Homework 7: Few-shot Learning in LLMs

Due November 17th at 10pm

In this homework, you will explore few-shot learning in large language models (LLMs). In few-shot learning, we don't train the model on any more data. Instead, we simply show the language model some examples in the desired format.

You should submit all of your Python files, your Pig Latin dataset, and your write-up (as a PDF) on Gradescope.

1 Using Llama 3.2 (1B)

In this assignment, you will work with a large language model called Llama 3.2 (1B), which was developed by Meta and released publicly. I am running this model on my research server.

I have given you a Python file called **query_remote_model.py**. This provides a function, **generate**, that sends a query request to Llama 3.2 (1B). Along with the prompt, you can optionally pass a maximum number of new tokens to generate. By default, this is set to 10.

2 Secret codes (40 pts)

When you were a kid, did you have a code language that you used with your friends? One common code language is pig latin. In pig latin, if the word starts with a consonant, you move it to the back of the word and append "ay". If it starts with a vowel, you simply append "ay."

Can LLaMA break this code?

2.1 Dataset creation

Write a program to generate your pig latin dataset. Your program should read a file containing a list of words and produce a file containing a corresponding list of pig latin words.

Use your program to generate a test set. **Your test set should contain 20 words**. Try to include a variety of words (low frequency, high frequency, words starting with vowels, etc.).

2.2 Few-shot learning

Next, write a program that tests how well a large language model performs on decoding pig latin. Your program should import and use the **query_remote_model.py** program that I have provided. Your task is to craft prompts to feed to the model so that it solves the pig latin decoding task.

2.3 Prompt Engineering

Start by writing a function called **make_prompt**. This function should take in a pig latin word and prepare a prompt. You should try out different formats for prompts. Try at least the following:

- Describing the task in different ways
- Adding different numbers of examples to the prompt
- Formatting the examples in different ways

2.4 Query model

Next, write a wrapper for the model query function called **query_model**. Your function should take in a prompt and call **query_remote_model.py** to get the model's prediction.

You can call the **generate** function from **query_remote_model.py** on a single prompt as follows:

```
query_remote_model.generate(prompt)
```

2.5 Evaluation

When you run a prompt through the LLM, you will need to do some additional string processing to get back an answer. Sometimes, the model will go on to produce many examples rather than just completing the single task that you set.

Write a function called **get_answer**. Your function should take in the response from the LLM. It should post-process the response and return a single answer as a string.

2.6 Running the experiment

Write a function called **run_one_prompt** that combine the functions that you have written above. It should take a pig latin word, its decoded answer, and return 1 or 0 based on whether the LLM successfully decoded the word.

Now you can finish the main function for your program. Your main function should read in the test items and labels from file. It should loop through the pig latin words, calling **run_one_prompt** on each word/label pair. It should sum up the scores returned by and print out the model's accuracy.

2.7 Analysis questions

1. How well does the LLM perform on the task?
2. Describe the different strategies you tried out for prompt engineering. What worked well? What didn't work?
3. What factors do you think affect the model's performance?

3 Commonsense reasoning (40 pts)

Many probe tasks have been proposed to evaluate the commonsense reasoning capabilities of LLMs. We will use the Choice of Plausible Alternatives (COPA) probe task from the SuperGLUE suite of LLM benchmarks to evaluate the fewshot performance of an LLM.

Here is an example from COPA (the correct choice is (b)):

1. Premise: The man broke his toe.
   Question: What was the CAUSE of this?
   (a) He got a hole in his sock.
   (b) He dropped a hammer on his foot.

I have provided you with two small subsets from COPA: train-small and val. We will test model performance on train-small, but you can take examples from val when building your prompts.

The questions are formatted as JSONL. Each example is in a JSON dictionary; the five key fields are *premise*, *choice1*, *choice2*, *question*, and *label*. You can ignore *idx*.

3.1 Reading the data

Write a function called **read_copa_data**. It should take a single argument, the name of the dataset file to be processed. It should return two lists. The first list should contain dictionaries from the training items file; the second should contain just the labels.

3.2 Crafting prompts

Next, write a function called **make_prompt**. This function should take in a COPA problem. It should prepend examples of solving the task to the target word, and return the full prompt. You should experiment with different formats for the prompt, as for the pig latin probe task.

3.3 Query model

Write a wrapper function called **query_model**. It should take a prompt, call the imported model query function, and return the result.

3.4 Evaluation

Write a function called **get_answer**. Your function should take in the response from the LLM. It should post-process the response and return a single answer as a string.

3.5 Running the experiment

Write a function called **run_one_prompt** that combine the functions that you have written above. It should take a COPA problem and return 1 or 2 to indicate which choice the LLM has selected.

Now you can finish the main function for your program. Your main function should read in the test items and labels from file. It should loop through the COPA items, calling **run_one_prompt** on each problem. It should sum up the scores returned by and print out the model's accuracy.

3.6 Analysis questions

1. How well does the LLM perform on the task?
2. What prompting formats did you experiment with? What worked well and what didn't work?
3. What factors do you think affect the model's performance?

4 Final project preparation (10 pts)

Your CS333 final project is on a topic of your choice. Many types of projects are acceptable, including model building, applying a model to a novel task, or evaluating models. Your project should involve a programming effort equivalent to a homework assignment. You will write a 6 page research paper about your project.

4.1 Project topic

Please answer the following questions:

- What is the goal of your project?
- What previous work is available on your topic?
- How will you know that you have accomplished it?
- What data do you plan to use?
- Are there other models or resources you plan to build on?
- How will you evaluate your success?

4.2 Getting access to ACCESS

For HW 8 and your final project, you will have access to GPUs hosted at the National Center for Supercomputing Applications, sponsored by the National Science Foundation,Äôs Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program.

Creating an Account

You should immediately register for an ACCESS account. To do so, go to this page and select *Register without an existing identity* on that page. Please use your Wellesley email address to register.

When you complete registration, you will be assigned an ACCESS username.

Logging in with an Account

Go to Login. Under Select an Identity Provider, select ACCESS CI (XSEDE) from the list. **DO NOT SELECT WELLESLEY.**

Creating a Delta Account

After you get an ACCESS account, we can request an NCSA Delta account on your behalf. **Add your username to this spreadsheet**. The Delta account process takes 3-4 days to complete.

4.3 Project timeline

Please sketch a timeline for completing the components of the project you have proposed.

5 Intellectual Curiosity (10pts)

As usual, you will receive 90 points for implementing everything described above. To increase your score further, you can extend your investigation in some way. **If you choose to do this, please briefly describe what you've done in your report.**

I've included a dataset in the starter code that contains posts from the AmItheAsshole forum on Reddit, labeled with NTA and YTA votes. Feel free to explore this task if you like!