

LLMs + External Tools

Andi @ Seminar.ML



Overview

1. Toolformer: Language Models Can Teach Themselves to Use Tools
 1. <https://arxiv.org/pdf/2302.04761.pdf>
2. ART: Automatic multi-step reasoning and tool-use for large language models
 1. <https://arxiv.org/abs/2303.09014>
3. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face
 1. <https://arxiv.org/pdf/2303.17580.pdf>
4. LangChain: How to use ChatGPT with your (research) documents?
 1. https://python.langchain.com/en/latest/getting_started/getting_started.html

Toolformer

- Design prompts which use API calls and fine-tune using these

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.

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:

Toolformer

- Fine-tune GPT-J model using API's
- Predict by adding API calls during prediction

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).

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

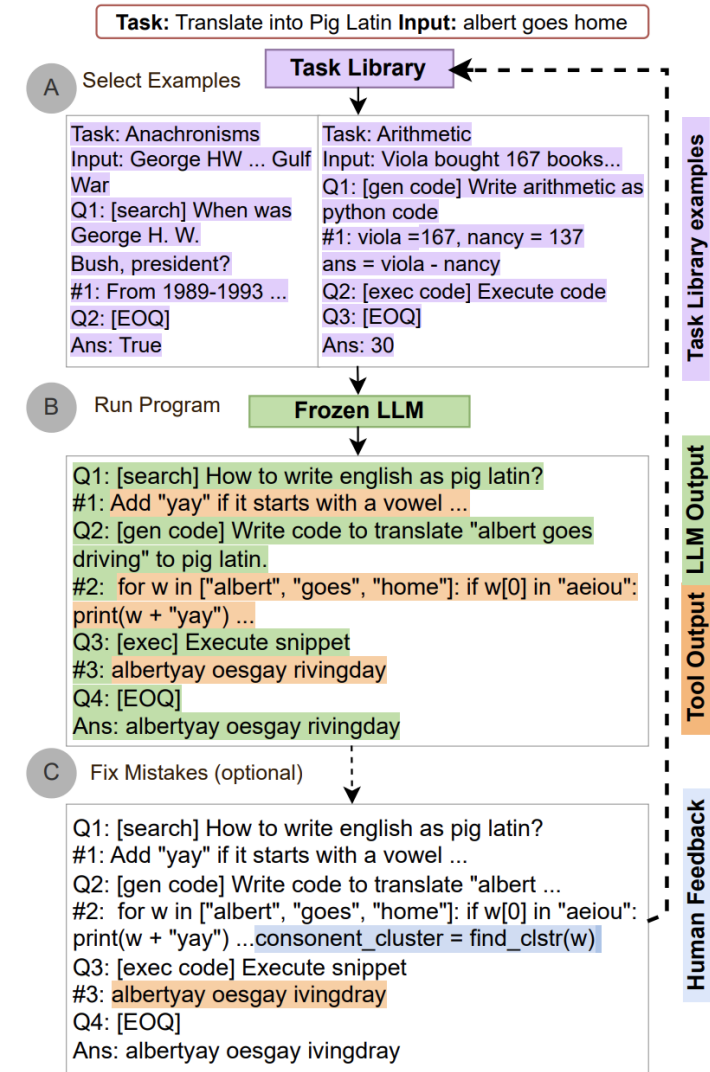
Table 4: Results for various benchmarks requiring mathematical reasoning. Toolformer makes use of the calculator tool for most examples, clearly outperforming even OPT (66B) and GPT-3 (175B).

Important

- Show the model a grammar, e.g. certain tokens which indicate tool / API usage
- Then you can choose when to call it using regular expressions

ART

- Use fixed Language model, not training
- Use task library to show similar problems
- In “few-shots of tasks” show how to use tools
- Optional human feedback: Change task or tool library



