Homework 5 Coding: Finetuning LLMs with SFT

Due: Friday, December 12th at 11:59 pm

Deliverables. You will be reading related works on finetuning models and then apply those techniques to finetune a model to do well on previous 189 exam problem while maintaining its general knowledge capabilities.

This homework includes a walkthrough Colab notebook, `cs189_mcq_sft_walkthrough.ipynb`, which you may use as a starter template. You will fine-tune a small instruction-tuned model using Supervised Fine-Tuning (SFT) and submit predictions to the Kaggle mini-competition:

`https://www.kaggle.com/t/11c8ffdc967fe3f27755cde6fb5810e8`

You must submit the following:

- **Paper Questions [PDF]**
- **Code [Notebook + PDF]:** Include your notebook, any other coding files, and the PDF export of your notebook into a single ZIP file. For the notebook PDF, make sure all code cells, outputs, and any plots are visible. Submit this zip to Gradescope under *HW 5 Coding*. The code in your zip must match the code in your PDF and reproduce the same outputs if re-run by course staff.
- **Writeup [PDF]:** After you have trained your model, write a short (1-2 pages max) write-up answering the following questions:
  1. What techniques did you use to train your model? Include a detailed explanation of your final methods, including your loss curves, hyperparameters, datasets, optimizers, etc. Make sure to explain what each dataset contains, not just its name and cite where you got it from. If you took tecniques from existing works, please cite those as well. *(3-6 sentences)*
  2. What did you try that didn't work, and why do you think it failed? *(1-2 sentences)*
  3. Provide 2 different examples of catastrophic forgetting (something the model could do before but can't do now). Include the input, the outputs from the base model and your finetuned model, and one sentence on why you think the model has degraded for this input. Note that you don't need to show examples from the provided datasets; you can use any type of input. *(2-3 sentences)*
  4. Provide 2 different examples of non-CS189 problems that your finetuned model gets right and your base model gets wrong (i.e. what unintended skills did your model gain). Include the input, the outputs from the base model and your finetuned model, and one sentence on why you suspect the model has gained this ability. *(2-3 sentences)*
- **Kaggle Submission:** Submit your predictions to the course Kaggle competition for this homework. You must also include:
  - Your **Kaggle username**, and
  - A **small screenshot of your Kaggle score** in your PDF submission.

Kaggle Grading Thresholds. Your Kaggle leaderboard accuracy is mapped to points as:

- **0.40 or above:** full (5) points
- **0.39:** 2 points
- **0.38:** 1 point

Assignment Overview

In this homework, you will put together an end-to-end **Supervised Fine-Tuning (SFT)** pipeline for a multiple-choice question-answering task. You will:

- Use the provided MCQ dataset, which mixes different question sources.
- Fine-tune a small instruction-following model (e.g. `Qwen/Qwen2.5-0.5B-Instruct`) using the `transformers` and `trl` libraries.
- Evaluate how SFT changes the model's behavior compared to the unfine-tuned baseline.
- Generate predictions for the hidden test set and submit them to Kaggle.

This assignment is designed to connect the general concepts from **paper questions** about SFT data composition with a practical implementation where you control the SFT data and observe its effects.

Key Learning Objectives

By the end of this homework, you should be able to:

- Understand how supervised fine-tuning changes a model's behavior on a specific task.
- Prepare MCQ-style datasets for SFT (prompt + answer format).
- Build an SFT pipeline using `AutoModelForCausalLM`, `AutoTokenizer`, and `SFTTrainer`.
- Evaluate task-specific performance (accuracy) for both baseline and fine-tuned models.

Description of the Mini-Competition

In this mini-competition, you will fine-tune a model to answer multiple-choice questions (MCQs) with five options per question: **A, B, C, D, E**.

Data Files

The Kaggle competition provides the following files:

- `test.csv`: unlabeled MCQs; you must predict an answer for each row.
- `sample_submission.csv`: example of the correct submission format.

Each row in your training data for SFT would ideally contain:

- A question prompt (possibly multi-sentence),
- Up to five answer choices,
- The correct answer label (one of `A`, `B`, `C`, `D`, `E`).

Your model must learn, from these examples, to predict the correct answer for each example in `test.csv`.

Objective

Your task is to:

1. Use the provided notebook to fine-tune the base model on `train.csv`.
2. Adapt the same pipeline to run inference on `test.csv`.
3. For each row in `test.csv`, output exactly one letter from the set $\{A, B, C, D, E\}$.
4. Save these predictions in the **strict submission format** described below and upload to Kaggle.

Evaluation

Metric

Submissions are evaluated using **accuracy**: the fraction of test examples for which your predicted answer matches the hidden correct answer. If we let N be the number of test examples, and $\widehat{y}_i$ and y_i be the predicted and true answers for example i, then

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\widehat{y}_i = y_i].$$

Example

id	True Answer	Your Prediction	Correct?
`test_00001`	A	A	Yes
`test_00002`	A	B	No
`test_00003`	A	A	Yes

Here, the accuracy would be $2/3 \approx 66.7\%$.

The leaderboard is split into **public** (50% of test data) and **private** (remaining 50%) subsets. Final rankings and grading are based on the private leaderboard.

Submission File Format (Strict)

For each `id` in `test.csv`, you must predict the most likely answer among $\{A, B, C, D, E\}$.

Columns and Header

Your submission file must:

- Be a CSV file.
- Contain **exactly two columns**, in this order:
  1. `id`
  2. `prediction`
- Include a **single header row**:

```
id,prediction
```

Example Submission

A valid submission might look like:

```
id,prediction
test_00001,A
test_00002,A
test_00003,C
test_00004,E
...
```

Important formatting rules:

- Every `id` from `test.csv` must appear *exactly once.*
- `prediction` must be a single uppercase letter from `A,B,C,D,E`.
- Do not include extra columns, spaces, or quotes around entries.
- Submissions with missing `id`s, extra `id`s, or invalid predictions will receive a score of **0.0**.

We strongly recommend you start from `sample_submission.csv`, fill in the `prediction` column using your model, and then save to a new CSV for upload.

Connecting to the Notebook

The provided Colab notebook `cs189_mcq_sft_walkthrough.ipynb` includes:

- Configuration for the base model: `MODEL_NAME = "Qwen/Qwen2.5-0.5B-Instruct"`.
- Loading the MCQ dataset from a CSV file.
- Construction of instruction-style prompts from each MCQ.

- Supervised fine-tuning using `SFTTrainer`.
- Evaluation code to compare baseline vs. fine-tuned accuracy.

Feel free to adapt the notebook or modify the training dataset for the Kaggle competition **(BUT you must use the same base model and MAY NOT change the model)**.

1. Train your model on any allowed extra data.
2. Make predictions with your model after SFT on `test.csv` and record the model's predicted letter (`A-E`) for each row.
3. Construct a DataFrame with the two required columns `id` and `prediction` and save it as a CSV in the required format.

Submission Checklist

Before submitting, make sure you have:

- Saved your Colab notebook as a **PDF**, with all code and outputs visible.
- Generated a **zip file** of your notebook code and notebook PDF.
- Submitted your PDF writeup
- Created a Kaggle submission file with columns `id,prediction` in the exact format.
- Submitted your CSV to the Kaggle competition and noted your score.
- Included your **Kaggle username** and a **screenshot of your leaderboard score** in your PDF.
- Included an **acknowledgement of any GenAI** you used, if applicable (see syllabus).

Good luck, and have fun exploring how SFT changes your model's behavior on real MCQs!