2019 Spring COM526000 Deep Learning - Homework 3 Report

Recurrent Neural Network

105061110 周柏宇

Problem 1

(a) Before feeding samples into the model, you have to further transform the words (now they are encoded as integer indices) to vector representations. We call this procedure Embedding. Explain why do we need embedding.

If we directly feed the integers into the network, it implies that words with similar integer will have similar meaning, which we don't intend to do that. Instead, we can use one-hot encoding to represent our integers, since they all have the same distance to one another. However, in our case, the vector will have 10000 dimensions, the number of parameters to train will be gigantic. Therefore, we resort to learning a matrix that transform our 10000 dimensions to the embedded dimensions. In my design, I choose 256, which is a lot less than 10000.

(b) Explain the idea of gated models and the main differences between GRU and LSTM.

In vanilla RNN, we always take the hidden state as input and store the output of the unit back to hidden state, which means that we always 'remember' all the information in the past. In some cases, it might be unnecessary and harmful to the inference. One of the solution is adding parameters to control whether we want to input the hidden state, forget the hidden state, or input the output of previous unit. This is the idea of LSTM, it has input gate, forget gate and output gate. While the GRU uses only one gate, the update gate, to represent the input gate and forget gate in LSTM. The update gate controls the proportion of previous hidden state and current hidden state.

(c) Train a vanilla RNN (simple RNN) first to do the binary classification task. Specify the model structure in your report clearly (don't forget the embedding layer!). You have to evaluate the performance by test accuracy. Also, you have to plot learning curve as in previous homework, receiver operating characteristic curve (ROC, as shown in Figure 1) and precision-recall curve (PRC, as shown in Figure 2) with their area-under-curve

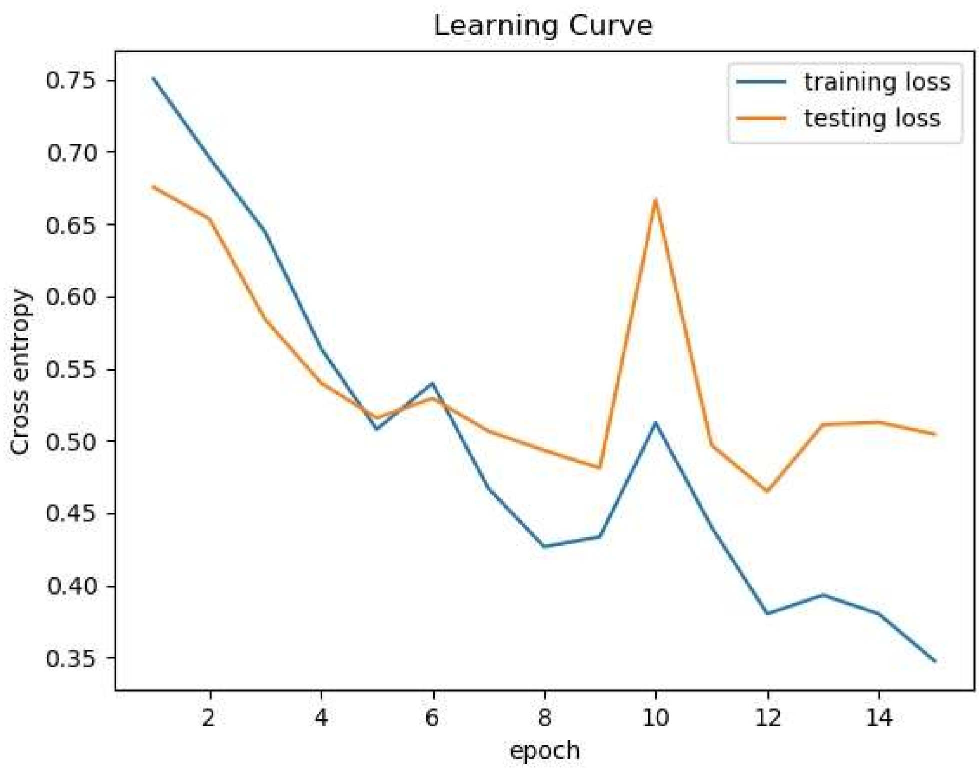
(AUROC and AUPRC). These two metrics are useful while evaluating model performance.

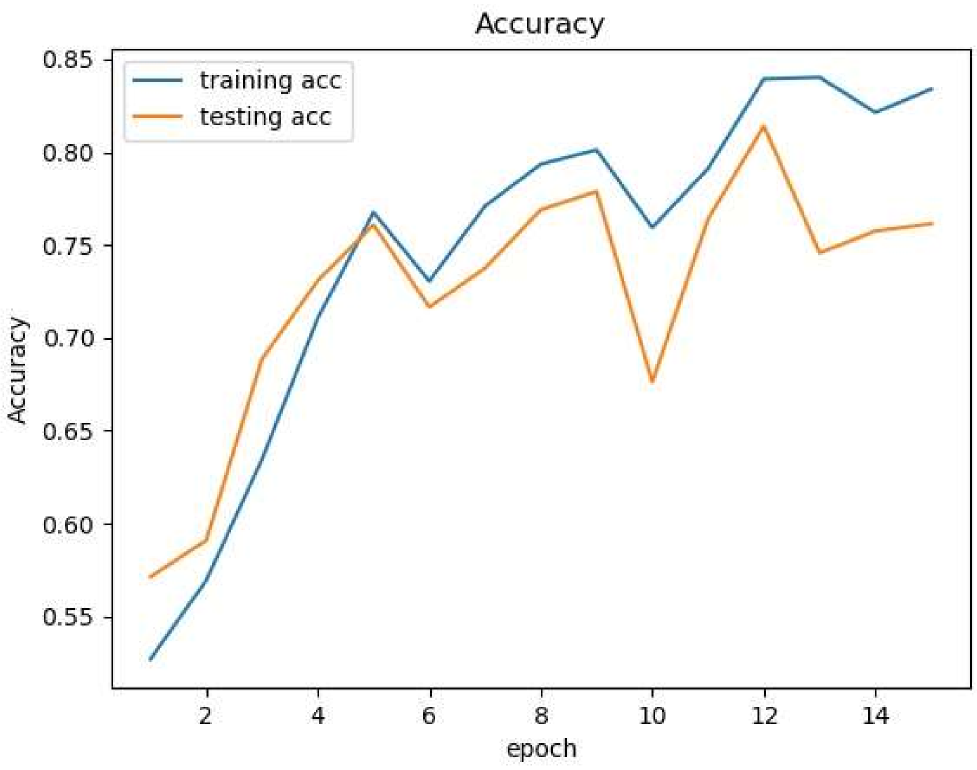
Network structure of the simple RNN:

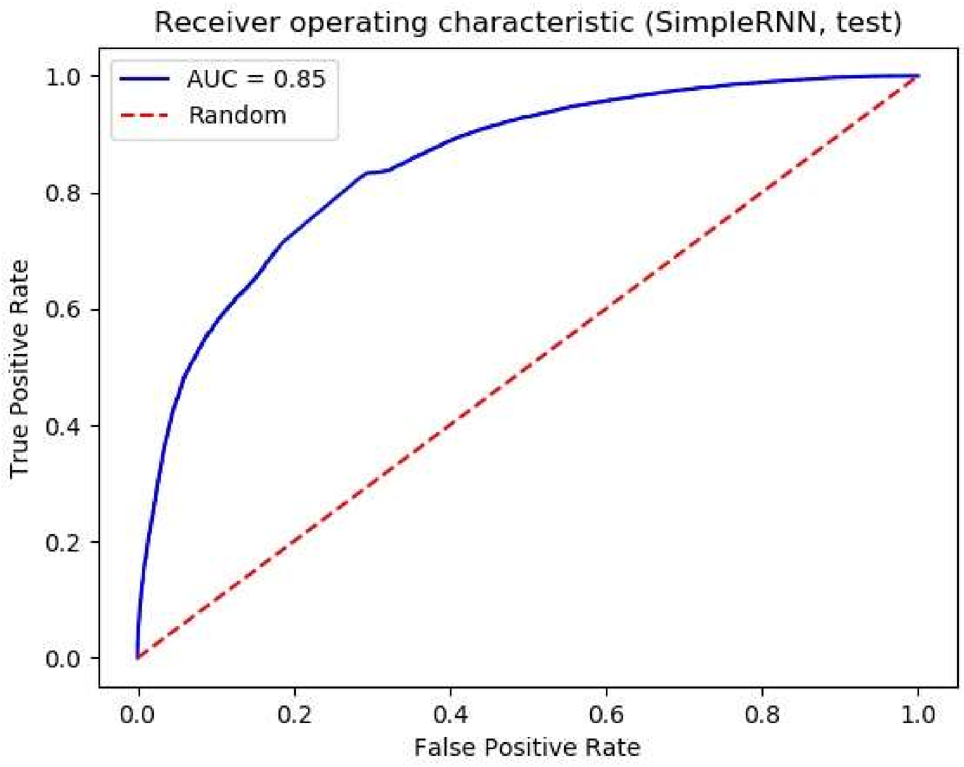
- Embedding layer:
    - Embedded dimension: from 10000 to 256
- RNN:
    - Layers: 1
    - Hidden units: 128
    - Activation functions: tanh
    - Drop-out rate: 0.5
- FCNN:
    - Layers: 1

