Homework3: Sentence-level Sentiment Classification with RNN

Zhuoer Feng CST 86 2017011998

Different Cell Properties

BasicRNNCell

In this test, learning rate is set 0.0050, num_units is set 512.

GRUCell

In this test, learning rate is set 0.0050, num_units is set 512. Though I tested the 1000 one (which was illustrated in the slides that this size might be optimal), results showed a severe overfit problem thus it cannot be used as comparison in this section.

BasicLSTMCell

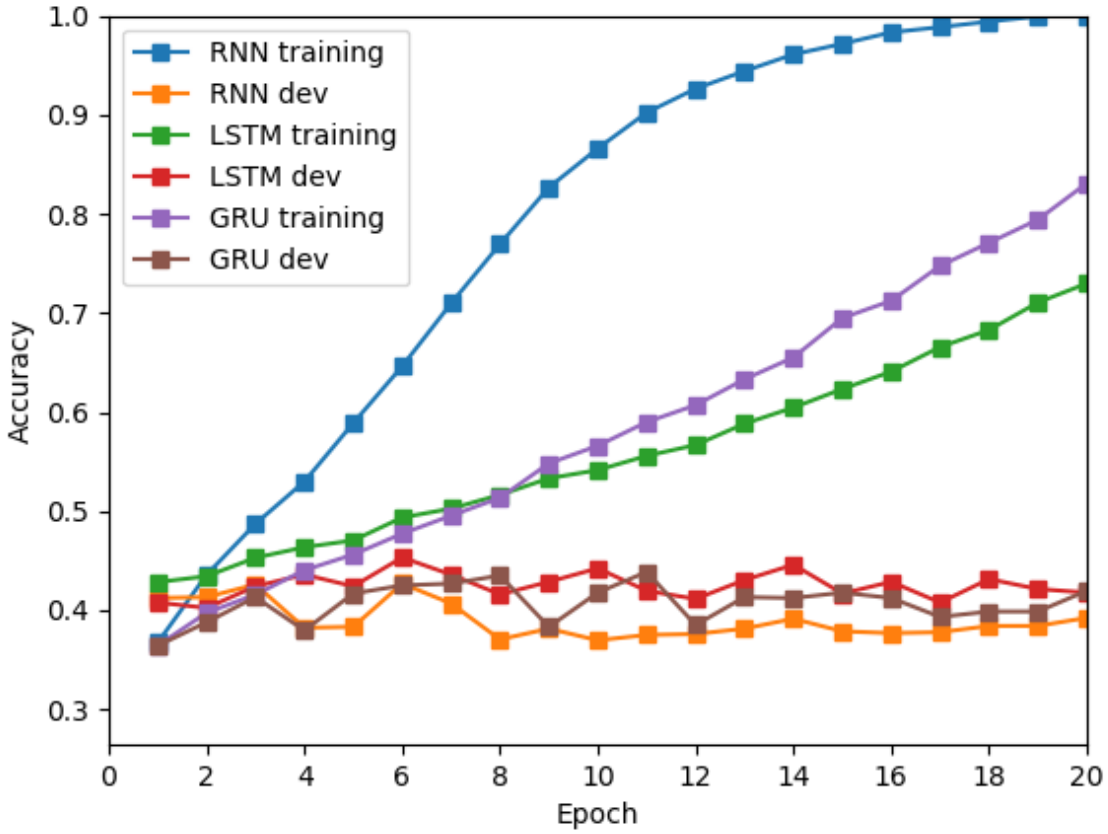
In this test, learning rate is set 0.0050, num_units is set 512. Though I tested the 100 one, the phenomenon of overfit prevale after epoch 10, and the final dev acc drop below 40%. Therefore, I choose units numbers to be 512.

