Advanced NLP Exercise 1

Submission: Until May 23 2023, 23:59 PM

1 Open Questions (40 points)

1. We talked about question answering (QA) as being an expressive format for annotating both intrinsic as well as extrinsic tasks. **List three QA datasets that use QA to annotate intrinsic concepts. For each, write a short explanation (1-2 sentences) for why it measures an intrinsic property of language understanding?**

2. In class we talked about abstractive vs. extractive as two general approaches towards summarization. There are many variants to these tasks, in particular, in this question we'll focus on (1) Interactive summarization and (2) Multi-document summarization. **For each of these, answer the following questions, each in 1-2 sentences:**

   (a) What is the task definition?

   (b) What are some notable benchmarks? What domain do they annotate? How many samples?

   (c) What are some of the inherent challenges presented by the task?

3. In class we discussed two methods for contextual representation: recurrent neural networks (RNNs) and Transformers. We mentioned that one of the benefits of Transformers is that they can be efficiently parallelized. Does this property apply to *training* or *inference*, or *both*? Please explain how each of these steps is affected by the choice of one of these modeling approaches in terms of efficiency.

4. Consider the following problems, please explain your answer.

   (a) You have a dataset of 10,000 labeled training examples and 5,000 test examples for the task of predicting the genre of movies out of a set of 5 different options. You have a single 12G GPU, and no available money to purchase online services (that is, you don't have a lot of computing resources). Which of the following models would you use? Please explain your answer.

   - Fine-tune ELECTRA-base
   - Prompt-tune T5 XXL (11B parameters)
   - Do in-context learning with InstructGPT

   (b) You have a dataset of pairs of sentences that are labeled as to whether they follow the same style (styles could be funny, ironic, sad, toxic, etc.). You have the BERT model at your disposal. How would you use it to get a model for solving your task?

   (c) You have a new cool prompting approach that works really well! Give at least two reasons for using ChatGPT as a baseline, and two reasons for not using it.

2 Programming Exercise (60 points)

In this exercise you will be asked to fine-tune pretrained large language models to perform the sentiment analysis task on the SST2 dataset.

2.1 What do I need to do?

1. Write Python code for fine-tuning the following three pretrained language models to perform sentiment analysis on SST2: *bert-base-uncased, roberta-base, google/electra-base-generator* (these are the names of the models in the HuggingFace hub). Here's some additional information:
   - **Hyper-parameters:** You should use the default training hyper-parameters (e.g., number of epochs, learning rate etc.).
   - **Model configuration:** You should use the default configuration for all the models.
   - **Split:** You should use the train split for training.
   - **Truncation and padding:** You should truncate inputs to the maximum sequence length possible for each model, and use dynamic padding.
   - **Weights&Biases:** Track each run with Weights&Biases.
2. Fine-tune each model with several seeds and compute the mean and standard deviation of the accuracy on the validation set for each of the three models. The number of seeds should be given as an external argument. The mean and std of the accuracies should be documented in the *res.txt* file, in the same format as the example file in the Moodle.
3. Select the model with the highest mean accuracy on the validation set, and use its best seed to run prediction on the test set. Predictions should be documented in the *predictions.txt* file, in the same format as the example file in the Moodle. **Note**: During prediction, unlike during training, you should not pad the samples at all.

2.2 What do I need to submit?

You are required to submit a path to a **public** Github repository. The repository should contain (at least) the following files:

- The README.md file that can be found in the Moodle.
- A python script named *ex1.py* that accepts four command line arguments, in the same order they are presented here:
  - Number of seeds to be used for each model. If the user specified n seeds should be used, you should use the seeds $0, 1, ..., n-1$.
  - Number of samples to be used during training or -1 if all training samples should be used. If a number $n \neq -1$ is specified, you should select the first n samples in the training set.
  - Number of samples to be used during validation or -1 if all validation samples should be used. If a number $n \neq -1$ is specified, you should select the first n samples in the validation set.
  - Number of samples for which the model will predict a sentiment or -1 if a sentiment should be predicted for all test samples. If a number $n \neq -1$ is specified, you should select the first n samples in the test set.

  And runs all the needed code to fine-tune the models, run prediction on the best model and generate the *res.txt* and *predictions.txt* files (see below). As mentioned above, the best model is the model with the highest mean accuracy, and you should use its highest performing seed for prediction.
- A file named *res.txt* containing the mean and standard deviation of accuracies of all the models you tested on the validation set, the training time, and the prediction time, in the format of the *res.txt* file that can be found in the Moodle. To generate both the *res.txt* file and the *predictions.txt* file (see below), run ex1.py with 3 seeds and with all the samples for training, validation and prediction.

- A file named *predictions.txt* containing the model predictions for the entire test set, in the format of the *predictions.txt* file that can be found in the Moodle.
- A file named *requirements.txt* containing all the required Python packages to run your script, so that a simple *pip install -r requirements.txt* command should do.
- A file named *train_loss.png* containing the train loss plot generated by Weights&Biases, including only the first seed run for each of the three models.

2.3 What do I need to pay attention to?

- You may draw inspiration from existing scripts. **However**, you are not allowed to copy an existing script, run it, analyze the resulting log files and using it to create *res.txt*. **You need to be able to explain every line of code in your ex1.py file.**
- There's no need to save checkpoints of the model during training, you only need the final model for the accuracy on the validation test. This might be important for disk space considerations.
- Read the *README.md* file in the Moodle and make sure your Github repository is compatible with the commands specified in this file. As a test before submitting the exercise, we recommend to git clone your repository, follow the instructions in the readme file and run ex1.py with a small number of samples.
- Use the *AutoModelForSequenceClassification* class to load your models.
- Remember to run the *model.eval()* command before prediction: some layers behave differently during training, and this command will signal the model that it should be in inference mode.