SI630 Homework 1

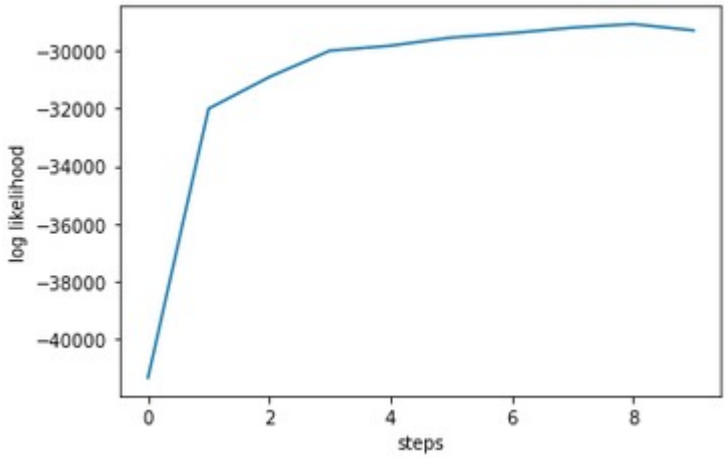
Part1

For `better_tokenize`, I first get rid of all things that are not alphabets by using regex expression. Then I use a stopword list from online that shows common stop words(I didn't use the stopword list from NLTK since I believe some words in that list could also contain information.) Then I make all words as their lower case.

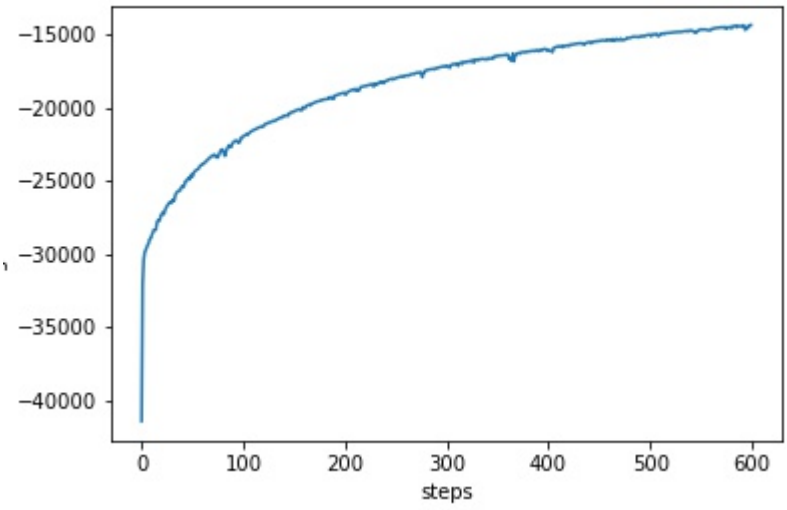
Part2

Train your model on the training data learning rate=5e-5 (i.e., a very small number) and just use num steps = 1000. Then make a plot for the full data every 100 steps.

For 1000 emails,

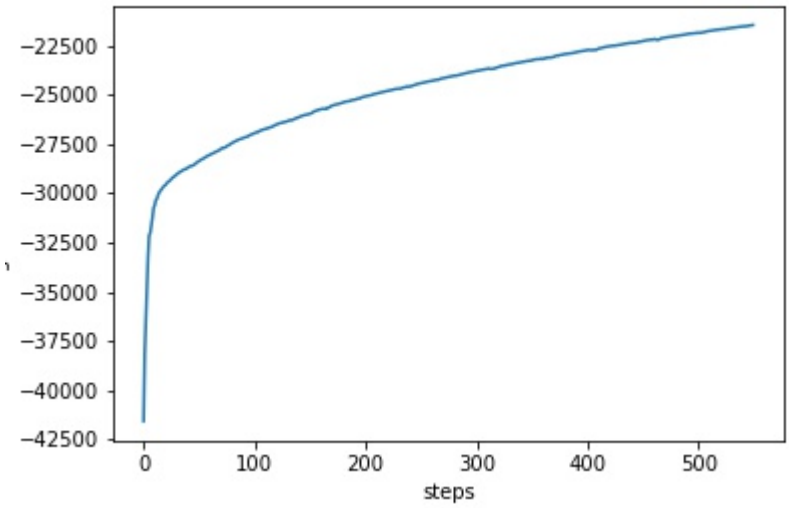


For the whole data,

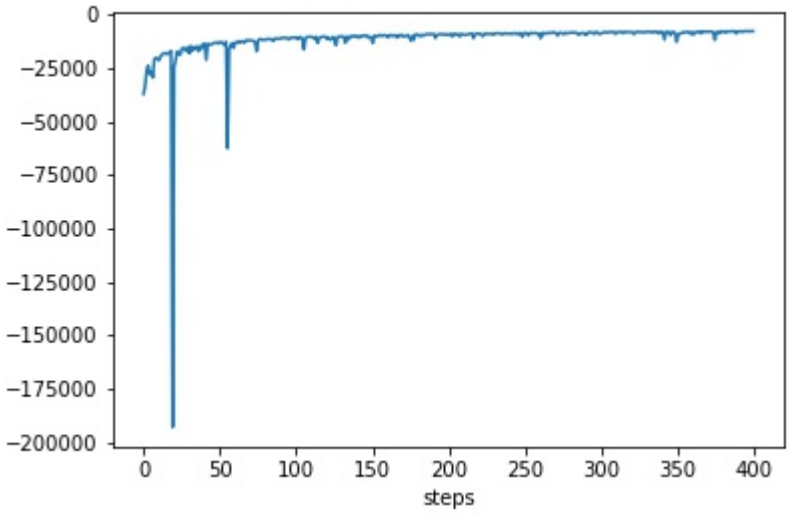


Train the model until it converges.

I set learning rate to 1e-5 and train for 55 epochs. It converges around 53 epochs.



Then I use learning_rate=9.5e-4 and train 40 epochs. It could be seen that it converges eariler(at around 30 epochs).



Use logistic regression classifier to make predictions on the validation dataset and report your performance using the F1 score

For learning_rate=1e-5 and training 55 epochs, the F1 score is 0.9042.

```
F1 value is: 0.9042171567381296
accuracy is: 0.83045
```

For learning_rate=9.5e-4 and train 40 epochs, the F1 score is 0.9683.

```
F1 value is: 0.9683424162635479
accuracy is: 0.9483
```

Part3

1. Train your model for 1000 steps.Showing it 1000 randomly sampled documents) and report the loss after each 100 steps. This should verify that the loss is going down.

