Homework 4 – Yutae Lee

1 Question 1 (Coding Task)

(a)

For this part we are asked to provide the size of each training parameter that was listed:

H(hidden transfomation matrix): $D \times D$
I(input transfomation matrix): $d \times D$
U(output transformation matrix): $D \times |V|$
b_1(bias for recurrent layer): $1 \times D$
b_2(bias for projection layer): $1 \times |V|$

(b)

In this part we need to show the relationship between the cross-entropy and perplexity

We learned what cross-entropy loss is from the beginning of the course which is

$H(y, \hat{y}) = -\sum y_t log(\hat{y}_t) = -log(\hat{y}_c)$ with class c.

The perplexity loss is actually inverse probability:

$PP^{(t)}(y^{(T)}, \hat{y}^{(t)}) = \frac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \hat{y}_j^{(t)}}$

Taking the log of that I get $-log(\sum_{j=1}^{|V|} y_j^{(t)} \hat{y}_j^{(t)}) = -log(\hat{y}_c)$ which is the cross-entropy.

Thus, we can conclude that the cross entropy is taking the log of preplexity loss.

i.e.

$CrossEntropy(y, \hat{y}) = log(PP(y, \hat{y}))$

(c)

This part is embedded in the code.

2 Question 2 (Attention mechanism)

(a)

In this part we have to compare the number of parameters between the single-head and multi-head attention.

For the single-head attention:

We have set that $W^Q \in \mathbb{R}^{d\times d}$, $W^K \in \mathbb{R}^{d\times d}$, $W^V \in \mathbb{R}^{d\times d}$
From this we can get that for the single-head attention the number of parameters are $d \times d \times 3 = 3d^2$
For the multi-head attention:

We have set that $W_i^Q \in \mathbb{R}^{d\times \frac{d}{h}}$, $W_i^K \in \mathbb{R}^{d\times \frac{d}{h}}$, $W_i^V \in \mathbb{R}^{d\times \frac{d}{h}}$

From this we can get that for the multi-head attention the number of parameters are $d \times \frac{d}{h} \times 3 = 3d^2/h$ for each head. And since we have h heads which means that the number of paramter in total is actually $3d^2$ as well. Which is same as single-head attention.
(b)
In this part we have compare the amount of computation between the single-head and multi-head notation:

For Single-head:
Note that the computation of matrix multiplication $(K \times M) \times (M \times N)$ is $\mathcal{O}(KMN)$.

Knowing this Computing QW^Q, KW^K, VW^V are nd^2 each. Which we get $\mathcal{O}(3nd^2)$ in total.
Computing $QW^Q(KW^K)^T$:

Since we know that QW^Q and $(KW^K)^T$ are $\mathbb{R}^{n\times d}$ and $\mathbb{R}^{d\times n}$ respectively.
Then we can easily get that the computation of doing that part before we are putting it into softmax is about $\mathcal{O}(n^2d)$

Computing softmax of $QW^Q(KW^K)^T$:

Since this is matrix of $\mathbb{R}^{n\times n}$ the computation would be $\mathcal{O}(n^2 + n)$

Lastly multiplying $QW^Q(KW^K)^T$ with VW^V is $\mathcal{O}(n^2d)$

In total I get $\mathcal{O}(3nd^2 + 2n^2d + n^2 + n)$ as a computational complexity for single-head algorithm.

For Multi-head:
The step is similar with just $\mathcal{O}(nd^2/h)$ for QW_i^Q, KW_i^K, VW_i^V each.
And $QW_i^Q(KW_i^K)^T$, $softmax(\dots)\times$ VW_i^V are $\mathcal{O}(n^2d/h)$
In total I get $\mathcal{O}(h \times (3nd^2/h + 2n^2d/h + n^2 + n)$
$= \mathcal{O}(3nd^2 + 2n^2d + n^2h + nh)$

Considering that h is just a constant, we can say that the computation complexity between the two are about similar.

