CMPE 460: Deep Learning
Assignment 3

Istanbul Bilgi University

Spring 2024-2025

Rules

- Submit all `ipynb` or `py` files for your solution, **do not compress the files**. If you are submitting a `py` file, then also submit your figures, tables, and comments in a `pdf` file.
- Plagiarism is strictly prohibited and will be penalized.

Transformer Fine-tuning

In this assignment you will fine-tune a pre-trained Transformer model for a specific downstream task, such as text classification, named entity recognition (NER), or sentiment analysis. You will explore the process of adapting a general-purpose Transformer model (e.g., BERT, GPT-2, T5) to your task, and evaluate the model's performance. You are free to choose any downstream task. Make use of efficient training techniques by following the HuggingFace guide[1].

1. Setup and Preprocessing

- Choose a pre-trained Transformer model from Hugging Face's Model Hub (e.g., BERT, DistilBERT, RoBERTa, T5).
- Select a downstream task. For example:
  - **Text Classification**: Classify news articles, product reviews, or tweets.
  - **Named Entity Recognition (NER)**: Extract named entities (persons, organizations, locations, etc.) from text.
- Select a dataset for the chosen task. Examples include:
  - IMDb dataset for sentiment analysis.
  - CoNLL-03 for NER.
  - AG News or Amazon Reviews for text classification.
- Preprocess the dataset:
  - Tokenize the text using the tokenizer associated with the chosen pre-trained model.

[1] https://huggingface.co/docs/transformers/perf_train_gpu_one

  - Convert labels to the required format (e.g., integers for classification, BIO format for NER).
  - Split the dataset into training, validation, and test sets if there are no predefined splits.

2. Model Fine-Tuning

- Load the pre-trained Transformer model and the tokenizer.
- Fine-tune the model on the downstream task:
  - Set up the model with an appropriate classification head or sequence labeling head (for NER).
  - Use a suitable loss function
  - Use an optimizer like AdamW, and **experiment with different learning rates**.
  - Implement training loops, validation, and model checkpoints to save the best model. (You can use the `Trainer` class)
- Fine-tune the model for at least 3–5 epochs, depending on the size of the dataset.

3. Hyperparameter Tuning

- Experiment with different hyperparameters (e.g., batch size, learning rate).
- Use techniques like learning rate scheduling or early stopping to improve performance.
- Compare results after tuning the hyperparameters.

4. Evaluation

- Evaluate the model on the test set.
- Report the appropriate metrics depending on the task.
- Compare the performance of the fine-tuned model with a baseline model (e.g., a simple logistic regression model or an untrained Transformer model).

5. Visualization and Reporting

- **Plot training and validation loss curves** to analyze the training process.
- Visualize the performance of the model on **a few test examples, highlighting the model's predictions.**
- **Discuss the challenges** faced during fine-tuning, such as overfitting, underfitting, and model convergence.
- **Summarize your findings** and the impact of fine-tuning the pre-trained model on the task.