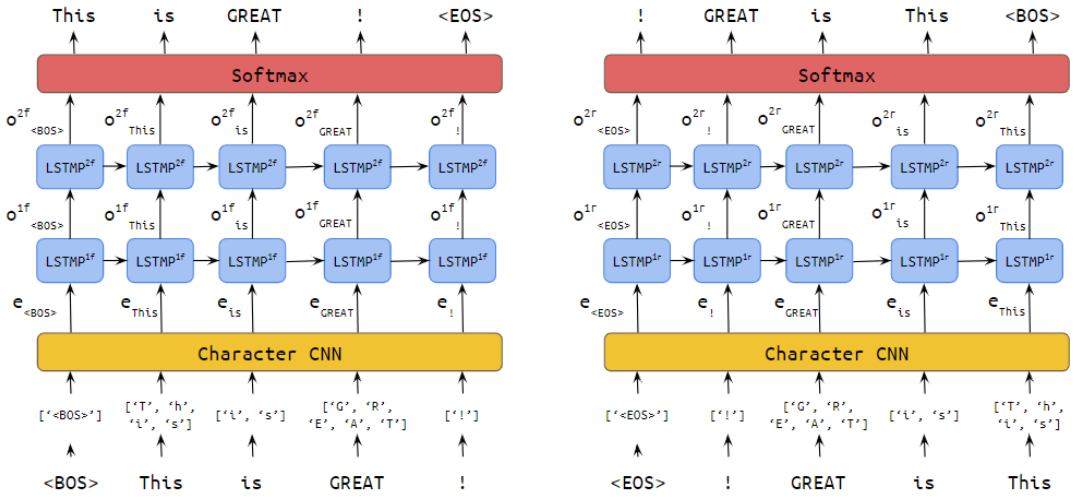
Homework2 Report

R07922058 張友誠

Q1. Elmo model

a. **Training corpus processing.** Random sample 1000k sentences from the corpus text file and split them into training and validation set in the 9:1 ratio. The word vocabulary includes the word that appears more than three times in the sampled sentences. With the similar method, we can create the character-level vocabulary containing the character that appears more than 1000 times. The other words and characters which are not in the vocabulary would be mapped into <UNK> token.

b. Model architecture.



Based on the hint provided by TA, my ELMo is composed of two network (forward and backward) which has a character CNN, two LSTM layers and an adaptiveSoftmax as the output layer.

Details.

- Character CNN: Shared by forward and backward model. Feed in the character-level indexes, and then this model would generate the character-wise embedding.
- $LSTMP^{1f}$, $LSTMP^{2f}$: LSTM with projection layer consists of the LSTM and FC layer.
- AdaptiveSoftmax: Because of the huge size of the word vocabulary, it's time-consuming to execute the softmax operation. AdaptiveSoftmax can speed up the process and with less influence in the prediction accuracy.

c. Hyper parameters.

Batch size: 64

Learning rate: 1e-3 (with the weight decay 1e-5)

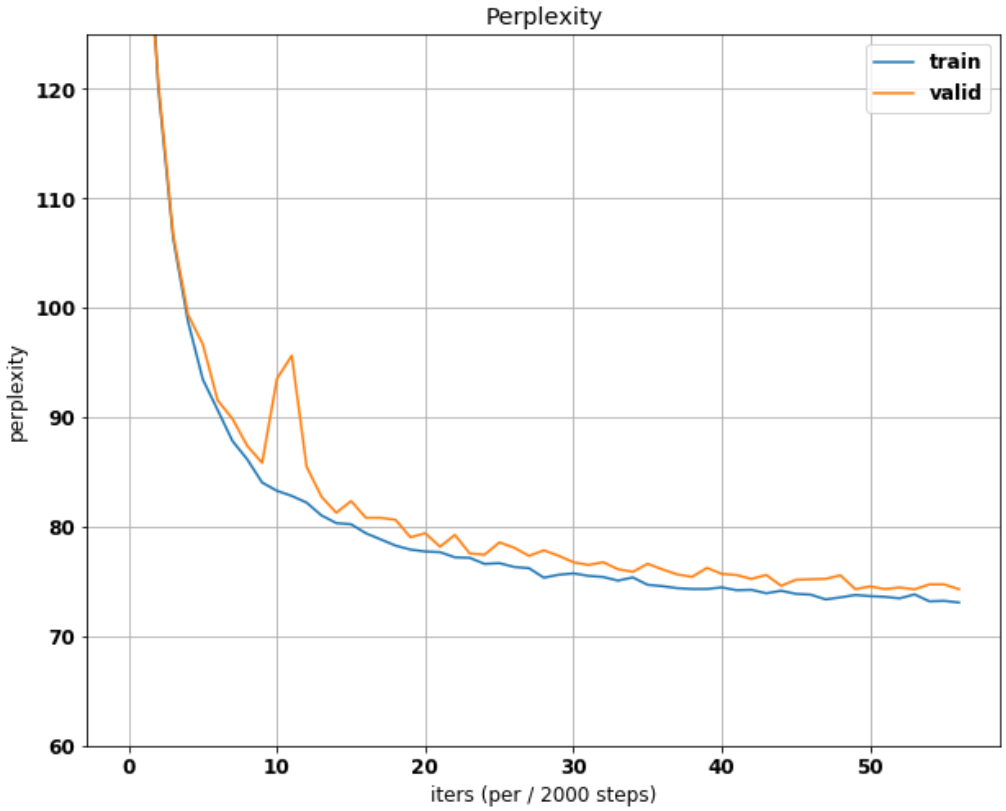
Optimizer: Adam

Feature dimension:

- Character CNN input feature: [batch, seq_len(max=64), word_len]
- Character CNN projection feature: [batch, seq_len, 512]
- $LSTMP^{1f}$, $LSTMP^{2f}$ output feature: LSTM [batch, seq_len, 2048], FC [batch, seq_len, 512]
- AdaptiveSoftmax output: [batch, seq_len] which is the probability of the correct next word.

