EEL6935 Programming HW2

Natural language processing (NLP)

Summary

The goal of this assignment is to do sentiment analysis by implementing and training a basic neural network in Python. The result show the accuracy of this basic neural network is not satisfactory (barely above 30%). This shows that sentiment analysis requires more complicated machine learning system.

Environment Setup

The project is setup in Python 2.x environment, fortunately, the package comes with a requirements.txt file showing all required python modules. After installing python with Anaconda, the setup can be easily done by simply running:

```
> pip install -r requirements.txt
```

In terminal, in my case, is Mac OS terminal running zsh.

Get Dataset

The code package doesn't comes with dataset, but a shell script to get them instead. Downloading the dataset can be done by running:

```
> cd big_data/datasets
> bash ./get_datasets.sh
```

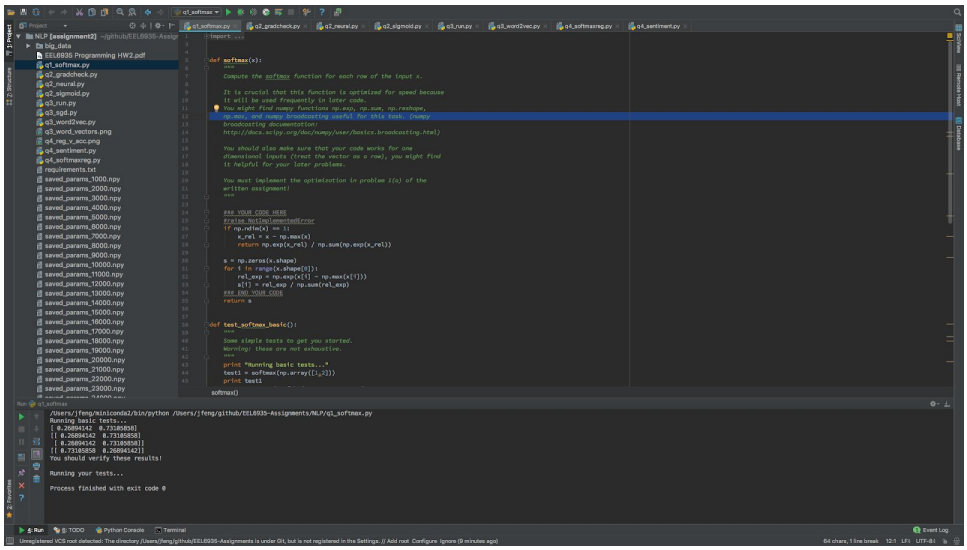
Implement and Training Neural Network

Following the homework instruction, the unfinished python code can be done. The finished codes can be found in the homework code repository hosted on GitHub:

https://github.com/ufjfeng/EEL6935-Assignments/tree/master/NLP

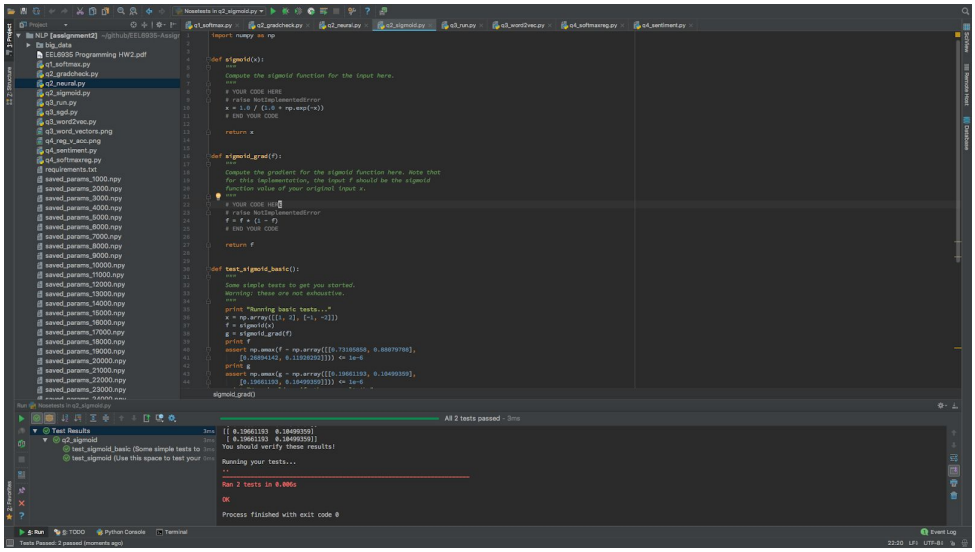
After finish each python file, the file is executed for testing purpose. The screenshots for each test result are:

q1_softmax.py



The implementation of soft max is simple, the tricky part is how to avoid the floating number overflow. The method I used here is calculate the difference between each number and the maximum one to reduce the number range.

