TOOL USE IS PROVABLY MORE SCALABLE THAN IN-WEIGHT MEMORY FOR LARGE LANGUAGE MODELS

Sam Houliston* ETH Zürich
Ambroise Odonnat* Inria, Univ. Rennes 2
Charles Arnal* FAIR at Meta
Vivien Cabannes* FAIR at Meta

ABSTRACT

Tool-augmented language models, equipped with retrieval, memory, or external APIs, are reshaping AI. Yet, their theoretical advantages remain underexplored. In this paper, we address this question by demonstrating the benefits of *in-tool learning* (external retrieval) over *in-weight learning* (memorization) for factual recall. We show that the number of facts a model can memorize solely in its weights is fundamentally limited by its parameter count. In contrast, we prove that tool-use enables unbounded factual recall via a simple and efficient circuit construction. These results are validated in controlled experiments, where tool-using models consistently outperform memorizing ones. We further show that for pretrained large language models, teaching tool-use and general rules is more effective than finetuning facts into memory. Our work provides both a theoretical and empirical foundation, establishing why tool-augmented workflows are not just practical, but provably more scalable.

`ambroiseodt/itl`

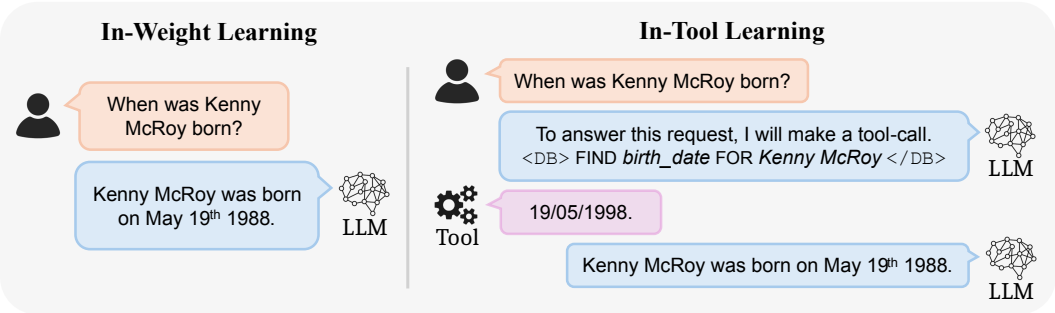

Figure 1: **Illustration of factual recall via *in-weight learning* versus *in-tool learning*.** While external retrieval may incur higher latency, it offloads storage, enabling scalable recall without increasing model size, and better preserves prior model capabilities.

1 INTRODUCTION

Large language models (LLMs) have revolutionized artificial intelligence, redefining how machines understand and generate human language (Achiam et al., 2024; Comanici et al., 2025; Grattafiori et al., 2024). Beyond this, LLMs are rapidly evolving from static predictors into dynamic, context-aware systems capable of reasoning, adapting, and acting over time. Coding assistants are accelerating software development and lowering the barrier to entry for programming, hinting at the emergence of highly automated, agentic workflows. Advances in architecture and interaction design drive this transformation. Retrieval-augmented generation (RAG, Lewis et al., 2020b) enables models to access external knowledge in real time, grounding their responses in contextually relevant information. In parallel, memory augmentation, through the use of scratchpads or memory modules, empowers models to organize their reasoning, break down problems, iteratively refine outputs, and maintain coherence across extended sequences.

*Equal contribution. Correspondence to vivc@meta.com

Together, these capabilities mark a shift away from purely in-weight learning, in which all knowledge and reasoning must be encoded within the parameters of a model, toward more modular and interpretable systems that can use tools, access external information, and leverage structured memory. Our study investigates the theoretical foundation for why such tool-augmented approaches outperform traditional monolithic models. This evolution raises natural questions: *What is the most efficient way a model can acquire and utilize knowledge? Should facts be internalized via parameter updates, or should models be taught to access and manipulate external sources of truth?*

At the heart of these questions lies a core tradeoff between *in-weight learning*, where information is compressed into the model's parameters during training, and *in-tool learning*, where the model learns to interact with external resources such as databases or APIs to retrieve information when needed (see Fig. 1 for an illustration). While the former is bounded by the model's capacity and sensitive to forgetting, the latter offers the potential for open-ended knowledge access, generalization, and interpretability. In this work, we formalize this tradeoff and provide a rigorous theoretical framework for understanding the benefits of tool use in large language models.

Contributions. Our main contributions are as follows:

1. **In-weight limitations:** We derive a theoretical lower bound showing that the parameter requirement grows linearly with the number of facts regardless of desired recall or data compressibility. This highlights a structural bottleneck of in-weight memorization (Section 3).
2. **In-tool benefits:** We provide an explicit upper bound showing that tool-augmented models can, in principle, retrieve an unbounded number of facts by learning to interface with an external database. This result is supported by a formal circuit construction demonstrating the feasibility and efficiency of such tool-use (Section 4).
3. **Controlled experiments:** We validate these theoretical results in a controlled experimental setting, where models are pretrained from scratch either to memorize facts or to learn to use an external tool. This setting reveals the learning mechanics of tool learning, and our findings empirically confirm the predicted scaling laws and memory limitations (Section 5.1).
4. **Large-scale experiments:** We demonstrate the practical implications of our findings on existing LLMs, showing that finetuning to learn new facts is significantly less effective than teaching models how to access structured tools or infer generalizable rules. This reinforces the argument that future LLM development should prioritize tool-use capabilities over rote memorization (Section 5.2).

Together, these results provide both a conceptual and empirical foundation to better understand why models that use tools outperform those that attempt to internalize all the knowledge. Our findings motivate a shift in model design philosophy: from training ever-larger monolithic models toward building modular systems that learn to query, not just to store information.

Related work. Recent papers have increasingly explored tool-augmented language models. This trend is evident in the open-source literature, with models such as Toolformer (Schick et al., 2023), ReAct (Yao et al., 2023), and HuggingGPT (Shen et al., 2023), as well as in industrial systems that now support function calling, enabling modular LLM-agent architectures. External memory mechanisms such as RAG (Lewis et al., 2020c), RETRO (Borgeaud et al., 2022), and retrieval-augmented transformers (Shao et al., 2024) serve as tools in their own right, enhancing knowledge access. In a similar spirit, scratchpads (Nye et al., 2021) and chain-of-thought prompting (Wei et al., 2022) externalize reasoning and expand model capacity (Besta et al., 2025). This is not surprising given the recent studies that quantify the limitations of in-weight knowledge (Allen-Zhu & Li, 2025c; Cabannes et al., 2025; Hahn, 2020; Merrill & Sabharwal, 2024), and highlight interference during editing (Dai et al., 2022; Meng et al., 2022). Taken together, these works form a growing line of empirical methods that depart from the in-weight learning through sophisticated mechanisms for retrieval, reasoning, and tool use. Given this empirical success, we aim to investigate in a principled manner the benefits of tool-augmented workflows, thus contributing to a broader effort towards better understanding large language models (see, e.g., Beirami et al., 2025; Bietti et al., 2023; Zekri et al., 2025). We provide additional related work in Appendix A for the interested reader.

2 METHODOLOGY

To highlight the differences between in-weight memorization and tool-augmented retrieval, we need to precisely quantify the amount of knowledge contained in a given dataset. Since there is no simple solution to do that (e.g., one could choose the total number of bits or use the notion of Kolmogorov complexity), we overcome this issue by following the literature on the *Physics of Language Models* (Allen-Zhu & Li, 2024, 2025a,c) and introducing a family of factual recall tasks. Let $\mathcal{D}$ be a family of datasets, where each dataset $D \in \mathcal{D}$ is a finite collection of facts. Each fact is a triplet (n, a, v) of strings, consisting of a name $n \in \mathcal{N}$ (e.g., "Kenny McRoy"), an attribute $a \in \mathcal{A}$ (e.g., "birthplace"), and a value $v \in \mathcal{V}_a$ (e.g., "Germany"). Each dataset D defines a mapping $V_D : (n, a) \mapsto v$, which assigns to each name–attribute pair its corresponding value.

Factual recall. The goal of the recall task is to produce the ground-truth value $V_D(n, a)$ when given a query (n, a). Let $\mathcal{M}$ be a class of models, where each model $f \in \mathcal{M}$ implements a recall rule $R(f) : \mathcal{N} \times \mathcal{A} \to \bigcup_{a \in \mathcal{A}} \mathcal{V}_a$. For instance, the model class $\mathcal{M}$ may be the set of all transformers with a given architecture, each model $f \in \mathcal{M}$ corresponding to a specific choice of weights, and the recall rule $R(f)$ being defined by prompting the model with a query asking for the value associated to a pair (n, a) and parsing the text obtained by auto-regressively sampling from the model until an end-of-sequence token is emitted.

Definition 2.1 (Learnability)**.** *We say that the model class $\mathcal{M}$* can learn *the recall task associated to a family of datasets $\mathcal{D}$ if, for each dataset $D \in \mathcal{D}$, there exists a model $f \in \mathcal{M}$ such that f achieves perfect accuracy on the value recall task for D, i.e. $R(f) = V_D$.*

Note that this definition abstracts away the training process and does not account for sample efficiency or optimization difficulty.

Sentence format. Our study concerns itself with factual recall performed by LLMs via question answering. We define a dataset with a strict grammar that specifies the format of all question–answer pairs. In the *in-weight* setting, the model is trained to directly generate the answer from its parameters. The user produces a query string Q using the format $Q = \varphi_1(a) \circ \varphi_2(n) \circ \varphi_3(a)$, and the assistant must then generate the answer string A in the form $A = \psi_1(n) \circ \psi_2(a) \circ \psi_3(v)$, where φ_i and ψ_i are deterministic string functions (i.e. templates) that encode the structure of the input and output, and $\circ$ denotes string concatenation. For example, given the previous fact example, we have:

$$Q = \underbrace{\text{Where was}}_{\varphi_1(a)} \ \underbrace{\text{Kenny McRoy}}_{\varphi_2(n)=n} \ \underbrace{\text{born?}}_{\varphi_3(a)}$$

$$A = \underbrace{\text{Kenny McRoy}}_{\psi_1(n)=n} \ \underbrace{\text{was born in}}_{\psi_2(a)} \ \underbrace{\text{Germany.}}_{\psi_3(v)=v}$$

In the *in-tool* setting, the model learns to issue a structured tool query that retrieves the value from an external database. The user query Q remains unchanged, but the assistant must first produce an intermediate tool query string $T = \chi_1(a) \circ \chi_2(n)$, where the χ_i are deterministic string templates encoding the attribute and name in a format expected by the tool interface. This query is sent to an external retrieval system (e.g., a database or API), which returns the corresponding value $\xi(f, a, n)$, where $\xi(f, a, n) = v$ if the assistant correctly formats the query, and an error message otherwise. The assistant then produces the final answer A, reusing the same output grammar as in the in-weight case.