3 Question 3 (GCN)

(a)
To fix problem of each center node sums up feature vectors of all neighboring nodes but not the nodes itself, you simply need to add Identity matrix with same dimension to make $\tilde{A}$

i.e.
$\tilde{A} = A + I_n$

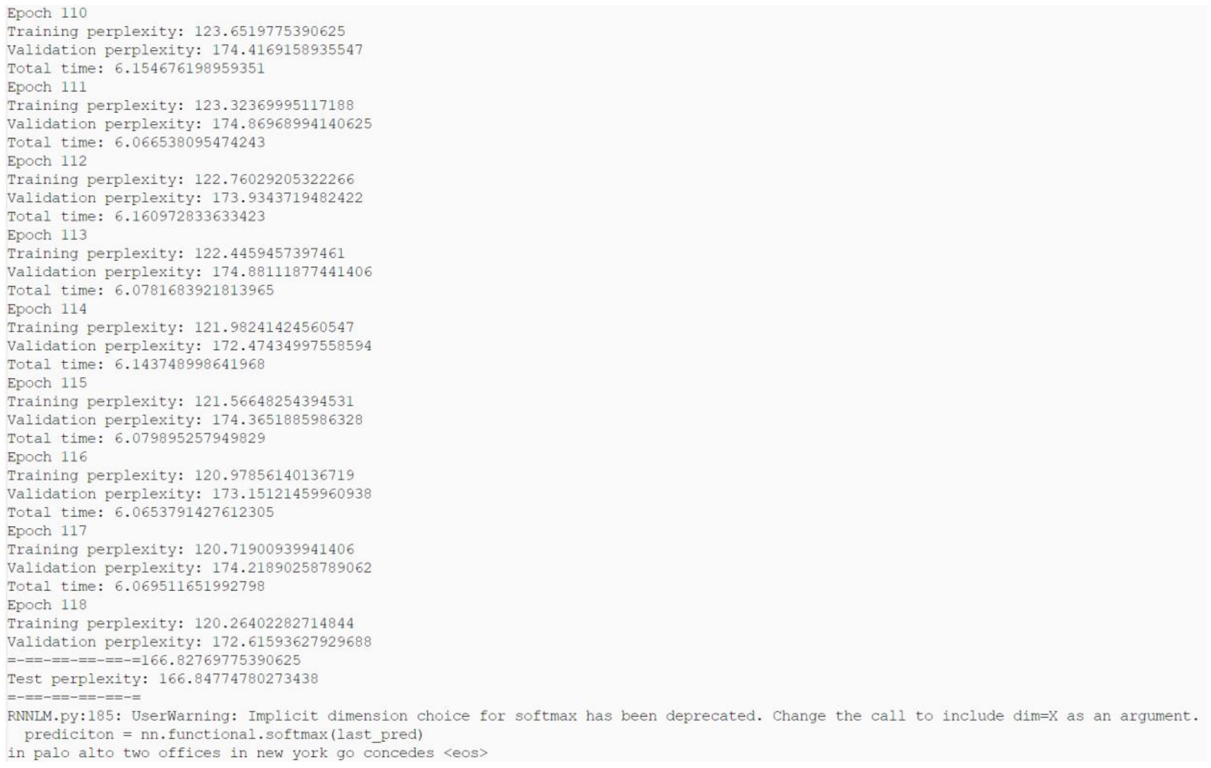
(a)
To normalise such that all rows sum to one, we have to get the $\hat{D}$ where it is diagonal node degree matrix.

Then if we diagonalize it by getting $\hat{D}^{-1/2}$ i.e. $\hat{D}^{-1/2}\tilde{A}\hat{D}^{-1/2}$

Part 2: Results and Analysis for Coding Task

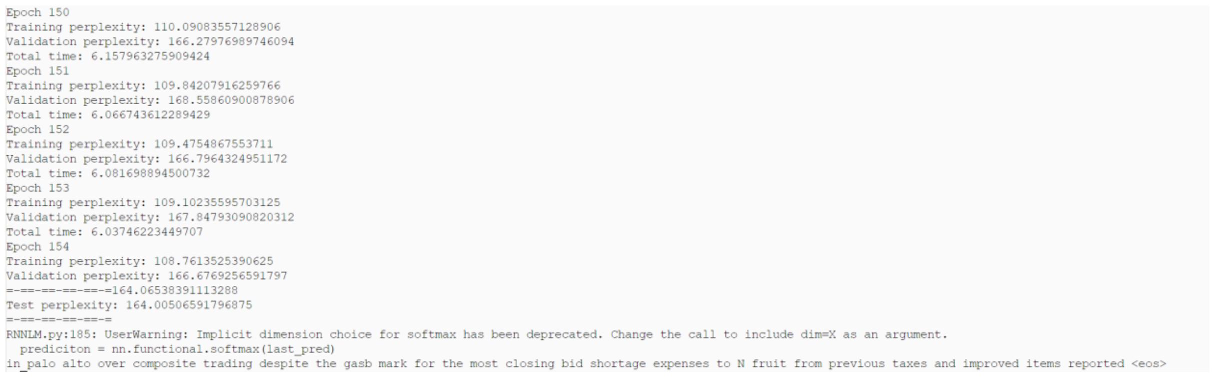
RESULTS:

Batch size = 128, Embed size = 265, Hidden size = 512, Number of steps = 10, Early Stopping = 3, Dropout = 0.1 LR = 0.001



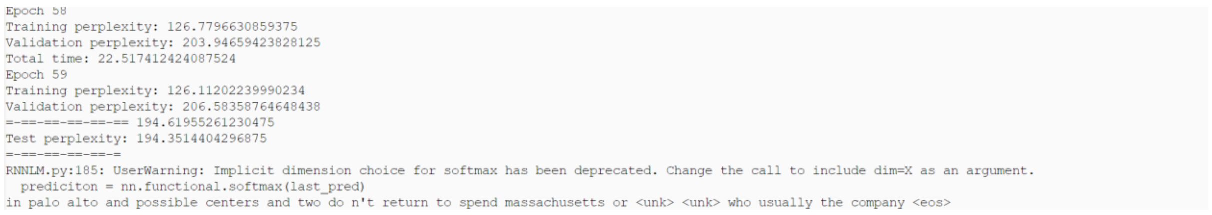
```
Epoch 110
Training perplexity: 123.6519775390625
Validation perplexity: 174.4169158935547
Total time: 6.154676198959351
Epoch 111
Training perplexity: 123.32369995117188
Validation perplexity: 174.86968994140625
Total time: 6.066538095474243
Epoch 112
Training perplexity: 122.76029205322266
Validation perplexity: 173.9343719482422
Total time: 6.160972833633423
Epoch 113
Training perplexity: 122.4459457397461
Validation perplexity: 174.88111877441406
Total time: 6.0781683921813965
Epoch 114
Training perplexity: 121.98241424560547
Validation perplexity: 172.47434997558594
Total time: 6.143748998641968
Epoch 115
Training perplexity: 121.56648254394531
Validation perplexity: 174.3651885986328
Total time: 6.079895257949829
Epoch 116
Training perplexity: 120.97856140136719
Validation perplexity: 173.15121459960938
Total time: 6.0653791427612305
Epoch 117
Training perplexity: 120.71900939941406
Validation perplexity: 174.21890258789062
Total time: 6.069511651992798
Epoch 118
Training perplexity: 120.26402282714844
Validation perplexity: 172.61593627929688
=-==-==-==-==-=166.82769775390625
Test perplexity: 166.84774780273438
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in palo alto two offices in new york go concedes <eos>
```

Batch size = 128, Embed size = 256, Hidden size = 512, Number of steps = 10, Early Stopping = 3, Dropout = 0.1 LR = 0.001



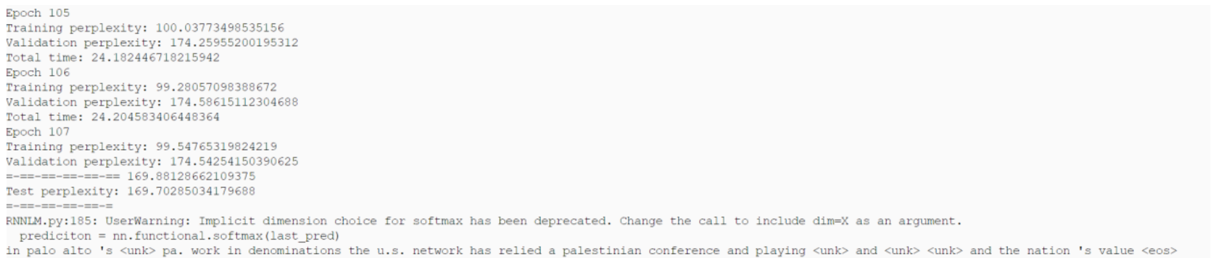
```
Epoch 150
Training perplexity: 110.09083557128906
Validation perplexity: 166.27976989746094
Total time: 6.157963275909424
Epoch 151
Training perplexity: 109.84207916259766
Validation perplexity: 168.55860900878906
Total time: 6.066743612289429
Epoch 152
Training perplexity: 109.4754867553711
Validation perplexity: 166.7964324951172
Total time: 6.081698894500732
Epoch 153
Training perplexity: 109.10235595703125
Validation perplexity: 167.84793090820312
Total time: 6.03746223449707
Epoch 154
Training perplexity: 108.7613525390625
Validation perplexity: 166.6769256591797
=-==-==-==-==-=164.06538391113288
Test perplexity: 164.00506591796875
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in palo alto over composite trading despite the gasb mark for the most closing bid shortage expenses to N fruit from previous taxes and improved items reported <eos>
```

