rodrigo Álvarez Lucendo



TU Delft EPFL

Abstract

Large language models (LLMs) can reproduce passages from their training data verbatim, raising privacy and copyright concerns. Prior work attributes memorization to factors such as model size, sequence entropy, context length, and repetition, but these findings lack a unified explanation. This thesis proposes a disambiguation complexity framework: memorization speed is governed not by the information content of a sequence, but by the difficulty of identifying it, specifically by the complexity of the minimal conditioning structure the model must extract from context to uniquely determine the correct continuation.

We demonstrate a counterintuitive regime in which random token sequences are memorized faster than structured natural language, contradicting standard explanations. We formalize a hierarchy of conditioning levels and introduce K-arity, a scalar complexity measure counting the number of prefix tokens jointly required to make a continuation deterministic. Through controlled experiments on synthetic datasets, we show that conditioning level and K-arity are predictive of memorization behavior. Attention analysis reveals that disambiguating cues are most clearly visible in early attention patterns. Natural language experiments show that, in text rich with redundant linguistic cues, isolated manipulations of conditioning complexity do not produce detectable differences, highlighting the gap between synthetic and naturalistic settings. This single principle connects input representation, entropy, identifying tokens, and context length within a common theoretical lens.

Acknowledgements

I would like to express my sincere gratitude to my daily supervisor, Madhur Panwar, for his guidance and for giving me the opportunity to carry out my thesis at the wonderful NLP Lab at EPFL. I am also grateful to Gail Weiss for the stimulating discussions on Transformers, and to Prof. Kubilay Atasu for his insightful feedback and for facilitating the collaboration across universities. I would like to thank my parents and my sister for their unconditional love and support throughout the years. Finally, I am deeply thankful to Pavlos Makridis and Jorge Romeu Huidobro for the unforgettable journey and lasting friendship during my study years and many years to come.

Rodrigo Alvarez Lucendo
Lausanne, May 2026.

Contents

1

Introduction

The ability of large language models (LLMs) to reproduce passages from their training data verbatim raises privacy and copyright concerns. At the same time, memorization is not purely harmful: the same capacity that allows a model to reproduce training sequences is closely related to its ability to generalize from limited data, and understanding the former may shed light on the latter.

Prior work shows that memorization is influenced by factors such as model size, context length, repetition, entropy, and tokenizer vocabulary [5, 9, 2, 16, 6]. However, these findings lack a common explanation.

This work proposes a framework that unifies these observations by shifting the explanatory focus from the *information content* of a sequence to the *operational difficulty* of identifying it. By *operational difficulty* we mean the work the model must perform to map a prefix to its correct continuation: which prefix tokens it must consult, where they are located, and how many of them must be combined. This is distinct from information content (entropy), which only quantifies how unpredictable the sequence is. Two sequences with identical entropy can differ sharply in operational difficulty: in one, a single prefix token may already determine the continuation; in the other, the model may need to jointly attend to several scattered positions. chapter 4 makes this distinction formal. Specifically, we argue that memorization is driven by the complexity of the minimal conditioning structure the model must extract from the context to disambiguate the correct continuation from all other sequences in the dataset.

Contributions. The main contributions of this thesis are as follows.

1. We identify a counterintuitive regime in which random token sequences are memorized *faster* than natural language sequences, which contradicts the standard structure-versus-noise explanation and any account that predicts memorization speed solely from sequence entropy. Identifying this regime is useful because it isolates a discrepancy that the entropy-based view cannot explain (section 5.2).
2. We formalize the notion of *disambiguation complexity* and introduce a hierarchy of conditioning levels that characterizes how structurally simple or complex the cue required for sequence identification is. Together with K-arity, this hierarchy operationalizes the otherwise informal notion of difficulty introduced above (section 4.2).
3. We construct synthetic datasets with deliberately embedded conditioning structures and demonstrate, through controlled experiments, that conditioning level is predictive of memorization speed (section 5.4).
4. We introduce *K-arity* as a complementary scalar complexity measure that counts the number of prefix tokens that must be jointly consulted to make the continuation deterministic, and show experimentally that it predicts memorization ordering (section 4.4).
5. We analyze attention patterns in trained models and show that the relevant disambiguating cues are most clearly visible in early attention patterns (section 5.5).

6. We test the framework in natural language settings by inserting unique-identifier tokens, translating prefixes, and pairing semantically equivalent paraphrases with shared suffixes, and discuss why these manipulations of conditioning structure do not produce easily detectable effects against the rich background of redundant linguistic cues, suggesting directions for future investigation (chapter 6).

Outline. Chapter 2 surveys the relevant prior literature and positions this thesis within it. Chapter 3 provides the technical background on autoregressive language models, tokenization, and information-theoretic tools used throughout this work. Chapter 4 develops the disambiguation framework, introduces the conditioning-level hierarchy, and defines the K-arity complexity measure. Chapter 5 presents the full set of controlled synthetic experiments and analyzes the results. Chapter 6 reports the natural-language experiments and discusses why the framework's predictions are difficult to isolate in this setting. Chapter 7 summarizes the findings, discusses limitations, and outlines future directions.

2

Related Work

Memorization in large language models has been studied from several angles: tokenization, sequence complexity, contextual cues, model capacity, and internal mechanisms. These perspectives have largely developed in isolation. In this chapter we review the most relevant prior work and draw connections among these perspectives before showing how our contribution ties them together under a unified view. Formal definitions of the information-theoretic concepts referenced below (entropy, conditional entropy, Kolmogorov complexity) are given in chapter 3.

Memorization depends on input representation. Kharitonov, Baroni, and Hupkes [6] show that the size of the tokenizer vocabulary learned by byte-pair encoding (BPE) significantly affects memorization behavior in transformers. Larger vocabularies shorten tokenized sequences, which the authors argue simplifies the attention computation, and they observe that this makes models more prone to reproducing training data verbatim. The key implication is that memorization is shaped not only by the semantic content of the data but by the *effective representation*: specifically, by how much information each token carries. This observation resonates with results from the entropy and complexity literature discussed next, where per-token information content also emerges as a key factor.

Sequence complexity predicts memorization, but which notion of complexity? Several studies link memorization difficulty to information-theoretic or complexity-based measures. Speicher et al. [14] train models on random strings and find that higher-entropy strings are memorized faster, while also observing that models can often recall memorized tokens from relatively small subsets of the full context. Huang et al. [5] report a linear relationship between data entropy and memorization score (measured via Levenshtein distance) in large language models, implying that lower-entropy data are at greater risk of exact memorization and leakage. Duan et al. [4] arrive at a similar conclusion using z-complexity, a computable approximation of Kolmogorov complexity: strings with lower z-complexity exhibit lower KL-Levenshtein distance and are thus more easily memorized.

Notably, these results point in seemingly opposite directions: Speicher et al. [14] find that *higher*-entropy strings are memorized *faster*, while Huang et al. [5] and Duan et al. [4] report that *lower*-entropy data are at greater risk of exact memorization. This tension suggests that entropy alone is an incomplete predictor, and that a more structural account is needed. More broadly, all three studies rely on scalar measures of overall sequence unpredictability and do not address *how* the model uses context to identify the correct continuation. The observation by Speicher et al. [14] that small context subsets often suffice for recall is especially suggestive: it implies that memorization may depend less on total entropy and more on the structure of the context, a point that connects naturally to the following line of work on identifying specific tokens in the context that correlate with memorization.

Some tokens act as sequence identifiers. Tirumala et al. [16] analyze memorization rates across parts of speech and find that nouns and numbers are memorized substantially faster than other categories. They hypothesize that these tokens function as unique identifiers for their respective training examples,

and support this interpretation by studying settings in which an explicit unique identifier is present. Stoehr et al. [15] provide a complementary perspective: they show that memorization in GPT-Neo 125M is often concentrated around a small number of distinctive or rare prefix tokens, and that these tokens are predominantly processed by a specific attention head. Additionally, Morris et al. [9] find a strong correlation between training-set TF-IDF and memorization score: TF-IDF, or term frequency–inverse document frequency, assigns high weight to tokens that are frequent inside a particular example but rare in the overall corpus; in this setting it is therefore a proxy for how distinctive or identifying a token is, and sequences containing rare tokens are memorized disproportionately. The sample with the highest TF-IDF in their entire training set (a sequence of Japanese words) has the third-highest measured memorization out of 260,000 training samples, with the model able to regurgitate the entire sequence from a single token.

Together, these results offer a more mechanistic picture than entropy-based accounts alone. Memorization can hinge on a few highly informative cues rather than on the full input sequence. Building on these findings, our own working interpretation, made explicit in chapter 4 and tested empirically in chapter 5, is that the model implements memorization partly as a retrieval process: first identifying a distinctive token, then using it to reconstruct the continuation. While this retrieval view is suggested by the prior empirical observations cited above, treating it as the operational mechanism underlying memorization speed is a hypothesis we adopt and test in this thesis, not a conclusion already established in the literature. This view is consistent with the previously mentioned observation by Speicher et al. [14] that small context subsets suffice, and it refines the entropy perspective: what may matter is not the overall unpredictability of the sequence, but whether it contains tokens that are informative enough to serve as identifiers.

Induction heads as a retrieval mechanism. Olsson et al. [10] identify *induction heads*, attention heads that implement a prefix-matching and copying operation, as a key circuit for in-context learning in transformers. An induction head scans the context for a previous occurrence of the current token and copies what followed it, effectively performing nearest-neighbor retrieval within the sequence. This mechanism is directly relevant to memorization: once a model has stored a training example, the act of recalling it from a partial prompt is structurally the same operation, locating a matching prefix and reproducing the learned continuation. In our attention analysis (section 5.5), we observe that the first transformer layer appears to locate disambiguating cues in the prefix. We use induction heads here only as an analogy for the cue-locating part of retrieval: the observation does not imply that an induction head alone produces the memorized output, nor that later layers simply read out a completed continuation.

Context length and exposure amplify memorization. Carlini et al. [2] quantify the extent to which language models emit memorized training data and identify approximately log-linear relationships between memorization and model capacity, the number of times an example is duplicated in training, and the number of context tokens provided in the prompt. Their finding that more context increases the likelihood of verbatim memorization is consistent with the identifier view: a longer prompt supplies more potential disambiguating information, increasing the chance that the model encounters a token or subsequence that uniquely identifies the training example. Lee et al. [7] show that removing near-duplicate training examples substantially reduces verbatim memorization, confirming that repeated exposure is a primary driver and that deduplication is an effective practical mitigation. However, neither study addresses *which* parts of the context are necessary, a question that the identifier-based results of Tirumala et al. [16] and Stoehr et al. [15] suggest has a non-trivial answer.

Model capacity determines how much can be memorized, not what is memorized first. Morris et al. [9] train over 100 GPT-style models (500K–1.5B parameters) and estimate model capacity at approximately 3.6–3.8 bits per parameter. This means a 1B-parameter model has roughly 450 MB of effective memorization storage. They argue that models continue memorizing until this capacity is saturated, after which they begin to generalize. Using a controlled synthetic setting with (name, attribute, value) knowledge tuples, Allen-Zhu and Li [1] arrive at a similar conclusion but with a much lower estimate for model capacity at 2 bits per parameter. This capacity perspective is complementary to the structural analysis of individual sequences: capacity sets the total volume of information a model

can store, while conditioning structure determines the order in which sequences are memorized.

Memorization as compression. As established in section 3.5, language modeling is equivalent to lossless compression: a model's negative log-likelihood on a sequence is its arithmetic code length. The intuition is that a model assigning probability p to the next token can be turned, via arithmetic coding, into a lossless code of length approximately $-\log_2 p$ bits for that token. A better language model is therefore also a better compressor, as made explicit by Delétang et al. [3] and by LLMZip [17]. Morris et al. [9] exploit this equivalence to define instance-level memorization as the difference in code length between a reference model and the trained model (Equation 3.11), a quantity computable directly from the two models' likelihoods without running an explicit compression algorithm. Memorization of a specific example thus corresponds to the trained model compressing that example beyond what the reference model would predict.

Under this view, earlier findings have a compression interpretation, but compression alone does not specify which contextual cue the model should use. Low-complexity sequences can require fewer bits to encode, while rare or high-TF-IDF tokens can create large code-length gains when stored. Our framework complements this by asking where the compressive gain comes from: which prefix tokens make the continuation identifiable, and how difficult it is for the model to retrieve and combine them.

Not all memorization is the same. Zhang et al. [19] introduce *counterfactual memorization*, which measures how much a model's output for a given example changes when that example is removed from the training set. This distinguishes memorization that arises from generalization (many similar examples would produce the same output) from memorization that is truly specific to an individual training example.