For example, using the same fact example, we have:

$$T = \underbrace{\text{To answer this request, I will make a tool-call. <DB> FIND birthplace}}_{\chi_1(a)} \ \underbrace{\text{FOR Kenny McRoy </DB>}}_{\chi_2(n)},$$

This formal grammar establishes a controlled setting to characterize the information storage and retrieval capabilities of LLMs precisely. An example of queries formatted in JSON files can be found in Appendix C.1.

Remark 2.2 (Beyond factual recall)**.** *Our setting is tailored to open-ended questions such as factual recall or TriviaQA (Joshi et al., 2017) (see Section 5.2). We discuss the extension to other question-answer categories, e.g., boolean or multiple choice, via the design of templates φ, ψ in Appendix C.1.*

3 Limitations of in-weight learning

This section illustrates the limitations of in-weight learning through theoretical lower bounds. The following lemma provides a lower bound on the size of a model class that can learn a given family of datasets.

Lemma 3.1 (Capacity lower bound.)**.** *Let $\mathcal{D}$ be a family of factual datasets defined over a fixed set of names $\mathcal{N}$ and attributes $\mathcal{A}$. Let $\mathcal{M}$ be a model class that can learn $\mathcal{D}$. Then, we have*

$$|\mathcal{M}| \geq |\mathcal{D}|. \tag{1}$$

Proof. This follows from a simple enumeration argument, since the definition of learnability (Definition 2.1) is equivalent to the inclusion $\{V_D \mid D \in \mathcal{D}\} \subseteq \{R(f) \mid f \in \mathcal{M}\}$. Considering the cardinality of those sets concludes the proof. □

Theorem 3.1 implies a lower bound on the number of parameters required to define a class of models that could memorize a given set of facts. This allows us to provide a lower bound on the parameter requirement.

Theorem 3.2 (Parameter lower bound)**.** *Let $\mathcal{M}$ be a class of models with P parameters, each quantized to b bits. Let $\mathcal{D}$ be the family of all datasets defined over names $\mathcal{N}$, attributes $\mathcal{A}$, and value sets $\mathcal{V}_a$. Then, if $\mathcal{M}$ can learn $\mathcal{D}$ without accessing external tools (in-weight learning), we must have:*

$$P \geq \frac{|\mathcal{N}|}{b} \sum_{a \in \mathcal{A}} \log_2 |\mathcal{V}_a| = c \cdot \#\textit{Facts},$$

where $\#\textit{Facts} = |\mathcal{N} \times \mathcal{A}|$ is the number of facts, and c is the average of $\log_2 |\mathcal{V}_a|/b$ over $a \in \mathcal{A}$.

Proof. This is a consequence of Theorem 3.1. The total number of distinct parameter configurations in the model family $\mathcal{M}$ is at most 2^{bP} (i.e., $|\mathcal{M}| \leq 2^{bP}$). Moreover, defining any dataset from $\mathcal{D}$ requires distinguishing between all possible assignments of values for the attributes of each name. For a single name, there are $\prod_{a \in \mathcal{A}} |\mathcal{V}_a|$ possible value assignments; for all $\mathcal{N}$ names, this yields $|\mathcal{D}| = \left(\prod_{a \in \mathcal{A}} |\mathcal{V}_a|\right)^{\mathcal{N}}$. Applying Theorem 3.1 with these quantities and taking the base-2 logarithm of both sides of the inequality completes the proof. □

Theorem 3.2 shows that the number of parameters needed to memorize arbitrary fact mappings grows linearly with the number of facts. It proves a hard capacity ceiling on in-weight learning: once the number of facts exceeds a threshold determined by model size, memorization becomes impossible without architectural changes or external memory.

We note that Theorem 3.2 could be refined in two directions. First, the capacity of a model class does not necessarily grow exponentially with the number of quantization bits. In practice, empirical studies such as Allen-Zhu & Li (2025c) suggest that the effective representational capacity of language models is often much lower than the theoretical maximum–typically around $b \approx 2$ bits per parameter.

Second, real-world factual datasets arise from structured data-generating processes, in which shared latent or causal factors induce both restricted support and statistical dependence among facts. As a result, the induced distribution over datasets in $\mathcal{D}$ —or equivalently, the conditional distribution of values V given factual queries (N, A)—has lower entropy than the uniform distribution considered above. For example, knowing someone is Japanese makes it more likely that they are older than 60 than if they were from Kenya, due to population-level age differences. This suggests that the worst-case notion of learnability used in Theorem 3.2, which requires perfect memorization of all datasets in $\mathcal{D}$, is overly pessimistic.

Average-case parameter lower bound. We refine the previous result by deriving a parameter lower bound that holds in expectation over datasets $\mathcal{D}$, and allows for some predefined level of model recall error. We assume factual queries (n, a, v) are drawn i.i.d. from a distribution $\mathbb{P}_{N,A} \times \mathbb{P}_{V|N,A}$. For a model $f \in \mathcal{M}$, we define its expected 0–1 recall error as $\ell(f) := \mathbb{P}\big[R(f)(N, A) \neq V\big]$, where the probability is taken over the data-generating distribution. Denote $H(\ell) = -\ell \log_2 \ell - (1 - \ell) \log_2(1 - \ell)$ as the entropy of a Bernoulli variable with parameter ℓ. The full proof is deferred to Appendix B.1.

Theorem 3.3 (Average-case parameter bound)**.** *Let $\mathcal{M}$ be a class of models with P parameters, each quantized to b bits. If there exists a model $f \in \mathcal{M}$ achieving expected recall error $\ell(f)$ on T factual queries, then*

$$P \geq \frac{T}{b}\Big[H(V \mid N, A) - H(\ell(f)) - \ell(f) \log_2(|\mathcal{V}| - 1)\Big].$$

Theorem 3.3 captures how allowing a non-zero expected recall error $\ell(f)$ monotonically reduces the parameter requirement. Specifically, for small error rates ($\ell(f) \approx 0$), the bound decreases sharply, whereas for larger error tolerance, the reduction becomes sublinear. Naturally, the bound fully depends on the conditional entropy of values $H(V \mid N, A)$, which may be reduced from a smaller support of values or a "tighter" distribution, as discussed above. These dependencies are illustrated in Fig. 8. In particular, when $\ell(f) = 0$ and the entropy $H(V \mid N, A)$ is maximal, the bound exactly recovers the worst-case result. Despite these refinements, the average-case parameter requirement for in-weight learning remains linear in the number of effectively learnable facts.

4 Benefits of in-tool learning

This section showcases that tool use allows models, and in particular transformers (Vaswani et al., 2017), to circumvent the limitations of in-weight learning and accurately recall an unlimited number of facts without needing additional parameters. More specifically, we exhibit an upper bound on the number of parameters needed for a transformer to implement the kind of database querying described in Section 2.

Let $\mathcal{D}$ be a family of factual datasets defined over some names, attributes, and value sets $\mathcal{N}, \mathcal{A}, \{\mathcal{V}_a\}_{a \in \mathcal{A}}$. We introduce the following notion of *query-based learnability* for a model augmented with access to an external database.

Definition 4.1 (Query-based Learnability)**.** *For a factual dataset D, the associated* retrieval system *R_D is a recall rule which can be queried in the manner described in Section 2 to retrieve any fact of D. We say that a tool-augmented model f solves the recall task for a family of datasets $\mathcal{D}$ if, for any $D \in \mathcal{D}$, it achieves perfect accuracy on the value recall task for D by querying R_D.*

Note that Definition 4.1 is stronger than Definition 2.1, as a single model (rather than a family of models) is tasked with "learning" any dataset $D \in \mathcal{D}$, provided it has access to a correct retrieval system. In the theorem below, we consider Llama3-like transformers (Grattafiori et al., 2024) with a fixed token vocabulary; we make a few mild technical assumptions, e.g., the queries and answers are shorter than the maximum model context length, which we detail in Appendix B.2, along with the proof of the statement.

Theorem 4.2 (Parameter upper bound)**.** *Let $\mathcal{D}$ be a family of factual datasets defined over some names, attributes, and value sets $\mathcal{N}, \mathcal{A}, \{\mathcal{V}_a\}_{a \in \mathcal{A}}$. Given $\mathcal{R}_\mathcal{D} = \{\mathcal{R}_\mathcal{D}\}_{\mathcal{D} \in \mathcal{D}}$ a family of retrieval systems associated to $\mathcal{D}$, there exists a transformer with at most 8 transformer blocks, an embedding dimension of at most $O(|\mathcal{A}|)$ and a total number of parameters at most $O(|\mathcal{A}|^2)$ that can solve the value recall task for $\mathcal{D}$ if equipped with the ability to access $\mathcal{R}_\mathcal{D}$.*

Note that the bounds do not depend on the number of names or possible values. While our theoretical results provide foundational insights, empirical validation remains crucial, since Theorem 3.2

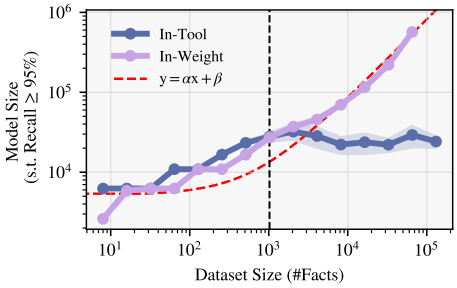

Figure 2: **Parameter requirements.** Minimum size to achieve 95% recall, versus training dataset size (10 seeds with standard errors). In-weight models require an approximately linear $(\alpha, \beta) = (8.14, 5171)$ increase in parameters. In contrast, the model size for in-tool models plateaus, once the models learn to master the tool (see Fig. 3).

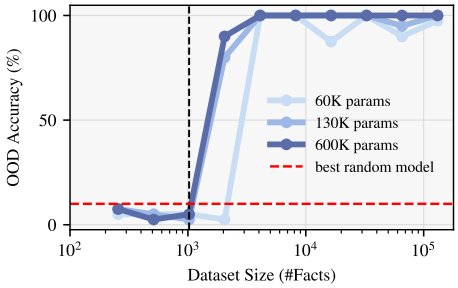

Figure 3: **Transition from memorization to retrieval in in-tool models.** Accuracy on unseen facts versus training dataset size. When trained on small datasets, tool-using models default to pure memorization. In this setting, the models properly learn to construct tool queries for datasets above 10^3 facts, enabling recall of unseen facts.

describes a worst-case scenario, and Theorem 4.2 is an existence result that does not account for optimization difficulty.

5 Experiments

In this section, we empirically validate the results of Sections 3 and 4, showcasing that in-tool learning leads to better performance at a lower memory load. The code and reproducibility instructions can be found at `https://github.com/ambroiseodt/itl`.

5.1 Controlled study

We pretrain small language models and compare the two training regimes illustrated in Fig. 1:

(i) *in-weight*: the model directly generates the answer using the knowledge integrated in its parameters,
(ii) *in-tool*: the model first queries an external database, which returns the requested value. The model then formats it into an answer,

and show that our findings support the theory: tool-augmented recall outperforms in-weight memorization in terms of the number of facts learned per parameter.