Batch size = 64, Embed size = 128, Hidden size = 512, Number of steps = 10, Early Stopping = 3, Dropout = 0.1 LR = 0.005



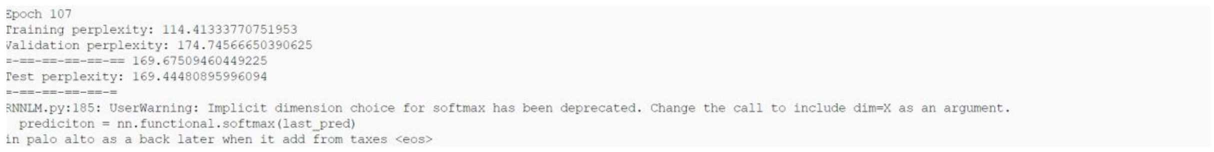
```
Epoch 58
Training perplexity: 126.7796630859375
Validation perplexity: 203.94659423828125
Total time: 22.517412424087524
Epoch 59
Training perplexity: 126.11202239990234
Validation perplexity: 206.58358764648438
=-==-==-==-==-== 194.61955261230475
Test perplexity: 194.3514404296875
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in palo alto and possible centers and two do n't return to spend massachusetts or <unk> <unk> who usually the company <eos>
```

Batch size = 64, Embed size = 256, Hidden size = 512, Number of steps = 10, Early Stopping = 3, Dropout = 0.1 LR = 0.001



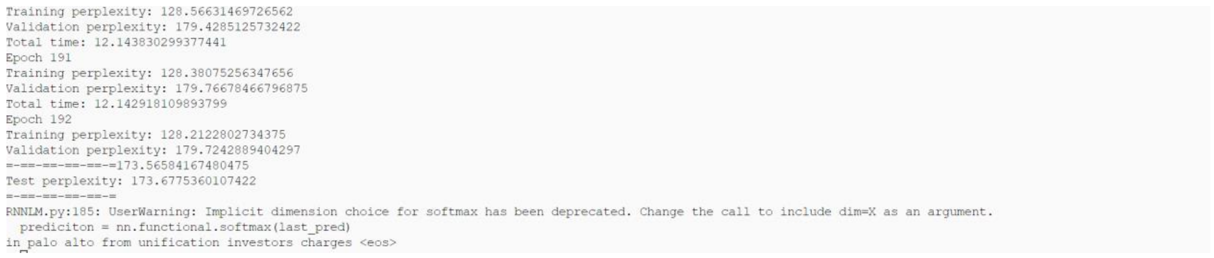
```
Epoch 105
Training perplexity: 100.03773498535156
Validation perplexity: 174.25955200195312
Total time: 24.182446718215942
Epoch 106
Training perplexity: 99.28057098388672
Validation perplexity: 174.58615112304688
Total time: 24.204583406448364
Epoch 107
Training perplexity: 99.54765319824219
Validation perplexity: 174.54254150390625
=-==-==-==-==-== 169.88128662109375
Test perplexity: 169.70285034179688
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in palo alto 's <unk> pa. work in denominations the u.s. network has relied a palestinian conference and playing <unk> and <unk> <unk> and the nation 's value <eos>
```

Batch size = 128, Embed size = 256, Hidden size = 512, Number of steps = 5, Early Stopping = 3, Dropout = 0.1 LR = 0.0005



```
Epoch 107
Training perplexity: 114.41333770751953
Validation perplexity: 174.74566650390625
=-==-==-==-==-== 169.67509460449225
Test perplexity: 169.44480895996094
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in palo alto as a back later when it add from taxes <eos>
```