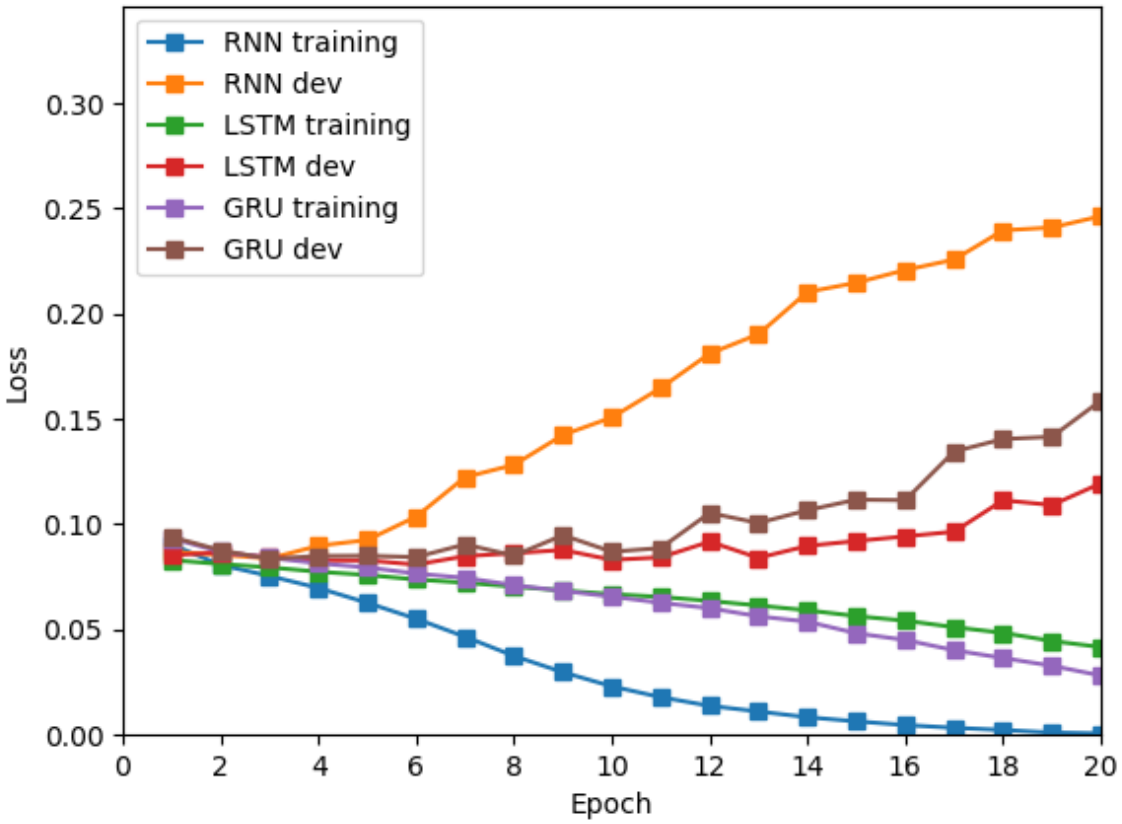
Results

Among all the cell models, the graph below shows that LSTM Cell got the best performance in this sentiment classification task, with low dev loss and high dev accuracy.

In loss graph, LSTM has lowest dev loss value while RNN gets the highest loss dev value. In the acc graph, LSTM gets the highest maximum dev acc while RNN gets the lowest. Therefore, the performance of each model on this task can be rated as: RNN < GRU < LSTM.

But as already learned, in this graph, the existence of overfit problem is evident. the problem is worst in the RNN modle, maybe as a result of limited perceptive ability. LSTM showed the greatest capability in depress overfit and raise the acc.





The Results are listed as the table below.

