LLMs Assignment 2

Release date: 10/29/2025

Due date: 11/12/2025 at 5:00 PM

As everyone is expected to share the same computing resources, it is advised **not** to wait until the last couple days to work on the assignment. This assignment will require fine-tuning a language model using LoRA to perform better on a task of interest.

1. Prepare your dataset and language model

1. To fine-tune your model, you need many examples of **question-answer pairs**
    - The questions are what you would input to a language model and the answers are the output responses that you would expect to get back
2. Some examples of things you could fine-tune (this is not an exhaustive list, be creative!)
    - Article Summarization
        - Example Question: Summarize this piece of text: {text}
        - Example Answer: Summary generated either by a human or a very large, capable LLM
    - Tone and writing style of outputs
        - Example Question: any question that you are interested in
        - Example Answer: Answers generated either by humans or large LLM that represents the style you desire
    - Reasoning and problem-solving ability
        - Example Question: math or coding problems
        - Example Answer: Answers generated by large LLM or humans that detail how the problem was solved, step-by-step
    - News article category classification
    - Social media sentiment analysis
3. The number of examples necessary will depend on the size of the model being tuned and the difficulty of task that is being tuned for
    - **You should try to obtain several hundreds of examples (question/answer pairs), possibly into the thousands if feasible**
4. Coming up with these question-answer pairs is time-consuming if a human generates the answers. For this reason, it is allowed and encouraged to use a large, capable LLM to generate answers to your questions to use in fine-tuning.

    - The model you use to build the answers should be much larger than the model you fine tune
    - Using Ollama on your local computer is one approach to doing this
5. If you generated your own data, please upload it to a cloud storage service such as Google Drive and create a shareable link with view/download permissions. Include this link in your README so we can access it when grading.
6. You will also want to set aside some questions to use for evaluating the base model and the fine-tuned model. At least 10-15 questions should be used for qualitatively assessing the effect of fine-tuning
7. Find a language model on HuggingFace that is 1.7B parameters or less. Here are some examples, but there are many more that exist
    - https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B-Instruct
    - https://huggingface.co/HuggingFaceTB/SmolLM2-360M-Instruct
    - https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct

2. Show sample outputs with the base language model

1. Using the questions you set aside, input these questions into your base language model from HuggingFace and observe the outputs
    - Preferably, the model should perform sub-optimally on some of the questions, since we are trying to demonstrate how fine tuning can improve base performance
        - Switching to smaller models may result in sub-optimal answers for the base model which you can improve through fine tuning

3. LoRA fine-tuning with PEFT

1. Use the PEFT library for LoRA finetuning which is developed and maintained by HuggingFace
    - https://github.com/huggingface/peft
    - https://huggingface.co/docs/peft/en/index
2. You should use the GPUs on DeepDish. Make sure you run your jobs on GPU cards and not CPU. You **will crash** the servers if you run them on CPUs.
3. Recommended steps are as follows.
    - Load your original pretrained language model
    - Initialize the PEFT config with the LoRA parameters
        - This includes the rank of the low-rank adaptation, which modules/layers to fine-tune, dropout, etc.
    - Use the transformers library Trainer() class to simplify training of the PEFT model

        - You will need to set various training parameters here such as batch size, learning rate, etc
    - Save your fine-tuned model locally or by pushing it to the HuggingFace model hub which requires a HuggingFace api key

4. Show sample outputs with fine-tuned model

1. Repeat the process from step 2 but with your fine-tuned model. Compare the outputs to see if your fine-tuned model performs better. Ensure the exact test questions were not used in fine-tuning your model
    - You can qualitatively inspect the outputs or use a quantitative measure if you want a more rigorous comparison
        - See some example metrics at https://huggingface.co/evaluate-metric

5. Create a Unit Test and Submit assignment

1. Create a short standalone "unit test" script that tests your fine-tuning function on a very small subset of data and for just a couple of iterations. This will allow us to make sure your fine-tuning code works without having to run your entire code. This code should be able to run by itself in around 5-10 minutes or less. This could be a separate .py file or .ipynb file, but please include how to run this short unit test in your README.
2. Submit the final project on Github. Include all the code files (this includes any data processing files – if you used that code for your project, we need to see it), a README on how to run the code (including where to download data and how to run the unit test), the Dockerfile and Docker image name/server location necessary to run the code, and a document showing sample outputs from your code. If you generated your own data, please upload it to a cloud storage service such as Google Drive and create a shareable link with view/download permissions. Include this link in your README so we can access it when grading.
    - The sample outputs should show some sample questions/outputs from the base model and your fine-tuned model
    - In your sample output document, include some discussion on how the fine-tuned model outputs differ from the base model
    - Include in your README how to run the short unit test

Grading

Sample Outputs Comparison (35 points):

- Sample outputs from both the base model and your fine-tuned model. (25 points)

- Ensure that the exact test questions were not used during fine-tuning; points will be deducted if this is not followed. (10 points)

Unit Test Script (40 points):

- Standalone unit test script as described in your README. (5 points)
- The script should run on a small data subset and complete in 5–10 minutes or less. (35 points)

Discussion of Model Outputs (25 points):

  - Discussion on differences between base and fine-tuned models. (15 points)
  - Clarity and readability of README. (10 points)