Intro_to_AI HW2

Part 0

- Briefly explain the method you implemented and give an example (such as the E.g in the remove_stopwords function) in the report.

ANS:

In the ‘remove_stopwords’ function, it removes words that are in the stopword list of nltk. And in my preprocessing_function( ), I first convert all the words to lower case. I found that ‘
’ often appears in the dataset, so I replace it with space. I import ‘re’, using regular expression [^a-zA-Z] to remove characters that are not an alphabet and then call remove_stopwords( ). What’s more, I use ‘SnowballStemmer’ from nltk to perform stemming.

E.g.,

Text = ‘One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked.

The first thing that struck me about Oz was its brutality and unflinching scenes of violence.’

Preprocessed_text = ‘one review mention watch oz episod hook first thing struck oz brutal unflinch scene violenc’

Part 1

- Briefly explain the concept of perplexity in report and discuss how it will be influenced.

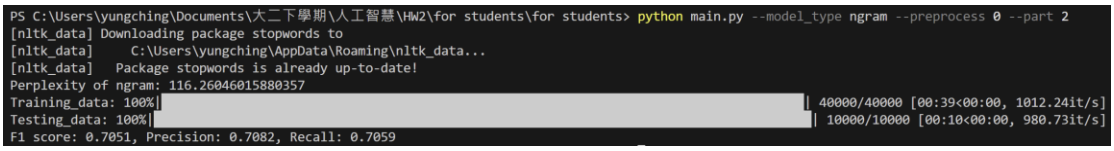
ANS:

Perplexity is a measurement of how well a probability distribution or probabilistic model predicts a sample. It is often used in NLP to evaluate the performance of language models. Perplexity is computed using formula ‘Perplexity = 2^(-entropy)’ and lower perplexity indicates that the model is better at predicting the test data.

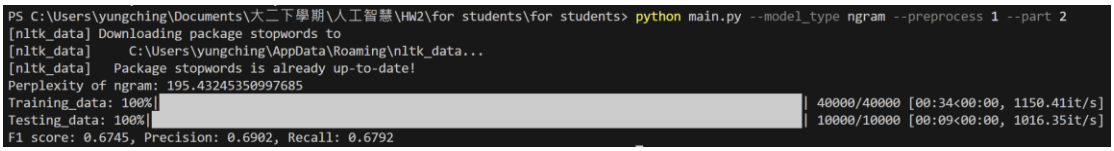
Perplexity is often influenced by data quality, data size, size of n-grams, preprocess method or model quality. Poor data quality or smaller data size may lead to higher perplexity.

- Screenshot the outputs and tell your observations about the differences in the perplexity caused by the preprocessing methods (1. Without preprocess 2. with remove stopwords 3. with your method).

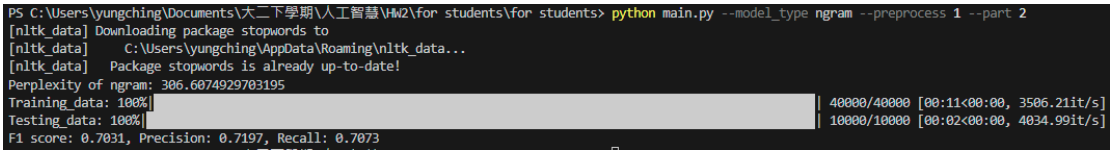
1. Without preprocess (Perplexity 116, F1 score 0.7051)



2. With remove stopwords (Perplexity 195, F1 score 0.6745)



3. With my method(Perplexity 306, F1 score 0.7031)



Perplexity: my method > remove stopwords > without preprocess.
It seems like the more complex the preprocess method is, the higher the perplexity is. The reason may be applying preprocess can lead to loss of important semantic information from the text. As a result, the model may struggle to accurately predict the next word or sequence of words. However, higher perplexity does not mean that the lower the precision is.

Part 2

- Briefly explain the two pre-training steps in BERT.

ANS:

The two pre-training steps are Masked Language Model (MLM) and Next Sentence Prediction (NSP).
In MLM, 15% of the words were masked and the model is trained to predict the original masked words from other unmasked words. This allow Bert to learn bidirectional representations of words.
In NSP, the model receives pairs of sentences and predict if the two sentences are subsequent. Half of the training examples pair consecutive sentences, while the other half pairs sentences randomly selected from the corpus.