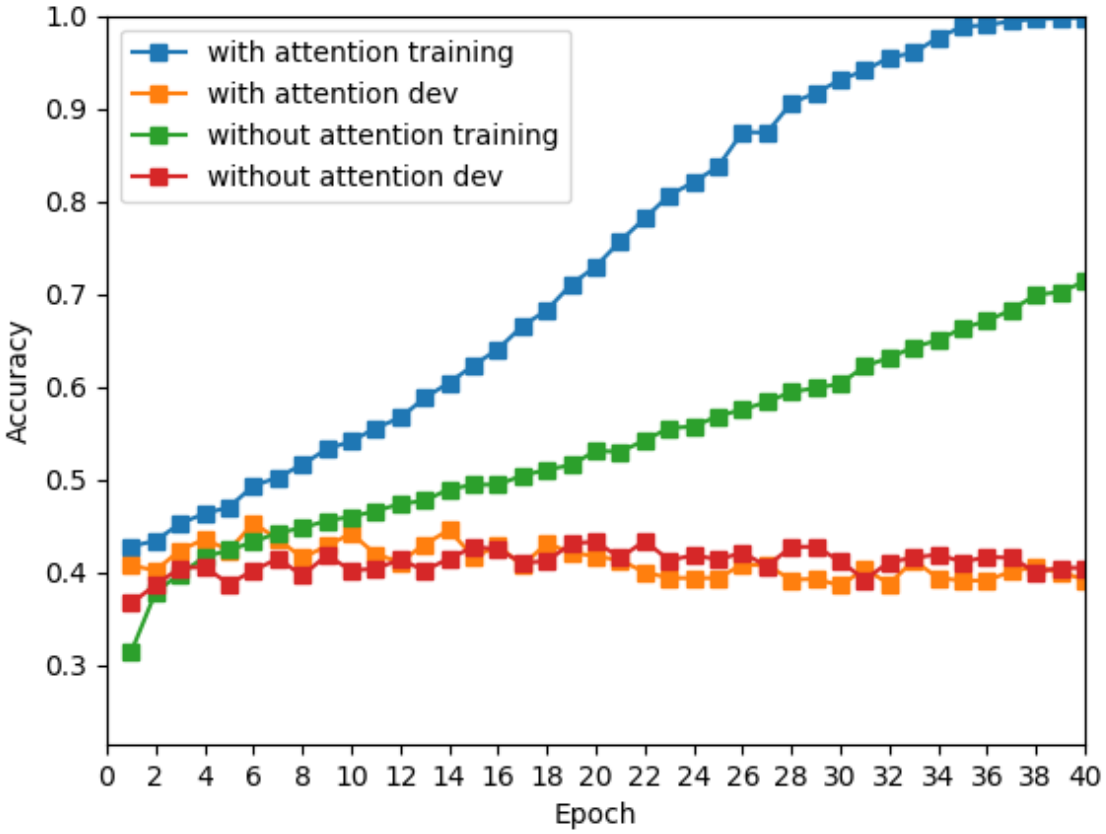
Cell Type	maximum training acc	maximum training acc' corelating loss	maximum dev acc	maximum dev acc's corelating loss
Basic RNN Cell	0.9998	19.9076	0.4269	0.1036
Basic LSTM Cell	0.6783	31.6365	0.4432	0.1116
Basic GRU Cell	0.8309	62.1985	0.4387	0.0887

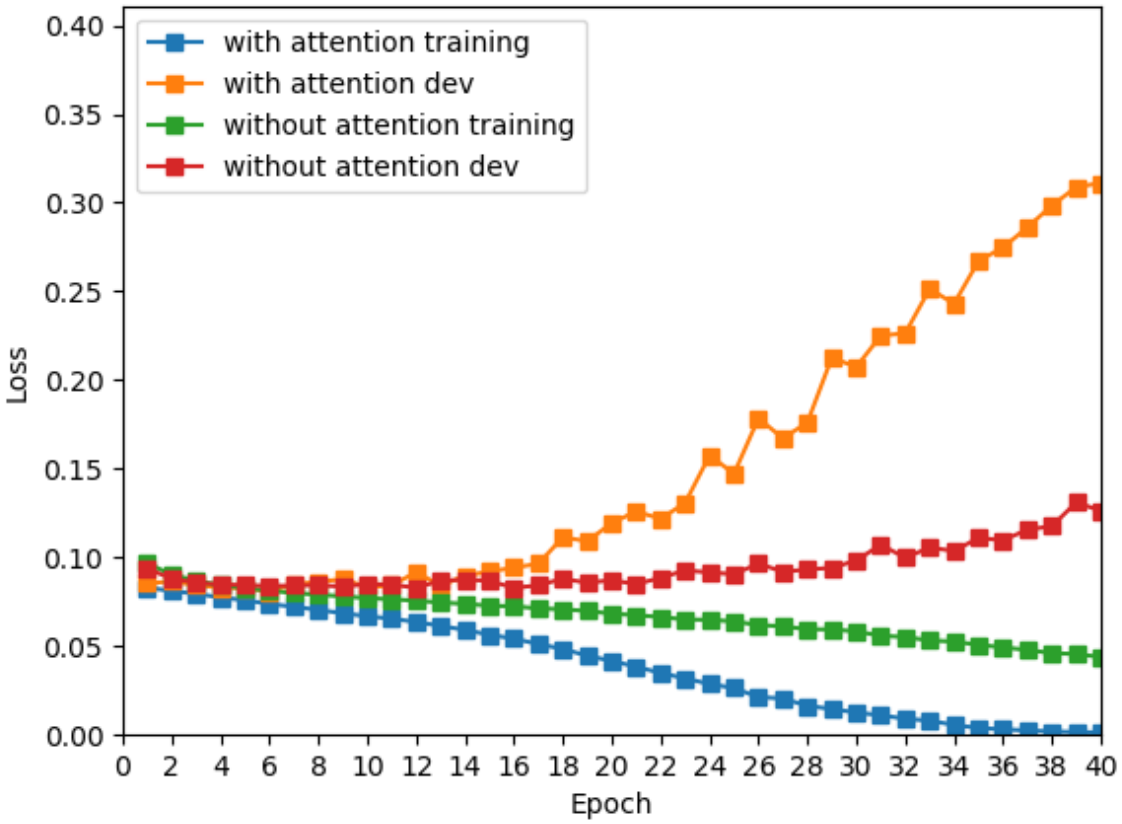
Self Attention

Using the LSTM Cell and deriving from the former LSTM model, I did the experiments below.

