ADL HW 2

Q1: Data processing

1. Tokenizer

   a. Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.
   I use the tokenizer provided along with the original pre-trained model. The tokenizer applies WordPiece Tokenization, which may break up a word into smaller subwords. We may see prefix(##) in some characters of vocabulary, which indicates they are concatenated after other characters. It can help to acknowledge new words that are not in the dictionary.
   The tokenization algorithm is shown as follows:

      i. Identify the subword vocabulary size.
      ii. Choose a word and split it into subwords.
      iii. Choose the sequence of subwords with the best score, and add the subwords into the vocabulary.
      iv. Repeat from step ii. until the subword vocabulary size is reached.

2. Answer Span

   a. How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
   The tokenizer can get the mappings from token to character start/end position with return_offsets_mapping. We may iterate through the tokens. Once we find a token with the character start position identical to the answer span start position, it is the start token position; once we find a token with the character end position identical to the answer span end position, it is the end token position

   b. After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?
   Clear unreasonable (start, end) pairs, e.g. start_position ¿ end_position, the answer span is out of the context scope. Then pair up the reasonable (start, end) pair and calculate the pairing score (Apply addition for start and end logits). Choose the pair with the highest pairing score as the final start/end position.

Q2: Modeling with BERTs and their variants

1. Describe

   a. your model (configuration of the transformer 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",
"transformers_version": "4.22.2",
"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",
"transformers_version": "4.22.2",
"type_vocab_size": 2,
"use_cache": true,
"vocab_size": 21128
}
```


   b. performance of your model.
   Multiple Choice: 94.65, Question Answering: 75.68

   c. the loss function you used.
   Cross Entropy Loss

   d. The optimization algorithm (e.g. Adam), learning rate and batch size.
   multiple-choice: AdamW, 5e-5, question-answering: AdamW, 3e-5,

2. Try another type of pretrained model and describe

   a. your model
   hfl/chinese-roberta-wwm-ext

   b. performance of your model
   Multiple Choice: 95.01, Question Answering: 81.98

   c. the difference between pretrained model (architecture, pretraining loss, etc.)
   According to the repo, the roberta pretrained model has the following differences:

      i. During pre-training, the masking is based on the Whole Word Masking(WWM) strategy, but dynamic masking is not applied.

ii. Next Sentence Prediction(NSP) loss is removed because it seems to be ineffective.

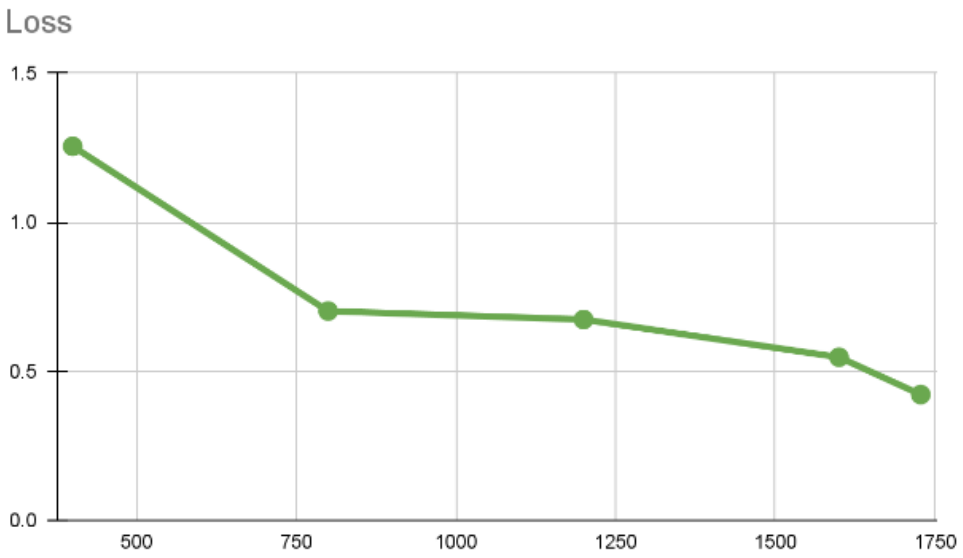
iii. It was trained with max_len 512 directly without training with max_len 128 first.

iv. The training steps are lengthened appropriately.

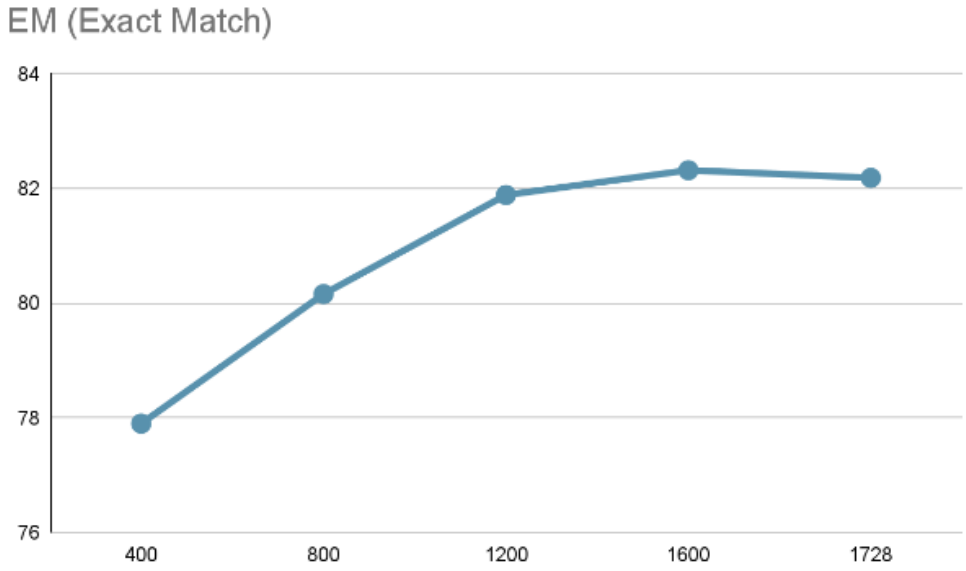
Q3: Curves

1. Plot the learning curve of your QA model

   a. Learning curve of Loss



   b. Learning curve of EM



Q4: Pretrained vs Not Pretrained

Train a transformer model from scratch (without pretrained weights) on the dataset (you can choose either MC or QA)
Describe

1. The configuration of the model and how do you train this model
   The configuration I use is the same as the original pre-trained model.


