Summary of Toolformer

Core Idea

- Toolformer is a language model (LM) trained to **teach itself** how to use external tools (like a calculator, search engine, Q&A system) via simple APIs.
- The goal is **to combine the broad reasoning abilities of LM**s with the factual accuracy and computational power of specialized tools.

Limitation of LLMs

- **Poor Mathematical Skills:** They struggle with precise arithmetic.
- **Outdated Knowledge:** They cannot access information created after their training date.
- **Factual Hallucination:** They have a tendency to invent facts.
- **No Awareness of Time:** They don't understand concepts like "today."

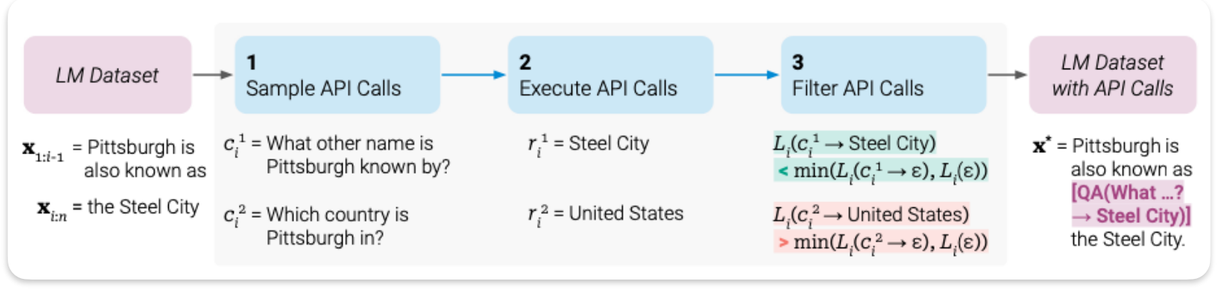
Self-Supervised Tool Learning (3 steps)
1) Sample API Calls: The LM reads through a massive text dataset and generates potential API calls that *might* be useful for predicting the next sequence of words.

2)**Filter API Calls:** It then executes these calls and evaluates the results. Only the API calls that **actually help the model predict future tokens more accurately** (i.e., reduce its prediction loss) are kept.

3)**Fine-Tune the Model:** he LM is then fine-tuned on this new, augmented dataset, which now includes the "useful" API calls embedded directly in the text.

Key advantages
1) No Human Annotation : It learns from just a handful of examples for each API, making the approach scalable and cost-effective.
2) Preserves Generality: The model decides for itself when and how to use a tool, so it doesn't lose its core language skills and isn't restricted to a specific task.
3)Improved Performance : Toolformer achieves substantially better zero-shot performance on a wide range of tasks, often competing with much larger models while maintaining its original abilities.

Process



Overall Strategy

Goal : To equip a language model (M) with the ability to use various tools through text-based API calls.
Methods: Augment a standard text dataset with useful API calls, then fine-tune the original model on this new, enriched dataset.
Performance: A 6.7B parameter Toolformer (based on GPT-J) significantly outperforms the much larger GPT-3 on several zero-shot tasks without losing its core language abilities.

API Call representation :
(api_name, input)

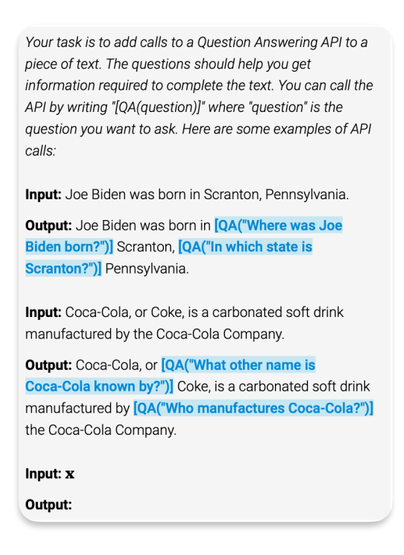
These calls are embedded into text using special tokens.

- **The call itself:** <API> api_name(input) </API>
- **The call with its result r:** <API> api_name(input) → r </API>

Three-Step Process to Create the Training Data

Step 1: Sampling API Calls

- For each tool (e.g., Question Answering), create a prompt with a few examples showing how to use its API
- **Identify Call Locations**: The model (M) analyzes a text and identifi the best positions to insert an API call by calculating the probability of the <API> token appearing at each position.
- **Generate Call Content :** At these identified positions, the model genera the actual API call content (e.g., [QA("Who manufactures Coca-Cola?")]).


Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: x

Output:

Step 2: Executing API Calls

- execute all API calls generated by M to obtain the corre- sponding results.

