Toolformer

Language Models Can Teach Themselves to Use Tools

Michal Pospěch

March 28, 2023

Introduction

- Classic LLMs
  - good at solving new tasks
  - struggle with some basic tasks (arithmetics, information lookup. . .)
- Provide them with external tools
- Model deciding which API to call, when to call it, with what parameters and how to incorporate the output
- Performance on pair with much larger model on downstream tasks

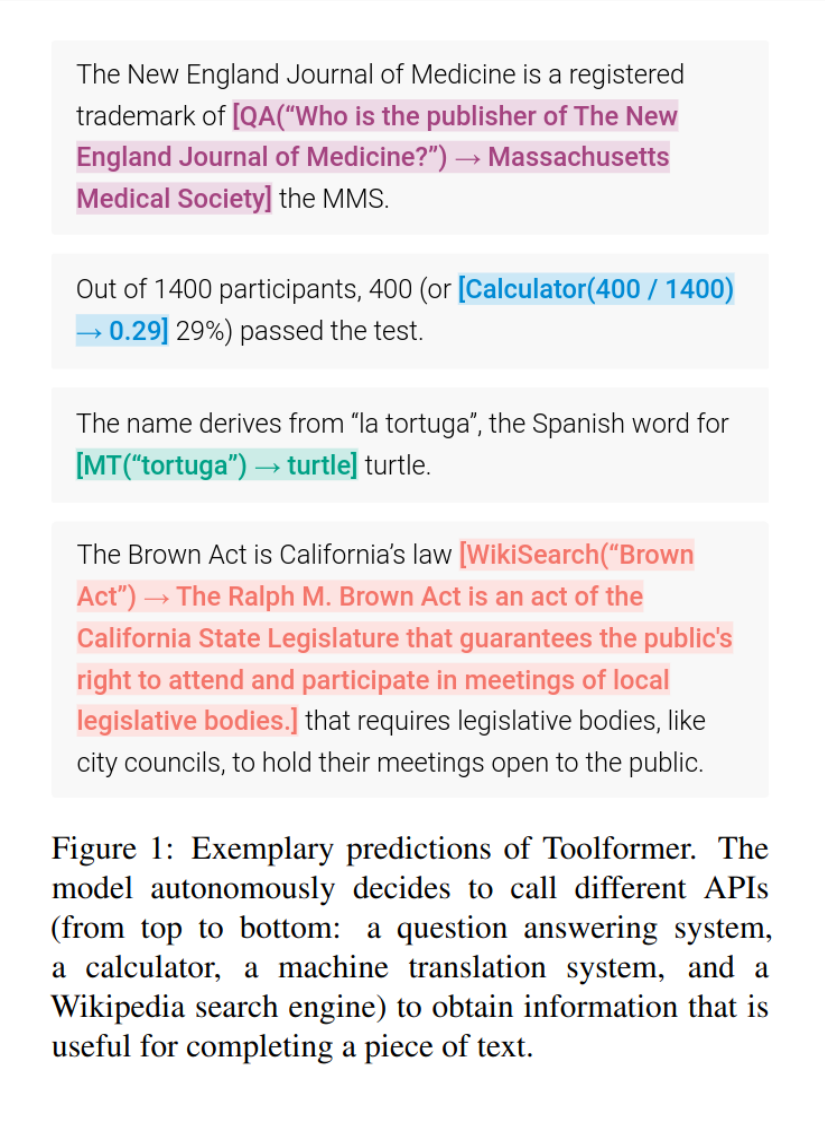

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

Outline

- Why?
- How does it work?
- Tools
- Experiments

Why?

Why?

- LLMs have limitations
  - Up-to-date info on recent events
  - Tendency to hallucinate facts
  - Difficulties understanding low-resource languages
  - Lack of mathematical skills
  - Unawareness of time progression
- Can be solved by incorporating tools
  - Large amount of human annotations
  - Task-specific setting for a particular tool

How does it work?

How?

- In-context learning
  - Dataset generation from scratch
  - Few human-written examples
  - LM annotated huge dataset with potential API calls
  - Self-supervised loss to filter the API calls
  - Finetune the model
- Dataset agnostic
  - Can use the same as the pretrained model $\rightarrow$The generalisation ability is kept

How?

