TOOLFORMER: LANGUAGE MODELS CAN TEACH THEMSELVES TO USE TOOLS

Timo Schick Jane Dwivedi-Yu Roberto Dessì† Roberta Raileanu

Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research †Universitat Pompeu Fabr

9 Feb 2023

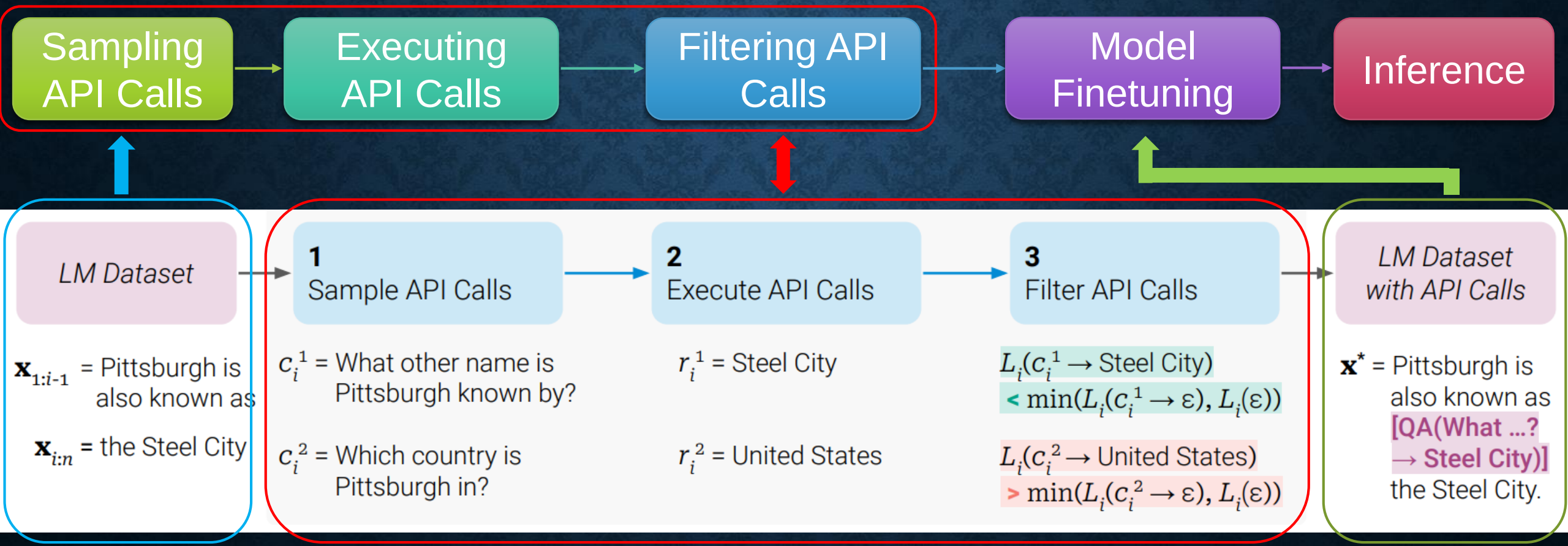
arxiv.org/pdf/2302.04761.pdf

TABLE OF CONTENTS

- Clarify the researcher's ideas
- Experiments & My comments
- Implement (code)

CLARIFY THE RESEARCHER'S IDEAS

APPROACH



Key steps for question answering tool

APPROACH

Datasetof plain texts (**C**)

$C = \{x^1, \ldots, x^{|C|}\}$

API call (**c**)

$c = (a_c, i_c)$

a_c: name of API

i_c: corresponding input

Linearized sequences of API (**e**)

$e(c) = < API > a_c(i_c) </API >$

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

r: API result

" [", "]", "->" instead of *"<API>", "</API>", "→" (special tokens) to work without modifying LM's vocabulary*

Dataset with API Calls (**C***)

