Advanced NLP Exercise 1

Submission: Until May 12 2025, 23:59 PM

You should submit all the answers to Section 1 along with the URL to the Github repo from Section 2 and the answers to the qualitative analysis question in a single pdf file through Moodle.

1 Open Questions (20 points)

1. Question answering (QA) can be 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 discussed several methods to implement inference-time scaling.
   (a) For each method we covered, answer the following:
      - Provide a brief description of the method.
      - Outline its advantages.
      - Identify its computational bottlenecks (i.e., the resources heavily consumed during its execution).
      - Indicate whether the method can be parallelized.

   (b) Suppose you must solve a complex scientific task requiring reasoning, and you have access to a single GPU with large memory capacity. Which method would you choose, and why?

2 Programming Exercise (80 points)

In this exercise you will be asked to fine-tune a pretrained large language model to perform the paraphrase detection task on the MRPC dataset from the GLUE benchmark (huggingface.co/datasets/nyu-mll/glue).

2.1 What do I need to do?

1. Write Python code for fine-tuning *bert-base-uncased* to perform paraphrase detection on MRPC. Follow these guidelines:
   - **Split:** Use the train split for training, validation split for evaluation during training, and test split for prediction (see below).
   - **Hyper-parameters:** Experiment with hyperparameters (number of epochs, learning rate, batch size) to maximize validation accuracy. After reaching 75% validation accuracy and testing at least three combinations, you may stop. Don't experiment with more than 5 epochs.
   - **Model configuration:** Use the default configuration.
   - **Truncation and padding:** Truncate inputs to the maximum sequence length allowed for each model and use dynamic padding.
   - **Weights&Biases:** Track all training runs using Weights&Biases. Make sure you log the training loss every step.
2. After performing hyperparameter search, use all trained checkpoints to make predictions on the test set. Then, answer the following:
   - Did the configuration that achieved the best validation accuracy also achieve the best test accuracy?
   - **Qualitative analysis:** Compare the best and worst performing configurations. Examine validation examples where the best configuration succeeded but the worst failed. Can you characterize the types of examples that were harder for the lower-performing model?

   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 the following command line arguments:
  - ⋆ *--max_train_samples*: 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.
  - ⋆ *--max_eval_samples*: 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.
  - ⋆ *--max_predict_samples*: Number of samples to be used during prediction or -1 if all prediction samples should be used. If a number $n \neq -1$ is specified, you should select the first n samples in the test set.
  - ⋆ *--num_train_epochs*: Number of training epochs.
  - ⋆ *--lr*: Learning rate.
  - ⋆ *--batch_size*: Train batch size.
  - ⋆ *--do_train*: Run training.
  - ⋆ *--do_predict*: Run prediction and generate the *predictions.txt* file (see below).
  - ⋆ *--model_path*: The model path to use when running prediction.

- A file named *res.txt* containing the validation accuracy of all the configurations you tested on the validation set, in the format of the *res.txt* file that can be found in the Moodle.
- A file named *predictions.txt* containing the predictions of the configuration with the highest validation accuracy for the entire test set, in the format of the *predictions.txt* file that can be found on 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 for the different configurations, see example in the Moodle.
- The answer to the questions in the qualitative anaylsis section above (Section 2.1, part 2, second bullet) should be submitted in the same PDF as the answers to the questions from Section 1.

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. **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.