CS310 Natural Language Processing - Assignment 2 Word2vec Implementation

12310520 芮煜涵

Requirement3

Printing Interval

In the train function, I set it to print the average loss after every epoch with

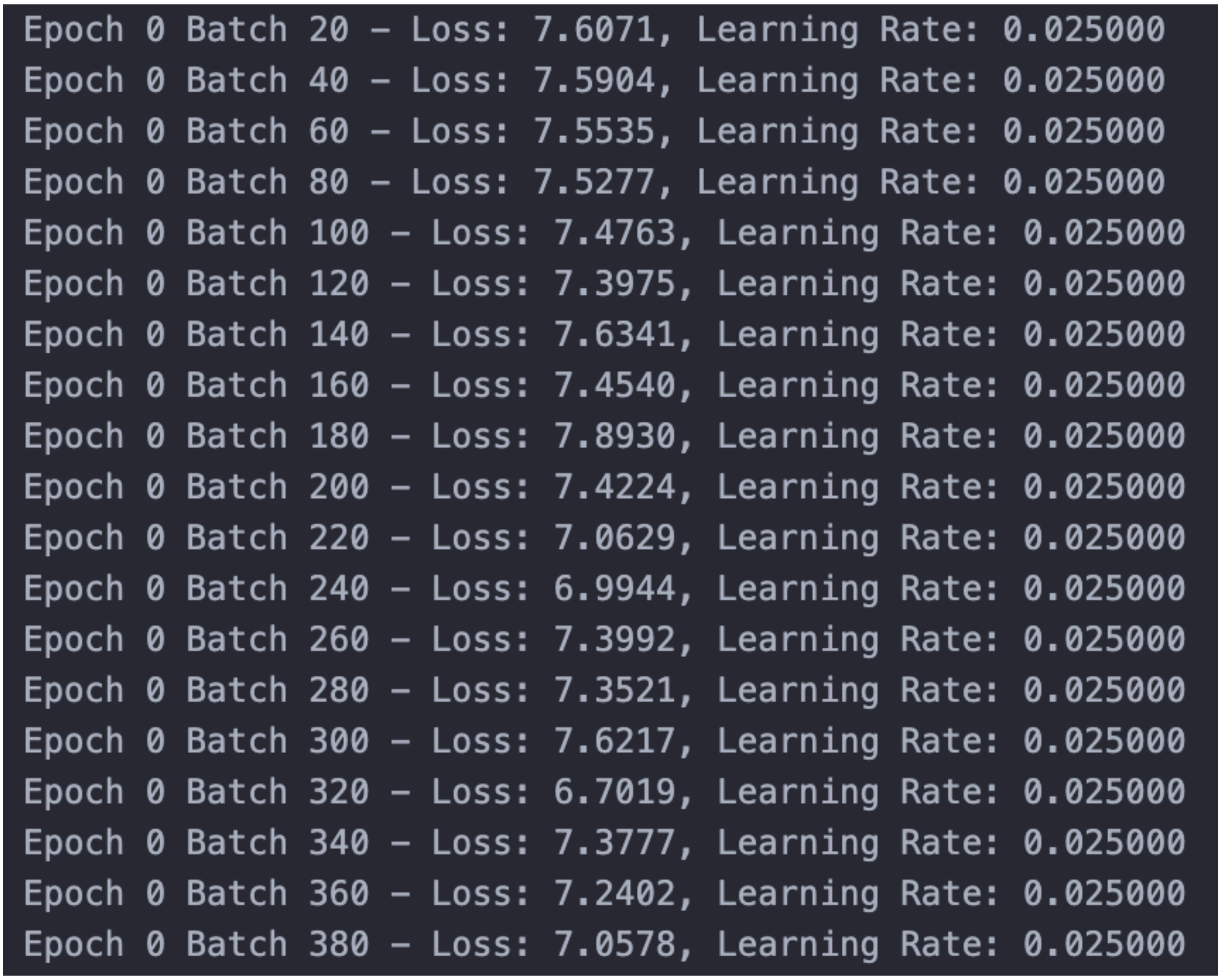
```
print(f"Epoch {epoch} - Average Loss: {total_loss/(num_batches):.4f}")
```

