CS480/680, Spring 2025

Assignment 3

Designer: Ahmad Rashid; Instructor: Hongyang Zhang

Released: July 2; Due: July 30, noon

Instructions

- We do not accept hand-written submissions.
- This assignment is due by noon on July 30, 2025.
- For questions labelled as "**coding**", please follow the instructions provided and implement the required features. Unless otherwise specified, all implementations should be in *Python* using a *Jupyter Notebook*. Before submission, please make sure that your code can run without any errors. Also, be sure to save the output of each cell, as any missing output may not be graded.
- Please submit the following TWO files to LEARN:
  - A write-up in PDF format: the written answers to ALL questions, including the reported results and plots of coding questions, in a single PDF file.
  - An IPYNB file: your implementations for ALL coding questions. Please save the output of each cell, or your coding questions may NOT be graded.

Question 1 (Large Language Model - Classification 10 points) *Large language models (LLMs) such as BERT can be fine-tuned for many downstream tasks. In this question, we will train a BERT model for sentiment classification using the popular IMDB dataset. BERT is an encoder-only model and is typically suited for classification. The IMDB dataset consists of movie reviews and positive or negative sentiment labels. You can load the dataset and the model using the following:*

```
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModel

ds = load_dataset("stanfordnlp/imdb")
tokenizer = AutoTokenizer.from_pretrained("google-bert/bert-base-uncased")
model = AutoModel.from_pretrained("google-bert/bert-base-uncased")
```

1. *(3 points). Write a BERTSentiment class which modifies the base model with a task specific layer and has methods to return the loss and the output when needed. You may implement that using a single forward method which returns the loss during training and the output during evaluation.*
2. *(3 points) Write a training loop and train the model on the IMDB training data for 1 epoch. Plot the training loss against the number of iterations.*
3. *(2 points) Write a function to calculate the accuracy of the trained model. Then evaluate and print the accuracy on the test set.*
4. *(2 points) We will now observe the robustness of the model when the data has been adversarially modified. Textfooler is a model-based adversarial attack method that is used to evaluate the robustness of text classification systems. Download the adversarial imdb dataset from `https://drive.google.com/file/d/1Nc_GfqIvOfARNfqc8RqOxyX7gyE2IIWT/view?usp=drive_link`. Each sample has the original version and an adversarial version. Calculate the total number of times the label changes when evaluating the normal versus adversarial sample.*

Note: You may use the following tempelate for the BERTSentiment class or modify as necessary

```
import torch
import torch.nn as nn
from transformers import AutoTokenizer, AutoModel


class BERTSentiment(nn.Module):
    def __init__(self):
        super().__init__()
        self.bert = AutoModel.from_pretrained("google-bert/bert-base-uncased")
        self.num_labels = #IMPLEMENT ME
        self.classifier = #IMPLEMENT ME

def forward(self, input_ids, attention_mask,token_type_ids=None):

    if torch.cuda.is_available:
        input_ids = input_ids.cuda()
        attention_mask = attention_mask.cuda()
    outputs = self.bert(
    input_ids,attention_mask=attention_mask,token_type_ids=token_type_ids)
    features = outputs[1]
    logits = self.classifier(features)
    loss = #IMPLEMENT ME

    if self.training:
        ########################################################################
        # IMPLEMENT ME
        # return the loss

    else:
        ########################################################################
        # IMPLEMENT ME
        # return the class probabilities using softmax.
```

Question 2 (Large Language Model - Generation 10 points) *In this question we will evaluate the GPT2 language model for text summarization. GPT2 is a decoder only model and is typically better suited for generation tasks such as summarization. We will be working with the Reddit based TL;DR dataset. For this work we will be experimenting with different decoding algorithms.*
You can load the model and the dataset using the following code.

```
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForCausalLM

ds = load_dataset("trl-lib/tldr")
tokenizer = AutoTokenizer.from_pretrained("openai-community/gpt2")
gpt2-small = AutoModelForCausalLM.from_pretrained("openai-community/gpt2")
gpt2-medium = AutoModelForCausalLM.from_pretrained("openai-community/gpt2-medium")
```

1. *(5 points). The prompts in the dataset are appended with the TL;DR string at the end so that the language model can generate a summary. Run the GPT2 medium model on the TL;DR test set and evaluate the Rouge-L score. You can use the generate() function in the transformers library. Run the experiment for a) greedy decoding, b) top-k sampling with k=20 and c) top-p sampling with p=0.9. Print the relevant rouge scores. The rouge score can be calculated using the following library*

```
import evaluate
rouge = evaluate.load('rouge')
```

2. *(5 points) As the LLMs scale in size efficient inference is a concern. One of the ways of efficient decoding is using speculative decoding. In speculative decoding, we use a small model to assist in the decoding of the larger model to reduce the number of calls made to the larger model. You can use the pretrained GPT2-medium as the target model and the pretrained GPT-2 model (GPT2-small) as the assistant model. Speculative decoding is supported by the model.generate() function in the transformers library. You can specify an assistant model by specifying the assistant_model field and specify the number of speculative tokens that are generated using the num_assistant_tokens field. Sample 100 prompts from the TL;DR test set and record the generation time when a) using just the larger model and b) using speculative decoding and setting num_assistant_tokens to* 5, 10 *and* 15 *respectively. Explain how the plot changes when changing the assistant tokens. Fix a random seed to evaluate on the same prompts.*

Note You can improve the performance of the GPT2 model by fine-tuning on the TL;DR training set using trainers from the transformers library or standard Pytorch trainers. You will not be graded on this.