CSC321 A1

February 2018

1 Part 1.

1.1 Q1

Total number of trainable parameters: 250*128 + 128*48 + 250 *16 + 250 + 128 = 42522

The hidden layer to output part has the largest amount of trainable parameters.

1.2 Q2

The table from lecture would have $250^3 * 250 = 3906250000$ entries.

2 Part 2.See codes

2.1 My codes:

```
return output_activations - expanded_target_batch

hid_to_output_weights_grad =np.dot(loss_derivative.T,activations.hidden_layer)
output_bias_grad = np.dot(loss_derivative.T, np.ones(100))
embed_to_hid_weights_grad = np.dot(hid_deriv.T, activations.embedding_layer)
hid_bias_grad = np.dot(hid_deriv.T, np.ones(100))
```

2.2 Print checking

```
loss_derivative[2, 5] 0.0013789153741
loss_derivative[2, 121] -0.999459885968
loss_derivative[5, 33] 0.000391942483563
loss_derivative[5, 31] -0.708749715825

param_gradient.word_embedding_weights[27, 2] -0.298510438589
param_gradient.word_embedding_weights[43, 3] -1.13004162742
param_gradient.word_embedding_weights[22, 4] -0.211118814492
param_gradient.word_embedding_weights[2, 5] 0.0
param_gradient.embed_to_hid_weights[10, 2] -0.0128399532941
param_gradient.embed_to_hid_weights[15, 3] 0.0937808780803
param_gradient.embed_to_hid_weights[30, 9] -0.16837240452
param_gradient.embed_to_hid_weights[35, 21] 0.0619595914046

param_gradient.hid_bias[10] -0.125907091215
param_gradient.hid_bias[20] -0.389817847348

param_gradient.output_bias[0] -2.23233392034
param_gradient.output_bias[1] 0.0333102255428
param_gradient.output_bias[2] -0.743090094025
param_gradient.output_bias[3] 0.162372657748
```

3 Part 3.

1. 'life','in', 'the'. The model predicts words such as game, city, which are pretty sensible.

```
model.predict_next_word('life', 'in', 'the')
life in the world Prob: 0.13516
life in the game Prob: 0.07251
life in the street Prob: 0.04327
life in the house Prob: 0.04150
life in the city Prob: 0.03721
life in the country Prob: 0.03480
life in the united Prob: 0.03299
life in the end Prob: 0.02759
life in the business Prob: 0.02640
life in the office Prob: 0.02450
```

'can','she','think' makes plausible predictions of words not in the training set.

```
lm.find_occurrences('can', 'she', 'think')
The tri-gram "can she think" did not occur in the training set.
model.predict_next_word('can', 'she','think')
can she think of Prob: 0.34233
can she think about Prob: 0.14897
can she think ? Prob: 0.12530
can she think . Prob: 0.03968
can she think we Prob: 0.03443
can she think i Prob: 0.03349
can she think it Prob: 0.03141
can she think that Prob: 0.02921
can she think so Prob: 0.02308
can she think this Prob: 0.02262
```

2. Examples of obvious clusters are:

- ”may, will, might, would, can, do, did, does” modal verbs
- ”about, against, into, through, for, like, of” prepositions
- ”five, three, two, seven, four, these, those, many, few” describing amounts
- ”day, year, night, days, time” describing time
- ”family, business, school, department...” nouns

Words are clustered either by their part of speech or their meaning.

3. No, in the two-dimensional representation, they are apart from each other. When we examine the distance between two words, we got

```
model.word_distance('new', 'york')
3.4396865351240438
```

And none of the word is in top ten closest words of each other.

```
model.display_nearest_words('new')
old: 2.5491467235
big: 2.64267006663
white: 2.73571273677
several: 2.83164705067
american: 2.93591566157
good: 2.94273501286
national: 2.97870321999
our: 2.99454425612
such: 3.02435200146
five: 3.10445680156

model.display_nearest_words('york', k=10)
john: 1.06856248964
public: 1.13634954107
political: 1.16210867768
general: 1.18044218468
ms.: 1.18280934796
national: 1.20183375286
state: 1.2021534661
department: 1.20337172887
west: 1.21182723776
school: 1.21855051002
```

The main reason is that we cluster the word based on their property, rather than how they are used. For example, new and old are closer together. On the other hand, new can be used in many ways, the frequency of "new york" as a sequence in the sentences might be less common than other usage of "new".

4. ('government', 'university') is closer together in the learned representation, which is measured by method 'word_distance' using the Euclidean distance metric.

```
model.word_distance('government', 'university')
1.2505099560359767
model.word_distance('government', 'political')
1.4425390261983047
```

From the way words are clustered, we see that words are closer when they have similar usage. In our case, because government and university are both nouns, they are more frequently to appear after similar words, and hence appear to be closer than government and political, which occurs in a fewer cases.