SI 630 – HW3

Lwenjing

Date: 3/9/2020

Documents submitted online:

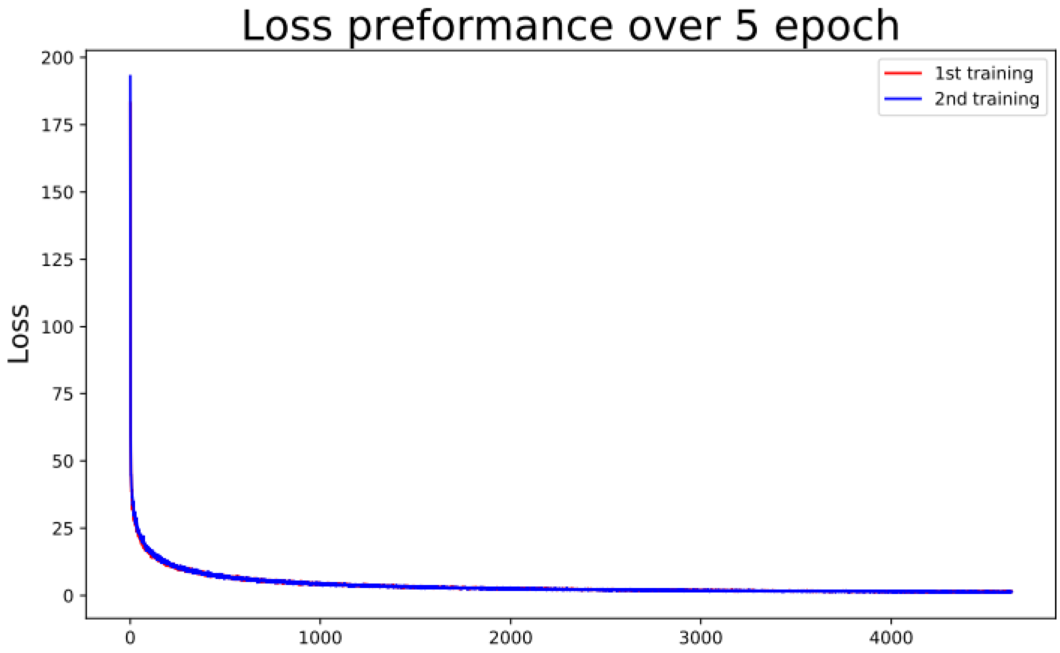
I changed the file directory since the original code doesn't work on windows system.

Filename: parser-epoch-%d.mdl – the SGD optimizer model

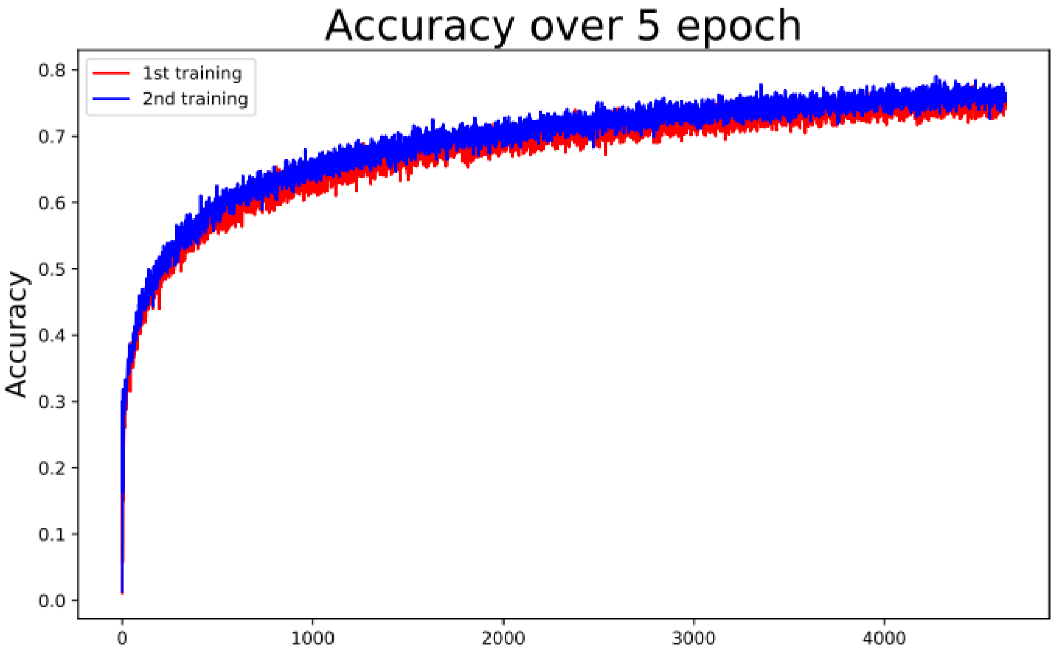
Parser-epoch-test-%d.mdl – the Adam optimizer model