Step 3: Filtering API Calls

- execute all API calls generated by M to obtain the corre- sponding results.

Intuitively, an API call is helpful to M if providing it with both the input *and* the output of this call makes it easier for the model to predict future tokens

Step 4: FineTuning

- The model (M) is then fine-tuned on the final dataset, which is augmented with only these useful, filtered API calls.

Step 5: Inference

- perform regular decoding until M produces the "→" token, indicating that it next expects the response for an API call

Tools

1. Question Answering : Use Atlas
2. Wikipedia Search Engine: BM25
3. Machine Translation: NLLB Model
4. Calendar: Return current date
5. Calculator

Experiments

1.whether our approach enables a model to use tools without any further supervision and to decide for itself when
2. how to call which of the available tools.

Dataset Generatio :

Experimental Setup

Core Model and Data:

Language Model GPT-J(M)
Language Modeling Dataset (C) - subset of CCNet
Batch Size : 128
Learning Rate: 1 * 10^-5 with linear warmup for the first 10% of training.

Baseline Models for Comparison

Language Model GPT-J without Finetuning
GPT-J + CC: Finetuned on C
Toolformer : GPT - finetuned on C
Toolformer (Disabled) **- same as toolformer but API calls are manually disabled during decoding**
Larger models :OPT(66B) and GPT-3(175B)

Toolformer (Disabled): The same model as Toolformer but API calls are disabled during decoding(achieved by setting the probability of token=0)

LAMA BenchMark Evaluation

Scenario : Complete short statements with missingfacts from LAMA SQuAD, Google-RE, T-Rex
Data-processing: Filtered for examples where the mask toen is the final token to enable left-to right processing
Evaluation Metric : Lenient to check if the correct word is within the first 5 worlds predctied by the model
Toolformer Restriction : blocked Wikipeadia Search API

Toolformer on the Question and Answering

Model	SQuAD	Google-RE	T-REx
GPT-J	17.8	4.9	31.9
GPT-J + CC	19.2	5.6	33.2
Toolformer (disabled)	22.1	6.3	34.9
Toolformer	**33.8**	**11.5**	**53.5**
OPT (66B)	21.6	2.9	30.1
GPT-3 (175B)	26.8	7.0	39.8

Table 3: Results on subsets of LAMA. Toolformer uses the question answering tool for most examples, clearly outperforming all baselines of the same size and achieving results competitive with GPT-3 (175B).

Toolformer on the mathematical reasoning

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

Table 4: Results for various benchmarks requiring mathematical reasoning. Toolformer makes use of the calculator tool for most examples, clearly outperforming even OPT (66B) and GPT-3 (175B).

Toolformer with API access outperforms =>the model independently decides to ask the question answering tool for the required information in all most all questions (98.1%)

↓

★ The power of knowledge via API
1. Proactive API Invocation
2. Access to vas External knowledge
3.Limitations of internal knowledge

Question Answering

Evaluation : check whether the first 20 words predicted by a model contain the correct answer

Model	WebQS	NQ	TriviaQA
GPT-J	18.5	12.8	43.9
GPT-J + CC	18.4	12.2	45.6
Toolformer (disabled)	18.9	12.6	46.7
Toolformer	**26.3**	**17.7**	**48.8**
OPT (66B)	18.6	11.4	45.7
GPT-3 (175B)	29.0	22.6	65.9

Toolformer, an AI model, performed better than other AI models
Toolformer falls short compared to the much larger and more powerful GPT-3 model

↓

Reasons why toolformer behind GPT-3
the simplicity of its search function =sometimes it returns irreleveant results

Multilingual QnA

Task : evaluate Toolformer and all baseline models on MLQA(Multilingual question-answering benchmark)
Description : (questions provided in English, Arabic, German, Spanish, Hindi, Vietnam, simplified Chinese)model needs to be able to understand both the paragraph and the question

Model	Es	De	Hi	Vi	Zh	Ar
GPT-J	15.2	**16.5**	1.3	8.2	**18.2**	**8.2**
GPT-J + CC	15.7	14.9	0.5	8.3	13.7	4.6
Toolformer (disabled)	19.8	11.9	1.2	10.1	15.0	3.1
Toolformer	**20.6**	13.5	**1.4**	**10.6**	16.8	3.7
OPT (66B)	0.3	0.1	1.1	0.2	0.7	0.1
GPT-3 (175B)	3.4	1.1	0.1	1.7	17.7	0.1
GPT-J (All En)	24.3	27.0	23.9	23.3	23.1	23.6
GPT-3 (All En)	24.7	27.2	26.1	24.9	23.6	24.0

Table 6: Results on MLQA for Spanish (Es), German (De), Hindi (Hi), Vietnamese (Vi), Chinese (Zh) and Arabic (Ar). While using the machine translation tool to translate questions is helpful across all languages, further pretraining on CCNet deteriorates performance; consequently, Toolformer does not consistently outperform GPT-J. The final two rows correspond to models that are given contexts and questions in English.