I choose learning rate to be 9.7e-4, since this value is relatively appropriate as training speed. Later parts for SGD I also use this value.

This is a screenshot of the sum of loss every 100 steps. As it could be seen, the loss is going down despite of some fluctuations.

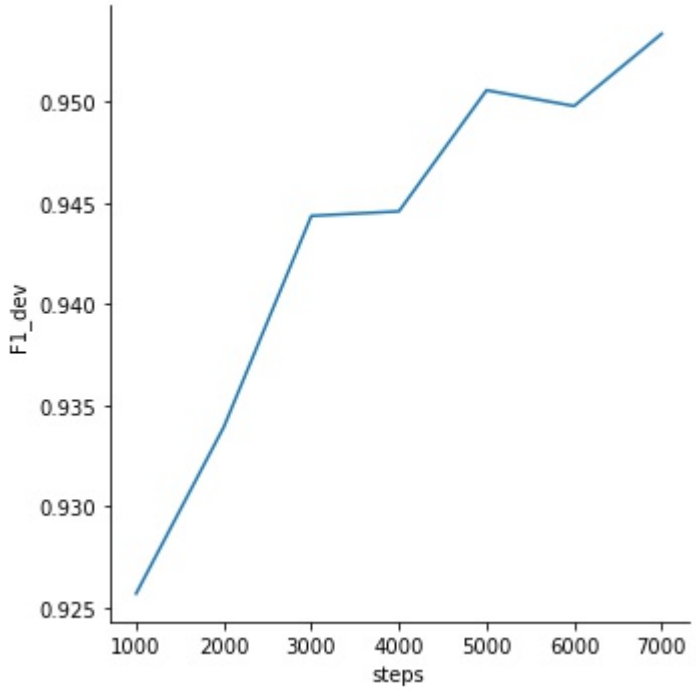
```
[100] loss: 50.3372219
[200] loss: 60.5590529
[300] loss: 39.0244211
[400] loss: 40.4771532
[500] loss: 45.5413563
[600] loss: 34.1800866
[700] loss: 42.8295864
[800] loss: 30.7136375
[900] loss: 31.1374027
[1000] loss: 33.5400345
finish
```

2. Once you're satisfied that the model is working, train your model for at least 5 epochs and compute (and save) both (1) the loss every 1000 steps and (2) the F1 score of the model on the development data.

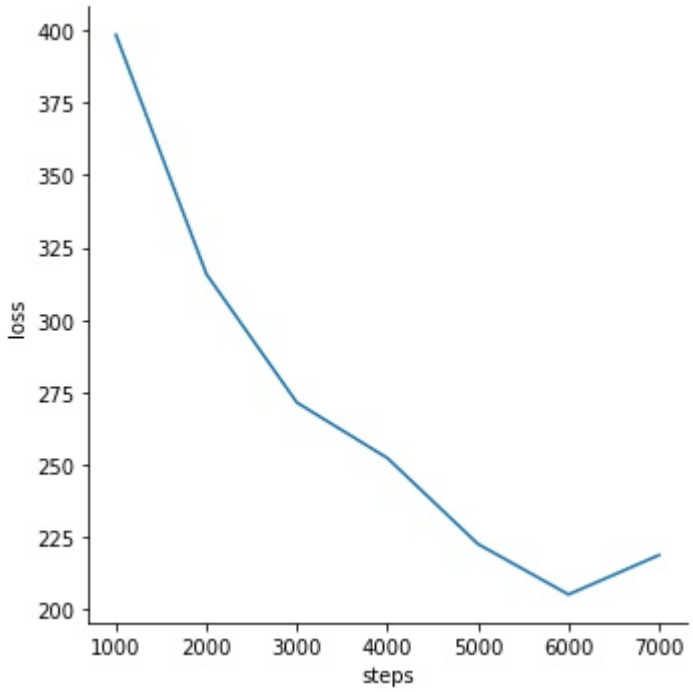
This is a screenshot of the loss and F1 value as well as training accuracy on development set every 1000 steps. I ran 7 epochs in total.

```
[1000] loss: 398.4125420
F1 value is: 0.9256844850065189
accuracy is: 0.87745
[2000] loss: 315.7804408
F1 value is: 0.9339093390933908
accuracy is: 0.88985
[3000] loss: 271.4409360
F1 value is: 0.9443504475959025
accuracy is: 0.90955
[4000] loss: 252.2728711
F1 value is: 0.9445752959065024
accuracy is: 0.90705
[5000] loss: 222.6211156
F1 value is: 0.9505609665885835
accuracy is: 0.9198
[6000] loss: 205.1914002
F1 value is: 0.9497814591802427
accuracy is: 0.91785
[7000] loss: 218.7829341
F1 value is: 0.9533513380800394
accuracy is: 0.924
finish
```

This is a plot for the F1 value every 100 epoches.



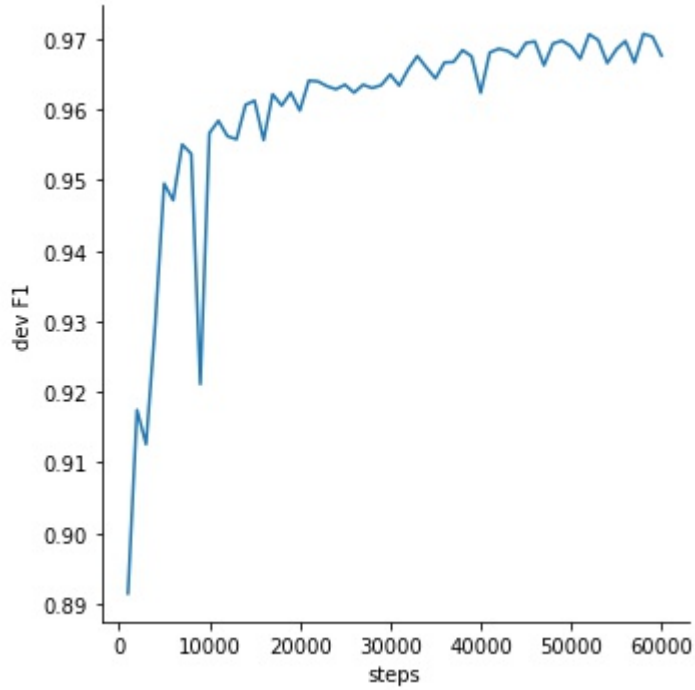
This is a plot for the accumulative loss every 100 epoches.

