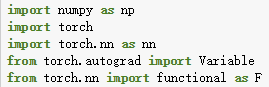
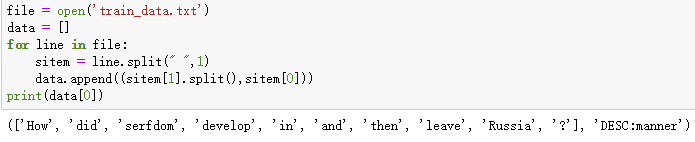
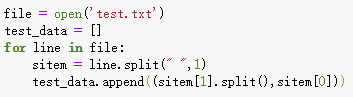
1.



Packages needed for import models.

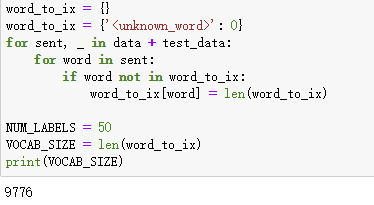
2.





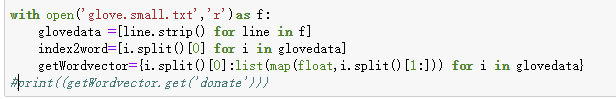
Read and process the training set, tokenize the sentence and extract label, the format is as shown in the figure. Test set data is processed in the same way as training data.

3.



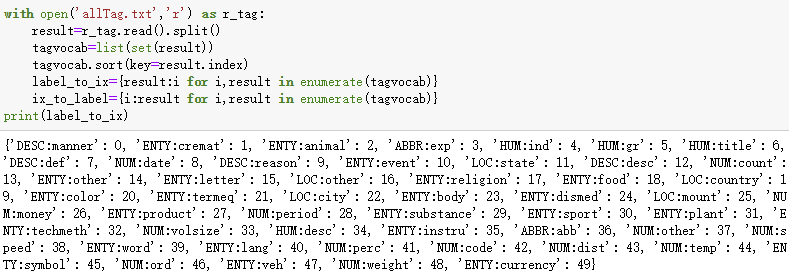
Create a dictionary ‘word\_to\_ix’ based on the words that appear in the dataset. ‘Index = 0’ in the dictionary corresponds to ‘unkown word’, which is used to process unknown words. And the number of words is the size of the dictionary length.

4.



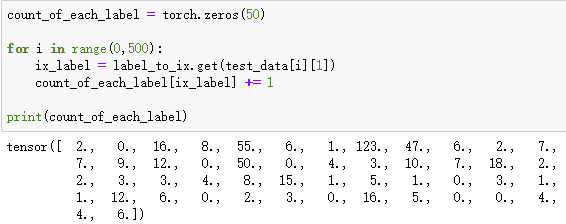
Extract the pre-trained word vector in glove, create‘index2word’, which can find the word by index. ‘getWordvector’ dictionary can input a word and return the word vector corresponding to this word in glove.

5.



Read all the labels in a allTag.txt, and create two dictionaries called ‘label\_to\_ix’ and ‘ix\_to\_label’.

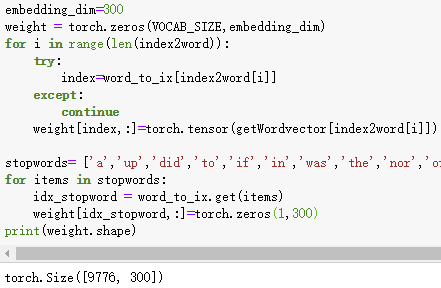
6.



This part is used to evaluate the accuracy of the model for each category of problem prediction,

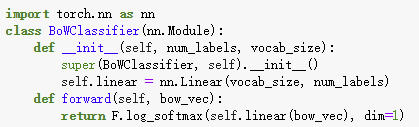
Define ‘count\_of\_each\_label’, the length of the tensor is 50, and each number in the tensor corresponds to the number of test sentences which belong to the label of this index. For example, from the figure, you can see that the value of ‘count\_of\_each\_label [2]’ is 16, and the corresponding label when index = 2 in {label\_to\_ix} is 'ENTY: animal', so there are 16 sentences whose label is 'ENTY: animal' in test set .