Positioning of the present work. The works reviewed above make multiple observations about memorization in large language models. Memorization is shaped by per-token information density [6], correlates with entropy and string complexity [14, 5, 4], can hinge on a small number of identifying tokens [16, 15], retrieved via dedicated attention circuits [10], and is amplified by context length and repeated exposure [2, 7] within the limits set by model capacity [9]. What is missing is a unifying framework that explains *why* these factors matter in terms of the operational problem the model faces: predicting the next token from its context. Without such a framework, the individual findings remain isolated empirical regularities: practitioners cannot predict which data are at risk of memorization from first principles, and mitigation strategies (deduplication, differential privacy, capacity scaling) lack a common theoretical basis for comparison. This thesis proposes such a framework, connecting the disparate findings above through a single principle: entropy matters because it correlates with how quickly prefixes become unique (section 4.1); identifying tokens matter because they reduce the required conditioning set to a single position (section 4.2); and context length matters because it determines how much disambiguating information is available. By constructing synthetic datasets with controlled conditioning structures (section 4.3) and comparing memorization dynamics across a hierarchy of conditioning levels (section 5.4), we provide direct experimental evidence for this account and delineate its limits when applied to natural language (chapter 6).

3

Background

This chapter introduces the technical concepts and notation used throughout this work. We cover autoregressive language models and their training objective, tokenization and the BPE algorithm, how memorization is described and measured, and the information-theoretic quantities that underpin our framework.

3.1. Autoregressive Language Models

A language model defines a probability distribution over sequences of discrete tokens. Given a vocabulary $\mathcal{V}$ and a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ with $x_t \in \mathcal{V}$, an autoregressive language model factorises the joint distribution as a product of conditionals:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_{<t}), \tag{3.1}$$

where $x_{<t} = (x_1, \ldots, x_{t-1})$ is the prefix up to position t. The model is trained by minimising the average negative log-likelihood (cross-entropy loss) over a corpus $\mathcal{D}$:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}). \tag{3.2}$$

3.1.1. Transformer Architecture

Modern large language models are almost universally based on the Transformer architecture [18], which processes a sequence through a stack of L identical blocks, each consisting of a multi-head self-attention sublayer followed by a position-wise feed-forward network. Self-attention allows each position to attend to all previous positions, enabling the model to capture long-range dependencies without the sequential bottleneck of recurrent networks.

Each token is first mapped to a dense vector via a learned token embedding, and a learned *positional embedding* is added so that the model can distinguish positions within a sequence. Concretely, the input representation at position t is $\mathbf{h}_t^{(0)} = \mathbf{e}_{x_t} + \mathbf{p}_t$, where $\mathbf{e}_{x_t}$ is the token embedding and $\mathbf{p}_t$ is the learned positional embedding. This is the mechanism through which the model has access to absolute position information, a quantity that plays a central role in our conditioning-level hierarchy (section 4.2).

In self-attention, the input sequence is projected into query (Q), key (K), and value (V) matrices. For a single attention head h, the output is computed as

$$\text{head}_h = \text{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_k}}\right) V_h, \tag{3.3}$$

where $Q_h = XW_h^Q$, $K_h = XW_h^K$, $V_h = XW_h^V$ are head-specific projections and d_k is the dimension of the key vectors. In multi-head attention, the outputs of H parallel heads are concatenated and projected:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\, W^O, \tag{3.4}$$

where W^O is a learned output projection. This allows different heads to attend to different positions simultaneously, which is directly relevant to our attention analysis in section 5.5, where we report the maximum attention weight across heads.

In the autoregressive (causal) setting, a triangular mask is applied so that position t can only attend to positions $1, \ldots, t$, ensuring the model cannot access future tokens during training or inference.

Each transformer block also contains a position-wise *feed-forward network* (FFN) applied independently to each position:

$$\text{FFN}(\mathbf{h}) = W_2\, \sigma(W_1 \mathbf{h} + \mathbf{b}_1) + \mathbf{b}_2, \tag{3.5}$$

where σ is a nonlinear activation (GELU in GPT-2). The attention sublayer and FFN are each followed by residual connections and layer normalisation.

We use GPT-2-style models [12] throughout this work, which are decoder-only transformers trained with the causal language-modeling objective in equation (3.2).

3.1.2. Greedy Decoding

At evaluation time, we generate sequences by *greedy decoding*: at each step, the model selects the token with the highest predicted probability:

$$\hat{x}_t = \arg\max_{v \in \mathcal{V}} \; p_\theta(v \mid x_{<t}). \tag{3.6}$$

Greedy decoding is the natural choice for measuring verbatim memorization, as it produces the single most likely continuation and avoids the stochasticity introduced by temperature-based sampling.

3.2. Tokenization and Byte-Pair Encoding

Raw text must be converted into a sequence of discrete tokens before it can be processed by a language model. The most widely used algorithm for this is Byte-Pair Encoding (BPE) [13], which constructs a vocabulary by iteratively merging the most frequent pair of bytes in the training corpus. The resulting vocabulary consists of both individual characters and frequently co-occurring subword units.

The choice of vocabulary size $|\mathcal{V}|$ has a direct effect on the average information carried per token. With a larger vocabulary, common subwords and even whole words become single tokens, so each token encodes more contextual meaning on average. Conversely, a smaller vocabulary forces the model to represent the same information using more tokens, each carrying less individual identifying power. As discussed in chapter 2, Kharitonov, Baroni, and Hupkes [6] show that this has measurable consequences for memorization behavior, a phenomenon we revisit in section 5.2.

In our synthetic experiments (chapter 5), we control vocabulary size directly: tokens are sampled uniformly from vocabularies of predetermined size V, which allows us to vary per-token information content without confounds from natural language statistics.

3.3. Memorization in Language Models

We adopt the notion of *extractable memorization* [2]: a string s is memorized if a prefix p of length k exists such that $[p \,\|\, s]$ appears in training and the model reproduces s from p via greedy decoding.

We use a strict *exact-match* criterion: a sequence is memorized iff its full suffix is reproduced without error. Formally,

$$\text{ExactMatch} = \begin{cases} 1 & \text{if } \hat{x}_{\geq k} = x_{\geq k}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

We report the dataset average; a score of 1 indicates full dataset memorization.

3.4. Information-Theoretic Preliminaries

We briefly recall the information-theoretic quantities used throughout this work.

3.4.1. Entropy

For a discrete random variable X with distribution p, the *Shannon entropy* is

$$H(X) = -\sum_{x} p(x) \log_2 p(x). \tag{3.8}$$

Entropy measures how much *information* each token carries, i.e., how uncertain the next token is: more possible outcomes imply more information per token. When tokens are drawn uniformly from a vocabulary of size V, each token carries $\log_2 V$ bits of information, so $H(X) = \log_2 V$. Thus, entropy directly reflects the information content of the sequence, with larger vocabularies yielding more information per token.

3.4.2. Conditional Entropy

For two jointly distributed random variables X and Y, the *conditional entropy* is

$$H(X \mid Y) = -\sum_{y} p(y) \sum_{x} p(x \mid y) \log_2 p(x \mid y). \tag{3.9}$$

It measures the remaining uncertainty about X once Y is known. In the language-modeling context, the quantity $H(x_t \mid x_{<t})$ captures how uncertain the next token is given the entire preceding context. If the context uniquely determines the next token, this equals zero; if multiple continuations are plausible, it is positive.

3.4.3. Kolmogorov Complexity and z-Complexity

The *Kolmogorov complexity* $K(s)$ of a string s is the length of the shortest program that outputs s. It is uncomputable in general, but any lossless compressor provides an upper bound, since the compressed output is itself a valid description of s.

The *z-complexity* of a string, introduced by Duan et al. [4], is the standard *data compression ratio* reinterpreted as a complexity proxy. It is defined as the ratio of compressed length to original length:

$$z\text{-complexity}(s) \;=\; \frac{|\text{compress}(s)|}{|s|}, \tag{3.10}$$

where compression is performed with the `zlib` deflate algorithm. The ratio lies in $(0, 1]$: values near zero indicate highly repetitive, structured strings, while values near one indicate incompressible strings. The novelty of Duan et al. [4] lies not in the ratio itself but in connecting it to memorization in language models. Because it is a computable upper bound on the normalized Kolmogorov complexity, z-complexity serves as a practical measure of intrinsic string complexity and is used in chapter 2 to relate sequence structure to memorization.

3.5. Compression and Language Modeling

Language modeling and lossless compression are equivalent: an autoregressive model assigning $p_\theta(x_t \mid x_{<t})$ implicitly defines a code with expected length equal to the cross-entropy, so lower cross-entropy means better compression [3]. Via *arithmetic coding*, the model's negative log-likelihood directly gives the compressed code length of any sequence [17].

This makes it possible to define memorization as a difference in code lengths. Given trained model $\hat{\theta}$ and a reference model θ_{ref} capturing population-level statistics:

$$\text{mem}(\mathbf{x}) \;=\; \big[-\log p_{\theta_{\text{ref}}}(\mathbf{x})\big] \;-\; \big[-\log p_{\hat{\theta}}(\mathbf{x})\big]. \tag{3.11}$$

A positive value means the trained model compresses $\mathbf{x}$ beyond what population-level regularities predict, signaling example-specific memorization [9].

Connection to our framework. Both z-complexity and mem($\mathbf{x}$) are scalar summaries of an entire sequence: they say *how compressible* it is, but not *where* inside the sequence the structure that drives that compressibility lives. Our framework addresses this structural gap. A low-z-complexity sequence can still be slow to memorize if its predictable bits are hard to compute by the model, and a high-z-complexity (near-random) sequence can be fast if a single prefix token already pins down its continuation. A natural follow-up experiment would be to compute both metrics on our synthetic conditioning-level datasets and check whether they recover the Level 0 < Level 1 < Level 2 ordering: if they do, the existing literature already implicitly captures most of what our framework adds; if they don't, that's direct evidence that disambiguation structure is a separate axis.

4

A Disambiguation Framework for Memorization

In this chapter we develop the theoretical framework that underlies all subsequent experiments. We begin by showing why entropy alone is insufficient to explain memorization speed (section 4.1), introduce a hierarchy of conditioning levels (section 4.2), describe how synthetic datasets are constructed to instantiate each level (section 4.3), and finally present K-arity as a complementary complexity measure (section 4.4).

4.1. From Entropy to Disambiguation

A natural starting point for explaining differences in memorization speed according to data type is to quantify the randomness of the data through its entropy. For tokens sampled uniformly from a vocabulary of size V, per-token entropy simplifies to

$$H = \log_2 V, \tag{4.1}$$

which equals the maximum possible entropy. A natural hypothesis, supported by prior work [14, 5], is that the entropy of a sequence correlates with how fast it is memorized by a model.

However, this view overlooks how autoregressive models actually learn. These models are trained to approximate a mapping from contexts to next tokens. From this perspective, memorization is easiest when the model can learn a **simple and consistent mapping** from a prefix to its continuation. What matters, then, is not only how much information the sequence contains, but whether the prefix provides enough information to *disambiguate* the correct continuation from all other sequences in the dataset.

A sequence may have high entropy due to a large vocabulary (Equation 4.1), yet still admit a simple mapping if a small number of its prefix tokens uniquely identify the correct continuation. In this case, the model only needs to extract a limited amount of information to make deterministic predictions. Conversely, if multiple sequences share the same relevant prefix tokens but diverge later, the mapping from prefix to next token becomes **ambiguous**: the model encounters identical contexts paired with different next tokens, which prevents it from learning a deterministic rule and therefore prevents it from memorizing those sequences.

The extent of this ambiguity depends on vocabulary size. In low-vocabulary regimes, individual tokens carry less information, so longer contexts may be required before a sequence becomes uniquely identifiable. This increases the complexity of the mapping the model must learn. In high-vocabulary regimes, individual tokens are more informative, and shorter prefixes may suffice to uniquely determine the continuation, leading to simpler mappings and faster memorization.

Per-token disambiguating power. We use the term *per-token disambiguating power* to describe how useful an individual prefix token is as a clue for locating the correct training sequence. A token has high disambiguating power when few other sequences contain the same token in the same position: after observing it, the model can rule out many alternatives. A token has low disambiguating power when it is shared by many sequences, because the model is still left with many possible continuations.

This is why vocabulary size matters in our synthetic settings. With a larger vocabulary, random tokens collide less often across examples, so a single observed token is more likely to be a distinctive clue. With a smaller vocabulary, many examples reuse the same token values, so the model usually needs a longer prefix or a more structured combination of cues before the continuation becomes unambiguous. Vocabulary size is therefore not the quantity we ultimately care about; it is one convenient way of changing how distinctive the available cues are.