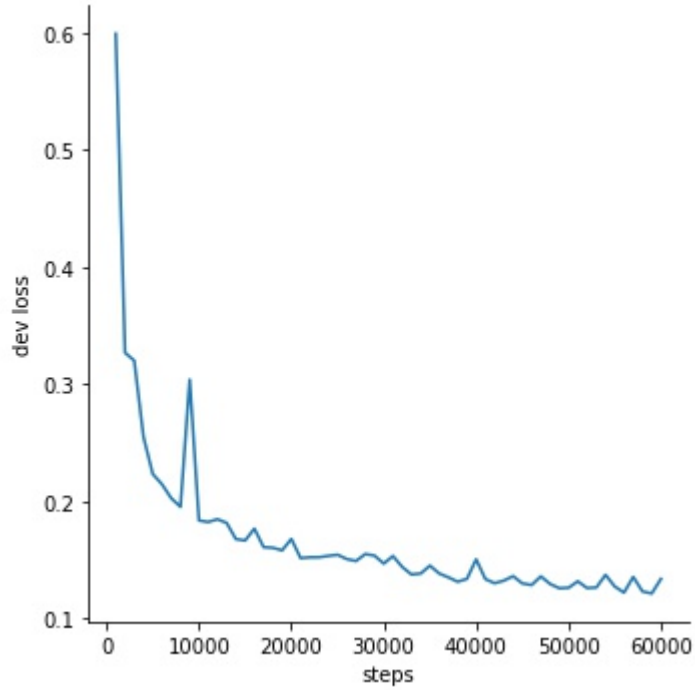


It could be seen that the F1 value is approaching 1, while the loss is going down.