q2_sigmoid.py



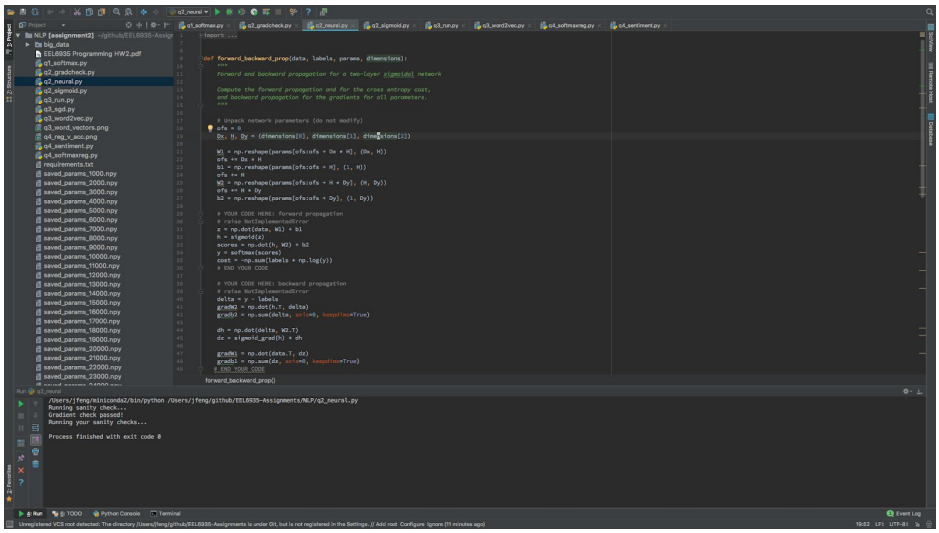
The sigmoid functions and its gradient is implemented in this file. Both of them can be implemented in a line of code

f = 1.0 / (1.0 + np.exp(-x))

g = f * (1 - f)

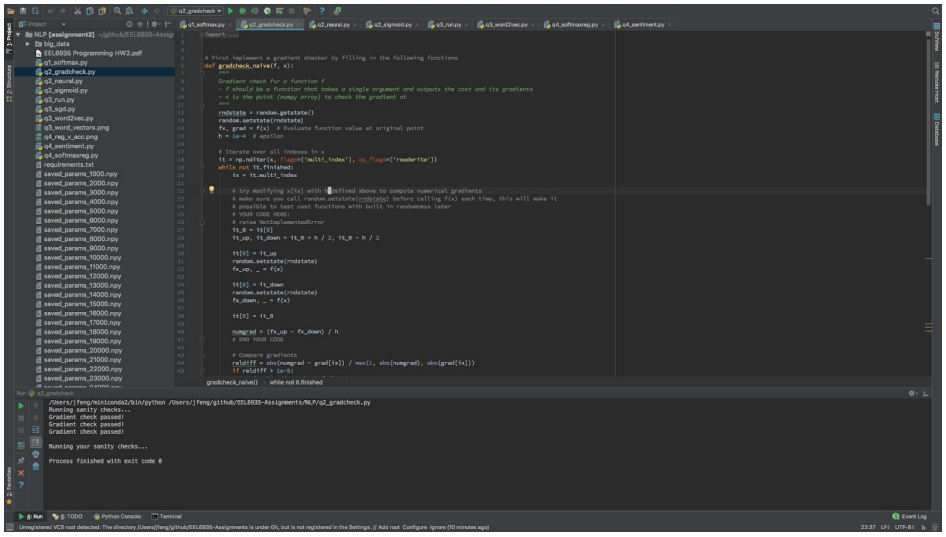
Where x is the input of sigmoid function and f is the function value, and g is the gradient of sigmoid function with value f.

q2_neural.py



The implementation of neural network serves as the core of this project. In this project, the neural network contains two layers with both forward and backward propagation. The function is implemented such that both cost and gradient can be calculated at the same time. Thanks to the python module numpy, the vector operation is much faster than the python built-in for loop.