To formalize this intuition, we move from measuring the overall randomness of a sequence to measuring the *remaining uncertainty* about the next token given a context. In our setting, the relevant quantity is the conditional entropy $H(x_t \mid x_{<t})$, which measures how uncertain token x_t is given the entire preceding context. If the prefix uniquely identifies the sequence, then $H(x_t \mid x_{<t}) = 0$, meaning the model can in principle make deterministic predictions, which corresponds to perfect memorization.

This leads to the central idea of this work:

> *Memorization depends not only on how much information a dataset contains, but on how easily the relevant disambiguating information can be extracted from the context.*

4.2. A Hierarchy of Conditioning Levels

While $H(x_t \mid x_{<t}) = 0$ is a necessary condition for perfect memorization, it does not explain differences in memorization *speed*: this condition holds for any dataset in which every sequence has a unique prefix.

The key distinction is *which parts* of the context are required and *how* the disambiguating information must be extracted. In practice, the model may identify and exploit a subset of context tokens that uniquely determines the correct continuation. Figure 4.1 provides a schematic illustration: a relatively small subset of the prefix may suffice to disambiguate the sequence and determine all subsequent tokens, but our focus is the minimal such subset.

Formally, for each position t, there may exist a subset $C_t \subseteq x_{<t}$ such that

$$H(x_t \mid C_t) = 0, \tag{4.2}$$

meaning that C_t alone contains enough information to uniquely determine x_t. Different datasets may require conditioning sets with very different structures: in some cases, the relevant information is local and easy to access (e.g., the previous token together with the position), while in others it may depend on distant or variable positions, or require computing a more complex function of the context.

To capture this variation, we introduce a hierarchy of conditioning levels, summarized in Table 4.1, that characterizes the complexity of the minimal structure required for deterministic prediction. Lower levels correspond to simple, local, or fixed conditioning cues; higher levels require the model to access more context or to compute a more complex mapping.

This hierarchy leads to the following central hypothesis.

Hypothesis 1 *The memorization speed of a dataset is governed by the complexity of the minimal conditioning structure required to uniquely identify the correct continuation. Datasets whose continuations can be determined from simpler conditioning structures are memorized faster.*

Level	Conditioning Set	Key Idea
0	$x_{t-1},\ t$	Position-aware bigram
1	$x_{t-1},\ t,\ x_i$	Fixed prefix hint
2	$x_{t-1},\ t,\ x_{f(t)}$	Position-dependent hint
p	$x_{<p}$	Full prefix

Table 4.1: Hierarchy of conditioning levels characterizing the minimal structure needed to deterministically predict the next token. Higher levels require access to more information or a more complex operation to recover the relevant cue.

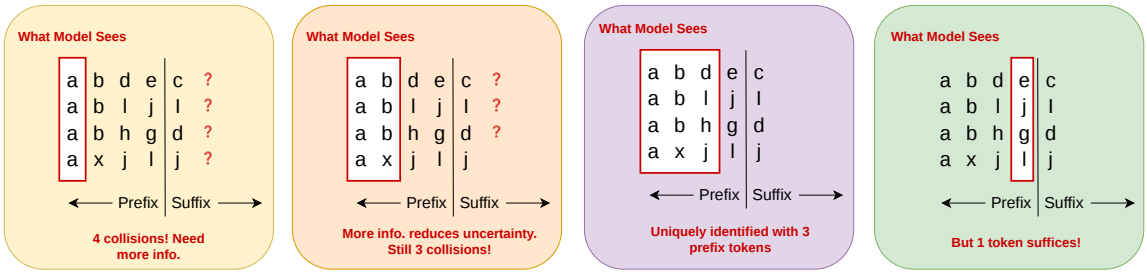


Figure 4.1: Illustration of how a subset of prefix tokens can suffice to uniquely disambiguate a sequence and determine its continuation.

4.3. Synthetic Datasets with Embedded Conditioning Rules

To test Hypothesis 1, we construct synthetic datasets in which a specific conditioning structure is deliberately embedded. Each dataset consists of a random prefix and a generated suffix. Suffix tokens are produced via a lookup table whose keys correspond to the relevant conditioning variables (as specified by the desired level in Table 4.1). The first time a particular key combination is encountered, its output token is drawn uniformly at random from the vocabulary and stored; every subsequent occurrence of the same key returns the same stored token. This ensures the mapping is deterministic yet arbitrary.

During training, the model sees the full sequence (prefix followed by suffix), but the loss is computed only over the suffix tokens; prefix positions are masked out of the loss. The model therefore receives the prefix as context but is only trained to predict the suffix. At evaluation time, the prefix is provided as a prompt and the model generates a continuation via greedy decoding, which is compared to the ground-truth suffix.

Figures 4.2 and 4.3 illustrate datasets constructed with embedded Level 0 and Level 2 conditions, respectively.

4.3.1. Level 0: Position-Aware Bigram

Level 0 is the simplest conditioning structure: each suffix token is determined solely by the preceding token and the current position. The dataset is generated via a random lookup table

$$x_t = \mathcal{T}(x_{t-1},\ t), \tag{4.3}$$

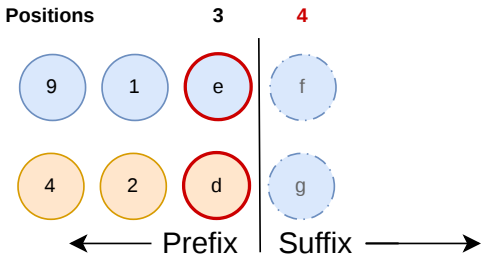
where $\mathcal{T} : \mathcal{V} \times \mathbb{N} \to \mathcal{V}$ maps each pair to a token drawn uniformly at random (sampled once and fixed). No prefix information is required; the model can predict every suffix token from local context alone.

4.3.2. Level 1: Fixed Prefix Hint

Level 1 extends Level 0 by conditioning on one additional prefix token at a *fixed* position i:

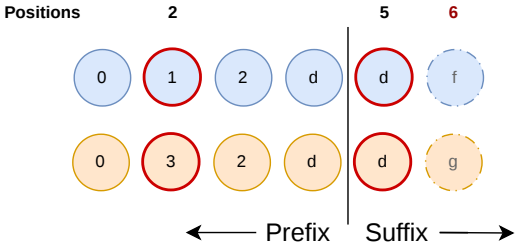
$$x_t = \mathcal{T}(x_{t-1},\ t,\ x_i), \tag{4.4}$$

where $\mathcal{T} : \mathcal{V} \times \mathbb{N} \times \mathcal{V} \to \mathcal{V}$. Because i is the same for all suffix positions, the model can disambiguate the sequence by attending to a single, known prefix location. The difficulty relative to Level 0 is that the



t, x_{t-1}	x_t
(4, e)	f
(4, d)	g

Figure 4.2: Example of a synthetic dataset with an embedded Level-0 conditioning rule. Each suffix token is determined by the preceding token and the current position.



t	$f(t)$
5	2
6	2

$t, x_{t-1}, x_{f(t)}$	x_t
(5, d, 1)	d
(5, d, 3)	d
(6, d, 1)	f
(6, d, 3)	g

Figure 4.3: Example of a synthetic dataset with an embedded Level-2 conditioning rule. Each suffix token is determined by the preceding token, the current position, and a prefix token at a position-dependent location.

model must now learn to retrieve information from the prefix rather than relying on local suffix context alone.

4.3.3. Level 2: Position-Dependent Hint

Level 2 captures sequences where each suffix token x_t is deterministic given the previous token, the position, and a single *hint* token drawn from an earlier position selected by a position-dependent function f:

$$H(x_t \mid x_{t-1},\, t,\, x_{f(t)}) = 0, \quad \text{for some } f(t) < t. \tag{4.5}$$

Again, the dataset is generated via a random lookup table $\mathcal{T}$:

$$x_t = \mathcal{T}\big(x_{t-1},\, t,\, x_{f(t)}\big), \tag{4.6}$$

where $\mathcal{T} : \mathcal{V} \times \mathbb{N} \times \mathcal{V} \to \mathcal{V}$ maps each distinct triple $(x_{t-1}, t, x_{f(t)})$ to a token drawn uniformly at random from the vocabulary $\mathcal{V}$ (sampled once and fixed). This ensures the mapping is fully arbitrary while remaining perfectly deterministic given the three inputs.

Hint-position strategies. We study five choices of $f(t)$, summarized in Table 4.2 and illustrated in Figure 4.4. These strategies span a spectrum from highly structured (lookback, modulo) to essentially unstructured (random), allowing us to assess whether the regularity of f affects how quickly a model memorizes Level 2 sequences.

4.3.4. Level Verification

A critical aspect of our experimental design is ensuring that each generated dataset satisfies its *target* conditioning level but does *not* trivially satisfy a simpler level. After generation, we run an independent verification pipeline that tests every dataset against *all* levels in the hierarchy (0, 1, 2, and the full-prefix

Table 4.2: Level 2 strategies and their position functions $f(t)$.

Strategy	Position function $f(t)$	Parameters
Lookback	$\max(0,\ t - L)$	$L = 2$
Modulo	$t \bmod M$	$M = 10$
Sinusoidal	$\lfloor(\sin(t) + 1)/2 \cdot t\rfloor$	—
Random	$\pi(t)$, random fixed mapping per t	—
Polynomial-prime	$(\sum_{k=1}^{d} t^k) \bmod p \bmod t$	$d = 3,\ p = 1009$

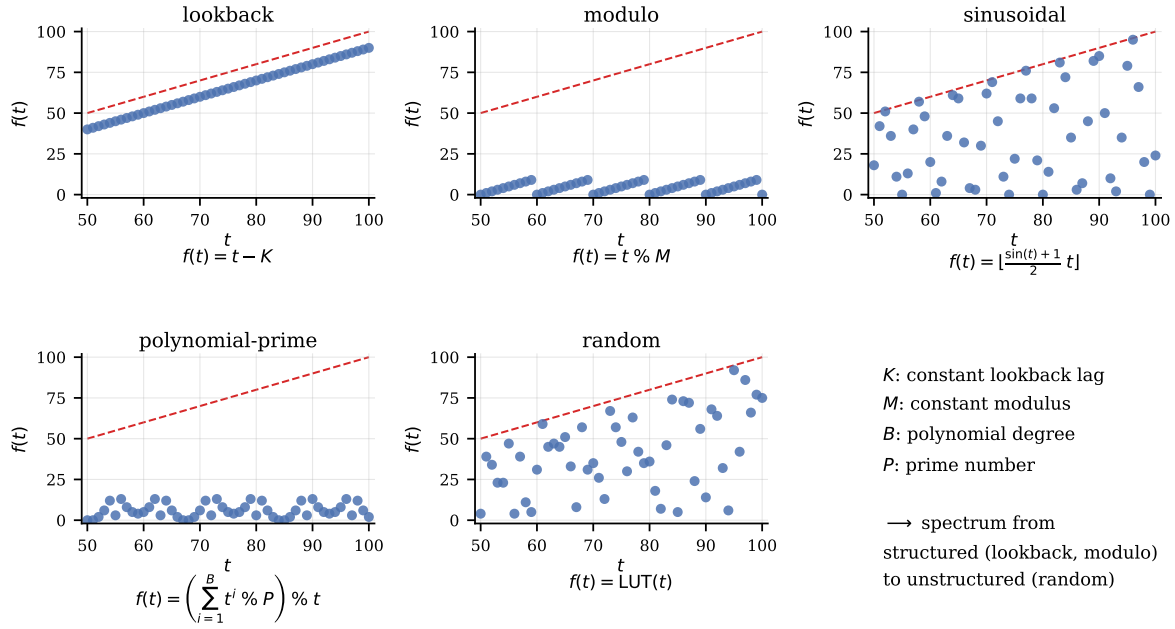


Figure 4.4: The five hint-position strategies for Level 2 datasets. Each plot shows the hint position $f(t)$ (blue dots) as a function of the current position t, with the diagonal (red dashed) indicating the current position as a reference. The strategies range from the fully structured lookback and modulo rules to the structureless random mapping, spanning a spectrum of long-range dependency patterns.

level p) by checking whether the conditional entropy $H(x_t \mid \text{conditions})$ equals zero for all suffix positions. The levels form a strict hierarchy: Level 0 $\subset$ Level 1 $\subset$ Level 2 $\subset$ Level p. A dataset generated at Level 2 must fail Level 0 and Level 1 checks; otherwise the generator has introduced unintended shortcuts. Any violation causes the affected dataset to be discarded and regenerated.

In practice, the check is straightforward: for each level, we group all suffix positions across the dataset by their conditioning-set values and verify that every group maps to exactly one distinct next token. If any group contains two or more different next tokens, the conditional entropy is positive and the dataset does not satisfy that level.