Setup. We pretrain small Llama3 models (Grattafiori et al., 2024) (2K to 0.6M parameters) from scratch on synthetic biographical databases with a fixed list of names $\mathcal{N}$ and four attributes $\mathcal{A}$: "birth place", "birth date", "current address", "occupation". This yields $4|\mathcal{N}|$ atomic facts per dataset instance. Our models are trained for up to 100,000 steps using the AdamW optimizer (Loshchilov & Hutter, 2019) with a batch size of 128 samples and a cosine learning rate scheduler. See Appendix C.2 for details.

Theory validation. Our empirical results support the theoretical predictions presented in Sections 3 and 4. As shown in Fig. 2, the in-weight regime exhibits an unbounded increase in the number of required parameters as the number of facts grows. This behavior demonstrates the existence of a lower bound on parameter requirements, consistent with the general claim of Theorem 3.2. Note that the empirical scaling is slightly sublinear for small dataset sizes, approximately following a square-root trend. This suggests the presence of subtle phenomena not captured by the theorem, which we leave to future work. This confirms that parameter count imposes a hard limit on memorization-based learning (we also confirm the impact of the recall on the parameter requirement from Theorem 3.3 in Fig. 11). In contrast, the in-tool regime displays a clear transition: after a critical dataset size

(around 1,000 facts in our setting), the parameter requirement saturates and remains nearly constant. This plateau confirms that the model has shifted from memorizing facts to delegating recall to an external database, in line with the existence results in Theorem 4.2.

From memorization to rule learning. This transition in the in-tool regime reflects more than a capacity shift; it reveals a learning phase transition. Initially, models faced with tool-augmented recall resort to memorizing fact–answer pairs, similar to the in-weight regime. However, once exposed to a sufficient diversity of facts, the model discovers the underlying logic for constructing tool queries. As shown in Fig. 3, where the best OOD accuracy achieved by in-tool models is displayed, this results in a dramatic improvement in generalization to out-of-distribution facts. This phenomenon mirrors the "grokking" effect observed in other settings: a delayed but sharp shift from brute-force memorization to systematic rule learning (Nanda et al., 2023; Power et al., 2022). Once the model learns the correct format for querying, retrieval generalizes across databases, decoupling recall accuracy from the number of stored facts.

What counts as a fact? Our study so far assumes a clean distinction between facts (e.g., "Kenny McRoy was born in Germany") and rules (e.g., "birthplace queries can be answered by a lookup function"). In our attribution task, this boundary is sharp, but in broader domains such as mathematics or commonsense reasoning, the line between facts and rules is often ambiguous. Are theorems facts, or derived consequences of axioms? Ontologies may reveal intricate interdependencies that blur this distinction. To explore how structure affects memorization, we introduce controlled correlations between facts using a parameter $\alpha \in [0, 1]$. For each pair name-attribute (n, a), we draw a Bernoulli variable $X \sim \text{Bern}(\alpha)$, and set $V_D(n, a) = V_1(n, a)$ if $X = 1$, or $V_2(n, a)$ otherwise, where, V_1 assigns values based on the family name, while V_2 assigns values independently at random. As such, $\alpha = 1$ amounts to family members sharing a birthplace, current address, and occupation, while $\alpha = 0$ corresponds to the unstructured case used in earlier experiments. As shown in Fig. 4, increasing the correlation reduces the number of parameters required for in-weight models to achieve high recall. This effect concurs with Theorem 3.3, our results suggest that transformers can adapt to this structure, effectively compressing related facts through shared patterns and achieving high recall with fewer parameters. This aligns with previous findings on how structural assumptions about the data influence learning in transformers and neural networks more broadly (Arnal et al., 2024; Dziri et al., 2023; Mahajan et al., 2025; Valvoda et al., 2022; Wang et al., 2025).

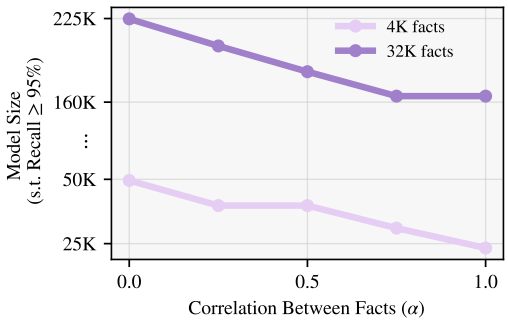

Figure 4: **Impact of correlation in facts on in-weight memorization.** Training data is generated with varying levels α of correlation between sampled attributes and values. Common patterns in facts make the datasets more compressible, reducing the in-weight parameter requirement, as predicted by Theorem 3.3.

5.2 Large-scale study

We show that the limitations of in-weight learning persist in real-world settings. In particular, despite large models being able to store vast amounts of information in their parameters, introducing new knowledge through finetuning impacts the existing capabilities due to finite capacity. In contrast, tool-augmented learning promises scalability without forgetting. We provide further related results in Appendix D.2.

Setup. We fine-tune instruction-tuned language models on synthetic factual databases until they reach at least 95% recall. We use the same training setup as in Section 5.1, with learning rates adapted to the model size. The models considered are SmolLM 2 Instruct (135M, 360M, 1.7B) (Allal et al., 2025) and Llama 3.1/3.2 Instruct (1B, 3B, 8B) (Grattafiori et al., 2024). Checkpoints are

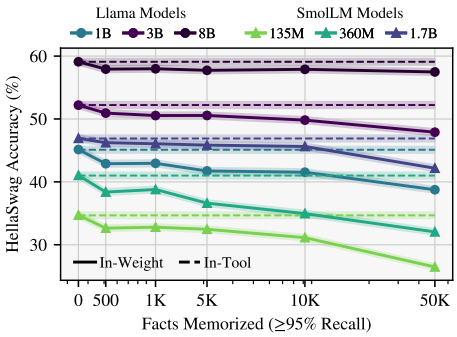

Figure 5: **Impact of factual memorization on general capabilities (HellaSwag)**. In-weight memory impacts general capabilities as the memory load increases. Tool-learning preserves prior capabilities almost perfectly. Overall, larger models suffer relatively less.

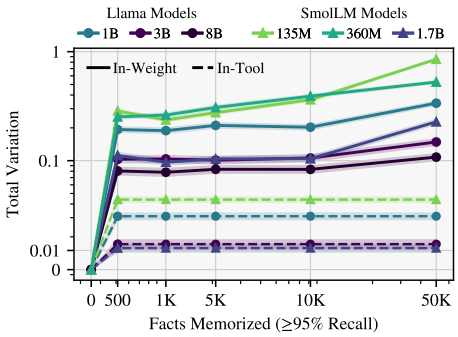

Figure 6: **Total variation distance induced by memorization load**. The setup is similar to Fig. 5. Models finetuned with in-weight learning deviate from their corresponding base models when memorizing bigger datasets, while in-tool models remain close.

evaluated across training to measure the factual recall, the Hellaswag accuracy (Zellers et al., 2019) used as a proxy for general language abilities, and the total variation (TV) distance between the output distributions of the base model and the finetuned model. See Appendix C.3 for details.

Overloading threatens general capabilities. Motivated by our theoretical results, we now investigate whether the benefits of in-tool learning persist in large, pretrained language models. Fig. 5 shows how HellaSwag accuracy evolves as models are trained to memorize increasing numbers of facts. In-tool learning (dashed lines) retains general performance across all scales, confirming that externalizing factual storage prevents interference. In contrast, in-weight learning (solid lines) shows a clear degradation, especially in smaller architectures. This supports our theoretical intuition: parameter-based memorization consumes capacity and risks overwriting prior knowledge. Note that larger models degrade more slowly, suggesting they can absorb new information with less interference. These results show the scalability advantage of tool-use to preserve general capabilities.

Token distribution change. The differences between in-weight and in-tool learning can be further understood by measuring how much each method alters the model's output distribution. As shown in Fig. 6, in-weight learning leads to significant drift—especially in smaller models—with total variation (TV) distance growing sharply with the number of memorized facts. In contrast, tool-learning causes minimal divergence, even at large scales. This offers a complementary perspective on our results: in-weight learning requires shifting the next-token distribution to encode new information, while in-tool learning preserves the model's original behavior by delegating recall.

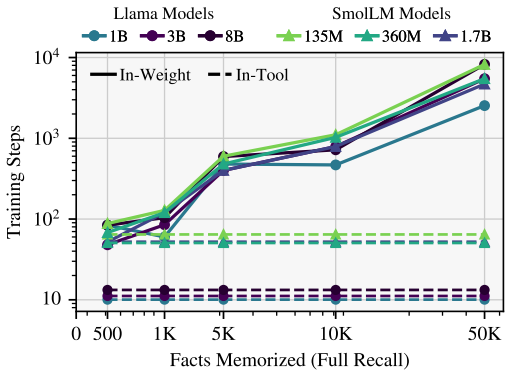

Figure 7: **Training steps required for memorization**. In-weight compute requirements scale with the memorization load, unlike tool models. Llama models are better predisposed to learn tool-use from their pretraining compared to SmolLM models.

Efficiency trade-offs. Another advantage of in-tool learning is its training efficiency: As shown in Fig. 7, models acquire the tool-use function call pattern in fewer than 20 steps, as soon as the set is large enough to elicit tool-use. In contrast, in-weight learning requires substantially more training to memorize individual facts. However, this efficiency comes with a trade-off: In-weight models support cheaper inference, producing answers directly without generating intermediate

Table 1: **Tool-augmented models preserve their capabilities**. For the same level of recall on open-ended questions (87.5% on TriviaQA), in-weight models suffer a loss in prior knowledge (HellaSwag scores below) compared to tool-augmented models that use a Python interpreter via system prompt instructions.

method	1K	5K	10K	50K
in-weight	59%	58%	58%	57%
in-tool	59%	**59%**	**59%**	**59%**

tool queries or relying on external calls. We also observe that tool-use can be slightly unstable to train; early stopping proved effective to prevent capability loss once the tool behavior was acquired.

Experiments on TriviaQA. It is standard practice to finetune LLMs to give them tool-use capabilities (Schick et al., 2023). We further demonstrate the benefits of tool use in another real-world setting where a strong model is prompted to use the tool. Since the model is not finetuned, previous capabilities on Hellaswag are necessarily preserved. With Llama3 8B and a Python interpreter, this approach results in an accuracy of 87.5% on a left-out subset of 1K questions from TriviaQA (Joshi et al., 2017). For in-weight learning, we conduct a similar study as before: we finetune Llama3 8B in a supervised fashion on an increasing amount of TriviaQA samples until it matches the in-tool accuracy of 87.5% on the left-out subset. We recover the performance on Hellaswag for each configuration. Results are displayed in Table 1. We observe a similar behavior to that in Fig. 5, where the capabilities of in-tool models are preserved while they decrease for in-weight models as more knowledge is integrated in the weights.

6 Conclusion

Mieux vaut une tête bien faite qu'une tête bien pleine.