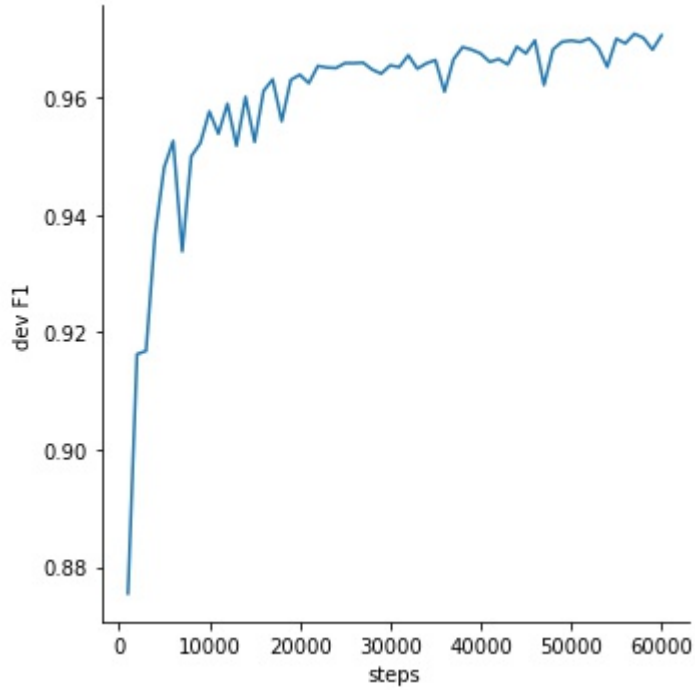
3. Adding Regularization

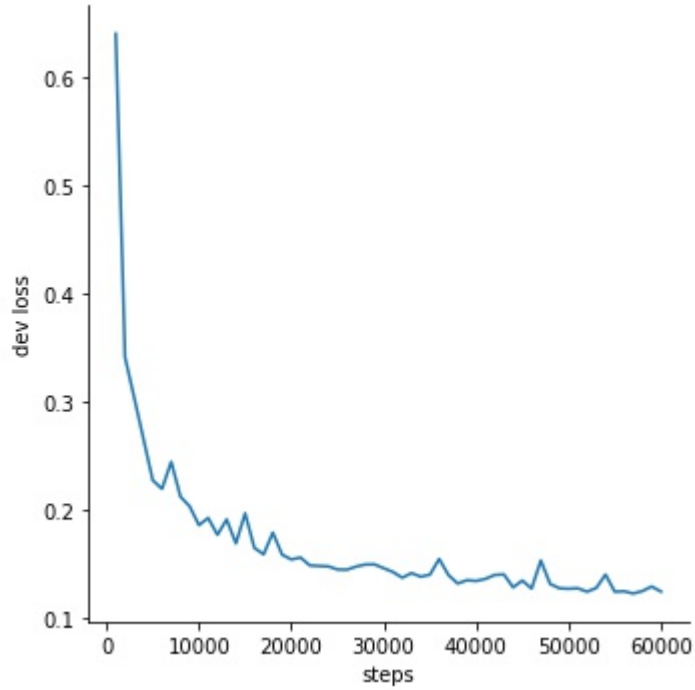
L2 penalty to 0



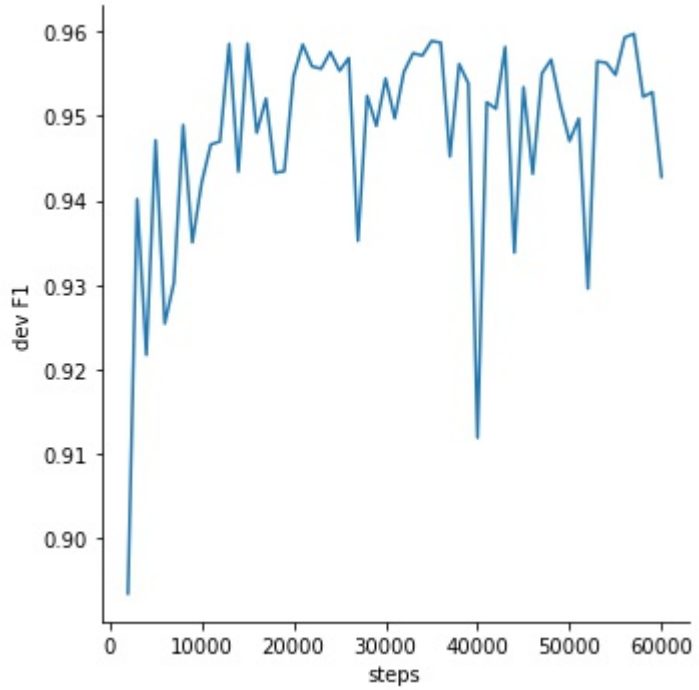


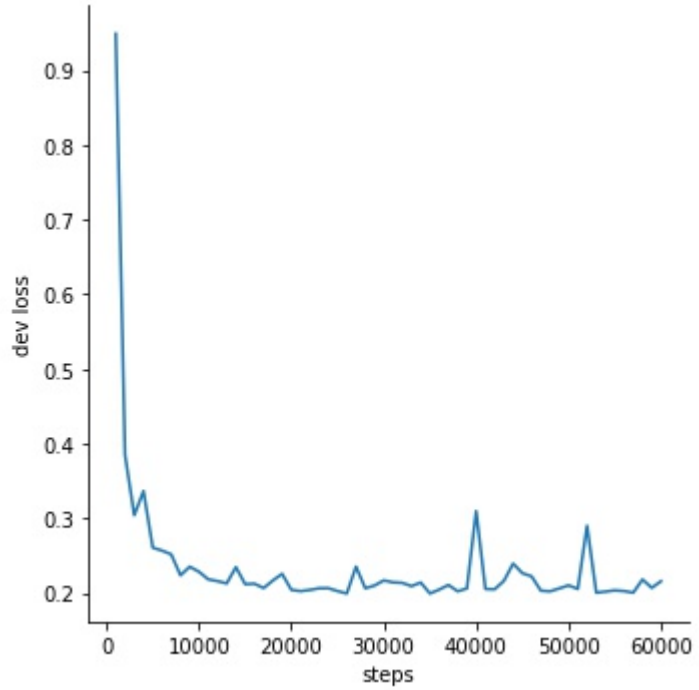
L2 penalty to 0.001





L2 penalty to 0.1



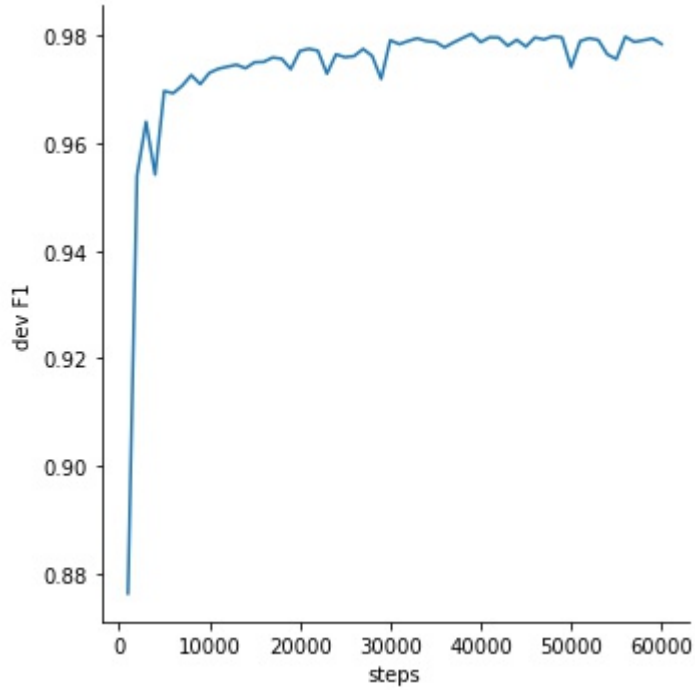


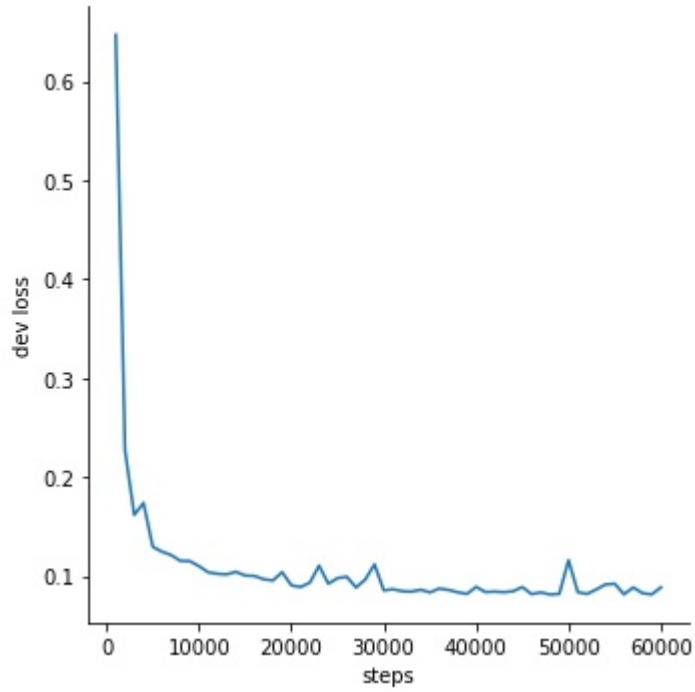
With the increase in L2 penalty, the stability of F1 decreases. F1 score is higher when L2 penalty is smaller. Loss is similar for weight decay=0 and 0.001, and is larger when weight decay=0.1

4. Different Optimization function

Adam:

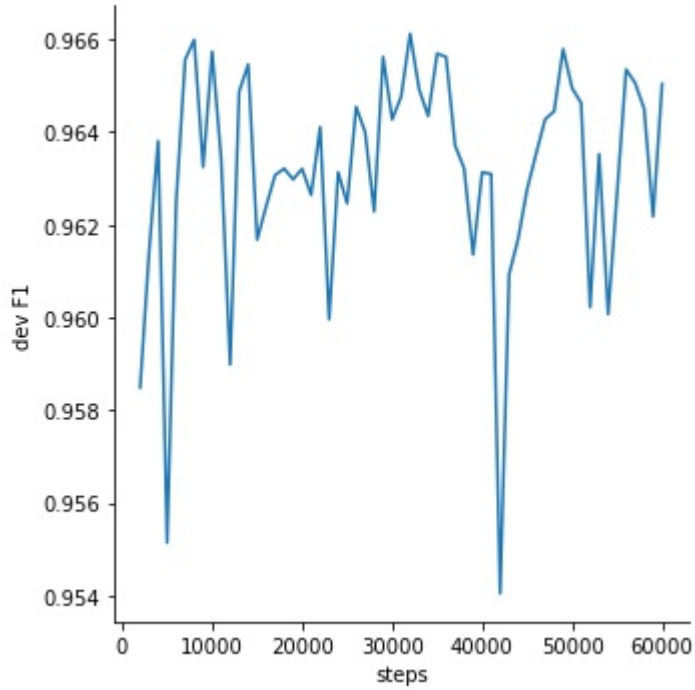
parameters:lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0.01, amsgrad=False

