# Provable Benefits of In-Tool Learning for Large Language Models

Ambroise Odonnat
Noah's Ark Lab, Inria

Imperial, CSML Reading Group

March 12, 2026

# Co-authors

Sam Houliston
ETH Zürich

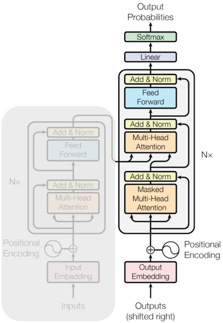
Charles Arnal
FAIR at Meta

Vivien Cabannes
FAIR at Meta

# Outline

# Outline

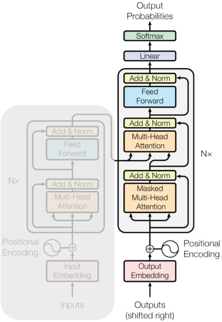
**Best models so far** → autoregressive decoder-only transformers.




(Vaswani et al., 2017)

# Large language models

**Best models so far** $\rightarrow$ autoregressive decoder-only transformers.




(Vaswani et al., 2017)

Vocabulary size $T$.
Context window $K$.
Parameter set $\boldsymbol{\Theta}$.

**GPT-3 :** $T = 50257$, $K = 2048$ and $|\boldsymbol{\Theta}| \sim 175B$

# Large language models

**Goal:** Predict the next token based on previous ones.

# Large language models

**Goal:** Predict the next token based on previous ones.
**Autoregressive:** Each token depends only on the past ones.

$\underbrace{\textbf{I} \quad \textbf{am} \quad \textbf{the}}_{\text{Previous tokens (context)}} \qquad \underbrace{\textbf{danger}}_{\text{Token being predicted}}$

# Large language models

**Goal:** Predict the next token based on previous ones.
**Autoregressive:** Each token depends only on the past ones.

$$\underbrace{\textbf{I} \quad \textbf{am} \quad \textbf{the}}_{\text{Previous tokens (context)}} \qquad \underbrace{\textbf{danger}}_{\text{Token being predicted}}$$

**Modelization:** Probability of a sequence $(x_1, x_2, \ldots, x_N)$:

$$\begin{aligned} \mathbb{P}(x_1, x_2, \ldots, x_N) &= \mathbb{P}(x_1)\mathbb{P}(x_2 \mid x_1) \cdots \mathbb{P}(x_N \mid x_1, x_2, \ldots, x_{N-1}) \\ &= \prod_{n=1}^{N} \mathbb{P}(x_n \mid x_1, x_2, \ldots, x_{n-1}) \end{aligned}$$

## Large-scale pretraining unlocks emerging capabilities!

**Language Models are Few-Shot Learners**

**Tom B. Brown*** **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

**Jared Kaplan†** **Prafulla Dhariwal** **Arvind Neelakantan** **Pranav Shyam** **Girish Sastry**

**Amanda Askell** **Sandhini Agarwal** **Ariel Herbert-Voss** **Gretchen Krueger** **Tom Henighan**

**Rewon Child** **Aditya Ramesh** **Daniel M. Ziegler** **Jeffrey Wu** **Clemens Winter**

**Christopher Hesse** **Mark Chen** **Eric Sigler** **Mateusz Litwin** **Scott Gray**

**Benjamin Chess** **Jack Clark** **Christopher Berner**

**Sam McCandlish** **Alec Radford** **Ilya Sutskever** **Dario Amodei**

OpenAI

GPT3 and in-context learning (Brown et al., 2020)

