Assignment-4

CNN classifier

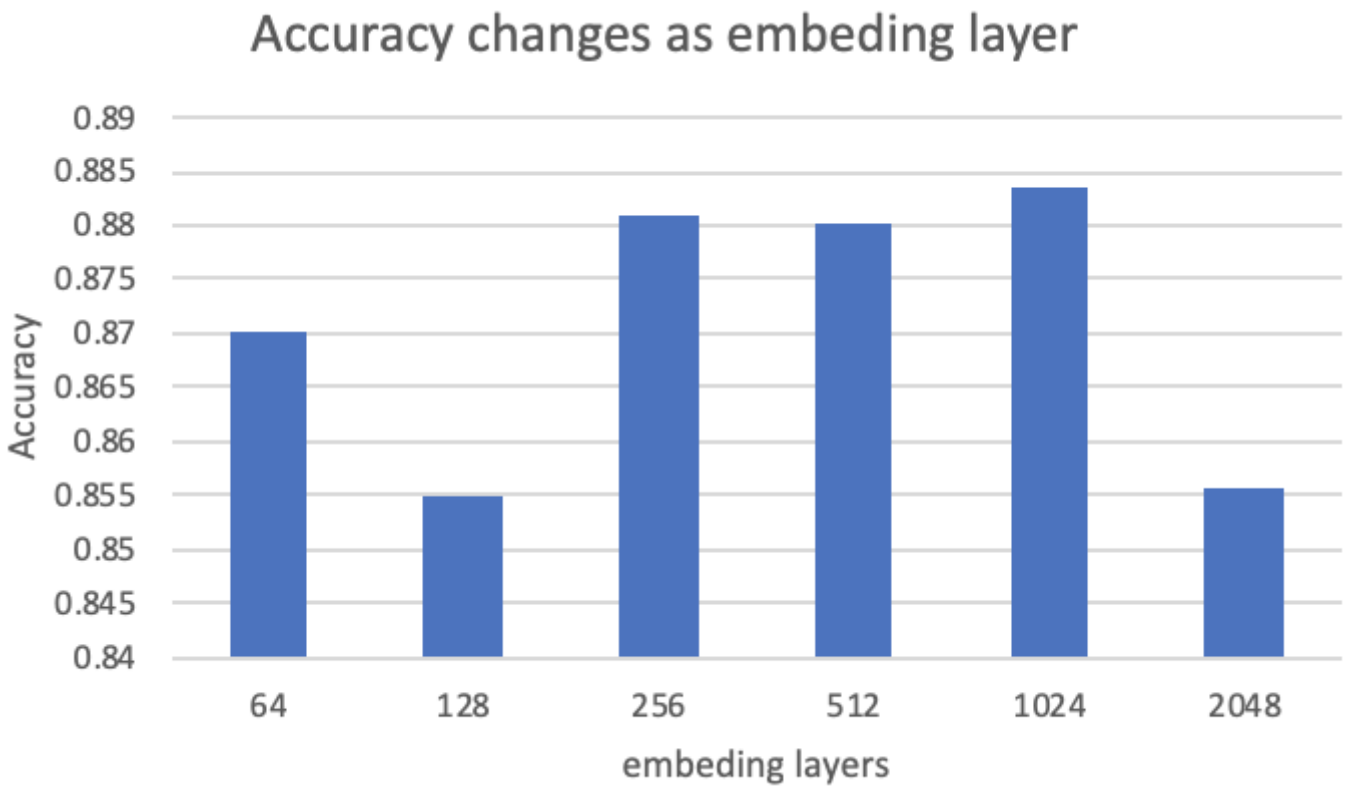
Method

To implement a CNN classifier with fastNLP, I code as the following steps:

1. get train data and test data by given function
2. replace \n \t with space, remove redundent spaces to make a document be a sentence and write sentence with their tags separated by \t as CSV format
3. use CSVLoader to load data
4. change all letter to lowcase
5. use Vocabulary class to statistic words and change word sequence to integer
6. use CNNText and Trainer to train data, get final result and get accuracy.