$C^* = \{x^{*1}, \ldots, x^{*|C|}\}$

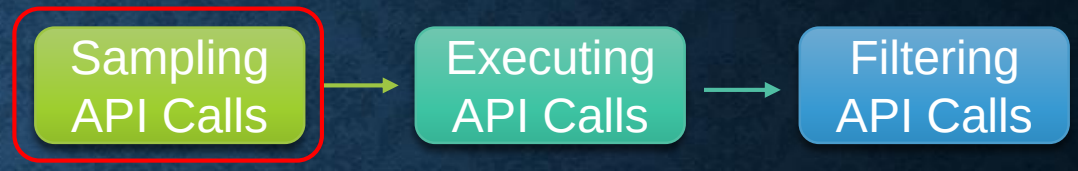
Finetuning language model **M** on **C***

C contains the exact same texts as C*

STEP BY STEP

- Sampling API Calls
- Executing API Calls
- Filtering API Calls
- Model Finetuning
- Inference

SAMPLING API CALLS



- Sequence $x = x_1, \ldots, x_n$
- Prompt **P(x)**
- $p_M = p_M(z_{n+1}|z_1, \ldots, z_n)$ probability that M assigns to token z_{n+1} as a continuation for the sequence $z_1, \ldots, z_n$ → For each i ∈ {1, . . . , n} → $p_i = p_M(< API > |P_X, x_{1:i-1})$ probability that M assigns to starting an API call at position i
- Sampling threshold τ_s → keep all positions $I = \{i|\ p_i > \tau_s\}$ → **k** candidate positions → keep top **k** positions
- For each position i ∈ I → **m** API calls $c_i^1, \ldots, c_i^m$ → $[P(x), x_1, \ldots, x_{i-1}, <API> response\ for\ API\ </API>]$
- M does not generate </API> → discard example

PROMPT P(X)

- Question Answering (2)
- Calculator (5)
- Wikipedia Search (3)
- Machine Translation System (3)
- Calendar (5)

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.

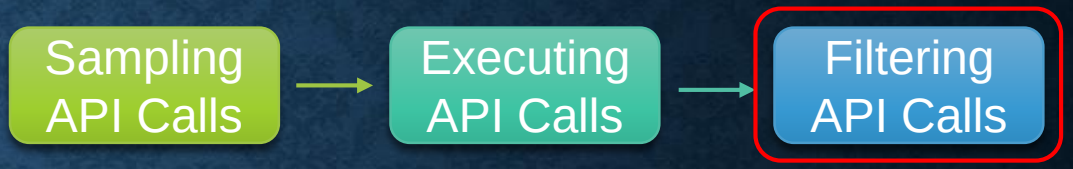
Input: x

Output:

EXECUTING API CALLS

- How this is done depends entirely on the API itself
  - Calling another neural network
  - Executing a Python script
  - Using a retrieval system to perform search over a large corpus
- API call c_i ➔ single text sequence r_i

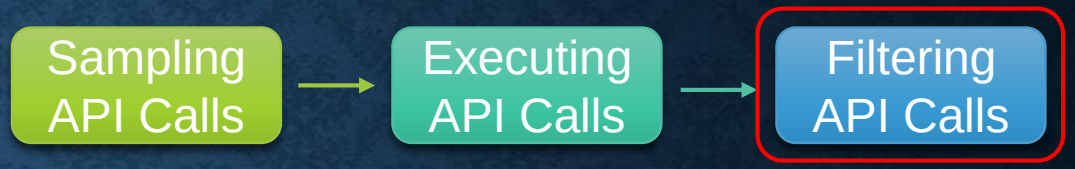
FILTERING API CALLS



- At position i in sequence $x = x_1, \ldots, x_n$ → API call c_i → response r_i → $e(c_i, r_i)$
- A sequence of weights $(w_i | i \in N)$
- $\mathbf{z} \triangleq e(c_i, r_i)$ as a prefix instead of inserting it at position i
- Over the tokens $x_1, \ldots, x_n$ → weighted cross entropy loss for M