I chose to print per epoch instead of every 20 batches,

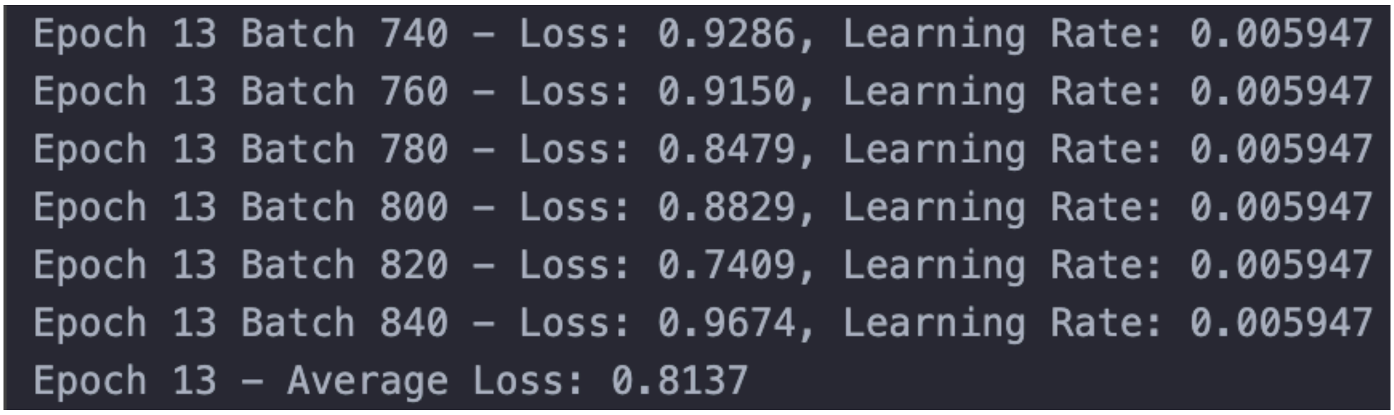
```
if num_batches % 20 == 0:
                lr = scheduler.get_last_lr()[0]
                print(f"Epoch {epoch} Batch {num_batches} - Loss: {loss.item():.4f}, 
Learning Rate: {lr:.6f}")
```

This print interval was determined based on the dataset size (lunyu_20chapters.txt) and batch size (batch_size = 16). Every 20 batches corresponds to approximately 320 samples (20 × 16), providing a good balance between observing the loss trend early in training and avoiding excessive output.

