CS 4/565: Introduction to AI
Project 4: Fine-Tuning Lightweight Language Models on Classification Tasks: A Case Study with the Mushroom Dataset

Due: May 7th, 11:59 PM

1. Introduction

This project explores how to transform classification problems—commonly handled by traditional machine learning—into instruction-following tasks suitable for fine-tuning lightweight **Large Language Models (LLMs)**.

The goal is to fine-tune a small LLM (e.g., TinyLlama or DistilBERT) to predict whether a mushroom is **edible** or **poisonous**, based on its natural language description derived from structured features (like color, shape, and odor).

Key Terms:

- **LLM (Large Language Model)**: A model trained to understand and generate natural language (e.g., GPT, TinyLlama, DistilBERT).
- **Lightweight LLM**: Models with fewer than 3 billion parameters, faster and more memory-efficient, suitable for Colab Free.
- **Prompt**: A natural language input that instructs the model.
- **LoRA (Low-Rank Adaptation)**: A parameter-efficient fine-tuning technique.
- **Transformers** – Load and fine-tune pretrained language models like TinyLlama and DistilBERT.
- **PEFT (Parameter-Efficient Fine-Tuning)** – Apply LoRA for efficient adaptation of large models.
- **BitsAndBytes** – Enable 4-bit quantized model loading to reduce GPU memory usage.
- **Datasets** – Load and manipulate datasets in JSONL, CSV, or other formats compatible with Hugging Face.
- **Alpaca-style Prompt Format** – Instruction-tuning format with `instruction`, `input`, and `output` fields.

2. Tasks

You are required to fine-tune at least two models to achieve the classification problems using mushroom dataset.

Step 1: Input – Data Format

The dataset contains approximately 8124 samples of mushrooms, each described using 22 categorical features such as cap shape, odor, gill attachment, and habitat. Each instance is labeled as either `e` (edible) or `p` (poisonous). The task is to convert each row into a natural language prompt that describes the mushroom's attributes in human-readable terms. These prompts are then formatted into Alpaca-style JSONL records, ready for use in instruction tuning.

Step 2: Model Selection and Constraints

You are required to select at least two models from a curated list to perform the fine-tuning task described above. These models will be used to learn the mapping between natural language descriptions of mushrooms and their classification labels (edible or poisonous).

Recommended models include TinyLlama-1.1B-Chat, which is particularly effective for instruction-following and supports LoRA integration, and DistilBERT-base-uncased, a smaller and faster model that is well-suited for classification tasks. GPT2-small may also be included as a baseline to compare generation-style models. Each selected model should be used in both the training and inference phases: during training, the model is fine-tuned using the prepared prompt-style data with LoRA; during inference, the model is evaluated on unseen prompt examples to test its generalization and classification ability.

Due to platform constraints on Colab Free (approximately 12GB GPU RAM on T4 or A100 GPUs), full model fine-tuning is not feasible. Thus, models can be loaded in 4-bit quantized format and trained using LoRA with a maximum batch size of 8. The expected outcome from this step is two fine-tuned models that can be directly compared on performance metrics such as accuracy and loss, using consistent prompts. The training configuration includes 3-6 epochs of training with a small batch size (between 4 and 8) to remain within Colab GPU limits. The optimizer used is AdamW, with a learning rate set to 2e-5. Input data must be formatted as Alpaca-style '.jsonl' prompts. This setup ensures memory efficiency while still allowing meaningful model adaptation.

Step 3: Evaluation and Comparison

After fine-tuning, you will evaluate model performance based on several criteria. Accuracy will be computed to measure how often the model correctly predicts the label (edible or poisonous) from the mushroom description. A training loss curve will be plotted to visualize model convergence over epochs.

1 Deliverables

There are two deliverables: report and code.

1. **Report (25 points)** The report should be delivered as a separate PDF file, and it is recommended for you to use the NIPS template to structure your report. You may include comments in the Jupyter Notebook, however, you will need to duplicate the results in the report. The report should describe the background, methodology, experimental results, and conclusions.

2. **Code (75 points)**

   The code for your implementation should be in Python only. The name of the Main file should be project3.ipynb. Please provide necessary comments in the code and show some essential steps for your group work.