IBOX: An Interactive Toolbox for Large Language Models

Stanford CS224N Custom Project

Cody Ho
Department of Computer Science
Stanford University
codyho@stanford.edu

Matias Shundi
Department of Computer Science
Stanford University
mshundi@stanford.edu

Abstract

While large language models show incredibly strong performance across many domains, they also struggle with many simple tasks that can be easily solved by other tools. We present IBOX, a large language model capable of interactively using tools through API calls and reasoning about tool use through chain of thought prompting. IBOX is a fine tuned GPT-3 model trained in a self-supervised manner on a small dataset that contains only a small handful of examples of tool use and associated explanations. We include a range of tools, including a question answering system, a Wikipedia summarizer, a date tool, and a mathematical tool that can simplify basic mathematical expressions. We evaluate IBOX on the TriviaQA, Web Questions, and GSM8K datasets, and find that it outperforms a both finetuned GPT-3 model and a noninteractive tool wielding LLM in nearly every case. We also investigate the effects of model size on IBOX, finding that LLMs need to be of a certain magnitude before they are able to use tools effectively.

1 Key Information

• Mentor: Jesse Mu
• External Collaborators: None
• Sharing project: No

2 Introduction

Motivation: The limitations of large language models (LLMs) are well known: accessing up to date information (Komeili et al., 2021), hallucination of incorrect facts (Ji et al., 2022), and a lack of mathematical skill (Patel et al., 2021). These challenges pose significantly challenges for even the most advanced LLMs. These challenges are only partially addressed by further scaling, however, they can all be solved by giving LLMs the ability to use external tools.

Shortcomings of existing approaches: Existing methods to enable LLMs to use tools either require extensive manual annotations (Komeili et al., 2021) (Thoppilan et al., 2022) or limit tool use to task-specific settings only (Gao et al., 2022). One notable exception is the Toolformer, first introduced by Schick et al. (2023) as it did not require extensive manual annotation nor task-specific settings. However, the Toolformer itself has a number of limitations, namely, that API calls cannot be chained together, tool use is noninteractive preventing refinement of queries, the Toolformer was extremely sensitive to the wording of the input, and it was difficult to recover from bad API calls (Schick et al., 2023).

Our contribution: We introduce IBOX, a large language model capable of reasoning about and interactively using tools using Chain of Thought prompting. We achieve performance significantly
greater than that of unmodified GPT-3 on a range of downstream tasks, including math and question answering benchmarks.

3 Related Work

3.1 Toolformer

First introduced by Schick et al. (2023), the Toolformer builds off previous work (Parisi et al., 2022) in teaching LLMs to use tools and serves as the primary inspiration for our work. The Toolformer is notable in that it is trained in an entirely self-supervised manner with very little manual annotation required, and the language model can choose for itself when and where it wants to use tools. However, the Toolformer has a number of limitations, namely, that API calls are generated independently, meaning that the output of one API call cannot be used as the input to another API; tool use is noninteractive preventing refinement of queries or recovery from an invalid API call; and the Toolformer was extremely sensitive to the wording of the input Schick et al. (2023).

3.2 Chain of Thought Prompting

Chain of thought prompting has been shown to be an effective method to induce reasoning in LLMs by prompting the model to follow a structured line of thought (Wei et al., 2022) Through this process, the LLM can generate a more comprehensive and logical sequence of words and phrases, which can enhance its coherence and effectiveness. Chain of thought prompting can also improve the efficiency of the model by limiting the search space and narrowing down the possible outcomes.

Our approach is motivated by the limitations of the Toolformer and the potential of chain of thought prompting. In particular, we want to make API use interactive and use chain of thought prompting to give language models further control and understanding of the tools they have been given.

4 Approach

IBOX are finetuned GPT-3 models trained to use tools accessible via various backend APIs, the implementation of which can be abstracted away from the model. All tools must only use strings for their inputs and outputs and are identifiable by a token. For finetuning, the default GPT-3 parameters were used.

4.1 Dataset Curation

**GSM8K** The GSM8K dataset is composed of question answer pairs where the answer contains not only the final solution but also the intermediate steps required for the computation and calculation annotations that include the exact mathematical expressions that should be calculated (one such annotation may be \(<\< 36 - 11 = 25 >>\)). It is straightforward to convert this notation to the API expected by IBOX using simple string manipulation. Because the answer already contains the intermediate steps, the question answer pairs with the annotations replaced by our API calls can be directly used as fine tuning data.

**Question Answering Datasets** In order to curate this dataset, GPT-3.5-turbo was given extensive examples of API usage. Then, it was told to solve problems from the sample dataset using these APIs. In short, we teach GPT-3.5-turbo to use these APIs using solely prompting. Then, the model output was filtered by removing all incorrect answers (defined as entries where none of the answer strings appeared in the result) and all responses with fewer than two API calls. This filtered data was combined with the original question to form the finetuning dataset.