- Briefly explain four different BERT application scenarios

ANS:

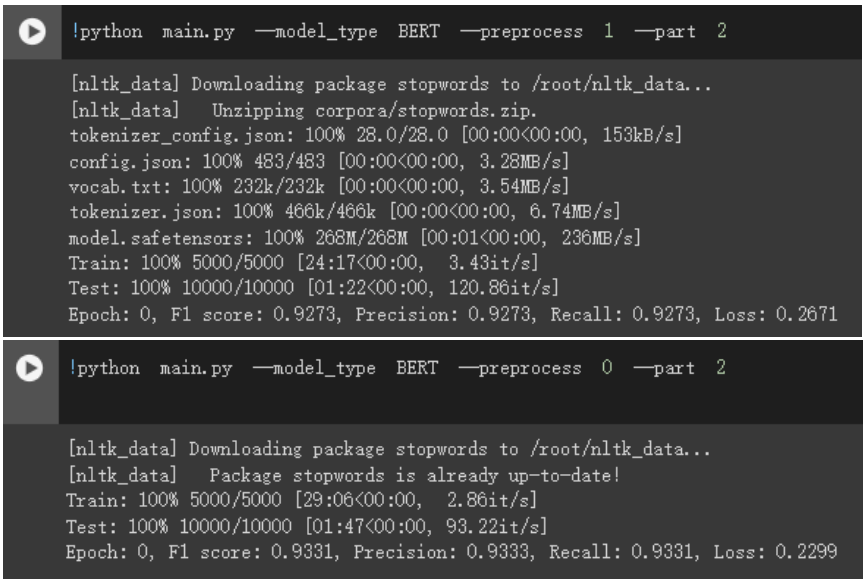
- Sequence Classification: Input single sentence / output class. For example, sentiment analysis.
- Token Classification: Input single sentence / output is class of each word. For example, slot-filling.
- Sequences Classification: Input two sentences / output class. For example, natural language inference.
- Question Answering: Input an article and ask a question / output answer.

- Discuss the difference between BERT and distilBERT?

ANS:

DistelBERT is a smaller and faster version of BERT, designed to be more efficient. DistelBERT has less transformer layers and parameters compared to BERT, thus masking it faster to train and deploy the model. And the training process of DistelBERT is different from BERT. It is trained using knowledge distillation, a technique where a smaller model is trained to mimic the behavior of a larger model .

- Screenshot the required test F1-score. The F1 score is 0.9273 which is slightly worse than F1 score(0.9331) without preprocessing.



- (BONUS 5%) Explain the relation of the Transformer and BERT and the core of the Transformer.

ANS:

Transformer architecture is the foundation upon which BERT is built. The name of BERT is “Bidirectional Encoder Representations from Transformers”. Transformer is consisted of multiple layers of encoder and decoder. BERT use the encode part of Transformer and pretrain additionally. The core of Transformer is “self-attention”. This mechanism allows the model to weigh the importance of different words in the input sequence when encoding or decoding.

Part 3

- Briefly explain the difference between vanilla RNN and LSTM.

ANS:

Vanilla RNN and LSTM are both recurrent neural networks used for sequential modeling tasks. While the main issue with vanilla RNNs is the vanishing gradient problem, which makes it difficult to capture long-range dependencies between sequences when processing long sequences. Therefore, LSTM was introduced. It adds a memory cell used to store long-term information and three gates (input gate, forget gate, and output gate). They allow the model to selectively update, retain, or discard information.

- Please explain the meaning of each dimension of the input and output for each layer in the model. For example, the first dimension of input for LSTM is batch size.

ANS:

In my model, I called nn.Embedding(), nn.LSTM(), nn.Linear() and nn.Dropout.

```
# TO-DO 3-2: Determine which modules your model should consist of
# BEGIN YOUR CODE
self.embedding = nn.Embedding(vocab_size, embedding_dim)
self.encoder = nn.LSTM(embedding_dim, hidden_dim, num_layers, dropout=dropout)
self.fc = nn.Linear(hidden_dim, output_dim)
self.dropout = nn.Dropout(dropout)
# END YOUR CODE
```

- Embedding layer:

  Input: a batch of sequences of tokens. (batch_size, sequence_length)

  Output: embedding vectors for each token in the input. (batch_size, sequence_length, embedding_dim)
- LSTM:

  Input: the embedded representation of the input sequences. (batch_size, sequence_length, embedding_dim)

  Output: a sequence of hidden states representing the encoded information. (batch_size, sequence_length, hidden_dim)
- Linear layer:

  Input: the hidden state from the last time step of the LSTM encoder.

  Output: the logits for each class in the classification task. (batch_size, output_dim) output_dim is the number of class.
- Dropout layer:

  Input: output from the linear layer.

  Output: a randomly zero-masked version of the input tensor. Shape is the same as input.

