LLM
ASSIGNMENT-3

Tanmay Singh
2021569
CSAI
Class of ‘25

GitHub Link: https://github.com/imtanmay46/LLM.git

Approach & Results

Loading & Preprocessing the Dataset

I loaded the ‘snli’ dataset using the ‘load_dataset’ library & extracted the train, validation and testing samples as described, in a dictionary format.

I created a custom ‘tokenize_samples’ function which takes the sample set (in dictionary format) as an input and returns the ‘input_ids’, ‘attention_masks’ & ‘labels’ using the in-built Tokenizer from the Transformers library.

These extracted tokenized samples (train, validation & test) are then converted into a Dataset format (original format in which the ‘snli’ model was loaded, to create ‘train_dataset’, ‘val_dataset’ & ‘test_dataset’).

Then, I created the required functions for computing the number of parameters of the model (both total & trainable), gpu utilization (total memory, used memory & free memory) & model evaluation (predicted labels, accuracy & f1 score) in the subsequent cells.

Later, I created the quantization config (bnb config) using the bits & bytes library, and loaded the base model (phi-2, pre-trained) using the AutoModelForSequenceClassification.

Once the model was loaded, the gpu utilization, model sizes (total & trainable parameters) and accuracy/f1 scores were computed by evaluating the model.

Once the evaluation is done on the base model, I defined a peft config for QLoRA fine-tuning with a rank(r) of 16, alpha of 64 & dropout rate of 0.05 along with other parameters.

I extracted the peft_model and computed the relevant details (gpu utilization and model sizes) of the model & set it up for QLoRA fine-tuning by defining the necessary 'training_args' and used the 'trainer' module to train the model (using train & val dataset).

The gpu utilized for fine-tuning the model & the time taken to fine tune it was computed once the model was trained for 5 epochs & the checkpoints saved for each of them.

For Inference, I deleted the above-defined models & tokenizers and loaded the base model & tokenizer again to evaluate the pre-trained model & compute the failure cases, alongside computing the accuracy & f1 scores.

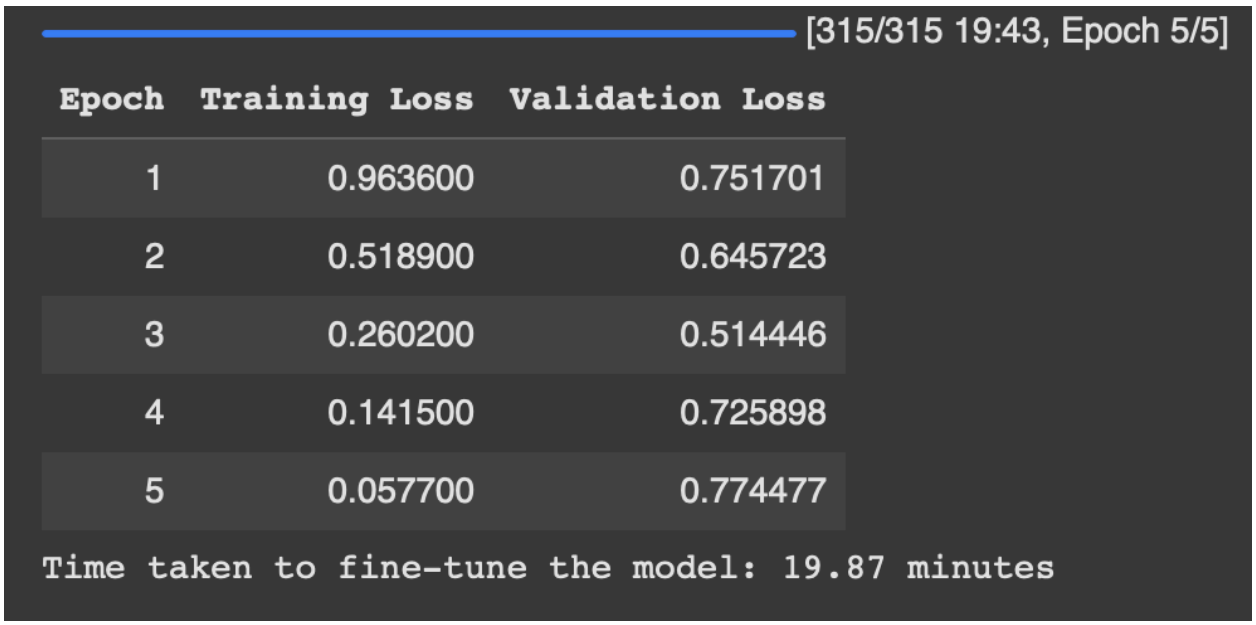
Later, I loaded the saved checkpoint and used the PeftModelForSequenceClassification to load the LoRA model (fine-tuned model) for evaluation.

I computed the failure cases, the successful ones (correctly predicted) & also found out which ones were missed by the fine-tuned model, which was correctly classified by the base model & the improved predictions (ones missed by the base model but corrected by the fine-tuned model).

Results

Metrics/ Model Type	Model Size (Parameters) Total	Trainable	GPU Utilization Used Memory	Evaluation Metrics Acc	f1 score
Base Model (pre-trained)	1,390,277,120	131,248,640	1.90 GB (out of total 15 GB available)	34.0%	0.3036
Fine-tuned Model (using QLoRA)	1,413,877,760	23,592,960	2.00 GB used for Fine-tuning the Phi-2 model (4.05 GB after fine-tuning, 2.05 GB before fine-tuning)	86.00%	0.8590

Training Results:



Reasoning for Sample Classification (Misclassification):

The base model was mostly classifying the labels into either of the classes (for my case, the model classified labels into class 0 & class 2, predominantly), hence accounting for the accuracy being in the 30s (33% approx) since the dataset contained an equal number of samples of each class & the model classifying only 1 label correctly led to this accuracy.

On the other hand, the fine-tuned model performed significantly better as it was accustomed to suit the downstream task (of classification on the ‘snli’ dataset), and predicted labels better ranging from class 0 to class 2 (i.e. all 3 classes instead of only 1 or 2 class), hence performing better than the base-model.