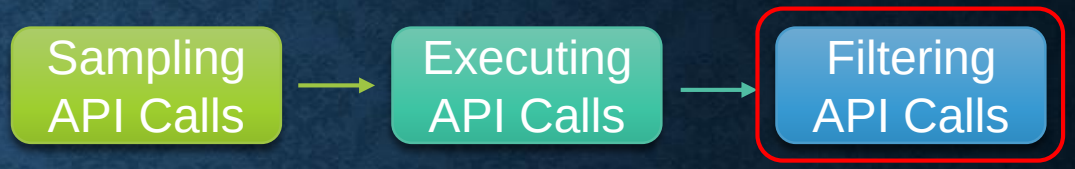
$$L_i(z) = -\sum_{j=i}^{n} w_{j-i} \cdot log p_M(x_j | z, x_{1:j-1})$$

FILTERING API CALLS



- Weighted cross entropy loss $L_i(z) = -\sum_{j=i}^{n} w_{j-i} \cdot log p_M(x_j | z, x_{1:j-1})$
- Consider 3 cases:
  - (a) doing no API call
  - (b) doing an API call and providing the response
  - (c) doing an API call, but not providing the response
- Efficiency of API call → compare (b) with (c) or (a)
- $L_i^B(z) < L_i^A(z)$ or $L_i^B(z) < L_i^C(z)$ → $L_i^B(z) < \min(L_i^A(z), L_i^C(z))$

FILTERING API CALLS



- Weighted cross entropy loss $L_i(z) = -\sum_{j=i}^{n} w_{j-i} \cdot logp_M(x_j|z, x_{1:j-1})$
- Efficiency of API call → $L_i^B(z) < \min(L_i^A(z), L_i^C(z))$
- Empty sequence ε →
  - $L_i^+ = L_i(e(c_i, r_i))$
  - $L_i^- = \min(L_i(\varepsilon), L_i(e(c_i, \varepsilon)))$
- Filtering threshold τ_f → keep API that $L_i^- - L_i^+ \geq \tau_f$

MODEL FINETUNING

- LM Dataset C → LM Dataset with API Calls C*
- Original input $x = x_1, \ldots, x_n$ → new input $x^* = x_{1:i-1}, e(c_i, r_i), x_{i:n}$
- Finetune M on C*, then based on its feedback $e(c_i, r_i)$, M will learn to decide when and how to use which tool can help predict future tokens.

INFERENCE

- $x^* = x_{1:i-1}, e(c_i, r_i), x_{i:n}$
- $e(c, r) = < API > a_c(i_c) \rightarrow r </API >$
- Finetuned M ➔ generated text t ➔ regular decoding, when M produces the "→" token ➔ interrupt decoding ➔ M produces </API> token ➔ continue decoding.

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Fine-tuning

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.

Inference

TOOLS

- **Question Answering**
  - QA system based on another LM - Atlas retrieval-augmented LM fine-tuned on Natural Questions
- **Calculator**
  - Simple numeric calculations - four operations, rounded to two decimal places
- **Wikipedia Search**
  - Search engine, given a search term, extract from comprehensive information and returns short text - a BM25 retriever that indexes the Wikipedia dump from KILT
- **Machine Translation System**
  - Detect source language using the fast-Text classifier, translate into English from any language using multilingual machine translation model - 600M parameter NLLB (200 languages)
- **Calendar**
  - Query, return the current date without taking any inputs, provide temporal context for predictions that require some awareness of time

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

EXPERIMENTS & MY COMMENTS

Subset of CCNet

Dataset (**C**)
$C = \{x^1, \ldots, x^{|C|}\}$

Computational cost ➔ define heuristics

Default: $\tau_s = 0.05, \tau_f = 1.0, k = 5, m = 5$

Calculator, machine translation: $\tau_s = 0.0, \tau_f = 0.5, k = 20, m = 10$

API call (**c**)
$c = (a_c, i_c)$

obtaining C* from C

Linearized sequences of API (**e**)
$e(c) = < API > a_c(i_c) </API >$
$e(c, r) = < API > a_c(i_c) \rightarrow r </API >$