q2_gradcheck.py



Gradient check plays important role in many machine learning system. In this part of the code, the partial gradient of each dimension has been tested in different method and checked for equality. The code used for calculating test gradient is:

it_0 = it[0]
it_up, it_down = it_0 + h / 2, it_0 - h / 2

it[0] = it_up
random.setstate(rndstate)
fx_up, _ = f(x)

it[0] = it_down
random.setstate(rndstate)
fx_down, _ = f(x)

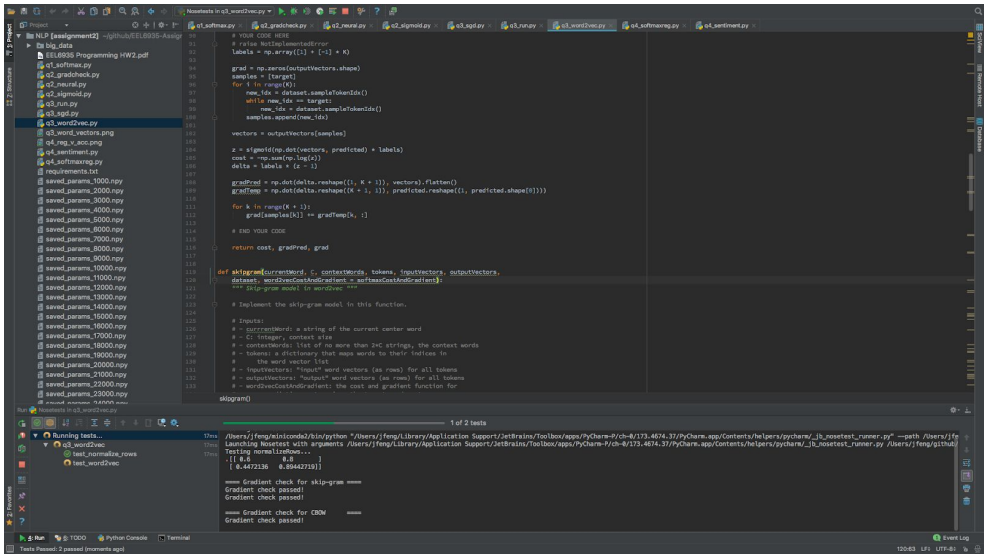
it[0] = it_0

numgrad = (fx_up - fx_down) / h

At the beginning of the q3_run.py, the PRNG is seeded as 314, so I added the same command in this code too.

random.seed(314)

q3_word2vec.py

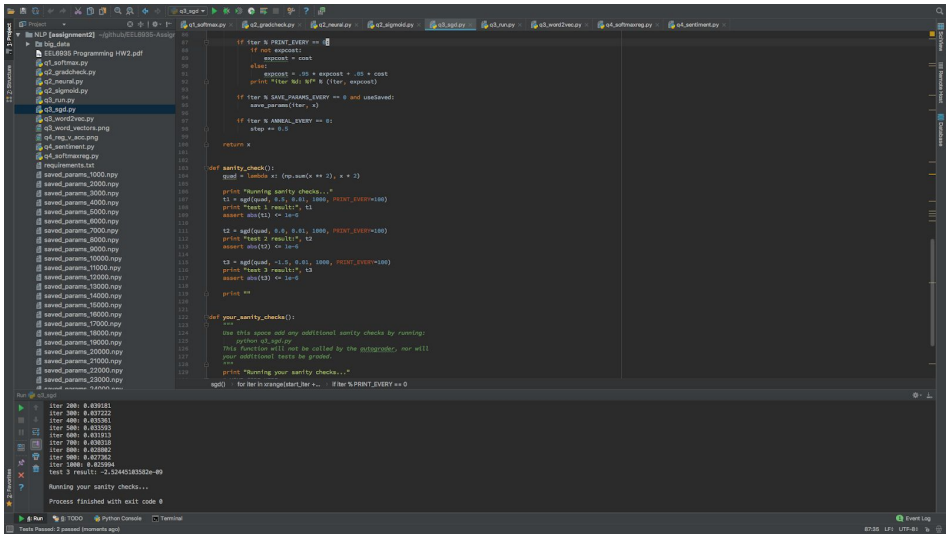


In this file, we need to implement three functions:

- softmaxCostAndGradient
- negSamplingCostAndGradient
- skipgram

In order to represent the word with vector and feed them into neural networks. Hence apply stochastic gradient descendent (SGD) algorithm on it.

q3_sgd.py



q3_run.py

For some reason yet to be discovered, q3_run.py won’t run with python directly and throws an error message:

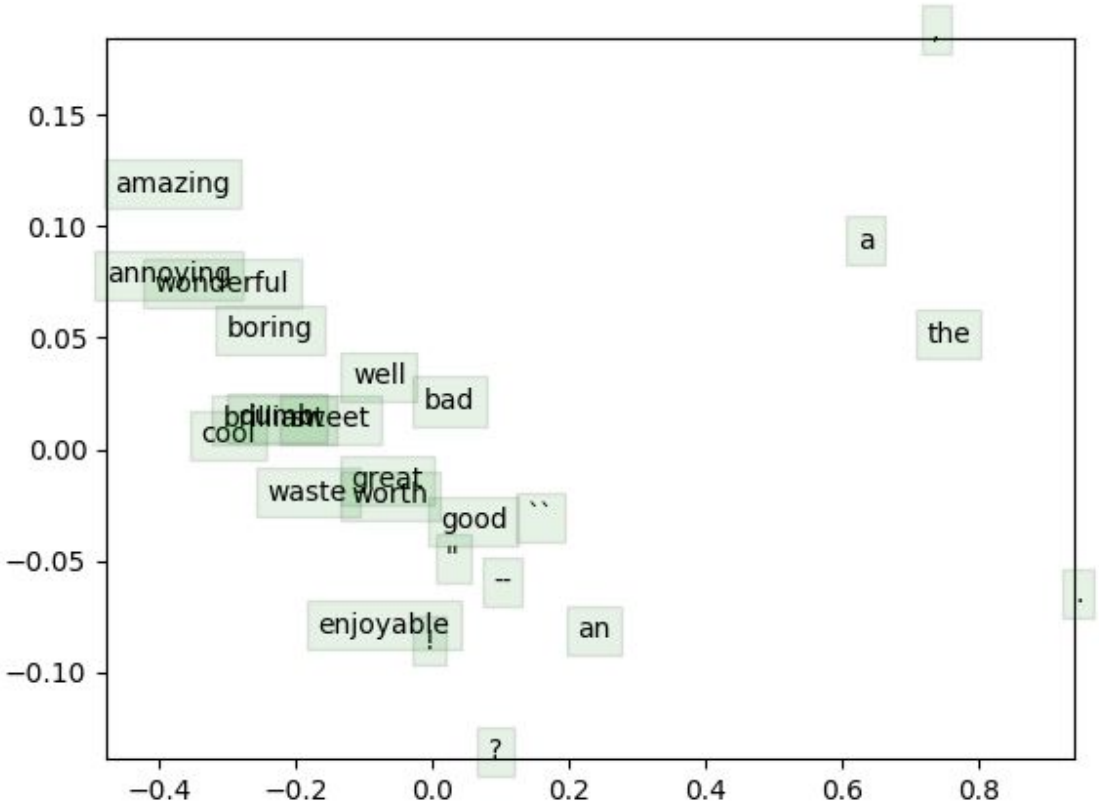
RuntimeError: Python is not installed as a framework. The Mac OS X backend will not be able to function correctly if Python is not installed as a framework. See the Python documentation for more information on installing Python as a framework on Mac OS X. Please either reinstall Python as a framework, or try one of the other backends. If you are using (Ana)Conda please install python.app and replace the use of 'python' with 'pythonw'. See 'Working with Matplotlib on OSX' in the Matplotlib FAQ for more information.

So this file has to be run in terminal.