We note that this approach differs from the original Toolformer training approach because our dataset contains examples of tools being used interactively. The original Toolformer dataset curation method
was to give a LLM a completed output, and then prompt the LLM to add API calls to this existing solution, however API calls were generated independently of each other, preventing chained API usage.

4.2 Chain of Thought prompting

During model execution, a domain relevant example is shown to the model demonstrating interactive API use and recovery from bad API calls. This example was created by hand and is derived from one of the questions in each of the datasets. The goal with this example, as with the chain of thought examples in the finetuning dataset, is to encourage the LLM to reason about tool use.

A chain of thought prompt for the GSM8K dataset is provided: The second tank is [MATH_API(48/2)->24] 24 gallons. Following her rule, Gail keeps [MATH_API(24/2)->12] 12 two-inch fish in the second tank. She keeps [MATH_API(48/3)->16] 16 fish in the first tank. If one fish in the first tank ate another, she would have [MATH_API(16-1)->15] 15 fish in the first tank. Thus, Gail would have [MATH_API(15-12)->3] 3 more fish in the first tank. My final answer is 3

When given the following prompt: "On a school trip to the seashore, Alan and his friends collected shells. Alan collected four times as many shells as Ben did. Ben got a late start and only collected a third of what Laurie did. If Laurie collected 36 shells how many did Alan collect?" we can compare the output of ITOOL with chain of thought:

Ben collected [MATH_API(36/3)->12] 12 shells. Alan collected four times as many shells as Ben did, so he collected [MATH_API(4*12)->48] 48 shells. My final answer is 48.

with the output

Alan collected [MATH_API(4*(36/3))->12] 48 shells. My final answer is 48.

While in this case both outputs are correct, we can observe that the model with chain of thought used two smaller API calls with the result of one chained into the other, demonstrating that it knows how to use tools interactively, while the model without chain of thought uses one larger call.

4.3 Tools

The following tools are available to our IBOX:

- **Wikipedia**: this tool returns the first paragraph of a Wikipedia article on a given topic
- **Math**: this tool can perform simple arithmetic operations, implemented using Python’s `eval()` function
- **Question Answering**: this tool can perform basic question answering, implemented using Wolfram Alpha’s question answering API
- **Date**: this tool returns the current date formatted as YYYY-MM-DD

Example inputs and outputs can be found in Table 1

Formally, each API call can be represented as a tuple \((t_i, a_i)\) where \(t_i\) is the API Identifier and \(a_i\) is the input to the API. Each API call is marked by special tokens `<API>` to indicate the start of an API call, `<API>` to indicate the end of an API call, and \(\rightarrow\) to denote where the output begins. In our model, the start token is \[\], the end token is \], and the output begin token is "->". These choices were made so that the LLM would not have to learn a custom vocabulary to query the APIs, and so that the output could be constrained to ASCII text for convenience purposes.

4.4 Tool Integration

API calls are done as follows:

1. Decoding continues until the API delimiter "->" is encountered
<table>
<thead>
<tr>
<th>API</th>
<th>API Identifier</th>
<th>Sample Input</th>
<th>Sample Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>WIKIPEDIA_API</td>
<td>The Guns of Navarone</td>
<td>The Guns of Navarone is a 1961 adventure war film directed by J. Lee Thompson by Carl Foreman, based on Alistair MacLean’s ... (full text excluded for brevity, but would be contained in the normal API call)</td>
</tr>
<tr>
<td>Math</td>
<td>MATH_API</td>
<td>24 / 7 + 5</td>
<td>8.42857142857</td>
</tr>
<tr>
<td>Question Answering</td>
<td>WOLFRAM_API</td>
<td>What is the capital of India?</td>
<td>Delhi, New Delhi, India</td>
</tr>
<tr>
<td>Date</td>
<td>DATE_API</td>
<td>None</td>
<td>2023-02-18</td>
</tr>
</tbody>
</table>

Table 1: API Example Table

2. Once the delimiter is found, decoding is halted
3. The body of the API call is extracted and the result computed
4. The return value of the API is appended to the text along with the end API token
5. Decoding continues as normal

5 Experiments

5.1 Data

We make use of three datasets, Web Questions Talmor and Berant (2018), TriviaQA Joshi et al. (2017), and GSM8k Cobbe et al. (2021) The Web Questions dataset is composed of 6642 question-answer pairs with questions derived from google search recommendations and answers derived from Freebase. TriviaQA contains over 650k question-answer-evidence pairs where evidence is a paragraph from another work which contains the answer somewhere in the text. GSM8K is composed of approximately 8.5k eighth grade level math problems along with an answer that contains the intermediate steps necessary for the final answer.