Montaigne, Les Essais

This work provides a unified theoretical and empirical account of the tradeoff between in-weight and in-tool learning. We demonstrate that the number of facts a model can store in its parameters is bounded by its size. It implies that scaling knowledge capacity through larger models is inherently inefficient. In contrast, models that learn to interact with external tools can access unbounded factual knowledge without increasing their parameter count. We experimentally validate these insights: while in-weight models require ever-larger architectures to memorize growing datasets, tool-augmented models exhibit a phase transition, rapidly shifting to rule-based querying once sufficient diversity is observed. This decouples memory capacity from model size. Large-scale experiments extend these findings to pretrained models, showing that in-weight finetuning for factual recall degrades general capabilities, a consequence of limited capacity forcing new information to overwrite prior knowledge. Tool-based approaches, by externalizing factual storage, preserve core skills, reduce training costs, and introduce minimal behavioral drift. These results underscore a key design insight: language models scale more effectively by learning to access and orchestrate information, not by internalizing it. Models evolve from static predictors to systems capable of modular interaction with structured external resources.

Limitations and future work. Our analysis provides conceptual clarity and theoretical insights through carefully controlled settings. Although insightful, such synthetic datasets may not fully reflect the complexity of real-world knowledge. Integrating the optimization dynamics in the theoretical bounds is a promising research direction. In addition, extending our analysis to richer settings with computer use, reasoning trace generation, or learnable memory modules integrated into the model architecture is a promising direction for future work.

IMPACT STATEMENT

We aim to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, and Mohammad Bavarian et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: part 3.1, knowledge storage and extraction. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=FxNNiUgtfa.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=oDbiL9CLoS.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=FxNNiUgtfa.

Charles Arnal, Clement Berenfeld, Simon Rosenber, and Vivien Cabannes. Learning with hidden factorial structure, 2024. URL https://arxiv.org/abs/2411.01375.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander Nicholas D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=u3U8qzFV7w.

Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, Tomasz Sternal, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Łukasz Flis, Hannes Eberhard, Zixuan Chen, Hubert Niewiadomski, and Torsten Hoefler. Reasoning language models: A blueprint, 2025. URL https://arxiv.org/abs/2501.11223.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=3X2EbBLNsk.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Vivien Cabannes, Berfin Şimşek, and Alberto Bietti. Learning associative memories with gradient descent. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Vivien Cabannes, Elvis Dohmatob, and Alberto Bietti. Scaling laws for associative memories. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=Tzh6xAJSll.

Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Alice Yang, Francois Charton, and Julia Kempe. Iteration head: a mechanistic study of chain-of-thought. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, and Aaron Parisi et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Thomas M. Cover and Joy A. Thomas. *Rate Distortion Theory*, chapter 10, pp. 301–346. John Wiley & Sons, Ltd, 2005. ISBN 9780471748823. doi: https://doi.org/10.1002/047174882X.ch10.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581/.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 933–941. JMLR.org, 2017.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: limits of transformers on compositionality. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning LLMs on new knowledge encourage hallucinations? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7765–7784, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.444. URL https://aclanthology.org/2024.emnlp-main.444/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, and Llama Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, December 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00306. URL http://dx.doi.org/10.1162/tacl_a_00306.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Sam Houliston, Alizée Pace, Alexander Immer, and Gunnar Rätsch. Uncertainty-penalized direct preference optimization, 2024. URL https://arxiv.org/abs/2410.20187.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. Can knowledge editing really correct hallucinations? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hmDt068MoZ.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020. doi: 10.1162/tacl_a_00324. URL https://aclanthology.org/2020.tacl-1.28/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, 2020a. ISBN 9781713829546.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020b. Curran Associates Inc. ISBN 9781713829546.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020c. Curran Associates Inc. ISBN 9781713829546.

Danny D. Leybzon and Corentin Kervadec. Learning, forgetting, remembering: Insights from tracking LLM memorization during training. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen (eds.), *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 43–57, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.4. URL https://aclanthology.org/2024.blackboxnlp-1.4/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Divyat Mahajan, Mohammad Pezeshki, Charles Arnal, Ioannis Mitliagkas, Kartik Ahuja, and Pascal Vincent. Compositional risk minimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=axN83wi3NH.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models, 2021. URL https://arxiv.org/abs/2112.00114.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models, 2023. URL https://arxiv.org/abs/2303.09014.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250/.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference, 2022. URL https://arxiv.org/abs/2211.05102.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022. URL https://arxiv.org/abs/2201.02177.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-rong Wen. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8), January 2025. ISSN 2095-2236. doi: 10.1007/s11704-024-40678-2. URL http://dx.doi.org/10.1007/s11704-024-40678-2.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5418–5426. Association for Computational Linguistics, November 2020. doi: 10.18653/v1/2020.emnlp-main.437. URL https://aclanthology.org/2020.emnlp-main.437/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang Wei Koh. Scaling retrieval-based language models with a trillion-token datastore. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=iAkhPz7Qt3.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with chatGPT and its friends in hugging face. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=yHdTscY6Ci.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Josef Valvoda, Naomi Saphra, Jonathan Rawski, Adina Williams, and Ryan Cotterell. Benchmarking compositionality with formal languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 6007–6018, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.525/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL https://github.com/facebookresearch/lingua.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: a mechanistic journey to the edge of generalization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Knowledge mechanisms in large language models: A survey and perspective. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7097–7135, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.416. URL https://aclanthology.org/2024.findings-emnlp.416/.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3), November 2024b. ISSN 0360-0300. doi: 10.1145/3698590. URL https://doi.org/10.1145/3698590.

Xiaohan Wang, Dian Li, Yilin Zhao, Sinbadliu, and Hui Wang. Metatool: Facilitating large language models to master tools with meta-task augmentation, 2024c. URL https://arxiv.org/abs/2407.12871.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Oussama Zekri, Ambroise Odonnat, Abdelhakim Benechehab, Linus Bleistein, Nicolas Boullé, and Ievgen Redko. Large language models as markov chains, 2025. URL https://arxiv.org/abs/2410.02724.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and

Huajun Chen. A comprehensive study of knowledge editing for large language models, 2024. URL https://arxiv.org/abs/2401.01286.

Zihan Zhang, Meng Fang, Ling Chen, Mohammad-Reza Namazi-Rad, and Jun Wang. How do large language models capture the ever-changing world knowledge? a review of recent advances. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8289–8311, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.516. URL https://aclanthology.org/2023.emnlp-main.516/.

APPENDIX

A Extended related work **17**

A.1 Memorization in language models 17

A.2 Tool-use in language models. 18

B Proofs **19**

B.1 Proof of Theorem 3.3 19

B.2 Proof of Theorem 4.2 21

C Implementation details **25**

C.1 Factual recall database 25

C.2 Controlled setting 25

C.3 Large-scale setting 27

D Additional results **28**

D.1 Controlled experiments 28

D.2 Large-scale experiments 28

A EXTENDED RELATED WORK

A.1 MEMORIZATION IN LANGUAGE MODELS

The "Physics of LLMs" series (Parts 3.1–3.3) offers a comprehensive and theoretically grounded account of various aspects of memorization with LLMs. Part 3.1 (Allen-Zhu & Li, 2024) covers knowledge storage, distinguishing between the storage of facts and whether they are extractable in practice. With controlled experiments on synthetic biographies, the authors showed that training on diverse or augmented forms of facts (such as paraphrasing or sentence re-ordering) is crucial for facts to be recoverable via prompting. This is done with the use of linear probing to diagnose where facts are encoded in hidden states. Part 3.2 – Knowledge Manipulation (Allen-Zhu & Li, 2025b) extends this discussion beyond storage and extraction by evaluating whether LLMs can flexibly use stored facts for downstream reasoning. Four factual manipulation tasks are defined (retrieval, classification, comparison, and inverse search) to show that LLMs, including GPT-4, are competent only at direct retrieval. Lastly, Part 3.3 – Knowledge Capacity Scaling Laws (Allen-Zhu & Li, 2025a) quantifies the knowledge storage limits of LLMs, demonstrating a consistent capacity of 2 bits per parameter across various models. Their experimental design spans multiple architectures, quantization, and positional encoding methods.

Links between memorization capacity and model size. Roberts et al. (2020) investigates the extent to which factual knowledge can be stored in model parameters by fine-tuning language models of varying sizes on closed-book QA tasks. They find that performance scales with the model size, with larger models being able to approach the accuracy of retrieval-based systems. Allen-Zhu & Li (2025a) present exhaustive experiments in a controlled setting by fine-tuning models on synthetic biographical datasets under different conditions (training duration, model architectures, quantization, and data augmentations) and conclude that most models achieve a knowledge capacity of 2 bits per parameter regardless of quantization.

Wider works on factual memorization. Early empirical studies establish that pretrained LMs encode relational and factual knowledge. Petroni et al. (2019) shows that BERT can answer cloze-style queries competitively with supervised baselines, even without fine-tuning. Radford et al. (2019) further demonstrates that autoregressive models like GPT-2 acquire zero-shot QA abilities through scale alone, with performance improving log-linearly with model size. Subsequent work formalizes these behaviors. Wang et al. (2024a) propose a taxonomy of memorization, comprehension, and application, highlighting the fragility of in-weight knowledge. Kadavath et al. (2022) show that, to some extent, LLMs can self-assess their factual correctness via self-evaluation metrics $P(\text{True})$ and $P(\text{I Know})$. Such estimations are found to improve with model size or with ensembles. These findings inform recent work on confidence modeling and selective generation. Leybzon & Kervadec (2024), the authors track training dynamics and find that factual memorization is highest early and late in training, with mid-training phases marked by forgetting. These observations have implications for curriculum learning and data curation strategies. Finally, Kandpal et al. (2023) links factual recall and the frequency of that information in pretraining corpora. The authors show that models underperform on rare facts, even at a large scale, pointing to a data coverage bottleneck and motivating retrieval-augmented approaches.

Knowledge extraction from LLMs. Prompt design plays a central role in retrieving stored knowledge. Jiang et al. (2020) shows that standard close prompts underestimate LLM knowledge and proposes automated prompt mining and paraphrasing to improve extraction accuracy on the LAMA benchmark. Similarly, Allen-Zhu & Li (2025c) demonstrates that without sufficient factual augmentation, stored facts may remain unrecoverable, highlighting a gap between memorization and extractability. Both works stress that retrieval performance depends on prompt diversity and internal representations.

Learning novel knowledge and avoiding forgetting. LLMs face challenges when integrating new knowledge post-pretraining. The authors of Gekhman et al. (2024) demonstrate that fine-tuning

on novel facts is both inefficient and destabilizing: new knowledge is learned slowly. It increases hallucination rates, suggesting that fine-tuning may conflict with pre-trained knowledge. The study in Allen-Zhu & Li (2024) echoes this by showing that only sufficiently augmented knowledge is robustly encoded and extractable. The broader review in Zhang et al. (2023) surveys continual learning methods, emphasizing the need for non-destructive updates that preserve anterior capabilities. Collectively, these works highlight the fragility of in-weight representations when adapting to novel information.