4.4. K-Arity: An Alternative Complexity Hierarchy

The conditioning-level hierarchy introduced in section 4.2 characterizes complexity by the *structure* of the disambiguating rule: whether the relevant cue is local, at a fixed position, or at a position-dependent location. Here we consider a complementary axis: the *number* of prefix tokens that must be jointly observed to make the next token deterministic.

Definition. The K-arity hierarchy introduced here is, to our knowledge, novel as a memorization-difficulty measure in language models. Borrowing from logic, where the *arity* of a function is the number of arguments it takes, we use arity as a controllable scalar for the conditioning structure of an autoregressive sequence. We say a dataset has *K-arity K* if, for every suffix position t, the token x_t is

uniquely determined by exactly K prefix tokens at positions $\varphi_1(t), \ldots, \varphi_K(t)$, together with the position index t:

$$H\Big(x_t \mid x_{\varphi_1(t)}, \ldots, x_{\varphi_K(t)}, t\Big) = 0, \tag{4.7}$$

where each $\varphi_k(t)$ maps suffix position t to a distinct prefix position. No strict subset of the K positions suffices. This yields a nested hierarchy: $K = 1$ is the simplest case, and the difficulty of disambiguation increases with K.

Dataset construction. Each dataset is built from a random prefix of length L_{prefix} drawn uniformly from a vocabulary of size V. For each suffix position t, a set of K distinct prefix positions is sampled via a random mapping $(\varphi_1(t), \ldots, \varphi_K(t))$, where $\varphi_k(t) \in \{0, \ldots, L_{\text{prefix}} - 1\}$. A lookup table then assigns a deterministic output token to every combination of K prefix token values and position:

$$x_t = \text{LUT}\big[x_{\varphi_1(t)}, \ldots, x_{\varphi_K(t)}, t\big]. \tag{4.8}$$

Prefix length, suffix length, vocabulary size, and number of training sequences are held constant across conditions, so that only K varies.

Toward a scalar metric for conditioning complexity. The K-arity hierarchy assigns a single integer K to an entire dataset, but real sequences may exhibit mixed arity across positions: some suffix tokens may be determined by a single prefix token, others only by two or three jointly. A scalar metric that captures this heterogeneity could predict memorization speed more precisely than K alone. Two natural candidates are

$$\mathcal{M}_\Sigma = \sum_t K(t), \tag{4.9}$$

$$\mathcal{M}_{\max} = \max_t K(t), \tag{4.10}$$

where $K(t)$ is the minimal number of prefix tokens required to make position t deterministic. $\mathcal{M}_\Sigma$ reflects aggregate conditioning complexity, while $\mathcal{M}_{\max}$ is governed by the single hardest position. We highlight sum and maximum because they bracket two qualitatively different views of difficulty: $\mathcal{M}_\Sigma$ assumes that every additional hard position contributes additively to the overall load on the attention mechanism, while $\mathcal{M}_{\max}$ assumes that memorization speed is bottlenecked by the single hardest position. Other aggregators are equally reasonable in principle: the *mean* $\overline{K} = \mathcal{M}_\Sigma / T$ reports difficulty per position and decouples it from suffix length (useful for comparing datasets of different lengths); the *minimum* reports the easiest position and is unlikely to track memorization speed; and the *standard deviation* or *range* of $K(t)$ captures heterogeneity rather than absolute difficulty (potentially useful as an additional, complementary descriptor). Whether either metric outperforms the simpler raw-K baseline on mixed-arity datasets remains an open empirical question.

Relationship to the conditioning-level hierarchy. The conditioning-level hierarchy (section 4.2) and K-arity capture complementary aspects of disambiguation complexity. The conditioning level characterizes the *structural complexity* of the rule linking prefix to suffix: whether the relevant cue is local (Level 0), at a fixed position (Level 1), or at a position-dependent location (Level 2). K-arity, by contrast, characterizes the *number* of prefix tokens that must be jointly consulted, regardless of how those positions are determined. The two axes are orthogonal in principle: a K=1 dataset can be Level 1 (fixed hint) or Level 2 (position-dependent hint), and a Level 1 dataset can have K=1 or K=2 (two fixed hint positions).

5

Experiments

This chapter provides direct experimental evidence for the disambiguation framework developed in chapter 4. We begin by describing the experimental setup (section 5.1), then present the motivating structured-versus-random experiment (section 5.2), controlled experiments isolating disambiguating power from information content (section 5.3), and the main hierarchy-of-levels comparison (section 5.4). We then test the robustness of the level ordering under varying dataset and vocabulary sizes (subsection 5.4.1, subsection 5.4.2), and conclude with attention analyses (section 5.5), dataset statistics (section 5.6), and the K-arity results (section 5.7).

5.1. Experimental Setup and Evaluation

We study memorization in autoregressive language models by measuring how well a model can reproduce sequences seen during training.

Models. We use GPT-2-style decoder-only transformers (subsection 3.1.1) trained from scratch with the causal language-modeling objective (equation (3.2)). We experiment with two different model sizes: a 6M-parameter and a larger 24M-parameter variant. Full architecture details are given in Table 5.1.[1]

Model	Parameters	Layers	Attention Heads	Embedding Dim	Context Length
GPT-2 (6M)	6.09M	1	8	512	1024
GPT-2 (24M)	24.37M	4	8	712	1024

Table 5.1: Architecture details for the two GPT-2-style models used in experiments.

Training. Models are trained by repeatedly presenting the same fixed dataset until full memorization is achieved, using the standard cross-entropy language-modeling objective (equation (3.2)). Training hyperparameters are summarized in Table 5.2.

Evaluation. Memorization is measured using the exact match metric defined in section 3.3: the model receives the prefix as a prompt, generates a continuation via greedy decoding (subsection 3.1.2), and the output is compared token-by-token to the ground-truth suffix (equation (3.7)). Unless otherwise noted, each sequence consists of 100 tokens split into a prefix of 50 tokens and a suffix of 50 tokens, with the prefix masked out of the training loss as described in section 4.3.

[1] Parameter counts exclude embedding and unembedding layers, whose size depends on the dataset vocabulary rather than the model architecture. Only transformer block parameters (attention, MLP, layer norms) are reported.

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	5×10^{-4}
LR Schedule	Cosine
Warmup Steps	100
Batch Size	512
Weight Decay	0.0
Precision	BF16
Hardware	Single NVIDIA A100

Table 5.2: Training hyperparameters used across all experiments.

5.2. Memorization of Structured and Random Digit Sequences

Previous work [11] shows that datasets with different internal patterns are memorized at different rates. In particular, sequences drawn from WikiText [8] were found to be memorized faster than uniformly random digit sequences, suggesting that language models preferentially memorize structured data over unstructured noise.

We first reproduce this setting with a mixed dataset of 500 sequences: 250 sequences sampled from WikiText and 250 randomly generated digit sequences. In addition to the original setting, we vary the vocabulary size of the random sequences. Rather than sampling random tokens from a vocabulary of size 10 (the ten digits), we also consider larger vocabularies of size 1,000 and 10,000. To control for the increased parameter count induced by larger vocabularies, we use the full GPT-2 vocabulary size (50,257) across all experiments and initialize the token and positional embeddings from a pretrained GPT-2 model, freezing both throughout training.

The results are shown in Figure 5.1. As shown by Panwar et al. [11], the model memorizes WikiText sequences faster than random digit sequences. However, this trend reverses as the vocabulary size increases: random sequences over larger vocabularies are memorized *faster* than WikiText. This observation contradicts a simple "structure versus noise" explanation, since sequences sampled from a larger vocabulary are, if anything, more random. The result suggests that memorization speed is not determined solely by whether data are structured or unstructured, and motivates the disambiguation framework developed in chapter 4.

Relation to the entropy-based view. At first sight, the result that random sequences over a larger vocabulary are memorized faster looks like a direct confirmation of Speicher et al. [14], who report that higher-entropy random strings are memorized faster than lower-entropy ones. We do reproduce that ordering in the random-sequence conditions: V=10 < V=1,000 < V=10,000 in memorization speed corresponds to increasing per-token entropy, so within the random group the entropy-based prediction is consistent with our observations. What our results add is a regime in which entropy alone makes the wrong prediction. WikiText has substantially *lower* per-token entropy than the V=10,000 random sequences (natural language is highly compressible), yet it is memorized *slower*, not faster, than those random sequences. A pure entropy-based account therefore predicts the wrong ordering between the WikiText and large-vocabulary random groups. The disambiguation framework reconciles the two: per-token entropy correlates with memorization speed only because it correlates with how useful individual prefix tokens are for disambiguation. Entropy is a useful proxy when prefixes are homogeneous (as in Speicher's all-random setting), but it breaks down when comparing across data with different distributions, which is exactly the WikiText-vs-random regime studied here.

We do not directly measure the entropy of WikiText in these experiments; the experiments are not designed to estimate it. A more direct comparison with published estimates of natural-language entropy or with the compression-based memorization metric of Morris et al. [9] (equation (3.11)) is left as future work.

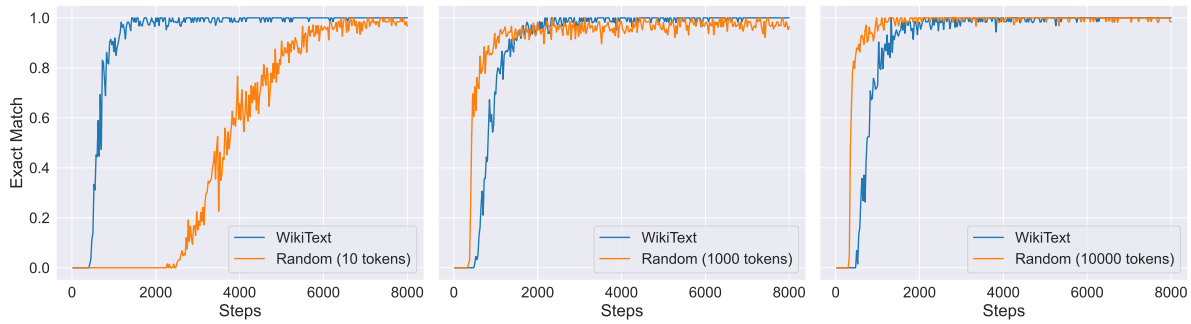


Figure 5.1: Exact match during training for a mixed dataset of 250 WikiText sequences and 250 random sequences under different random-token vocabulary sizes (10, 1,000, and 10,000). The memorization advantage of WikiText reverses at large vocabulary sizes.

5.3. Disambiguating Power vs. Information Content

The structured-versus-random reversal in section 5.2 suggests that memorization speed is not determined by whether data are structured or unstructured, but by how effectively the conditioning context identifies the correct continuation. We now test this hypothesis directly with two complementary experiments. Both vary the prefix vocabulary size, thereby changing the disambiguating power of each prefix token while controlling for information content, but they do so in different ways: the first holds *total* sequence information constant to show that the distribution of bits across regions matters; the second holds *prefix* information constant to show that, even at identical bit counts, the per-token structure of those bits matters.

5.3.1. Experiment 1: Controlling Total Information

We construct two datasets, A and B, each composed of three regions: a prefix, a first suffix segment (*Suffix-1*), and a second suffix segment (*Suffix-2*). The design is illustrated in Figure 5.2, and the exact configurations are summarized in Table 5.3. The only difference is that Dataset B uses a much larger prefix vocabulary (1024 vs. 16), so each prefix token carries more information. Suffix-1 is kept identical across datasets, and Suffix-2 is adjusted in length so that the total information content remains approximately constant (≈998 bits).

If memorization were governed by total information content alone, both datasets should be memorized at similar rates. Instead, the results in Figure 5.3 show that the dataset with the more informative prefix is memorized faster, supporting the claim that what matters is not how many bits the sequence contains, but how effectively those bits identify the correct continuation.

However, this experiment confounds vocabulary size with prefix information content: Dataset B's prefix carries 500 bits versus 200 for Dataset A. One might argue that the effect arises simply because Dataset B has more bits in the prefix, not because of the vocabulary size per se. The next experiment disentangles these two factors.

	Prefix			Suffix-1			Suffix-2			Total	
Dataset	Vocab	Length	Bits	Vocab	Length	Bits	Vocab	Length	Bits	Length	Bits
A	16	50	200	128	50	350	128	64	448	164	998
B	1024	50	500	128	50	350	128	21	147	121	997

Table 5.3: Configuration of the controlled disambiguation experiment shown in Figure 5.2. The prefix information density differs between datasets while total information content is approximately matched.

5.3.2. Experiment 2: Controlling Prefix Information

To resolve the confound above, we now hold prefix information content exactly constant while varying the vocabulary size. This uses a simpler two-region design (prefix and suffix only), accepting variation in sequence length rather than introducing an additional suffix region as a potential confound.

