CSI 5138
Introduction: Deep Learning & RL

Homework 3

Date: 2020/11/01

NAME: Yunfeng Li

Student Number: 300139547

Table of Contents

Experiment Description

The purpose of this experiment is to design two different classifiers for IMDB dataset. The experiment is carried out per the procedures below.

1. Understand the process the data.
2. Develop two text classification models using Vanilla RNN and LSTM for the dataset.
3. Examine five different values of state dimension.
4. Compare and summarize through results

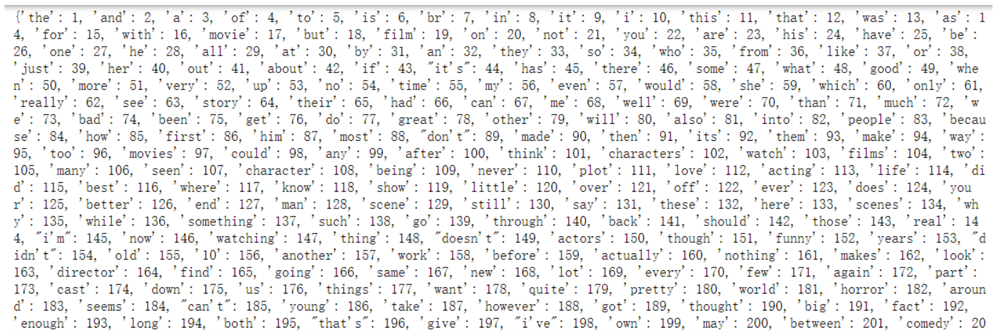
Results Analysis

This section will be separated into three parts. First, we will introduce the data processing. Then two classification models are developed, and through different state dimensions, we will explore which state dimension outperforms to others. Lastly, we will summarize the results we found.

Data processing

IMDB dataset is labeled data which consists of 50000 movie reviews. It is specially selected for natural language processing studying. The sentiment of reviews is binary, which means that there are only two types of labels: positive and negative. We read the file into variables, and give them labels as positive 0 and negative 1.