Knowledge editing. Another line of research considers the editing of stored knowledge in LLMs via direct mechanistic intervention on the weights, aiming to alleviate the computational burden of retraining. A wide range of editing techniques is reviewed by Zhang et al. (2024). Wang et al. (2024b) frame editing techniques in terms of cognitive-inspired mechanisms (external injection, model merging, and intrinsic editing), and introduce a unified benchmark (KnowEdit) alongside analyses of knowledge localization. Huang et al. (2025) tackle the core question of whether knowledge editing can correct hallucinations, introducing HalluEditBench and evaluating editing methods along multiple axes such as locality and generalization. Their results underscore both the promise and limitations of editing techniques in practice. Complementarily, Part 3.2 of the Physics of LLMs series (Allen-Zhu & Li, 2025b) highlights a deeper structural limitation: even when facts are faithfully stored in the model, their downstream manipulation (e.g., comparison or inverse search) remains brittle—suggesting that editing alone may not grant models flexible reasoning over internalized knowledge.

Mechanistic understanding of memorization. A series of recent works uncovers the mechanistic basis of how transformers store and recall information, with a particular focus on two mechanisms: induction heads, which enable in-context learning by attending to repeated structures within a prompt, and associative memories, where weight matrices implicitly store token associations via outer products. Bietti et al. (2023) analyzes a simplified training setup, showing that transformers first learn global bigram statistics before gradually developing induction heads that enable pattern completion from local context. They also interpret certain weight matrices as associative memories formed during training. Building on this, Cabannes et al. (2024a) provides a theoretical analysis of gradient-based learning in associative memory modules, revealing oscillatory learning dynamics caused by imbalanced token frequencies and correlated embeddings. These oscillations affect how quickly and reliably memories are stored. Cabannes et al. (2024b) derive scaling laws for memory capacity and retrieval reliability, validating them empirically in small transformer models. Collectively, these works offer a theoretical and empirical foundation for how memorization emerges through optimization dynamics in LLMs.

A.2 Tool-use in language models.

Retrieval-augmented generation (RAG) (Lewis et al., 2020a) represents an early and influential line of work that treats external tools (e.g., retrievers) as memory extensions, enabling LLMs to access factual information without storing it parametrically. More broadly, the recent survey (Qu et al., 2025) systematically categorizes tool-use into planning, selection, calling, and response stages, and highlights its promise for improving factuality, compositional reasoning, and generalization. A growing number of works explore how models acquire tool-use capabilities. Toolformer (Schick et al., 2023) introduces a self-supervised setup where LLMs learn when and how to call APIs (e.g., calculators, search) by generating and filtering synthetic training data. MetaTool (Wang et al., 2024c) introduces a benchmark to assess tool-use awareness and selection under realistic agent settings, showing that current LLMs still struggle with tool choice and reasoning under ambiguity. Houliston et al. (2024) offers a principled offline preference-based RL algorithm tailored to LLM alignment under ambiguous context by leveraging reward or preference uncertainty estimates - such a method could naturally extend to tool learning, where ambiguity in delegation decisions is a central challenge. Combining tool use with multi-step reasoning, ART (Paranjape et al., 2023) proposes a framework where LLMs write reasoning steps as executable programs that can call tools and integrate their

outputs seamlessly. This work illustrates how tool use can serve as a functional memory system to complement and augment the limited and potentially brittle parametric memory of LLMs.

B PROOFS

B.1 PROOF OF THEOREM 3.3

We provide a formal derivation of the average-case parameter lower bound stated in Theorem 3.3. In contrast to the worst-case analysis of Section 3, which requires a model class to represent *all* datasets in $\mathcal{D}$ with perfect accuracy, we consider a probabilistic setting in which factual queries are drawn from an underlying data-generating distribution and models are evaluated by their expected recall performance.

Setting. A factual query is generated as follows. First, a name–attribute pair (N, A) is drawn according to a distribution $\mathbb{P}_{N,A}$ over $\mathcal{N} \times \mathcal{A}$. Conditioned on $(N, A) = (n, a)$, the corresponding value is drawn as $V \sim \mathbb{P}_{V|N,A}(\cdot \mid n, a)$. This defines a joint distribution over (N, A, V).

A model $f \in \mathcal{M}$ produces a prediction via its recall rule $\hat{V} := R(f)(N, A)$, for which we define the expected 0–1 recall error as $\ell(f) := \mathbb{P}\Big[\hat{V} \neq V\Big]$, where the probability is taken over the data-generating distribution and model recall rule.

We now restate the average-case lower bound to be proved.

Theorem B.1 (Average-Case Parameter Lower Bound)**.** *Let $\mathcal{M}$ be a class of models with P parameters, each quantized to b bits. Suppose there exists a model $f \in \mathcal{M}$ achieving expected recall error $\ell(f)$ on $T = |\mathcal{N}| \cdot |\mathcal{A}|$ factual queries drawn i.i.d. from the data-generating distribution. Then*

$$P \;\geq\; \frac{T}{b}\Big[H(V \mid N, A) - H(\ell(f)) - \ell(f)\log_2(|\mathcal{V}| - 1)\Big],$$

where $H(\ell) = -\ell \log_2 \ell - (1-\ell)\log_2(1-\ell)$ denotes the binary entropy of a bernoulli with parameter ℓ.

Proof. We begin by characterizing the information required to answer a single factual query.

For a fixed query (n, a), the uncertainty over the correct value is quantified by the conditional entropy $H(V \mid N{=}n, A{=}a)$. By the source coding theorem, this quantity represents the minimum expected number of bits required to specify the correct value for that query under the distribution $\mathbb{P}_{V|N,A}(\cdot \mid n, a)$. Averaging over the query distribution yields the expected per-query information requirement

$$H(V \mid N, A) \;=\; \mathbb{E}_{(N,A)}\big[\, H(V \mid N{=}n, A{=}a)\,\big]\,.$$

We next relate this information requirement to the model's predictions. The prediction $\hat{V}$ can be viewed as a randomized decoding of the ground-truth value V based on the model's parameters.[1]

The mutual information

$$I(V; \hat{V} \mid N, A) \;=\; H(V \mid N, A) - H(V \mid \hat{V}, N, A),$$

quantifies the expected reduction in uncertainty about V induced by observing the model's output, beyond what is already known from the query (N, A). Equivalently, it measures the number of bits about the true value that are recoverable from the prediction under the data-generating distribution.

If $\ell(f) = 0$, then $\hat{V} = V$ almost surely and the conditional entropy vanishes, so $I(V; \hat{V} \mid N, A) = H(V \mid N, A)$. For a non-zero error, residual uncertainty remains, and the mutual information is strictly smaller.

[1] Fixing θ induces a joint distribution over $(V, \hat{V})$ via the data-generating process and recall rule $\hat{V} = R_\theta(N, A)$, defining an information-theoretic channel from V to $\hat{V}$ with noise due to model error.

To bound the residual uncertainty, we apply Fano's inequality.

Lemma 1 (Fano's inequality, conditional form)**.** *Let V be a discrete random variable taking values in a finite set $\mathcal{V}$, and let $\hat{V}$ be an estimate of V. Then*

$$H(V \mid \hat{V}) \;\leq\; H(\ell) + \ell \log_2(|\mathcal{V}| - 1),$$

where $\ell := \mathbb{P}[\hat{V} \neq V]$ and $H(\ell) = -\ell \log_2 \ell - (1-\ell)\log_2(1-\ell)$.

Applying Lemma 1 conditionally on (N, A) and averaging over the query distribution yields

$$H(V \mid \hat{V}, N, A) \;\leq\; H(\ell(f)) + \ell(f) \log_2(|\mathcal{V}| - 1).$$

Combining this inequality with the definition of mutual information gives

$$I(V; \hat{V} \mid N, A) \;\geq\; H(V \mid N, A) \;-\; H(\ell(f)) \;-\; \ell(f) \log_2(|\mathcal{V}| - 1). \tag{2}$$

We now aggregate this information requirement across multiple queries. Consider a collection of T independent factual queries drawn from the data-generating distribution. By the additivity of mutual information across i.i.d. samples, for a model to achieve expected recall error $\ell(f)$ on these queries, its predictions must collectively represent at least $T \cdot I(V; \hat{V} \mid N, A)$ bits of information about the corresponding ground-truth values. A model class with P parameters quantized to b bits can realize at most $|\mathcal{M}| \leq 2^{bP}$ distinct parameter configurations, and hence can encode at most bP bits of information. It follows that

$$bP \;\geq\; T \cdot I(V; \hat{V} \mid N, A).$$

Substituting the lower bound from Eq. (2) yields

$$P \;\geq\; \frac{T}{b}\Big[H(V \mid N, A) - H(\ell(f)) - \ell(f)\log_2(|\mathcal{V}| - 1)\Big],$$

which completes the proof. □

A note on the tightness of the bound. The bound in Theorem B.1 is information-theoretically tight up to constant factors. When $\ell(f) = 0$, the bound is exact and reduces to $P \geq \frac{T}{b} H(V \mid N, A)$, recovering the same linear dependence on the number of facts as in the worst-case analysis.[2] The application of Jensen's inequality in Fano's inequality introduces looseness in the bound, which increases monotonically with the recall error $\ell(f)$. The entropy level $H(V \mid N, A)$ determines how significant this looseness is relative to the total information required per fact: for high-entropy data, the bound remains tight up to constants even at moderate error rates, whereas for low-entropy data the same error can render the bound loose or vacuous (Cover & Thomas, 2005). This behavior is observed in Fig. 8.

Generally, this result shows that accounting for structure in the data distribution, as well as tolerance for non-zero error (on expected recall performance), reduces the information required to model individual facts; the total parameter requirement remains linear in T, despite the potential looseness of the bound.

[2]Further assuming uniform sampling of attributes implies $H(V \mid N, A) \leq \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \log |\mathcal{V}_a|$, giving the exact worst-case result.

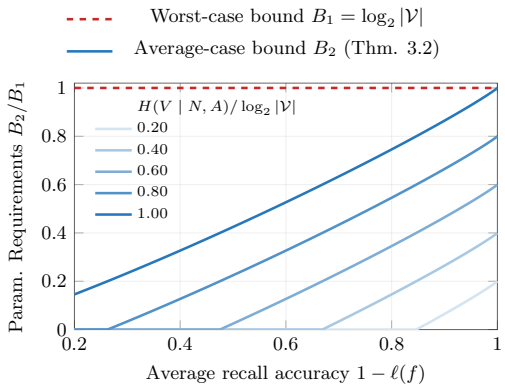

Figure 8: In-weight parameter requirements, normalized. We set $|\mathcal{V}| = 1000$. Each curve corresponds to a different normalized conditional entropy $\frac{H(V|N,A)}{\log_2 |\mathcal{V}|}$. The average-case bound increases monotonically with recall. Negative values are due to the looseness of the bound under either low accuracy or high data entropy.

