CS412 Machine Learning, Spring 2024: Homework 4

May 5, 2024

Instructions

- Please submit all of your files in a zip file. Name your submission as CS412-HW4-YourGroupName.zip, replacing 'YourGroupName' with your project group name. One submission per group is enough.
- Zip file must include 3 files. CS412_HW4_Step1_Data.csv, CS412_HW4_Step2_Predictions.csv and CS412_HW4_Finetuning.ipynb (don't delete outputs in the notebook.)
- The responsible TA for this homework is Mert Pekey. But first, ask your questions on the Homework Forum.
- Please avoid using any Generative AI tools, like ChatGPT, for this homework, as it can be easily detected.
- If you are submitting the homework late (see the late submission policy in the syllabus), we will grade your homework based on the time stamp of submission and your remaining late days.

1 Modifying Instruction-Response Data

Annotating chat data is crucial for developing a high-performing large language model. In this homework, you will first annotate and correct the chat data (instruction—response data)(Section 1) and use your annotations to fine-tune the model (Section 2).

For the first part of the homework, we'll share a Google Sheet file with pairs of instructions and responses. First, copy this Google Sheet to your drive so that you can edit it. In this dataset, responses to the instructions have a lot of errors. Your task is to modify the responses in this data to improve the response and better fit the instructions. In addition, for each response, you need to select the type of error from the dropdown list. You need to follow the Chat Style Guide to do this. At the end of Section 1, you will download this updated data as a CSV file containing four columns: 'id', 'instruction', 'output', and 'fail_mode'. Save this file as **CS412_HW4_Step1_Data.csv**. You'll use it for Section 2 and also submit it with your homework.

You can access the dataset by clicking Google Sheet Section 1 Dataset

1.1 Chat Style Guide

To make models communicate with us human-like, we follow a couple of fine-tuning and alignment approaches, the most common of which is supervised fine-tuning. This approach includes training on a bunch of data points formatted in a conversational manner. These data points come in pairs of prompts and completions. A prompt is an instruction given to the model. It could vary from \What is the capital of France?" to \Give me an outline for the presentation I am about to give about unicorns." A completion is the model's response (or the response we want the model to give).

During training, we feed the model the prompts and expect it to predict the completion or at least something similar. Then, we penalize the model if it steers too much away from the expected distribution of completions. Thus, we teach the model how we want it to communicate through this process. The quality of completion data is crucial for the quality of its output. For the question about the capital of France, the completion could be as simple as `paris` or it could be more proper like `The capital of France is Paris.` or it could be an entire paragraph about Paris. Although it is somewhat subjective, we can all agree that we do not want the model to respond so succinctly and without following the written rules of the language. So, the first completion can be regarded as bad.

Almost everyone can agree on certain aspects of annotating chat data. Understanding and applying these aspects to the training data is crucial for developing a high-performing large language model. We will go over those aspects in this document. The rule of thumb is always to think of yourself as a user of the model and try to assess whether you would be satisfied with the completion you received.

1.1.1 Criteria For Assessing Prompt Quality

1. Being able to capture the distribution in the wild: Prompts come in all shapes and sizes. Some people make typos and grammatical errors, whereas others use proper language. The model should not be thrown off by either; discard the style of the prompt and follow the instructions only. To achieve this, there should be a variety of prompts in the training data with different styles and tones.

2. Complexity: The model's pre-training data may contain a lot of mundane information since it is trained by scraping tons of information on the web. Therefore, it is helpful to teach the model about interesting topics at this stage. A conversation about biomechanics may be more engaging and useful to the model than the capital of France prompt.

Another way to think about complexity is the controls asked in the prompt. `Give me an itinerary to Hong Kong in JSON format` is a more valuable data point to train on than a prompt asking for the itinerary.

1.1.2 Criteria For Assessing Completion Quality

1. Factual correctness: Language models suffer heavily from hallucination, meaning they sometimes make stuff up, especially if the prompt is out of distribution. If we feed the model inaccurate information at any step, we increase the chances of it hallucinating and spreading misinformation. We should never let our model get trained on a data point that claims that the capital of France is Ankara. Note that we might sometimes want the model to give creative outputs, and we might prompt it with Tell me a story where Putin is the president of Turkey. In this case, factual correctness does not apply because there is no factuality in fiction.
2. Perfect grammar, punctuation, and spelling: The model should produce outputs that perfectly adhere to the grammatical rules of the language it is communicating in.
3. Tone: The model's tone should be professional, respectful, and helpful.
4. (Right amount of) Chattiness: Responding with a few words to every prompt would make the model look unhelpful and unprofessional. Responding with a large block of text to a simple question would be unhelpful for the user since they would be forced to search for the information in the text block the model provided. When rating completions, always try to put yourself in the shoes of the person who asked the question to the model and assess if you would be satisfied with the response.
5. Prompt compliance: The completion should follow the ask in the prompt. Even though we specified many criteria that make the completion good above, there is one rule that trumps it all: The customer is always right. If the prompt wants the model to output in the style of a 14-year-old girl from the valley, the completion should adhere to that, and we should reconsider (2) and (3) above. Or if the prompt asks the output to be as short as possible, we should discard (4) and penalize longer completions.
6. Non-repetitiveness: Repetition hurts model performance since it teaches the model to repeat itself, which can result in the model being stuck in a loop. Avoid repetition as much as possible.
7. No plagiarism: When training models, especially for profit, it is really important not to copy and paste information directly from websites since that might result in copyright violations and lawsuits.

2 Fine-tuning

After modifying prompt responses, leveraging this data to enhance the performance of a Large Language Model (LLM), specifically LLama2, in generating responses is the next step of the Assignment. Although LLama2 is trained on vast amounts of data, its proficiency in responding to specific instructions is lacking. To improve LLama2's responsiveness, we'll use a process called fine-tuning with instruction-response pairs.

The LLama2 model has 7 billion parameters, requiring substantial computational resources. However, we will employ a method called QLora to mitigate these requirements, enabling fine-tuning on the freely available T4 GPU provided by Google Colab.

A fine-tuning notebook has already been prepared for this purpose. Your task is to provide the data you prepared in Section 1 and execute the provided cells. You can access the notebook by clicking Fine-tuning Notebook

Please follow the steps below to run the code without encountering any errors:

1. Make a copy of the notebook to your Google Drive by selecting `File -> Save a copy in Drive`.
2. Verify that the notebook is configured to utilize the T4 GPU by navigating to `Runtime -> Change Runtime Type`.
3. Upload your CSV file prepared in Section 1 to Colab. The filename must be **CS412_HW4_Step1_Data.csv**
4. Download the Test Data as CSV file from the Google Sheet Test Data and upload it (filename must be **CS412_HW4_Test_Data.csv**) to Colab.
5. The fine-tuning process may take about 10 minutes.
6. Before starting the Inference Section, restart the notebook to avoid memory errors. You can select `Runtime -> Restart Session`. Restarting won't delete your uploaded data or saved models. Avoid deleting the runtime, as it will delete all files; restart it instead.
7. Exporting predictions to a CSV file is done automatically at the end of the notebook and may take about 10 minutes. Upon completion, download the CSV file (You may need to refresh the folder sidebar on the left side of Colab) and upload it to SuCourse accordingly. The filename should be **CS412_HW4_Step2_Predictions.csv**.