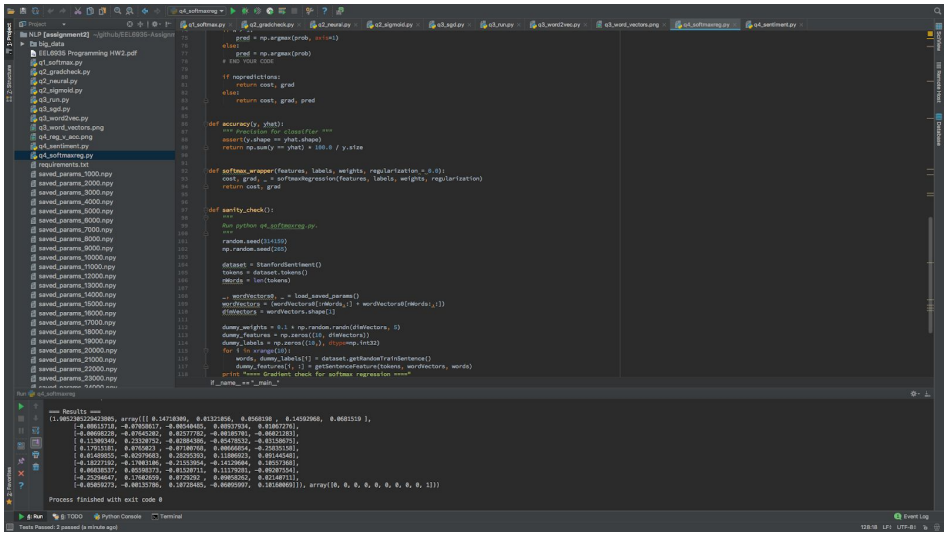
After implementing the neural network, we can train the network with downloaded dataset by running:

> pythonw q3_run.py

This will take a while since the training has 40,000 iterations and the neural network is implemented in python. After training, a plot indicating the training result of each word vector is shown as:



q4_softmaxreg.py



This part of the code convert a sentence into features vector (sentence featurizer) and a softmax regression.

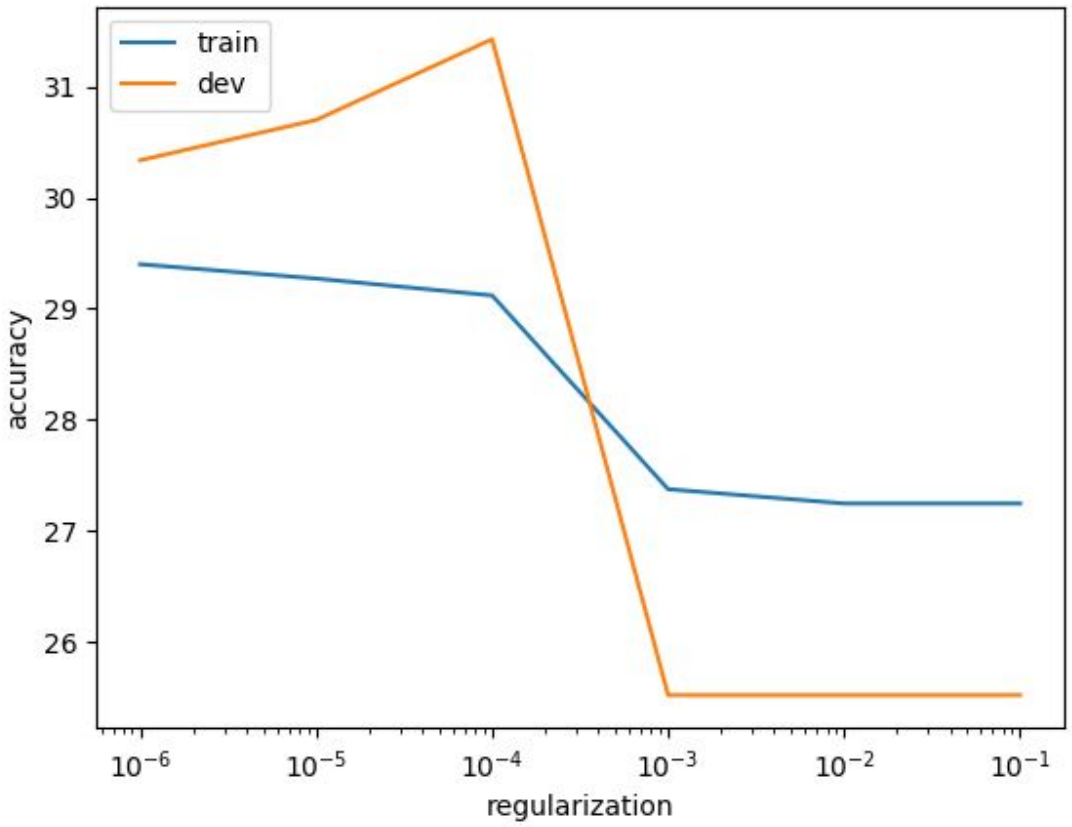
q4_sentiment.py

A proper machine learning system will not be done without the regularization term setup properly. In order to select the best regularization parameter, we ran a experiment in Q4 by testing the network performance with regularization coefficient vary from 1e-1 to 1e-6. This can be achieved by add a line:

REGULARIZATION = [10 ** -i for i in reversed(range(1, 7))]

Into q4_sentiment.py.