Dataset (**C***)
$C^* = \{x^{*1}, \ldots, x^{*|C|}\}$

more likely

to be helpful

batch size: 128
learning rate: 1·10−5
linear warmup: first 10% of training

Finetuning **M** on **C***

GPT-J

Make sure that API is actually helpful ➔ weighting function

$$w_t = \frac{\tilde{w}_t}{\sum_{s \in \mathbb{N}} \tilde{w}_s} \text{ with } \tilde{w}_t = \max(0, 1 - 0.2 \cdot t)$$

- GPT-J: w/o finetuning
- GPT-J + CC: finetuned on C
- Toolformer: GPT-J finetuned on C*
- Toolformer (disabled): disable API calls (decode)

API	Number of Examples		
	$\tau_f = 0.5$	$\tau_f = 1.0$	$\tau_f = 2.0$
Question Answering	51,987	18,526	5,135
Wikipedia Search	207,241	60,974	13,944
Calculator	3,680	994	138
Calendar	61,811	20,587	3,007
Machine Translation	3,156	1,034	229

Table 2: Number of examples with API calls in $\mathcal{C}^*$ for different values of our filtering threshold τ_f.

Filtering threshold τ_f ➔ keep API that $L_i^- - L_i^+ = \tau_f$

DOWNSTREAM TASKS

- SETUP
- prompted zero shot setup - instructed to solve but do not provide any in-context examples
- greedy decoding - selects the word with the highest probability as the next word $w_t = \underset{w}{argmax}\, P(w|w_{1:t-1})$, but one modification
- Standard: the most likely token → Modification: the k most likely tokens → that w_t = <API> → k = 10
- one API call/input → does not get stuck in a loop in case API call not providing the response

DOWNSTREAM TASKS

Standard: the most likely token ➔ Modification: the k most likely tokens ➔ that w_t = <API> ➔ k = 10

	T-REx				WebQS			
k	All	AC	NC	%	All	AC	NC	%
0	34.9	–	34.9	0.0	18.9	–	18.9	0.0
1	47.8	53.0	44.3	40.3	19.3	17.1	19.9	8.5
3	52.9	58.0	29.0	82.8	**26.3**	26.5	6.6	99.3
10	**53.5**	54.0	22.5	98.1	**26.3**	26.4	–	100.0

increasing k ➔ model doing API calls for more examples

k = 1 ➔ model is calibrated to some extent ➔ decide to call APIs (better perform) instead of NOT call APIs (badly perform).
higher values of k ➔ lose this calibration

EVALUATE MODELS

- Considering the following criterions:
- The factual and commonsense knowledge contained in pretrained language models (on SQuAD, Google RE and T-REx subsets of the LAMA)
- Performance on specific tasks when compared with other models
  - Mathematical reasoning abilities on ASDiv, SVAMP, MAWPS
  - Question Answering on Web Questions, Natural Questions, TriviaQA
  - Multilingual Question Answering on MLQA
  - Calendar API's utility on TEMPLAMA, generated DATESET
- Language modeling performance on WikiText, CCNet not used in training (10,000)
- Ability to ask external tools for help affects performance using GPT-J and GPT-2 family (124M, 355M, 775M, 1.6B) for tools: question answering, calculator, Wikipedia search.

