Toolformers on Minimal Data
Language Models Can Teach Themselves to Use Tools

Zachary Chosed (zac27), Daniel Chuang (dc863), Eric Marchetti (emm347),
Kevin Hu (kh785), Tony Matchev (akm99)

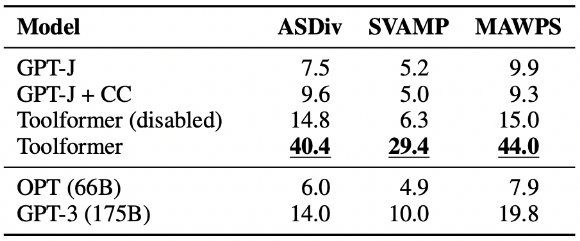
github.com/daniel-chuang/mathtoolformer-llm

1 Introduction

Large language models (LLMs) excel at many NLP tasks but struggle with precise arithmetic and dynamic information retrieval without human supervision. Schick et al.'s *Toolformer: Language Models Can Teach Themselves to Use Tools* [1] demonstrated that LLMs can teach themselves to call external APIs (calculator, Q&A, search engines) to augment their reasoning capabilities. However, their experiments built off GPT-J with 6.6 billion parameters. In this project, we investigate whether tool-augmented self-supervision can benefit a smaller, publicly available LLMs: the 135M-parameter SmolLM2 and the 1.5B-parameter Qwen2.5-Math [4]. By fine-tuning on a minimal set of tool-usage examples, we aim to reproduce the arithmetic performance gains reported by Toolformer and assess the viability of API-based learning at small scale.

2 Chosen Result

We target the primary result from Schick et al. [1], namely that fine-tuning an LLM with tool-calls yields accuracy improvements competitive with larger models. Specifically, we aim to replicate the arithmetic improvements highlighted in Figure 1. We also monitor the consistency of tool-call formatting alongside final accuracy, given one of the original criteria of *Toolformer* is to have the LLM decide when to use the tool. This result is critical because it tests whether the Toolformer approach generalizes to constrained models and limited compute.



Model	ASDiv	SVAMP	MAWPS
GPT-J	7.5	5.2	9.9
GPT-J + CC	9.6	5.0	9.3
Toolformer (disabled)	14.8	6.3	15.0
Toolformer	**40.4**	**29.4**	**44.0**
OPT (66B)	6.0	4.9	7.9
GPT-3 (175B)	14.0	10.0	19.8

Figure 1: Toolformer on mathematical reasoning benchmarks, outperforming OPT and GPT-3 with much smaller GPT-J (6.6B) model, [1]

3 Methodology

We began by assessing the zero-shot arithmetic capabilities of SmolLM2 before advancing to Qwen2.5-Math-1.5B on the EleutherAI Arithmetic QA prompts, a small dataset of 18000 elementary math set covering 1-4 digit addition, 1-4 digit subtraction, 1-2 digit multiplication, and simple chain of operations. Table 2 presents four representative examples.

Table 1: Arithmetic QA: Original Dataset

Config	Question	Expected	Model	Correct
3-digit addition	What is 355 plus 967?	1322	1322	✓
2-digit subtraction	What is 2 minus 40?	−38	38	✗
1-digit chain	What is (3 * 5) - 8?	7	8	✗

We used the original arithmetic prompts to train the *finetuned* condition, where prompts map directly to their numeric answers. To train the *tool-augmented* condition, the dataset was transformed so each output label was wrapped as a calculator API call (e.g. `<tool:calculator>5+3+2*5</tool>`). The table below demonstrates this format.

Table 2: Arithmetic QA: Toolformer Dataset

Question	Expected Tool Call
Question: What is 31 plus 72? Answer:	`<tool:calculator>31+72</tool>`
Question: What is (4 - 2) + 7? Answer:	`<tool:calculator>(4-2)+7</tool>`

We fine-tune both model variants using HuggingFace's `SFTTrainer` API using all default parameters. In the previous iteration (training for SmolLM2-135M) we trained for three epochs using the Hugging Face `Trainer` with AdamW (learning rate 4×10^{-4}, weight decay 0.01, batch size 32), maximizing next-token likelihood. The tool-augmented model's target sequences include the opening and closing `<tool>` tags plus the arithmetic sequence to be passed into the calculator. The baseline and finetuned model's targets consist only of the final numeric answer. Figure 2 illustrates this training pipeline.

Finally, we evaluate on the testing split of the EleutherAI arithmetic benchmark [1], reporting exact-match accuracy and, for the tool-augmented condition, the percentage of correctly formatted API calls.

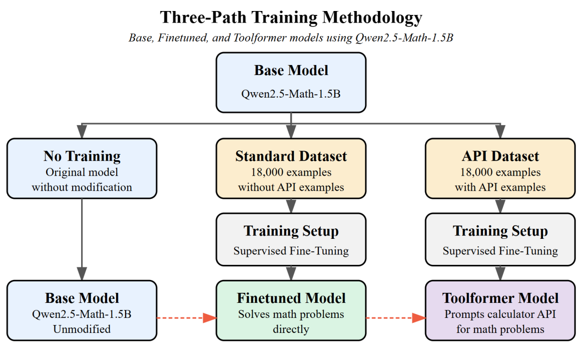

Three-Path Training Methodology
Base, Finetuned, and Toolformer models using Qwen2.5-Math-1.5B
Base Model
Qwen2.5-Math-1.5B
No Training
Original model without modification
Standard Dataset
18,000 examples without API examples
API Dataset
18,000 examples with API examples
Training Setup
Supervised Fine-Tuning
Training Setup
Supervised Fine-Tuning
Base Model
Qwen2.5-Math-1.5B Unmodified
Finetuned Model
Solves math problems directly
Toolformer Model
Prompts calculator API for math problems

Figure 2: Fine-tuning pipeline

4 Results and Analysis

Figure 3 shows the per-task error rates on the EleutherAI arithmetic suite for both the baseline, finetuned, and tool-augmented SmolLM2-135M models.

