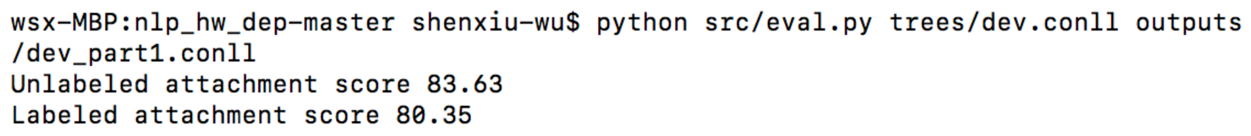
hw3 code part   Name: Shenxiu Wu   UNI: sw3196

Part 1

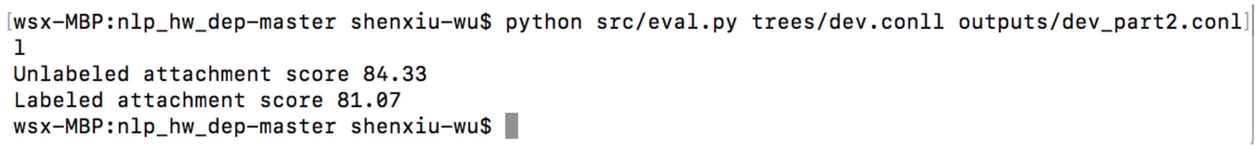
The results are shown as below screenshot: Unlabeled attachment score 83.63; Labeled attachment score 80.35



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs
/dev_part1.conll
Unlabeled attachment score 83.63
Labeled attachment score 80.35
```

Part 2

The results are shown as below screenshot: Unlabeled attachment score 84.33; Labeled attachment score 81.07

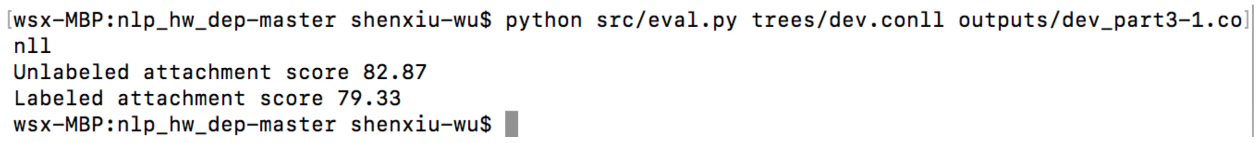


```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part2.conl
l
Unlabeled attachment score 84.33
Labeled attachment score 81.07
wsx-MBP:nlp_hw_dep-master shenxiu-wu$
```

As we can see above, there is a slight improvement in the results as compared with the results of part 1. I think this is because we increase the first and second hidden layer dimensions and as a result complexity of this model increases and more neurons are allowed to compute in this model.

Part 3

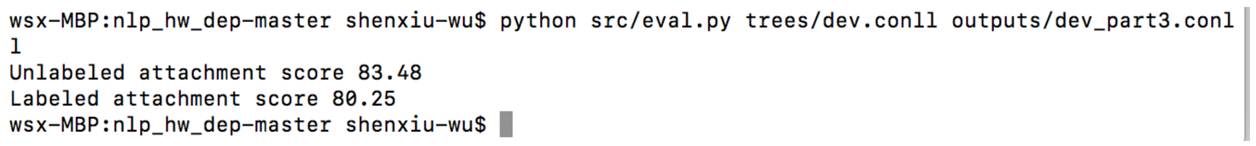
1. I firstly change the mini-batch size from 1000 to 1500. Both dimensions of hidden layers are equal to 200. All other parameters' value is the same in part 1. The screenshot of results is as follows.



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part3-1.co
nll
Unlabeled attachment score 82.87
Labeled attachment score 79.33
wsx-MBP:nlp_hw_dep-master shenxiu-wu$
```

We can see that compared to the part1 case, only increasing the minibatch size decreases unlabeled attachment and labeled attachment accuracies by roughly 0.8 percentage point. I think this may due to overfitting. So, I continue to try another configuration.

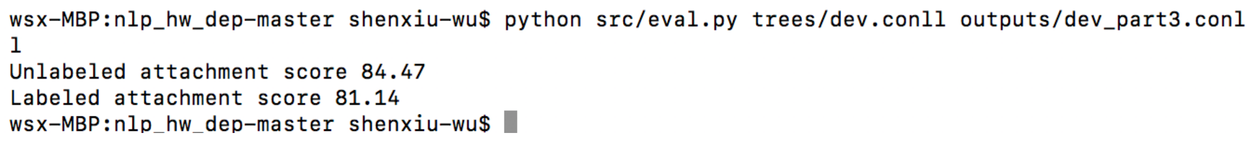
2. Try: mini-batch size = 1500. d_{h1} = 400, d_{h2} = 400. Epochs = 8. Other parameters are the same in part 1.