DOWNSTREAM TASKS

The factual and commonsense knowledge on SQuAD, Google RE and T-REx subsets of the LAMA
Task: complete a short statement with a missing fact (e.g., a date or a place)

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

question answering tool (98.1%); a different tool (0.7%) or no tool (1.2%)

LAMA was designed to evaluate masked language models ➔ filter out mask token is not the final token.
Lenient evaluation criteria instead of exact match, the correct word is in the top five predicted words.
LAMA is based on statements (Wikipedia), prevent Toolformer from using the Wikipedia Search API

DOWNSTREAM TASKS

Performance on specific tasks

Mathematical reasoning abilities on ASDiv, SVAMP, MAWPS

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

Operations: "+", "−", "∗", "/"
Heuristic filters for CCNet & processing criterions

- window of 100 tokens → >= 3 numbers, one is the result of the operation of the other two.
- "=", "equals", "equal to", "total of", "average of" + a number
- >= 3 numbers (random 1%)

calculator tool (97.9%)

Lenient evaluation criterion, output is a number → check for the first predicted number

Surmise: finetuned on many examples of API calls and their results → improving mathematical capabilities

DOWNSTREAM TASKS

Performance on specific tasks

Question Answering on Web Questions, Natural Questions, TriviaQA

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

Simplicity (not a good match)
Inability to interact with itself, reformulating or browsing through the top results ➔ future work

Wikipedia search (99.3%)

Lenient evaluation criteria instead of exact match, the correct word is in the top 20 predicted words.
Question-answering tool: Atlas model finetuned on Natural Questions. Atlas large (obtaining C*), Atlas-xxl (Inference) ➔ task solving is trivial ➔ disable question-answering tool

DOWNSTREAM TASKS

Performance on specific tasks

Multilingual Question Answering on MLQA

Machine translation tool: NLLB 600M (training & inference), fastText classifier (source language detection)
Text chunks (multilingual language) ➔ target language (English)
Preprocessing:

- Text chunks size: 10 tokens
- Middle text chunk is in a language other than English (fastText classifier, confidence > 0.8)
- Any text chunks only numbers or special symbols

Input: 近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 **En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales** 研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲

DOWNSTREAM TASKS

Performance on specific tasks

Multilingual Question Answering on MLQA

Input ➔ API calls (Machine translation tool) ➔ 近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 <API> response of API </API> En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales <API> response of API </API> 研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲 <API> response of API </API>

<API> response of API </API>

近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 <API> response of API </API> ~~En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales~~ ~~研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲~~

DOWNSTREAM TASKS

Performance on specific tasks

Multilingual Question Answering on MLQA

Input ➔ API calls (Machine translation tool) ➔ 近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 <API> response of API </API> En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales <API> response of API </API> 研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲 <API> response of API </API>

<API> response of API </API>

近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales <API> response of API </API> ~~研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲~~

DOWNSTREAM TASKS

Performance on specific tasks

Multilingual Question Answering on MLQA

Input ➔ API calls (Machine translation tool) ➔ 近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 <API> response of API </API> En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales <API> response of API </API> 研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲 <API> response of API </API>

<API> response of API </API>
近年来，科学家们根据蛋白质的结构创作了音乐，作为更好地向公众普及科学的创造性方式，但由此产生的歌曲并不总是悦耳 En un estudio que aparece el 29 de septiembre en la revista Heliyon, los investigadores utilizan el estilo de los géneros musicales existentes para guiar la estructura de las canciones proteicas y hacerlas más musicales 研究人员以肖邦的幻想即兴曲和其他古典作品的风格为指导，将蛋白质转化为具有巨大音乐性的歌曲 <API> response of API </API>

DOWNSTREAM TASKS

Performance on specific tasks

Multilingual Question Answering on MLQA

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

*Distribution shift: CCNet compared to GPT-J's original pretraining data

*Multilingual aspect: OPT and GPT-3 fail to provide answers in English with language differences between context and question. Context and question in English ➔ best performance (GPT-3)

Machine translation tool (63.8-94.9%), Hindi (7.3%)

Context in English, question in Arabic, German, Spanish, Hindi, Vietnamese, Simplified Chinese ➔ understand paragraph and question. Percentage of times that correct word is in the top 10 predicted words.