First step: output the performance

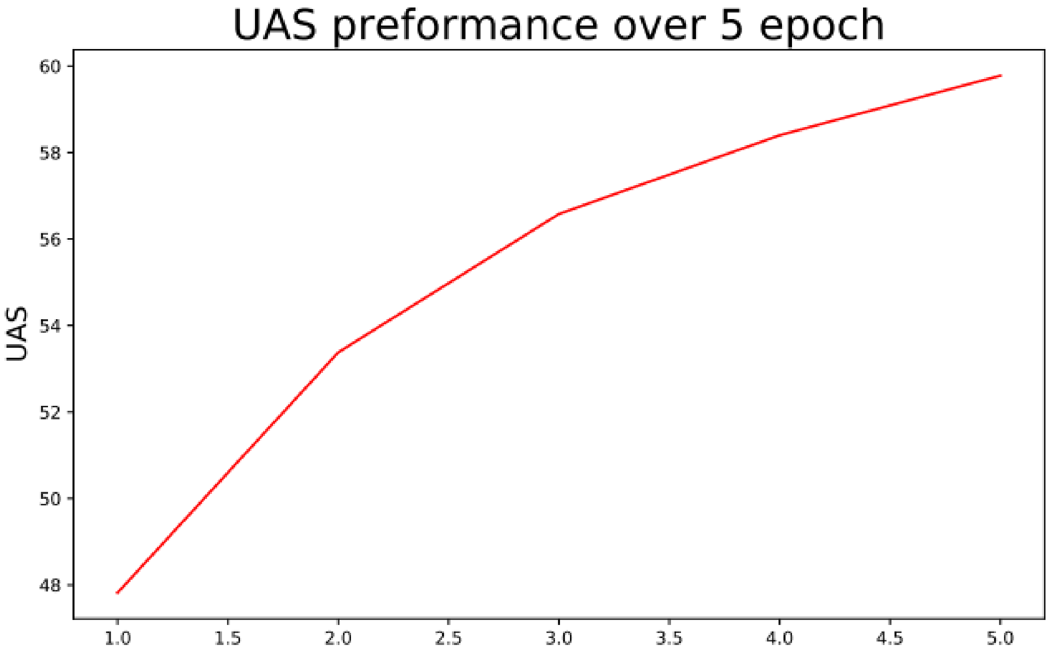
The performance plots are as below:



We can find the difference between 2 training can be ignored. The variance is relatively small. The loss is rapidly decreased in the first stage and becomes stable in a short time.



The accuracy plot also shows the stability of these two models. They become stable in the 5th epoch. Also, these two training model do not have a large difference.



Second step: use Adam optimizer

The UAS of the 5th epoch is around 60, which is also acceptable. To warp up, our model is stable and relatively accurate, so the training model can stop here.