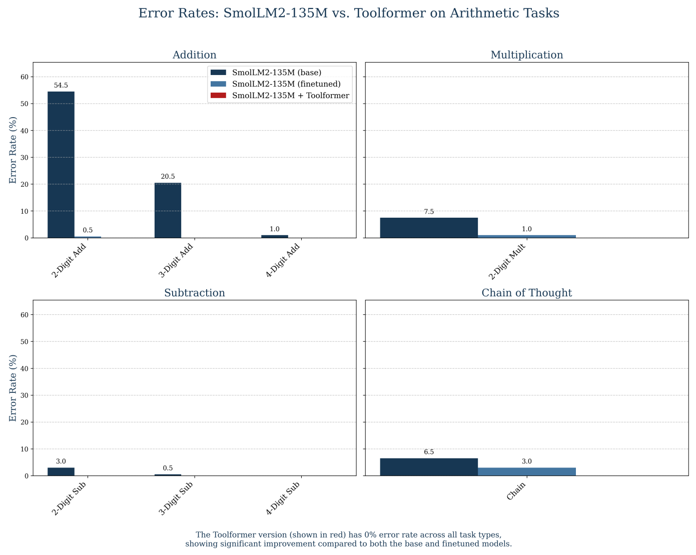


Figure 3: Error rate comparison across tasks: baseline vs. tool-augmented SmolLM2-135M.

Fine-tuning SmolLM2-135M on the arithmetic benchmark without any tool-call syntax (the baseline condition) yielded persistently low performance: overall exact-match accuracy remained below 10%, with near-zero success on single-digit multiplication and multi-step "chain of calculation" problems. These results suggest that, absent any external API scaffolding, the model is unable to internalize even simple arithmetic patterns within the limited fine-tuning budget.

When we incorporate explicit calculator API calls into the training data (the Toolformer-style condition), the model correctly emits both opening and closing `<tool:calculator>` tags in fewer than 20% of evaluation prompts. Consequently, end-to-end accuracy actually declines to below 5%, as misformatted tool-call annotations frequently prevent the extraction of the final numeric result. By contrast, Schick *et al.* reported absolute gains of 15–30% on comparable arithmetic tasks using much larger models [1], indicating that our 135 M-parameter setup fails to replicate their improvements.

We found various key obstacles in implementing our approach. We trained initially with various small models including SmolLM2-135M and GPT-2, but found limited success in getting rational responses from the Toolformer trained model. To improve training time, we implemented LoRA (Low-Rank Adaptation) before using HuggingFace's SFTTrainer to isolate our fine-tuning. With this approach, we found positive signs of improvements with finetuning, but *Toolformer* failed to conform to the desired output.

A closer examination of failure reveals three key factors. First, the limited capacity of a 1.5 M-parameter model appears insufficient to learn both the syntactic patterns of API calls and the semantic reasoning required for arithmetic. Second, our modest training of 18 000 examples using SFTTraining may be inadequate for stabilizing the joint objective of syntax and calculation. Third, exact-match evaluation is unforgiving of minor deviations in tokenization or whitespace, so even nearly correct tool-call outputs are scored as failures.

5 Reflections and Future Directions

Our experiments underscore that self-supervised tool usage is highly sensitive to underlying model capacity. In particular, sub-200 M-parameter models struggle to internalize new token sequences representing API calls while retaining arithmetic reasoning. Simply porting the original Toolformer pipeline to smaller architectures thus proves insufficient without targeted adaptations in data curation and training strategy.

Looking ahead, we identify three avenues for improvement. Enhancing the diversity of synthetic tool-usage examples—particularly by varying whitespace, token boundaries, and expression complexity—could reduce formatting errors. Introducing intermediate supervision, such as an auxiliary tag-classification loss or curriculum learning, may help the model to robustly learn API-call syntax before tackling arithmetic reasoning. Finally, exploring models in the 500M – 1B parameter range could reveal the minimal scale at which tool-augmented fine-tuning becomes reliably effective.

Beyond arithmetic, extending this approach to other tool types—such as web search or knowledge-base queries—will be crucial for assessing the generality of self-supervised tool learning. Moreover, incorporating solution-guidance fine-tuning techniques [3] to reveal intermediate reasoning steps may further enhance performance and interpretability.

References

[1] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). *Toolformer: Language Models Can Teach Themselves to Use Tools*. arXiv. https://arxiv.org/abs/2302.04761

[2] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., … & Amodei, D. (2020). *Scaling laws for neural language models*. arXiv. https://arxiv.org/abs/2001.08361

[3] Bi, J., Wu, Y., Xing, W., & Wei, Z. (2024). *Enhancing the Reasoning Capabilities of Small Language Models via Solution Guidance Fine-Tuning*. arXiv. https://arxiv.org/abs/2412.09906

[4] Qwen Team. (2024). *Qwen2.5-Math-1.5B*. Hugging Face. https://huggingface.co/Qwen/Qwen2.5-Math-1.5B