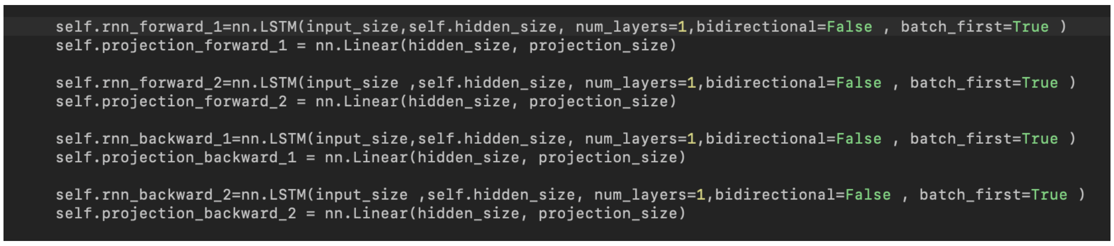
R07922118 資工碩一 翁子騰

ADL A2 Report

Q1: Describe your ELMo model. (2%)

1. **Training corpus processing. (tokenization, vocabulary building) (0.4%)**
   Tokenized_corpus.txt 中取前 100 萬句當作 training set。在每一行 sentence 前加上<BOS>，句尾加上<EOS>，然後把句子切成 max_sent_len=64。
   Vocabulary 有做兩個，一個是 word vocabulary，另一個是 character vocabulary，其中 word vocabulary 是取至少出現 3 次以上的 word，character 是取至少出現 1000 次的 character 才加入 vocabulary。
2. **Model architecture and implementation details. (0.4%)**
   第一層是 Character CNN，第二層接兩個雙層的 LSTM+Linear(一個 forward，一個 backward)，最後是接 torch.nn.AdaptiveLogSoftmaxWithLoss。
   其中第二層的詳細結構如下：(更詳細請參考程式碼 elmo.py)

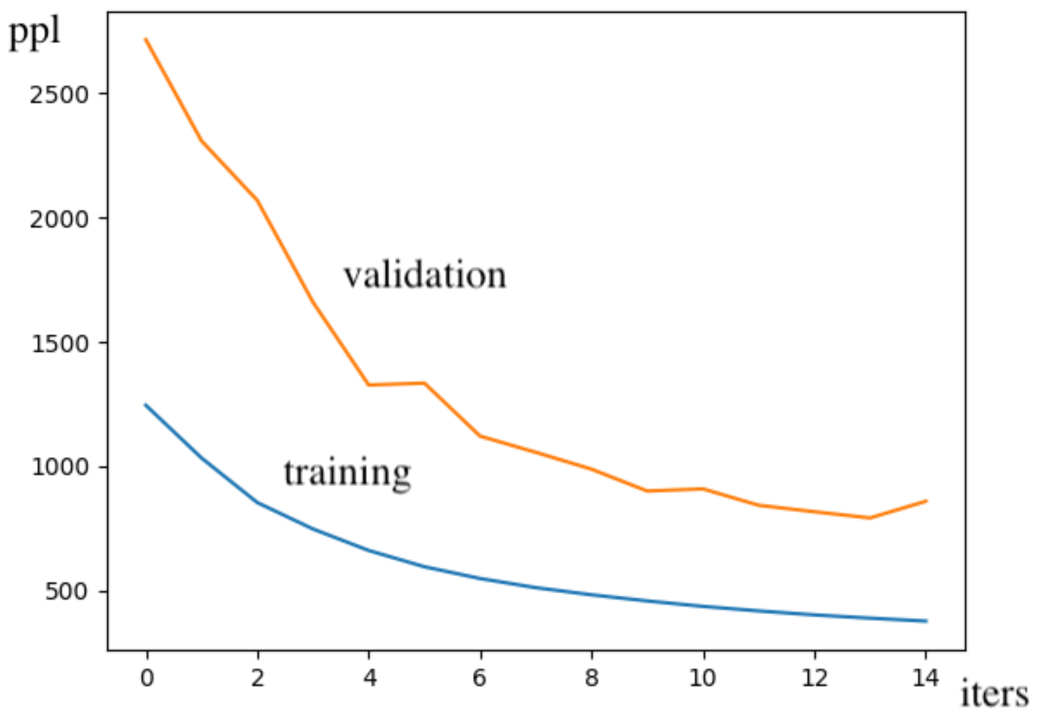


```
    self.rnn_forward_1=nn.LSTM(input_size,self.hidden_size, num_layers=1,bidirectional=False , batch_first=True )
    self.projection_forward_1 = nn.Linear(hidden_size, projection_size)

    self.rnn_forward_2=nn.LSTM(input_size ,self.hidden_size, num_layers=1,bidirectional=False , batch_first=True )
    self.projection_forward_2 = nn.Linear(hidden_size, projection_size)

    self.rnn_backward_1=nn.LSTM(input_size,self.hidden_size, num_layers=1,bidirectional=False , batch_first=True )
    self.projection_backward_1 = nn.Linear(hidden_size, projection_size)

    self.rnn_backward_2=nn.LSTM(input_size ,self.hidden_size, num_layers=1,bidirectional=False , batch_first=True )
    self.projection_backward_2 = nn.Linear(hidden_size, projection_size)
```