Cell Type	maximum training acc	maximum training loss	maximum dev acc	maximum dev loss
With	0.6783	31.6365	0.4432	0.1116
Without	0.5295	46.1329	0.4314	0.0880

The results showed that after removing the self-attention mechanism and replace it with an 'average' attention, the model's performace get decrease a little bit. To be specific, the learning efficiency decreased as the maximum training accuracy is decreased and its correlating loss is increased. Moreover, the maximum accuracy of the model without attention also decreased while there is a little optimization in loss. But overall, the model performs better with the attention machanism, though the existence of overfit makes the one with attention seem to own extremely high loss. But this is an inevitable result since I didn't find a better way to overcome it. But it in turns illustrated the fact that with the attention, LSTM can 'learn' at a faster speed.


with attention training
with attention dev
without attention training
without attention dev
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40


with attention training
with attention dev
without attention training
without attention dev
Loss
Epoch
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40

Final Model

The final model I chose is simply the Basic LSTM Cell adding attention. The performance can be seen in the following graph.

Parameters

Globle Settings

vocabulary size = 18430(which does not matter in this specific structure),

labels nums = 5(total five classifier),

training epoch = 40,

batch size = 16

Model parameters

learning_rate=0.005, max_gradient_norm=5.0, param_da=150, param_r=10,

cell units = 512(optimal to the theoretical size in the lecture slides)

embedded units = 300,

layers = 1

NOTE: as for the cell units, I did the following experiments

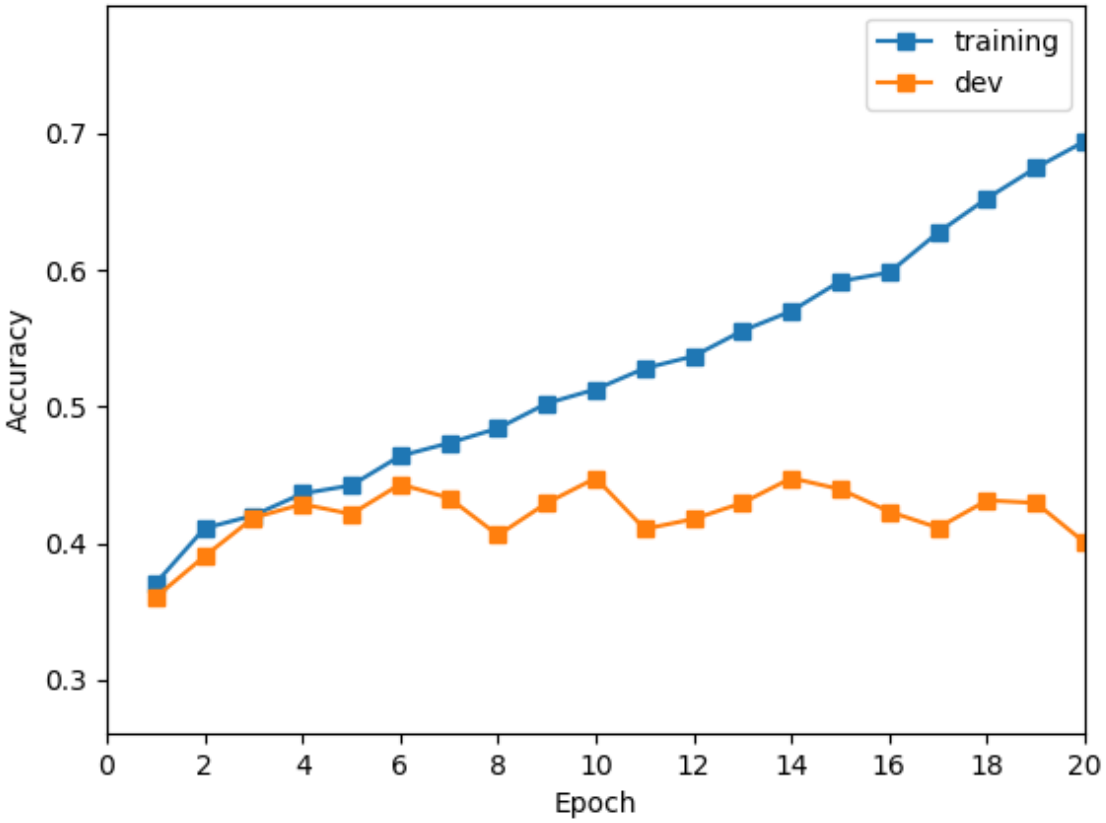
numbers of cell units	**256**	**512**	**800**
dev acc	0.4496	0.4541	0.4514
dev loss	0.0818	0.0845	0.0874
training acc	0.6642	0.6943	0.6953
training loss	0.0510	0.0473	0.0468