# Emerging capabilities of (static) LLMs

**Large-scale pretraining unlocks emerging capabilities!**

**Model Input**

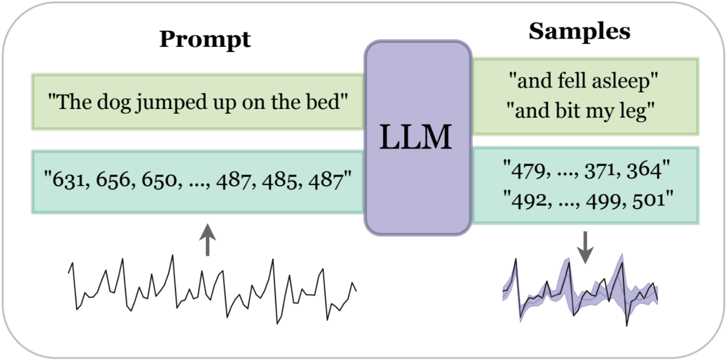
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

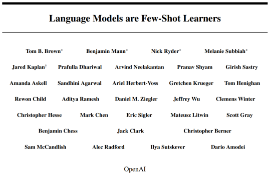
Chain-of-thought (Wei et al., 2022)

**Large-scale pretraining unlocks emerging capabilities!**




Generalization to other modalities (Gruver et al., 2023)

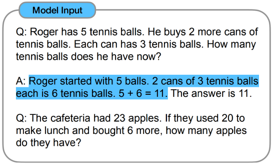
## Large-scale pretraining unlocks emerging capabilities!

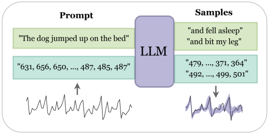


Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

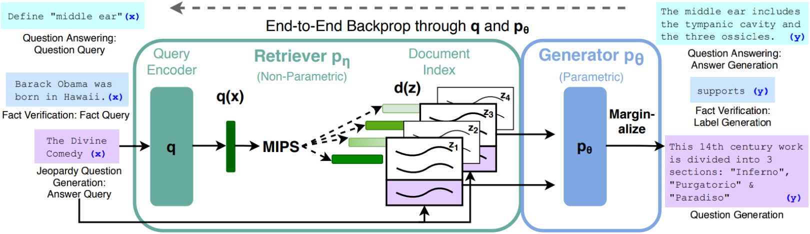
OpenAI



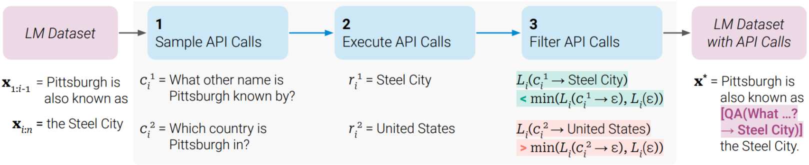

× Those methods extract knowledge from static predictors.

**Tool use to externalize memory and adapt to the context.**



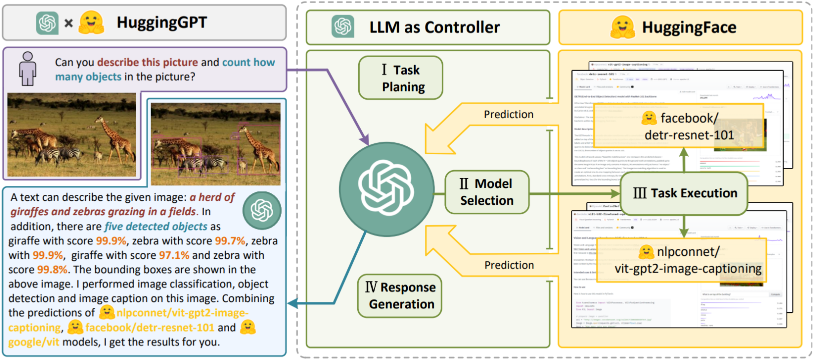

RAG (Lewis et al., 2020)

**Tool use to externalize memory and adapt to the context.**




Toolformer (Schick et al., 2023)

**Tool use to externalize memory and adapt to the context.**




HuggingGPT (Shen et al., 2023)

# Towards tool-augmented LLMs workflows

**Tool use to externalize memory and adapt to the context.**

- ✓ LLMs evolve towards dynamic context-aware systems,
- ✓ It allows them to reason, adapt, and act over time,
- ✓ Emergence of agentic workflows (e.g., Claude, OpenAI)

# Motivation

What is the most efficient way to acquire and use knowledge?

