SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

50.040 Natural Language Processing, Fall 2025

Homework 3

Due 30 November 2025, 23:59pm

Homework 3 will be graded by Yuhao Wu
[wu_yuhao@mymail.sutd.edu.sg]

This homework focuses on the practical implementation of sentiment analysis using modern large language models. You will explore three different approaches to fine-tuning: zero-shot inference with a base model, full-parameter supervised fine-tuning (SFT), and parameter-efficient fine-tuning using LoRA (Low-Rank Adaptation). Through this assignment, you will gain hands-on experience with state-of-the-art language models and understand the trade-offs between different fine-tuning strategies in terms of computational efficiency, parameter efficiency, and model performance.

Project Overview

This homework implements a complete sentiment analysis pipeline for IMDB movie reviews using:

- **Model**: Qwen/Qwen3-0.6B (600M parameters)
- **Dataset**: stanfordnlp/imdb (50,000 movie reviews)
- **Task**: Binary sentiment classification (positive/negative)

Methodology

You will compare three approaches:

1. **Base Model**: Zero-shot performance without fine-tuning
2. **Full-Parameter SFT**: Fine-tuning all model parameters
3. **LoRA-SFT**: Parameter-efficient fine-tuning using LoRA

Important Notes:

- If you don't have a GPU, we recommend using Kaggle or Colab resources
- Estimated GPU processing time: <2 hours
- Kaggle: `https://www.kaggle.com/`
- Colab: `https://colab.research.google.com/`
- **All code must run completely for full credit (14 points)**

1 Data Processing and Preparation

Question 1.1 [written] (4 points)

Why do we need to reformat the dataset instead of using the original one directly? Explain the purpose of converting the raw IMDB reviews into an instruction-following format.

Question 1.2 [written] (4 points)

How can we train LLM if we don't perform this conversion? Discuss the implications of using raw data versus formatted instruction data for language model training.

Question 1.3 [code] (7 points)

Complete the data splitting and sampling implementation in the notebook:

- Split the training data into train and validation sets
- Perform random sampling to limit dataset size
- Ensure balanced class distribution

2 Dataset Class Implementation

Question 2.1 [code] (2 points)

Implement the input combination logic in the `InstructionDataset` class. Combine the tokenized prompt and response to create the full input sequence.

Question 2.2 [code] (2 points)

Create the labels for training. The labels should:

- Set prompt tokens to -100 (ignored in loss calculation)
- Set response tokens to their actual token IDs

Question 2.3 [code] (3 points)

Implement sequence truncation to ensure inputs don't exceed `MAX_SEQ_LENGTH`.

Question 2.4 [code] (3 points)

Implement padding logic to ensure all sequences have the same length within a batch.

3 Evaluation Functions

Question 3.1 [code] (2 points)

Implement the decoding logic in the `predict_sentiment` function. Extract only the generated tokens (not the input prompt) from the model output.

Question 3.2 [code] (2 points)

Implement sentiment extraction from the generated text. Handle cases where the output may not exactly match "positive" or "negative".

Question 3.3 [code] (4 points)

Calculate the following evaluation metrics in the `evaluate_model` function:

- Accuracy
- Macro F1 score
- Macro Precision
- Macro Recall

Question 3.4 [code] (2 points)

Generate and save a confusion matrix visualization for the evaluation results.

4 Full-Parameter Supervised Fine-Tuning

Question 4.1 [code] (2 points)

Count the total and trainable parameters of the full model. Calculate:

- Total number of parameters
- Number of trainable parameters
- Percentage of trainable parameters

Question 4.2 [code] (6 points)

Configure the training arguments for full-parameter SFT. Set appropriate values for:

- Output directory
- Number of training epochs
- Batch sizes (training and evaluation)
- Learning rate
- Gradient accumulation steps
- Mixed precision training (bf16)
- Logging and evaluation strategies

Total for Section 4: 10 points

5 LoRA Configuration and Training

Question 5.1 [code] (4 points)

Configure the LoRA parameters using `LoraConfig`. Set:

- `r=8`: Rank of low-rank matrices
- `lora_alpha=16`: Scaling factor for LoRA weights
- `target_modules=["q_proj", "v_proj", "k_proj", "o_proj"]`: Target attention layers
- `lora_dropout=0.05`: Dropout for regularization
- `bias="none"`
- `task_type="CAUSAL_LM"`

Question 5.2 [code] (2 points)

Implement the LoRA weight merging operation. Merge the trained LoRA adapter weights with the base model weights.

6 Model Architecture Analysis

Question 6 [written] (10 points)

Please examine the model architecture of Qwen-3-0.6B (`https://huggingface.co/Qwen/Qwen3-0.6B`) and mathematically derive the number of trainable parameters under the current LoRA configuration.

Your derivation should include:

- The base model architecture specifications (hidden size, number of layers, etc.)
- The formula for calculating LoRA parameters per layer
- The total trainable parameters across all layers

Show your mathematical steps clearly.

7 Performance Analysis

Question 7 [written] (5 points)

Why is LoRA-SFT training time shorter than full-parameter SFT? Provide a detailed explanation considering:

- Number of parameters being updated
- Gradient computation complexity
- Memory bandwidth requirements

8 Robustness and Generalization

Question 8 [code + written] (10 points)

For example, if we switch to a different prompting format, will accuracy drop significantly when the train and test prompts are not completely consistent?

Conduct an experiment to demonstrate this:

- Design an alternative prompt format
- Test the trained model with the new format
- Report and analyze the results

Question 9 [written] (5 points)

How can we increase robustness in this scenario, given that users' queries are highly diverse? Propose at least three strategies to improve model robustness to prompt variations.

9 Error Analysis

Question 10 [code + written] (5 points)

Please propose five different test cases that will cause the SFT-trained model to classify incorrectly, and verify them experimentally.

For each test case:

- Describe the characteristics that make it challenging
- Show the actual model prediction
- Explain why the model failed

How to submit

1. Fill up your student ID and name in the Jupyter Notebook.
2. Click the Save button at the top of the Jupyter Notebook.
3. Select Cell - All Output - Clear. This will clear all the outputs from all cells (but will keep the content of all cells).
4. Select Cell - Run All. This will run all the cells in order, and will take approximately 1-2 hours depending on your GPU.
5. Once you've rerun everything, select File Download as PDF via LaTeX.
6. Look at the PDF file and make sure all your solutions are there, displayed correctly. **The PDF is the only thing your graders will see!**
7. Submit your PDF on eDimension.

Grading Rubric

Component	**Points**
Code runs completely	14
Q1.1: Data reformatting explanation	4
Q1.2: Training without conversion	4
Q1.3: Data splitting and sampling	7
Q2.1-2.4: Dataset class implementation	10
Q3.1-3.4: Evaluation functions	10
Q4: Full-parameter SFT	10
Q5: LoRA configuration and training	6
Q6: Model architecture analysis	10
Q7: Training time analysis	5
Q8: Prompt format robustness	10
Q9: Robustness strategies	5
Q10: Error analysis	5
Total	**100**