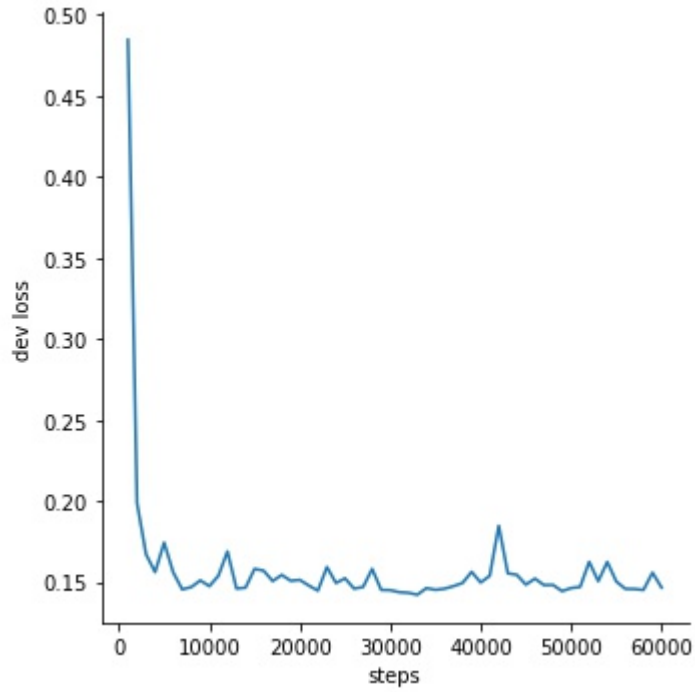




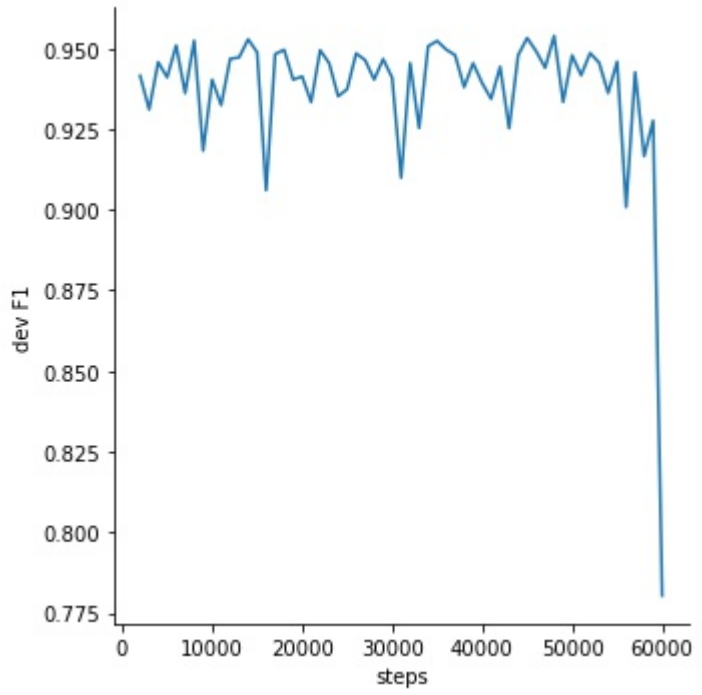
RMSprop

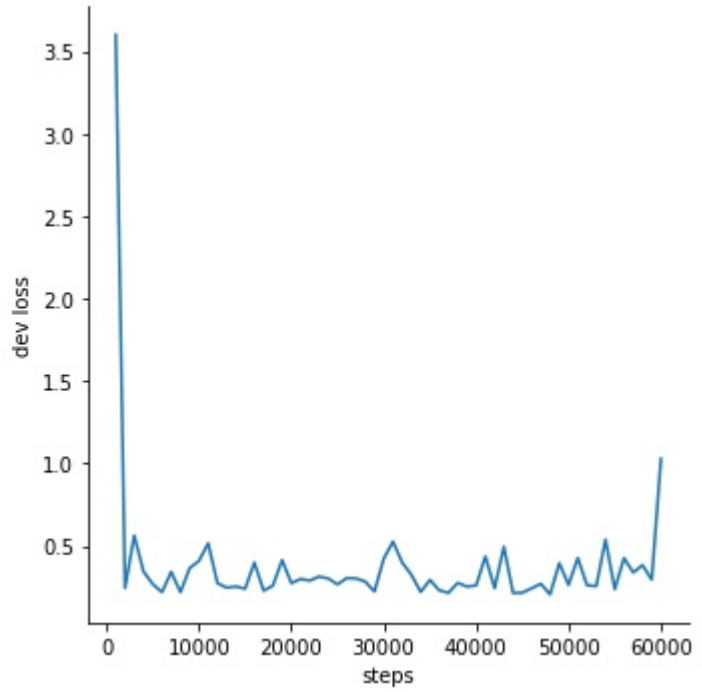
parameters:lr=0.001, alpha=0.99, eps=1e-08, weight_decay=0.01, momentum=0, centered=False





parameters: lr=0.01, alpha=0.99, eps=1e-08, weight_decay=0.01, momentum=0.00001, centered=False