After another simulation of, the result of different regularization result is shown.



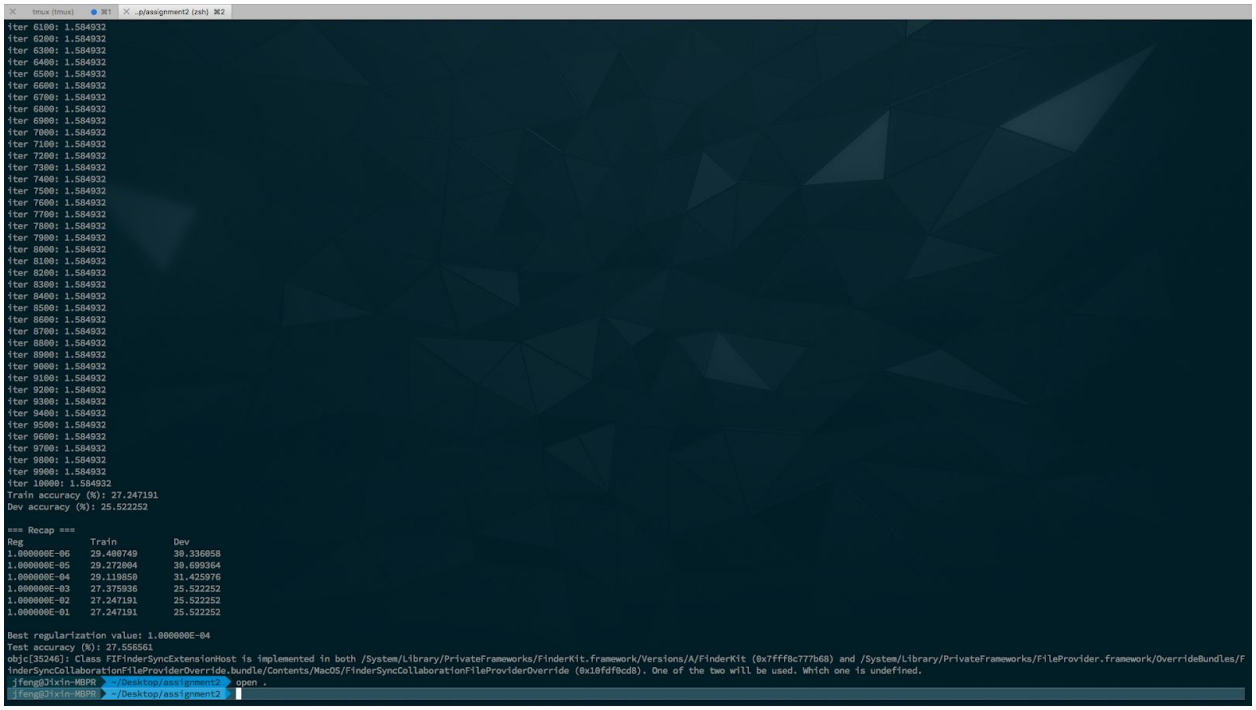
=== Recap ===

Reg	Train	Dev
1.000000E-06	29.400749	30.336058
1.000000E-05	29.272004	30.699364

1.000000E-04 29.119850 31.425976
1.000000E-03 27.375936 25.522252
1.000000E-02 27.247191 25.522252
1.000000E-01 27.247191 25.522252

Best regularization value: 1.000000E-04
Test accuracy (%): 27.556561

The plot and recap shows that 1e-4 seems to be a relative good choice for regularization since it produces the best result for sentiment analysis.



Analyze

With the help of numpy implementation, the vector operation used for neural networks can be done much faster than the build-in python methods, but due to the limitation of python Global Interpreter Lock (GIL), only one thread can be executed by CPU for each python process. And the vector operation can get huge performance boost from parallelization. Hence the need for GPU is obvious. A implementation in TensorFlow or Caffe should have much better performance. And the result also shows the naive neural network implementation only provides about 30% of accuracy, which is even worse than flipping a coin (50%). So an applicable sentiment analysis system should be built based on more complicated neural networks, for example CNN or RNN.