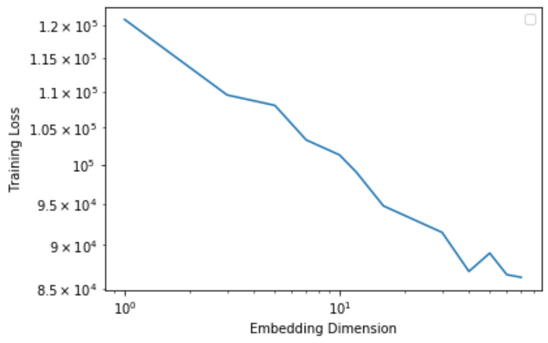
Vincent Fan

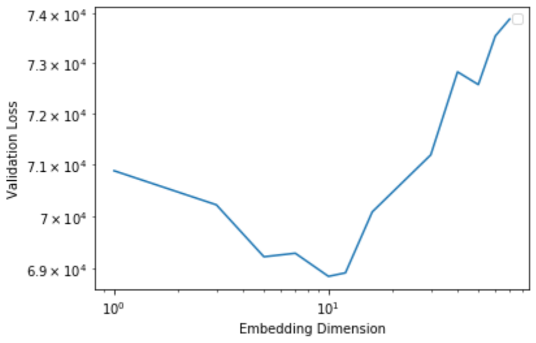
Part 1

1. GLoVE has one weight matrix and one bias, which adds up to:
$$V \times d + V \times 1 = V(d+1)$$
2. $\frac{dL}{dw_i} = 2\sum_{j=1}^{V}(w_i^{\mathrm{T}}w_j + b_i + b_j - \log X_{ij})w_j$
3. See below snippet

```
def grad_GLoVE(W,  b, log_co_occurence):
  "Return the gradient of GLoVE objective w.r.t W and b."
  "INPUT: W - Vxd; b - Vx1; log_co_occurence: VxV"
  "OUTPUT: grad_W - Vxd; grad_b - Vx1"
  n,_ = log_co_occurence.shape
  grad_W = 2*(W @ W.T + b + b.T - log_co_occurence) @ W
  grad_b = 2*(W @ W.T + b + b.T - log_co_occurence) @ np.ones([n,1])
  return grad_W, grad_b
```

4. After couple tries, I found that having $d = 10$ would lead to the most optimal validation performance on average. Larger d does not always lead to better validation error because having larger d means the number of trainable parameters will increase. Therefore, there is a higher chance of the model getting overfitted with the training data. In fact, we can validate this by looking at the training loss graph, which shows higher d will have less training loss. However, the validation loss is much higher. This concludes that the model indeed overfitted and thus cannot generalize to the validation data.





Part 2

1. The total number of trainable parameters is: $W^{(0)}$ (3 x 16) + $W^{(1)}$ (48 x 128) + $b^{(1)}$ (1 x 128) + $W^{(2)}$ (128 x 250) + $b^{(2)}$ (1 x 250) = 38570 trainable parameters in the model. $W^{(2)}$, `hid_to_output_weights` has the largest number of trainable parameters.
2. If we store all the counts explicitly where the order of the 4-grams matters, then the total number of entries would be 250^4 since each of the 4 words are equally likely to be extracted from the dictionary where duplicates are allowed.

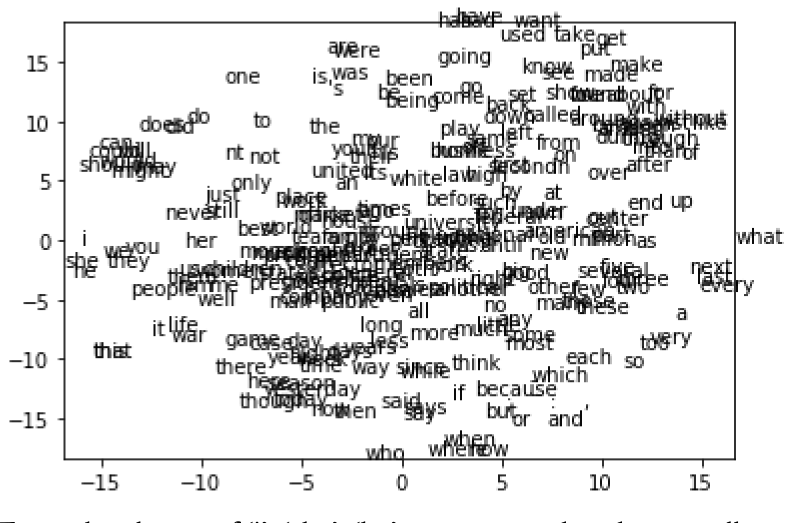
Part 3

Output from `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] -0.27199539981936866
param_gradient.word_embedding_weights[43, 3] 0.8641722267354154
param_gradient.word_embedding_weights[22, 4] -0.2546730202374648
param_gradient.word_embedding_weights[2, 5] 0.0

param_gradient.embed_to_hid_weights[10, 2] -0.6526990313918257
param_gradient.embed_to_hid_weights[15, 3] -0.13106433000472612
param_gradient.embed_to_hid_weights[30, 9] 0.11846774618169399
param_gradient.embed_to_hid_weights[35, 21] -0.10004526104604386