5.2 Evaluation Method

We evaluate the effectiveness of our model based on its accuracy. For the Web Questions and TriviaQA dataset, a correct answer was defined as having the answer string somewhere within the body of the LLM output (which was necessary because often the LLM would give its answer as a complete sentence). For the GSM8K dataset, the output was correct if the final number outputted by the model matched the answer from the dataset.

5.3 Experimental details

We use the default finetuning parameters for the different GPT-3 models. In order to evaluate the effect model size has on tool output, we perform finetuning on all the GPT-3 models, Ada, Babbage, Curie, and Davinci and evaluate their performance.

5.4 Baselines

We will be comparing our model, IBOX with chain of thought prompting, with four baselines:

- GPT-3 without any finetuning or additional adjustment
- GPT-3 finetuned on the particular task at hand
- IBOX with API calls disabled. We include this baseline to ensure that the tool use is actually what is improving performance, not just exposure to domain relevant data during the fine
tuning process. API calls are disabled via prompting and any API calls that were made (which never occurred in testing) returned invalid.

- IBOX with no Chain of Thought Prompting

5.5 Results

The experimental results for the Davinci GPT-3 model are summarized in [3]. We note that IBOX achieved the greatest performance out of any of the models, however in Web Questions, IBOX with a chain of thought prompt performed worse than the model without a chain of thought prompt. This is the expected result since Toolformer showed that tool usage improves model performance, and we provide a question answering API that can be used directly by IBOX for the Web Questions and TriviaQA datasets.

It is unclear why IBOX with chain of thought performed worse than IBOX without chain of thought. The difference is quite small and thus may just be stochastic effects, especially since performance is already high in general, thus it is difficult to come to clear conclusion.

<table>
<thead>
<tr>
<th>Model</th>
<th>GSM8K</th>
<th>TriviaQA</th>
<th>Web Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3</td>
<td>0.053</td>
<td>0.0713</td>
<td>0.712</td>
</tr>
<tr>
<td>Finetuned GPT-3</td>
<td>0.351</td>
<td>0.870</td>
<td>0.892</td>
</tr>
<tr>
<td>IBOX Disabled APIs</td>
<td>0.257</td>
<td>0.821</td>
<td>0.874</td>
</tr>
<tr>
<td>IBOX no CoT</td>
<td>0.568</td>
<td>0.910</td>
<td><strong>0.903</strong></td>
</tr>
<tr>
<td>IBOX</td>
<td><strong>0.643</strong></td>
<td><strong>0.923</strong></td>
<td>0.894</td>
</tr>
</tbody>
</table>

Table 2: Results summary for the largest GPT-3 model (Davinci)

6 Analysis

6.1 API Usage

API usage is summarized in the table below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GSM8K+CoT</th>
<th>TriviaQA+CoT</th>
<th>Web Questions+CoT</th>
<th>GSM8K</th>
<th>TriviaQA</th>
<th>Web Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Math</td>
<td>2.34</td>
<td>0</td>
<td>0</td>
<td>1.184</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>0</td>
<td>1.248</td>
<td>.435</td>
<td>0</td>
<td>.854</td>
<td>.197</td>
</tr>
<tr>
<td>QA</td>
<td>0</td>
<td>0.934</td>
<td>.985</td>
<td>0</td>
<td>.953</td>
<td>.975</td>
</tr>
</tbody>
</table>

Table 3: Average number of API calls by dataset for the largest IBOX model with and without chain of thought. A value of 2 means that, on average, one question from that dataset resulted in 2 API calls.

**Date** The Date API was never used, which is unsurprising since none of the tasks require knowledge of the current date.

**Math** Also unsurprisingly, the math tool was used most extensively on the GSM8K dataset. Of note however is the fact that chain of thought resulted in significantly higher tool use than without chain of thought. This is likely because the chain of thought prompt contained many examples of interactive tool use, allowing it to make better use of the Math tool. An example where this occurs is:

Prompt: Tim rides his bike back and forth to work for each of his 5 workdays. His work is 20 miles away. He also goes for a weekend bike ride
of 200 miles. If he can bike at 25 mph how much time does he spend biking a week?

Chain of Thought: Tim bikes \([MATH_API(20\times2)->40]\) 40 miles each day for work and \([MATH_API(40\times5)->200]\) 200 miles each week for work. If he also bikes 200 miles on the weekend, then he bikes \([MATH_API(200+200)->400]\) 400 miles total which takes him \([MATH_API(400/25)->16]\) 16 hours total. My final answer is 16 hours.

No Chain of Thought: Total miles: \([MATH_API(40\times5\times2+40)->440]\) 440. Total time: \([MATH_API(440/25)->17.6]\) 17.6 hours. My final answer is 17.6 hours.

**Wikipedia:** The Wikipedia API was used more extensively in the tests with chain of thought prompting in the question answering tasks. We found that the Wikipedia API was often used as a backup in the event that the question answering API failed to give a useful return value.

