AI6127: Deep Learning for Natural Language Processing

Ong Jia Hui (G1903467L)
`JONG119@e.ntu.edu.sg`

Assignment 3

1 Question One [30 marks]

Modify the model of Question 3 of Tutorial 10 "Subword models" to replace the character embeddings with Byte Pair Encoding (BPE) for wordpiece model.

1.1 Pre-requisites [5 marks]

For easier dependencies installation, please download Anaconda and run the following command:

```
conda env create --file deepnlpa3.yml
```

Do make sure you have the following softwares and dependencies installed in your virtual environment:

- **Python 3.7**
- **PyTorch**
- **Pandas**: pip install pandas
- **TQDM**: pip install tqdm
- **SentencePiece****: pip install sentencepiece

** Have to be installed separately.

1.2 Implementation [5 marks]

The source codes referenced Sennrich et al.'s Subword Neural Machine Translation codes [1] to learn the BPE encoding from the training news dataset in Vectorizer's **from_dataframe**.

The source codes are developed in a way to allow the user to switch between different tokenizers using the arguments. To switch between the different subword tokenizers for the Vectorizer, update the arguments namespace accordingly:

```
model_mode = "bpe-char", # choose from word, char, bpe-char, bpe-word, sent
vocab_size = 1000,  # 1000, 3000, 10000
```

1.3 Source Code Changes [5 marks]

A new vocabulary called *SubwordSequenceVocabulary* was added to the codes. It stores the trained BPE codes within the vocabulary. There is a new Section called "BPE Trainer and Segmenter" that contains codes, which adapted BPE for subword tokenization training and application.

Two approaches were experimented in the Vectorizer. For the first approach named "BPE-char", the title sentence is first splitted into words. Each word is then tokenized into subword tokens using a BPE tokenizer. The out_vectors are then represented in a shape of (max_seq_length, max_word_length) and passed into the embedding layer followed by a 1D convolutional layer to train the subword embeddings.

The second approach, "BPE-word" segments the title into subword tokens and constructed the out_vectors in a shape of (max_sent_length,). This vector passes through the token embedding layer (same as word embedding layer, but named as char_emb for simplicity).

The NewsClassifier model was modified to handle both modes.

1.4 Results [10 marks]

Table 1: Test Loss and Test Accuracy of WordPiece in comparison with various models

Model	**Vocab Size**	**Test Loss**	**Test Accuracy**
char	82	0.683	82.042
BPE-char	1,000	0.647	**83.672**
BPE-char	3,000	0.646	**84.035**
BPE-char	10,000	0.593	**85.353**
word	3,686	0.832	78.465
word_glove	3,686	0.563	82.935
BPE-word	1,000	0.958	**78.644**
BPE-word	3,000	0.938	**80.140**
BPE-word	10,000	1.173	**81.317**

The first approach "BPE-char" of tokenizing sentences into words then words into subwords produced much better test accuracies than the second approach. It also outperforms character level and word level tokenizers significantly.

2 Question Two [20 marks]

Modify the model with BPE for sentencepiece model.

2.1 Implementation [10 marks]

The source codes were modified to use Google's unsupervised text tokenizer [2]. Unlike the WordPiece model, the SentencePiece model does not segment the words before segmenting the tokens, instead it segments the entire raw input. Therefore, whitespaces are included in the tokenization and they are replaced with the underscore ("_"). The following example shows how the two subword tokenizers differs in output:

```
[Original Text]
I love Natural Language Processing
[BPE]
"I", "love", "natural", "lang@@", "uage", "process@@", "ing"
[SentencePiece]
"I", "_love", "_natural", "_lang", "uage", "_process", "ing"
```

2.2 Source Code Changes [5 marks]

A new vocabulary called *SentenceSequenceVocabulary* was added to the codes. Like the *WordSequence-Vocabulary*, the *SentenceSequenceVocabulary* also reserves the IDs for padding, begin of sentence, end of sentence and unknown tokens. The **load_model_file** function will initialize the SentencePiece segmenter using the trained model file. It is called in the Vectorizer's **from_dataframe** class method, where the SentencePieceTrainer will load the trained model.

The Vectorizer's **vectorize** method was also modified to use the SentencePiece segmenter to retrieve the vocabulary IDs to produce the out_vectors. The out_vectors have a shape of (max_sent_length,), where the "max_sent_length" is the maximum length of all titles, obtained in the *Dataset*'s **init** method.

In order to switch the classifier to use SentencePiece tokenizer, please modify the following parameter arguments in Namespace:

```
model_mode = "sent", # choose from word, char, bpe-char, bpe-word, sent
vocab_size = 1000,  # 1000, 3000, 10000
```

2.3 Results [5 marks]

Table 2: Test Loss and Test Accuracy of SentencePiece in comparison with various models

Model	Vocab Size	Test Loss	Test Accuracy
word	3,686	0.832	78.465
word_glove	3,686	0.563	82.935
BPE-word	1,000	0.958	78.644
BPE-word	3,000	0.938	80.140
BPE-word	10,000	1.173	81.317
SentencePiece	1,000	0.912	**79.308**
SentencePiece	3,000	0.703	**80.692**
SentencePiece	10,000	0.851	**82.818**

In this experiment, the SentencePiece model performed better than the BPE subword token embedding models denoted as "BPE-word". It also performed better than the word embedding layer model without the use of pre-trained glove embedding weights. In order to alleviate the overfitting issue, other than increasing the dropout probability, a weight decay value of 1e-5 was added to the Adam optimizer to regularize the high variance in the test set.

References

[1] Alexandra Birch Rico Sennrich Barry Haddow. *Subword Neural Machine Translation*. https://github.com/rsennrich/subword-nmt. 2016.

[2] John Richardson Taku Kudo. *SentencePiece*. https://github.com/google/sentencepiece. 2018.