EE-559 Deep learning – Practice 5, Students' questions

A question is denoted by **Q**; the corresponding answer is denoted by **A**. The questions-related exercises are marked by their numbers in Practice_5.pdf and Practice_5.ipynb documents.

Data Processing and Tokenization

Q: Why are all values in token_type_ids equal to zero in exercise 5.1?
A: Some models are designed for tasks such as question answering. This requires "two different sequences to be joined in a single input_ids entry, which usually is performed with the help of special tokens, such as the classifier ([CLS]) and separator ([SEP]) tokens (example of paired sequences: [CLS] HuggingFace is based in NYC [SEP] Where is HuggingFace based? [SEP]). This is enough for some models to understand where one sequence ends and where another begins. However, other models, such as BERT, also deploy token type IDs (also called segment IDs). They are represented as a binary mask identifying the two types of sequence in the model." The above example of paired sequences will be represented with the following token_type_ids: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]. "The first sequence, the context used for the question, has all its tokens represented by a 0, whereas the second sequence, corresponding to the question, has all its tokens represented by a 1." [source]

In the classification task of exercise 5, there is only one type of sequence as the input sequence is not paired. Thus the token_type_ids are all zeros.

Q: How does the .map() function in 5.1 work? How to see tokens rather than token IDs?

A: The imdb is a Hugging Face DatasetDict object. It is similar to a dictionary structure, where the keys could be the "train" and "test", and the values are the Dataset objects [link]. The map() function applies the transformation to the specified dataset. This function is available in both DatasetDict and Dataset objects. The map function can (1) change the existing values of the features in the Dataset, or (2) append the new ones.

For example, the code below will add a prefix to all strings which are stored in 'text', illustrating (1).

```python
cache_files_2 = {
    "train": "~/.cache/imdb/imdb_train_modified.arrow",
    "test": "~/.cache/imdb/imdb_test_modified.arrow"
}
def add_prefix(example):
    example["text"] = "NEW TEXT " + example["text"]
    return example
imdb_modified = imdb.map(add_prefix, cache_file_names=cache_files_2)
print(imdb_modified)
```

Now consider the tokenize_function() provided in the Practice 5 notebook.

```python
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length",
        truncation=True)
```

Here, the function returns the output of the tokenizer function, which is a dictionary with the following keys: 'input_ids', 'token_type_ids', 'attention_mask'. In this case, the features are appended to the Dataset object, illustrating (2).

To obtain the tokens, you can use the following code:

```python
cache_files_3 = {
    "train": "~/.cache/imdb/imdb_train_tokenized_tokens.arrow",
    "test": "~/.cache/imdb/imdb_test_tokenized_tokens.arrow"
}
def obtain_tokens(example):
    return {"tokens": tokenizer.convert_ids_to_tokens(example['
        input_ids'])}

tokenized_imdb_tokens = tokenized_imdb.map(obtain_tokens,
   cache_file_names=cache_files_3)
print(tokenized_imdb_tokens)
```

You can check the dictionary for the bert_uncased_L-2_H-128_A-2 used in Practice 5 here.

Model and optimizer

Q: For the learning rate scheduler in exercise 5.2, why do we use num_warmup_steps=1?
A: Setting num_warmup_steps=1 means the learning rate increases from zero to the base value in num_warmup_steps=1 step before linearly decaying. Since the dataset is small, here we use a small value as an example. You can experiment with higher values of num_warmup_steps to evaluate its impact on model performance and convergence.

Additional useful material

- tips and tricks for training transformers can be found in this article
- papers addressing the complexity of transformers:

  Linformer `https://arxiv.org/pdf/2006.04768.pdf`

  Reformer `https://arxiv.org/pdf/2001.04451.pdf`
- working with long sequences: Longformer `https://arxiv.org/abs/2004.05150`