CS-497: Deep Learning for Natural Language Processing
Group Project #2 - Question Answering with Transformers

(40.0 points)

Winter 2023

Description: Question-answering with transformer-based architectures is becoming the dominant paradigm in natural language processing. Among the various formulations of this task, multiple-choice question-answering is among the most tractable. In this assignment, you will perform question-answering on the OpenBookQA dataset (see https://arxiv.org/abs/1809.02789). OpenBookQA is a multiple-choice question-answering dataset of elementary school-level scientific questions. Questions are posed in natural language with four possible choices. Background knowledge in the form of a one-sentence "fact" is also provided.

This task's primary formulation requires you to identify the correct "fact" from among 1,326 facts. To focus on transformer architectures in this assignment, you are given the "fact" associated with each question. The training dataset consists of approximately 5,000 observations; the validation and test sets have 500 observations each. Each observation contains a question stem, the associated fact, four labeled choices and a correct answer label. You may ignore all other information.

You will use transformer-based architecture to perform multiple-choice question-answering for classification and generative approaches. Starter code is provided to read the JSON dataset files. You may need to install Huggingface Transformers (see https://huggingface.co/docs/transformers/installation) on your machine.

Tasks:

1. **Classification Approach** (20 pts): For this approach, you must use the `bert-base-uncased` checkpoint for the `BertModel` (see https://huggingface.co/docs/transformers/v4.26.1/en/model_doc/bert) and fine-tune the model on the training set using your own code. Implementations using `BertForMultipleChoice` or other specialized variants will not be accepted. Selecting an encoding and methodology for this task is up to you. Among the large number of example implementations available, I found this on to be helpful https://colab.research.google.com/github/NielsRogge/Transformers-Tutorials/blob/master/BERT/Fine_tuning_BERT_(and_friends)_for_multi_label_text_classification.ipynb.

   Please describe your approach sufficiently so that someone familiar with transformers can replicate your results. Also, report your zero-shot and fine-tuned accuracies on the validation and test sets. Please discuss any limitations of your approach and possible solutions. Combined, this should be one page at maximum. Using the approach described in class, I was able to achieve an accuracy of 55.9% on the test set.

2. **Generative Approach** (20 pts): For this approach, you are free to select a decoder-only or an encoder-decoder architecture of your choice. You must design a prompt and output format to perform this task. The design will include a methodology of interpreting generated output to evaluate model performance (which may consist of exact match, ROUGE-F1 scores, etc.) and must accurately capture model performance. You may fine-tune your architecture using the scripts in `transformers/examples` and use functions like `model.generate()` (see https://huggingface.co/blog/how-to-generate).

   Please describe your approach in sufficient detail such that someone familiar with transformers can replicate your results. Also, report your zero-shot and fine-tuned accuracies on the validation and test sets, and provide examples of your prompts and outputs. Please discuss any limitations of your approach and

possible solutions. Discuss the performance of your generative approach compared to your classification approach. Combined, this should be at most two pages. Using an exact match performance metric based on `A, B, C or D` encoding, I can achieve an accuracy of 40.8% on the test set using GPT-2 (https://huggingface.co/gpt2#training-procedure) and my own code for fine-tuning with an approach similar to (https://reyfarhan.com/posts/easy-gpt2-finetuning-huggingface/). Note: fine-tuning using `transformers/examples` enables me to achieve an accuracy of only 27.6%.

What to Submit: Please submit a single `.zip` file for the group. The zip file should contain the following:

- A single `.pdf` for all written responses and results.
- Individual `.py` files (two files maximum) for your Classification and Generative approaches (note: please do not submit Jupyter Notebooks).

Only one group member needs to submit.