Batch size = 128, Embed size = 256, Hidden size = 256, Number of steps = 10, Early Stopping = 5, Dropout = 0.1 LR = 0.001



```
Training perplexity: 128.56631469726562
Validation perplexity: 179.4285125732422
Total time: 12.143830299377441
Epoch 191
Training perplexity: 128.38075256347656
Validation perplexity: 179.76678466796875
Total time: 12.142918109893799
Epoch 192
Training perplexity: 128.2122802734375
Validation perplexity: 179.7242889404297
=-==-==-==-==-=173.56584167480475
Test perplexity: 173.6775360107422
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in palo alto from unification investors charges <eos>
```

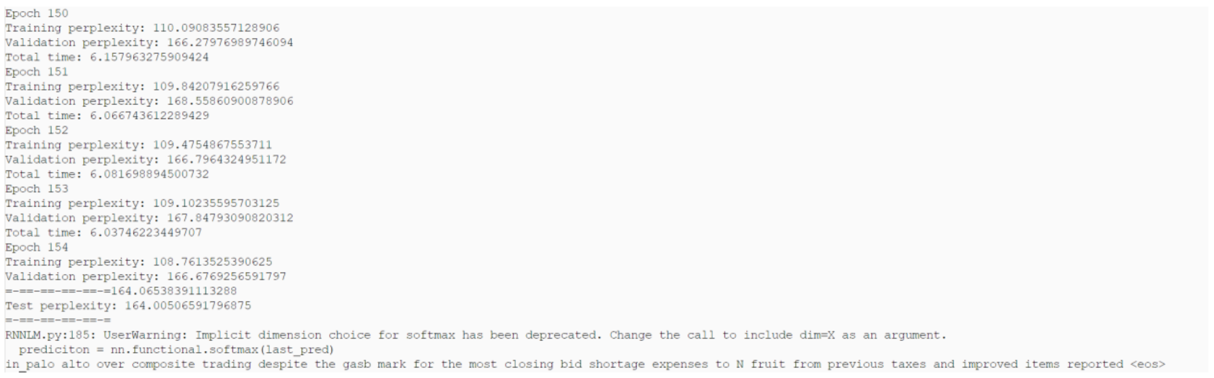
Analysis:

From the Result, I was able to find at in general having Batch Size of 128, Embed Size of 256 and Hidden Size of 512 returned the best perplexity. Also with bigger batch size, the amount of time that it took per epoch reduced significantly. Also, I realized that if the learning rate is too big, the epoch stops too early and test perplexity is not as good because there hasn't been sufficient epochs to train. However, having bigger size of Early Stopping does not necessarily mean that it will give you better result, because when I had Early Stopping = 5, which is greater than other experiment, it did not show significant improvement. In fact, the test perplexity compared to the best model where I had early stopping = 3 was significantly worse.

Thus, the best sets of hyperparameter was

Batch size = 128, Embed size = 256, Hidden size = 512, Number of steps = 10, Early Stopping = 3, Dropout = 0.1 LR = 0.001.

Which returned 164.00506591796875 of Test Perplexity



```
Epoch 150
Training perplexity: 110.09083557128906
Validation perplexity: 166.27976989746094
Total time: 6.157963275909424
Epoch 151
Training perplexity: 109.84207916259766
Validation perplexity: 168.55860900878906
Total time: 6.066743612289429
Epoch 152
Training perplexity: 109.4754867553711
Validation perplexity: 166.7964324951172
Total time: 6.081698894500732
Epoch 153
Training perplexity: 109.10235595703125
Validation perplexity: 167.84793090820312
Total time: 6.03746223449707
Epoch 154
Training perplexity: 108.7613525390625
Validation perplexity: 166.6769256591797
=-==-==-==-==-=164.06538391113288
Test perplexity: 164.00506591796875
=-==-==-==-==-=
RNNLM.py:185: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  prediciton = nn.functional.softmax(last_pred)
in_palo alto over composite trading despite the gasb mark for the most closing bid shortage expenses to N fruit from previous taxes and improved items reported <eos>
```

Looking at the sentence that was generated by the best model (in_palo alto over composit trading despite the gasb mark for the most clsing bid shortage expenses to N fruit from previous taxes and improved items reported <eos>), I can see that ordering of the sentence regarding grammar sometimes make sense. However, the overall performance of the model is not satisfactory upto point where we can rely. A way to maybe improve this model may be significantly increasing the embedding size and hidden size with ratio of 1 to 2. However, this cannot actually be done with my technology as I do not have GPU and computer that can process that without disconnecting.