Homework 6

11-967

Due date: 03/02/2025 11:59 PM EST

Language models struggle to perform tasks that require complex reasoning, knowledge that they did not see during training, or interactions with the real world. There are two strategies to improve language model capabilities at these complex tasks: tool-use and retrieval augmentation. Tool-use involves training an LM to call a computer program that runs externally to the LM. Retrieval-augmentation involves conditioning a language model's generations on retrieved information. In this homework, you will train a very simple tool-use LM, where the tool is a basic calculator.

*Note: there are **two** Gradescope submission pages, one for your code and one for your `report.pdf`. Please submit your code to **Assignment** as instructed, and your `report.pdf` to **Assignment (PDF)**.*

Problem 1: Language Model with a Calculator Tool

Why do we want language models to use tools anyway? It turns out that many tasks are difficult, or fundamentally impossible for a language model to perform. One such task is basic arithmetic. In this question, you will explore how to teach a language model to use calculators in math word problems.

[Question 1.1] (***Writing, 5 points***) First, let's convince ourselves that numerical computation is indeed difficult for even the best language models. Go to OpenAI's API playground (https://platform.openai.com/playground/chat?models=gpt-4o) to experiment with inference on GPT-4o. [1] Note, we are using the playground interface because the "ChatGPT" interface already has tool-use built in. Try to come up with an instruction that:

- requires an arithmetic expression involving basic operations (e.g., +, -, *, /) to answer
- the output of GPT-4o is off from the expected value by $> 10\%$.

Feel free to use any sampling strategy that you like.

DELIVERABLES FOR Q1.1

In your report, write down the following five items:

- prompt
- model output
- result expected
- relative error rate
- choice of model (if choosing one other than GPT-4o)

[Question 1.2] (***Coding, 5 points***) We are going to use an elementary math word problem dataset, ASDiv for training our language model to use a calculator. To do this, we have pre-processed the ASDiv dataset for you into a format that allows language models to learn *when* and *how* to invoke the calculator tool. Inspect the dataset at https://huggingface.co/datasets/yimingzhang/asdiv, and examine the text and target outputs. The model will be fine-tuned to produce `target` when prompted with `text`. For example, given the prompt "`Question: Rachel bought 8 music albums online. If each album had 2 songs, how many songs did she buy total? Answer:`", the model is trained to first parse the expression to be calculated inside angular brackets ("`«8*2»`"), followed by the value of the the expression ("`16`").

[1] You will need to set up billing information with OpenAI in order to use the playground. If this is not an option for you, you may alternatively use any other frontier language model which does not have tool use incorporated (e.g. LLaMA Instruct).

During training, the model is just fine-tuned to produce the entire target (`"«8*2»16"`). You will implement an inference-time check `can_use_calculator()` that returns `true` if the angular brackets are completed (i.e., generation ends with `"»"`), which suggests that the calculator tool can be used.

DELIVERABLES FOR Q1.2

Implement `can_use_calculator()` in `src/calculator/utils.py`.

[Question 1.3] (***Coding, Writing, 8 points***) Implement `use_calculator()`, the function that calls a calculator (`safe_eval`) on partial model generation and appends calculator output back to the input. For example, on the input string `"Question: Rachel bought 8 music albums ... Answer:«8*2»"`, `use_calculator()` should return `"Question: Rachel bought 8 music albums ... Answer:«8*2»16"`. Return the input string if it does not end with a well-formed arithmetic expression.

DELIVERABLES FOR Q1.3

A. Implement `use_calculator() in src/calculator/utils.py`.

B. What if we were to use the built-in Python `eval()` function instead of `safe_eval()` for numerical evaluation? Specifically, what can go wrong when executing unsanitized language model output? Comment with at most 3 sentences.

[Question 1.4] (***Writing, 5 points***) Now you should have all you need to fine-tune a small language model (Pythia-1b) that can use a calculator. Run `python src/calculator/main.py` to train and evaluate the model (this should take no more than 10 minutes). You should now expect significant gains in accuracy when the model is provided calculator access.

DELIVERABLES FOR Q1.4

Report test accuracy on ASDiv, with and without calculator access.

[Question 1.5] (***Writing, 6 points***) Model generations for the test instances (both with and without calculator access) are dumped to `pythia-1b-asdiv/eval.jsonl`. Use this file to answer the following questions.

DELIVERABLES FOR Q1.5

A. Find a test instance where the fine-tuned model produces an incorrect answer without calculator access, and produces the correct answer with calculator access. Report the example prompt, as well as generations with and without calculator access. Include an one sentence discussion.

B. Find a test instance where the fine-tuned model produces an incorrect answer even with calculator access. Where did things go wrong? Report the example prompt, as well as generations with and without calculator access. Include an one sentence discussion.