HW1

ADL HW1 Report

ID: R12922192

Name: 邱冠珅

Q1: Data processing

Tokenizer:

▼ Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.

I use BERT tokenizer and it will first separate the words in a sentence and then map the separated words to the vocabulary for encoding.

Initially, when segmenting the characters in the text, BERT treats different languages differently. For Chinese, it takes out each character one by one, while for English, a word is composed of many alphabets and subword tokenization is adopted to segment it.

ex: 深度學習之應用 is tokenized into [’深’, ‘’度, ‘學’, ‘習’, ‘之’, ‘應‘, ‘用’] . Where is Himalayas in the world map is tokenized into [”where”, “is”, “himalayas”, “in”, “the”, “world”].

After segmenting the text, the tokenizer also adds five special tokens:

1. **[CLS] (Classification)**: This special token is typically added at the beginning of each input sequence for classification tasks, indicating the start of processing and generating predictions for classification tasks.
2. **[SEP] (Separation)**: Used to separate different sentences or paragraphs, helping the model understand the text's structure.
3. **[UNK] (Unknown)**: Used to represent unknown words or tokens. When the model encounters words not in the vocabulary, it is represented as "[UNK]."

4. **[PAD] (Padding)**: Used for padding input sequences to ensure they have the same length for batch processing.
5. **[MASK] (Mask)**: Used in the training process of BERT for predicting positions in a masked language modeling task, helping the model learn to handle missing information.

Answer Span:

▼ How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?

When using the Tokenizer with return_offset_mapping, it will return (char_start, char_end). Iterate through the offset mappings, and when span_start matches char_start, that's the start position. When span_end matches char_end, that's the end position.

▼ After your model predicts the probability of answer span start/end positions, what rules did you apply to determine the final start/end position?

Use post-processing function to eliminate invalid answers. Afterward, iterate through all probabilities and select the highest (start, end), which becomes the final start/end position.

Q2: Modeling with BERTS and their variants

Describe:

train two models, one for paragraph selection and the other for span selection, both based on the "bert-base-chinese" pre-trained model.

▼ Model


```
{
  "_name_or_path": "bert-base-chinese",
  "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,
  "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": "4.34.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_name_or_path": "bert-base-chinese",
  "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,
  "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": "4.34.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


▼ Performance

- paragraph selection

Data	Accuracy
validation	0.9614

- span selection

Data	EM
validation	0.7856
public	0.7495
private	0.73351

▼ Loss function

- Paragraph Selection: Cross Entropy
- Span Selecction: Cross Entropy

▼ Optimization algorithm, Learning rate and Batch size

- Optimization algorithm: AdamW
- Learning rate: 1e-5
- Learning schedule: Linear
- Batch size: 8(2 training batch size * 4 accumulation steps)
- Max sequence length: 512

Try another type of pre-trained LMs and describe:

train two models, one for paragraph selection and the other for span selection, both based on the “chinese-macbert-large" pre-trained model

▼ Model