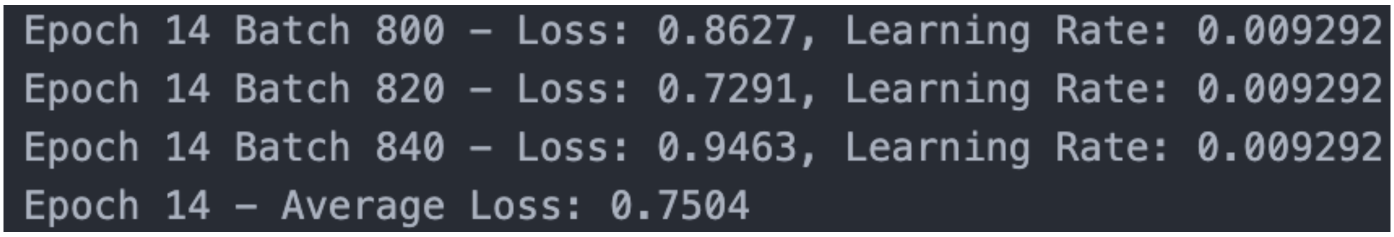
Screenshots of loss change:



```
Epoch 0 Batch 20 - Loss: 7.6071, Learning Rate: 0.025000
Epoch 0 Batch 40 - Loss: 7.5904, Learning Rate: 0.025000
Epoch 0 Batch 60 - Loss: 7.5535, Learning Rate: 0.025000
Epoch 0 Batch 80 - Loss: 7.5277, Learning Rate: 0.025000
Epoch 0 Batch 100 - Loss: 7.4763, Learning Rate: 0.025000
Epoch 0 Batch 120 - Loss: 7.3975, Learning Rate: 0.025000
Epoch 0 Batch 140 - Loss: 7.6341, Learning Rate: 0.025000
Epoch 0 Batch 160 - Loss: 7.4540, Learning Rate: 0.025000
Epoch 0 Batch 180 - Loss: 7.8930, Learning Rate: 0.025000
Epoch 0 Batch 200 - Loss: 7.4224, Learning Rate: 0.025000
Epoch 0 Batch 220 - Loss: 7.0629, Learning Rate: 0.025000
Epoch 0 Batch 240 - Loss: 6.9944, Learning Rate: 0.025000
Epoch 0 Batch 260 - Loss: 7.3992, Learning Rate: 0.025000
Epoch 0 Batch 280 - Loss: 7.3521, Learning Rate: 0.025000
Epoch 0 Batch 300 - Loss: 7.6217, Learning Rate: 0.025000
Epoch 0 Batch 320 - Loss: 6.7019, Learning Rate: 0.025000
Epoch 0 Batch 340 - Loss: 7.3777, Learning Rate: 0.025000
Epoch 0 Batch 360 - Loss: 7.2402, Learning Rate: 0.025000
Epoch 0 Batch 380 - Loss: 7.0578, Learning Rate: 0.025000
```



```
Epoch 13 Batch 740 - Loss: 0.9286, Learning Rate: 0.005947
Epoch 13 Batch 760 - Loss: 0.9150, Learning Rate: 0.005947
Epoch 13 Batch 780 - Loss: 0.8479, Learning Rate: 0.005947
Epoch 13 Batch 800 - Loss: 0.8829, Learning Rate: 0.005947
Epoch 13 Batch 820 - Loss: 0.7409, Learning Rate: 0.005947
Epoch 13 Batch 840 - Loss: 0.9674, Learning Rate: 0.005947
Epoch 13 - Average Loss: 0.8137
```



```
Epoch 14 Batch 800 - Loss: 0.8627, Learning Rate: 0.009292
Epoch 14 Batch 820 - Loss: 0.7291, Learning Rate: 0.009292
Epoch 14 Batch 840 - Loss: 0.9463, Learning Rate: 0.009292
Epoch 14 - Average Loss: 0.7504
```

Determining the Required Training Epochs

I determine the number of epochs by monitoring the average loss per epoch (total_loss / num_batches). When the loss stops decreasing significantly (e.g., by less than 0.05) over several epochs, it suggests the model has converged.

In my code, I **set epochs = 15** and used the CosineAnnealingLR scheduler to adjust the learning rate dynamically. By observing the output, I can identify when the loss stabilizes.

If the loss decrease over 2-3 consecutive epochs is **less than 0.05**, I consider training complete. In this case, the loss stabilizes by epoch 14(the loss of epoch 13 is 0.8137, the loss of epoch 14 is 0.7504, the loss decrease is quite near 0.05), making 15 epochs a reasonable choice.

