NLP2 course work2 Taeseung You

Task1

Method Description

For this course work, the main key point was the way to decide the preprocessing and text cleaning because this text file has a lot of tagging. Mostly, t tag was the most important. t tag is the divider of each review. So firstly, get the data from document and get all lines from raw data. Next, checking line if t tag is coming out then regard next few sentences as a same review. The way reviews are stored is the split each review sentences with ## tag so divide with ## tag and only store the sentences. I used the get_tokenizer from pytorch to get tokenize words. And after tokenization, lemmatize and pos tagging the tokenize word to make better performance. Also remove punctuation and some weird symbol. Lastly, remove the stop word to get the more accurate words. This is how I preprocess the each document.

After cleaning data, I read each text file and put every thing in wordvec which represent the position of each words' position. And also put the word in an array(wordlist) regardless the position of the word to check top 50 words. I used counter method to check the how many words are used. After put the all words in the wordlist and put in the counter. After that check the word if reversed word is same as the word itself. If it is, then delete the word. Then use most_common method to get top 50. After that, check whether how many words came out, and divide by two because checking how many time I should change the original word to reversed word. Moreover, get all the indices of each top 50 words. Randomly select the indices and change the word with reversed word. After changing original documentation with reverse word, put this data in to word2vec function. Use word2vec for model to get the word embedding vectors. Build vocabulary with wordvec and train with wordvec, After training, get the all the vectors and indices. Only Get vectors for top50 words and reversed top50 words.

Put all the vectors in the GAAC cluster which is included in nltk. GAAC cluster evaluate cluster based on all similarities between documents which equate similarity with the similarity of a single pair of documents.

Check the label of each terms if word in top50 and reversed one has same label, then regard it as correct pair. After this, calculate the percentage of correct pair and mean and standard deviation of correct percentages.

Run 10 times with this procedure from randomly select the indices of top50 words to clustering so check the mean and standard deviation of all the percentages.

Result analysis

For the parameter of word2vec mode, I set the min_count as 5. So frequency of word below min count are dropped before training occurs. I was thinking 5 is quite reasonable number to make drop. And also decided window as 10. If set the window size as too small number, It will only see the local area and can't capture semantics but at the same time set it as too large number, It won't focus on syntax. For training part, I set epoch as 20, the reason why I choose is just try and error. When I change the value of epoch,20 give more high accuracy.

I've got 74.4 mean accuracy and standard deviation as 3.07 as you can see in Task 1 result in appendix.

Task2

Method Description

For this part of preprocessing, I need to make different preprocessing method because I need to capture whether this review is good or not. For preprocessing, get each lines and check whether t tag exist or not. If it doesn't, regard it sentence as a review. only read the sentence and just check the + or – and assign the pos_reviews or neg_reviews but which doesn't have any + or – before, then regard the sentence as same opinion as last one. In the situation of t tag exist, check rating. After I check all the document, I realise there's only +3 to -3 so can manually add or subtract the numbers and decide that review is positive or not. If rating is bigger than 0, then assign it as positive other wise assign it to negative. Moreover, It tag appear just once, then think it same as none t tag document.. After that, remove punctuation and delete stop word from the sentences. Assign sentiment as positive if review is in positive reviews and assign negative if review in negative reviews. Append all the data in the same data frame, then tokenize all the sentences in the data frame. After got tokenization, make input sequence in the same length for modelling and also converting the labels in to numbers.

The reason to use LSTM as a method is effective in memorization of important data. LSTM can use several word string to find out class which It belongs. If we use appropriate layer of embedding, LSTM will be able to find out the actual meaning input string and will give the most accurate output.

The output will create 2 output values which are positive or negative. Activation function is SoftMax for multi-class classification because it needs to be multi class so categorical_crossentropy should be used as the loss function.

For the K-fold cross validation part, Use KFold with 5 split and shuffle true unless it will use just original data which might be biased. Then select the index of training and testing and assign things in to X and Y. Inside the 5 times looping, put X_train and Y_train in to model and train with these values so finally get mean and standard deviation of each accuracy with looping 5 times.

