ADL HW1 Report

Student ID: m11203404
Name: 陳旭霖

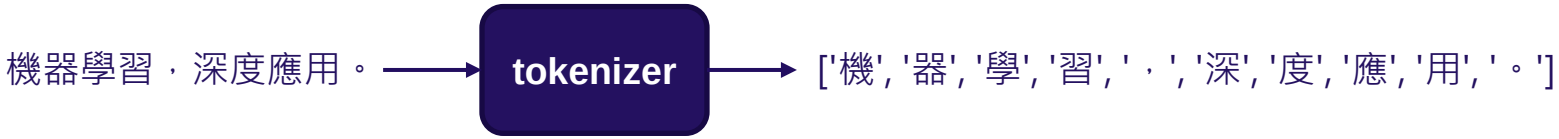
Q1: Data processing

- Tokenizer

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

A: The tokenizer I am using here is "**bert-base-Chinese**". Its algorithm is based on character-based tokenization, where each character in Chinese is treated as an individual token.

➢ Workflow



➢ Details:

1. [CLS]: This token represents the starting point of the entire text.
2. [SEP]: Inserting this token in the middle of two sentences.
3. [UNK]: Words that do not appear in the bert-base-Chinese dictionary will be replaced by this token.
4. [PAD]: If the sentences in the same batch have different lengths, this token will be used to fill the shorter ones.

Q1: Data processing

- Answer Span

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

A: First, locate the starting and ending character indices of the answer in the text. Then, find the **token_start_index** and **token_end_index** within the current range of text. Finally, move the **token_start_index** and **token_end_index** to the two ends of the answer.

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

A: First, tokenize my question and the content. Then, input into the model and get the logits. Finally, obtain the indices of the highest probability for the starting and ending positions from the output logits.

Q2: Modeling with BERTs and their variants

- Describe

Q: Your model.

A: I used the **bert-base-chinese** pre-trained model for both the multiple-choice and question-answering tasks. Its structure is the same as the original BERT, but it was pre-trained on Chinese datasets.

➢ **Crucial "Model" Configurations**

1. **Backbone:** BERT
2. **Activation:** gelu
3. **hidden_size:** 768
4. **num_hidden_layers:** 12
5. **num_attention_heads:** 12
6. **Vocab_size:** 21128

➢ **Crucial "Tokenizer" Configurations**

1. **Backbone:** BERT
2. **model_max_length:** 512
3. **cls_token:** [CLS]
4. **mask_token:** [MASK]
5. **pad_token:** [PAD]
6. **sep_token:** [SEP]
7. **unk_token:** [UNK]

Q2: Modeling with BERTs and their variants

- Describe

Q: The performance of your model.

A: →

	EM
Valid	0.791
Public	0.708

Q: The loss function you used.

A: Cross-entropy

Q: The optimization algorithm (e.g. Adam), learning rate and batch size.

A:

1. optimizer: AdamW
2. lr: 0.00003
3. batch_size: 8
4. epoch: 1
5. max_len: 512

Q2: Modeling with BERTs and their variants

- Try another type of pre-trained LMs and describe

Q: Your model.

A: →

	Model name
Paragraph	chinese-macbert-base
QA	chinese-lert-large

Q: The performance of your model.

A: →

	EM
Valid	0.960 (combine Train+Valid)
Public	0.785

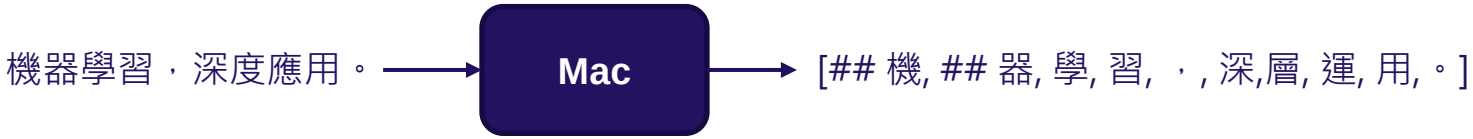
Q2: Modeling with BERTs and their variants

- Try another type of pre-trained LMs and describe

Q: The difference between pre-trained LMs.

A: chinese-macbert-base→ To address the issue of inconsistency between pre-training and downstream tasks, the authors of the paper replaced masked tokens in pre-training with similar words. Text similarity was computed using word2vec. When the similarity of the nearest words fell below a certain threshold, words were randomly substituted.

