1 Part 1: Linear Embedding – GLoVE

1.1 Given the vocabulary size V and embedding dimensionality d, how many trainable parameters does the GLoVE model have?

There is a total of $Vd + V$ parameters.

1.2 Write the gradient of the loss function with respect to one parameter vector wi.

$$y = \frac{\mathrm{d}L}{\mathrm{d}w_k} = 2\sum_{i \neq k} w_i(w_i^T w_k + b_i + b_k - logX_{ik}) + 2\sum_{j \neq k} w_j(w_k^T w_j + b_j + b_k - logX_{jk}) + 4w_k(w_k^T w_k + 2b_k - logX_{kk})$$

1.3 Implement the gradient update of GLoVE in language model.ipynb.

1.4 Which d leads to optimal validation perfor- mance? Why does / doesn't larger d always lead to better validation error?

The optimal is given by d = 10. There will be overfitting problem when d gets too large.

2 Network architecture

2.1 What is the total number of trainable parameters in the model? Which part of the model has the largest number of trainable parameters?

Input to Embedding Weight: 250*18 = 4500
Embedding to Hidden Weight: 128*3*18 = 6912
Hidden bias: 128
Hidden to Output Weight: 128*250 = 32000
Output bias: 250
The total number of parameter is 43790
The output layer has the largest number of trainable parameters.

2.2 n-grams: If we stored all the counts explicitly, how many entries would this table have?

$250^4 = 3,906,250,000$

3 Third Exercise

As in the notebook. The printed result is as the following graph.

```
#check_gradients()
print_gradients()
```

```
loss_derivative[2, 5] 0.001112231773782498
loss_derivative[2, 121] -0.9991004720395987
loss_derivative[5, 33] 0.0001903237803173703
loss_derivative[5, 31] -0.7999757709589483

param_gradient.word_embedding_weights[27, 2] -0.2719953998
param_gradient.word_embedding_weights[43, 3] 0.86417222673
param_gradient.word_embedding_weights[22, 4] -0.2546730202
param_gradient.word_embedding_weights[2, 5] 0.0

param_gradient.embed_to_hid_weights[10, 2] -0.652699031391
param_gradient.embed_to_hid_weights[15, 3] -0.131064330004
param_gradient.embed_to_hid_weights[30, 9] 0.1184677461816
param_gradient.embed_to_hid_weights[35, 21] -0.10004526104

param_gradient.hid_bias[10] 0.2537663873815642
param_gradient.hid_bias[20] -0.0332673916363535 7

param_gradient.output_bias[0] -2.0627596032173052
param_gradient.output_bias[1] 0.0390200857392169
param_gradient.output_bias[2] -0.7561537928318482
param_gradient.output_bias[3] 0.2123517205112363 5
```

Abbildung 1: Part 3 Fig

4 Forth Exercise

4.1 Use the model to predict the next word. Does the model give sensible predictions? Try to find an example where it makes a plausible prediction even though the 4-gram wasn't present in the dataset

The model gives the next word "yorkäfter "city of new"with Prob: 0.97". the model gives sensible predictions. 'her','home', 'is' are followed by "not"with Prob: 0.18.

4.2 What do the words in each cluster have in common? How do the t-SNE embeddings for both models compare? How does this compare to the t-SNE embeddings?

1. In "tsne_plot_representation", Words with similar functions grammatically are group together, like verytoo, thesethose on the top left corner.
2. Compared with the first graph, tsne glove representation puts words with similar type close, like twothreefour, peoplechildren etc.
plot_2d_GLoVE_representation tends to put words that appears together often closer.

4.3 Are the words ‘new’ and ‘york’ close together in the learned representation? Why or why not?

No. They have a distance of 3.5, which is not close. This is because this model pays attention to the grammatical function for the word.

4.4 Which pair of words is closer together in the learned representation: (‘government’, ‘political’), or (‘government’, ‘university’)? Why do you think this is?

Distance between ‘government’, ‘political’ is 1.55. Distance between ‘government’, ‘university’ is 1.17. As discussed before, this model pays attention to the grammatical function. In this case the latter one has both words as nonu, thus have a lower distance.

+ Code + Text

[22]
```
trained_model.predict_next_word("city","of","new")
```

```
city of new york Prob: 0.97109
city of new . Prob: 0.01537
city of new , Prob: 0.00183
city of new life Prob: 0.00097
city of new ? Prob: 0.00095
city of new home Prob: 0.00086
city of new business Prob: 0.00070
city of new world Prob: 0.00051
city of new season Prob: 0.00041
city of new family Prob: 0.00039
```

```
find_occurrences("her","home","is")
trained_model.predict_next_word("her","home","is")
```

```
The tri-gram "her home is" did not occur in the training set.
her home is not Prob: 0.17877
her home is . Prob: 0.09093
her home is nt Prob: 0.07028
her home is over Prob: 0.04218
her home is the Prob: 0.03287
her home is just Prob: 0.03132
her home is still Prob: 0.02931
her home is good Prob: 0.02851
her home is all Prob: 0.02742
her home is about Prob: 0.02518
```

[24]
```
trained_model.word_distance("new","york")
```

```
3.5030235071123608
```

[25]
```
trained_model.word_distance("government","political")
```

```
1.557156015209253
```

[26]
```
trained_model.word_distance("government","university")
```

```
1.1773016773151586
```