DOWNSTREAM TASKS

Performance on specific tasks

Calendar API's utility

TEMPLAMA (Wikidata) contains cloze queries about facts that change with time.

DATESET generated through a series of templates, but populated using a combination of random dates/durations. Randomly 500 "current dates" ➔ randomly within four-year range "past date" and "future date" ➔ fill templates

Assumption: The calendar date is created document date. Extract from the URL, if present. Otherwise, filter out and leave ~18%.

Template	Size
How many days {ago was, are there until} {*past_date*, *future_date*}?	400
What {day of the week, day of the month, month, year} was it (*current_date* – *past_date*) {days, weeks, months, years} ago?	800
What {day of the week, day of the month, month, year} will it be in (*future_date* – *current_date*) days?	800
What day of the week {is, was} it on {*past_date*, *future_date*}?	400
What {day of the week, day of the month, month, year} {is, was} it {the day before yesterday, yesterday, today, tomorrow, the day after tomorrow}?	4,000
What {day of the week, day of the month, month} {is, was} *holiday* this year?	1,800
How many {days, weeks, months, years} {ago was, are there until} *holiday* this year?	1,200
Total	9,400

DOWNSTREAM TASKS

Performance on specific tasks

Calendar API's utility on TEMPLAMA, generated DATESET

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

A little help (exact date) because of specific and rare on TEMPLAMA

The best action: query current date (calendar), then query question answering with this date.

But two API calls per example & Independently sampled API calls on training.

Mostly to Wikipedia search and question answering, calendar (0.2%) for TEMPLAMA.

But calendar (54.8%) for DATESET.

Lenient evaluation criteria instead of exact match, the correct word is in the top five predicted words.

DOWNSTREAM TASKS

Language modeling performance on WikiText, CCNet not used in training (10,000)

Model	WikiText	CCNet
GPT-J	**9.9**	10.6
GPT-J + CC	10.3	**10.5**
Toolformer (disabled)	10.3	**10.5**

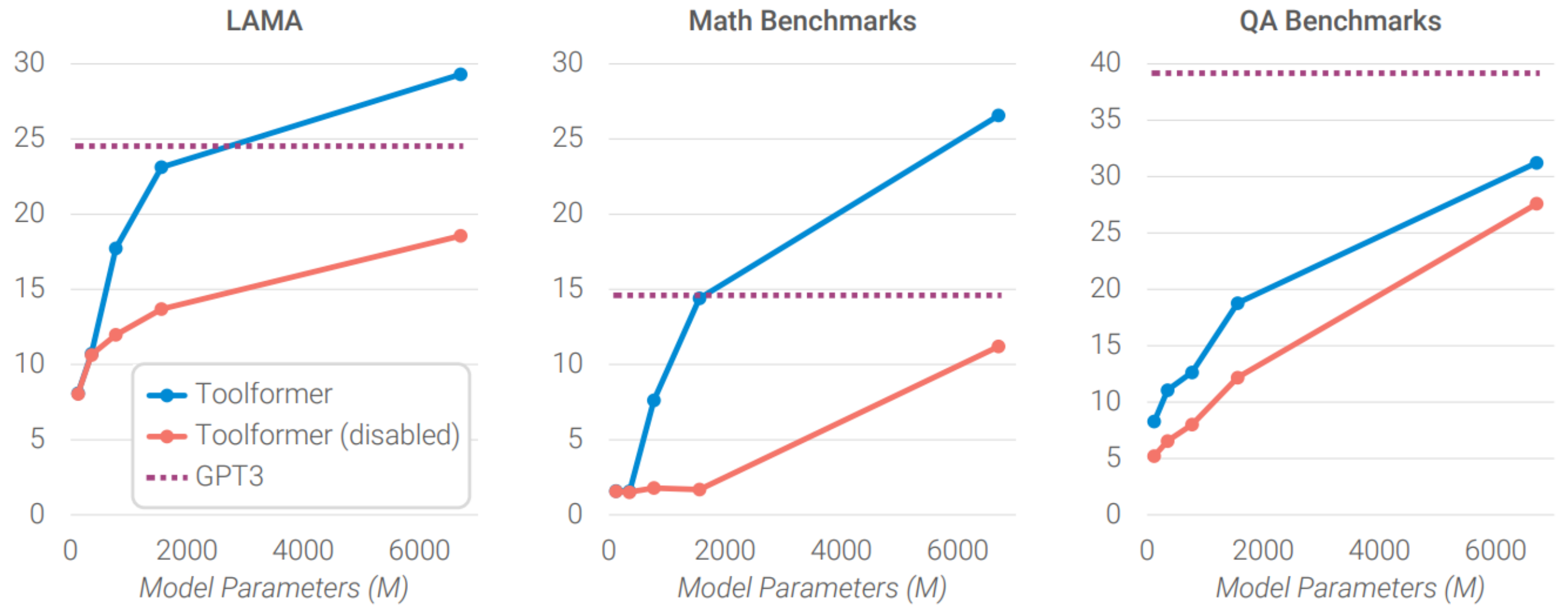
Original pretraining data for GPT-J is more similar to WikiText than randomly subset of CCNet. Training on C* compared to C does not lead to an increase in perplexity when API calls are disabled at inference time.

