CS310 Natural Language Processing
Assignment 6: BERT Pretraining
Total points: 50

Tasks

- Implement a mini-BERT model and carry out the masked language modeling (MLM) next sentence prediction (NSP) tasks, using the provided code framework.
- Read README.md carefully.

Submission

- Python files: model.py and train.py
- The saved model file `bert_model.pt`.

Requirements

1) Implement the model (35 points)

The majority of BERT model are already implemented, and there are two places that require your implementation: the `EncoderLayer` and `BERT` classes.

The `EncoderLayer` class defines how input go through the multihead self-attention layer and the feedforward layer properly. The input to the former includes the encoded sequence and an attention mask. Note that the attention mask here is not the one for masking out the future tokens in training a transformer decoder, but rather it is for masking out all the [PAD] tokens.

The BERT class defines the data flow of the entire model. Your implementation focuses on the forward function, in which you need to produce the outputs for the MLM and NSP tasks:

- For NSP, you goal is to use the last layer representation of [CLS] to produce the logits for a binary classifier. You should use the provided fully-connected layer, tanh activation, and classifier components accordingly.
- For MLM, you goal is to gather the last layer representations of all the masked tokens, whose indices are denoted by the input tensor maked_pos. The gathered mask representations are then passed through a fully-connected layer (with a special GELU activation) and a layer norm, and finally a $V \times d$ decoder. The decoder's job is to map the representations to logits of vocabulary size, and it shares the same weight as the input token embedding layer.
- The `forward` function of `BERT` takes as input three tensors: `input_ids`, `segment_ids`, and `masked_pos`. You should use `masked_pos` to gather the mask representations. The function returns two logits, `logits_lm` and `logits_clsf`, which you will use later in the training loop to compute the loss.

2) Implement the training loop (15 points)

In the training loop, you conduct forward pass on the model, take the returned logits to compute the cross-entropy losses, `loss_lm` and `loss_clsf`, respectively. You should sum them up as the final loss. Note that `masked_tokens` serves as the ground truth for the MLM task, and `is_next` is the ground truth for the NSP task. By default, we use the same loss function instance (`criterion`) for the two tasks, but you can feel free to try separate losses.

****

Grading rubrics

The training loss at the end of 1 epoch should be below 10.0

You get full 50 points for loss ≤ 10.0, otherwise 35 points.

****