3. **Hyperparameters of your ELMo model. (number of layers, hidden dimension, output dimension, optimization algorithm, learning rate and batch size) (0.4%)**

Num of layers	2
Hidden dimension	2048
Output dimension	512
Optimization algorithm	Adam
Learning rate	0.001
Batch size	32

4. **Plot the perplexity score on train/dev set while training. (0.4%)**



橘線是 validation set，藍線是 training set。縱軸是 perplexity，橫軸是 num of iterations(每經過 32 個 batch 就算一次 iteration。)。

5. **Show the performance of the BCN model with and without ELMo on the public leaderboard. (0.4%)**

	BCN with ELMo	BCN without ELMo
Accuracy	0.47601	0.42986

Q2: Compare different settings for ELMo. (2%)

1. Different number of training steps. (1%)

You can train one model for large number of training steps, then take the intermediate checkpoints for this problem.

0.1 million sentences of training corpus.

all hyperparameters remain the same as in Q1-3, and I trained BCN with ELMo for 10 epochs

Epochs for ELMo	Accuracy
10	0.46877
1	0.43800

R07922118 資工碩一 翁子騰

1 million sentences of training corpus

all hyperparameters remain the same as in Q1-3, and I trained BCN with ELMo for 10 epochs

Epochs for ELMo	Accuracy
2	0.45701
1	0.47601

2. Different hyperparameters. (1%)

You can train a smaller ELMo model and compare it to your original model.

Change num_of_layer only, all the other hyperparameters remain the same as in Q1-3. And I trained BCN with ELMo for 10 epochs with 100,000 training sentences.

Num_of_layer	Epochs for ELMo	Accuracy
2	10	0.46877
1	10	0.44162

You need to report the performance of downstream task (at least the accuracy score on public leaderboard).

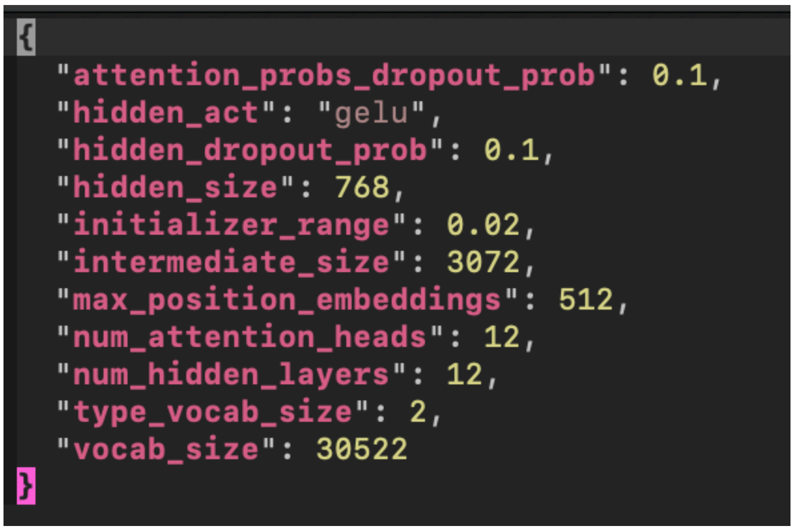
Q3: Describe your model that passes strong baseline. (1%)

1. **Input of your model. (word embedding? character embedding?) (0.4%)**

   I used BERT-base model to pass the strong baseline. The input of the model is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

2. **Model architecture. (0.4%)**

   It's a multilayer bidirectional Transformer encoder.



```
{
  "attention_probs_dropout_prob": 0.1,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "type_vocab_size": 2,
  "vocab_size": 30522
}
```

3. Hyperparameters of your model. (optimization algorithm, learning rate, batch size and other model-specific options) (0.2%)

Epochs	3
Learning rate	2e-5
Max_seq_length	64
Train_batch_size	32
Eval_batch_size	8

Q4: Describe your best model. (1%)

