ADL HW1 Report

Q1: Data processing

Tokenizer:

The Bert tokenizer uses WordPiece tokenization, which is a subword tokenization technique. It breaks down text into smaller units, providing the following benefits:

1. **Avoiding Unknown Words:** When processing text, we may encounter previously unseen words or similar words that could pose challenges for the model. WordPiece tokenization allows the model to handle such cases by breaking words down into smaller subwords, enabling it to understand previously unseen terms.
2. **Better Capturing Language Structure**: WordPiece tokenization helps the model capture language structure more effectively. By breaking words into smaller subwords, it makes it easier for the model to comprehend the grammar and structure of the text.

The WordPiece algorithm involves a series of steps, starting with defining the vocabulary size, then breaking words into subwords. Subsequently, a language model is built based on subword data, and the subwords are chosen based on maximum likelihood, repeating this process until a certain threshold is reached.

Answer Span:

- How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
    1. The code uses the `char_to_token` method to find the token positions that correspond to the character positions of the answer span within the tokenized paragraph.

2. It adjusts these token positions to account for the added tokens (e.g., [CLS], [SEP]) in the tokenized question and paragraph.
3. The adjusted token positions are used to define the answer span within the tokenized input for training the model.

- After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?
  1. Find the most probable start and end positions for each window.
  2. Select the window with the highest combined probability as the source of the answer.
  3. Adjust positions to account for special tokens.
  4. Handle [UNK] tokens by recovering from context.
  5. Ensure correct formation of quotes in the answer.

Q2: Modeling with BERTs and their variants

Bert

- Your model :
  - Paragraph Selection (hfl/chinese-bert-wwm-ext)


```
{
  "_name_or_path": "hfl/chinese-bert-wwm-ext",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
```

```
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


- Span selection (hfl/chinese-bert-wwm-ext)


```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
```

```
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


- Performance:
  - Paragraph Selection eval: 0.9568
  - Span Selection EM: 0.765
  - Public score : 0.76401
- Loss function:
  - cross-entropy loss
- Training argument
  - Paragraph Selection:
    - Optimizer: AdamW

        - Learning rate: 3e-5
        - Batch size: 8
        - Gradient accumulation: 4
    - Span Selection:
        - Optimizer: AdamW
        - Learning rate: 3e-5
        - Batch size: 1
        - Gradient accumulation: 4

RoBERTa

- Your model:
    - Paragraph Selection (hfl/chinese-roberta-wwm-ext)


```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
```

```
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


- Your model:
  - Span Selection (hfl/chinese-roberta-wwm-ext)


```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext-large",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
```

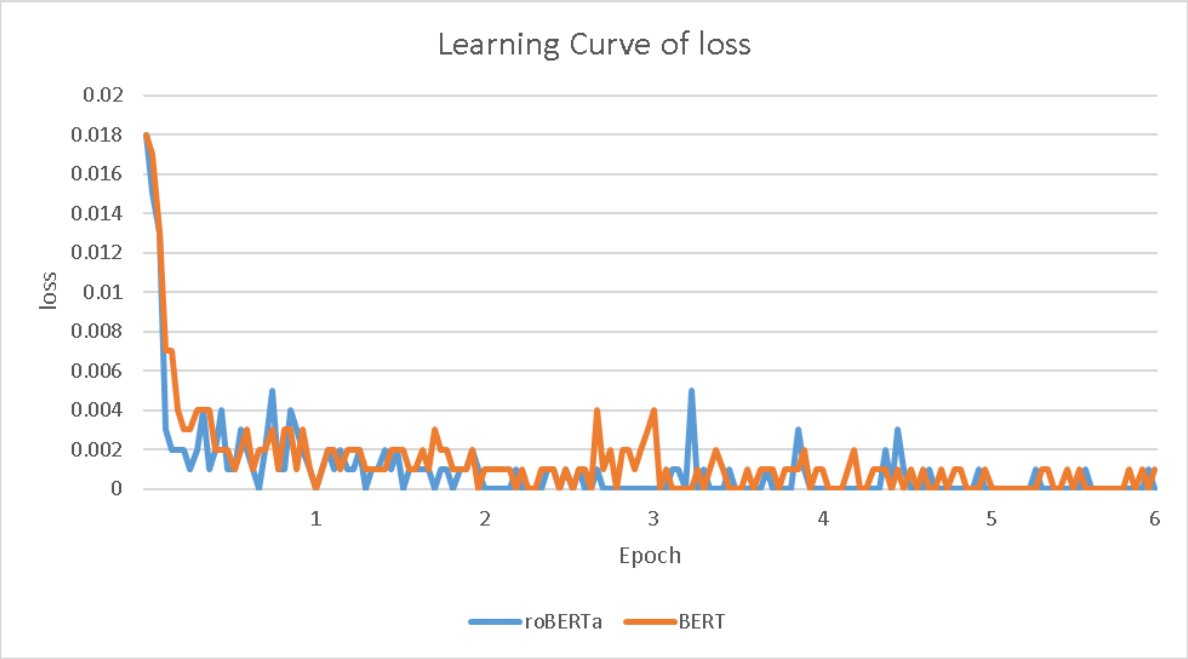
```
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


