Coding Project 4 Instructions

Deep Learning 2023 Spring

Due on 2023/5/12

In this coding assignment, you need to implement LSTM and Transformer-based models for text generation and reading comprehension tasks. Similar to Coding Project 2, this project requires a relatively long training time. Therefore, we provide Python scripts instead of Jupyter Notebooks, with which you can conveniently train your model on the server. Note that you only need to submit one `pdf` file as the report for this coding project and report counts for 3 pts in this project. Your submission(including all of your codes and models) should not exceed the file size limit (1024MB) of web learning. Make sure you can run `[task]/evaluation.py` in your submitted folder, we will not grade your submission if we cannot run your model on test split! When submitting, please name the compressed file in the format "`{Name}_{StudentID}.zip`." See `README.md` for running instructions.

Part I

Environment

The TA will test your model on the following environment:

- Operating System: Linux (Ubuntu 22.04)
- GPU: NVIDIA RTX 3090

The required Python packages and their versions are:

Package	**Version**
huggingface-hub	0.13.4
numpy	1.24.2
tokenizers	0.13.3
torch	1.13.1
tqdm	4.65.0
transformers	4.28.1

Part II

Text Generation(4pt)

1 Task

In this task, you need to train sequence-to-sequence models for the conditional generation and language models for the unconditional generation. **You are forbidden to use pre-trained word embedding and pre-trained models in this task.**

1.1 Writing Couplets with Sequence-to-Sequence Models

Sequence-to-sequence aims to train a model

$$\mathbb{P}_{\mathsf{Y}|\mathsf{X}}(Y|X) = \prod_{t=1}^{L} \mathbb{P}_{\mathsf{Y}_t|\mathsf{Y}_{<t},\mathsf{X}}(Y_t|f_{\text{dec}}(Y_{<t}), f_{\text{enc}}(X))$$

on paired data (X, Y), where f_{enc} and f_{dec} are the encoder and decoder model respectively. In this task, we provide a couplet corpus, and you need to:

1. Complete the code of `Seq2SeqModel` in `generation/lstm.py` and enhance it with the attention mechanism.
2. Complete the code for modules in `generate/transformer.py` and train a sequence-to-sequence model with transformer architecture.
3. Complete the code in function `generate` of `Seq2SeqModel` with beam search for LSTM and Transformer respectively.

1.2 Writing Poems with Language Models

Language models learn the sequence distribution in an auto-regressive manner:

$$\mathbb{P}_{\mathsf{X}}(X;\theta) = \prod_{t=1}^{L} \mathbb{P}_{\mathsf{X}_t|\mathsf{X}_{<t}}(X_t|X_{<t};\theta).$$

In this task, you are given a Chinese classical poetry corpus. You need to complete the code of `LMModel` in `generation/lstm.py` and `generate/transformer.py`. Construct the model and implement function `generate` with beam search.

2 Submission

You need to submit all codes, your trained models and your report.

- In this task, we expect 4 models named "lstm_lm.pt", "lstm_seq2seq.pt", "transformer_lm.pt" and "transformer_seq2seq.pt". **Place them appropriately** and make sure that `generation/evaluation.py` runs fine.
- In your report, we expect to see **training curves** and **generated samples** for LSTM and transformer-based model respectively.
- Show an ablation study of the attention mechanism on LSTM model in your report.

3 Grading

We will grade this task according to **the quality of your generated samples and the perplexity in the test set**.

4 Tips

1. Make sure you can run `generation/evaluation.py` ****in your submitted folder, we will not grade your submission if we cannot run your model on test split****!
2. You **can modify any code for training if necessary**, but make sure that we can use the original code for inference and evaluation. (We will overwrite all codes without "TODO" blocks before running the evaluation.)

Part III

Reading Comprehension(3pt)

1 Task

In this task, we provide a reading comprehension dataset. There are thousands of articles in the dataset, and each article has several questions. For each question, there are 2 to 4 choices, and only one of them is correct. You need to build and train a model to choose the correct answer.

2 Submission

You need to submit all codes, your trained model and your report.

- We expect your model named "cls_best.pt". **Place it appropriately** and make sure that our evaluation code runs fine.

- In your report, we expect to see your **the details of your model**, **all the hyper-parameters**, **all the tricks or training techniques you use**, and **the training curve**.

3 Grading

We will grade your model according to the accuracy in the test set. In this task, you will receive points based on your test accuracy (*acc*). The points are calculated as follows:

$$\text{Points} = \min\left(\frac{acc - 26}{30}, 1\right) * 3$$

where *acc* represents your test accuracy in percentage.

4 Tips

1. Make sure you can run `classification/evaluation.py` ****in your submitted folder, we will not grade your submission if we cannot run your model on test split****!
2. You **can modify any code you for training if necessary**, but make sure that we can use the original code for inference and evaluation. **Do not make any modification to *classification/evaluation.py***.
3. You are encouraged to use pre-trained word embedding and pre-trained models to improve the performance, but keep in mind that we will not install extra packages when testing your model.
4. If you are using a pre-trained model, make sure it can run ****offline****, we will not download weights when running test.

5 Bonus

The best submission with the highest testing accuracy will **get** 1 **bonus point** for the final course grade.