1. In-weight learning: memorize facts in the parameters,
2. In-tool learning: learn to access external sources of truth.

# Motivation

What is the most efficient way to acquire and use knowledge?

1. In-weight learning: memorize facts in the parameters,
2. In-tool learning: learn to access external sources of truth.

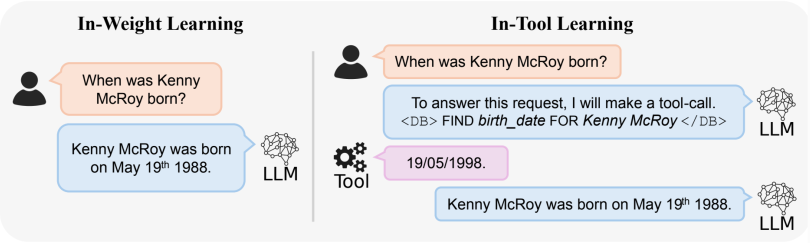



Figure: In-weight (memorization) vs. in-tool learning (external retrieval).

# Outline

# Knowledge representation

**Factual dataset**

Let $\mathcal{D}$ be a family of factual datasets $D$ defined as finite collections of facts $(n, a, v)$ with

- ⋆ a name $n \in \mathcal{N}$ (e.g., "Kim Perial"),
- ⋆ an attribute $a \in \mathcal{A}$ (e.g., "birthplace"),
- ⋆ a value $v \in \mathcal{V}_a$ (e.g., "United Kingdom").

# Knowledge representation

**Factual dataset**

Let $\mathcal{D}$ be a family of factual datasets $D$ defined as finite collections of facts $(n, a, v)$ with

- ⋆ a name $n \in \mathcal{N}$ (e.g., "Kim Perial"),
- ⋆ an attribute $a \in \mathcal{A}$ (e.g., "birthplace"),
- ⋆ a value $v \in \mathcal{V}_a$ (e.g., "United Kingdom").

- ✓ Akin to Physics of Language models (Allen-Zhu et al., 2024),
- ✓ Allows to quantify the amount of facts ($\#\text{Facts} = |\mathcal{N} \times \mathcal{A}|$).

# Factual recall

- ✓ Let $\mathcal{M}$ be the set of all transformers with a given architecture,
- ✓ Each model $f \in \mathcal{M}$ amounts to a specific choice of weights,
- ✓ Each model $f \in \mathcal{M}$ is associated with a recall rule $R(f)$.

# Factual recall

- ✓ Let $\mathcal{M}$ be the set of all transformers with a given architecture,
- ✓ Each model $f \in \mathcal{M}$ amounts to a specific choice of weights,
- ✓ Each model $f \in \mathcal{M}$ is associated with a recall rule $R(f)$.

**Recall rule**

$R(f)$ is defined by prompting a model to query the value of a pair $(n, a)$. The recall accuracy of $f$ on a dataset $D$ is the percentage of retrieval by $R(f)$ of the correct value $v$ over facts $(n, a, v) \in D$.

# Learnability

- ✓ Let $\mathcal{M}$ be the set of all transformers with a given architecture,
- ✓ Each model $f \in \mathcal{M}$ amounts to a specific choice of weights,
- ✓ Each model $f \in \mathcal{M}$ is associated with a recall rule $R(f)$.

**Learnability**

We say that the model class $\mathcal{M}$ *can solve* the recall task on the family of datasets $\mathcal{D}$ if, for each dataset $D \in \mathcal{D}$, there exists a model $f \in \mathcal{M}$ that achieves a perfect recall accuracy on $D$.