results showed that the 512 units is optimal for LSTM in this task, with both high dev acc and relatively low dev loss. The reason might be that under this size, the cell will neither get too small to cause sparcity in acc while not too high to cause overfit problem.

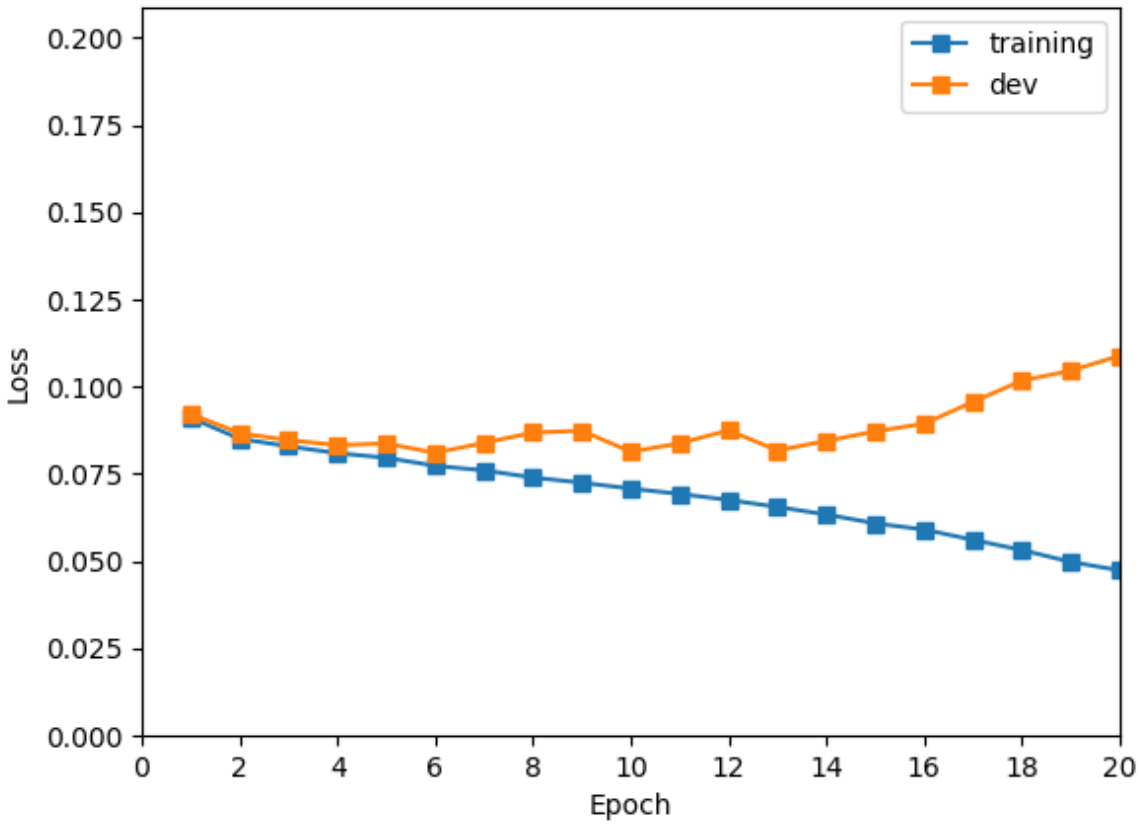
Other constructions

There are three gates in this structer: input gate, output gate and forget gate, which are all implemented in the codes. Moreover, the attention machanism is used in this model.

Accuracy


training
dev
Accuracy
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
10
12
14
16
18
20
Epoch

Loss



Analysis

Due to the existence of overit, I finally chose the former 20 epoch of the 40 epoch result as my final model properties.

	accuracy	**loss**
final training -	0.69428839	0.04739094
max training -	0.69428839	0.04739094
final validation -	0.40054496	0.10888828
max validation -	0.45413261	0.08450293

The result of which is put in the file called `./result.txt`