- Performance:
  - Paragraph Selection eval: 0.9644
  - Span Selection EM: 0.808
  - Public score: 0.78300
- Loss function:
  - cross-entropy loss
- Training argument
  - Paragraph Selection:
    - Optimizer: AdamW
    - Learning rate: 3e-5
    - Batch size: 8
    - Gradient accumulation: 4

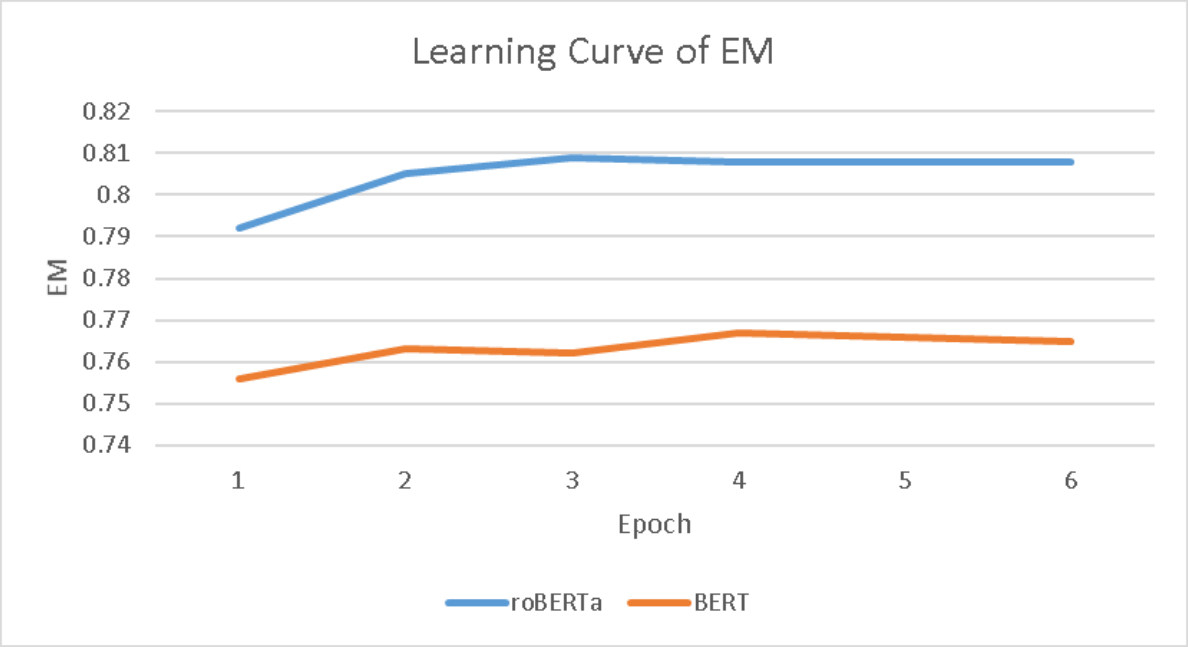
    - Span Selection:
        - Optimizer: AdamW
        - Learning rate: 3e-5
        - Batch size: 8
        - Gradient accumulation: 4
- The difference between pre-trained LMs (architecture, pretraining loss, etc.)
    - BERT:
        - BERT uses two main pretraining objectives - the MLM(masked language model) objective (predicting masked words) and the NSP objective (predicting whether two sentences are contiguous or not).
        - BERT's training is bidirectional but masks out random words in the input. It uses a segment embedding to distinguish between sentences in the input.
    - RoBERTa:
        - RoBERTa primarily focuses on the MLM objective but removes the NSP(next sentence prediction) task, which makes the training process simpler and more efficient. It also uses larger mini-batches and trains on more text data, resulting in better performance.
        - RoBERTa uses dynamic masking during training, and it also uses a more extensive range of masking. It drops the segment embedding used in BERT, and the input is treated as one continuous stream of text.

Q3: Curves

- Plot the learning curve of your span selection
    - Learning Curves of loss



- Learning Curves of EM



Q4: Pre-trained vs Not Pre-trained

- Changing the "Span Selection" model to a not pre-trained model with the same configuration, while keeping the "Paragraph Selection" part the same as the "chinese-bert-wwm-ext":

```
{
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


- Performance

	Bert	Not Pre-trained model
EM	0.765	0.061
Public score	0.76401	0.00723

- Compare

As we can see, the performance of not pre-trained model is so much worser, with only 7.2% accuracy. The large-scale and comprehensive pretraining that pretrained models undergo provides them with a significant advantage in natural language understanding tasks compared to training models from scratch, which typically struggle to attain the same level of performance.