We construct five datasets, each consisting of $N = 2{,}000$ sequences divided into a *prefix* and a *suffix*. The

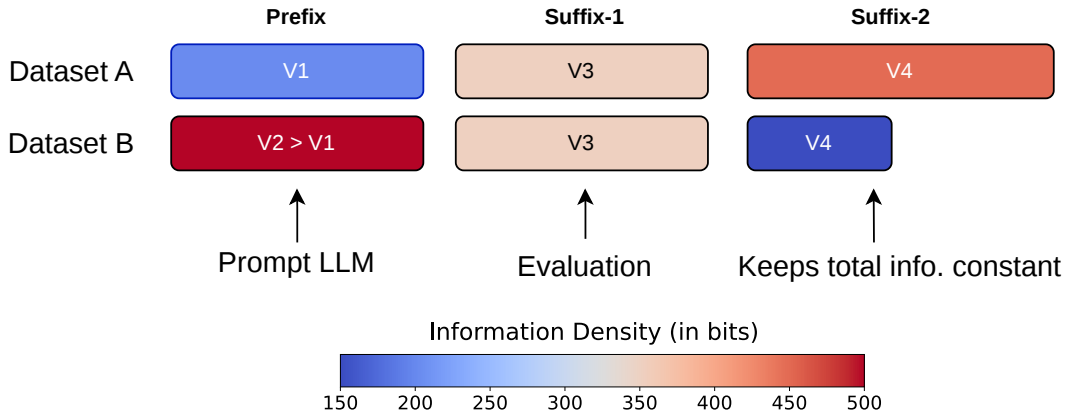


Figure 5.2: Controlled disambiguation experiment. Dataset B uses a larger prefix vocabulary, resulting in higher information density in the prefix. Suffix-1 is identical across datasets. Suffix-2 is adjusted so that total information content is approximately matched.

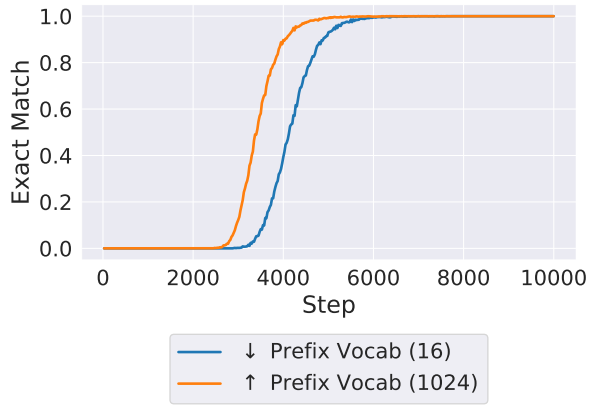


Figure 5.3: Exact match during training for the controlled disambiguation experiment. The dataset with the more informative prefix is memorized faster, despite approximately matched total information content.

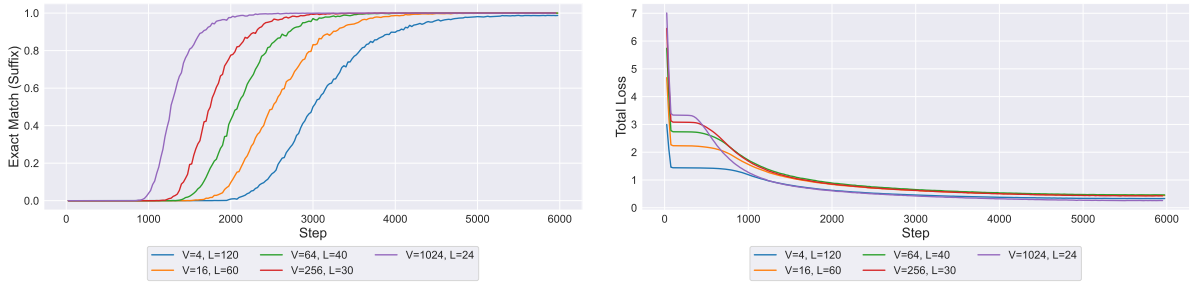
prefix vocabulary $V \in \{4, 16, 64, 256, 1024\}$ varies across datasets, and the prefix length L is set so that

$$H_{\text{prefix}} = L \cdot \log_2 V = 240 \text{ bits}$$

is held exactly constant; this requires $L \in \{120, 60, 40, 30, 24\}$ for $V \in \{4, 16, 64, 256, 1024\}$, respectively. The suffix is identical across all datasets: vocabulary size $V_s = 128$, length $L_s = 50$ tokens, carrying $H_{\text{suffix}} = 350$ bits. The total information content of every sequence is therefore $H = 590$ bits. The exact configurations are summarized in Table 5.4. Sequences are generated by sampling tokens uniformly at random from non-overlapping token ranges: prefix tokens from $\{0, \ldots, V-1\}$ and suffix tokens from $\{V, \ldots, V+127\}$.

Hypothesis. If memorization speed were governed solely by the number of bits in the prefix, all five conditions should yield identical learning curves, since every prefix carries exactly 240 bits. Carlini et al. [2] show a log-linear relationship between prompt length and verbatim memorization, which might suggest that the condition with the longest prefix ($V = 4$, $L = 120$) should be memorized most easily. Our hypothesis is the opposite: it is not the raw number of context tokens that matters, but how much each token helps disambiguate the sequence.

Concretely, in this uniform synthetic setting, larger V means that prefix information is concentrated into fewer, more distinctive tokens. At a fixed total prefix budget, larger V therefore implies stronger



(a) Exact match.

(b) Training loss.

Figure 5.4: Iso-prefix-information experiment. Each curve corresponds to a different prefix vocabulary size V, with prefix length adjusted so that all conditions carry exactly 240 prefix bits (Table 5.4). Left: exact match on the suffix region. Right: training loss.

per-token cues but *shorter* prefixes, so the comparison disentangles vocabulary size from sheer prefix length. This is the dimension we vary here, and is conceptually distinct from Speicher et al. [14], who vary the total entropy of a fixed-length sequence: in their setup, larger vocabularies always come with more total bits, while our design holds total prefix bits constant and isolates how concentrated those bits are in each prefix token. The conditioning-level and K-arity experiments then study the separate question of how the model must use those cues.

The key mechanism is the *collision structure* of the prefix. At each prefix position, the probability that two distinct sequences share the same token by chance is $1/V$. With $V = 4$ and $L = 120$, there are 120 positions at which pairwise collisions can occur, each with probability 0.25, yielding many positions that contribute no pairwise disambiguation. The model must aggregate evidence across a large number of positions before it can reliably identify the sequence. With $V = 1024$ and $L = 24$, the per-position collision probability falls to ≈ 0.001: any single prefix token is highly likely to be unique to its sequence, so the model can identify it by attending to just one or two positions – a far simpler computation for a shallow transformer.

Results. Figure 5.4 confirms the hypothesis: conditions with larger vocabulary V are memorized faster, despite all prefixes carrying exactly 240 bits. The V=1024 condition reaches full exact match earliest, while V=4 is slowest (Figure 5.4a). Training loss curves (Figure 5.4b) show the same ordering.

Table 5.4: Configuration of the iso-prefix-information experiment. The prefix information content is exactly 240 bits in all conditions; the suffix is identical across datasets. $N = 2{,}000$ sequences per dataset.

	Prefix			Suffix			Total	
V	Length	Bits	Bits/tok	Vocab	Length	Bits	Length	Bits
4	120	240	2.0	128	50	350	170	590
16	60	240	4.0	128	50	350	110	590
64	40	240	6.0	128	50	350	90	590
256	30	240	8.0	128	50	350	80	590
1024	24	240	10.0	128	50	350	74	590

5.3.3. Summary

Taken together, the two experiments suggest that memorization speed depends on the usefulness of individual prefix cues, not only on raw bit count. Experiment 1 showed that concentrating information in the prefix accelerates memorization even when total sequence bits are matched; Experiment 2 showed that this effect persists even when prefix bits are held exactly constant, ruling out information content as the explanation. The collision-structure analysis provides a mechanistic account: higher per-token information reduces the number of pairwise collisions, allowing the model to identify sequences from fewer attended positions. This motivates the question addressed next: how does the *structural complexity* of the conditioning rule affect memorization?

5.4. Memorization Across Conditioning Levels

Having established that the usefulness of individual prefix cues affects memorization speed, we now compare datasets that differ in the *structural complexity* of the conditioning rule. We train both GPT-2 6M and GPT-2 24M models from scratch on datasets constructed at each conditioning level (Table 4.1) and track their memorization dynamics. All datasets are validated using the verification pipeline described in subsection 4.3.4.

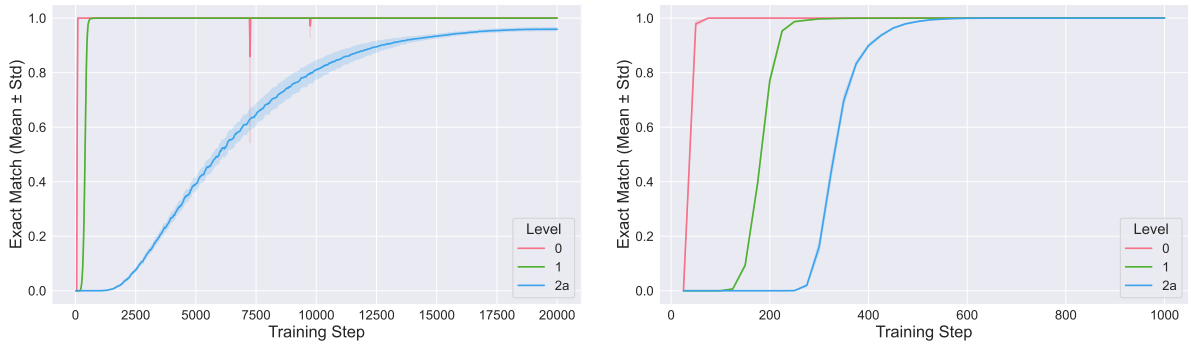
Dataset configuration. We generate synthetic datasets of increasing conditioning complexity using the configuration summarized in Table 5.5. Each sequence is split into a uniformly random *prefix* and a *suffix* whose tokens are deterministically generated according to a level-specific rule conditioned on the prefix, using the lookup-table scheme (section 4.3).

Table 5.5: Dataset parameters for the conditioning-levels experiment.

Parameter	Value
Sequence length (L)	100
Vocabulary size (V)	100
Prefix length (P)	50
Number of sequences (N)	6,000
Random prefix	True
Aggregation	Lookup
Seeds	{0, 1, 2}

Figure 5.5 shows the exact match accuracy on the conditioned suffix region as a function of training steps for both model sizes, averaged across seeds and (for Level 2) across all five hint-position strategies (Table 4.2). Shaded bands indicate ±1 standard deviation. The individual Level 2 strategies show comparable memorization speeds and are therefore aggregated here.

The results reveal a clear ordering: datasets whose continuations can be recovered from simpler conditioning structures are memorized more quickly, consistent with Hypothesis 1. Level 0 (local context only) is learned rapidly, while Level 2 (position-dependent hints) requires substantially more training. Across both model sizes, the ordering is preserved: Level 0 is memorized first, followed by Level 1, then Level 2. The 24M model reaches perfect accuracy on all levels substantially earlier than the 6M model, indicating that increased capacity accelerates memorization uniformly across the hierarchy.



(a) GPT-2 6M (1 layer).

(b) GPT-2 24M (4 layers).

Figure 5.5: Exact match on the suffix region across conditioning levels for both model sizes, averaged across three random seeds. The larger model memorizes all levels faster, with the gap between levels narrowing considerably.

A particularly important implication is that the full prefix is not necessarily the easiest signal for the model to exploit, even though it always contains enough information to identify the sequence. In practice, the model appears to favor smaller or simpler cues when such cues are available. This helps explain the reversal observed in section 5.2: larger-vocabulary random sequences are easier to memorize not because they are less random, but because they are more likely to contain highly informative tokens

that uniquely identify the sequence from a short context. This links the controlled hierarchy back to the related work: Carlini et al. [2] show that more prompt context can increase verbatim recall, while Tirumala et al. [16] and Stoehr et al. [15] show that only a small number of identifying tokens may be doing most of the work. The hierarchy tests this identifier view under controlled conditions where the relevant cue is known by construction.

5.4.1. Scaling with Dataset Size

We further test whether the conditioning-level ordering is robust to changes in dataset size by varying $N \in \{500, 2{,}000, 6{,}000, 12{,}000, 24{,}000\}$ and recording, for each condition, the first training step at which exact match exceeds 99%. The results (Figure 5.6) show that the Level 0 < Level 1 < Level 2 ordering is preserved across all dataset sizes. Level 0 requires a nearly constant number of steps regardless of N. Level 1 and Level 2 both require more steps as N grows, with the gap between levels widening at larger N.