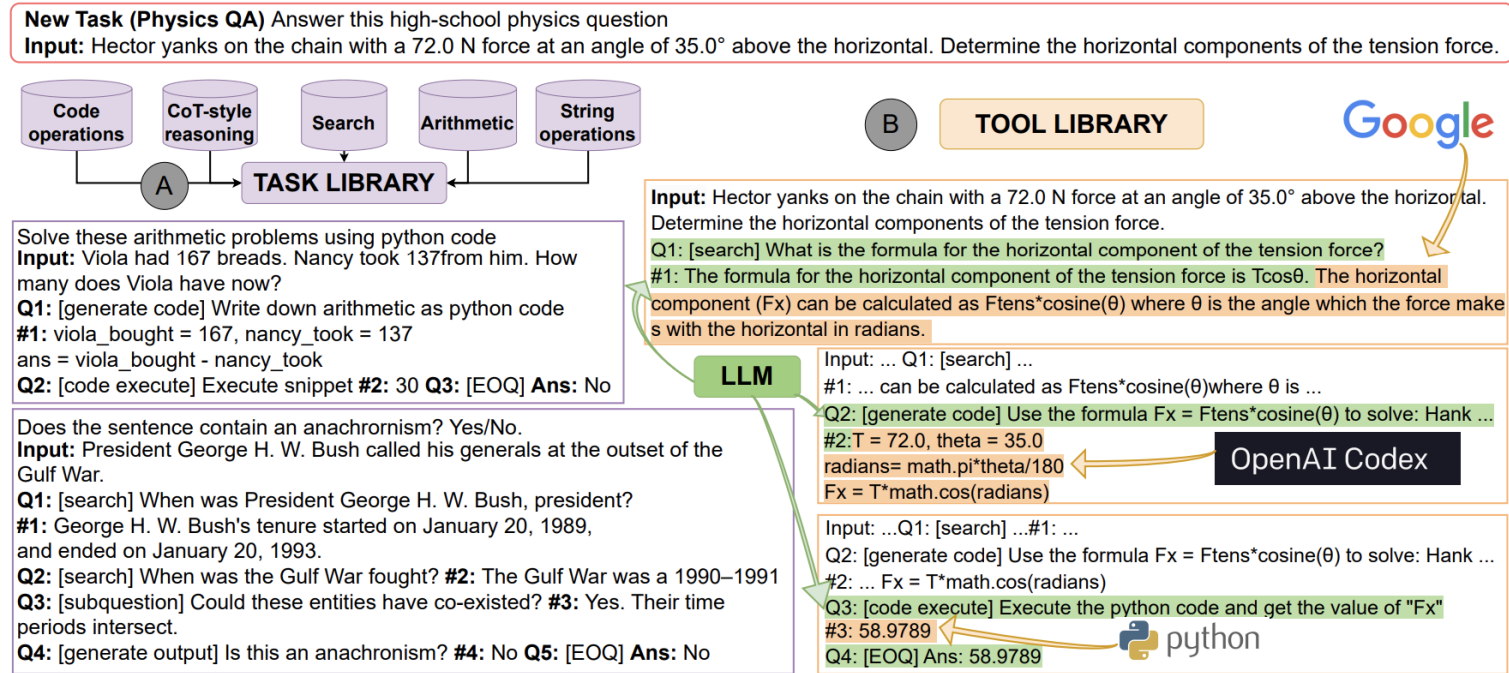


Figure 2: A run-through of ART on a new task, Physics QA. (A) Programs of related tasks like anachronisms and Math QA provide few-shot supervision to the LLM — related sub-steps and tools in these programs can be used by the LLM for cross-task generalization (shown in purple). (B) Tool use: Search is used to find the appropriate physics formula, and code generation and execution are used to substitute given values and compute the answer (shown in orange).