B.2 Proof of Theorem 4.2

In this section, we prove the bound from Theorem 4.2 on the number of parameters needed for a transformer to learn when and how to query a retrieval system. Using the same example and notations as in Section 2, given a question of the form

$$Q = \underbrace{\text{Where was}}_{\varphi_1(a)} \underbrace{\text{Thierry de Sienne}}_{\varphi_2(n)=n} \underbrace{\text{born?}}_{\varphi_3(a)},$$

the model f must output a well-formatted string

$$T = \underbrace{\text{To answer this request, I will make a tool-call. <DB> FIND birthplace}}_{\chi_1(a)} \underbrace{\text{FOR Thierry de Sienne </DB>}}_{\chi_2(n)},$$

where <DB> and </DB> are special tokens which, when parsed, result in the query being provided to the retrieval system. Let us call this subtask the querying task. The retrieval system then returns a string

$$\xi(f, a, n) = \text{“Germany”},$$

containing the answer to Q. Then, given the context $Q, T, \xi(f, a, n)$, f must return a well-formatted answer

$$A = \underbrace{\text{Thierry de Sienne}}_{\psi_1(n)} \underbrace{\text{was born in}}_{\psi_2(a)} \underbrace{\text{Germany}}_{\psi_3(v)}.$$

Let us call this second subtask the formatting task.

We only prove that the querying task, i.e., outputting T as a function of Q, can be learnt by a transformer for which the bounds from Theorem 4.2 apply: similar arguments are enough to prove that the formatting task can be learnt as well (and simultaneously) by such a model.

Furthermore, we make the following simplifying assumptions:

1. The map $\varphi_1 : a \mapsto \varphi_1(a)$ is injective.
2. The only question mark in each question Q is at the very end of the string.
3. The string Q is provided to the model's tokenizer in such a way that each substring $\varphi_1(a), \varphi_2(n), \varphi_3(a)$ is separately tokenized, i.e., the tokenized version of Q is such that no token contains characters belonging to several substrings.
4. There does not exist $a, a' \in \mathcal{A}$ such that $a \neq a'$ but $\varphi_1(a)$ is a substring of $\varphi_1(a')$.
5. The number of tokens in the questions and answers does not exceed a given constant.

The first, second, and third assumptions are reasonable and could be relaxed at the cost of a more complex proof. Some variant of the fourth assumption cannot be avoided without additional constraints on the names $\mathcal{N}$, as two distinct pairs $(a, n), (a', n')$ could otherwise be mapped to the same question Q. The last condition forces the number of tokens in the names, attributes, and values to be less than some large number, e.g., 10k. This weak constraint is essentially a technicality linked to the limitations of positional embeddings, hence to the context length of any transformer, rather than a true algorithmic constraint.

Theorem B.2 (Parameter Upper Bound, querying task)**.** *Consider a finite set $\mathcal{A}$ of attributes and a set $\mathcal{N}$ of names. Then there exists a transformer with at most 8 transformer blocks, an embedding dimension of at most $O(|\mathcal{A}|)$, and a total number of parameters at most $O(|\mathcal{A}|^2)$, which can achieve perfect accuracy on the querying task, i.e., that outputs*

$$T = \chi_1(a) \circ \chi_2(n)$$

as a function of

$$Q = \varphi_1(a) \circ \varphi_2(n) \circ \varphi_3(a)$$

for any $a \in \mathcal{A}, n \in \mathcal{N}$.

Proof. We focus on the most informative elements of the proof and omit some tedious details.

Given a transformer with L layers and $k \in \{1, \dots, L\}$ and internal dimension D, we let $a_k \in \mathbb{R}^D$ denote the input of the feedforward network of the k-th transformer block (i.e. the renormalization of the sum of the output of the multi-head attention and the residual connection), and $y_k \in \mathbb{R}^D$ denote the output of the k-th transformer block (i.e. the renormalization of the sum of the output of the feedforward network and of the block's second residual connection, see Fig. 9 for an illustration of a Transformer block).

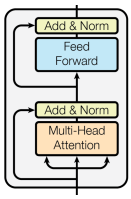

Figure 9: Architecture of a Transformer block, Vaswani et al. (2017).

In what follows, we mean by "the vector a_k above the i-th token of sequence S" the vector a_k found within the k-th layer of the transformer and to which a direct line of residual connections starting at the embedding of the i-th token leads (and same for y_k).

By "an encoding of X", for some quantity or object X that can take values in some set S, we mean the image of X by some injection $I : S \to \mathbb{R}^n$.

Committing a minor abuse of conventions, we say that a vector $v \in \mathbb{R}^m$ (e.g. a_k) "contains" another vector $u \in \mathbb{R}^n$ (e.g. an encoding of some object) if the coordinates u are a subset of the coordinates of v, i.e. $u = \pi(v)$ for some orthogonal projection along a sub-family of the orthogonal basis. By convention, whenever we state that a vector v contains several vectors $u_1, \dots, u_l$, the dimensions in which those vectors are contained do not intersect, i.e. $v = \{u_1\} \times \dots \times \{u_l\} \times \{w\}$ for some vector w and up to some permutation of its coordinates.

When we say that a vector a_k or y_k above a given token contains "a flag" corresponding to a certain condition on the token and its predecessors, we mean that one of the coordinates of the vectors (without overlap with the coordinates used to store other information) is equal to 1 if this condition is fulfilled (the flag is raised), and to 0 otherwise.

Remember that we must design a transformer that outputs the sequence

$$T = \underbrace{\text{To answer this request, I will make a tool-call. <DB> FIND birthplace}}_{\chi_1(a)} \underbrace{\text{FOR Thierry de Sienne </DB>}}_{\chi_2(n)},$$

when provided with

$$Q = \underbrace{\text{Where was}}_{\varphi_1(a)} \underbrace{\text{Thierry de Sienne}}_{\varphi_2(n)=n} \underbrace{\text{born?}}_{\varphi_3(a)}.$$

Concretely and due to the auto-regressive nature of transformers, the model will progressively concatenate tokens to Q until a joint string $Q \circ T$ has been produced.

We are going to show that there exists a transformer with 8 layers and embedding dimension $C|\mathcal{A}|$ for $C > 0$ large enough such that for any input question $Q = \varphi_1(a) \circ \varphi_2(n) \circ \varphi_3(a)$ (for some $a \in \mathcal{A}, n \in \mathcal{N}$), the following facts hold simultaneously[3].

► **Fact: any y_k above any token contains an encoding of the absolute position of the token within the entire sequence.** The positional encodings, which contain the desired information, are added to the tokens' learned embedding to form the input to the model's first transformer block. For a large enough embedding dimension and a judicious choice of learned embeddings, and as long as the number of tokens does not exceed a given limit, the absolute position can be recovered by the first transformer block (using the fact that two-layer neural networks are universal approximators and assuming again that the embedding dimension is large enough). This information can then be stored in some of the coordinates of y_1, and be passed to all consecutive y_k (e.g., using the residual connections and trivial attention heads). Note that the embedding dimension needed does not depend on $\mathcal{A}$ or $\mathcal{N}$.

► **Fact: y_3 contains one flag per attribute $a' \in \mathcal{A}$. The flags corresponding to $a' \neq a$ are never raised, and the flag corresponding to a is only raised above the last token of $\varphi_1(a)$.**

At any step, let t denote the last token of the currently processed sequence, and let p be its absolute position. As stated above, y_1 contains an encoding of p. It can also contain an encoding of t (using similar arguments).

The attention layer of the second transformer can then pass this information to its feedforward network. Using the approximation properties of neural networks again, for each $a' \in \mathcal{A}$ there can be a dedicated entry $v_{a'} \in \mathbb{R}$ in the network's output y_2 such that:

- if $p > \text{len}(\varphi_1(a))$, then $v_{a'} = -2$,
- if $p \leq \text{len}(\varphi_1(a))$ and t is not equal to the p-th token of $\varphi_1(a)$, then $v_{a'} = -2$.
- if $p \leq \text{len}(\varphi_1(a))$ and t is equal to the p-th token of $\varphi_1(a)$, then $v_{a'} = 1/\text{len}(\varphi_1(a))$ and

In the third attention layer, the coordinates $v_{a'}$ corresponding to each $a' \in \mathcal{A}$ can be summed. If $a \neq a'$, this sum is strictly negative. If $a = a'$, then:

- if $p < \text{len}(\varphi_1(a))$, then the sum is positive but strictly smaller than 1,
- if $p = \text{len}(\varphi_1(a))$, the sum is exactly 1, and
- if $p > \text{len}(\varphi_1(a))$, the sum is strictly negative.

Hence, all cases are differentiated. The following feedforward network can then output within y_3 a flag for all $a' \in \mathcal{A}$, which is raised only if $a = a'$ and the current token is the last token of $\varphi_1(a)$. As the lengths of the substrings $\{\varphi_1(a')\}$ are bounded, this can be done with an embedding dimension in $O(|\mathcal{A}|)$.

► **Fact: y_1 contains a flag that is only raised above the question mark at the end of $\varphi_3(a)$, and another flag that is raised above every token after the question mark.**

This is essentially trivial (using our second assumption).

► **Fact: y_4 contains a vector that encodes the length in tokens of n above every token after the question mark at the end of $\varphi_3(a)$.**

The facts above state that y_3 contains an encoding of the absolute position of the current token, a flag f_a corresponding to a that is only raised above the last token of $\varphi_1(a)$, a flag $f_?$ that is only raised above the question mark at the end of $\varphi_3(a)$, and a flag $f_{\geq ?}$ that is raised above every token after the question mark (these two flags are passed from y_1 to y_3).

At each token, two non-trivial queries are made in the fourth attention layer if and only if $f_{\geq ?}$ is raised. The first non-trivial query matches only with a key generated from f_a and its output value is an encoding of the position of the last token of $\varphi_1(a)$ as well as an encoding of the value of $a \in \mathcal{A}$.

[3] In the proof, approximating functions, for instance needed to compute the absolute position, amounts to a perfect approximation on a finite subset using the fact that the set of tokens is discrete

The second non-trivial query matches only with a key generated from $f_?$ and its output value is an encoding of the position of the question mark at the end of $\varphi_3(a)$. The fourth feedforward network then takes these two positional encodings and the identity of $a \in \mathcal{A}$, and outputs (in some subset of the coordinates of y_4) the distance between the last token of $\varphi_1(a)$ and the question mark, minus the length in tokens of $\varphi_3(a)$. This is precisely the length in tokens of n. This can be done with an embedding dimension in $O(|\mathcal{A}|)$.

► Fact: the model successfully outputs $\chi_1(a) \circ$ "FOR" after Q.

A non-trivial query is made in the second attention layer if and only if the flag $f_{\geq ?}$ is raised. This query matches the flag $f_?$ and brings to y_2 an encoding of the position of the question mark. y_2 also contains an encoding of the position of the current token, and a copy of the flag $f_{\geq ?}$. This information is passed to y_3. The fourth attention layer can match the flag above the last token of $\varphi_1(a)$ which describes the value of $a \in \mathcal{A}$ and pass this value to the fourth feedforward network, along with the encoding of the position of the question mark, the encoding of the position of the current token and the copy of the flag $f_{\geq ?}$. If $f_{\geq ?}$ is raised, then the fourth feedforward network can compute the distance l between the question mark and the current token, and output an encoding of the l-th token of $\chi_1(a) \circ$ "FOR" (or a flag saying that no token should be output if l is larger than the length of $\chi_1(a) \circ$ "FOR"). This can be done with an embedding dimension in $O(|\mathcal{A}|)$. This encoding can then be passed to the following layers, until the correct l-th token is output.