```
{
  "_name_or_path": "hfl/chinese-macbert-large",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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": "4.34.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```



```
{
  "_name_or_path": "hfl/chinese-macbert-large",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
```

```
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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": "4.34.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


▼ Performance

- paragraph selection

Data	Accuracy
validation	0.969

- span selection

Data	EM
validation	0.829
public	0.806
private	0.785

▼ Difference between pre-trained LMs

MacBERT is an improved BERT with novel MLM as correction pre-training task, which mitigates the discrepancy of pre-training and fine-tuning in BERT.

Instead of masking with [MASK] token, which never appears in the fine-tuning stage, **MACBERT** propose to use **similar words** for the masking purpose. A similar word is obtained by using Synonyms toolkit, which is based on word2vec similarity calculations. If an N-gram is selected to mask, we will find similar words individually. In rare cases, when there is no similar word, MACBERT will degrade to use random word replacement.

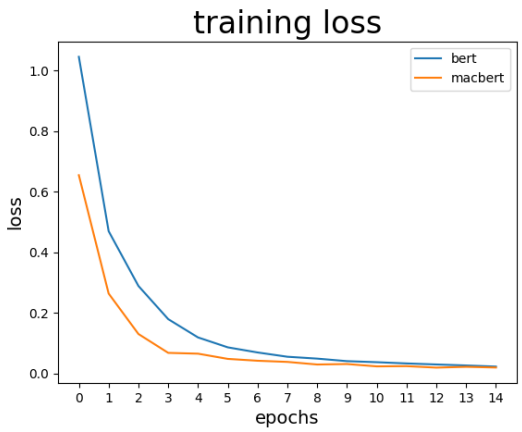
Here is an example of MACBERT pre-training task.

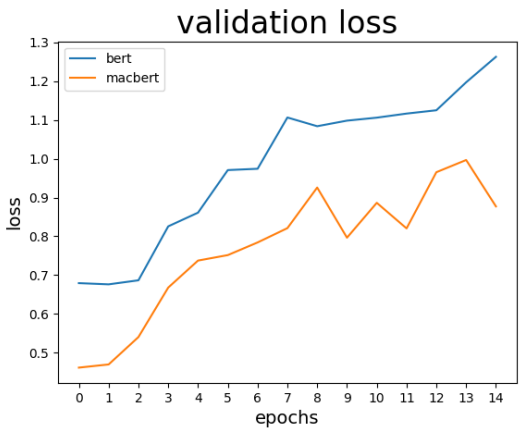
	Example
Original Sentence	we use a language model to predict the probability of the next word.
MLM(bert)	we use a language [M] to [M] ##di ##ct the pro [M] ##bility of the next word .
Whole word masking	we use a language [M] to [M] [M] [M] the [M] [M] [M] of the next word .
N-gram masking	we use a [M] [M] to [M] [M] [M] the [M] [M] [M] [M] [M] next word .
MLM as correction	we use a text system to ca ##lc ##ulate the po ##si ##bility of the next word .

With MLM as correction mitigates the discrepancy of pre-training and fine-tuning MACBERT achieve higher accuracy than BERT.

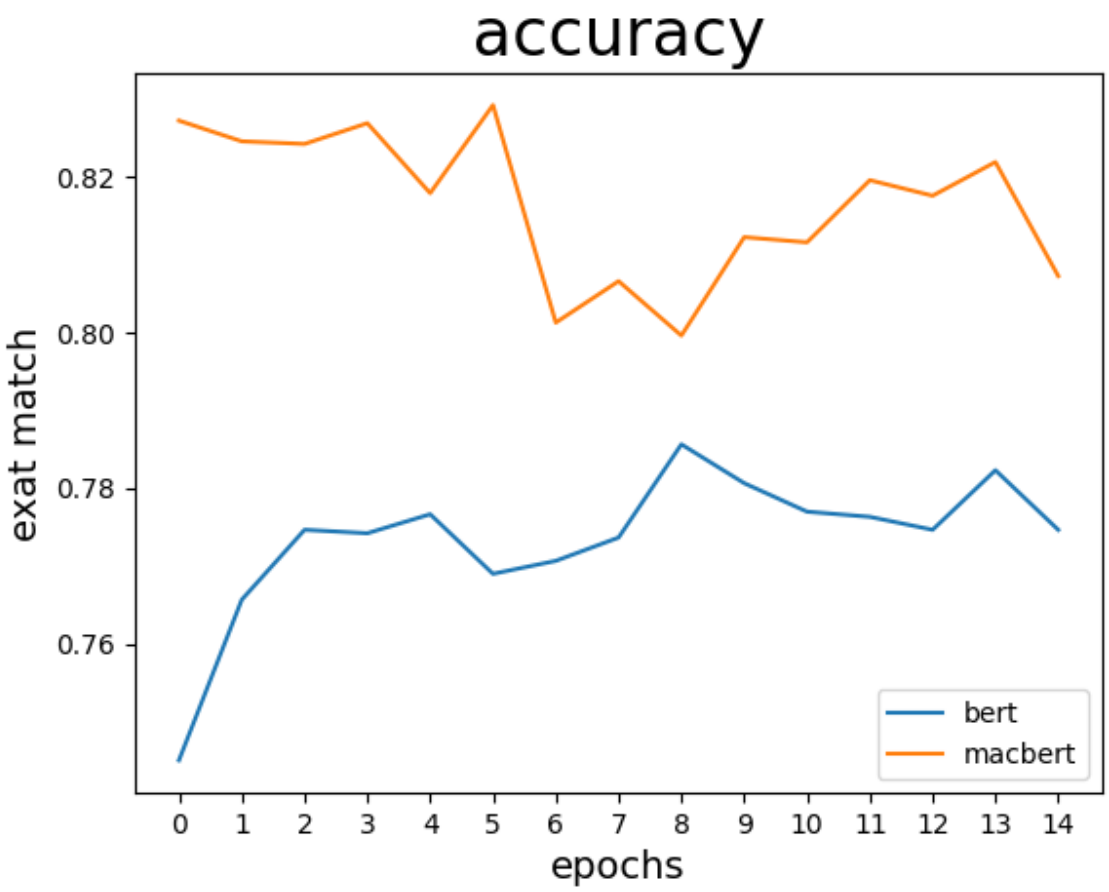
Q3: Curves

Loss curves:





Exact Match curves:



Q4: pre-trained vs Not Pre-trained

Train two transformer-based model one for paragraph selection and the other for span selection from scratch

Describe

▼ model

paragraph selection: bert-base-chinese(without pretrain only architecture)

span selection: bert-base-chinese(without pretrain only architecture)

```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [ß
    "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,
  "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": "4.34.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_name_or_path": "bert-base-chinese",
  "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,
  "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": "4.34.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


▼ hyper_parameters

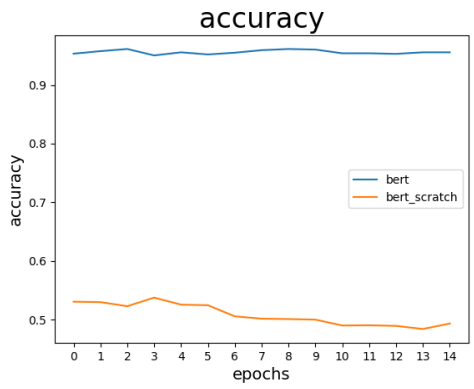
identical to the bert-base-chinese in Q2

- Optimization algorithm: AdamW
- Learning rate: 1e-5
- Learning schedule: Linear
- Batch size: 8(2 training batch size * 4 accumulation steps)
- Max sequence length: 512

Performance

In the two charts below, the blue line represents the performance of the pretrained BERT Base Chinese model, while the orange line represents the performance of the scratch (non-pretrained) version of the BERT Base Chinese model. It can be observed that with the assistance of pretrain, the performance is significantly higher.

▼ paragraph selection (validation dataset)



▼ span selection (validation dataset)