netid: lw2350
name: Lizhong Wang

Homework 4

DS-GA 1008 Deep Learning
December 5, 2020

Problem 1

(a) Bidirectional Encoder Representations from Transformers

(b) BERT used two datasets for pretraining: BooksCorpus (800M words) and English Wikipedia (2,500M words).

(c) The backbone to BERT-large is 24 layers of Transformers. Each Transformer layer has 16 self-attention heads, and the dimension of the hidden representation is 1024.

(d) Position embedding is a representation of positional information. Position embedding tells the model where the input token is located in the input sequence.
We need position embedding because all tokens in the input sequence operate in parallel in the Transformer architecture. In other words, there's no notion of word order when we use Transformer or BERT. Therefore, we need to use position embedding to retrieve this positional information.

(e) BERT has two objectives: Masked Language Model (MLM) Next Sentence Prediction (NSP).

MLM: Randomly mask some tokens, and use other words in the context to predict the masked words.

NSP: To predict whether one sentence is logically followed by the other. During pretraining, we input a pair of sentences A and B. 50% of time, B is the actual sentence follwing A; 50% of time, B is just a random sentence from the corpus. NSP objective is to predict if B actually follows A.

(f) negative log likelihood $= -\log p_\theta(y_{t1}|y_{\neq t1,t2}) - \log p_\theta(y_{t2}|y_{\neq t1,t2})$

(g) using MLM increases the performance on downstream tasks comparing to using left-to-right language modeling. According to Table 5, the accuracy on MNLI dataset is increased from 82.1% to 83.9% by switching to MLM.

(h) BERT-large was pretrained on 16 Cloud TPUs (64 TPU chips), and it took 4 days.

(i) BERT paper used the second way - finetune both the backbone and the classification head. The benefit of this method is that the representation of the input text can be changed in order to better work with the current downstream task. Therefore, the model will have a better performance if we finetune both the backbone and the classification head.

(j) [CLS] is a special token. The hidden states of the [CLS] token are aggregated representations of the entire input sequence on each layer.
I would use the last layer hidden representation of [CLS] as the input features to the classifier.

(k) Since the original [CLS] token is a weighted average of the final hidden states of all input tokens (the weights are the attention weights), we can train a weighted average layer of all last hidden states to estimate the [CLS] token. Then we can use this estimated [CLS] token as the input to the pooler layer and the final classification layer.
For example, suppose the input sequence length is 512, and the dimension of the final hidden state for each token is 768. Then we can pack all final hidden states into one matrix, $\mathbf{H} \in \mathbb{R}^{768\times512}$, where each column is a final hidden state vector of one token. Then we can train a vector $\mathbf{x} \in \mathbb{R}^{512}$, such that estimated [CLS] = $\mathbf{Hx}$. We can fine-tune this weighted average layer along with the language model and the final classifier.

(l) The BERT paper uses a slanted triangular scheduler for pretraining. The learning rate warms up over the first 10,000 steps, then it linearly decay over the remaining 990,000 steps.

Problem 2

We can use the first 6 layers of the pre-trained English BERT as the encoder of this translation model, and we take the hidden state of the [CLS] token after the 6th BERT layer as the hidden state of the entire input English sentence.

Then the decoder takes two inputs: 1) the target Burmese sentence $\mathbf{y}$ and 2) the output of BERT encoder $encoder(\mathbf{x})$. The decoder models the probability of each y_t conditioned on all previous Burmese words $\mathbf{y}_{<t}$, and the representation of the corresponding English sentence $encoder(\mathbf{x})$, that is $p_\theta(\mathbf{y}|\mathbf{x})$. Finally, the model is optimized to maximize $p_\theta(\mathbf{y}|\mathbf{x})$.

We can first fix the encoder and just learn the randomly-initialized decoder. Then after several epochs, we can start to fine-tune the pre-trained encoder as well. By doing this, we only need to learn the decoder from scratch, and almost all the training samples we have are used to train the decoder.

Problem 3

(a) Catastrophic forgetting is a problem that can happen in transfer learning models. It happens when the model forgets what it learned previously during the fine-tuning stage. In other words, the information that model learned during the pre-training stage is overwritten by the new information relevent to the current task in the fine-tuning stage.
It is a problem because catastrophic forgetting decrease the generality of the model. Also, it reduce the model performance because the information acquired during pre-training stage is lost.

(b) Compare to single-task training, multi-task training makes the model generalize better by aggregating the training signal from other related tasks. MTL can be seen as introducing inductive bias from auxiliary tasks such that the model prefer hypotheses that can explain other tasks as well.

(c) Multi-task training requires simultaneous access to data used for all tasks. Therefore, we have to retrain the whole model if we want to add a new task, which makes it difficult to add new tasks.

Problem 4

(a) MNLI task has three labels: entailment, neutral, and contradiction; while RTE task has only two labels: entailment and not entialment.

(b) The best validation accuracy is 0.563177 (seed 42), 0.545126 (seed 66), and 0.563177 (seed 1024). The mean is 0.55716, and the standard deviation is 0.008509.

(c) The best validation accuracy is 0.653430 (seed 42), 0.660650 (seed 66), and 0.646209 (seed 1024). The mean is 0.65343, and the standard deviation is 0.00590.

(d) The number of trainable parameters is 109,482,240.

(e) Training from scratch gives inferior results because the from-scratch model does not contain the information that the pre-trained model has. When we fine-tune the pre-trained model, we start with a good initialization so that when we optimize the objective function related to the current task, the model will end up with a better local optimum. When we train the model from scratch, the random initialization does not guarantee a good and robust local optimum.

Problem 5

The best validation accuracy is 0.736462 (seed=2020), 0.750903 (seed=666), and 0.736462 (seed=1024). The mean is 0.741276, and the standard deviation is 0.00681.

I used the following three adapters:

- MultiNLI
  MNLI (Multi-Genre Natural Language Inference) is a very similar task to RTE. In addition, MNLI adapter is trained on a much larger dataset than RTE.

- SICK
  SICK is another similar task to RTE. SICK task outputs a score from 1 to 5 indicating the relatedness of two sentences. SICK dataset also has a label indicating whether the two sentences are entailment, neutral, or contradict.

- MRPC
  MRPC (Microsoft Research Paraphrase Corpus) task is to predict whether a pair of sentences are semantically equivalent. Since entailment is similar to semantic equivalence, I think this adapter should be helpful for RTE task.