- API call $c = (a_c, i_c)$
- representation

$$e(c) = \texttt{<API>}a_c(i_c)\texttt{</API>}$$

$$e(c, r) = \texttt{<API>}a_c(i_c) \rightarrow r\texttt{</API>}$$

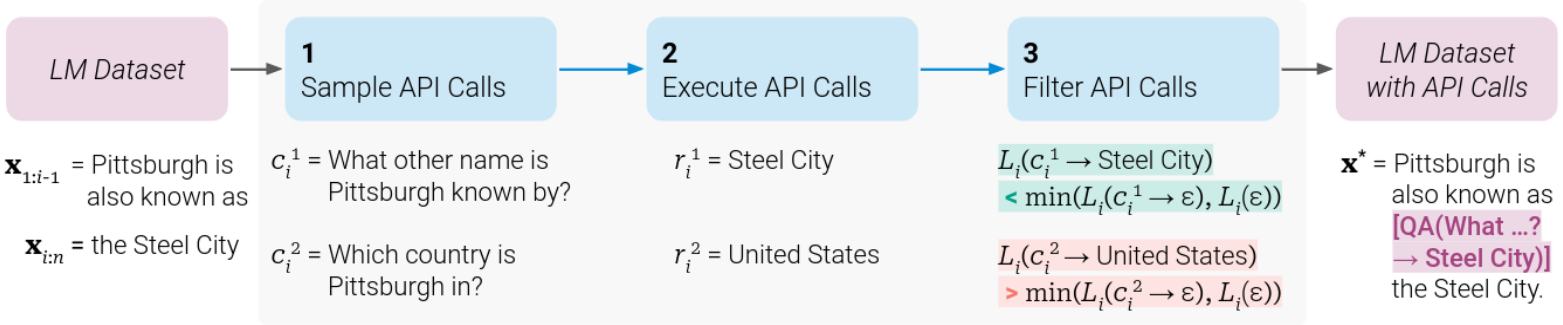

Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text $\mathbf{x}$, we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \ldots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text $\mathbf{x}^*$.

Sampling API Calls

- Prompt $P(\mathbf{x})$ for each API call to annotate example $\mathbf{x} = x_1, \ldots x_n$ with API calls
- Generate k candidates for API calls by computing

$$p_i = p_M(\texttt{<API>} | P(\mathbf{x}), x_{1:i-1})$$

- Keep at most k positions for which $p_i > \tau_s$
- Sample at each position to get up to m API calls

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:

Figure 3: An exemplary prompt $P(\mathbf{x})$ used to generate API calls for the question answering tool.

Filtering API calls

- Weighted crossentropy loss for M if x prefixed with z

$$L_i(\mathbf{z}) = -\sum_{j=i}^{n} w_{j-i} \cdot \log p_m(x_j|\mathbf{z}, x_{1:j-1})$$

- 2 instantiations

$$L_i^+ = L_i(e(c_i, r_i))$$
$$L_i^- = \min(L_i(\varepsilon), L_i(e(c_i, \varepsilon)))$$

- Keep if $L_i^- - L_i^+ \geq \tau_f$

Model Finetuning

- From text $\mathbf{x}$ with API call and result (c_i, r_i) at position i we create $\mathbf{x}^* = x_{1:i-1}, e(ci, ri), x_{i:n}$
- We do so for all $\mathbf{x} \in \mathcal{C}$ and get $\mathcal{C}^*$ and use it to finetune M using stanard language modelling objective
- Finetuning on $\mathcal{C}^*$ exposes M to same content as finetuning on $\mathcal{C}$

Tools

Tools

- Question answering - Atlas
- Calculator
- Wikipedia search - BM25 retrieval
- Machine Translation system - NLLB (600M parameter model) + fasttext
- Calendar

API Name	Example Input	Example Output
Question Answering	Where was the Knights of Columbus founded?	New Haven, Connecticut
Wikipedia Search	Fishing Reel Types	Spin fishing > Spin fishing is distinguished between fly fishing and bait cast fishing by the type of rod and reel used. There are two types of reels used when spin fishing, the open faced reel and the closed faced reel.
Calculator	27 + 4 * 2	35
Calendar	ϵ	Today is Monday, January 30, 2023.
Machine Translation	sûreté nucléaire	nuclear safety

Table 1: Examples of inputs and outputs for all APIs used.

Experiments

Experimental setup

- Dataset - CCNet
- Language Model - GPT-J
- Thresholds τ set individually per tool
- Baseline models
  - GPT-J
  - GPT-J + CC
  - Toolformer
  - Toolformer (disabled)
  - OPT (for comparison)
  - GPT-3 (for comparison)

Downstream Tasks

- Zero-shot setup
- Greedy decoding with slight modification (API call when <API> one of $k = 10$ most likely tokens)
- Tasks:
  - LAMA
  - Math
  - Question answering
  - Multilingual question answering
  - Temporal datasets

LAMA

- SQuAD, GoogleRE, T-REx subsets of LAMA benchmark
- Complete statement with a missing fact
- More lenient evaluation
- No Wikipedia Search API (unfair advantage)
- 98.1% of times the question asking tool is used

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).

Math

- ASDiv, SVAMP, MAWPS
- More lenient evaluation
- Toolfomer (disabled) has strong results
- 97.9% of times the calculator tool is used

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).