Numbers

Task Name (Cluster)	Few Shot	AutoCot	ART w/o Tool Use	ART	GPT-3 Best
Anachronisms (Search)	71.3 ⁵	51.48	70.87	75.66	-
Musique (Search)	2.03 ⁵	12.88	10.04	19.19	15.2 ³
Hindu Knowledge (Search)	85.02 ⁵	73.03	83.42	87.98	-
Known Unknown (Search)	68.90 ⁵	56.09	80.43	80.43	-
Δ with ART (Search)	+9.0	+17.44	+4.6		+4.0
Elementary Math QA (Arithmetic)	56.40 ⁷	74.52	58.04	68.04	-
Aqua-rat (Arithmetic)	20.54 ⁷	34.41	36.29	54.20	54.1 ⁴
GSM8K (Arithmetic)	7.79 ⁷	21.99	53.4	71.00	71.6 ⁴
Navigate (Arithmetic)	60.7 ⁷	61.7	72.4	72.4	85.90 ¹
Δ with ART (Arithmetic)	+30.0	+18.25	+11.4		-4.7
K'th letter concatenation (String)	3.2 ⁵	0.64	8.19	40.00	98.0 ²
Language games (String)	35.14 ⁵	18.58	11.19	23.08	-
Date Understanding (String)	37.53 ⁵	38.90	52.05	-	70.41 ¹
Auto Debugging (Code)	62.94 ⁵	38.24	55.29	62.94	-
Code Description (Code)	97.99 ⁷	88.67	84.67	88.00	-
Formal Fallacies (CoT)	44.84 ⁵	56.4	64.76	-	58.4 ¹
Hyperbatation (CoT)	62.72 ⁵	55.4	80.80	-	72.4 ¹
Δ with ART (Misc)	+9.6	+16.4	+13.7		-15.4
Δ with ART (Overall)	+14.90	+17.17	+7.91		-9.0

	SQuAD	T-REx	SVAMP	MAWPS	NQ	TriviaQA
GPT3 (175B)	29.90	39.8	10.0	19.8	22.6	65.9
Toolformer	33.8	53.5	29.4	44.0	17.7	48.8
ART	39.34(+5.5)	50.4(-3.1)	76.2(+46.8)	71.00(+27.0)	33.8(+16.1)	66.13(+17.33)

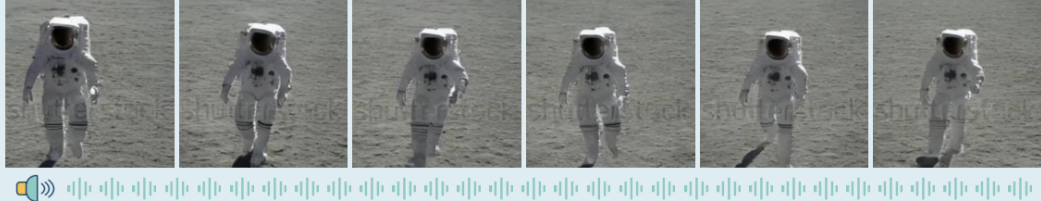
Table 4: Comparing ART results on GPT3 (175B) model and (Schick et al., 2023), which is a smaller GPT-J model finetuned for tool-use. Results are reported from their paper (their code and models are not publicly available).

HuggingGPT

- Give a pretrained models descriptions of other models and let it plan consecutive actions.
- Disclaimer: I don't fully get it

Query: generate a video with the title "An astronaut is walking in space" and dub it.

Response: Sure. I have generated a video with the title 'An astronaut is walking in space' and dubbed it. The workflow I used is as follows: First, I used the text-to-video model ' 🤖 damo-vilab/text-to-video-ms-1.7b ' to generate a video from the text. The inference result is the generated video ' 📺 /videos/5696.mp4 '. Second, I used the text-to-speech model ' 🗣️ facebook/tts_transformer-es-css10 ' to generate an audio from the text. The inference result is the generated audio ' 🎧 /audios/3f9e.flac '. Finally, I combined the video and audio to generate the dubbed video. I hope this answers your question. Is there anything else I can help you with?



Task Planning:

```
1: {"task": "text-to-video", "id": 0, "dep": [-1], "args": {"text": "An astronaut is walking in space" }}
2: {"task": "text-to-speech", "id": 1, "dep": [-1], "args": {"text": "An astronaut is walking in space" }}
```

Execution Results:

```
1. {"generated video": "/videos/5696.mp4"} 2. {"generated audio": "/audios/3f9e.flac"}
```

Query: read the image example.jpg for me. **INVOICE NO : 18301/102/T0305**

Response: Sure. I have read the image 🖼️ example.jpg for you. The inference result is 'INVOICE NO : 18301/102/T0305'. The model used for this task is 🤖 microsoft/trocr-base-printed, which is best suited for the task of image-to-text as it is fine-tuned on the SROIE dataset which is specifically designed for this task. Then I have converted the text to audio, the generated audio is 🎧 /audios/da5s.wav 🗣️ and the model used for this task is 🗣️ facebook/fastspeech2-en-ljspeech, which is a FastSpeech 2 text-to-speech model which is suitable for the task of text-to-speech. Is there anything else I can help you with?

