ADL Homework 1: Question Answering

Department: CSIE Student ID: d11922016 Name: Jia-Wei, Liao

October 23, 2023

1 Data processing

1.1 Tokenizer

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

Answer.

We use `bert-base-chinese` as the tokenizer in Hugging Face, which converts individual Chinese characters into numerical representations. During the conversion process, various special tokens are added to the string for different purposes. Table 1 illustrates the purposes of various special tokens, while Table 2 demonstrates how the tokenizer handles the string.

Token	Index	Function
[PAD]	0	Padding tokens with zeros to a specified length.
[UNK]	100	Replacing characters not found in the BERT dictionary.
[CLS]	101	Placed at the beginning of the sequence, the output from the final layer of the model can be regarded as the representation of the sequence.
[SEP]	102	Used as a separator between sentences.
[MASK]	103	Used as a mask for individual characters.

Table 1: Functions of special tokens.

	Example
String	“同學要簽博嗎“
Char list	[[CLS], ’ 同’, ’ 學’, ’ 要’, ’ 簽’, ’ 博’, ’ 嗎’, ’ ? ’, [SEP]]
`char_to_token`	[101, 1398, 2119, 6206, 5087, 1300, 1621, 8043, 102]

Table 2: Tokenization example.

1.2 Answer Span

- How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
- After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?

Answer.

- We employ a tokenizer to generate an offset mapping, which maps tokens to tuples representing their respective start and end positions within the original string. Notably, the tuple (0, 0) corresponds to special tokens. An illustrative example is presented in Table 3.
- After predicting the probabilities of the start and end positions, denoted as p_i^s and p_j^e respectively, we calculate the score of $p_i^s + p_j^e$ for $0 \leq j - i \leq 30$, that is

$$(i, j) = \underset{i,j}{\text{argmax}}\{p_i^s + p_j^e \mid 0 \leq j - i \leq 30\}.$$

  Finally, we utilize the offset mapping to convert the result into the prediction string.

	Example
String	“同學要簽博嗎 “
Char list	[[CLS], ’ 同’, ’ 學’, ’ 要’, ’ 簽’, ’ 博’, ’ 嗎’, ’ ? ’, [SEP]]
`char_to_token`	[101, 1398, 2119, 6206, 5087, 1300, 1621, 8043, 102]
Offset mapping	[(0, 0), (0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (0, 0)]

Table 3: Tokenization example with offset mapping.

2 Modeling with BERTs and their variants

2.1 Describe

- Your model.
- The performance of your model.
- The loss function you used.
- The optimization algorithm (e.g. Adam), learning rate and batch size.

Answer.

BERT (Bidirectional Encoder Representations from Transformers) [1] is a model introduced by Google in 2019. It leverages the encoder component of the Transformer architecture [2] and achieved state-of-the-art results in numerous tasks at that time. The CKIP Lab at Academia Sinica [1] trained BERT on a Traditional Chinese dataset, resulting in a series of pre-trained models. For the purposes of this homework, we adopted `bert-base-chinese` as our baseline model. Table 4 presents the parameters of the `bert-base-chinese` model.

[1] https://github.com/ckiplab/ckip-transformers

Parameters	Value
num_attention_heads	12
num_hidden_layers	12
pooler_fc_size	768
pooler_num_attention_heads	12
pooler_num_fc_layers	3
pooler_size_per_head	128
attention_probs_dropout_prob	0.1
hidden_act	GELU
hidden_dropout_prob	0.1
hidden_size	768
max_position_embeddings	512

Table 4: Parameters of `bert-base-chinese` model.

To circumvent potential instability in the optimization process due to smaller batch sizes, necessitated by limited memory resources, we employed gradient accumulation. This approach effectively approximates training with larger batch sizes. For optimization, we utilized the AdamW optimizer [3], which exhibits faster convergence compared to the original Adam optimizer. Additionally, we implemented a warm-up mechanism, whereby a smaller learning rate is employed during the initial stages of model training when stability is still evolving. As the model gradually stabilizes, the learning rate is progressively increased. Subsequently, we employed a cosine decay schedule to ensure stable convergence during optimization, contributing to enhanced generalization capabilities of our model. Table 5 presents the training hyperparameters.