This behavior can be understood through the collision structure introduced in subsection 5.3.2. With N sequences over a vocabulary of size V=100 and prefix length P=50, the expected number of sequences sharing the same token at any given prefix position is N/V. As N grows from 500 to 24,000, this expected collision count rises from 5 to 240, meaning that any single prefix token becomes increasingly unlikely to uniquely identify a sequence. The impact of this degradation depends on the conditioning level. Level 0 relies only on local suffix context, so prefix collisions are irrelevant and training steps remain constant. Level 1 conditions on a single prefix position; as N/V grows, the model must learn to combine that position with additional context to resolve collisions, but the fixed lookup position still provides a strong initial cue. Level 2, by contrast, requires the model to first identify *which* prefix position to attend to, a search problem whose difficulty compounds with the number of colliding sequences, since more collisions produce more candidate positions that appear equally plausible. This compounds the difficulty for Level 2: each increase in N not only adds more sequences to disambiguate but also degrades the per-position signal the model relies on to locate the correct conditioning cue.

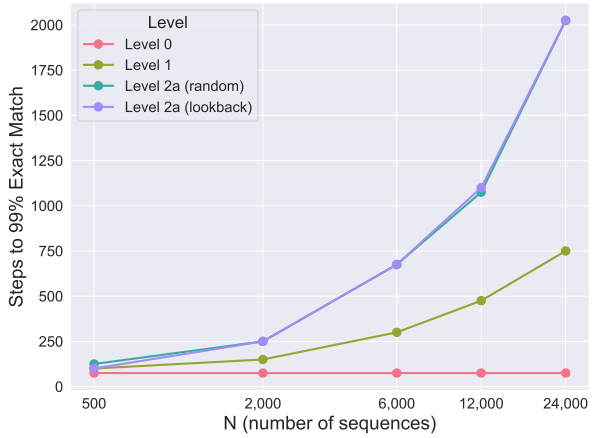


Figure 5.6: Training steps required to reach 99% exact match as a function of dataset size N, broken down by conditioning level. The ordering across levels is preserved at all dataset sizes, with the gap widening as N grows.

5.4.2. Scaling with Vocabulary Size

The results above establish the Level 0 < Level 1 < Level 2 ordering at a fixed vocabulary size of $V = 100$. A natural question is whether this ordering is robust to changes in vocabulary size, particularly larger vocabularies where each token carries more disambiguating information.

Setup. We repeat the conditioning-levels experiment (Table 5.5) with vocabulary sizes $V \in \{10, 100, 1{,}000, 10{,}000\}$, keeping all other parameters fixed. This directly complements the iso-prefix-information experiment (subsection 5.3.2), which showed that larger V accelerates memorization at a fixed conditioning level; here we test whether V interacts with conditioning complexity.

Conditioning structure and disambiguating power. The two axes studied in this thesis are not independent. The conditioning level (section 4.2) describes the *structural complexity* of the rule that links prefix to suffix: which prefix positions are relevant and whether they vary with the suffix index. Per-token disambiguating power is the complementary intuition that some prefix positions are more distinctive cues than others. They interact: a Level 2 dataset over a small vocabulary is hard for two reasons at once: the relevant position is unknown and the model must search for it, and each candidate position is only a weak cue. Increasing V makes individual prefix tokens more distinctive in this uniform setting, which can shorten the effective search and eventually make Level 2 indistinguishable from Level 1, as we observe at V=10,000. Conversely, holding the conditioning level at Level 0 makes this distinction largely irrelevant for suffix prediction, since the relevant cue is local. The conditioning level thus determines *whether* the model needs to exploit prefix cues, while their disambiguating power shapes how easy those cues are to use when needed.

Hypothesis. We expect the Level 0 < Level 1 < Level 2 ordering to be preserved across vocabulary sizes, because the structural complexity of the conditioning rule is independent of V. However, the gap between levels may narrow as V increases: with higher per-token information content, even the harder levels benefit from shorter effective disambiguation prefixes, potentially reducing the relative advantage of simpler conditioning structures.

Results. Figure 5.7 shows the training steps required to reach 96% exact match as a function of vocabulary size for each conditioning level. The Level 0 < Level 1 < Level 2 ordering is preserved across all vocabulary sizes up to $V = 1{,}000$, confirming that the hierarchy is robust to changes in per-token information content.

At $V = 10$, all levels cluster together at roughly 100–200 steps, reflecting the high collision rate: with only 10 tokens available, prefix positions provide minimal disambiguating power regardless of the conditioning structure, so the model must rely on memorizing longer contexts for all levels alike. As V increases to 100 and 1,000, the levels separate sharply: Level 0 remains easy since it depends only on local suffix context, while Level 1 and Level 2 require progressively more steps as the model must learn to exploit prefix information in an increasingly large token space.

At $V = 10{,}000$, the gap between Level 1 and Level 2 collapses: both converge to approximately 2,300 steps. This convergence can be understood through the collision framework. With $N = 6{,}000$ sequences and $V = 10{,}000$, the expected number of sequences sharing the same token at any prefix position is $N/V = 0.6$, meaning that nearly every prefix token uniquely identifies its sequence. At this point, the structural difficulty that distinguishes Level 2 from Level 1 – having to *locate* the relevant prefix position rather than attending to a fixed one – is largely neutralized, because almost any position the model attends to already provides a unique cue. Level 0, however, still increases in difficulty at $V = 10{,}000$ because it must encode more distinct suffix patterns without leveraging prefix information.

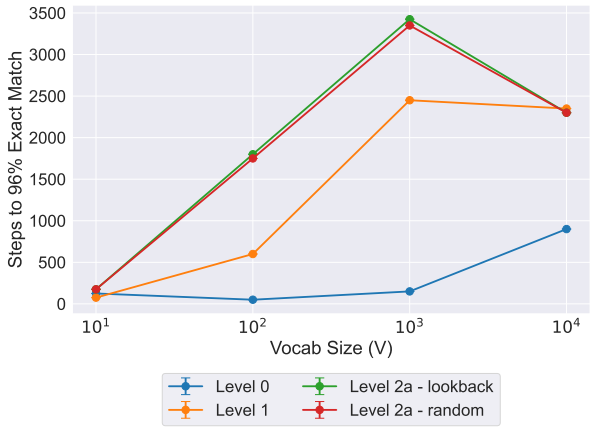


Figure 5.7: Training steps required to reach 96% exact match as a function of vocabulary size V, broken down by conditioning level. The Level 0 < Level 1 < Level 2 ordering is preserved up to $V = 1{,}000$; at $V = 10{,}000$ the gap between Level 1 and Level 2 collapses as prefix tokens become nearly unique.

5.5. Attention Analysis

The preceding sections established that conditioning complexity predicts memorization speed. A natural follow-up question is whether the model's internal representations reflect the embedded conditioning structures. To investigate this, we inspect attention patterns in trained models. For each model, we average attention scores across all training sequences and report, for a given layer, the maximum attention value across heads.

The resulting patterns are shown in Figure 5.9 and Figure 5.10. In Layer 0, the attention maps exhibit clear structure that correlates with the corresponding embedded conditioning level: Level 0 models attend primarily to the immediately preceding token, while Level 1 and Level 2 models show additional attention to the relevant prefix positions. This suggests that the earliest layer learns to detect or retrieve the relevant disambiguating cues from the prefix. By contrast, Layer 1 shows no equally clear condition-specific pattern, indicating that the retrieval of the conditioning cue is concentrated in the first layer, while later layers perform less interpretable transformations.

Caveat on the layer-0 reading. We caution that the Layer 0 versus Layer 1 contrast above does not by itself imply that Layer 0 "does the disambiguation work". Later transformer layers operate on residual representations that already mix information across positions, so by Layer 1 the queries and keys no longer correspond directly to original tokens; it is therefore expected that the cleanest token-level disambiguation pattern appears in Layer 0, regardless of where in the depth of the network the cue is functionally exploited. The attention maps support the weaker but still informative claim that token-level disambiguating cues are *retrieved* early; whether the disambiguation is ultimately *used* early is a separate question that requires causal interventions (head ablations, activation patching) to answer. We treat this as a caveat on the interpretation rather than as a positive conclusion about layer-wise division of labor.

These observations are consistent with Hypothesis 1: memorization is facilitated when the model can cheaply recover a small, informative cue that identifies the correct continuation.

5.5.1. One-Layer Transformer

To isolate the role of attention more directly, we repeat the analysis using a single-layer transformer. In a transformer block, the attention mechanism is the only component that mixes information across positions: it allows each token to read from and aggregate representations of earlier tokens. The feed-forward network, by contrast, applies the same learned transformation independently at every position. Consequently, any prefix-dependent disambiguation must be mediated by the attention layer, making its weights a natural diagnostic for whether the model has learned to retrieve the relevant conditioning cue. In a single-layer model this reasoning is especially clean: there is only one attention step followed by one feed-forward block, so any prefix information that the model uses must be gathered in this single attention step; there is no second layer that could compensate or redistribute attention.

Figure 5.8 shows the resulting attention maps for four conditioning variants. The two Level 1 models (fixed hint at position 5 and position 20) concentrate attention sharply on a single prefix column corresponding to the hint location, with all suffix positions attending uniformly to the same key. The Level 2 lookback model displays a clean diagonal pattern, where each suffix query attends to the prefix position at a fixed offset behind it. The Level 2 modulus 10 model distributes attention across periodic prefix positions, producing a scattered but structured pattern. In all cases the attention structure closely mirrors the conditioning rule embedded in the dataset.

These patterns are qualitatively consistent with those observed in Layer 0 of the multi-layer model (Figure 5.9), suggesting that the retrieval of disambiguating cues is already implemented in a single attention layer. However, we note that attention weights reflect correlational structure: they show *where* the model looks, not whether the retrieved information is causally necessary for the prediction. It remains possible that alternative mechanisms (e.g., positional embeddings combined with the feed-forward network) could in principle achieve similar results without the observed attention pattern. Intervention-based analyses, such as ablating specific attention heads, would be needed to establish a causal link.

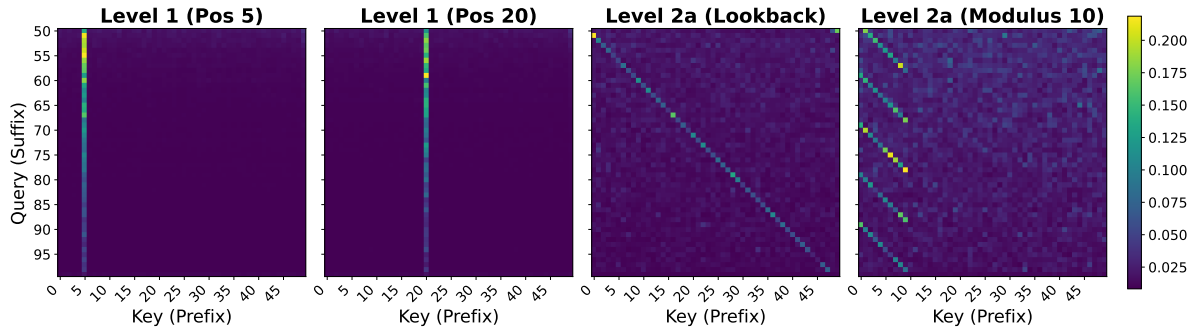


Figure 5.8: Attention patterns of a one-layer transformer trained on four conditioning variants. Level 1 models attend to a single fixed prefix column, while Level 2 models exhibit position-dependent patterns (diagonal for lookback, periodic for modulus 10). In all cases the attention structure mirrors the embedded conditioning rule.

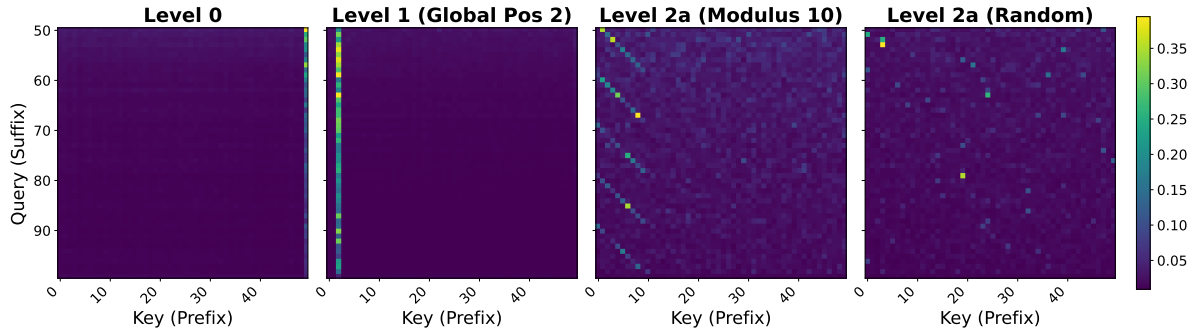


Figure 5.9: Average attention patterns across training sequences for datasets with different conditioning levels. In **Layer 0**, the maximum attention value across heads reveals patterns that align with the embedded conditioning structure.