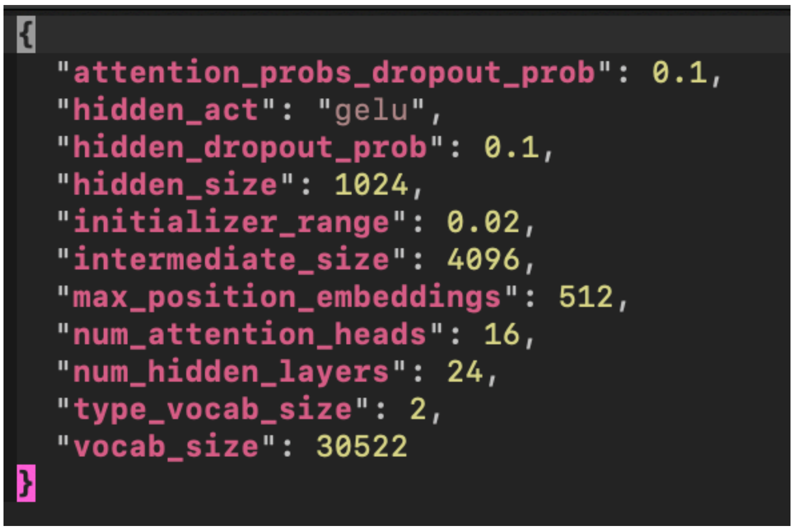
1. Describe your best model (0.5%)

a. Input to your model

I used pretrained BERT-large-uncased model to get the best accuracy. The input of the model is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

b. Model architecture

It's a multilayer bidirectional Transformer encoder



```
{
  "attention_probs_dropout_prob": 0.1,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "max_position_embeddings": 512,
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "type_vocab_size": 2,
  "vocab_size": 30522
}
```

c. Hyperparameters (optimization algorithm, learning rate, batch size and other model-specific options)

Optimization algoritm	BertAdam
Learning rate	2e-5
Epochs	3
Max seq	64
Train_batch_size	32

2. **Describe the reason you think why your best model performs better than other models. (0.5%)**

It has more parameters(340M) than BERT base(110M) and it has more layers(24 vs 12). So the contexualized embeddings produced by BERT Large can contain more information than that produced by BERT base. Thus giving higher accuracy.

Q5: Compare different input embeddings. (1%)

	Character embedding	Word embedding	Byte pair encoding
Pros	1. 所需要的 embedding 總數比較少，ex. 英文裡面可能只需要 26 個字母，10 個數字等等。如此 embedding 的維度會比較小。(Google Brain2016 發表的 Exploring the Limits of Language Modeling 只用 16 維) 2. 可以處理拼錯字，OOV 等問題	1. 使用 RNN 時訓練時間相較於 character embedding 和使用 byte pair encoding 做 subword embedding 的情形，會比較短。 2. Word embedding 可以包含 sentiment，而 character embedding 或 subword 沒辦法包含 sentiment	1. 可以用來做 subword embedding，考慮到 word 的 morphology information，可以表現一些比較少見的詞
Cons	因為把 word 分成 characters，如果使用 RNN 訓練的話，會使得訓練時間變長。	1. 沒有考慮到 word 的 morphology information，例如: “dog” 和”dogs”有不同的 word embedding。	若使用 RNN，訓練時間也會比 word embedding 長。因為把 word 切成許多 subwords。

		2. 如果 vocabulary size 很大的話，embedding 維度會很大(可能到 300 維) 3. 少見的詞容易被 map 成 OOV 而無法使用	

Q6: BERT (2%)

1. **Please describe the tasks that BERT used for pre-training. What benefits do they have comparing to language model pretraining used in ELMo? (1%)**
   BERT 在 pretrain 時用了兩個 tasks：Masked LM 和 Next Sentence Prediction。

　Masked LM 是將句子隨機選取一定比例的 token 改成[MASK]，在 BERT 論文中，市隨機選 15%的 tokens 改成[MASK]，再利用 BERT 模型去預測被 masked 掉的詞。

　Next Sentence Prediction 是在一個 corpus 中隨機選取 A, B 兩個句子(其中 A, B 兩個句子也有經過 masked)，然後 A, B 這兩個句子，有 50%機率是前後句，有 50%不是(此時隨機選 B 這個句子)，如果是前後句就標注 Label= IsNext，不是的話就標注 Label= NotNext，再利用 BERT 模型去預測 A, B 兩句是 IsNext 還是 NotNext。

　ELMo pretrain 時所用的 task 是 Language modeling。BERT 最主要的優點在於，他可以做 bidirectional training，不像 ELMo 是 left-to-right, right-to-left 兩邊獨立訓練，如此 BERT 可以同時考慮到左邊和右邊的 contexts。

2. **Please describe in detail how you would formulate the problem in HW1 and apply BERT on it. (1%)**

HW1 中，是希望能夠預測對話中的下一個句子。可以把每一段對話的最後一個句子當作 A sentence，其中在 100 個 candidate answers 中依序選 1 個當作 B sentence，concat 起來之後，利用 BERT pretrained BertForNextSentencePrediction，預測看看 B 是不是 next sentence。用 for loop 預測 100 次，取其中機率最大的當作 correct answer。

Q7: Bonus

1. Compare more than 2 contextualized embedding on this task. (1%)

	Accuracy
ELMo+BCN (see Q1)	0.46877
BERT-base+ linear classifier (see Q3)	0.52760
BERT-large+ linear classifier (see Q4)	0.53574