# ADL hw1

b10902044

October 23, 2023

## Q1: Data processing

1. Tokenizer

   (a) Character Segmentation:
   split the input text into individual character and consider them as basic units.

   (b) Word Segmentation:
   segment Chinese words(composed of multiple characters) into subword tokens

   (c) special tokens

      i. CLS token is usually added at the beginning of the input

      ii. SEP token is used to separate segments of text in tasks like question-answering or text classification

      iii. UNK to represent out-of-vocabulary terms

2. Answer Span

   (a) iterate through the list of tokens and determine which tokens cover the character positions of the answer span.
   Start with the start_char position. Check which token covers this character position, if the character position is within the boundaries of a token, consider that token as the starting token. Repeat the same process for the end_char position to find the ending token.

   (b) The BERT model predicts the probability distribution for all token positions for the start and end of the answer span by multiplying the probability of start and end for the pair (start, end).
   To determine the start and end position of answer span, choose the token position with the highest probability and also checking start $\leq$ end

## Q2: Modeling with BERTs and their variants

1. Model with bert

   (a) paragraph selection

```
{
  "_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": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


(b) span selection


```
{
  "_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": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


(c) performance:

i. phase1: validation accuracy: 0.9564639415088069

ii. phase2: validation exact match: 79.76071784646062

(d) loss function: cross entropy loss

(e) optimization algorithm: adamw

(f) Hyperparameter

i. phase1:
learning rate: 2e-5
batch size: 2

ii. phase2:
learning rate: 1e-5
batch size: 4

2. Model with shibing624/text2vec-base-chinese

(a) paragraph selection


```
{
  "_name_or_path": "shibing624/text2vec-base-chinese",
  "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": 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": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


(b) span selection


```
{
  "_name_or_path": "shibing624/text2vec-base-chinese",
  "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": 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": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
```

```
}
```

(c) performance:

i. phase1: validation accuracy: 0.9651046859421735

ii. phase2: validation exact match: 80.62479228979727

(d) Hyperparameter

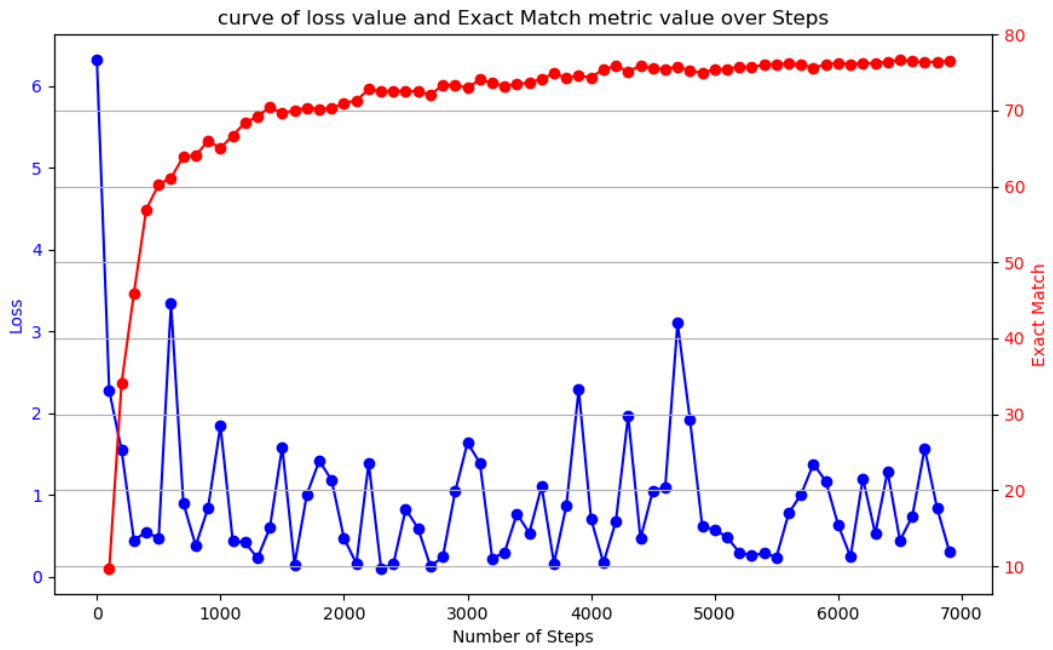
i. phase1:
learning rate: 2e-5
batch size: 2

ii. phase2:
learning rate: 1e-5
batch size: 4

3. difference between pre-trained LMs
shibing624/text2vec-base-chinese is a pre-trained CoSENT model that enhances BERT. It achieves this by substituting the initial cosine similarity loss function with an alternative loss function that is more forgiving towards negative example pairs. The core concept here is that words with low similarity aren't necessarily completely unrelated. Consequently, the new loss function does not rigidly enforce a high dissimilarity(-1) for negative example pairs.

## Q3: Curves

Learning curve of the loss value and the Exact Match metric value
model: span selection model with bert



## Q4: Pre-trained vs Not Pre-trained

1. configuration of the model


```
{
    "_name_or_path": "model_bert_notrain_p2/",
    "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": "4.34.0",
    "type_vocab_size": 2,
    "use_cache": true,
    "vocab_size": 21128
}
```


2. how to train this model:
   first train the model on question answering with bert and epoch 0 to get the not pre-trained model, then use the not pre-trained model just get from previous step to train with data and validation

```
python run_qa_no_trainer.py --train_file converted_train.json --validation_file
    converted_valid.json --test_file converted_test.json --do_predict --
    per_device_train_batch_size 4 --per_device_eval_batch_size 1 --learning_rate 1e-5
    --num_train_epochs 0 --max_seq_length 512 --output_dir model_bert_notrain_p2/ --
    model_name_or_path bert-base-chinese

python run_qa_no_trainer.py --train_file converted_train.json --validation_file
    converted_valid.json --test_file converted_test.json --do_predict --
    per_device_train_batch_size 4 --per_device_eval_batch_size 1 --learning_rate 1e-5
    --num_train_epochs 1 --max_seq_length 512 --config_name model_bert_notrain_p2/ --
    tokenizer_name model_bert_notrain_p2/ --output model_bert_notrain_final
```

3. performance:

   (a) pre-trained(model with bert in q2):
   validation exact match: 79.76071784646062

   (b) not pre-trained:
   validation exact match: 3.3233632436025258

   there is a clear difference of performance between two model, i think that means the pre-trained model play a crucial role in this kind task, it should be difficult to have proper performance by only using the train data we get without the help of pre-trained model because the size of our train data is much less compared the data use to train this pre-trained model.

## Q5: Bonus

1. model:
   i use the span selection model to train this end-to-end transformer-based model, by modifying the train data to combine the four paragraph(eg. each paragraph remain first 512 characters and combine to one paragraph with 2048 characters) and change the hyperparameter –max_seq_length 2048(or other number that can have better performance)

2. performance:
   need more time to train and the performance still worse than the bert model with paragraph selection + span selection pipeline approach because it can be affect by other three incorrect paragraphs and need to watch more characters per data.

3. loss function, optimization algorithm, learning rate and batch size:
   same as the model used in Q2 span selection model with bert.