Random seed: 3319

d. Perplexity score on the training/validation set.



e. The performance of the BCN.

- Without ELMo: 0.42443
- With ELMo: 0.49140

Q2. Compare different settings for ELMo

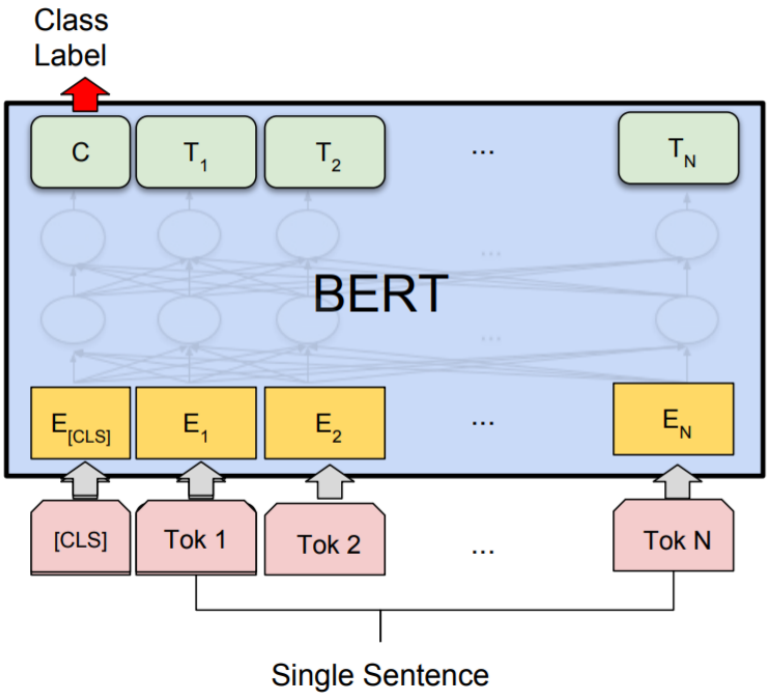
a. **Different number of the training step.** (2000 training steps / per iterator)
   - 10 iterators: 0.43800
   - 16 iterators: 0.45339
   - 21 iterators: 0.47149 (higher than simple baseline)
   - 90 iterators: 0.49140 (best)

b. Different hyper-parameters settings.
   - Random seed: [3319: 0.49140], [1282: 0.47692], [1054: 0.48416]
   - Hidden feature of the LSTM (performance after 90 training iterators): [2048: 0.49140, 1024: 0.48416, 512: 0.45158]

Q3. Strong baseline model – BERT

a. **Inputs.** word embedding (using bert tokenizer which performs split and **wordpiece** tokenization)

b. **Model architecture.**



The BertForSequenceClassification model has the pretrained weights. The input sentence is added with the special tokens “[CLS]”, “[SEP]” and tokenized into the words and sub-words set. Look up the pre-defined vocabulary and covert the words into indexes, and then we can feed the data into this model. Finally, we can get the predicted label of the given sentence.

c. **Hyper-parameters.**

Batch size: 32

Max sentence len:128

Optimizer: BertAdam (learning rate: 3.0e-6)

Pretrain model:

bert-base-cased (12-layer, 768-hidden, 12-heads, 110M parameters)

Random seed: 8981

Q4. Best Model.

The best model is composed of several BERT models with different random seed. Therefore, the inputs data, model architecture, and hyper parameters settings are mentioned in the previous question (ref. Q3).

The detail of the processing of the ensemble model is just simply summing up the logits which is predicted by each BERT model. And apply *argmax* to the summed logits to get the final predicted labels.

The reason why the ensemble model could outperform than any previous models is that each model could take care different faces of the input sentences. With the efforts from every composed models, the ensemble model can get the robust and accurate predictions.

Q5. Compare different input embeddings.

a. **Character embedding.**

Character embedding can handle unknown words in the character-level aspect and is a general embedding for any NLP related tasks. But the disadvantage is the larger computation cost. Besides, this embedding method is hard to understand the semantic meaning of the word.

b. **Word embedding.**

Word embedding projects the word into the feature with the semantic meanings. If two words are the similar, there is some specific distribution pattern between their embedding features. But the shortcut is that it could not deal with the unknown words. So the word embedding would need to be generated from the huge corpus data.

c. **Byte pair embedding.**

This is the combination of the previous two embeddings. It embeds one words into several sub-words or a complete word. If the target is composed of several sub-words, the results would contain a list of those sub-words. But when the target is just a quite simple word, (like ‘a’, ‘the’, and ‘one’), the embedding would

contain only the original target word. Sub-words allow guessing the meaning of unknown / out-of-vocabulary words, and it's usually good enough to represent the given word.

Q6. More details for BERT.

BERT uses two pre-training tasks including masked language model and next sentence prediction.

Mask language model (MLM). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike the left-to-right language model pretrained in the ELMo, the MLM objective allows the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer.

Next sentence prediction. So as to make BERT be able to deal with the QA and NLI related tasks, it needs to be introduced to learn the text-pair representations.

With the benefits of these two pre-training, BERT can handle more tasks with the robust word representations.

Homework 1 aims to solve the dialogue problem. It can also be seen as a next sentence prediction problem. As mentioned above, BERT can take care the next sentence prediction problem, so we can follow up the steps proposed in the paper and solve the homework1 task.