Ability to ask external tools for help affects performance using GPT-J and GPT-2 family (124M, 355M, 775M, 1.6B) for tools: question answering, calculator, Wikipedia search engine

Leverage the provided tools at models ~775M. Smaller models achieve similar performance w/ and w/o tools. But achieve difference for QA benchmarks (Wikipedia search engine mainly be used). When the model size grows, it can solve tasks without API calls. So, it also can make good use of the provided API. And the gap between model predictions with and without API calls remains high.

DOWNSTREAM TASKS

Ability to ask external tools for help affects performance using GPT-J and GPT-2 family (124M, 355M, 775M, 1.6B) for tools: question answering, calculator, Wikipedia search engine



LIMITATIONS

Generated independently tools ➔ limitation in using tools in chain (output of one as input for another)

NOT allow to use tool in an interactive way (search engines) ➔ Solution: take advantage of possible results or refine search query by itself

LMs - sensitive to a prompt ➔ sensitive to exact wording (input when deciding to call APIs)

Sample-inefficient (tool-dependent): million results but few thousand useful calls ➔ Solution: iteratively apply our approach, similar to bootstrapping approaches

API calls ➔ NOT consider computational cost for tool-dependent

My confused

Lenient evaluation criteria instead of exact match (tool-dependent), the correct word is in the top k predicted words ➔ what happened in case they used exact match? It too bad?

C* was obtained from C (C* contains the exact same texts as C) and their assumption that finetuning M on C* exposes it to the same content as on C ➔ Why is this true in case there are response of API calls in C* while not in C? Mabe the reason is that when obtaining C* from C, they filtered alters the distribution of training and assume that the remaining examples are close enough to the original distribution so that M's language modeling abilities remain unaffected? But how?

They used the weighting function to make sure that API calls happen close to where the information provided by the API is actually helpful for the model BUT I can find more clearly explaination for this function nowhere in their paper.

$$w_t = \frac{\tilde{w}_t}{\sum_{s \in \mathbb{N}} \tilde{w}_s} \text{ with } \tilde{w}_t = \max(0, 1 - 0.2 \cdot t)$$

IMPLEMENT (CODE)

- lucidrains/toolformer-pytorch: Implementation of Toolformer, Language Models That Can Use Tools, by MetaAI (github.com)
- conceptofmind/toolformer(github.com) - Tool Former Source Code: LLM using API (youtube.com)
- xrsrke/toolformer: Implementation of Toolformer: Language Models Can Teach Themselves to Use Tools (github.com)