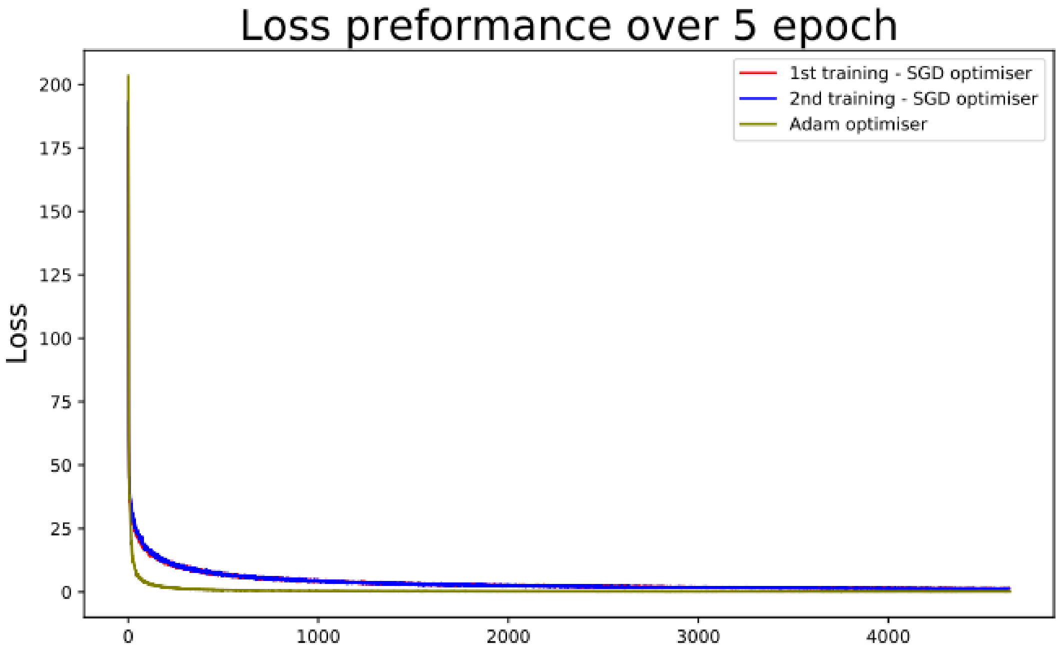
The highest performance I can get is changing the SGD optimizer to the Adam optimizer. The only changes in the code is changing

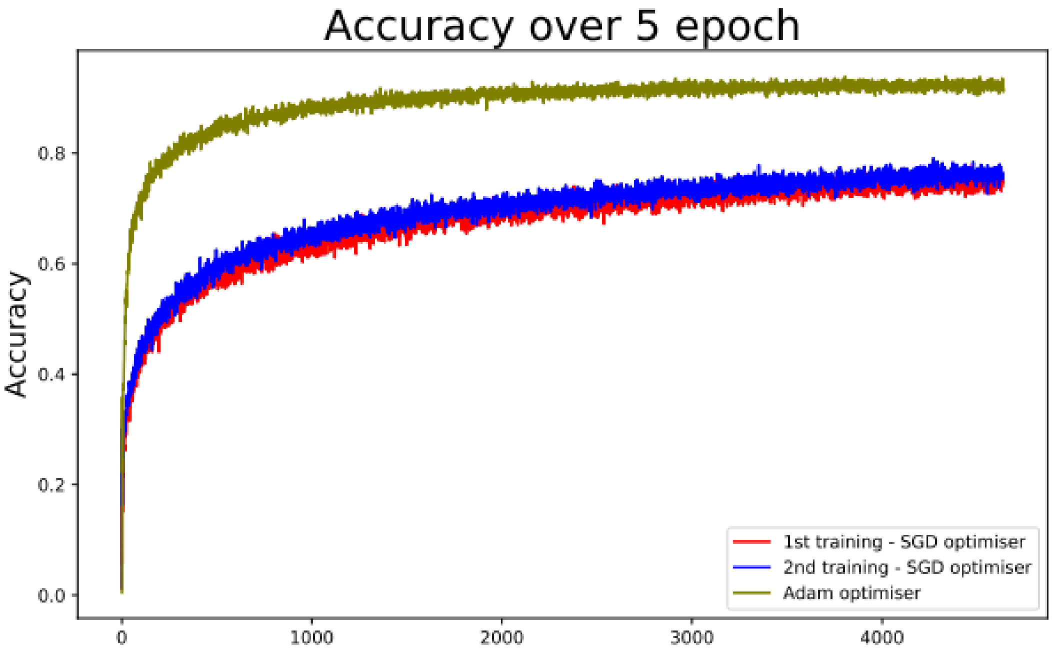
```
optimizer = torch.optim.SGD(parser.parameters(), lr=lr)
```