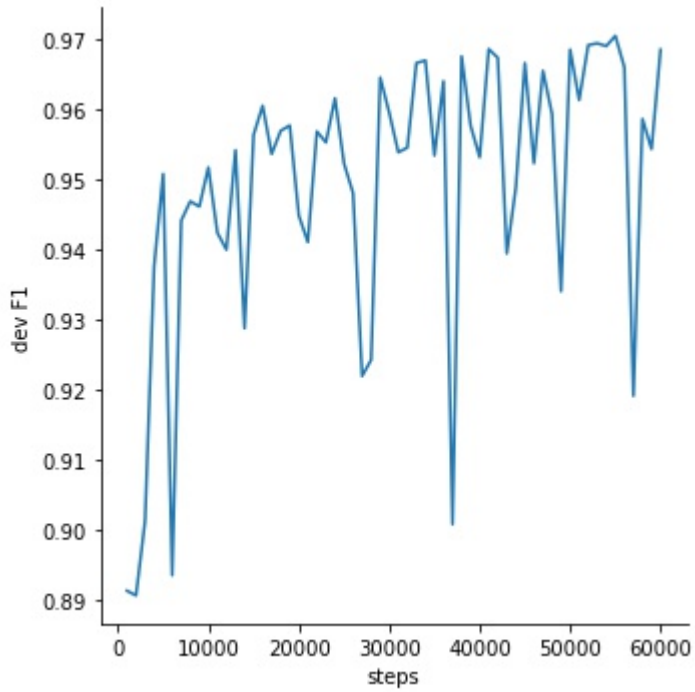


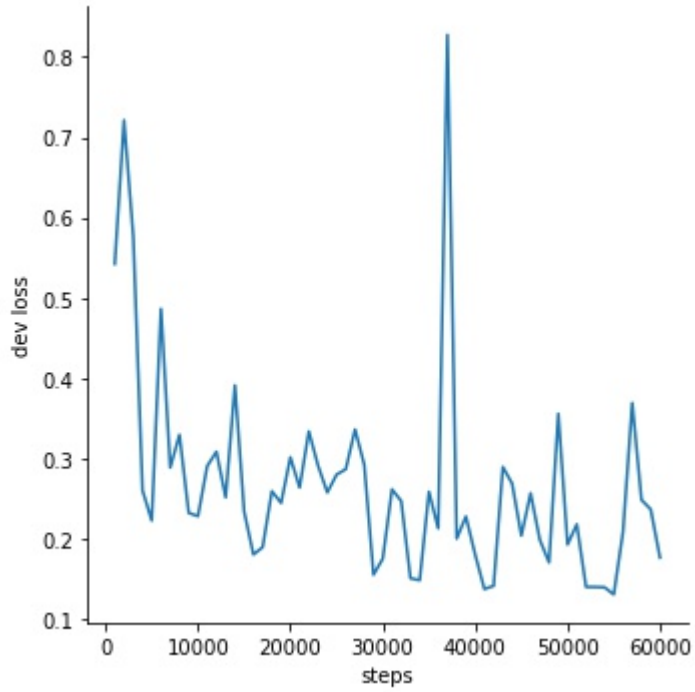


I use AdamW and RMSprop optimizer. For the RMSprop, I first set learning rate and weight-decay to be same as that of the Adam optimizer, and do not add momentum. Then I set its learing rate to default value 0.01, and add a little momentum.
For RMSprop, the F1 value is far less stable than AdamW, where the latter increase steadily. The loss for AdamW optimizer is also smaller. For two RMPprop optimizers, loss is smaller when a smaller learning rate with no momentum is applied. In general, AdamW performs better on this model.

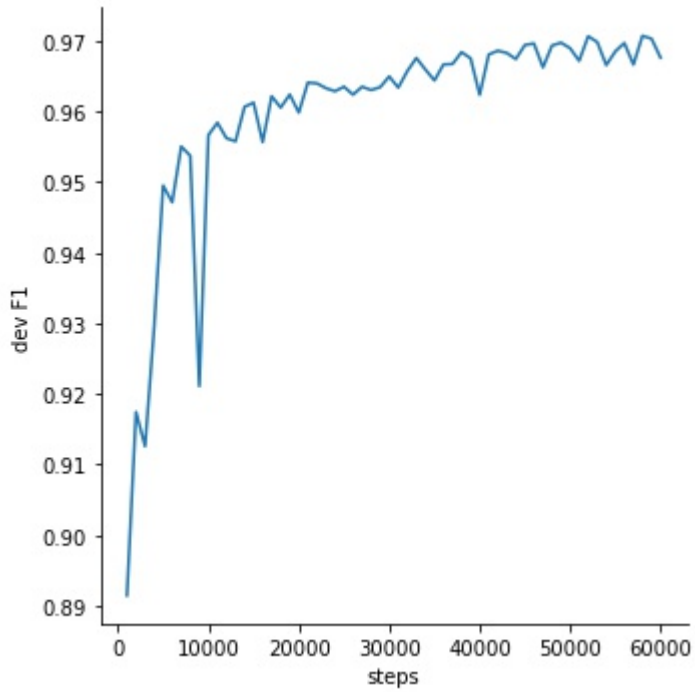
5. Two ways of tokenizing

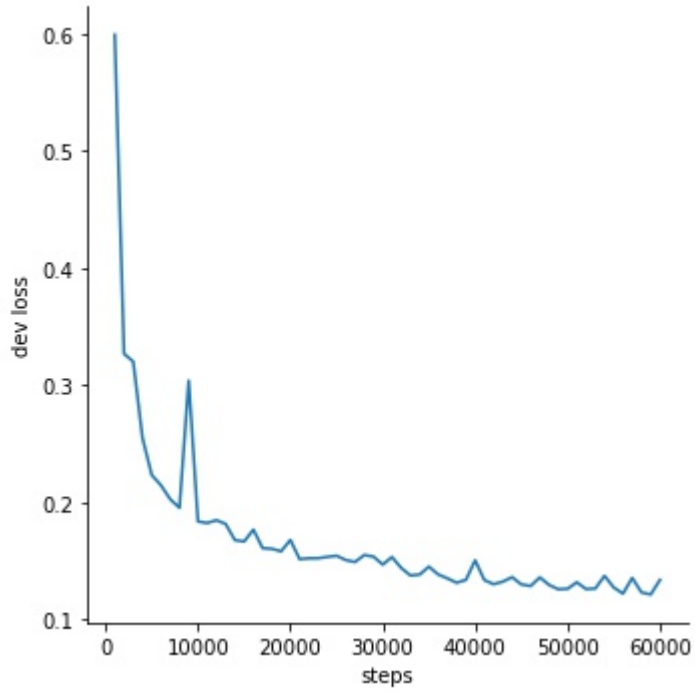
- For the basic optimizer, I also train with learing rate=9.7e-4.