# In-weight learning

**In-weight: Train the model to generate the answer directly.**

# In-weight learning

**In-weight: Train the model to generate the answer directly.**

- ✓ Templates $\phi_i$ for queries, $\psi_i$ for answers,
- ✓ Query $Q = \phi_1(a) \circ \phi_2(n) \circ \phi_3(a)$,
- ✓ Answer $A = \psi_1(n) \circ \psi_2(a) \circ \psi_3(v)$.

# In-weight learning

**In-weight: Train the model to generate the answer directly.**

- ✓ Templates $\phi_i$ for queries, $\psi_i$ for answers,
- ✓ Query $Q = \phi_1(a) \circ \phi_2(n) \circ \phi_3(a)$,
- ✓ Answer $A = \psi_1(n) \circ \psi_2(a) \circ \psi_3(v)$.

$$Q = \underbrace{\text{Where was}}_{\phi_1(a)} \ \underbrace{\text{Kim Perial}}_{\phi_2(n)} \ \underbrace{\text{born?}}_{\phi_3(a)}$$

$$A = \underbrace{\text{Kim Perial}}_{\psi_1(n)} \ \underbrace{\text{was born in}}_{\psi_2(a)} \ \underbrace{\text{United Kingdom}}_{\psi_3(v)}$$

# In-tool learning

**In-tool: Train the model to use a tool to retrieve the answer.**

# In-tool learning

**In-tool: Train the model to use a tool to retrieve the answer.**

- ✓ Templates $\phi_i$ for queries, $\psi_i$ for answers, $\chi_i$ **for tool calls**,
- ✓ Query $Q = \phi_1(a) \circ \phi_2(n) \circ \phi_3(a)$,
- ✓ **Tool** $T = \chi_1(a) \circ \chi_2(n)$,
- ✓ Answer $A = \psi_1(n) \circ \psi_2(a) \circ \psi_3(v)$.

# In-tool learning

**In-tool: Train the model to use a tool to retrieve the answer.**

- ✓ Templates $\phi_i$ for queries, $\psi_i$ for answers, $\chi_i$ **for tool calls**,
- ✓ Query $Q = \phi_1(a) \circ \phi_2(n) \circ \phi_3(a)$,
- ✓ **Tool** $T = \chi_1(a) \circ \chi_2(n)$,
- ✓ Answer $A = \psi_1(n) \circ \psi_2(a) \circ \psi_3(v)$.

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

$$\underbrace{\text{FOR Kim Perial </DB>}}_{\chi_2(n)},$$

# In-weight learning vs. in-tool learning

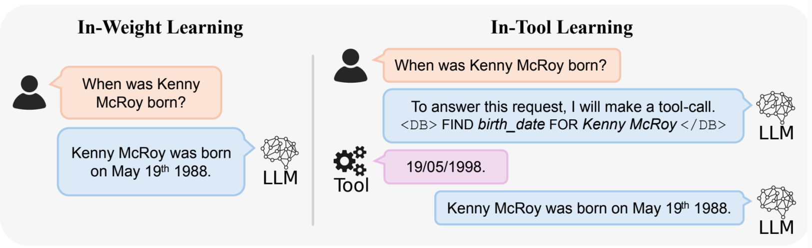



Figure: In-weight (memorization) vs. in-tool learning (external retrieval).

# Beyond open-ended questions

**Extension to, e.g., boolean: "Is Paris the capital of France?"**

- ✓ $\phi_1(a) =$ is, $\phi_2(n) = \text{Paris}$, $\phi_3(a) =$ the capital of France,
- ✓ Define query as before $Q = \phi_1(a) \circ \phi_2(n) \circ \phi_3(a)$;
- ✓ $\psi_1$ a binary function outputting "Yes" or "No",
- ✓ Define answer as before $A = \psi_1(a, n) \in \{\text{Yes}, \text{No}\}$.

