Homework 8: Fine-tuning and Proxy-tuning

Due November 24th

In this assignment, you will work with a dataset of math word problems called GSM8K. An example item is shown below:

Question: *James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?*

Answer: *He writes each friend 3*2=«3*2=6»6 pages a week*
*So he writes 6*2=«6*2=12»12 pages every week*
*That means he writes 12*52=«12*52=624»624 pages a year*
624

Note that the Answer column in the dataset contains some reasoning, followed by 4 hash signs, followed by the numerical answer to the problem.

You will work in three separate Jupyter notebooks for this assignment, though you will reuse code between files.

1 Evaluation (25pts)

For this section, work in the **evaluate.ipynb** notebook. You can do this portion in Google Colab.

Your first task is to write some functions to evaluate large language models on this dataset.

1.1 Running models locally and remotely

I have given you functions to send queries to various models:

- Smol LM2 (135M): **query_local_model**
- Llama 3.2 (1B): **query_remote_model**
- Llama 3.1 (8B): **query_remote_model**

To use the remote models, you will need to upload the **logit_client.py** file into your Colab file system. This program is required to connect to the server where the remote models are running.

1.2 Running a model on the dataset

Write a function called **run_evaluation**. It should take five arguments: a dataset, a **query function** (one of the functions listed above), a model, a tokenizer, and the model's name.

Your function should do the following:

- Iterate through the problems in the dataset

- For each item, call **make_prompt** to construct a prompt
- Call the query function to prompt the model
- Return a list of the model's responses

1.3 Extracting answers

If you print out some of the responses that you get from the model, you will see that they often have more information than just the numeric answer.

Write a function called **extract_answer** that returns the last numeric expression in the generated response.

If your function cannot find a number in the response, it should return None instead.

1.4 Evaluating a model

Next, modify your **run_evaluation** function so that it calls **extract_answer** on the model's response.

If the extracted answer is None, append the response to a list of errors. If the extracted answer is a number, compare it to the correct answer for the item.

Your function should **print the model's name and accuracy** and **return the list of errors**.

1.5 Questions

1. How well does the Smol model perform?
2. How well does the Llama 3.2 (1B) model perform?
3. How well does the Llama 3.1 (8B) model perform?
4. What kinds of errors do you see when you print out the list of non-numeric responses?

2 Fine-tuning (25pts)

In this part of the assignment, you will experiment with fine-tuning a small model on the GSM8K dataset. Work in the **finetune.ipynb** notebook.

Warning: finetuning this (very small) model on our entire dataset will take about an hour on a single GPU. I recommend that you use a subset of the dataset while writing/testing your code.

2.1 Using the NCSA Delta Cluster

This part of the assignment should be done on the NCSA Delta cloud computing system, because it requires GPU resources.

Delta has a JupyterLab interface. This is the easiest way to use the cluster. You can start a Jupyter session by following this link: https://openondemand.delta.ncsa.illinois.edu/pun/sys/dashboard and logging in with your Delta account.

When you create a Jupyter notebook, make sure you select:

- bftp-delta-gpu as the *Name of account*
- A *Partition* with GPUs. I recommend gpuA40x4
- A *Duration of job* that seems reasonable. E.g., 1-2 hours.
- Sufficient CPU (e.g., 8) and RAM (48GB). The more CPU/RAM you select, the longer you may need to wait for the job to start.
- 1 GPU
- Leave the *Working Directory* blank; it will default to /scratch/bftp/<USERNAME>

It may take a couple of minutes for your request to become active.

2.2 Fine-tuning Your Model

I have given you a function to tokenize the datasets: **tokenize_dataset**. This function takes a **mode** parameter that controls whether the dataset is processed for testing or training. Read over this code so that you understand the format of the data.

There is code in **main** to construct DataLoaders for the training and test datasets.

I have also given you a function called **make_or_restore_model**, which either loads the model or the most recent fine-tuning checkpoint. It returns the model, optimizer, and current epoch.

This means that you will be able to restart finetuning if you get disconnected in the middle. However, if you want to train from scratch, you will need to delete the contents of your checkpoint directory.

Your task is to write 1) a training function, **train**, to run one epoch of training; and 2) to write a training loop in main.

2.3 Train function

Your training function will need to do the following on each batch:

- Move the input ids and attention mask to device
- Call model.forward on the items
- Call zero_grad() on the optimizer to zero out the gradients
- Call backward() on the loss for the batch
- Call step() on the optimizer
- Add the batch loss to the total loss for the epoch

Your training function should print out the average loss per batch at the end of the epoch.

2.4 Training loop

Your training loop should train the Smol model on the training data for 4 epochs and evaluate its performance on the test data after each epoch using the **run_evaluation** function from Part 1.

2.5 Questions

1. How well does the Smol model before training?
2. How well does the Smol model do after 4 epochs of training?

3 Proxy-tuning (25pts)

For this section, work in the **proxytune.ipynb** notebook. You can do this portion in Google Colab.

You will work entirely with remotely served models, since the Smol LM models are not performant enough to show gains from proxy-tuning.

Reminder: To use the remote models, you will need to upload the **logit_client.py** file into your Colab file system.

3.1 Logit shift

You will need to implement the core proxy forward function. This function should take the names of the three models and the input_ids for the problem, and predict the next token according to the proxy-tuned model.

Your function will need to do the following steps:

- Retrieve logits from each of the three models using the client.get_logits function
- Compute the difference between the expert and antiexpert logits
- Apply the difference to the big model logits
- Softmax over the logits to get a probability distribution
- Return the token with the highest probability in the steered distribution

3.2 Questions

1. How well does the proxy-tuned model do?
2. Why do you think this dataset is so challenging for language models?

4 Final project preparation (15pts)

4.1 Project update

- What progress have you made on your final project?
- Have your plans changed or evolved since last week?

4.2 Literature review

Please identify three academic research papers that provide relevant background for your proposed project. Read them and provide a paragraph summary of each.

Two good places to look for NLP papers are Semantic Scholar and the ACL Anthology.

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.**

5.1 Wrapping Up

When you're finished, you should submit your three notebook files and your answers to the questions (as a PDF) to Gradescope.