CSS321 ASSIGNMENT1 Name: Jiawei Li Stu.No: 999563823

PartI

1(a).

Total number of trainable parameter in this network is

250*16 + 48*128 + 128 + 128*250 + 250 = 42522

The hidden to output activation has the most number of trainable parameter.

1(b)

There are 250^4 entries in this table, since there are 250 possible choice in vocabularies and we choose 4 words out 250 words with replacement.

PartII

Two functions are below:

Compute_loss_derivative:

```
def compute_loss_derivative(self, output_activations, expanded_target_batch):
    """Compute the derivative of the cross-entropy loss function with respect to the inputs
    to the output units. In particular, the output layer computes the softmax

        y_i = e^{z_i} / \sum_j e^{z_j}.

    This function should return a batch_size x vocab_size matrix, where the (i, j) entry
    is dC / dz_j computed for the ith training case, where C is the loss function

        C = -sum(t_i log y_i).

    The arguments are as follows:

        output_activations - the activations of the output layer, i.e. the y_i's.
        expanded_target_batch - each row is the indicator vector for a target word,
            i.e. the (i, j) entry is 1 if the i'th word is j, and 0 otherwise."""

    ############################   YOUR CODE HERE  ##############################

    return (output_activations - expanded_target_batch)

    ############################################################################
```

Back_propagate:

```
def back_propagate(self, input_batch, activations, loss_derivative):
    """Compute the gradient of the loss function with respect to the trainable parameters
    of the model. The arguments are as follows:

        input_batch - the indices of the context words
        activations - an Activations class representing the output of Model.compute_activations
        loss_derivative - the matrix of derivatives computed by compute_loss_derivative

    Part of this function is already completed, but you need to fill in the derivative
    computations for hid_to_output_weights_grad, output_bias_grad, embed_to_hid_weights_grad,
    and hid_bias_grad. See the documentation for the Params class for a description of what
    these matrices represent."""

    # The matrix with values dC / dz_j, where dz_j is the input to the jth hidden unit,
    # i.e. y_j = 1 / (1 + e^{-z_j})
    hid_deriv = np.dot(loss_derivative, self.params.hid_to_output_weights) \
                * activations.hidden_layer * (1. - activations.hidden_layer)
```

```
############################   YOUR CODE HERE  ##############################

'''OUTPUT LAYER.'''
hid_to_output_weights_grad = np.dot(loss_derivative.T, activations.hidden_layer)
output_bias_grad = sum(loss_derivative)

'''HIDDEN LAYER.'''
embed_to_hid_weights_grad = np.dot(hid_deriv.T, activations.embedding_layer)
hid_bias_grad = sum(hid_deriv)

###########################################################################
# The matrix of derivatives for the embedding layer
embed_deriv = np.dot(hid_deriv, self.params.embed_to_hid_weights)

# Embedding layer
word_embedding_weights_grad = np.zeros((self.vocab_size, self.embedding_dim))
for w in range(self.context_len):
    word_embedding_weights_grad += np.dot(self.indicator_matrix(input_batch[:, w]).T,
                                          embed_deriv[:, w*self.embedding_dim:(w+1)*self.embedding_dim])

return Params(word_embedding_weights_grad, embed_to_hid_weights_grad, hid_to_output_weights_grad,
              hid_bias_grad, output_bias_grad)
```

Output of checking gradient:
In [7]: check_gradients()
The loss derivative looks OK.
The gradient for word_embedding_weights looks OK.
The gradient for embed_to_hid_weights looks OK.
The gradient for hid_to_output_weights looks OK.
The gradient for hid_bias looks OK.
The gradient for output_bias looks OK.

PartIII

			first	second	third
government	of	united	own	life	states
city	of	new	york	.	?
life	in	the	world	game	united
he	is	the	best	same	only

Some predictions here make sense like "government of united states", however this combination doesn't appear in the data set. Meanwhile, there are still some predictions doesn't makes sense at all, like "government of united own". We may need a large data sets to avoid this.

1.
In [25]: model.predict_next_word("government","of","united",3)
government of united own Prob: 0.18486
government of united life Prob: 0.10271
government of united states Prob: 0.05529

In [36]: language_model.find_occurrences("government","of","united")
The tri-gram "government of united" did not occur in the training set.

In [26]: model.predict_next_word("city","of","new",3)
city of new york Prob: 0.95655
city of new . Prob: 0.01289
city of new ? Prob: 0.00451

In [37]: language_model.find_occurrences("city","of","new")
The tri-gram "city of new" was followed by the following words in the training set:
    york (8 times)

In [27]: model.predict_next_word("life","in","the",3)
life in the world Prob: 0.15953
life in the game Prob: 0.07801
life in the united Prob: 0.05246

In [35]: language_model.find_occurrences("life","in","the")
The tri-gram "life in the" was followed by the following words in the training set:
    big (7 times)
    united (2 times)
    department (1 time)
    world (1 time)

In [28]: model.predict_next_word("he","is","the",3)
he is the best Prob: 0.26497
he is the same Prob: 0.13878
he is the only Prob: 0.10073

In [38]: language_model.find_occurrences("he","is","the")
The tri-gram "he is the" was followed by the following words in the training set:
    one (4 times)
    only (4 times)
    president (4 times)
    man (4 times)
    best (2 times)
    city (1 time)
    group (1 time)
    government (1 time)
    first (1 time)
    second (1 time)
    same (1 time)
    director (1 time)

2.After observe the plot, I figure out that the words with same part-of-speech tend to stay in same cluster.

3. No, actually the distance between “new” and “york” is big. The reason for this is perhaps because they don’t have the same part-of-speech. “new” is a adj, and “york” tends to be a noun.

In [29]: model.word_distance("new","york")
Out[29]: 3.9224628551052767

In [39]: model.display_nearest_words("new")
big: 2.92249945907
old: 2.94013049169
white: 2.94443995216

In [40]: model.display_nearest_words("york")
national: 1.02674551089
states: 1.04705511199
ms.: 1.05459599604

4. The distance between("government","university") is shorter than the distance between ("government","political"), the reason for this should be "government" and "university" are both nouns, and "government" and "political" have different part-of-speech.

In [30]: model.word_distance("government","political")
Out[30]: 1.4771193581714717

In [31]: model.word_distance("government","university")
Out[31]: 0.98818426333020293