Temporal Datasets

Task : evaluate the utility of calendar API => test two temporal datasets (TEMPLAMA and DATESET)
Result : Toolformer outperformes all baseline models on both datasets

Model	TEMPLAMA	DATESET
GPT-J	13.7	3.9
GPT-J + CC	12.9	2.9
Toolformer (disabled)	12.7	5.9
Toolformer	**16.3**	**27.3**
OPT (66B)	14.5	1.3
GPT-3 (175B)	15.5	0.8

Scaling Laws

Task : Investigate how the ability to ask external tools for help affects performance
How : applied four smaller models from GPT-2 with 124M, 355M, 775M and 1.6B
Tools : QnA system, calculating system, and wikipedia search engine

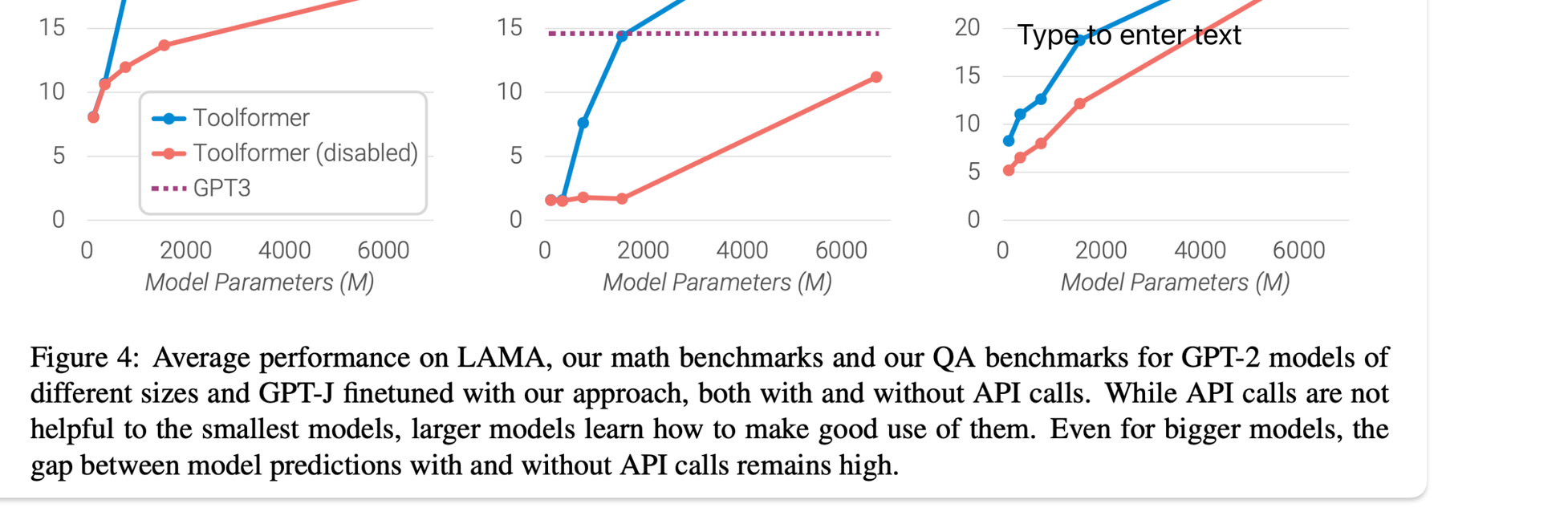


Figure 4: Average performance on LAMA, our math benchmarks and our QA benchmarks for GPT-2 models of different sizes and GPT-J finetuned with our approach, both with and without API calls. While API calls are not helpful to the smallest models, larger models learn how to make good use of them. Even for bigger models, the gap between model predictions with and without API calls remains high.

Result :
The ability to leverage the provided tools only emerges at around 775M parameters
While models become better at solving tasks without API calls they grow in size, their ability to make good use of the provided API improves at the same time
A large gap btw predictions with/ without API calls in bigger models

My learnings:

- Fine-tuning with API calls does not degrade the fundamental language modeling performance
- As the model size increases, model gets better at solving tasks without API calls , but their ability to effectively use the provided API also improves
- Even for larger models there are remained a significant gap on predictions between ones with API calls and not

↓

The smarter the model gets, the better it uses tools and the better it uses tools , the smarter it becomes