- For the improved tokenizer



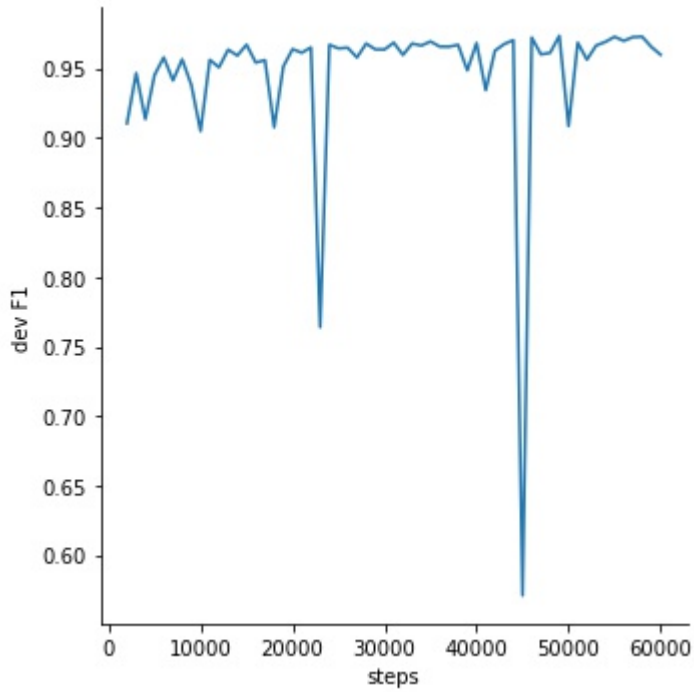


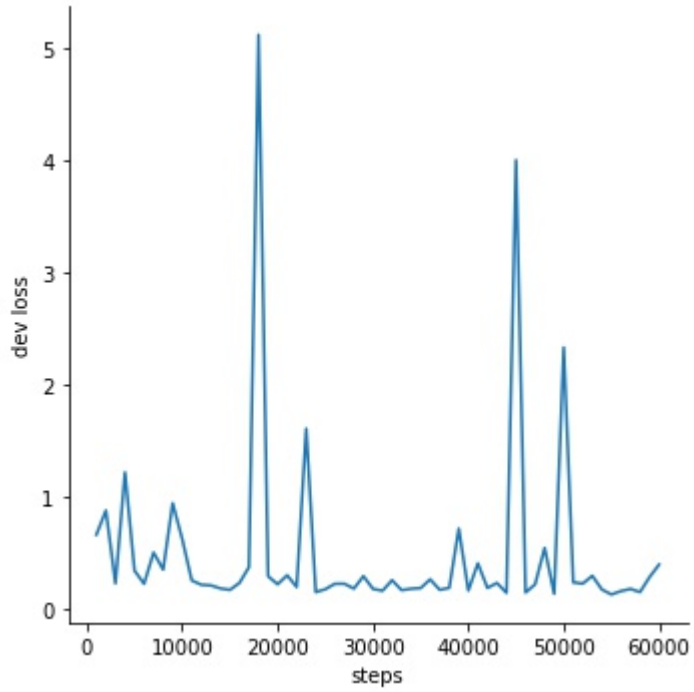
The improved tokenizer performs better and is more stable. The basic tokenizer brings more token than the improved ones(since punctuations and stop words are also included), therefore it do not converge easily.

6. Different learning rate

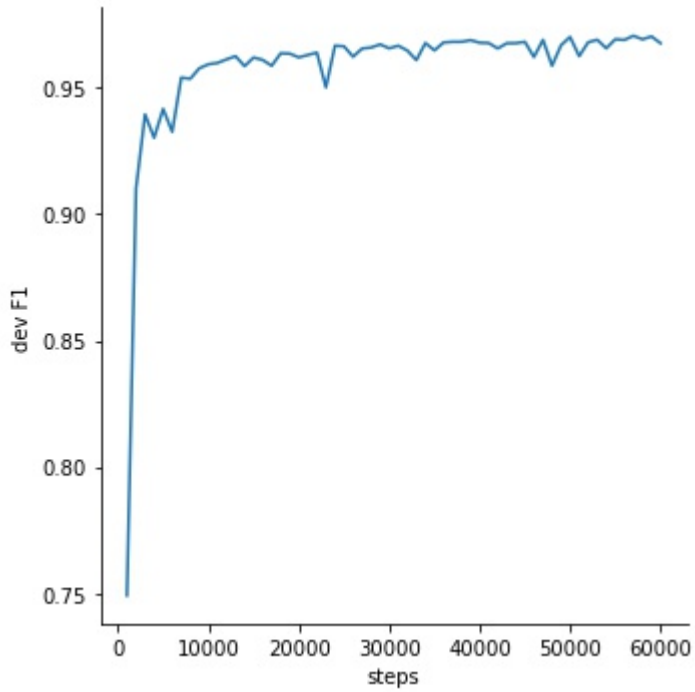
I choose 4 learning rate: 0,01,0.001,9.7e-4,1e-7

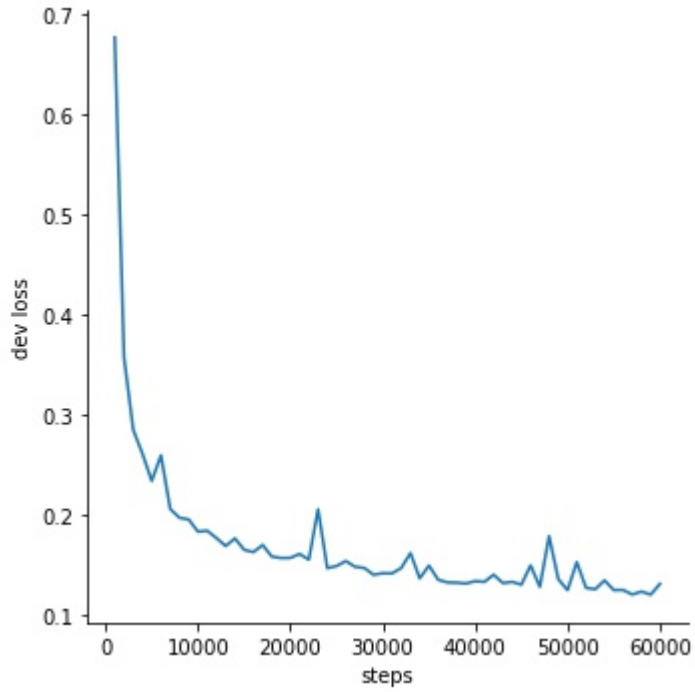
- 0.01:


dev F1
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
10000
20000
30000
40000
50000
60000
steps