► Fact: y_5 contains a flag that is only raised above the last token of $\chi_1(a) \circ$ "FOR".

This follows from the same arguments as the previous fact.

► Fact: above every token after $\chi_1(a) \circ$ "FOR", y_6 contains an encoding of the distance between the current token and the last token of $\chi_1(a) \circ$ "FOR".

The sixth attention layer can match with the flag from the previous fact to pass its position to the sixth feedforward layer, as well as the position of the current token. These can be processed by the feedforward layer to output the desired distance.

► Fact: above every token after the last token of $\chi_1(a) \circ$ "FOR", y_5 contains an encoding of the distance d between the first token of $\varphi_2(n) = n$ and the last token of $\chi_1(a) \circ$ "FOR".

This distance is equal to the length of n plus the length of $\varphi_3(a) \circ \chi_1(a) \circ$ "FOR". The length of n is encoded in y_4 above every token after the question mark at the end of $\varphi_3(a)$ (using an earlier fact). This quantity can then be passed to the fifth feedforward network. y_3 contains a flag corresponding to a above the last token of $\varphi_1(a)$ (using an earlier fact again), which is raised nowhere else. This quantity can be passed to y_4, queried by a dedicated attention head in the fifth attention layer, and passed to the fifth feedforward network. As there are finitely many $a' \in \mathcal{A}$, the fifth feedforward network can memorize every length $\varphi_3(a') \circ \chi_1(a') \circ$ "FOR" with an embedding dimension in $O(|\mathcal{A}|)$. It adds the length of $\varphi_3(a') \circ \chi_1(a') \circ$ "FOR" to that of n, and encodes it into y_5.

► Fact: the model successfully outputs n after $\chi_1(a) \circ$ "FOR", and y_7 contains a flag that is only raised above the last token of (this copy of) n.

For every token after $\chi_1(a) \circ$ "FOR", the sixth layer can use the encoding of the distance d from the previous fact and the encoding of the position of the current token to output an encoding of the position p of the token that is d entries before the current token. The seventh attention layer can then use this encoding to match the token at position p (which is part of n) and get its value t. We know that y_6 also contains an encoding of the distance d' between the current token and the last token of $\chi_1(a) \circ$ "FOR". If d' is smaller than the length of n, the seventh feedforward network outputs a copy of t, which lets the model output t. If d' is exactly the length of n (i.e., the current token is the last token of the copy of n), the seventh feedforward network outputs a flag within y_7. If d' is greater than n, the flag is not raised. Hence the model does output n after $\chi_1(a) \circ$ "FOR".

► **Fact: the model successfully outputs `</DB>` after n, and thus successfully outputs T in response to Q.**

The eighth attention layer matches the flag in y_7 that is raised above the last token of n. It passes the existence of this flag, the position of the last token of n, and the position of the current token to the eighth feedforward network, which computes the distance d'' between the two positions and outputs an encoding of the d''-th token of `</DB>`, which the model can then output.

The combination of these facts is enough to complete the proof of Theorem B.2. □

C Implementation details

In this section, we provide details on our experimental setup, specifically regarding the database construction, language modeling pipeline, and training runs. We also provide additional results in both controlled and large-scale settings. Seeds, when provided, are relative to distinct data generation and network initialization, and as such, results are presented with the standard deviation across runs (e.g., we conducted 10 different runs to obtain Fig. 2).

Reproducibility. The code is available at `https://github.com/ambroiseodt/itl`, along with the reproducibility instructions. We provide in the main paper and in the appendix all the details needed to reproduce our results. Experiments were conducted using high-performance GPUs such as NVIDIA V100 and A100. Our implementation supports distributed training, but we have also ensured that it can be run on a single device, and even on a single CPU, as shown in Section 5.1.

C.1 Factual recall database

In our experiments, we construct synthetic biographical datasets using a fixed list of names $\mathcal{N}$ (each name is a pair {first name-last name}) and four attributes $\mathcal{A}$: "birth place", "birth date", "current address", "occupation"; which can take 7, 16800, 213, and 100 values respectively. This yields $4|\mathcal{N}|$ atomic facts per dataset instance. We present in Fig. 10 an example of such a dataset, where we query the birth date of Kenny McRoy, comparing the *in-weight* and *in-tool* settings. We adopt a dialog format, with a chat template. We design our tokenizer to take care of the special tokens needed (see Appendix C.2). We can see that the in-weight setting leads to a single-turn dialog for the LLM, while the in-tool one requires a multi-turn dialog.

Extension to other question-answering categories. The setting introduced in Section 2 is tailored to open-ended questions, which are suitable for the factual recall task. It can be extended to other QA categories, such as boolean or multiple choice questions, using the template functions φ, ψ. For instance, to adapt our setting to boolean question-answering datasets, we could define the query similarly $Q = \varphi_1(a) \circ \varphi_2(n) \circ \varphi_3(a)$ with $\varphi_1(a) = is$, $\varphi_2(n) = Paris$, $phi_3(a) =$ the capital of France and the answer $A = \psi_1(a, n)$ where ψ_1 a binary function outputting "Yes" or "No". An extension of Theorem 3.2 could be obtained with a similar proof and the counting argument from Theorem 3.1.

C.2 Controlled setting

In our experiments of Section 5.1, we train small Llama3-style language models (Grattafiori et al., 2024) from scratch. To that end, we build a language modeling pipeline following the literature standard (Videau et al., 2024). Below, we provide details on the tokenization, the architecture, and the training setup, alongside further results.

Tokenizer. Transformer models (Vaswani et al., 2017), like most neural networks, operate on numerical data, and in particular, sequences of vectors. To perform language modeling, a given sequence is mapped to a sequence of numerical vectors by first splitting it into words or subwords and then encoding each word or subword into a vector. This process is referred to as the tokenization process and is typically performed using tokenizers. Most state-of-the-art language models have an associated pre-trained tokenizer; it is also possible to encode text using their byte representation. Since we pretrain our model from scratch, we considered the latter byte tokenizer to avoid potential

```
# In-weight query
{
"dialog": [
      {"source": "user", "content": "When was Kenny McRoy born?"},
      {"source": "assistant", "content": "Kenny McRoy was born on 19/05/1998."},
   ],
   "people_id": 0,
   "answer": "19/05/1998",
}

# In-tool query
{
   "dialog": [
      {"source": "user", "content": "When was Kenny McRoy born?"},
      {
         "source": "assistant",
         "content": "To answer this question I will make a request to the database:\
             n```sql\nFIND birth_date FOR Kenny McRoy\n```",
      },
      {"source": "database", "content": "19/05/1998"},
      {"source": "assistant", "content": "Kenny McRoy was born on 19/05/1998."},
   ],
   "people_id": 0,
   "answer": "19/05/1998",
}
```

Figure 10: Examples of queries for the in-weight versus and in-tool settings.

bias from pretrained tokenizers. It simply splits words into single-character tokens and maps each of them to their raw byte representation (taking values in $[0, 256)$). It results in a vocabulary space of size 256 for the byte encoding ($256 = 2^8$), which can be augmented by adding special tokens. The vocabulary size reaches 260 with the addition of 4 special tokens: ***user***, ***assistant***, ***database*** and ***eod***. Each plays a specific role during the dialog between a user and a language model assistant:

- The ***user*** token marks the beginning of the dialogue by the user,
- The ***assistant*** token marks the language model assistant turn, which can either answer directly or make a call to the tool,
- The ***database*** token marks the beginning of the database result to the request,
- The ***eod*** token indicates the end of the dialog.

Chat templates. We model the factual recall task as a dialog between a user and a language model assistant. The user asks questions about a given person, for instance, their date of birth, birthplace, occupation, or current address. The assistant can then either answer directly using the information contained in its weights (in-weight learning) or make a call to a tool, implemented in our code as an SQL agent that has access to a database of biographies of persons. To decode such interactions, we implement a wrapper around the tokenizer to allow for dialog interactions, akin to the HuggingFace chat templates.

Online generation of batches. Data is organized into subsets that each represent a source of data. It follows the standard in the literature, which accounts for the fact that, in practice, pretraining data aggregates texts from different sources like Wikipedia or arXiv. To optimize data generation, batches are created on the fly by sampling with pre-defined proportions from the different subdirectories. Text is tokenized on the fly by iterating through each line of the sampled JSONL files.

Transformer architecture. Our models are small transformer decoders following the Llama3 implementation (Grattafiori et al., 2024) with Rotational Positional Embedding (Su et al., 2024),

Flash Attention (Dao et al., 2022), efficient attention for prefilling and token generation, gated linear units (Dauphin et al., 2017; Shazeer, 2020) with a SiLU activation (Hendrycks & Gimpel, 2023), RMSNorm layers (Zhang & Sennrich, 2019), and KV caching (Pope et al., 2022) for the inference. The feed-forward hidden dimension is fixed to 4 times the embedding dimension. For each dataset size, we train a family of small Llama3-style transformer models (Grattafiori et al., 2024) with 2 layers, 2 attention heads, a vocabulary size of 300, and a context window of 257. The embedding dimension ranges from 4 to 128 (to ensure that ROPE constraints between the number of heads and the embedding dimension are met), resulting in models with between 2K and 0.6M parameters.

Training. Our models are trained for up to $100{,}000$ steps using the AdamW optimizer (Loshchilov & Hutter, 2019), with a batch size of 128 samples, a decoupled weight decay coefficient of 0.1, and $(\beta_1, \beta_2) = (0.9, 0.95)$. We use a cosine learning rate scheduler with a warmup phase of 50 steps and a maximum learning rate of 0.001, and a final learning rate ratio of 0.

Evaluation and inference. At inference time, we need to generate tokens sequentially, which is known to be slow. To optimize it, we make use of the standard practices in the literature, like KV caching. To evaluate the factual recall of our model, we feed it prompts to be completed. We implement additional attention modes, such as prefilling and generation, to decode the dialog and allow interacting with the agent. These operations are parallelized by defining a specific prompt loader during evaluation. It efficiently deals with queues of prefilled tokens to generate prompt completions.

Tool implementation. In our work, the tool is an SQL agent relying on `sqlite3` to access a database containing biographies of people with their first name, last name, date of birth, place of birth, occupation, and current address. The language model assistant can learn to make queries to this database thanks to in-tool learning. The SQL agent can be used to create the database and insert new elements in it, and has agentic capabilities not only to execute a query and return a result but also to parse instructions from an LLM prompt, execute them, and answer in natural language to the language model assistant.