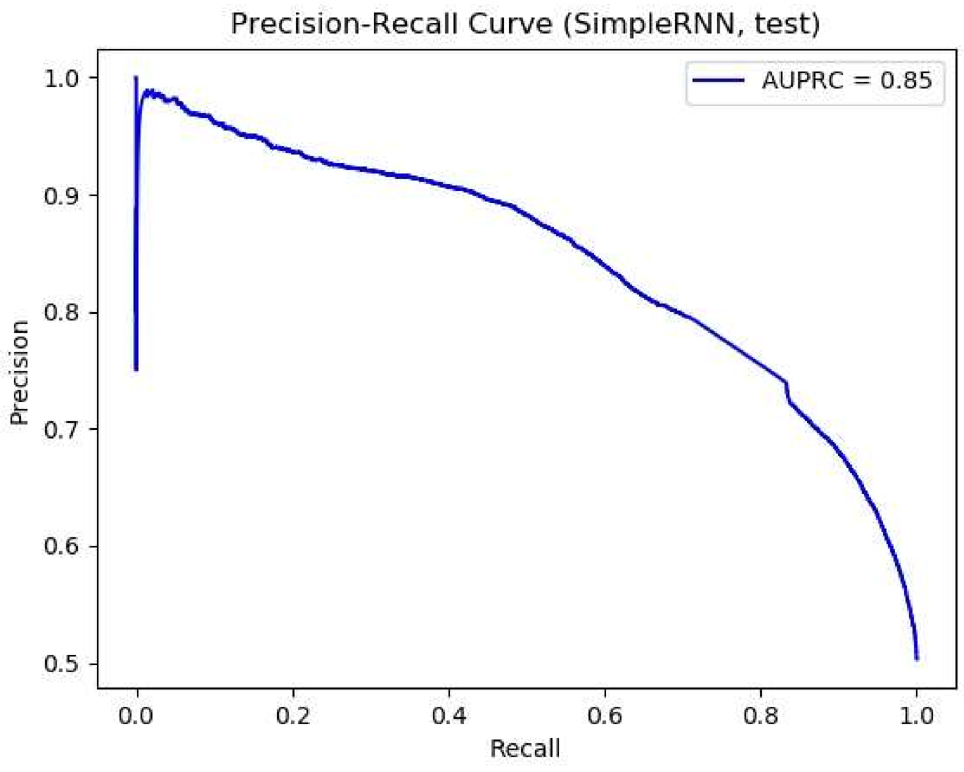
Training hyper parameters:

- Epochs: 15
- Batch size: 500
- Optimizer: Adam
- Learning rate: 0.0005




Accuracy
training acc
testing acc
Accuracy
epoch
0.85
0.80
0.75
0.70
0.65
0.60
0.55
2
4
6
8
10
12
14


Receiver operating characteristic (SimpleRNN, test)
AUC = 0.85
Random
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0



(d) Based on the definition of ROC, explain why random guess forms the red dotted line in Figure 1.

Suppose we guess the label to be positive with probabilities *r* and the positive label contribute p% to the whole dataset. Then the true positive rate is $TPR = \frac{rp}{p} = r$, the false positive rate is $FPR = \frac{r(1-p)}{1-p} = r$. Thus the true positive rate is always the same as the false positive rate.

(e) What curve would random guess form in the PRC case?

Suppose we guess the label to be positive with probabilities *r* and the positive label contribute p% to the whole dataset. Then the precision is $Precision = \frac{rp}{r} = p$, the recall is $Recall = \frac{rp}{p} = r$. Thus the result will be a constant line at *y*=*p*.

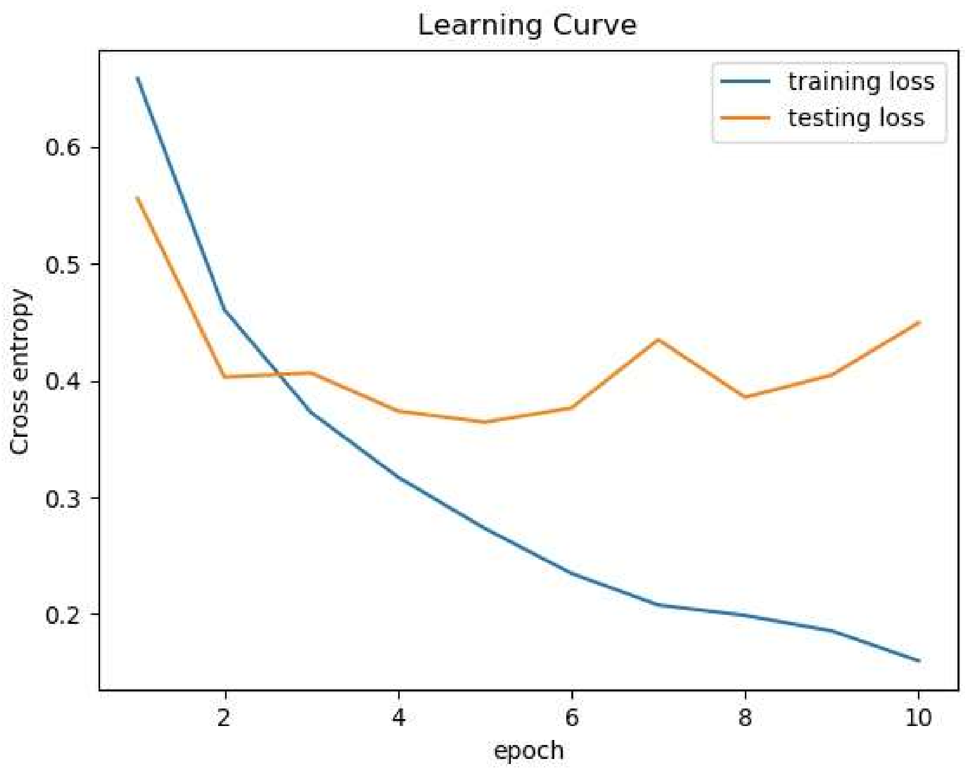
(f) Repeat (c) with GRU and LSTM.

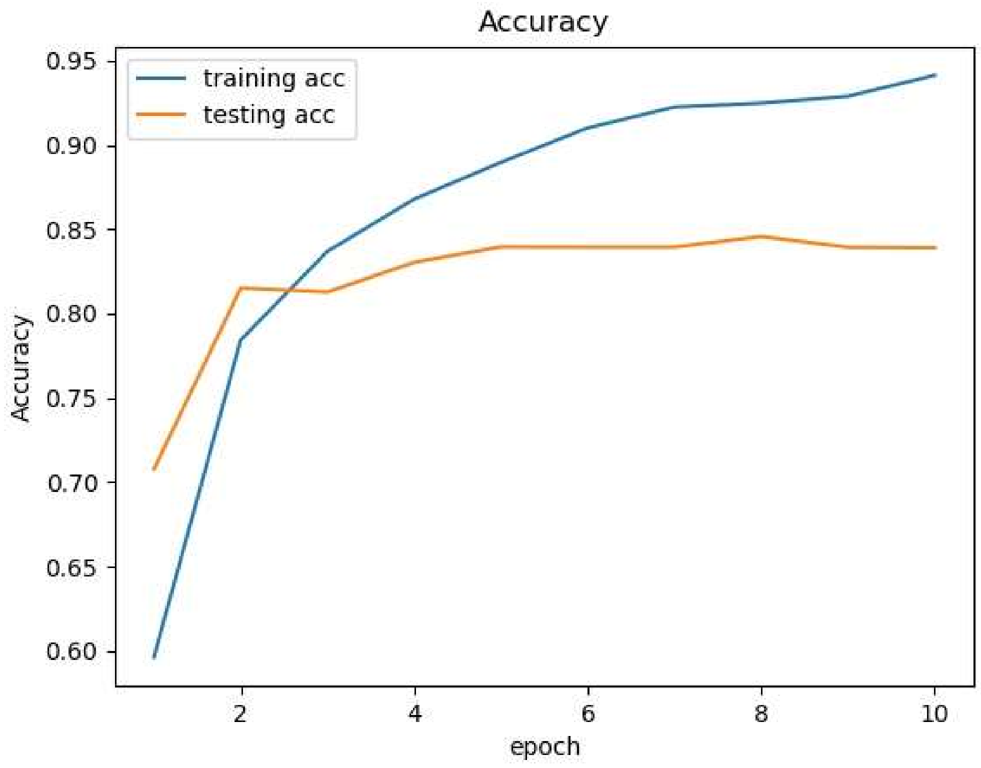
Network structure of the GRU:

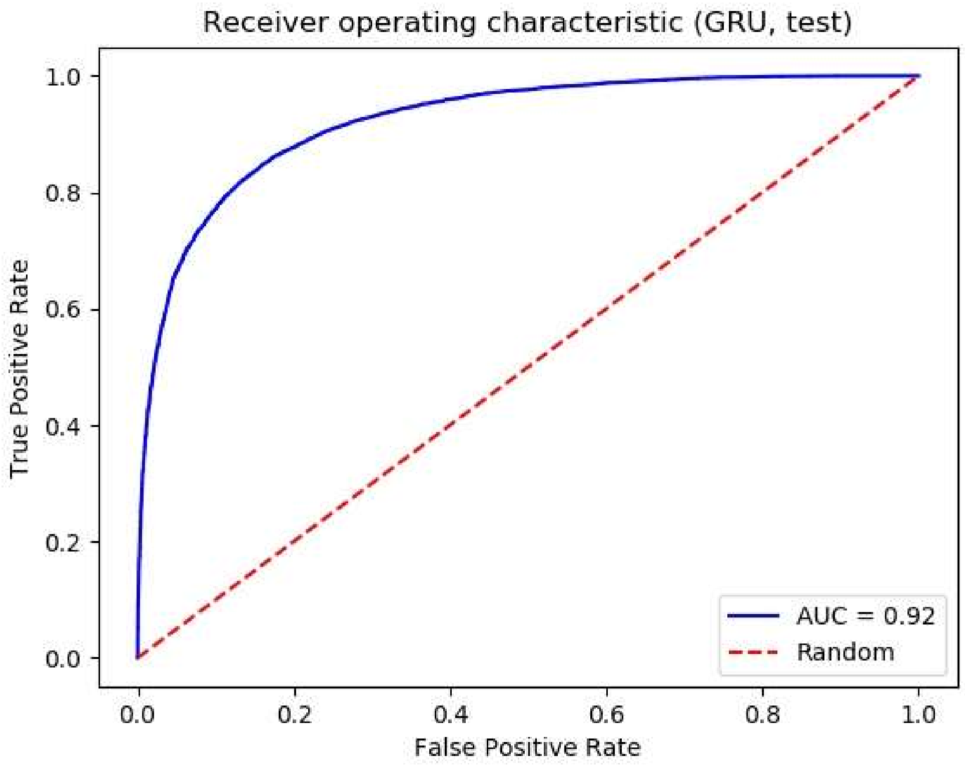
- Embedding layer:
  - Embedded dimension: from 10000 to 256
- RNN:
  - Layers: 1
  - Hidden units: 128
  - Activation functions: tanh
  - Drop-out rate: 0.5
- FCNN:
  - Layers: 1

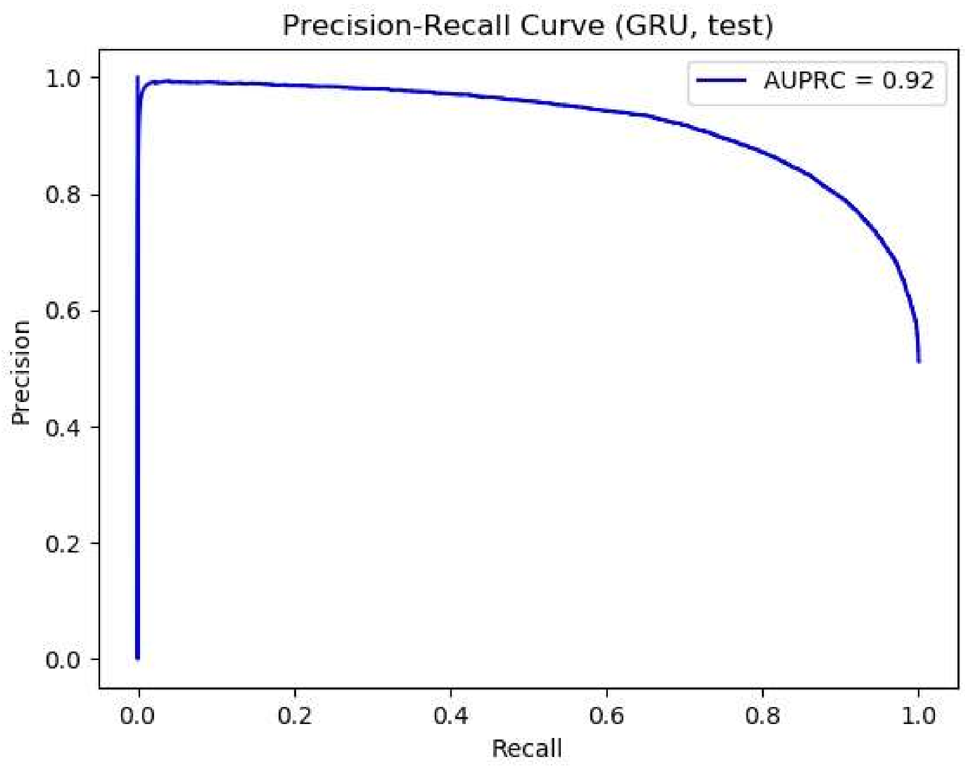
Training hyper parameters:

- Epochs: 10
- Batch size: 250
- Optimizer: Adam
- Learning rate: 0.0005




Accuracy
training acc
testing acc
Accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
2
4
6
8
10
epoch


Receiver operating characteristic (GRU, test)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.92
Random

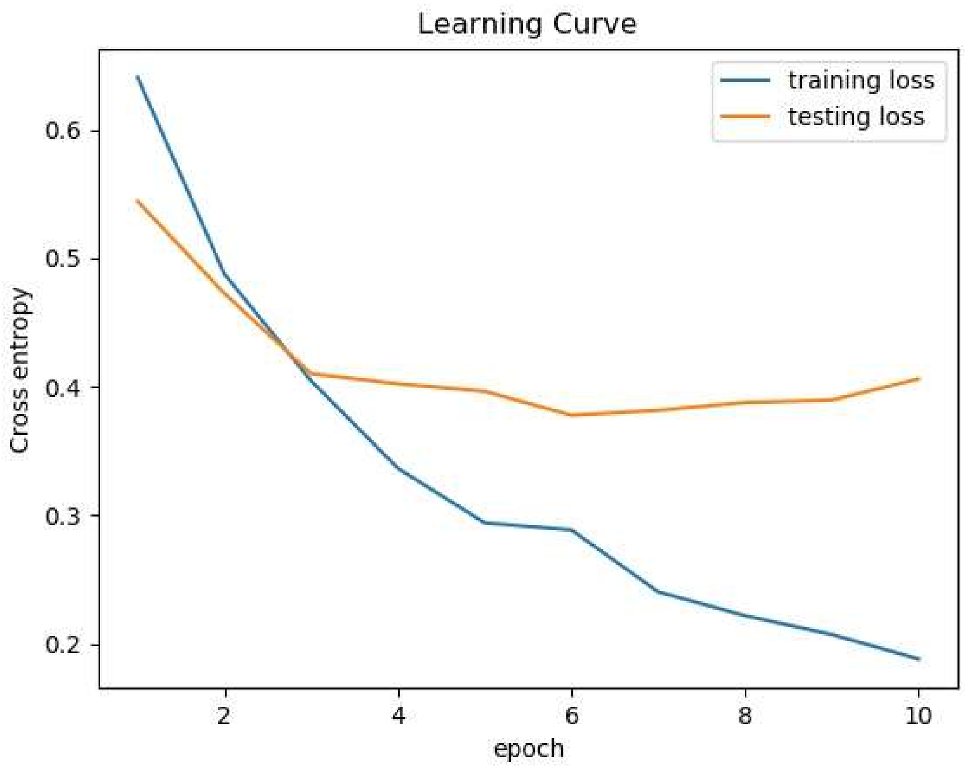


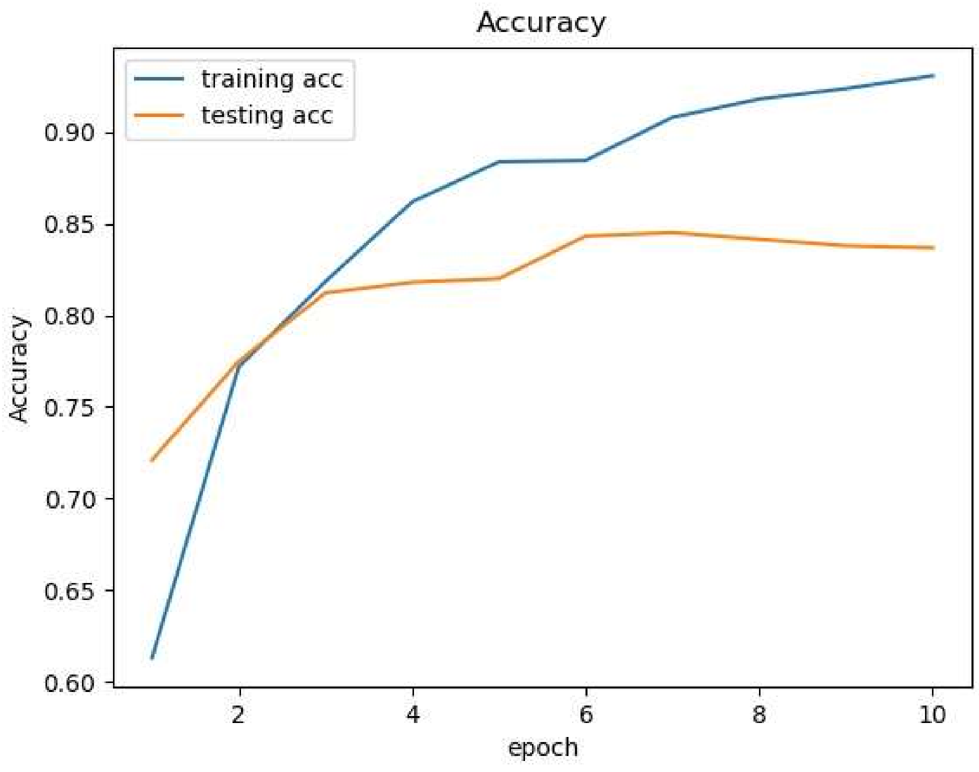
Network structure of the LSTM:

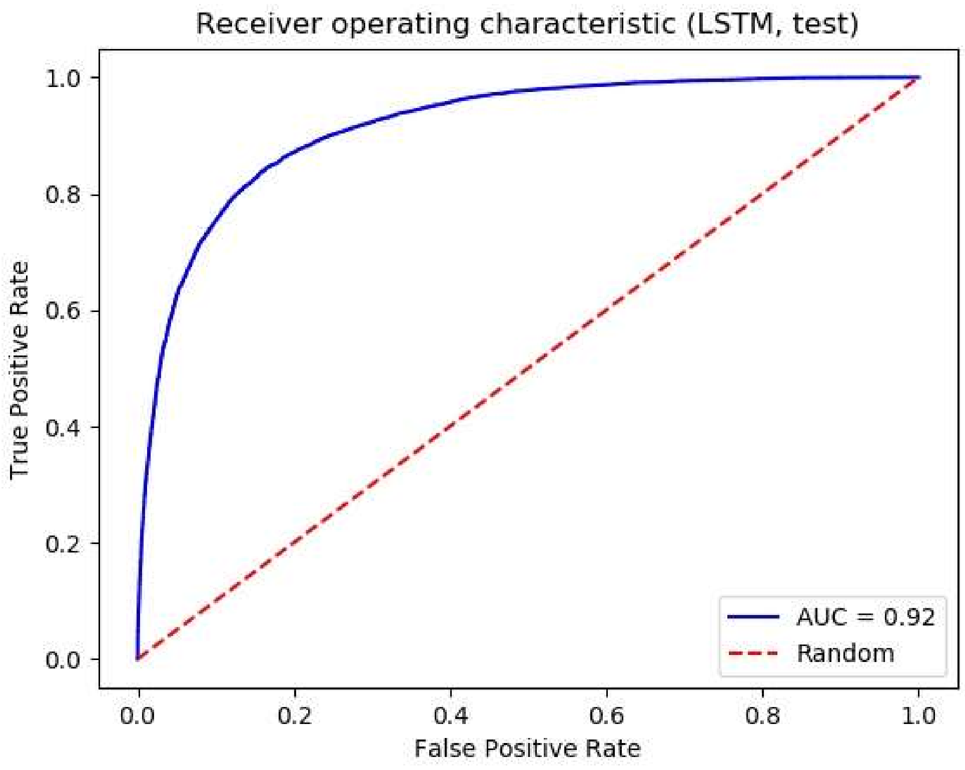
- Embedding layer:
  - Embedded dimension: from 10000 to 256
- RNN:
  - Layers: 1
  - Hidden units: 128
  - Activation functions: tanh
  - Drop-out rate: 0.5
- FCNN:
  - Layers: 1

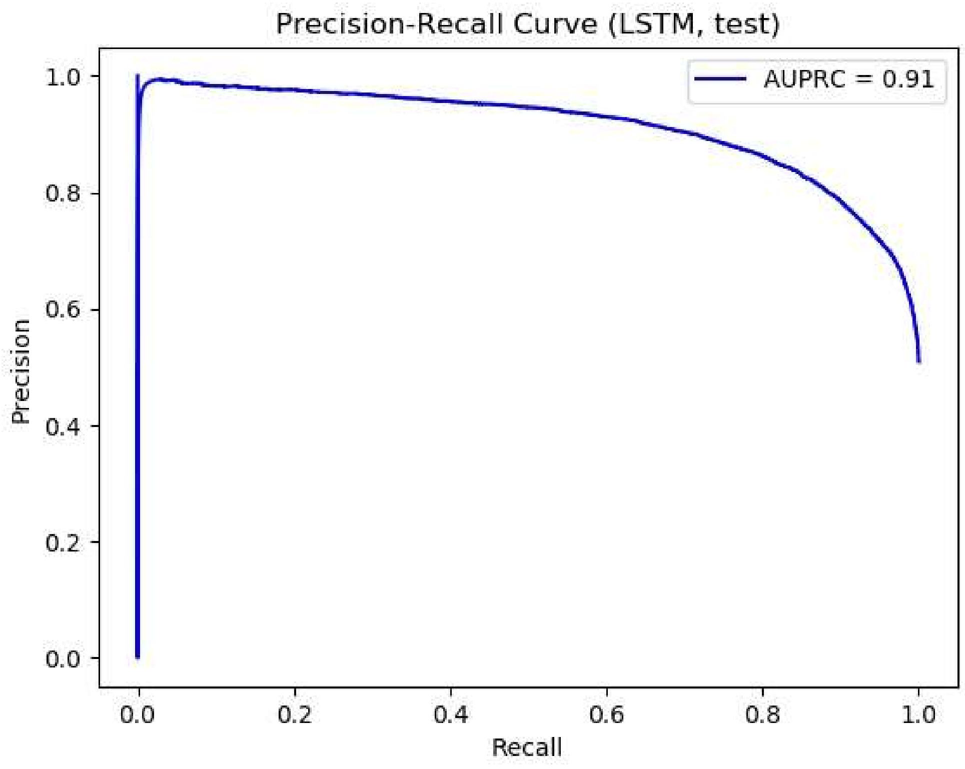
Training hyper parameters:

- Epochs: 10
- Batch size: 250
- Optimizer: Adam
- Learning rate: 0.0005


Learning Curve
training loss
testing loss
Cross entropy
0.6
0.5
0.4
0.3
0.2
2
4
6
8
10
epoch


Accuracy
training acc
testing acc
Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
2
4
6
8
10
epoch





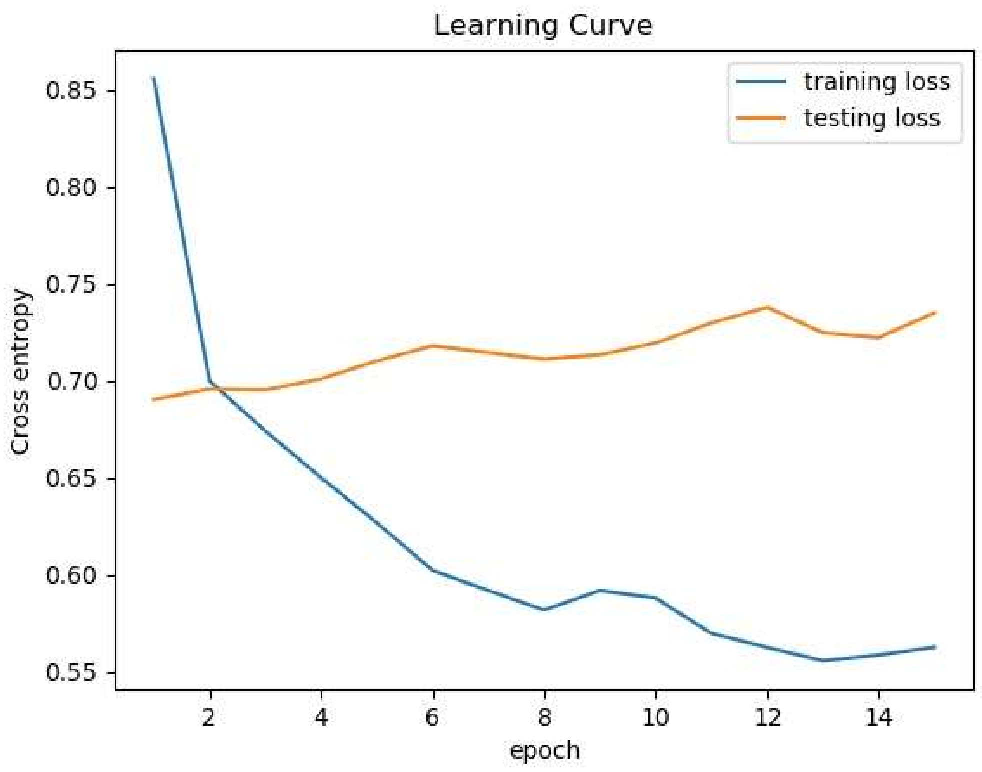
(g) Change the maximum length of each review from 120 words to 256 words in (c) and (f). State your observations and comments.