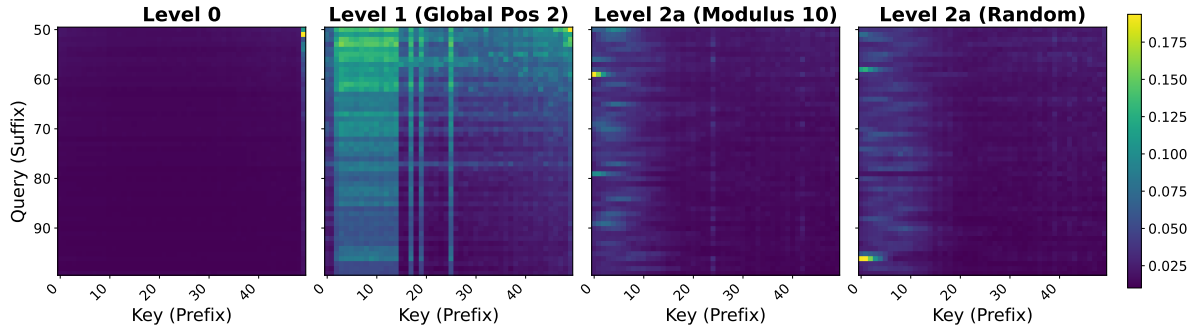


Figure 5.10: Average attention patterns for the same models in **Layer 1**. The attention maps no longer show a clear condition-specific structure.

5.6. Dataset Statistics and Potential Confounds

The previous experiments suggest that conditioning complexity influences memorization speed. However, the different synthetic datasets may also differ in simpler distributional statistics, such as their unigram, bigram, or trigram distributions. We therefore analyze several dataset-level statistics to determine whether the observed differences can be attributed to low-order distributional properties alone.

5.6.1. n-gram Entropy

For each dataset, we extract all n-grams of length n from the 50-token suffix window where the conditioning rule is embedded, pooling across all training sequences to obtain an empirical distribution. Let $p(g)$ denote the empirical frequency of a distinct n-gram g. The n-gram entropy is

$$H = -\sum_{g} p(g) \log_2 p(g), \tag{5.1}$$

where the sum runs over all distinct observed n-grams. Higher entropy indicates a more uniform, and therefore less repetitive, distribution of local patterns. For reference, the theoretical maximum unigram entropy for a vocabulary of size $|\mathcal{V}| = 100$ is $H_{\max} = \log_2(100) \approx 6.64$ bits.

The results are shown in Figure 5.11a. All conditioning levels exhibit nearly identical unigram entropy, confirming that the marginal token distribution is effectively controlled. Differences emerge only for larger n: higher conditioning levels exhibit substantially larger trigram entropy, indicating richer higher-order sequential structure.

5.6.2. Type-Token Ratio

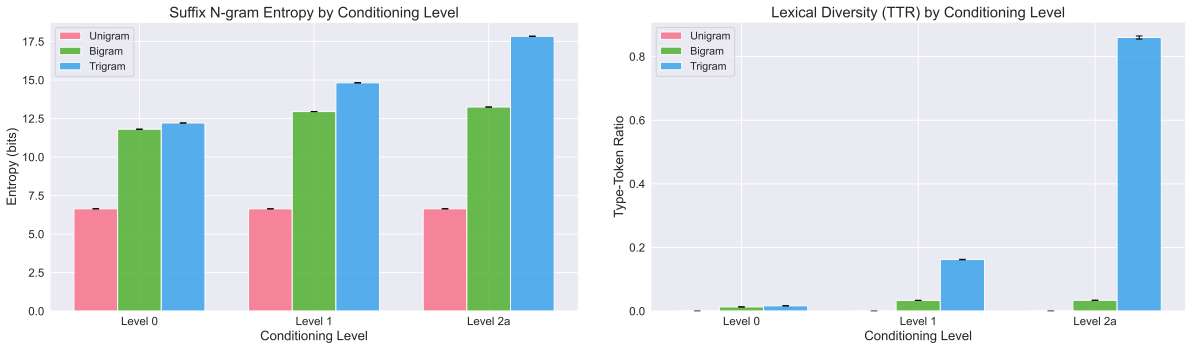
We also compute the type-token ratio (TTR), defined as

$$\text{TTR} = \frac{\text{number of distinct } n\text{-grams observed}}{\text{total number of } n\text{-gram tokens}}. \tag{5.2}$$

TTR measures how many unique patterns appear relative to the total count of observed positions. It approaches zero when patterns are heavily reused and approaches one when nearly every position contains a distinct n-gram.

The results in Figure 5.11b mirror the entropy analysis: unigram TTR is nearly identical across levels, while higher-order TTR diverges sharply, with higher conditioning levels containing a much larger proportion of unique trigrams.

This does not invalidate the conditioning-level hypothesis; rather, it clarifies that conditioning complexity and higher-order sequential diversity are closely related in the constructed datasets. The harder datasets require the model to capture more diverse local dependencies to recover the relevant disambiguating rule.



(a) n-gram entropy across conditioning levels.

(b) Type–token ratio across conditioning levels.

Figure 5.11: Distributional statistics of the synthetic datasets. While unigram statistics are matched, higher-order structure differs substantially across conditioning levels. Error bars show ±1 standard deviation across random seeds.

5.7. K-Arity Results

The previous sections characterized memorization difficulty along the axis of *structural complexity*: how the disambiguating cue relates to the current position. We now test a complementary axis introduced in section 4.4: the *number* of prefix tokens that must be jointly consulted. Hypothesis 1 predicts that models should memorize K=1 datasets faster than K=2, which in turn should be faster than K=3, since higher K requires the model to jointly attend to more prefix positions before the suffix becomes predictable. The results in Figure 5.12 confirm this ordering: higher arity consistently requires more training steps to reach full memorization. This result is another controlled version of the small-context-subset observations in Speicher et al. [14]: instead of asking whether some subset suffices after training, we vary how many prefix tokens must jointly suffice and observe the effect on memorization speed.

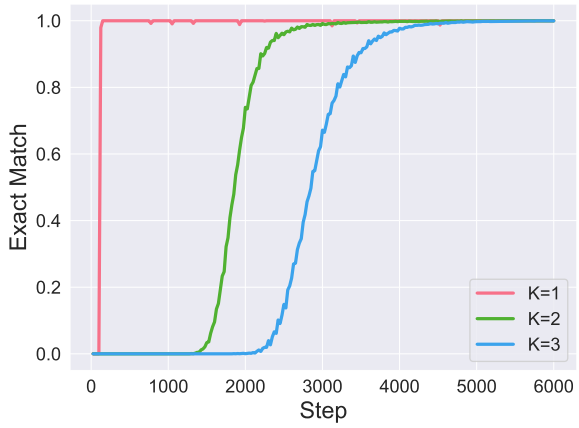


Figure 5.12: Memorization speed (exact match vs. training steps) for datasets of K-arity $K = 1, 2, 3$. Higher arity consistently requires more steps to reach full memorization.

5.7.1. Attention Patterns under K-Arity

To understand *why* higher arity slows memorization, we inspect the attention patterns of single-layer transformers trained on K-arity datasets. In order to simplify the interpretation of the attention maps, we use a variant in which each position function $\varphi_k(t)$ maps every suffix position to the *same* fixed prefix location, rather than to a position-dependent random lookup table. This ensures that the only variable across conditions is K itself: the model must attend to exactly K fixed prefix columns to predict the suffix.

Figure 5.13 shows the resulting attention maps for $K = 1, 2, 3$. For K=1, attention concentrates sharply on a single prefix column corresponding to the hint position, closely mirroring the Level 1 patterns observed in subsection 5.5.1. For K=2, the model distributes attention across two distinct prefix columns, though the per-column signal is weaker (peak attention ≈ 0.16 versus ≈ 0.60 for K=1). For K=3, the pattern degrades further: while some attention concentrates near the three hint positions, the map is substantially noisier and the model appears to exploit alternative shortcuts rather than cleanly retrieving all three cues.

This progressive degradation reflects a mechanistic bottleneck: because softmax attention produces a distribution summing to one, the total attention weight is a fixed budget. When that budget must be split among K relevant positions, the per-position weight drops and the designated columns become difficult to distinguish from the uniform baseline of irrelevant ones. The value vector read out by attention is therefore increasingly diluted by contributions from non-hint positions. At K=3, this dilution becomes severe enough that the model can no longer reliably decompose the task into K independent lookups and instead resorts to spurious correlations or alternative retrieval paths that diverge from the intended conditioning structure.

This attention-budget mechanism provides direct mechanistic support for Hypothesis 1: a conditioning structure requiring joint consultation of K positions imposes a progressively harder retrieval problem on the attention layer, explaining why datasets with simpler conditioning structures are memorized

faster. The finding also motivates the design choice to conduct the main conditioning-level hierarchy (section 4.2) at K=1, where the gap between the intended conditioning rule and the model's learned strategy remains small enough to keep attention patterns interpretable and experimental conclusions well-controlled. Finally, it offers a reason why controlled experiments on natural language are difficult: natural text effectively has high and heterogeneous arity, with the next token depending jointly on many scattered context positions, making it far harder to isolate individual conditioning structures.

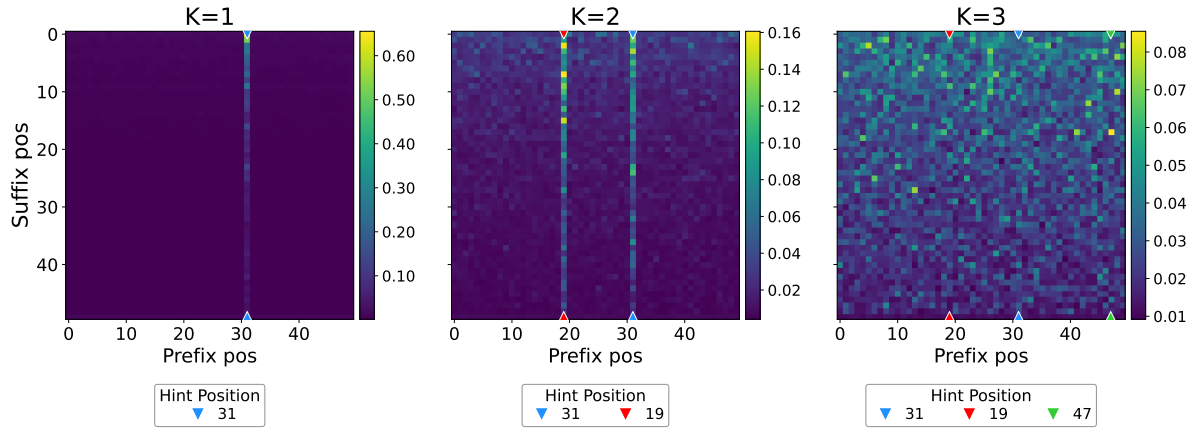


Figure 5.13: Attention patterns of a single-layer transformer trained on K-arity datasets with fixed hint positions. For K=1, attention concentrates sharply on the single hint column. For K=2, attention splits across two hint columns with reduced per-column signal. For K=3, the pattern is substantially noisier, suggesting the model exploits alternative shortcuts rather than cleanly retrieving all three cues.

6

Natural Language Experiments

The synthetic experiments in chapter 5 isolate the role of conditioning structure in a controlled setting. The remaining question is whether the same mechanisms apply to natural language, where multiple overlapping cues are present simultaneously.

6.1. Fixed-Position Cues in Natural Language

To test this, we curate two datasets from WikiText. The first is constructed to satisfy a Level 1-style property: one fixed position in the prefix contains a token that is unique across all sequences and can therefore, in principle, serve as a single-token identifier for the correct continuation. The second dataset is constructed so that no such single fixed-position cue exists.

If memorization in natural language were dominated by this single explicit cue, the Level 1-style dataset should be memorized faster. In practice, however, both datasets are memorized at nearly the same rate. This suggests that natural language provides many competing cues (syntactic patterns, lexical co-occurrence, semantic regularities) that may overshadow the effect of an isolated embedded identifier.