7.



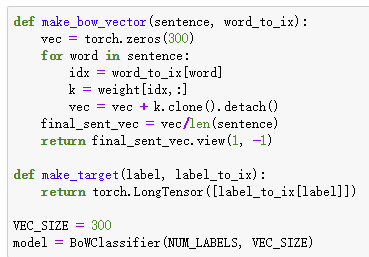
A weight matrix is created here to store the word vectors of the words in the dataset. At the same time, because there are stopwords in the data set, we filtered some of them and chose to define their word vector as 0, so as to reduce the impact of stopwords on the accuracy of the model.

8.



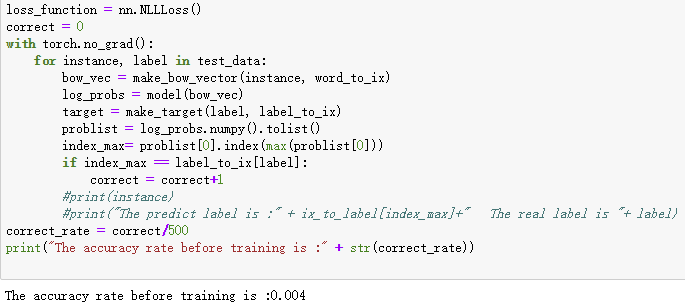
The BOW classifier model processes the data through the linear layer. Finally, we choose to use the softmax function to output the result. The output is a probability distribution vector with a length of 50.

9.



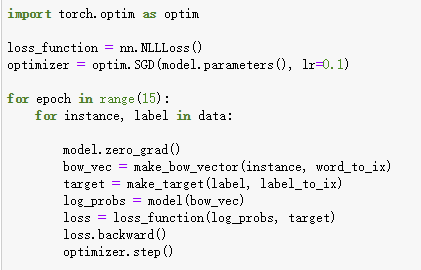
The make\_bow\_vector function is used to generate sentence vectors. make\_target is used to extract the label value of the sample and return the index corresponding to the label.

10.



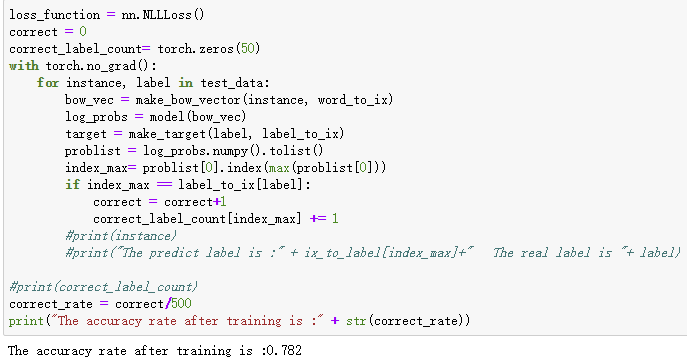
Here, in this step, an untrained model is used to test the test set and the output accuracy is to compare with the accuracy of the trained model later.

11.



The training step, we use the NLLLoss function as the loss function because it is suitable for multi-classification problems, and we use stochastic gradient descent to optimize the model.

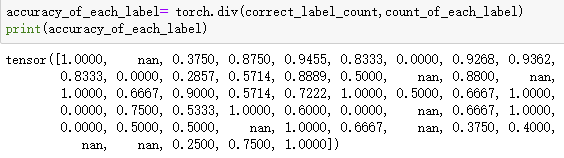
12.



After training, use the test set to test again, and the accuracy has been improved.

At the same time, We define a tensor named correct\_label\_count with a length of 50 to store the number of correct judgments for each label. After each prediction, if the prediction is correct, add 1 to the index of the corresponding label.

13.



For the model performance for each class, the correct rate can be expressed by

torch.div (correct\_label\_count, count\_of\_each\_label). For example:

accuracy\_of\_each\_label [2] = 0.3750, which means that for sentence whose the label is 'ENTY: animal', the accuracy of the classifier is only 0.3750.