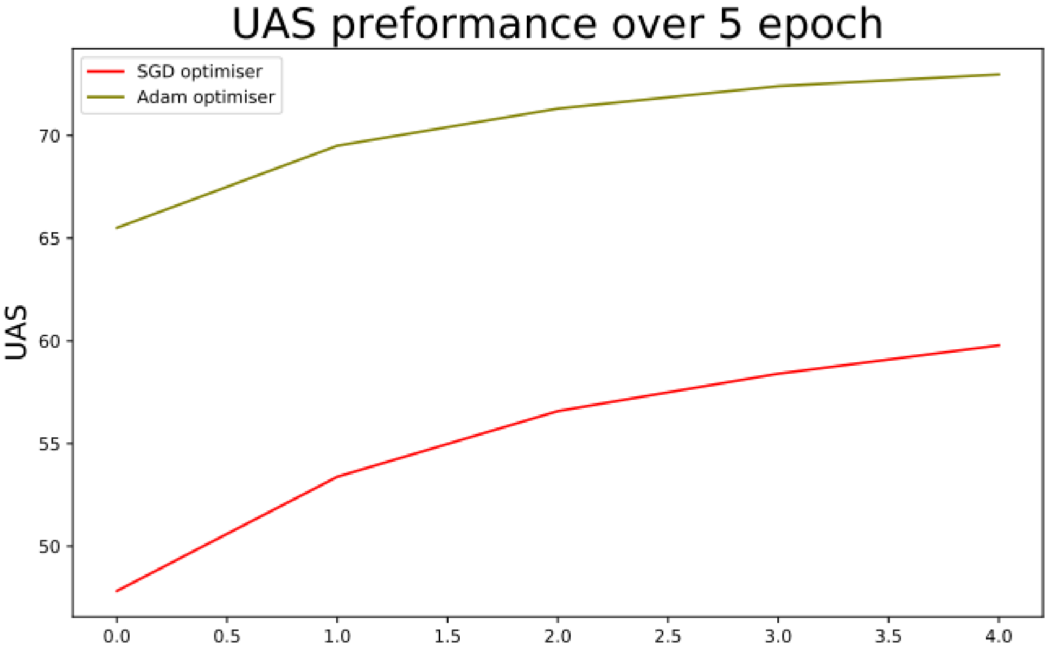
to

```
optimizer = torch.optim.Adam (parser.parameters(), lr=lr)
```

And the performance improvement plots are as below:







In 5 epoch, the performance of the new model experiences a tremendous improve. The accuracy of the new model reaches to 0.93 while the old model stables at 0.75 at the 5th epoch. The loss performance and the UAS score are as well.

Next step, evaluation.

For the **SGD optimizer** model, we can get result as below:

```
----
buffer: ['the', 'big', 'dog', 'ate', 'my', 'homework']
stack:  ['<root>']
action: shift
----
buffer: ['big', 'dog', 'ate', 'my', 'homework']
stack:  ['<root>', 'the']
action: shift
----
buffer: ['dog', 'ate', 'my', 'homework']
stack:  ['<root>', 'the', 'big']
action: shift
----
buffer: ['ate', 'my', 'homework']
stack:  ['<root>', 'the', 'big', 'dog']
action: shift
----
buffer: ['my', 'homework']
stack:  ['<root>', 'the', 'big', 'dog', 'ate']
action: left arc, <d>:compound
----
buffer: ['my', 'homework']
stack:  ['<root>', 'the', 'big', 'ate']
action: left arc, <d>:amod
----
buffer: ['my', 'homework']
stack:  ['<root>', 'the', 'ate']
action: left arc, <d>:det
----
buffer: ['my', 'homework']
stack:  ['<root>', 'ate']
action: shift
----
buffer: ['homework']
stack:  ['<root>', 'ate', 'my']
action: shift
----
buffer: []
stack:  ['<root>', 'ate', 'my', 'homework']
action: right arc, <d>:punct
----
buffer: []
stack:  ['<root>', 'ate', 'my']
action: right arc, <d>:punct
```

```
----
buffer: []
stack:  ['<root>', 'ate']
action: right arc, <d>:root
   <root>
     |
    ate
  _____|___________
 |    |     |     my
 |    |     |     |
the  big   dog  homework
```