	Multiple Choice	Question Answering
Model	`bert-base-chinese`	`bert-base-chinese`
Batch size	8	10
Epoch	5	5
Loss function	Cross-entropy	Cross-entropy
Optimizer	AdamW	AdamW
Initial learning rate	2e-5	2e-5
Weight decay	1e-5	1e-5
Learning rate schedule	cosine decay	cosine decay
Warm up step	300	300
Gradient accumulation step	4	4

Table 5: Hyperparameters for training `bert-base-chinese` models.

We present the outcomes of training `bert-base-chinese` for the Multiple Choice and Question Answering tasks over a span of 5 epochs, as depicted in Tables 6 and 7. Due to the intrinsically straightforward nature of the Multiple Choice task, the validation accuracy surpasses 0.95, while the Question Answering task exhibits a matching accuracy that approaches 0.8.

Epoch	Training		Validation	
	Loss	Accuracy	Loss	Accuracy
5	0.0012	0.9973	0.0157	0.9614

Table 6: Performance evaluation of `bert-base-chinese` model in the context of Multiple Choice.

Epoch	Training Loss	Validation Exact Match
5	0.0143	0.7983

Table 7: Performance evaluation of `bert-base-chinese` model in the context of Question Answering.

2.2 Try another type of pre-trained LMs and describe

- Your model.
- The performance of your model.
- The difference between pre-trained LMs (architecture, pretraining loss, etc.)
- For example, BERT → xlnet, or BERT → BERT-wwm-ext. You can find these models in the huggingface's Model Hub.

Answer.

RoBERTa [4], a variant of BERT, is notable for its advancements in training techniques through a series of experiments. Although a paper submitted to ICLR in 2019 was rejected due to perceived lack of novelty. It has made significant contributions. As of today, it has accumulated over 9,000 citations on Google Scholar. Furthermore, Google Research introduced the Whole Word Masking (WWM) pre-training strategy for enhancing BERT in 2019. Chinese researchers subsequently proposed `chinese-bert-wwm-ext` [5]. This variant not only confirmed the effectiveness of WWM on Chinese language data but also improved performance by refining training techniques. Table 8 and 9 presents the parameters for the `bert-base-chinese` and `bert-base-chinese` models.

Parameters	Value
num_attention_heads	12
num_hidden_layers	12
pooler_fc_size	768
pooler_num_attention_heads	12
pooler_num_fc_layers	3
pooler_size_per_head	128
attention_probs_dropout_prob	0.1
hidden_act	GELU
hidden_dropout_prob	0.1
hidden_size	768
max_position_embeddings	512

Table 8: Parameters of `chinese-bert-wwm-ext` model.

Parameters	Value
num_attention_heads	16
num_hidden_layers	24
pooler_fc_size	768
pooler_num_attention_heads	12
pooler_num_fc_layers	3
pooler_size_per_head	128
attention_probs_dropout_prob	0.1
hidden_act	GELU
hidden_dropout_prob	0.1
hidden_size	1024
max_position_embeddings	512

Table 9: Parameters of `chinese-roberta-wwm-ext-large` model.

Table 10 presents the training hyperparameters for `bert-base-chinese` and `chinese-roberta-wwm-ext-large`.

	Multiple Choice	Question Answering
Model	`chinese-bert-wwm-ext`	`chinese-roberta-wwm-ext-large`
Batch size	8	10
Epoch	5	5
Loss function	Cross-entropy	Cross-entropy
Optimizer	AdamW	AdamW
Initial learning rate	2e-5	2e-5
Weight decay	1e-5	1e-5
Learning rate schedule	cosine decay	cosine decay
Warm up step	300	300
Gradient accumulation step	4	4

Table 10: Hyperparameters for training `bert-base-chinese` and `chinese-roberta-wwm-ext-large` models.

Tables 11 and 12 present the training results for variants of the BERT model. In comparison to Tables 6 and 7, improvements are observed across all tasks, with the most notable enhancement evident in the Question Answering task.