Experimental setup of Section 5.1. We construct synthetic biographical datasets from a fixed list of names $\mathcal{N}$ and four attributes $\mathcal{A}$: "birth place", "birth date", "current address", "occupation"; which can take 7, 16800, 213, and 100 values respectively. This yields $4|\mathcal{N}|$ atomic facts per dataset instance. For each dataset size, we train a family of small Llama3-style transformer models (Grattafiori et al., 2024) with 2 layers, 2 attention heads, a vocabulary size of 260 (byte encoding with 4 special tokens), and a context window of 257. The embedding dimension ranges from 4 to 128, resulting in models with between 2K and 0.6M parameters. Our models are trained for up to $100{,}000$ steps using the AdamW optimizer (Loshchilov & Hutter, 2019), with a batch size of 128 samples, a decoupled weight decay coefficient of 0.1, and $(\beta_1, \beta_2) = (0.9, 0.95)$. We use a cosine learning rate scheduler with a warmup phase of 50 steps and a maximum learning rate of 0.001, and a final learning rate ratio of 0.

C.3 Large-scale setting

In this section, we detail the experimental setup of Section 5.2. We fine-tune instruction-tuned language models on the synthetic factual datasets from Section 5.1, ranging from 500 to 50k atomic facts. Models include SmolLM 2 Instruct (135M, 360M, 1.7B) (Allal et al., 2025) and Llama 3.1/3.2 Instruct (1B, 3B, 8B) (Grattafiori et al., 2024), each trained to reach at least 95% recall. We use the same training setup as in Section 5.1, with learning rates adapted to model size. For SmolLM models, we use the values reported in the original paper: 10^{-3} for 135M and 360M, and $3 \cdot 10^{-4}$ for 1.7B. For Llama models, we scale learning rates η with model size P, setting $\eta = 2 \cdot 10^{-5}$ at 8B, and following the established inverse power law $\eta \propto P^{-1/2}$ for stable updates, with P the number of parameters (Gao et al., 2023; Kaplan et al., 2020).

Frequent checkpoints are saved to evaluate: (i) factual recall, (ii) HellaSwag accuracy, used as a proxy for general language abilities; this benchmark presents four candidate sentence completions, and the model is evaluated on its ability to select the most plausible one (Zellers et al., 2019), and (iii) Total Variation (TV) distance from the base model's output distribution, estimated over 100

natural language prompts by generating completions with the base model and computing the mean token-level ℓ_1 distance between base and fine-tuned output probabilities.

D ADDITIONAL RESULTS

D.1 CONTROLLED EXPERIMENTS

In this section, we provide additional results related to Section 5.1.

Impact of recall on parameter requirement. In the experiments displayed in Section 5.1, the recall accuracy is fixed at 95%. To better understand how it can impact the parameter requirement, we display in Fig. 11 the minimum parameter requirement as a function of the factual recall for 4 different numbers of facts. We notice that the higher the recall, the bigger the models, and that this increase is steeper when the number of facts to retrieve increases. This can be explained from Theorem 3.2, which makes the dependency between the model size and the number of facts explicit. In addition, Theorem 3.3 showcases the impact of the recall level on the parameter requirement. Given that tool use enables the retrieval of an unbounded number of facts without the need to increase the number of parameters, as shown in Fig. 2, this implies that in-tool learning is all the more beneficial when one wants a more effective model on a bigger dataset. This follows the increasing trends of toll use in the literature for harder benchmarks, such as mathematics.

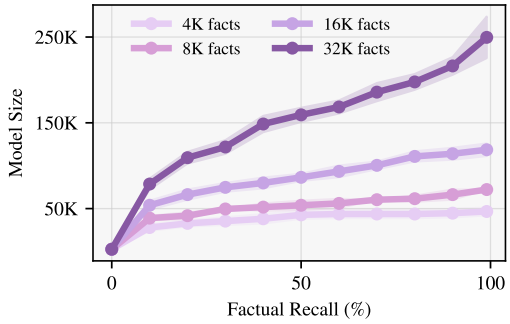

Figure 11: **Scaling of parameter requirements with the level of factual recall.** Minimum number of parameters required as a function of the factual recall. The higher the number of facts retrieved, the bigger the models. When the total number of facts to retrieve increases, we observe that the parameter requirement increases too. This shows that in-tool learning is all the more beneficial when very effective models are needed on large databases.

D.2 LARGE-SCALE EXPERIMENTS

In this section, we provide additional results related to Section 5.2.

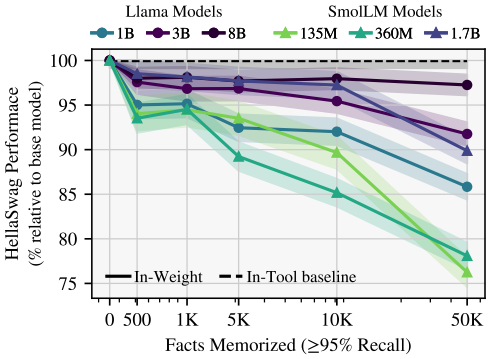

Figure 12: **HellaSwag performance (relative to base model) versus memorization load.** Same setup as Fig. 5; the dashed line represents the worst (lowest) performance among tool-models (Smol-135M).

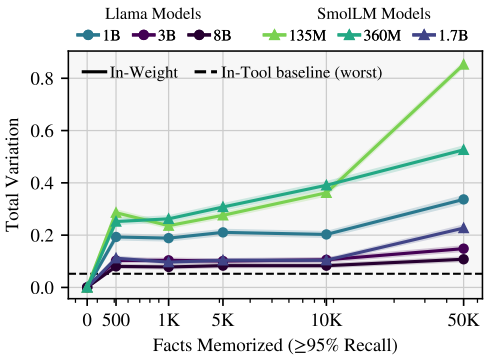

Figure 13: **Total Variation versus memorization load.** Same setup as Fig. 6; the dashed line represents the worst (highest) TV distance attained among tool-models (Smol-135M).

HellaSwag performance relative to base model. Fig. 5 in the main section of the paper displays the absolute HellaSwag accuracy attained by different models finetuned on datasets of increasing size. To complement this, we examine the relative drop in HellaSwag performance from the base model as the memorization load increases, in Fig. 12. This perspective offers a clearer view of the dynamics by which larger models maintain their prior capabilities under heavier memorization loads. Tool-learning (dashed line representing the worst performance over all tool-models) preserves general capabilities almost perfectly across model and dataset sizes. In contrast, in-weight learning leads to noticeable degradation, especially for smaller models and larger factual loads. This supports our theoretical prediction that parameter-based memorization causes interference and loss of prior capabilities, due to finite capacity and update interference. Larger models are more robust to such forgetting, but still exhibit performance decay beyond 10K facts.

A simplified intuition helps interpret the consistent ordering across model sizes: larger models possess more "capacity volume," so absorbing a fixed amount of new information replaces a smaller fraction of prior knowledge. While this view captures the trend, it likely omits key confounding factors: larger models are typically trained with more compute and data, and more complex architectures may also be more brittle to local updates—suggesting the optimization dynamics themselves differ in important ways.

Total variation versus memorization load. The intuition laid out above is also observed in the TV distance suffered by models as the memorization load grows: larger models diverge less than smaller counterparts. Fig. 13 depicts the Total Variation of finetuned models from their base-model, for different memorization loads. The figure shows the same results as Fig. 6 for in-weight models but replaces in-tool data with the highest (worst) TV achieved (by SmolLM 135M), represented as the dashed baseline. We notice in-tool TV baseline is constant across dataset sizes, and is lower than in-weight learning deviations - this is due to tool-use mastery unlocking unbounded factual recall. Overall, bigger models deviate less from their base model for the same in-weight memorization load. The distance is dramatically greater for smaller models on bigger datasets, close to 50k facts.

Training dynamics. For databases of 500K and 50K facts, we display the recall accuracy, the HellaSwag performance, and the total variation attained by checkpoints throughout training until complete memorization in Figures 14 and 15. Across model scales, we observe that recall accuracy improves rapidly during early training, with most models nearing saturation within 30–40 steps. However, a substantial number of additional steps are spent refining the final few percent of recall, suggesting that later training primarily sharpens the output distribution rather than acquiring new associations. Notably, most of the degradation in HellaSwag performance and increase in total variation occur early in training, indicating that the bulk of distributional shift and capability loss is

incurred in the initial stage of training for memorization, while later steps have comparatively minor effects.

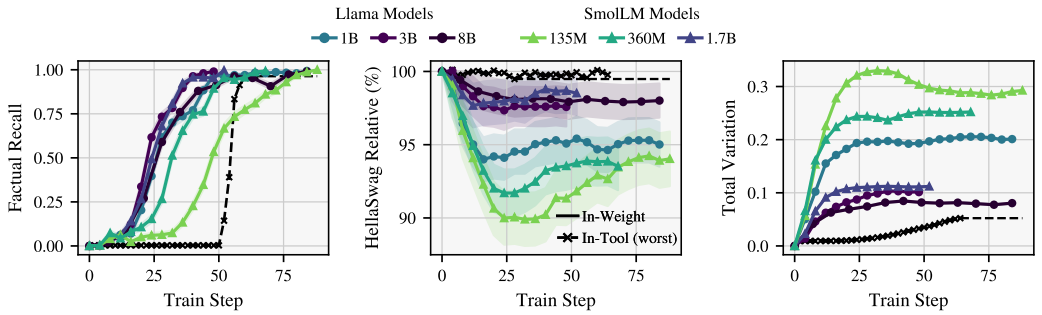

Figure 14: Metrics throughout training on a database of 500 facts. The dashed line represents the worst in-tool learning baseline, highlighting that tool use attains full recall while maintaining a very high ($\geq 98\%$) level of prior capabilities on HellaSwag, and by deviating less than 0.04 in TV from its base model.

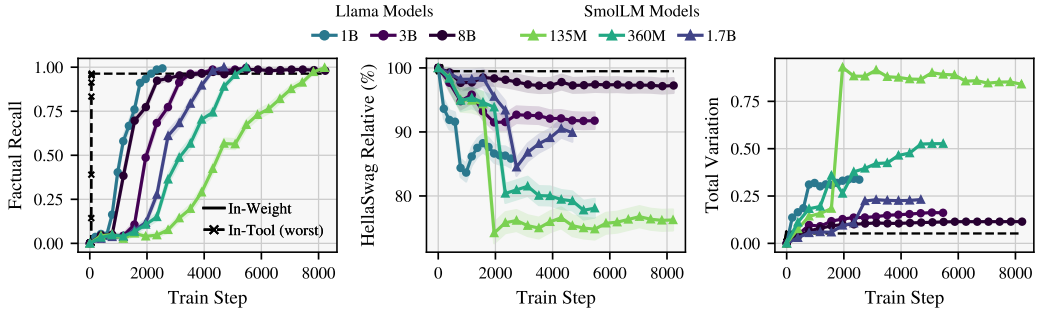

Figure 15: Metrics throughout training on a dataset of 50K facts. The worst in-tool model (dashed) mastered the tool in very few training steps, conserving higher HellaSwag capabilities and deviating less from its base model.