# Outline

# Limitations of in-weight learning

**In-weight memorization is limited by the size of models.**

# Limitations of in-weight learning

**In-weight memorization is limited by the size of models.**

**Theorem (In-weight lower bound - informal)**

*Let $\mathcal{M}$ be a set of transformers with $P$ parameters and $\mathcal{D}$ be a family of factual datasets. If $\mathcal{M}$ can solve the recall task only with in-weight learning, then the number of parameters $P$ must satisfy:*

$$P \geq \mathcal{O}(\#\textit{Facts}).$$

# Limitations of in-weight learning

**In-weight memorization is limited by the size of models.**

**Theorem (In-weight lower bound - informal)**

*Let $\mathcal{M}$ be a set of transformers with $P$ parameters and $\mathcal{D}$ be a family of factual datasets. If $\mathcal{M}$ can solve the recall task only with in-weight learning, then the number of parameters $P$ must satisfy:*

$$P \geq \mathcal{O}(\#\textit{Facts}).$$

- × When the number of facts grows, memorization is impossible,
- × Architectural changes or external memory are needed.

# Benefits of in-tool learning

**In-tool learning solves the limitations of in-weight learning.**

# Benefits of in-tool learning

**In-tool learning solves the limitations of in-weight learning.**

**Theorem (In-tool upper bound - informal)**

*Let $\mathcal{D}$ be a family of factual datasets. Then, there exists a transformer with at most $\bar{P}$ parameters that can solve the recall task if augmented with the proper retrieval tool.*

# Benefits of in-tool learning

**In-tool learning solves the limitations of in-weight learning.**

**Theorem (In-tool upper bound - informal)**

*Let $\mathcal{D}$ be a family of factual datasets. Then, there exists a transformer with at most $\bar{P}$ parameters that can solve the recall task if augmented with the proper retrieval tool.*

- ✓ Model's size $\bar{P}$ independent of the number of people $|\mathcal{N}|$,
- ✓ No additional parameters needed when number of facts grows.

# Outline

# Controlled experiments

**Pretraining small language models on factual datasets.**

# Controlled experiments

**Pretraining small language models on factual datasets.**

✓ Data organized à la FineWeb,

# Controlled experiments

**Pretraining small language models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Byte tokenizer with chat template for tool use,

# Controlled experiments

**Pretraining small language models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Byte tokenizer with chat template for tool use,
- ✓ Custom pretraining pipeline akin to Meta Lingua,

# Controlled experiments

**Pretraining small language models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Byte tokenizer with chat template for tool use,
- ✓ Custom pretraining pipeline akin to Meta Lingua,
- ✓ Small Llama3-like language models,

# Controlled experiments

**Pretraining small language models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Byte tokenizer with chat template for tool use,
- ✓ Custom pretraining pipeline akin to Meta Lingua,
- ✓ Small Llama3-like language models,
- ✓ Tool is an SQL agent that queries an external database.

# Scaling of parameter requirements

**Empirical validation of the lower and upper bounds.**

**Empirical validation of the lower and upper bounds.**

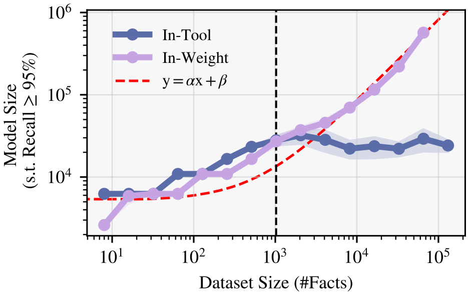



Figure: Minimal model size $P$ to achieve at least a $95\%$ recall.

**Empirical validation of the lower and upper bounds.**

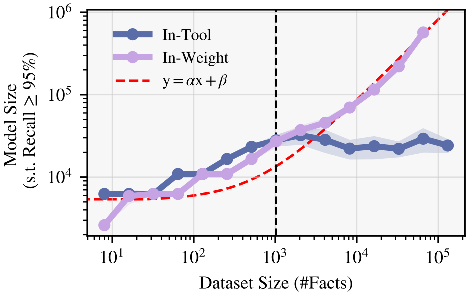



Figure: Minimal model size $P$ to achieve at least a $95\%$ recall.