Epoch	Training		Validation	
	Loss	Accuracy	Loss	Accuracy
5	0.0010	0.9983	0.0119	0.9688

Table 11: Performance evaluation of `chinese-bert-wwm-ext` model in the context of Multiple Choice.

Epoch	Training Loss	Validation Exact Match
5	0.0227	0.8408

Table 12: Performance evaluation of `chinese-roberta-wwm-ext-large` model in the context of Question Answering.

3 Curves

Plot the learning curve of your span selection (extractive QA) model. You can plot both the training and validation set or only one set. Please make sure there are at least 5 data points in each curve.

- Learning curve of the loss value.
- Learning curve of the Exact Match metric value.

Answer.

Figure 1 illustrates that throughout the training process of the Multiple Choice model, the validation accuracy of `chinese-bert-wwm-ext` consistently surpasses that of `bert-base-chinese`. Figure 2 displays the results for the Question Answering model, where the validation exact match score of `chinese-roberta-wwm-ext-large` consistently maintains a 5% higher performance than that of `bert-base-chinese`.

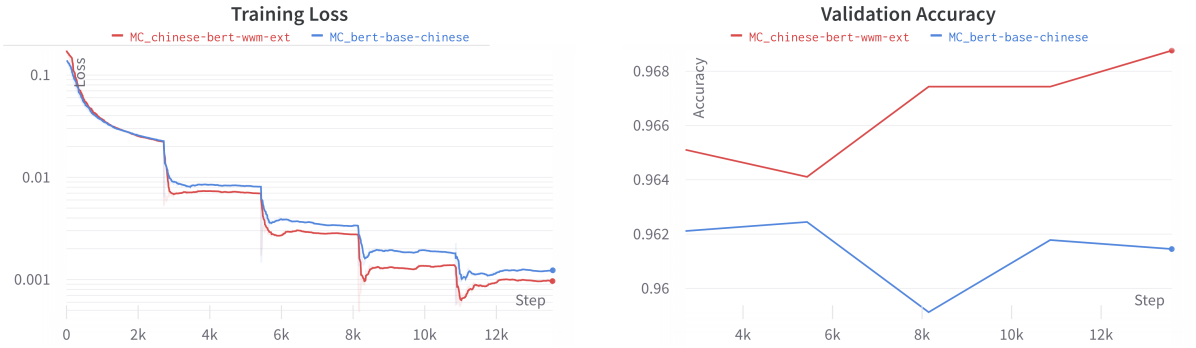

Figure 1: Loss and accuracy curves for the Multiple Choice model.

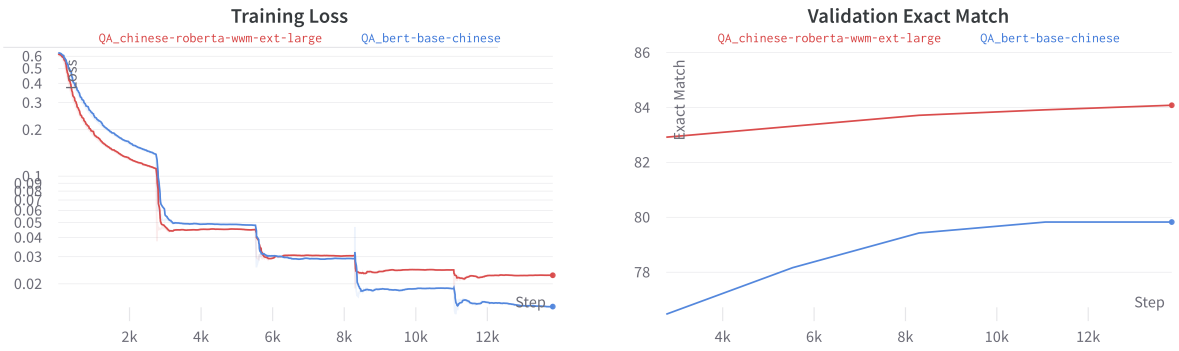

Figure 2: Loss and exact match curves for the Question Answering model.