Result

tuning dropout and embed_dim



As I tune the embeding dimsion, I found that the accuracy peak at 0.88369 when embeding dimension is 1024

RNN(LSTM) classifier

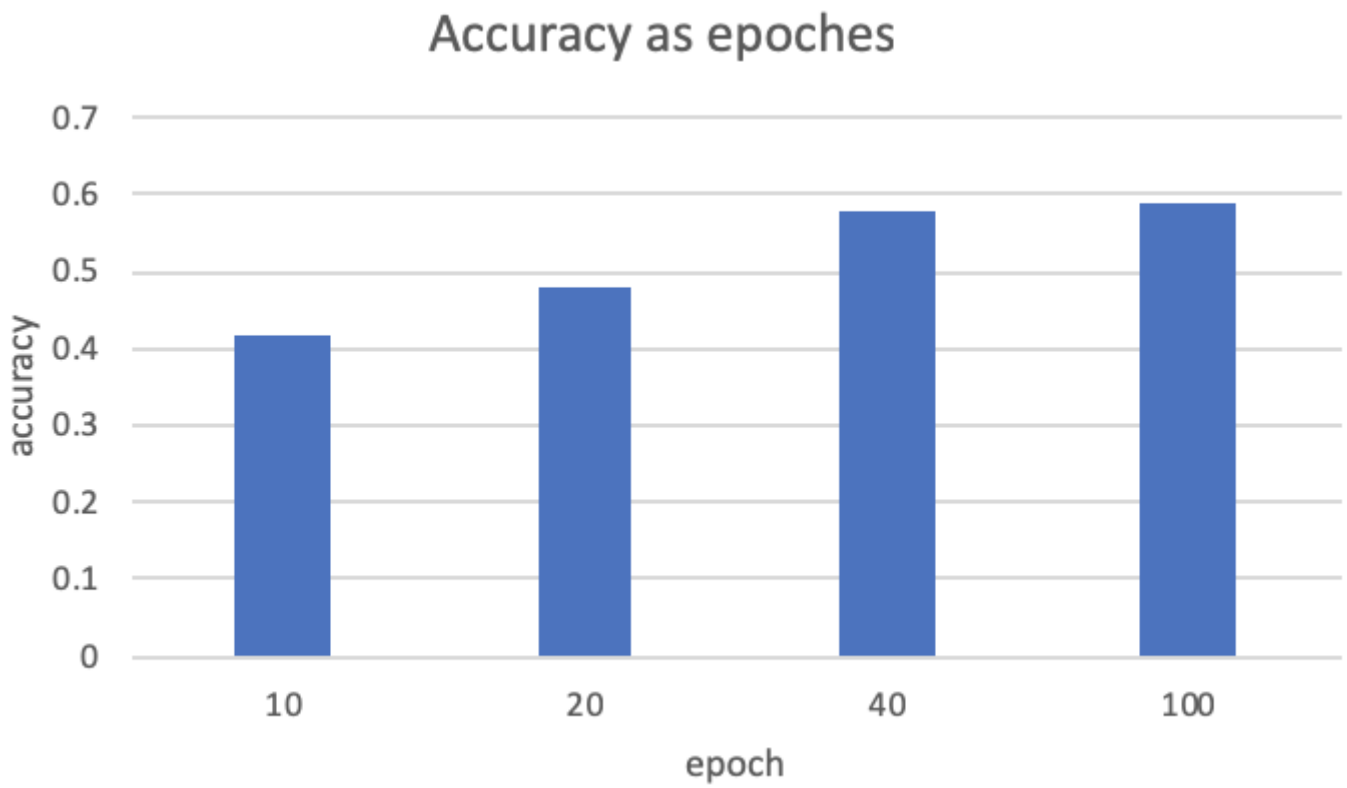
Method

Similar to CNN classifier but replace model with RNN.

input layer-->lstm layer-->softmax layer

Result

At first, I tune the word embeding dimension, LSTM layers and hidden dimension. I found the 1024 embeding dimension, 128 hidden dimension and 4 LSTM layers is the best. Then I tune trainer epochs by 10,20,40,and 100. I got following picture. Because of lack of GPU, I cannot test larger epoch, though I think larger epoch may get better result.
Accuracy peek at 0.58623.



Comments

1. Compatibility: during using fastNLP, I found that it is compatible with pytorch so users have little learning cost and can use fastNLP without changing any exisiting codes.
2. Detailed Comments: code comments detailed. Good comments give us opportunity to learn source code and can make reusing codes easy.
3. Convenience: a lot of functions can release developers' burden of writing non-critical codes(such as visualization of train process) so that they can focus on main funciton, such as

trainer.

4. Chinese feature: fastNLP facilitates us with tools to process chinese by deep learning while other foreign NLP frameworks cannot. As a framework developed by Chinese developer, it is important.
   However, I want to give some little tiny suggestion: when I use trainer, I found that I have to define extra functions like predict in my model and have to return a certain dict. In this way, we have to change existing codes have to use trainer. It may be better to develop a trainer that can be used without limitation
   In summary, it is quite good a framework