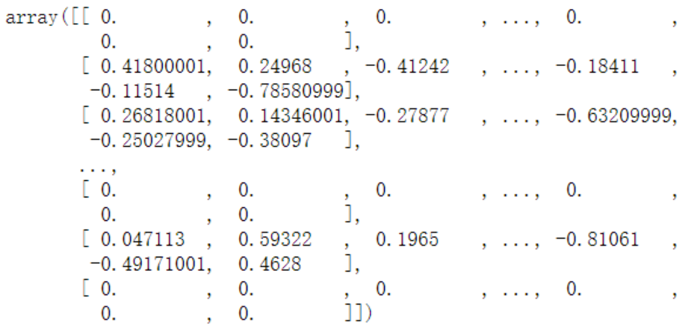
After having the data, we tokenize reviews with keras' Tokenize function. Here we found totally 124252 words. The word dictionary can also be obtained. The word embedding vector is constructed by the Glove data. Pre-trained word vectors differ in feature dimension. In order to reduce the time of computation and in consideration of limitation of computer RAM, we choose 50d for constructing vector. According to word dictionary as below.



```
{'the': 1, 'and': 2, 'a': 3, 'of': 4, 'to': 5, 'is': 6, 'br': 7, 'in': 8, 'it': 9, 'i': 10, 'this': 11, 'that': 12, 'was': 13, 'as': 1
4, 'for': 15, 'with': 16, 'movie': 17, 'but': 18, 'film': 19, 'on': 20, 'not': 21, 'you': 22, 'are': 23, 'his': 24, 'have': 25, 'be':
26, 'one': 27, 'he': 28, 'all': 29, 'at': 30, 'by': 31, 'an': 32, 'they': 33, 'so': 34, 'who': 35, 'from': 36, 'like': 37, 'or': 38,
'just': 39, 'her': 40, 'out': 41, 'about': 42, 'if': 43, "it's": 44, 'has': 45, 'there': 46, 'some': 47, 'what': 48, 'good': 49, 'whe
n': 50, 'more': 51, 'very': 52, 'up': 53, 'no': 54, 'time': 55, 'my': 56, 'even': 57, 'would': 58, 'she': 59, 'which': 60, 'only': 61,
'really': 62, 'see': 63, 'story': 64, 'their': 65, 'had': 66, 'can': 67, 'me': 68, 'well': 69, 'were': 70, 'than': 71, 'much': 72, 'w
e': 73, 'bad': 74, 'been': 75, 'get': 76, 'do': 77, 'great': 78, 'other': 79, 'will': 80, 'also': 81, 'into': 82, 'people': 83, 'becau
se': 84, 'how': 85, 'first': 86, 'him': 87, 'most': 88, "don't": 89, 'made': 90, 'then': 91, 'its': 92, 'them': 93, 'make': 94, 'way':
95, 'too': 96, 'movies': 97, 'could': 98, 'any': 99, 'after': 100, 'think': 101, 'characters': 102, 'watch': 103, 'films': 104, 'two':
105, 'many': 106, 'seen': 107, 'character': 108, 'being': 109, 'never': 110, 'plot': 111, 'love': 112, 'acting': 113, 'life': 114, 'di
d': 115, 'best': 116, 'where': 117, 'know': 118, 'show': 119, 'little': 120, 'over': 121, 'off': 122, 'ever': 123, 'does': 124, 'you
r': 125, 'better': 126, 'end': 127, 'man': 128, 'scene': 129, 'still': 130, 'say': 131, 'these': 132, 'here': 133, 'scenes': 134, 'wh
y': 135, 'while': 136, 'something': 137, 'such': 138, 'go': 139, 'through': 140, 'back': 141, 'should': 142, 'those': 143, 'real': 14
4, "i'm": 145, 'now': 146, 'watching': 147, 'thing': 148, "doesn't": 149, 'actors': 150, 'though': 151, 'funny': 152, 'years': 153, "d
idn't": 154, 'old': 155, '10': 156, 'another': 157, 'work': 158, 'before': 159, 'actually': 160, 'nothing': 161, 'makes': 162, 'look':
163, 'director': 164, 'find': 165, 'going': 166, 'same': 167, 'new': 168, 'lot': 169, 'every': 170, 'few': 171, 'again': 172, 'part':
173, 'cast': 174, 'down': 175, 'us': 176, 'things': 177, 'want': 178, 'quite': 179, 'pretty': 180, 'world': 181, 'horror': 182, 'aroun
d': 183, 'seems': 184, "can't": 185, 'young': 186, 'take': 187, 'however': 188, 'got': 189, 'thought': 190, 'big': 191, 'fact': 192,
'enough': 193, 'long': 194, 'both': 195, "that's": 196, 'give': 197, "i've": 198, 'own': 199, 'may': 200, 'between': 201, 'comedy': 20
```