Result analysis

For the hyper parameter, I've set MAX_NB_WORD with 50000 which is the maximum number of words to be used most frequently. And MAX_SEQUENCE_LENGTH as 250 which is the maximum number of words in each review. For the model side, the first layer is the embedded layer which uses 100 length vector to represent each word. Set the drop out values as 0.2. Next layer is the LSTM layer with 100 memory units so this output layer will create 2 output values. SoftMax is used for activation function for multi-class classification. I used the categorical cross entropy for loss function because this is the multi-class classification

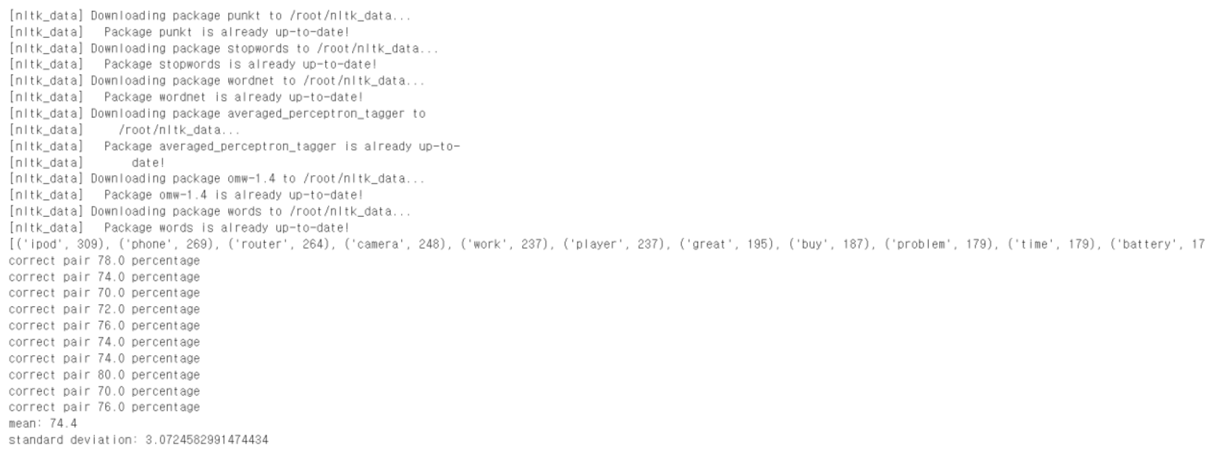
Epoch is used 50 however patience is set so the number of epoch doesn't have effectiveness as usual. Also set batch_size as 32. If set to 16,it's too slow to train and also result a bit drop down. Set validation set as 0.1 so just use 10 percentage as evaluate data. And min_delta is set as 0.0001

As an result, I've got 70 percent of mean accuracy and 0.026 as a standard deviation.

As you can see the result in appendix Task 2 Result.