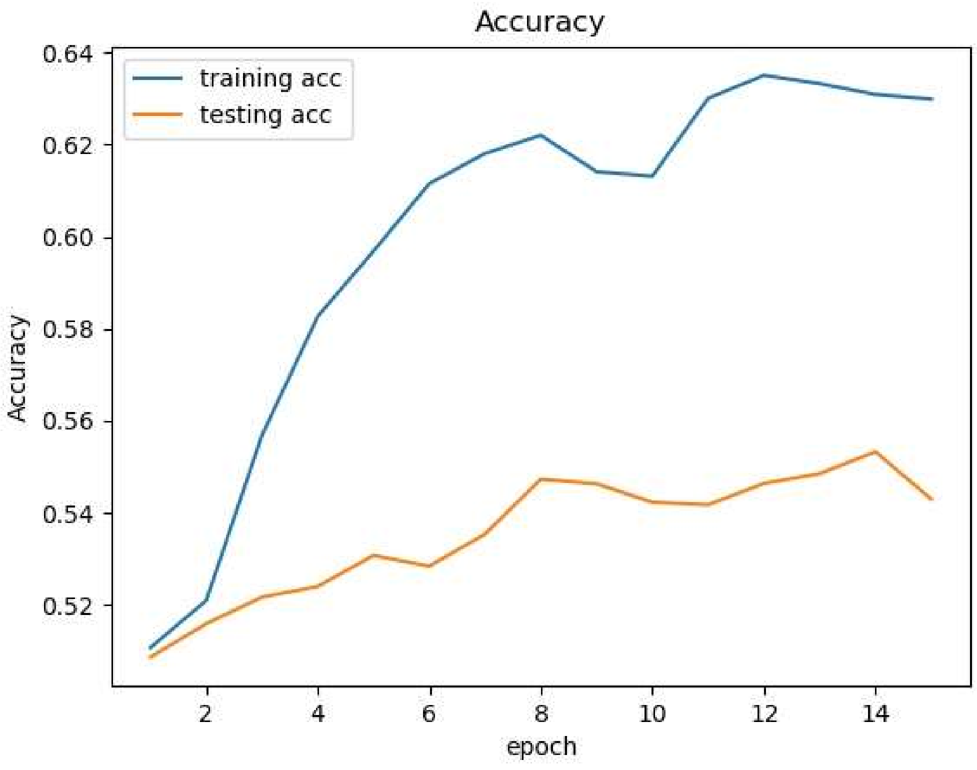
Network structure of the simple RNN:

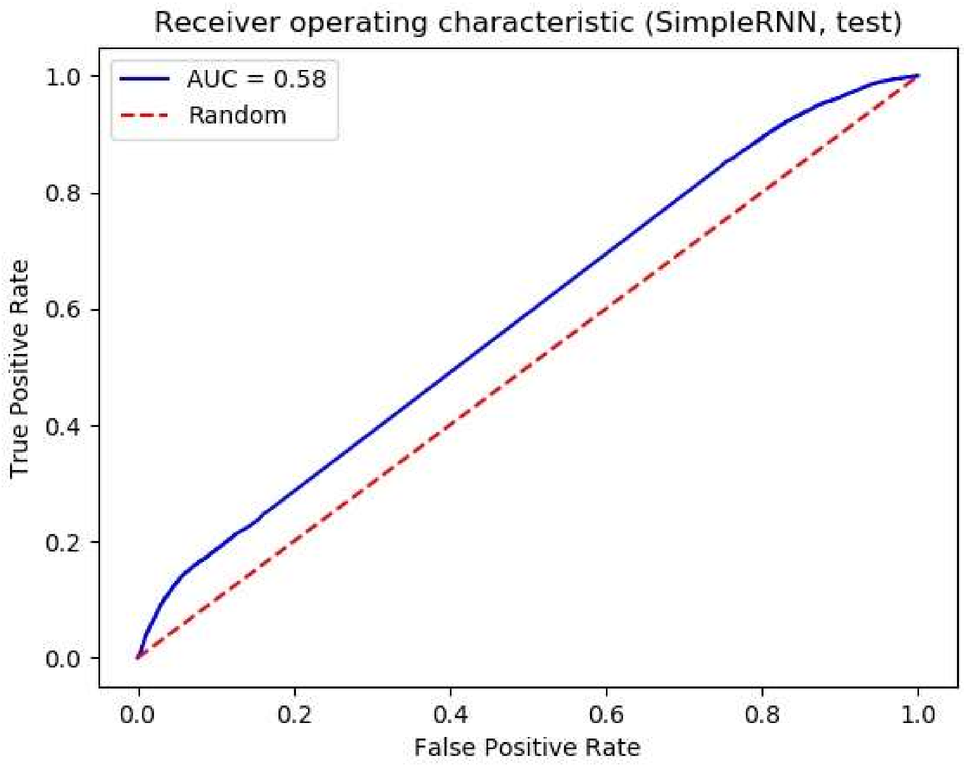
- Embedding layer:
  - Embedded dimension: from 10000 to 256
- RNN:
  - Layers: 1
  - Hidden units: 128
  - Activation functions: tanh
  - Drop-out rate: 0.5
- FCNN:
  - Layers: 1

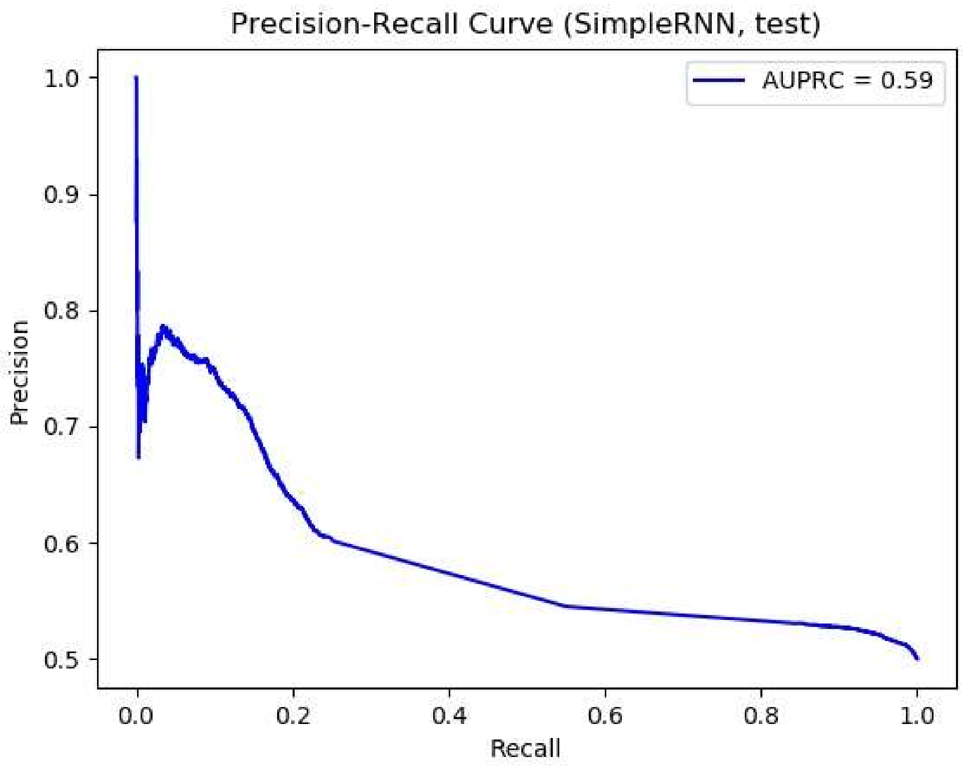
Training hyper parameters:

- Epochs: 15
- Batch size: 500
- Optimizer: Adam
- Learning rate: 0.008




Accuracy
training acc
testing acc
Accuracy
epoch
0.64
0.62
0.60
0.58
0.56
0.54
0.52
2
4
6
8
10
12
14


Receiver operating characteristic (SimpleRNN, test)
AUC = 0.58
Random
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