× In-weight scales linearly with the number of facts,

✓ In-tool remains constant after a critical dataset size.

# Transition from memorization to retrieval

**In-tool models transition from memorization to rule learning.**

# Transition from memorization to retrieval

**In-tool models transition from memorization to rule learning.**

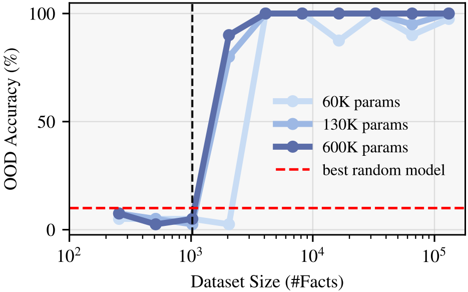



Figure: In-tool recall accuracy on out-of-distribution databases.

# Transition from memorization to retrieval

**In-tool models transition from memorization to rule learning.**

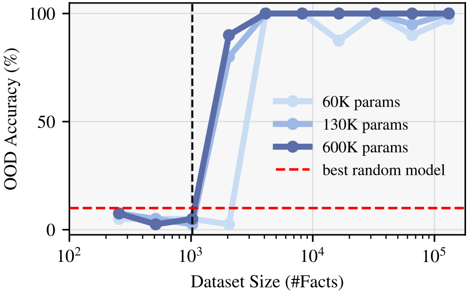



Figure: In-tool recall accuracy on out-of-distribution databases.

× Models start to memorize facts, similar to in-weight learning,

✓ After the transition, models learn to formulate in-tool queries.

# What counts as a fact?

**In-weight memorization is easier with interdependent facts.**

**In-weight memorization is easier with interdependent facts.**

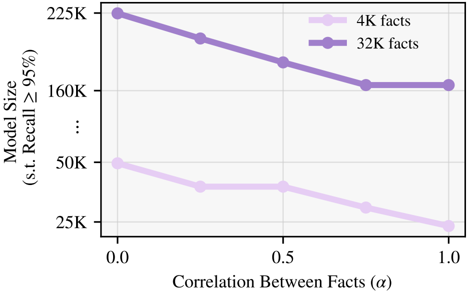



Figure: Minimal model size $P$ to achieve at least a $95\%$ recall.

# What counts as a fact?

**In-weight memorization is easier with interdependent facts.**

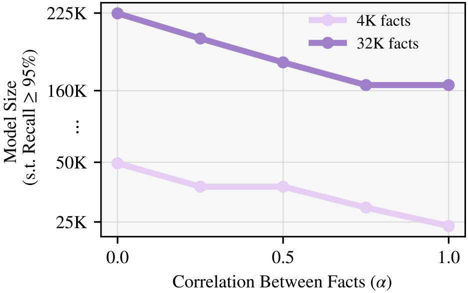



Figure: Minimal model size $P$ to achieve at least a $95\%$ recall.

- ✓ Correlation breaks independence between triplets $(n, a, v)$,
- ✓ Reduced number of "effective" facts, easier to store.

# Large-scale experiments

**Finetuning large instruct models on factual datasets.**

# Large-scale experiments

**Finetuning large instruct models on factual datasets.**

✓ Data organized à la FineWeb,

# Large-scale experiments

**Finetuning large instruct models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Default tokenizers with special tokens and chat templates,

# Large-scale experiments

**Finetuning large instruct models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Default tokenizers with special tokens and chat templates,
- ✓ SmolLM (135M, 360M, 1.7B) and Llama3 (1B, 3B, 8B),

# Large-scale experiments

**Finetuning large instruct models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Default tokenizers with special tokens and chat templates,
- ✓ SmolLM ($135$M, $360$M, $1.7$B) and Llama3 ($1$B, $3$B, $8$B),
- ✓ Tool is an SQL agent that queries an external database,

# Large-scale experiments

**Finetuning large instruct models on factual datasets.**

- ✓ Data organized à la FineWeb,
- ✓ Default tokenizers with special tokens and chat templates,
- ✓ SmolLM ($135$M, $360$M, $1.7$B) and Llama3 ($1$B, $3$B, $8$B),
- ✓ Tool is an SQL agent that queries an external database,
- ✓ Evaluation with LM Evaluation Harness.