6.2. Modulating Prefix Structure

We further consider three variants designed to modulate the amount and nature of contextual structure in the prefix:

1. the original natural-language prefix,
2. a translated version of the prefix paired with the same suffix,
3. a translated prefix augmented with a unique identifier at a fixed position, again paired with the same suffix.

Despite these manipulations, the exact match curves remain largely overlapping, indicating that changing the explicit conditioning level is not sufficient to produce large differences in memorization speed when rich linguistic structure is already present.

The attention maps in Figure 6.1 provide a qualitative explanation. Rather than focusing narrowly on the inserted cue, the model distributes attention across a broad range of positions associated with natural-language regularities. This diffuse pattern closely resembles the degraded K=3 attention maps observed in the synthetic K-arity experiments (subsection 5.7.1), where the attention budget becomes too diluted across multiple relevant positions for the model to cleanly retrieve any single cue. In natural language, the effective arity is even higher: the model appears to exploit many redundant signals in the prefix, making any single embedded cue comparatively unimportant.

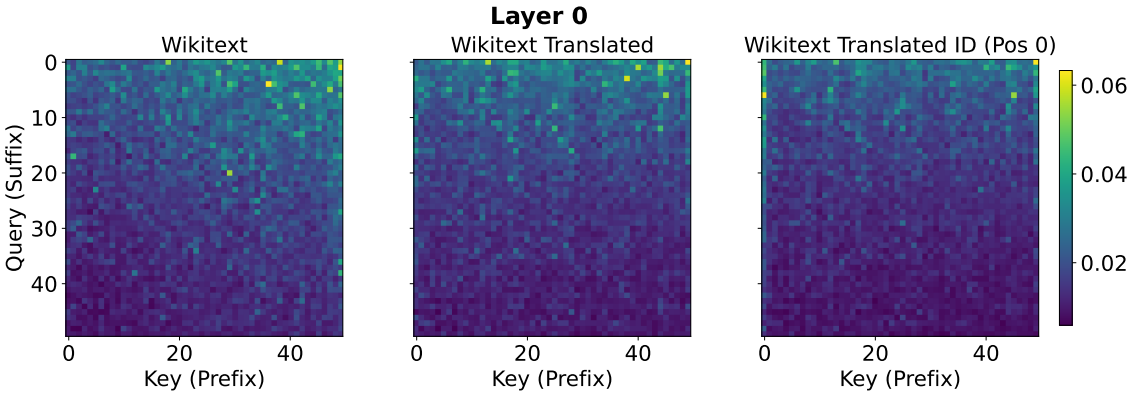


Figure 6.1: Average attention patterns for models trained on natural-language datasets. The model distributes attention across multiple linguistic cues rather than concentrating on a single fixed-position identifier.

6.3. Additional Controls

Two additional experiments further corroborate this picture. In the first, we train on multilingual WikiText datasets (one in which sequences are translated into Latin and one drawing from a diverse set of languages) using a multilingual tokenizer to test whether reducing the model's prior linguistic exposure alters memorization dynamics. In the second, we construct a paraphrase dataset of 100 US-politics sequences from WikiText, each paired with nine semantically equivalent paraphrases generated by a language model, and compare memorization against a control dataset of 1000 independent WikiText sequences. If shared semantic content across paraphrases made suffixes harder to disambiguate, the paraphrase group should memorize more slowly. In both cases, all conditions are memorized at similar rates.

6.4. Discussion

The null results do not constitute evidence against the hypothesis. Because the residual linguistic cues present across all conditions are sufficient for disambiguation on their own, no single manipulation can isolate the effect of conditioning complexity. The result is therefore consistent with two interpretations: either the hypothesis holds but the effect is masked by redundant cues, or the hypothesis does not generalize to natural language. Distinguishing between these possibilities remains an important direction for future work.

7

Discussion and Conclusion

We now discuss the broader implications of our findings, identify the main limitations of the current work, suggest directions for future research, and give concluding remarks.

7.1. Summary of Findings

The central contribution of this thesis is a disambiguation-based framework for understanding memorization speed in autoregressive language models. Our main findings can be summarized as follows.

Vocabulary size changes the structured-random comparison. We replicated and extended a prior finding that natural language is memorized faster than random digit sequences, and showed that this advantage reverses when the random sequences are drawn from a sufficiently large vocabulary (section 5.2). Within the random-sequence conditions, this pattern is consistent with the entropy-based account of previous work [14]: higher per-token entropy goes together with faster memorization. The WikiText comparison suggests an additional factor. Although WikiText is more compressible than the V=10,000 random condition, it is memorized more slowly in our experiment. We interpret this as evidence that individual prefix tokens can matter as disambiguating cues, not only as contributors to total sequence entropy.

The disambiguation framework predicts memorization ordering. Across a hierarchy of four conditioning levels, we observed a consistent ordering: simpler conditioning structures correspond to faster memorization (section 5.4). This held even when the datasets were matched in their marginal (unigram) token distributions, suggesting that the differences in memorization speed are not simply artifacts of low-order statistical differences.

K-arity is an independent predictor of memorization difficulty. Varying the number of prefix tokens that must be jointly consulted (the K-arity) independently modulates memorization speed (section 5.7), confirming that the *number* of required cues is as important as the *structural complexity* of the conditioning rule. The two axes (conditioning level and K-arity) capture complementary aspects of memorization difficulty.

The model retrieves disambiguating cues in its first layer. Attention analysis (section 5.5) revealed that the relevant conditioning information is predominantly visible in Layer 0 of the transformer, with later layers showing no equally clean condition-specific structure. As discussed in the caveat at the end of section 5.5, this is partially expected on architectural grounds – later layers operate on residual representations that no longer correspond directly to original tokens – and so we treat the observation as evidence that disambiguating cues are *retrieved* early rather than as a strong claim about where in depth the disambiguation is functionally executed. Causal interventions (head ablations, activation patching) are needed to support the latter claim.

Natural-language results are neutral. Our attempts to modulate conditioning complexity in natural-language settings (inserting unique identifiers, translating prefixes, or exploiting semantic paraphrases) produced no detectable differences in memorization speed (chapter 6). This is consistent with the hypothesis but does not confirm it, because the rich redundancy of natural-language cues prevents any single manipulation from being decisive.

7.2. Limitations

Several limitations of this work deserve explicit acknowledgment.

Scale. All experiments use small GPT-2 models (6M and 24M parameters) trained from scratch on small, fixed datasets. It is unclear whether the qualitative findings would transfer to models orders of magnitude larger, or to settings in which the training corpus is large enough that most sequences are seen only once.

Synthetic datasets. The controlled synthetic experiments rely on datasets in which the conditioning structure is exactly embedded by construction. This makes the experimental claims precise, but it also means the datasets are qualitatively different from naturally occurring text. Extending the framework to settings where the conditioning structure must be estimated rather than known a priori is an important step for future work.

Causal claims. While the attention analyses are consistent with the hypothesis that models exploit embedded conditioning cues, they do not establish causality. It is possible that the observed attention patterns are a consequence of memorization rather than its cause. Intervention-based analyses (for instance, ablating specific attention heads and observing the effect on memorization speed) would provide stronger evidence.

Alternative conditioning metrics. The conditioning-level hierarchy and K-arity are two specific ways to measure disambiguation complexity, but other metrics are conceivable. The minimum description length of the suffix given the prefix and compressibility-based measures may yield complementary characterizations.

7.3. Future Work

Several natural directions follow from the results and limitations described above.

Estimating conditioning complexity in natural text. The key challenge in applying the framework to natural language is that the conditioning structure is not known in advance. Estimating the effective K-arity or conditioning level of a real text passage would allow the framework to make predictions about which natural-language passages are memorized first.

Scaling laws for disambiguation complexity. How does the relationship between conditioning complexity and memorization speed change as model capacity increases? One might expect larger models to handle higher-arity conditions more easily, effectively reducing their apparent conditioning complexity. Studying scaling laws along this dimension could reveal why memorization becomes easier at scale.

Connections to grokking and generalization. Morris et al. [9] argue that memorization and generalization are separated by a capacity threshold at approximately 3.6 bits per parameter: when the dataset's information content exceeds this budget, the model can no longer store all examples individually and is forced to discover shared structure, producing the characteristic double-descent curve. Our framework characterizes the order in which sequences enter the memorized regime; it would be natural to ask whether sequences with lower conditioning complexity also generalize sooner, or whether they remain memorized without ever being generalized.

Mixed-arity datasets and scalar metrics. The scalar metrics $\mathcal{M}_\Sigma$ and $\mathcal{M}_{\max}$ proposed in section 4.4 were not tested experimentally. Constructing datasets in which different suffix positions have different conditioning complexity and comparing how well each metric predicts memorization speed would validate these measures.

Privacy and copyright applications. Understanding which sequences are memorized fastest has direct implications for data deduplication strategies and privacy-preserving training procedures. If sequences with simple conditioning structures are disproportionately at risk of verbatim reproduction, they may require special treatment during dataset curation. The K-arity measure, being computable from the dataset alone, could serve as a practical pre-training filter.

7.4. Conclusion

This thesis introduced a disambiguation-based framework for memorization in autoregressive language models and provided experimental evidence that memorization speed is governed by the complexity of the minimal conditioning structure required to uniquely identify a sequence from its prefix. The framework unifies several previously isolated findings about vocabulary size, entropy, identifying tokens, and context length under a single mechanistic principle: the model memorizes fastest when it can locate a simple, highly informative cue that resolves all ambiguity about the correct continuation.

Through controlled synthetic experiments, we demonstrated that conditioning complexity predicts memorization ordering independently of marginal sequence entropy, and that the corresponding cues are visible in early attention patterns. The complementary K-arity measure confirmed that the number of jointly required prefix tokens is an additional independent axis of difficulty.

The natural language experiments reveal the limits of our current approach: in text rich with redundant linguistic cues, isolated manipulations of conditioning complexity are insufficient to produce detectable differences. Developing methods to measure aggregate conditioning complexity in naturally occurring text is the most important open challenge for future work, and would transform the framework from a theoretical account into a practical tool for understanding, predicting, and mitigating memorization in large language models.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. "Physics of language models: Part 3.3, knowledge capacity scaling laws". In: *arXiv preprint arXiv:2404.05405* (2024).

[2] Nicholas Carlini et al. "Quantifying memorization across neural language models". In: *The Eleventh International Conference on Learning Representations*. 2022.

[3] Grégoire Delétang et al. *Language Modeling Is Compression*. 2024. arXiv: 2309.10668 [cs.LG]. URL: https://arxiv.org/abs/2309.10668.

[4] Sunny Duan et al. "Uncovering latent memories: Assessing data leakage and memorization patterns in frontier ai models". In: *arXiv preprint arXiv:2406.14549* (2024).

[5] Yizhan Huang et al. "Entropy-Memorization Law: Evaluating Memorization Difficulty of Data in LLMs". In: *arXiv preprint arXiv:2507.06056* (2025).

[6] Eugene Kharitonov, Marco Baroni, and Dieuwke Hupkes. "How bpe affects memorization in transformers". In: *arXiv preprint arXiv:2110.02782* (2021).

[7] Katherine Lee et al. "Deduplicating training data makes language models better". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 2022.

[8] Stephen Merity et al. *Pointer Sentinel Mixture Models*. 2016. arXiv: 1609.07843 [cs.CL].

[9] John X Morris et al. "How much do language models memorize?" In: *arXiv preprint arXiv:2505.24832* (2025).

[10] Catherine Olsson et al. "In-context learning and induction heads". In: *arXiv preprint arXiv:2209.11895* (2022).

[11] Madhur Panwar et al. "For Better or for Worse, Transformers Seek Patterns for Memorization". In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: https://openreview.net/forum?id=98NrkXPRZ9.

[12] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[13] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. 2016. arXiv: 1508.07909 [cs.CL]. URL: https://arxiv.org/abs/1508.07909.

[14] Till Speicher et al. "Understanding memorisation in llms: Dynamics, influencing factors, and implications". In: *arXiv preprint arXiv:2407.19262* (2024).

[15] Niklas Stoehr et al. "Localizing paragraph memorization in language models". In: *arXiv preprint arXiv:2403.19851* (2024).

[16] Kushal Tirumala et al. "Memorization without overfitting: Analyzing the training dynamics of large language models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 38274–38290.

[17] Chandra Shekhara Kaushik Valmeekam et al. *LLMZip: Lossless Text Compression using Large Language Models*. 2023. arXiv: 2306.04050 [cs.IT]. URL: https://arxiv.org/abs/2306.04050.

[18] Ashish Vaswani et al. "Attention is All You Need". In: 2017. URL: https://arxiv.org/pdf/1706.03762.pdf.

[19] Chiyuan Zhang et al. "Counterfactual memorization in neural language models". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023.