**Prompt:** Who directed the movie Inception?

Chain of Thought: We can use the Wolfram Alpha API to answer this \([WOLFRAM_API(Who\ directed\ the\ movie\ Inception)->INVALID_API_CALL]\). Since that didn’t work, we can try to find more information on Inception \([WIKIPEDIA_API(Inception)->Inception\ is\ a\ 2010\ science\ fiction\ action\ film\ written\ and\ directed\ by\ Christopher\ Nolan...\ (full\ text\ omitted)]\). My final answer is Christopher Nolan.

No Chain of Thought: \([WOLFRAM_API(Who\ directed\ the\ movie\ Inception?)->INVALID_API_CALL]\) My final answer is Martin Scorsese.

**Question Answering:** The use of the question answering API was mostly constant over the question answering tasks and was unused for the GSM8K dataset, which was the expected result (logically, if the task is question answering, one would expect that the first thing to try would be the tool designed for this exact purpose)

### 6.2 Error Analysis and Effect of Chain of Thought

We see that the reason the model failed on this GSM8K task is due to a logical error in the problem— it erroneously divides by 6 and multiplies by 8 when it should be multiplying by 6 and not dividing by anything. This is a logical failure on the part of the LLM, not an error in the tooling. By contrast, the reason for the failure of the TriviaQA and Web Questions is due to insufficient information in the output of the API. Because the API did not have enough information, the model simply made a guess, which in this case turned out to be incorrect.

### 6.3 Effects of Model Size

Regarding model size, our results mirror those of the original Toolformer— it appears that a certain model size is necessary before tools can be used effectively— the performance of different GPT-3 models are summarized below.

![Graphs showing performance of different GPT-3 models](image)

Smaller GPT-3 models frequently outputted invalid API calls or API calls that had very little to do with the prompt. For example, IBOX based on ada was seen asking questions on unrelated topics:

**Prompt:** What is the chemical formula for water?
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Prompt</th>
<th>Output</th>
<th>Sample Correct Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM8K</td>
<td>&quot;Artemis is making tea for a party. She knows her mom drinks an 8-ounce cup of tea and uses one ounce of tea. She will use this same ratio for the party. The party has 12 people there and each of them wants a 6-ounce cup of tea. How many ounces of tea does she need?&quot;</td>
<td>Artemis needs to use one ounce of tea for every eight ounces of water. People at the party will drink $\text{MATH_API}(12\times8/6)\rightarrow16$ ounces of water total. That means she needs $\text{MATH_API}(16/8)\rightarrow2$ ounces of tea total. My final answer is 2 ounces of tea.</td>
<td>&quot;She is making 72 ounces of water because 12 x 6 = «12*6=72»72 She needs 9 ounces of tea because 72 / 8 = «72/8=9»9 9&quot;</td>
</tr>
<tr>
<td>TriviaQA</td>
<td>&quot;Which Lloyd Webber musical premiered in the US on 10th December 1993?&quot;</td>
<td>We can use the Wolfram Alpha API to solve this [WOLFRAM_API](Which Lloyd Webber musical premiered in the US on 10th December 1993?)-»INVALID_API_CALL]. Let's try Wikipedia instead [WIKIPEDIA_API(Lloyd Webber)-»Andrew Lloyd Webber... (full text omitted)]. That does not have the answer. My final answer is The Phantom of the Opera.</td>
<td>Sunset Boulevard</td>
</tr>
<tr>
<td>Web Quest</td>
<td>what is the best sandals resort in st lucia?</td>
<td>We can use the Wolfram API to solve this [WOLFRAM_API](What is the chemical formula for air)-»INVALID_API_CALL]. My final answer is Sandals Resort.</td>
<td>Micoud Quarter</td>
</tr>
</tbody>
</table>

Table 4: Table of various errors from different datasets

Output: [WOLFRAM_API(What is the chemical formula for air)-»INVALID_API_CALL]

Furthermore, larger models tended to benefit more from chain of thought prompting than smaller models, which mirrors the result in the chain of thought paper which found that reasoning ability scales well with model size.

7 Conclusion and Future Work

In summary, we:

- Provide further support that LLMs can make use of tools
- Outline a novel training method to show LLMs to use tools that incorporates interactive tool use
- Demonstrate that LLMs can effectively reason about tools and their outputs when prompted
• Evaluate the effect of model size on the effectiveness of tool use
• Provide failure analysis surrounding when tool use failed

One major area yet to be explored is the extent to which LLMs can make use of tools they have never seen before without explicit prompting, and the complexity of tools they are able to use. While we did show that they exhibit some ability to do so from our use of gpt-3.5-turbo, it is unclear the extent this abilities goes. In addition, all the tools available to ITOOL and Toolformer were both very simple tools with limited scope– one can imagine more complex tools and whether LLMs are able to understand their use as well.

References


Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models