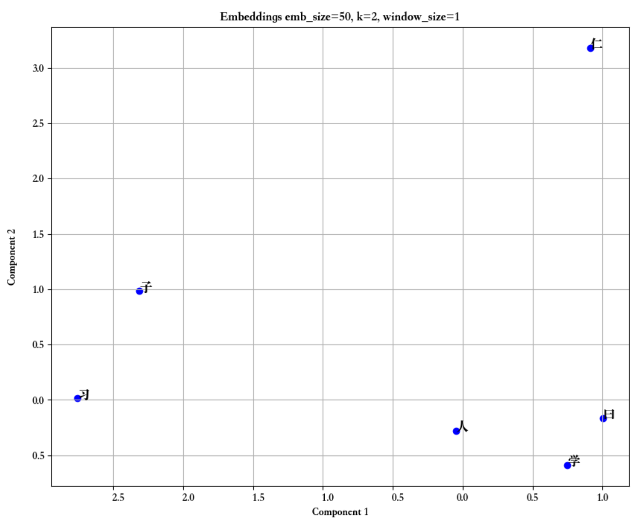
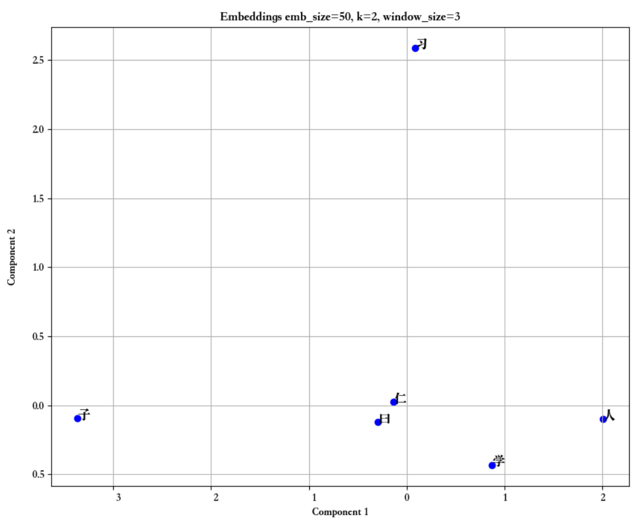
Requirement4

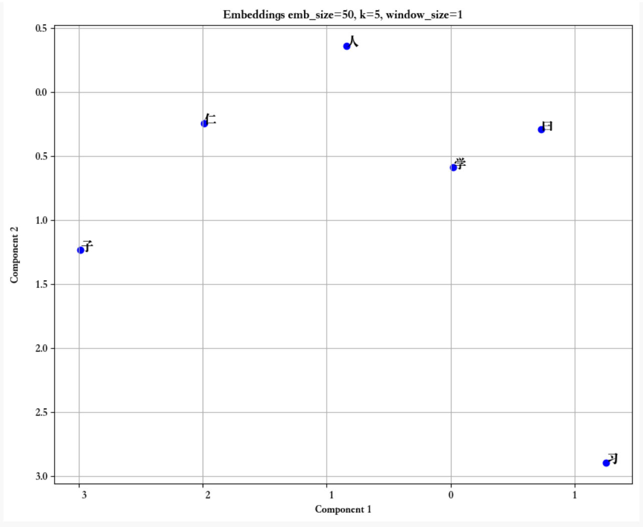
`emb_size`	`k`	`window_size`	`time`
50	2	1	2.48s
50	2	3	7.22s
50	5	1	2.52s
50	5	3	7.56s
100	2	1	2.49s
100	2	3	7.49s
100	5	1	2.41s
100	5	3	7.23s

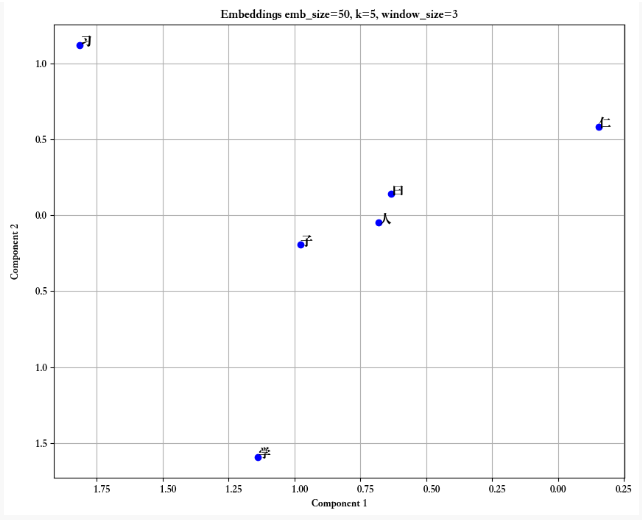
The dominant factor of running time is `window_size`.