➢ MLM as correction (Mac) example:



Q: The difference between pre-trained LMs.

A: chinese-lert-large→ The author proposes learning linguistic features during pre-training of the model to enable the model to grasp the deep relationships within the language. This is achieved through training based on three types of language features and the completion of the original MLM pre-training task.

Q3: Curves

- Try another type of pre-trained LMs and describe

  Q: Learning curve of the loss value

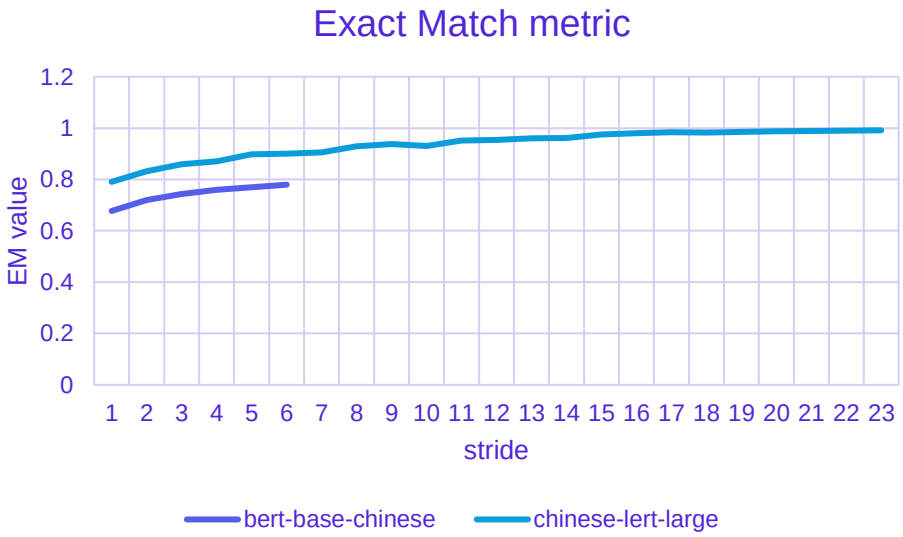
  A: stride = 500 steps, combine train+valid, test on valid

Q3: Curves

- Try another type of pre-trained LMs and describe

  Q: Learning curve of the Exact Match metric value

  A: stride = 500 steps, combine train+valid, test on valid



Q4: Pre-trained vs Not Pre-trained

- Train a transformer-based model from scratch

Q: The configuration of the model and how do you train this model.

A: I use the same 'bert-base-chinese' model as in Q2, and the configurations are also the same as Q2.

➢ **Crucial "Model" Configurations**

1. **Backbone:** BERT
2. **Activation:** gelu
3. **hidden_size:** 768
4. **num_hidden_layers:** 12
5. **num_attention_heads:** 12
6. **Vocab_size:** 21128

➢ **Crucial "Tokenizer" Configurations**

1. **Backbone:** BERT
2. **model_max_length:** 512
3. **cls_token:** [CLS]
4. **mask_token:** [MASK]
5. **pad_token:** [PAD]
6. **sep_token:** [SEP]
7. **unk_token:** [UNK]

Q4: Pre-trained vs Not Pre-trained

- Train a transformer-based model from scratch

Q: The performance of this model v.s. BERT.

A: After training using the 'from scratch' approach, it became apparent that more extensive training time is required to achieve better performance. However, when applied to the public domain, the EM (Expectation-Maximization) performance was exceptionally poor, indicating severe overfitting. The reason could possibly be attributed to the limited size of the training dataset.

	EM on validation set
Pre-trained	0.791
From scratch (Epoch 1)	0.677
From scratch (Epoch 15)	0.994

	EM on public
Pre-trained	0.708
From scratch (Epoch 15)	0.078

Q4: Pre-trained vs Not Pre-trained

- Train a transformer-based model from scratch

  Q: The performance of this model v.s. BERT.

  A: After training using the 'from scratch' approach, it became apparent that more extensive training time is required to achieve better performance. However, when applied to the public domain, the EM (Expectation-Maximization) performance was exceptionally poor, indicating severe overfitting. The reason could possibly be attributed to the limited size of the training dataset.