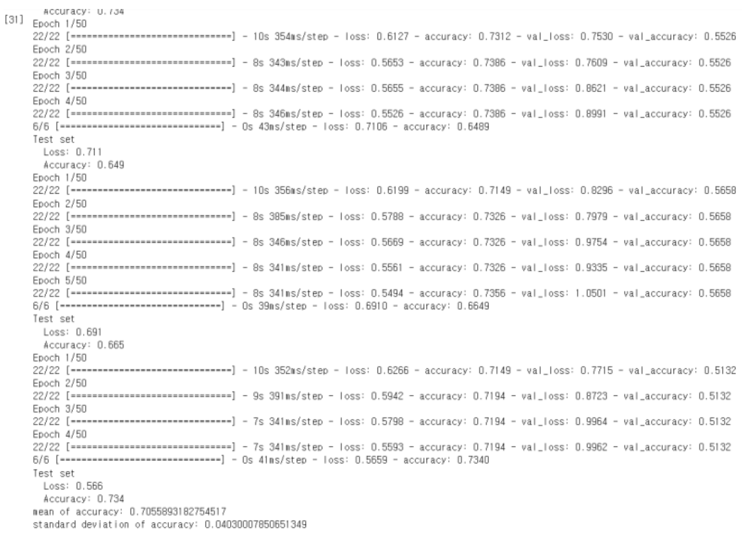
Appendix

Task 1 Result



```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[('ipod', 309), ('phone', 269), ('router', 264), ('camera', 248), ('work', 237), ('player', 237), ('great', 195), ('buy', 187), ('problem', 179), ('time', 179), ('battery', 17
correct pair 78.0 percentage
correct pair 74.0 percentage
correct pair 70.0 percentage
correct pair 72.0 percentage
correct pair 76.0 percentage
correct pair 74.0 percentage
correct pair 74.0 percentage
correct pair 80.0 percentage
correct pair 70.0 percentage
correct pair 76.0 percentage
mean: 74.4
standard deviation: 3.0724582991474434
```

Task2 Result



```
      Accuracy: 0.734
[31] Epoch 1/50
     22/22 [==============================] - 10s 354ms/step - loss: 0.6127 - accuracy: 0.7312 - val_loss: 0.7530 - val_accuracy: 0.5526
     Epoch 2/50
     22/22 [==============================] - 8s 343ms/step - loss: 0.5653 - accuracy: 0.7386 - val_loss: 0.7609 - val_accuracy: 0.5526
     Epoch 3/50
     22/22 [==============================] - 8s 344ms/step - loss: 0.5655 - accuracy: 0.7386 - val_loss: 0.8621 - val_accuracy: 0.5526
     Epoch 4/50
     22/22 [==============================] - 8s 346ms/step - loss: 0.5526 - accuracy: 0.7386 - val_loss: 0.8991 - val_accuracy: 0.5526
     6/6 [==============================] - 0s 43ms/step - loss: 0.7106 - accuracy: 0.6489
     Test set
       Loss: 0.711
       Accuracy: 0.649
     Epoch 1/50
     22/22 [==============================] - 10s 356ms/step - loss: 0.6199 - accuracy: 0.7149 - val_loss: 0.8296 - val_accuracy: 0.5658
     Epoch 2/50
     22/22 [==============================] - 8s 385ms/step - loss: 0.5788 - accuracy: 0.7326 - val_loss: 0.7979 - val_accuracy: 0.5658
     Epoch 3/50
     22/22 [==============================] - 8s 346ms/step - loss: 0.5669 - accuracy: 0.7326 - val_loss: 0.9754 - val_accuracy: 0.5658
     Epoch 4/50
     22/22 [==============================] - 8s 341ms/step - loss: 0.5561 - accuracy: 0.7326 - val_loss: 0.9335 - val_accuracy: 0.5658
     Epoch 5/50
     22/22 [==============================] - 8s 341ms/step - loss: 0.5494 - accuracy: 0.7356 - val_loss: 1.0501 - val_accuracy: 0.5658
     6/6 [==============================] - 0s 39ms/step - loss: 0.6910 - accuracy: 0.6649
     Test set
       Loss: 0.691
       Accuracy: 0.665
     Epoch 1/50
     22/22 [==============================] - 10s 352ms/step - loss: 0.6266 - accuracy: 0.7149 - val_loss: 0.7715 - val_accuracy: 0.5132
     Epoch 2/50
     22/22 [==============================] - 9s 391ms/step - loss: 0.5942 - accuracy: 0.7194 - val_loss: 0.8723 - val_accuracy: 0.5132
     Epoch 3/50
     22/22 [==============================] - 7s 341ms/step - loss: 0.5798 - accuracy: 0.7194 - val_loss: 0.9964 - val_accuracy: 0.5132
     Epoch 4/50
     22/22 [==============================] - 7s 341ms/step - loss: 0.5593 - accuracy: 0.7194 - val_loss: 0.9962 - val_accuracy: 0.5132
     6/6 [==============================] - 0s 41ms/step - loss: 0.5659 - accuracy: 0.7340
     Test set
       Loss: 0.566
       Accuracy: 0.734
     mean of accuracy: 0.7055893182754517
     standard deviation of accuracy: 0.04030007850651349
```