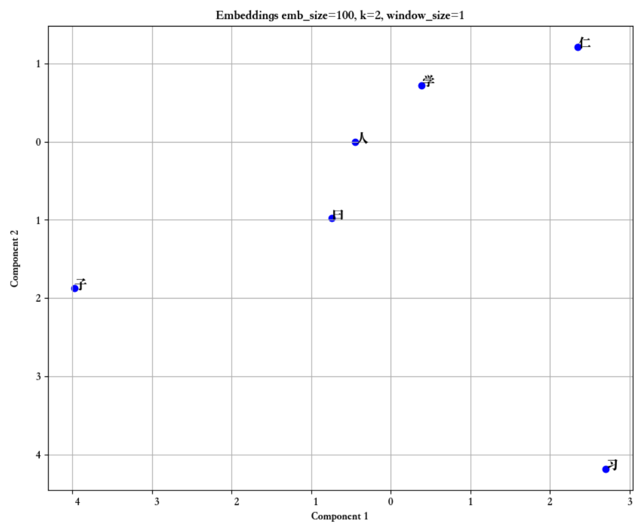
Requirement5

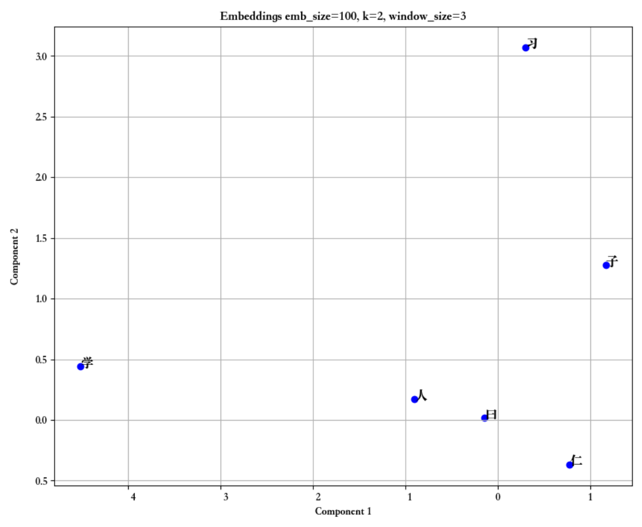
8 embeddings

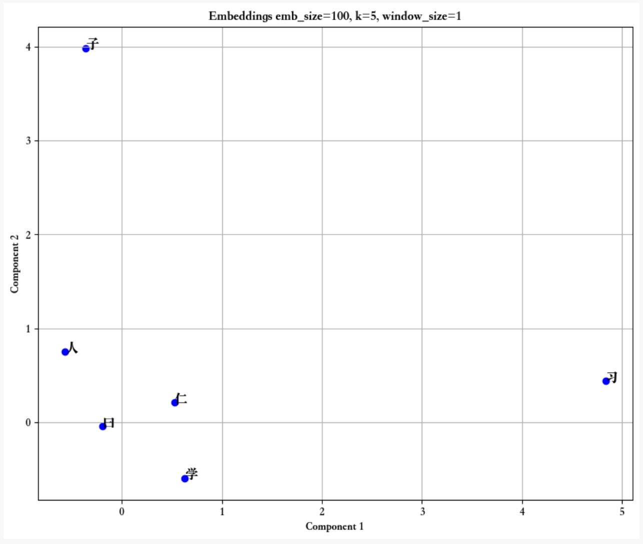
	

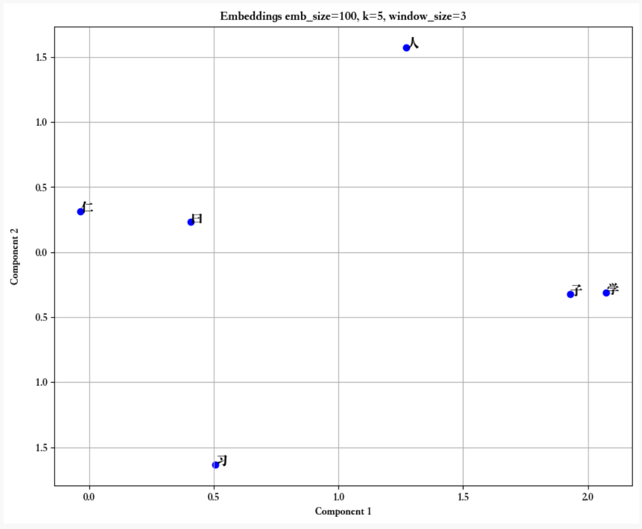




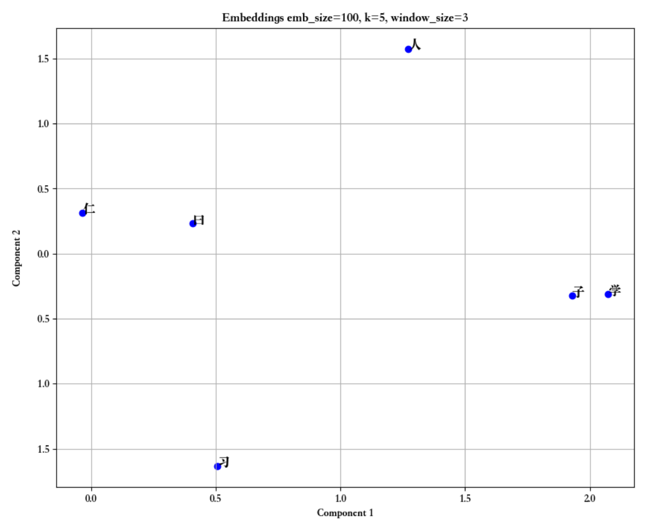


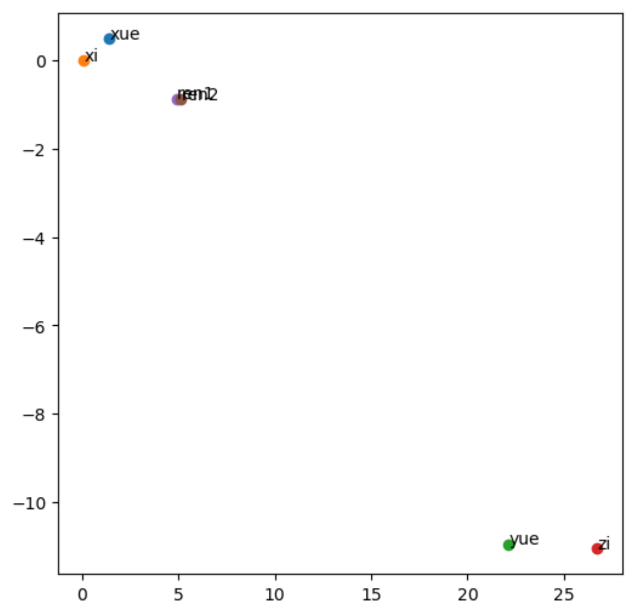






Difference from that in Lab4





In the A2 task, the trained word embeddings exhibit a scattered distribution. After applying Truncated SVD, the two components of '子' (zǐ) and '学' (xǔe) are relatively close in distance, but in the context of the Chinese language, their meanings differ significantly.

In contrast, in the embeddings from Lab 4, '学' (xǔe) and '习' (xí), as well as '人' (rén) and '仁' (rén), are positioned close to each other, which aligns well with their semantic similarity in the Chinese context, indicating that this embedding performs effectively.