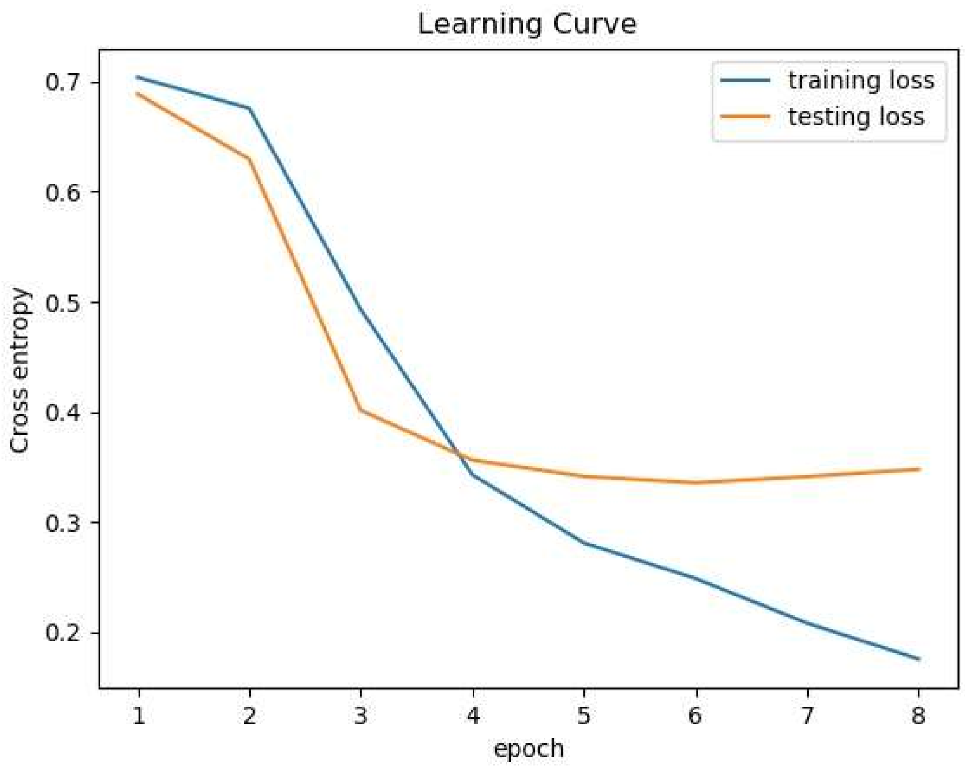


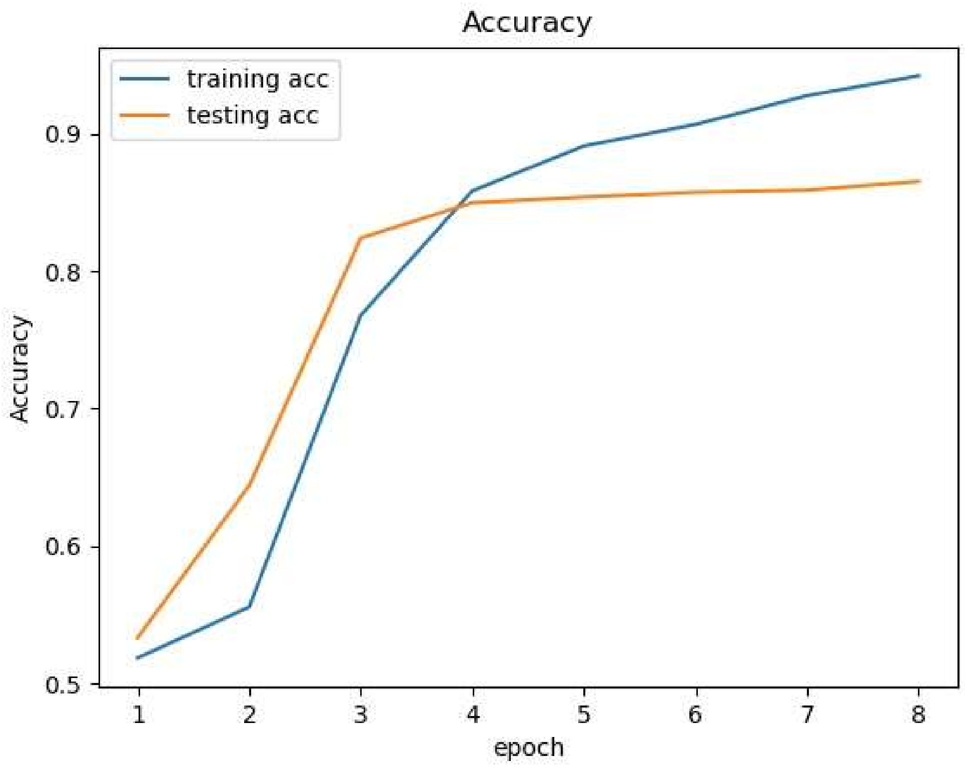
Network structure of the GRU:

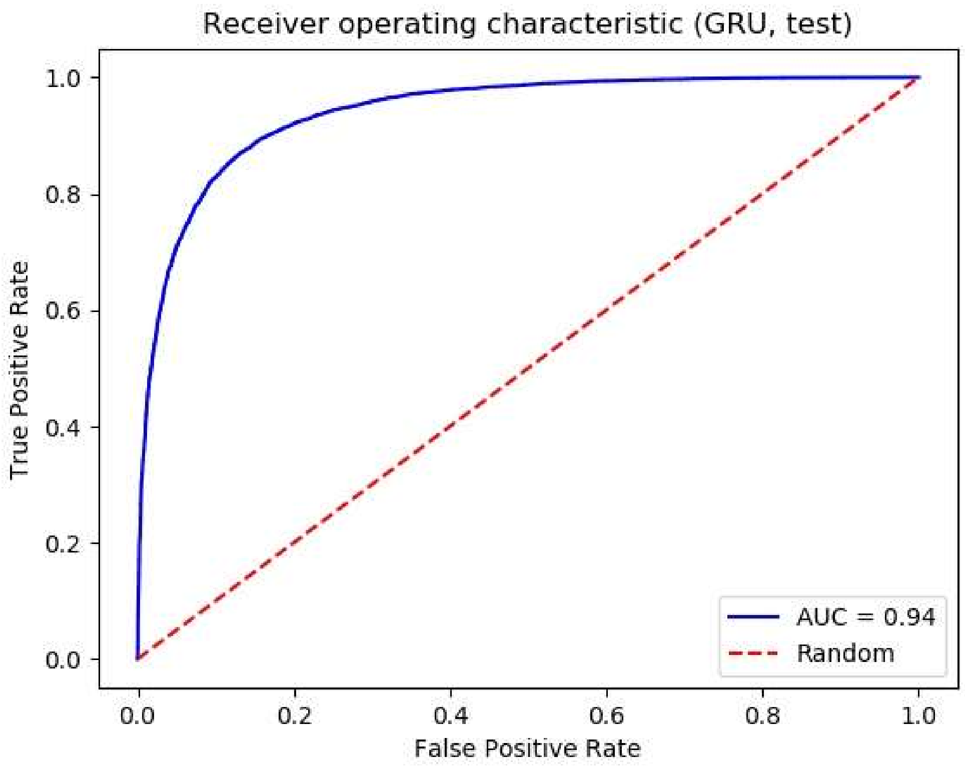
- Embedding layer:
  - Embedded dimension: from 10000 to 256
- RNN:
  - Layers: 1
  - Hidden units: 128
  - Activation functions: tanh
  - Drop-out rate: 0.5
- FCNN:
  - Layers: 1

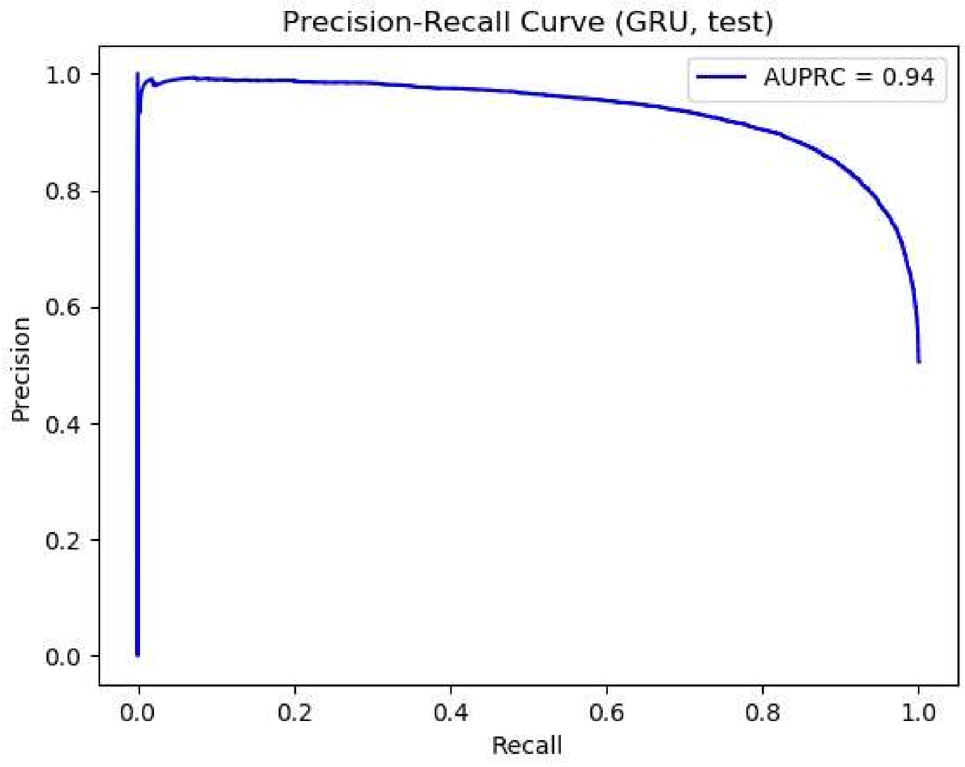
Training hyper parameters:

- Epochs: 8
- Batch size: 250
- Optimizer: Adam
- Learning rate: 0.0005


Learning Curve
training loss
testing loss
Cross entropy
0.7
0.6
0.5
0.4
0.3
0.2
1
2
3
4
5
6
7
8
epoch


Accuracy
training acc
testing acc
Accuracy
0.9
0.8
0.7
0.6
0.5
1
2
3
4
5
6
7
8
epoch





Network structure of the LSTM:

Embedding layer:

Embedded dimension: from 10000 to 256

RNN:

Layers: 1

Hidden units: 128

Activation functions: tanh

Drop-out rate: 0.5

FCNN:

Layers: 1

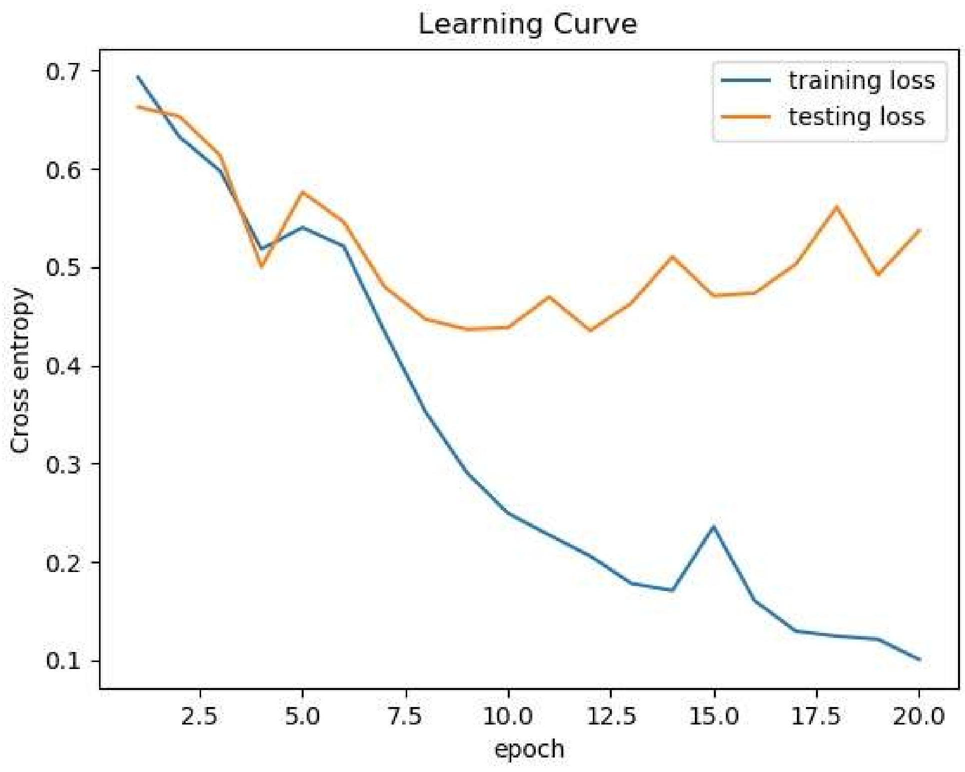
Training hyper parameters:

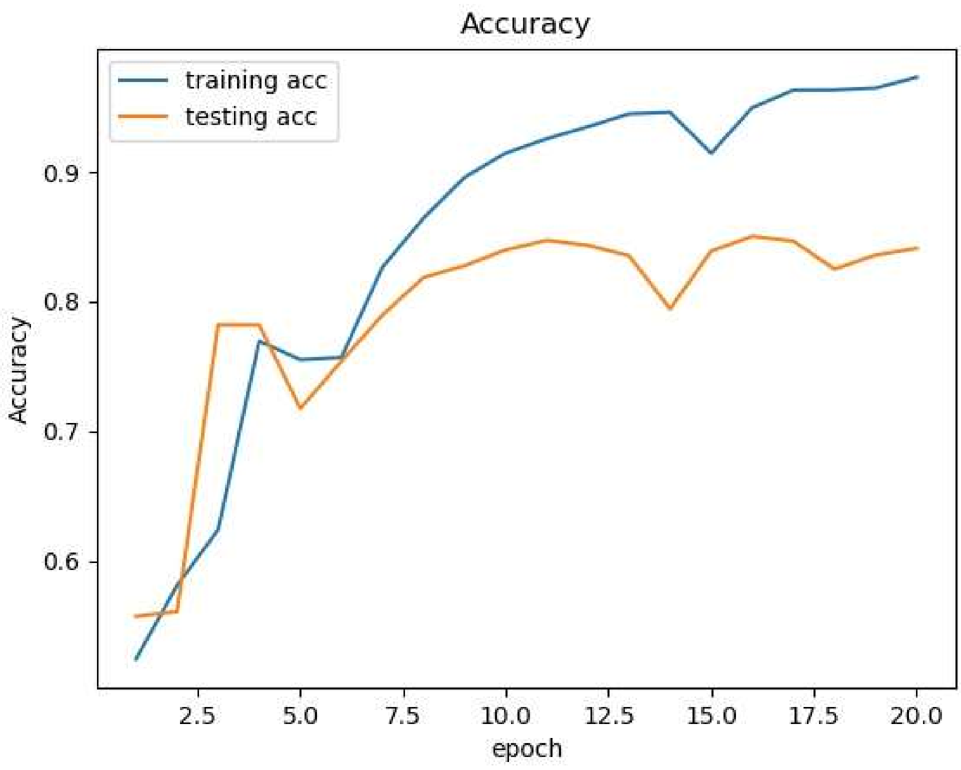
Epochs: 20

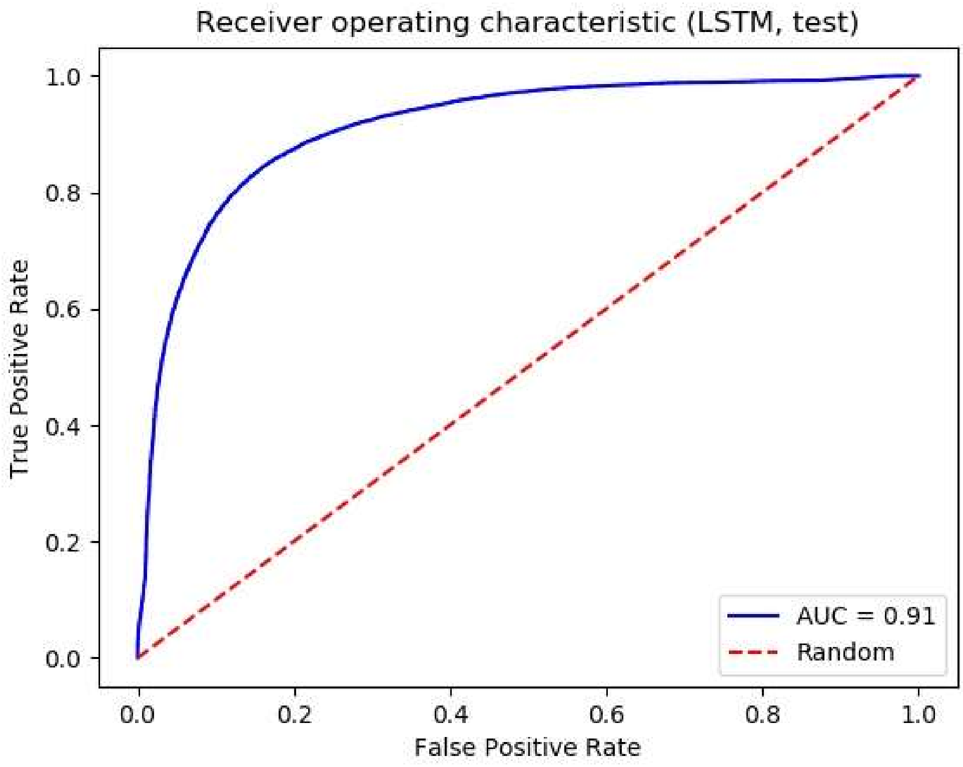
Batch size: 500

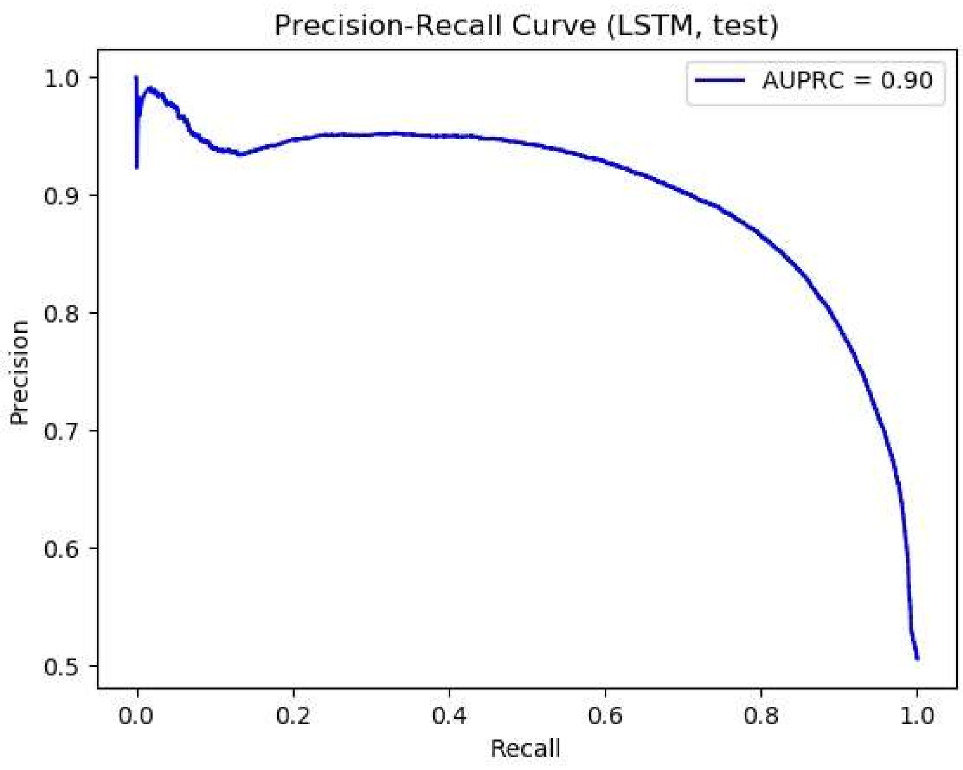
Optimizer: Adam

Learning rate: 0.002




Accuracy
training acc
testing acc
Accuracy
epoch
0.6
0.7
0.8
0.9
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Receiver operating characteristic (LSTM, test)
AUC = 0.91
Random
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0



Comments

With more information, we expect our models to have a better performance. However, this is not the case for the simple RNN, because it remembers all the information in the past. In our case it actually kills the performance. On the other hand, the GRU has a significant performance gain as we expected, as the ROC and PRC shows. For the LSTM, the performance has a slight drop, it is because I didn’t have time to fine tune it to the best. LSTM takes a lot more time to train, in theory it can perform better since it has more parameters to control. In practice, we need to consider the training time, since fine-tuning is also an important part of deep learning.