And the **Adam model** can give us a more reasonable structure:

```
----
buffer: ['the', 'big', 'dog', 'ate', 'my', 'homework']
stack:  ['<root>']
action: shift
----
buffer: ['big', 'dog', 'ate', 'my', 'homework']
stack:  ['<root>', 'the']
action: shift
----
buffer: ['dog', 'ate', 'my', 'homework']
stack:  ['<root>', 'the', 'big']
action: shift
----
buffer: ['ate', 'my', 'homework']
stack:  ['<root>', 'the', 'big', 'dog']
action: shift
----
buffer: ['my', 'homework']
stack:  ['<root>', 'the', 'big', 'dog', 'ate']
action: shift
----
buffer: ['homework']
stack:  ['<root>', 'the', 'big', 'dog', 'ate', 'my']
action: shift
----
buffer: []
stack:  ['<root>', 'the', 'big', 'dog', 'ate', 'my', 'homework']
action: left arc, <d>:csubjpass
----
buffer: []
stack:  ['<root>', 'the', 'big', 'dog', 'ate', 'homework']
action: right arc, <d>:<null>
----
```

```
buffer: []
stack:  ['<root>', 'the', 'big', 'dog', 'ate']
action: left arc, <d>:cop
----
buffer: []
stack:  ['<root>', 'the', 'big', 'ate']
action: left arc, <d>:xcomp
----
buffer: []
stack:  ['<root>', 'the', 'ate']
action: left arc, <d>:expl
----
buffer: []
stack:  ['<root>', 'ate']
action: right arc, <d>:<null>
    <root>
      |
     ate
  _____|___________
 |     |      |    homework
 |     |      |         |
the  big   dog    my
```

However, a more reasonable result should be like this:

```
          <root>
             |
           ate
  ___ ___|___________
  dog            homework
 |     |                  |
the  big               my
```

And the **correct operation** at each step:

```
----
buffer: ['the', 'big', 'dog', 'ate', 'my', 'homework']
stack:  ['<root>']
action: shift
----
buffer: ['big', 'dog', 'ate', 'my', 'homework']
stack:  ['<root>', 'the']
action: shift
----
buffer: ['dog', 'ate', 'my', 'homework']
stack:  ['<root>', 'the', 'big']
action: shift
----
buffer: ['ate', 'my', 'homework']
```

```
stack:  ['<root>', 'the', 'big', 'dog']
action: left arc
----
buffer: ['ate', 'my', 'homework']
stack:  ['<root>', 'the', 'dog']
action: left arc
----
buffer: ['ate', 'my', 'homework']
stack:  ['<root>',  'dog']
action: shift
----
buffer: ['my', 'homework']
stack:  ['<root>', 'dog', 'ate']
action: left arc
----
buffer: ['my', 'homework']
stack:  ['<root>', 'ate']
action: shift
----
buffer: ['homework']
stack:  ['<root>', 'ate', 'my']
action: shift
----
buffer: []
stack:  ['<root>', 'ate', 'my', 'homework']
action: left arc, <d>:csubjpass
----
buffer: []
stack:  ['<root>', 'ate', 'homework']
action: right arc, <d>:<null>
----
buffer: []
stack:  ['<root>', 'ate']
action: left arc, <d>:cop
```

The operation which should be modified has been highlighted above.