param_gradient.hid_bias[10] 0.25376638738156426
param_gradient.hid_bias[20] -0.03326739163635369

param_gradient.output_bias[0] -2.062759603217304
param_gradient.output_bias[1] 0.03902008573921689
param_gradient.output_bias[2] -0.7561537928318482
param_gradient.output_bias[3] 0.21235172051123635
```

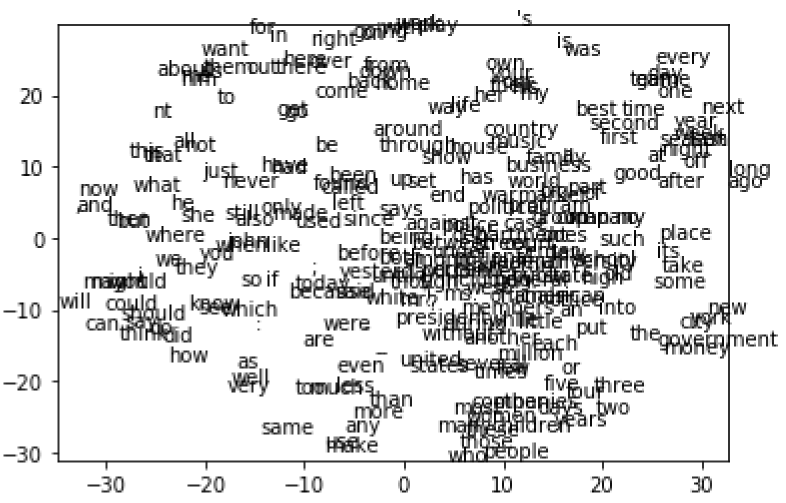
Part 4

1. Picking ‘he’, ‘is’, ‘the’ gives the following top 3 choices: ‘best’, ‘first’, and ‘same’ which all make sense. I also tried ‘as’, ‘if’, ‘he’ which using `find_occurrences()`, there is only 1 occurrence of ‘were’. However, the model outputs ‘ ’s’, ‘did’, and ‘is’ which all make a lot of sense as well.
2. From `tsne_plot_representation()`:



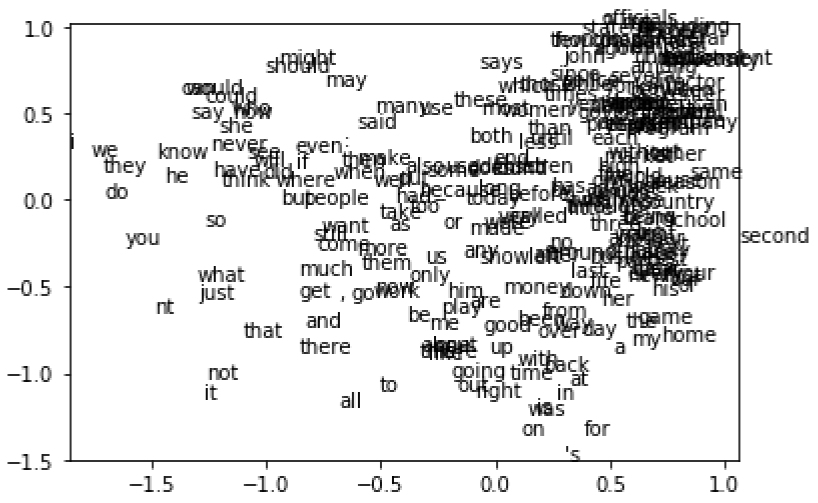
From the cluster of ‘i’, ‘she’, ‘he’, we can see that they are all pronouns. Also, notice the cluster of ‘said’, ‘say’, ‘says’, they are different tenses of the same word. Moreover, we can look at ‘are’, ‘were’, ‘was’, ‘is’, ‘ ‘s’, they are all different form of ‘be’. Therefore, this embedding groups words with similar grammatical meanings.

From `tsne_plot_GLoVE_representation()`:



Although this embedding also puts words with similar meaning together, the words are not as clustered as the previous one since the words are more evenly spread out. To conclude, this embedding groups words that are often used together.

From `plot_2d_GLoVE_representation()`:



We can see that this embedding is very different from the t-SNE embeddings as it only forms a couple huge clusters of words. On the other hand, the most common words like ‘you’, ‘we’, ‘so’ are all evenly spread out. Therefore, this embedding is different from the previous embeddings where it tries to keep equal distance from the common words and unique words which are the mega clusters.

3. After using `display_nearest_words()` on ‘new’, the top 10 words did not include ‘york’. Moreover, all of these words’ distance were closer than the distance between ‘new’ and ‘york’. This is be explained because ‘new’ by itself is an adjective whereas the ‘new’ in ‘New York’ is part of a noun. Also, if we investigate the top 10 choices, ‘new’ is perceived as an adjective.
4. After using `word_distance()` on both pairs, we can see that ‘government’ is closer to ‘university’ than ‘political’. This can be explained with similar reasons as above, which is because both ‘government’ and ‘university’ are nouns, and both often announce new policy/rules. On the other hand, ‘political’ is an adjective that relates to politics which might not necessarily connect to government directly, i.e. “we can have political science at our university”.