Question answering

- Web Questions, Natural Questions, TriviaQA
- More lenient evaluation
- No Question Answer API (unfair advantage, since the QA system was trained on Natural Questions)
- 99.3% of times Wikipedia API is used
- No interactivity (query reformulation) $\rightarrow$advantage of GPT-3 and possible future work

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

Table 5: Results for various question answering dataset. Using the Wikipedia search tool for most examples, Toolformer clearly outperforms baselines of the same size, but falls short of GPT-3 (175B).

Multilingual Question Answering

- MLQA
- Context paragraph in English, question in another language
- 63.8% to 94.9% of times translation is used, 7.3% for Hindi
- Not better than base models, finetuning deteriorates performance
- OPT and GPT-3 fail tu provide answer in English even after being instructed to do so

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

- TempLAMA, DATESET (new)
- only 0.2% of TempLAMA evaluations used calendar tool (mostly Wikipedia and question answering)
- 54.8% of DATESET evaluations used calendar tool

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

Table 7: Results for the temporal datasets. Toolformer outperforms all baselines, but does not make use of the calendar tool for TEMPLAMA.

Scaling

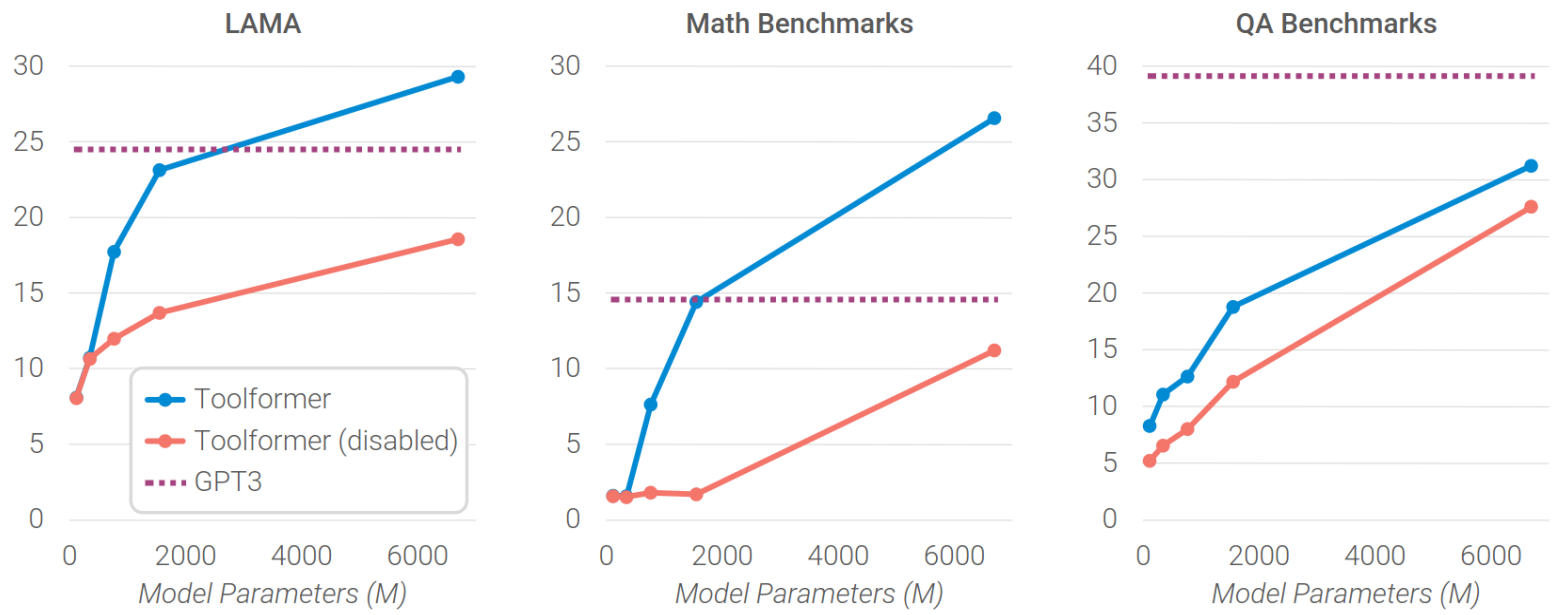

Figure 4: Average performance on LAMA, our math benchmarks and our QA benchmarks for GPT-2 models of different sizes and GPT-J finetuned with our approach, both with and without API calls. While API calls are not helpful to the smallest models, larger models learn how to make good use of them. Even for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

Conclusion

Limitations

- No tool chaining
- No interactive usage of tools
- Sensitive to wording
- Sample-inefficiency
- Computational cost not taken into account

Conclusion

- Self-supervised learning of tool usage
- Finetuning on a large number of sampled API calls
- Better zero-shot performance than base model
- Outperforms much larger models