Figure 1

The word embedding vector consists of 124252 words and each word has 50 dimensions which is placed in the list structure of Python. This word embedding vector is designed as below. Note that there are many rows only have zeros, since this word do not exist in Glove pre-trained vectors.



```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.41800001,  0.24968   , -0.41242   , ..., -0.18411   ,
        -0.11514   , -0.78580999],
       [ 0.26818001,  0.14346001, -0.27877   , ..., -0.63209999,
        -0.25027999, -0.38097   ],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.047113  ,  0.59322   ,  0.1965    , ..., -0.81061   ,
        -0.49171001,  0.4628    ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

Figure 2

In IMDB reviews data, each document has a movie review. Therefore, length of different documents varies completely. Sometimes there are only several words, however some documents have more than 1000 words. In order to reduce this dimension, we need to set a common standard of word length for these documents. The methods of deciding this length we use in this experiment is considering mode and visualizing the data.

In this dataset, we find the mode is 123. Nevertheless, we know that most documents have different length. It is rarely seen that many documents have same length. So, we visualize the data and try to decide this length. The sequence length is plotted as below.

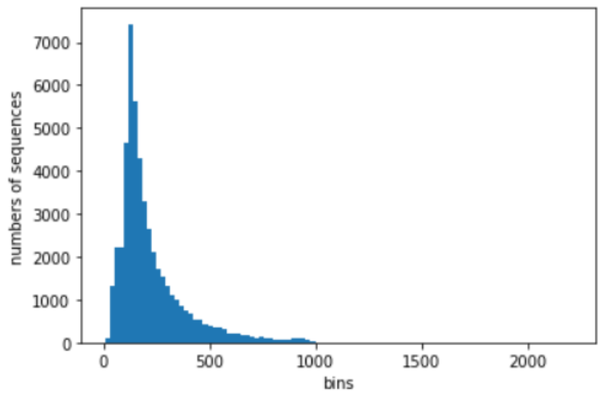


Figure 3

From the diagram above, we can find most documents are less than 300. In order to contain more words, we choose the max length of sequence is 500. Note that for those documents which have words less than 500, we will fulfill the sample with zeros.

Model Designing

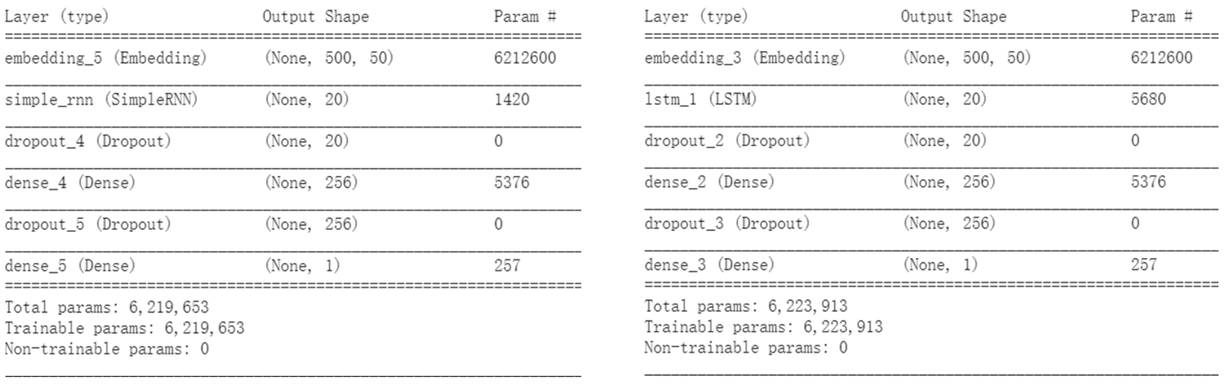
Classification models in this experiment is developed with Keras library. In this assignment, we are required to design two text classifiers which are Vanilla RNN and LSTM. These models are set up with Sequential function. There are some parameters set in models.

The parameters, which are commonly used in these two models, are shown in the table below.

Table 1 Parameters

Parameters	Value
Max Sequence Length	500
Epoch	10
Validate Split Rate	0.1
Loss Function	binary crossentropy
Optimizer	rmsprop
Gating Function	Sigmoid
Dropout Probility	0.1

The structure of Vanilla RNN and LSTM is shown below.



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 500, 50)           6212600
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 20)                1420
_________________________________________________________________
dropout_4 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               5376
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 257
=================================================================
Total params: 6,219,653
Trainable params: 6,219,653