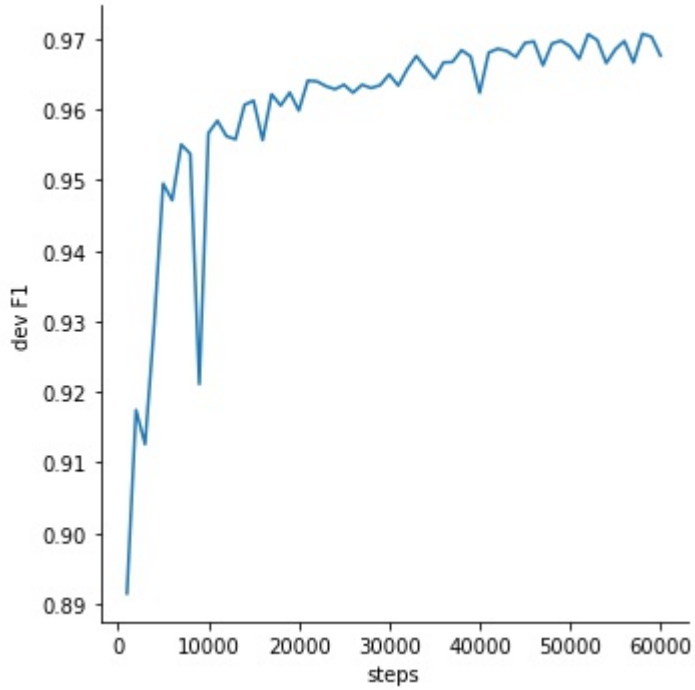

dev loss
5
4
3
2
1
0
0
10000
20000
30000
40000
50000
60000
steps

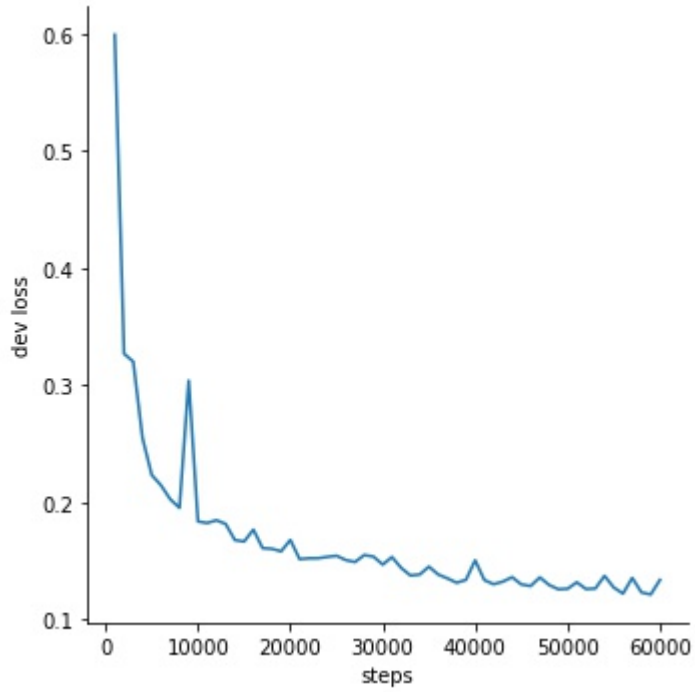
- 0.001:


dev F1
0.95
0.90
0.85
0.80
0.75
0
10000
20000
30000
40000
50000
60000
steps

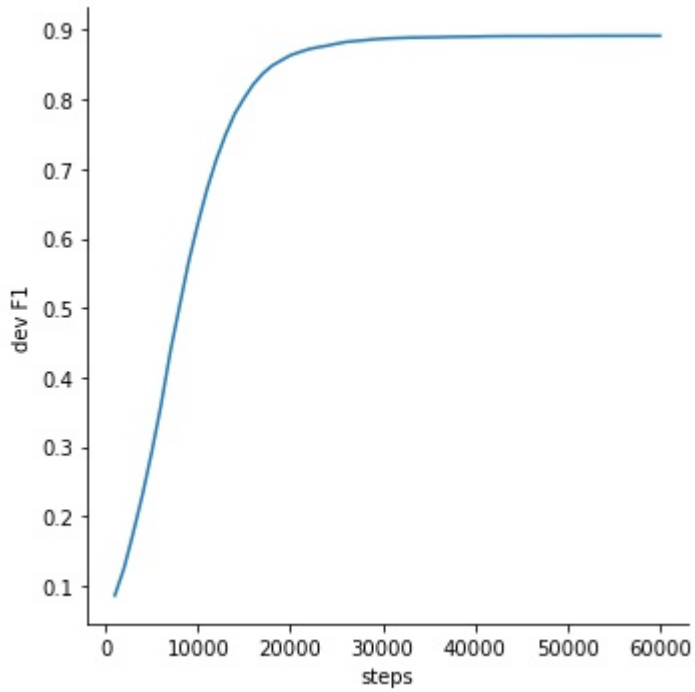

dev loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10000
20000
30000
40000
50000
60000
steps

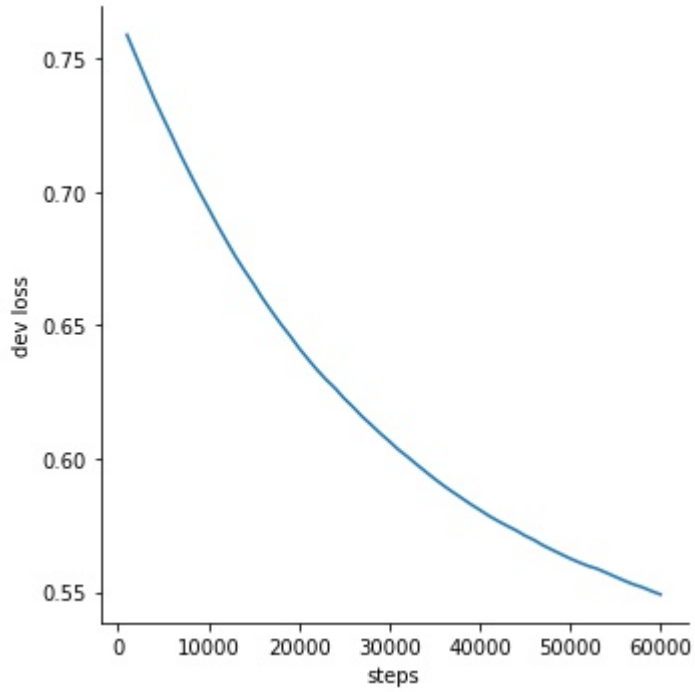
- 9.7e-4:


dev F1
steps
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
0
10000
20000
30000
40000
50000
60000


dev loss
steps
0.6
0.5
0.4
0.3
0.2
0.1
0
10000
20000
30000
40000
50000
60000

- 1e-7:





It could be seen that with the decrease of learning rate, the slope of F1 and loss becomes smoother and converge earlier. When the learning rate is quite large, it's quite hard for the model to converge.

Kaggle Username: OkabeRintarou