- Screenshot the required test F1-score.

  The test F1 score of last epoch is 0.879.

```
[4]  !python  main.py  --model_type  RNN  --preprocess  1  --part  2

     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     Train: 100% 5000/5000 [02:09<00:00, 38.58it/s]
     Test: 100% 10000/10000 [00:11<00:00, 855.12it/s]
     Epoch: 0, F1 score: 0.5866, Precision: 0.5868, Recall: 0.5867, Loss: 0.6936
     Train: 100% 5000/5000 [02:07<00:00, 39.19it/s]
     Test: 100% 10000/10000 [00:10<00:00, 938.99it/s]
     Epoch: 1, F1 score: 0.8255, Precision: 0.8258, Recall: 0.8255, Loss: 0.6207
     Train: 100% 5000/5000 [02:07<00:00, 39.32it/s]
     Test: 100% 10000/10000 [00:11<00:00, 881.10it/s]
     Epoch: 2, F1 score: 0.8687, Precision: 0.8688, Recall: 0.8687, Loss: 0.455
     Train: 100% 5000/5000 [02:08<00:00, 38.88it/s]
     Test: 100% 10000/10000 [00:11<00:00, 894.88it/s]
     Epoch: 3, F1 score: 0.8769, Precision: 0.8769, Recall: 0.8769, Loss: 0.3662
     Train: 100% 5000/5000 [02:08<00:00, 38.84it/s]
     Test: 100% 10000/10000 [00:10<00:00, 938.46it/s]
     Epoch: 4, F1 score: 0.8749, Precision: 0.8761, Recall: 0.875, Loss: 0.3134
     Train: 100% 5000/5000 [02:07<00:00, 39.14it/s]
     Test: 100% 10000/10000 [00:12<00:00, 819.03it/s]
     Epoch: 5, F1 score: 0.8824, Precision: 0.8834, Recall: 0.8825, Loss: 0.2764
     Train: 100% 5000/5000 [02:10<00:00, 38.20it/s]
     Test: 100% 10000/10000 [00:11<00:00, 849.05it/s]
     Epoch: 6, F1 score: 0.8777, Precision: 0.8803, Recall: 0.8779, Loss: 0.2492
     Train: 100% 5000/5000 [02:11<00:00, 38.06it/s]
     Test: 100% 10000/10000 [00:11<00:00, 860.52it/s]
     Epoch: 7, F1 score: 0.879, Precision: 0.8806, Recall: 0.8791, Loss: 0.2315
     Train: 100% 5000/5000 [02:10<00:00, 38.39it/s]
     Test: 100% 10000/10000 [00:10<00:00, 918.13it/s]
     Epoch: 8, F1 score: 0.884, Precision: 0.884, Recall: 0.884, Loss: 0.2189
     Train: 100% 5000/5000 [02:09<00:00, 38.72it/s]
     Test: 100% 10000/10000 [00:11<00:00, 898.30it/s]
     Epoch: 9, F1 score: 0.879, Precision: 0.879, Recall: 0.879, Loss: 0.2103
```

Discussion

- Discuss the innovation of the NLP field and your thoughts of why the technique is evolving from n-gram -> LSTM -> BERT.

ANS:

The Transformer architecture revolutionized NLP with 'Attention is All You Need', replacing RNNs with self-attention for parallel sequence processing. This paved the way for models like BERT and GPT, which rely on pre-training and transfer learning on massive text corpora to understand language nuances. Notably, GPT exemplifies multimodal NLP, where models are trained across text, images, and audio.

N-grams are the most basic and straightforward approach in NLP. Although they compute efficiently, they often lack context, struggle with understanding nuances in language and are limited to long-range dependencies. LSTM improves by learning long-term dependencies but faces issues with sequential processing like vanishing or exploding gradients. Therefore, BERT is introduced. It employs attention mechanisms and processes the entire input sentence in parallel, with preprocessing, which demonstrates the importance of advancements in deep learning for complex NLP challenges.

- Describe problems you meet and how you solve them.

ANS:

In the preprocess function, I use the SnowballStemmer for stemming, but its results are not as good as I thought. For example, ‘violence’ will be stemmed to ‘violenc’, but it is incorrect.

I initially ran BERT on my local CPU and it expected to take about 6-7 hours. It spent too much time and was ineffective. Therefore, I switched to using Colab, but initially forgot to configure it to utilize the GPU. Once I made that change, the process became much faster, only needing around 20-30 minutes.

However, the F1-score of BERT has been consistently stuck at around 0.92 no matter how I change the design of model. I’m not quite sure how to properly configure the various layers of the models and their parameters.