Non-trainable params: 0
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 500, 50)           6212600
_________________________________________________________________
lstm_1 (LSTM)                (None, 20)                5680
_________________________________________________________________
dropout_2 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               5376
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 6,223,913
Trainable params: 6,223,913
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Vanilla RNN (left) LSTM (right)

Conclusion

In this section, we will analyze the results in two parts. Firstly, comparing the performance of models, and then we will explore the performance of the state dimension.

Model Comparison

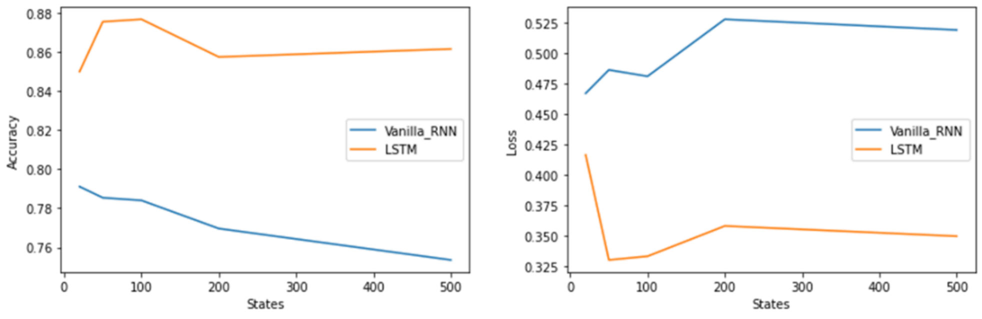
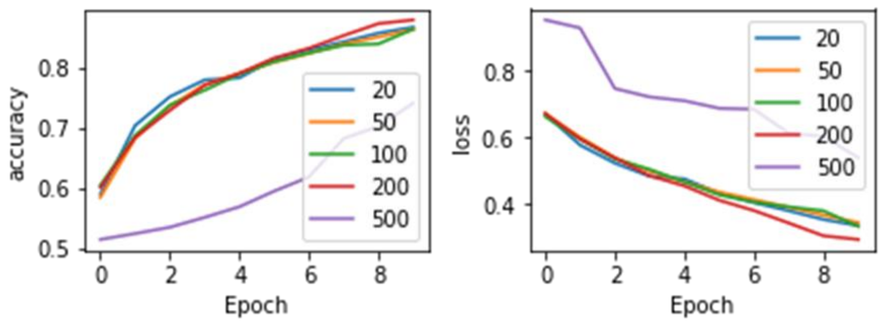


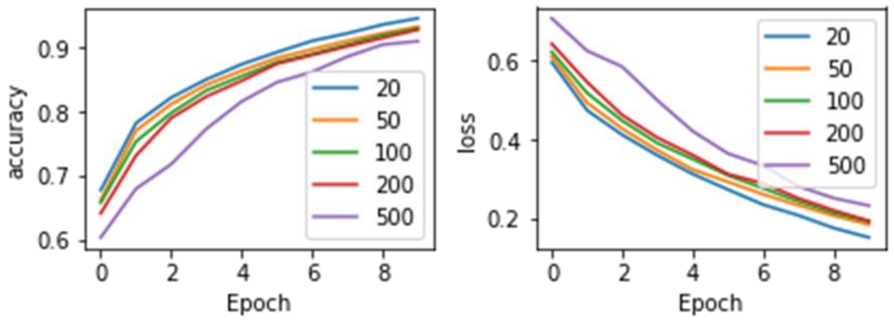
Figure 5. Accuracy (left) and Loss (right)

The diagrams above are the test accuracy and loss of two models in different models' complexity. Through the results of images, it is obvious that the LSTM performs better than the Vanilla RNN no matter in accuracy or loss. This conclusion is applicable in all state dimensions. It makes sense since the LSTM is generally outperforming the Vanilla RNN.

Performance of Different State Dimension



Vanilla RNN



LSTM

Figure 6. Different Dimension of Vanilla RNN and LSTM

It is clearly shown that training accuracy is not always better, with the state dimension growing larger. Besides, the training speed is rarely related to state dimension. We can also find that when state dimension is 500, the result is even worse than 20.

But we can still find that when state dimension increase from 20 to 200, accuracy of the Vanilla RNN also increase. For this experiment, the state dimension, which is 200, has the best performance. For the LSTM, the results of state dimension from 20 to 200 are nearly same.

The results above is analyzed based on results of training set. In order to find a precise conclusion, we also need to consider the test set which is shown in Figure 5. According to the diagram, we can find that the state dimension from 20 to 100 of Vanilla RNN is optimal solutions. For LSTM model, it is 50 that outperforms others.

Summary

In this assignment, we design two models which are Vanilla RNN and LSTM. We compare the performance of each models, and we find that LSTM generally outperforms the Vanilla RNN in each different state dimensions which are required. In order to find the influence of different state dimension, we set it with [20, 50, 100, 200, 500]. We find that the optimal solution exists for this dataset. For Vanilla RNN, they are from 20 to 100, for LSTM, it is 50. Therefore, it is concluded that the state dimension will influence the results, and there will be optimal solutions for the data.