# Learning while preserving prior capabilities

**In-tool preserves capabilities while learning new facts.**

# Learning while preserving prior capabilities

**In-tool preserves capabilities while learning new facts.**

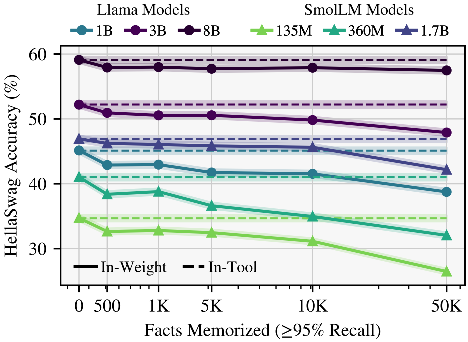



Figure: Hellaswag accuracy of models finetuned on factual datasets.

**In-tool preserves capabilities while learning new facts.**

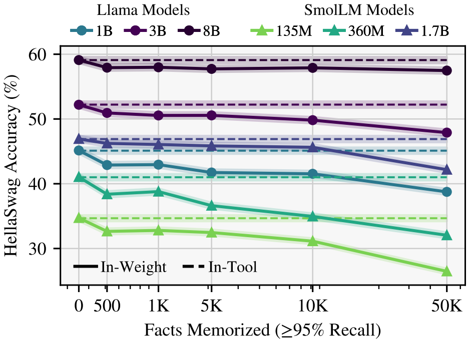



Figure: Hellaswag accuracy of models finetuned on factual datasets.

× In-weight learning impacts prior knowledge due to overloading,

✓ In-tool learning enables scalability without forgetting.

# Token distribution changes

**In-tool learning helps preserve models' behavior.**

# Token distribution changes

**In-tool learning helps preserve models' behavior.**

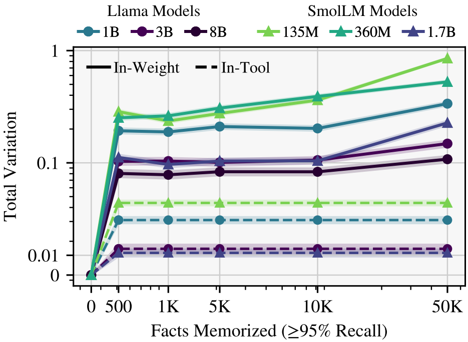



Figure: TV distance between base and finetuned models' distributions.

# Token distribution changes

**In-tool learning helps preserve models' behavior.**

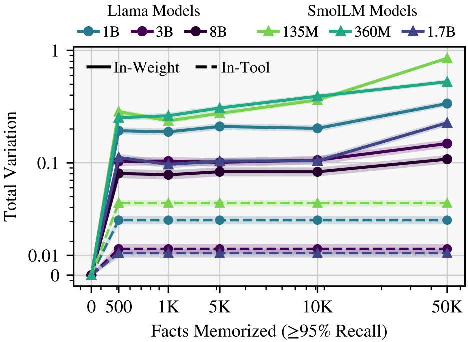



Figure: TV distance between base and finetuned models' distributions.

× In-weight learning alters models' token distribution,

✓ In-tool models remain close to the base models.