4 Pre-trained vs Not Pre-trained

- Train a transformer-based model (you can choose either paragraph selection or span selection) from scratch (i.e. without pretrained weights).
- Describe
  - The configuration of the model and how do you train this model (e.g., hyper-parameters).
  - The performance of this model v.s. BERT.
- Hint
  - You can use the same training code, just skip the part where you load the pre-trained weights.
  - The model size configuration for BERT might be too large for this problem, if you find it hard to train a model of the same size, try to reduce model size (e.g. num_layers, hidden_dim, num_heads).

Answer.

Table 13 displays the hyperparameters of the not pre-trained model. Its parameters are identical to those selected for the pre-trained model.

Not Pre-trained	Multiple Choice	Question Answering
Batch size	8	10
Epoch	5	5
Loss function	Cross-entropy	Cross-entropy
Optimizer	AdamW	AdamW
Initial learning rate	2e-5	2e-5
Weight decay	1e-5	1e-5
Learning rate schedule	cosine decay	cosine decay
Warm up step	300	300
Gradient accumulation step	4	4

Table 13: Performance of not pre-trained BERT model.

Tables 14 and 15 present the training results for the not pre-trained BERT model. In the case of Multiple Choice, the validation accuracy is approximately 0.5, while Question Answering did not exhibit significant progress, failing to converge during training. Both tasks may necessitate extended training duration and a larger training dataset to yield noticeable training effects.

Epoch	Training		Validation	
	Loss	Accuracy	Loss	Accuracy
5	0.0897	0.6571	0.1229	0.5101

Table 14: Performance of not pre-trained Multiple Choice model.

Epoch	Training Loss	Validation Exact Match
5	0.4091	0.0499

Table 15: Performance of not pre-trained Question Answering model.

5 Summary

In this study, we conducted experiments involving various BERT models and parameters, selecting representative results for comparison. Tables 16 and 17 present a comprehensive overview of the outcomes achieved in both the Multiple Choice and Question Answering tasks. Our experiments clearly demonstrate that the utilization of pre-trained models is essential for achieving satisfactory training performance in these two tasks. Moreover, as the model's parameter count increases, there is a noticeable improvement in accuracy; however, this improvement comes at the cost of increased computational resources.

Notably, the `chinese-roberta-wwm-ext-large` model exhibits the best validation results in the Question Answering task. Leveraging this model for inference, we submitted our predictions to Kaggle, achieving the 11th position on the public leaderboard and an impressive 4th place on the private leaderboard.

Model	Training		Validation	
	Loss	Accuracy	Loss	Accuracy
Not pre-trained	0.0897	0.6571	0.1229	0.5101
bert-base-chinese	0.0012	0.9973	0.0157	0.9614
chinese-bert-wwm-ext	**0.0010**	**0.9983**	**0.0119**	**0.9688**

Table 16: Performance of all the Multiple Choice models.

Model	Training Loss	Validation EM	Public EM	Private EM
Not pretrained	0.4091	0.0499	-	-
bert-base-chinese	0.0143	0.7983	-	-
chinese-roberta-wwm-ext-large	**0.0227**	**0.8408**	**0.8074**	**0.8121**

Table 17: Performace of all the Question Answering models.

References

[1] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[5] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514, 2021.

Appendix

All training sessions were conducted on a PC running Ubuntu 20.04.2 with a 12th Gen Intel(R) Core(TM) i7-12700 CPU, equipped with a single NVIDIA GeForce RTX 4090 GPU featuring 24 GB of dedicated memory.

Environment

To set the environment, you can run this command:

```
pip install -r configs/requirements.txt
```

The packages and their respective versions are displayed as follows:

```
tqdm==4.66.1
gdown==4.7.1
torch==1.12.1
transformers==4.22.2
datasets==2.5.2
evaluate==0.4.0
wandb==0.15.12
```

Download

To download the datasets and model checkpoint, you can run the commad:

```
bash ./download.sh
```

Reproduce

To reproduce the result of inference, you can run the commad:

```
bash ./run.sh data/context.json data/test.json pred/prediction.csv
```

Training

- To train the multiple choice model, you can run the commad:

```
bash scripts/train_mc.sh
```

- To train the question answering model, you can run the commad:

```
bash scripts/train_qa.sh
```