Task Planning:

```
1: {"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "example.jpg"}}
2: {"task": "text-to-speech", "id": 1, "dep": [0], "args": {"text": "<resource-1>"}}
```

Execution Results:

```
1. {"generated text": "INVOICE NO : 18301/102/T0305"} 2. {"generated audio": "/audios/da5s.wav"}
```

Figure 4: Qualitative analysis of multi-model cooperation on video and audio modalities.

How?

Task Planning	Prompt	#1 Task Planning Stage - The AI assistant can parse user input to several tasks: <code>{{"task": task, "id": task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}}</code> . The "dep" field denotes the id of the previous task which generates a new resource that the current task relies on. A special tag <code><resource>-task_id</code> refers to the generated text image, audio and video in the dependency task with id as task_id. The task MUST be selected from the following options: <code>{{ Available Task List }}</code> . There is a logical relationship between tasks, please note their order. If the user input can't be parsed, you need to reply empty JSON. Here are several cases for your reference: <code>{{ Demonstrations }}</code> . The chat history is recorded as <code>{{ Chat Logs }}</code> . From the chat logs, you can find the path of the user-mentioned resources for your task planning.
	Demonstrations	<pre> Look at /exp1.jpg, Can you tell me how many objects in the picture? [{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "/exp1.jpg"}}, {"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "/exp1.jpg"}}] In /exp2.jpg, what's the animal and what's it doing? [{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "/exp2.jpg"}}, {"task": "image-classification", "id": 1, "dep": [-1], "args": {"image": "/exp2.jpg"}}, {"task": "object-detection", "id": 2, "dep": [-1], "args": {"image": "/exp2.jpg"}}, {"task": "visual-question-answering", "id": 3, "dep": [-1], "args": {"text": "What's the animal doing?", "image": "/exp2.jpg"}}] Given an image /exp3.jpg, first generate a hed image, then based on the hed image and a prompt: a girl is reading a book, you need to reply with a new image. [{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "/examples/boy.jpg"}}, {"task": "openpose-control", "id": 1, "dep": [-1], "args": {"image": "/examples/boy.jpg"}}, {"task": "openpose-text-to-image", "id": 2, "dep": [1], "args": {"text": "a girl is reading a book", "image": "<resource>-1"}}] </pre>
Model Selection	Prompt	#2 Model Selection Stage - Given the user request and the call command, the AI assistant helps the user to select a suitable model from a list of models to process the user request. The AI assistant merely outputs the model id of the most appropriate model. The output must be in a strict JSON format: <code>"id": "id", "reason": "your detail reason for the choice"</code> . We have a list of models for you to choose from <code>{{ Candidate Models }}</code> . Please select one model from the list.
	Candidate Models	<pre> {"model_id": model id #1, "metadata": meta-infos #1, "description": description of model #1} {"model_id": model id #2, "metadata": meta-infos #2, "description": description of model #2} [Select top-K candidate models with the same task type as the current task] {"model_id": model id #K, "metadata": meta-infos #K, "description": description of model #K} </pre>
Response Generation	Prompt	#4 Response Generation Stage - With the input and the inference results, the AI assistant needs to describe the process and results. The previous stages can be formed as - User Input: <code>{{ User Input }}</code> , Task Planning: <code>{{ Tasks }}</code> , Model Selection: <code>{{ Model Assignment }}</code> , Task Execution: <code>{{ Predictions }}</code> . You must first answer the user's request in a straightforward manner. Then describe the task process and show your analysis and model inference results to the user in the first person. If inference results contain a file path, must tell the user the complete file path. If there is nothing in the results, please tell me you can't make it.

Table 5: The details of the prompt design in HuggingGPT. In the prompts, we set some injectable slots such as `{{ Demonstrations }}` and `{{ Candidate Models }}`. These slots are uniformly replaced with the corresponding text before being fed into the LLM.

LangChain

- My goal:
 - Present a library which aims to fasten up ideas with tools or back-and-forths with LLM's (such as ChatGPT)
 - Present one idea how you could use external tools
 - Database with similarity of embedding search
 - Use ChatGPT to answer question using searched tool