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part3.conl
l
Unlabeled attachment score 83.48
Labeled attachment score 80.25
wsx-MBP:nlp_hw_dep-master shenxiu-wu$
```

As we could see, unlabeled attachment and labeled attachment accuracies increase by 0.7-0.8 percentage, even though we have increased the epochs. This indicates that not enough complexity might have been achieved with original 200 dimensions of both hidden layers in part1. But does the dimensions bigger, the accuracy more been improved? So I try another configuration as follows.

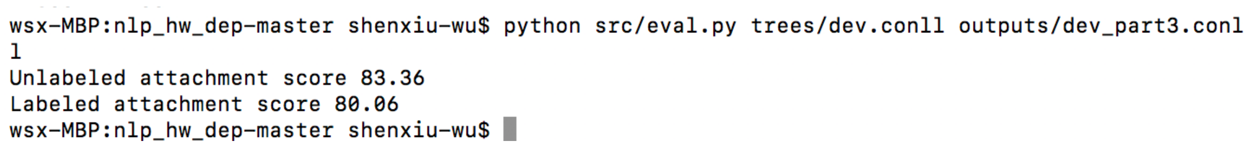
3. Try: $d_{h1} = 500$, $d_{h2} = 500$. Other parameters are the same in part 2.



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part3.conl
l
Unlabeled attachment score 84.47
Labeled attachment score 81.14
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ 
```

As we can see, not much improvements happened in results when compared with the results of part 2. I think this is because enough complexity might have been achieved with 400 dimensions with both first and second hidden layers. What's more, this is time-consuming which almost cost me 50 minutes. But, in subsequent tests, if the data complexity (like word embeddings) increased later, I should still consider use this configuration when 400 dimensions are not enough.

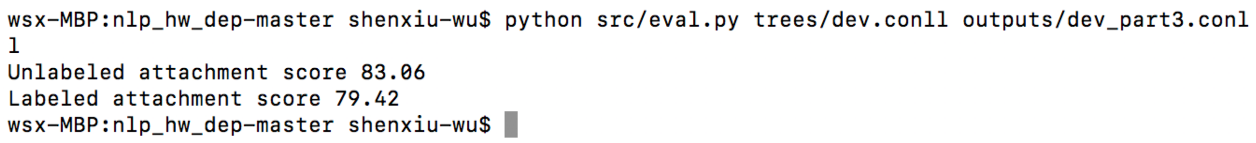
4. Try: Word embedding dimension = 128, POS embedding dimension = 64, Dependency embedding dimension = 64. Other parameters are the same in part 2.



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part3.conl
l
Unlabeled attachment score 83.36
Labeled attachment score 80.06
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ 
```

The result is not improved compared with the one in part 2. And what's worse is the setting is tremendously time-consuming which cost me 70 minutes to get the result. Thus, the increase in embedding dimension is a not an appropriate way and the value set in part 2 is enough. Thus, we should use Word embedding dimension = 64, POS embedding dimension = 32, Dependency embedding dimension = 32 in subsequent tests.

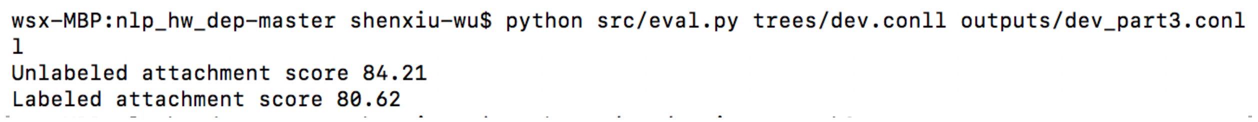
5. Try: set the probability p of dropout during training equals to 0.1. Other parameters are the same in part 2.



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part3.conl
l
Unlabeled attachment score 83.06
Labeled attachment score 79.42
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ 
```

The result doesn't reach my expectation and the performance degraded compared with the result of part 2. I think this is because the model data is just appropriate and after dropout, the model become so simple where it cannot provide enough data to train.

6. Try: Initializing word embeddings with pre-trained vectors from glove; word dim = 200 d_{h1} = 500, d_{h2} = 500. Other parameters are the same in part 2. The result is as in the following screenshot.



```
wsx-MBP:nlp_hw_dep-master shenxiu-wu$ python src/eval.py trees/dev.conll outputs/dev_part3.conl
l
Unlabeled attachment score 84.21
Labeled attachment score 80.62
```

Having a good performance with a high time complexity.

As from these above tests we could conclude, the best model is from step3(i.e. model3-3). It gives the best results obtained so far and even when compared with part 2, it still has a little improvement. Finally, I use this best model to output outputs/dev_part3.conll and outputs/test_part3.conll files.