Problem 2

(a) **Train a sequence-to-sequence model to accomplish the translation task. Specify how does your model work (e.g. how do you preprocess the sentences in the dataset, how do you design your encoder/decoder, what technique do you use to enhance the performance, etc.) and summarize the model structure. Report your test accuracy (state your definition for sentence accuracy clearly) and plot the learning curve. Show some translation results as examples of your work in report (e.g. as shown in Figure.4).**

Data preprocessing:

For en.txt, I read them line by line and split them into list of words. Secondly, I pad the sentence with '<PAD>' to the length of the sentence with most words and I map every word to a unique integer.

For fr.txt, most procedures are the same except I add '<GO>' and '<EOS>' at the beginning and the end of the sentence respectively, then do the padding.

Finally, I split 20% of the whole corpus for testing data.

Embedding:

Embedded dimension: from 228 to 16 (en.txt), from 357 to 16 (fr.txt)

Encoder:

Layers: 1
Cell type: GRU
Hidden units: 256
Activation function: tanh

Decoder:

Layers: 1
Cell type: GRU
Hidden units: 256
Activation function: tanh

FCNN:

Layers: 1

Training hyper parameters:

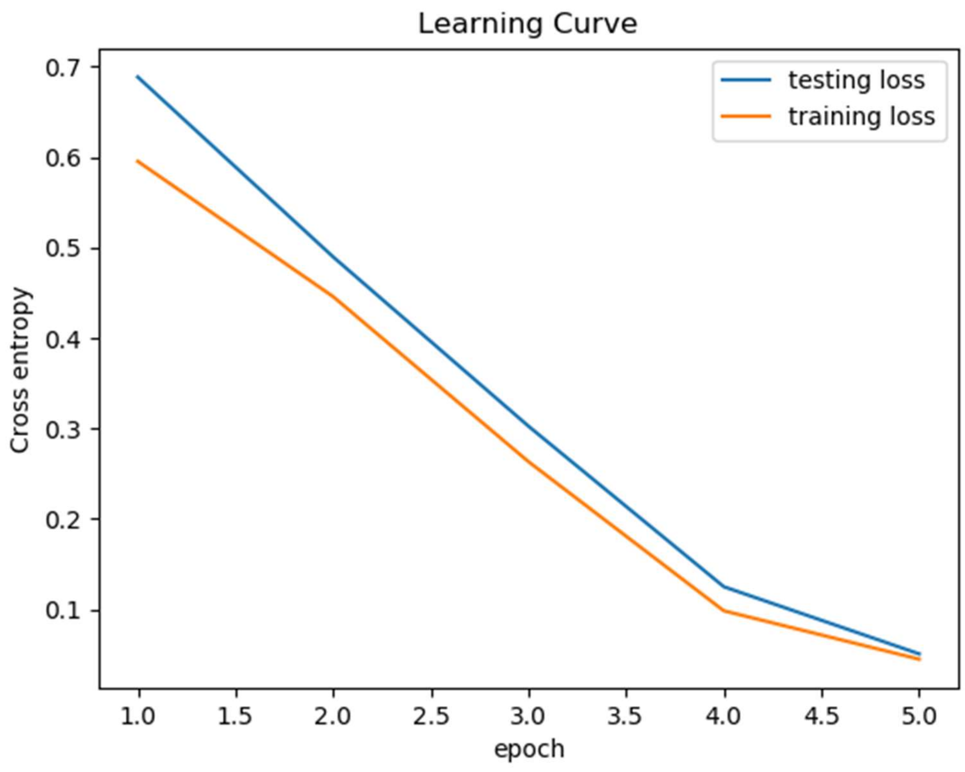
Epochs: 5
Batch size: 1120
Optimizer: Adam
Learning rate: 0.0005

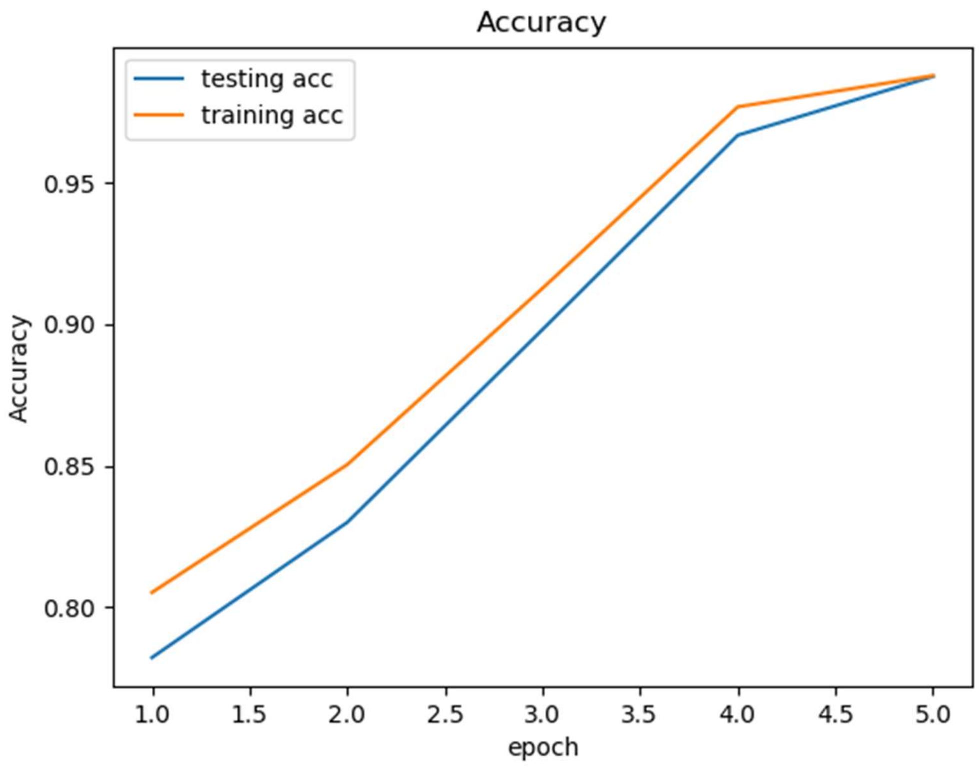
Extra techniques:

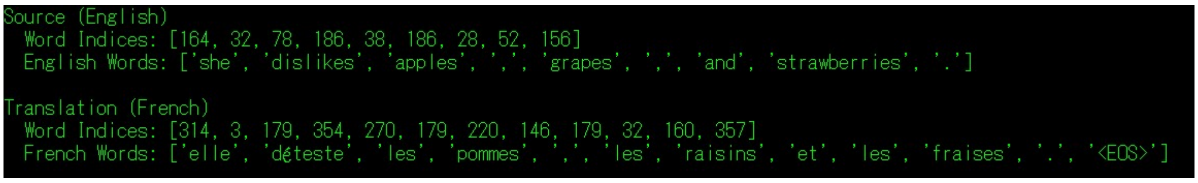
Gradient clipping: clip the gradient to -5 ~ +5

Accuracy definition:

Given two lists with same number of words, comparing the words pair-by-pair. (including <'PAD'> and <'EOS'> but not including <'GO'>).


Learning Curve
testing loss
training loss
Cross entropy
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
epoch


Accuracy
testing acc
training acc
Accuracy
0.95
0.90
0.85
0.80
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
epoch



```
Source (English)
  Word Indices: [164, 32, 78, 186, 38, 186, 28, 52, 156]
  English Words: ['she', 'dislikes', 'apples', ',', 'grapes', ',', 'and', 'strawberries', '.']

Translation (French)
  Word Indices: [314, 3, 179, 354, 270, 179, 220, 146, 179, 32, 160, 357]
  French Words: ['elle', 'déteste', 'les', 'pommes', ',', 'les', 'raisins', 'et', 'les', 'fraises', '.', '<EOS>']
```