```
  "_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",
  "transformers_version": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


   In the original sample code, the model will be loaded with the following code segment:

```
model = AutoModelForQuestionAnswering.from_pretrained(
    model_args.model_name_or_path,
    from_tf=bool(".ckpt" in model_args.model_name_or_path),
    config=config,
    cache_dir=model_args.cache_dir,
    revision=model_args.model_revision,
    use_auth_token=True if model_args.use_auth_token else None,
)
```

   To create a model from scratch, I use the following code instead:

```
model = AutoModelForQuestionAnswering.from_config(config)
```

2. The performance of this model v.s. BERT
   The model without pre-trained weight does not attain reasonable performance the EM is about 4.95, which is far less than the bert-base-chinese model 75.68.

Q5: Bonus: HW1 with BERTs

Train a BERT-based model on HW1 dataset and describe your model performance of your model.

1. Intent classification
   eval_accuracy: 0.9597,
   eval_loss: 0.3303,

2. Slot tagging
   eval_accuracy: 0.9715,
   eval_f1: 0.8267,
   eval_loss: 0.0877,
   eval_precision: 0.8208,
   eval_recall: 0.8325,

3. The loss function you used.
   Both: Cross Entropy Loss

4. The optimization algorithm (e.g. Adam), learning rate and batch size.
   Intent classification: AdamW, 2e-5, 32
   Slot tagging: AdamW, 5e-5, 8