SHORT REPORT

ID:20085189G

NAME:HUANGZHEMING

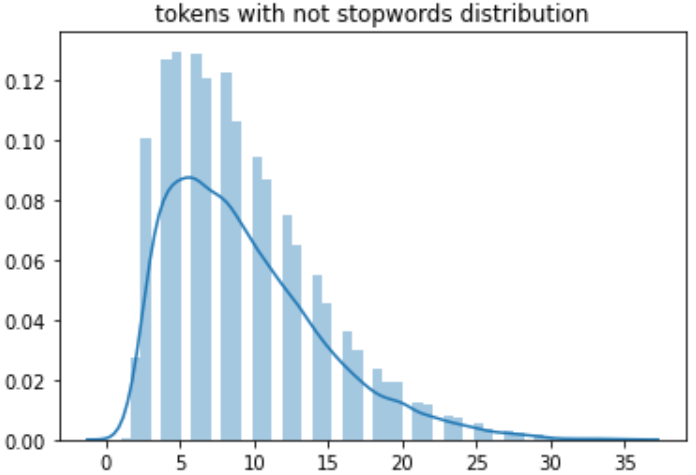
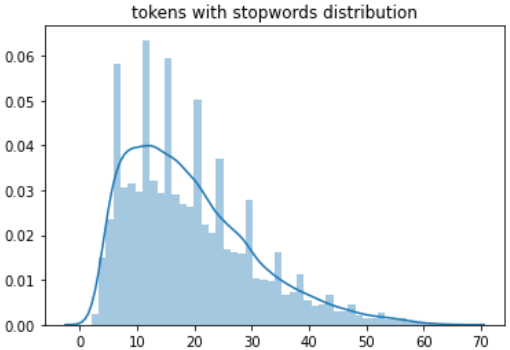
The project focus on the classification on the emotion of a text input. Here we design a flow to handle them : **Data Cleaning -> Word Embedding -> Preprocessing -> Model Training -> Model Evaluation -> Prediction.** For such a short text sentiment classification task, it seems to be inefficient to use a traditional ML model which highly relies on the embedding effect, but here we have just about 18,000 texts to use to train the model. Anyway the document representation is a problem too. So we decide to use the **LSTM** DL model, which can handle the sentences more efficient and explainable.

1. **Data Cleaning**

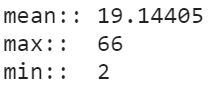
Firstly, we read the data files (txt) as python list while splitting the labels and text. And to ensure the data amount and prepare for the embedding task, i decide to combine all the text data. (validation data, train data,test data)

And the most important thing is to tokenize the raw text, here we use WordNetLemmatizer, and then we can get a list of words to represent every sentences. And after several evaluation, i decide not to delete the stop words.

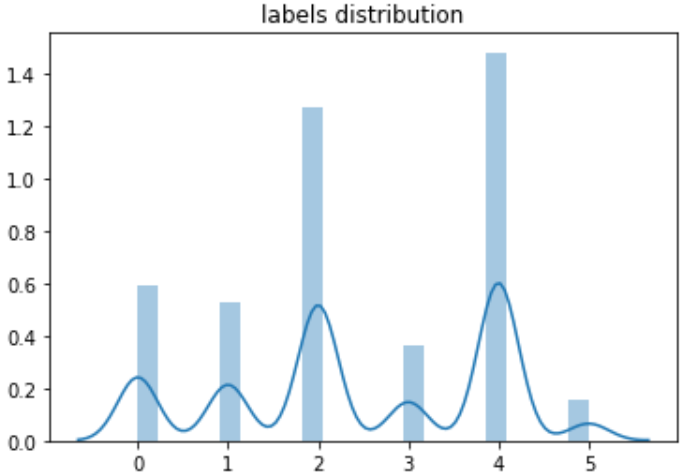
Let`s go through our data, this is our length of sentence in different situation.



If we do not delete the stopwords, we can exactly know the average number of the words in a sentence, which may lead to different fixed sentence length in the **Preprocessing** part



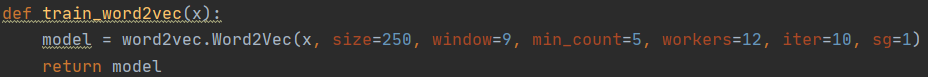
We can also notice that the imbalance of the labels, it may challenge the classification model,





1. **Embedding**

In this part we should input all the tokens into the word2vec model, and set the parameters below,



After that we can get a word2vec model, in which we can get every vectors according to the input words and the model`s hidden layer parameters. And we should save as a .model file.

1. **Preprocessing**

Before we training the model, we should handling our data by the help of the word2vec model. In this part we focus on 4 things:

- get the embedding matrix from our w2v model before

- transform our str type words into int id type

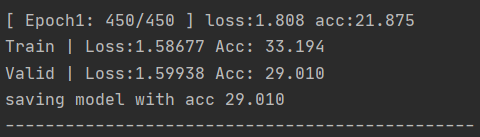
- Padding the sentence to **30** words

- handling the unknown words

1. **Model Design and Training**

Here we design a simple LSTM classifier:one embedding layer + one LSTM layer + one Linear model. The cost function is CrossEntropy which is always good to evaluate every class`s score. And we use the Adam as the optimization method.

After randomly split the handled data into train data set (0.8) and validation data set (0.2), Here we go, let`s train the model!

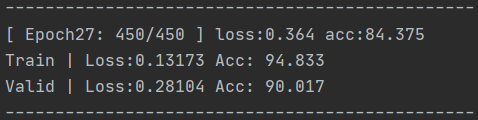


1. **Evaluation**

For every epoch, we get different training loss and validation loss as well as the accuracy.



we can see, after about 20 epoch, our model learn to much to well perform the validation data set. And finally we hit the best score as below



And now, we save our model with best score, and use it in the next stage.

1. **Prediction**

In this part all we need to do, is to transform our text data as what we have done and input them into the model, and get the predict the labels.

Here we generate two txt files, one is to hand in my homework (only labels), another is to manually check the results (text and labels) as below



Emmm, it seems that the result is not bad.

**Project Structure**