# Efficiency-tradeoffs

**In-tool learning requires less training compute than in-weight.**

**In-tool learning requires less training compute than in-weight.**

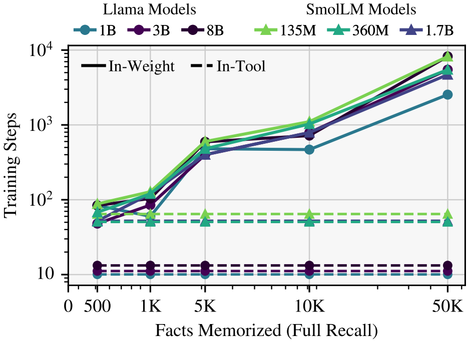



Figure: Training steps required to achieve a $100\%$ recall.

# Efficiency-tradeoffs

**In-tool learning requires less training compute than in-weight.**

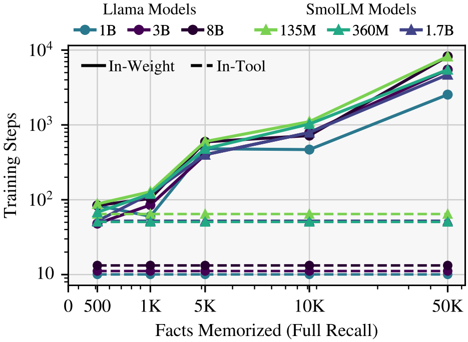



Figure: Training steps required to achieve a $100\%$ recall.

- × Memorizing individual facts requires many training steps,
- ✓ Learning to use a tool requires very few training steps.

# Outline

# Take Home Message

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

---

Montaigne, Les Essais

# Take Home Message

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

---

Montaigne, Les Essais

- ✓ In-weight learning is fundamentally limited by models' size,
- ✓ In-tool learning is more efficient and preserves capabilities.

# Take Home Message

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

Montaigne, Les Essais

- ✓ In-weight learning is fundamentally limited by models' size,
- ✓ In-tool learning is more efficient and preserves capabilities.

**Better to learn to use tools than to memorize facts.**

# Going further

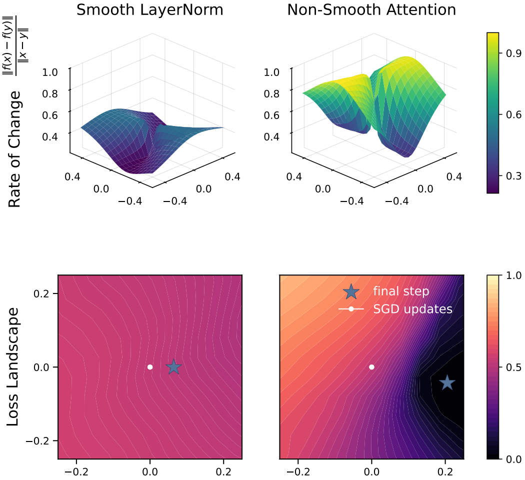
- ✓ Additional QA types (e.g., boolean, multiple choices),
- ✓ Additional tools use (e.g., Python interpreter, internet access),
- ✓ Extension to complex LLM agents.

# To Know More

This paper has been accepted at ICLR 2026 Workshop on MemAgents. Check it out along with the code to know more!

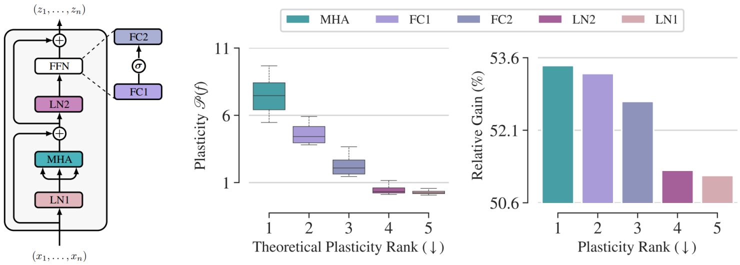
paper    code    website

## Vision Transformer Benefits from Non-Smooth Components




paper - code

**Finetuning high-plasticity components (non-smooth) yields larger performance gains than the stiff ones.**




paper - code

Thanks for your attention!