NLP HW3 Machine Reading Comprehension with Multiple Choice

109550003 Mao-Siang Chen

1 Methodology

In this machine reading comprehension task, I choose the pretrained BERT as the basic model. Since it is a multiple choice problem, a `AutoModelForMultipleChoice` model from HuggingFace is selected.

Preprocess Data

Since the number of choices are different in each question, in order to form a batch as the input of the model, I add empty strings to make sure that all of them have the same number of choices. Then, sort the data by the length of the article to avoid forming a batch with two articles that have a huge difference in length, which will lead to unnecessary padding.

Reading Data and Construct the Dataset

First, read the data using the `json` package. Then, for each question, I concatenate the article, question and one of the choices like this:

> [CLS]article[SEP]question[SEP]choice 1[SEP]

Therefore, a single question with choices will generate four sequences. Next, we make a list with these four sequences and encode them with a tokenizer with `truncation`="only_first".
Then, put them into a custom dataset for trainer API later.

Train with Trainer API

In order to use the trainer API from HuggingFace, we need to declare a datacollator and a compute metrics function. Both of them can be found in the example code on Documents from HuggingFace. Fine-tune BERT on SWAG
The hyperparameters and their values are listed below.

- learning rate: 1e-5
- epoch: 4
- batch size: 2

Inference

I load the last model, which is trained after 4 epochs, to make inference on the test data. I concatenate the article, question and choices, pass through the tokenizer and unsqueeze them to match the shape of a batched input. After putting them into the model, get the result by finding the argument with the largest value. Finally, add 1 to the results and write in a csv file. (labels range from 1 to 4)

2 Problems of Preprocessing Data

Since the number of choices are different in each question, in order to form a batch as the input of the model, I add empty strings to make sure that all of them have the same number of choices. Then, sort the data by the length of the article to avoid forming a batch with any two articles that have a huge difference in length, which will lead to unnecessary padding.

I have encounter the problem that the sequence length is larger than the model input. Using tokenizer from HuggingFace with truncation='only_first' will perform automatically truncation on the first sequence in the input.

3 Difficulties

GPU RAM

My training environment is Google Colab, which has a limited GPU resource. While setting a batch size larger than 4, it will overload the RAM of the GPU. Thus, I can only test the batch size smaller than 4.

Validation loss

While in the training process, although the training loss kept decreasing, the validation loss start increasing at the very beginning of the training part. However, it doesn't mean that the fine-tune process is wrong. The reason might be that while fine-tuning a huge model like BERT, it is hard to really make the validation loss keep decreasing.