CS310 - Natural Language Processing
Assignment 2 Report

Name: Tan Hao Yang SID: 12212027

Introduction

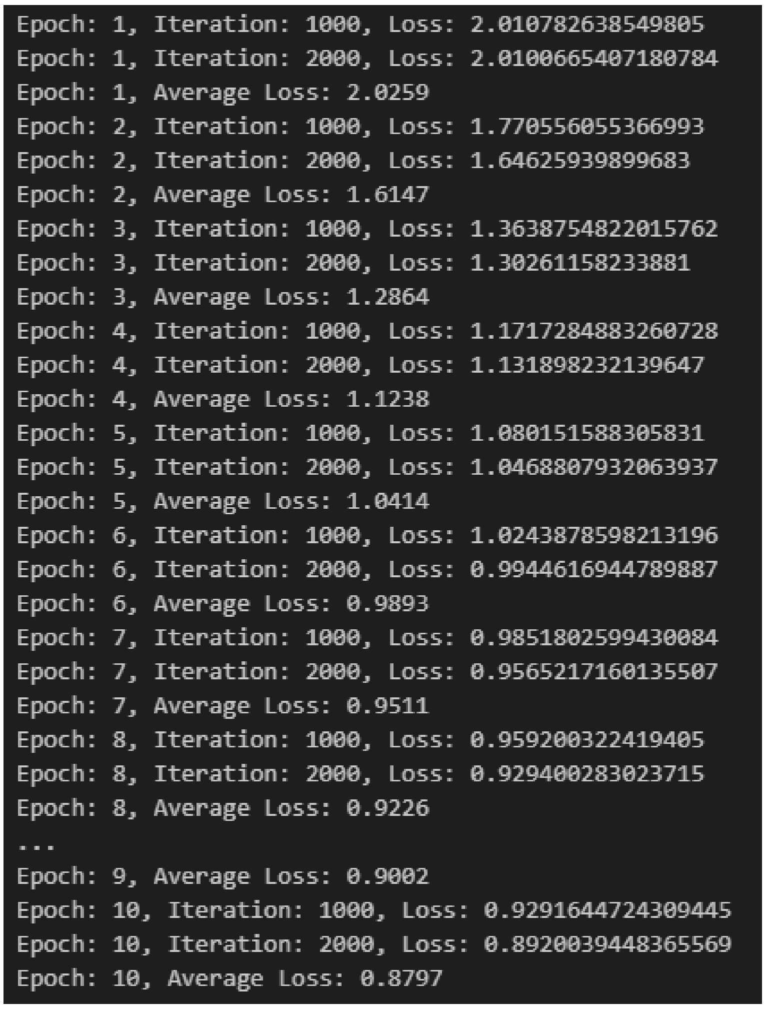
This assignment focuses on implementing and training a Word2Vec model using the Skip-Gram architecture with negative sampling on the text of《论语》 (The Analects of Confucius). The task involves preprocessing the corpus, implementing the Skip-Gram model, and training it with various hyperparameters, including embedding size, negative sample count, and context window size. The training process is monitored through loss trends, and the learned embeddings are visualized using Truncated SVD for comparison. The goal is to explore how different hyperparameters impact embedding quality and to analyze the semantic relationships captured by the model.

Results and Discussions

1. Training Epochs

I determined the optimal number of training epochs by experimenting with different values: 5, 10, 15, and 20. The corresponding losses were 0.9886, 0.8797, 0.8610, and 0.8439, respectively. Although 10 epochs did not yield the lowest loss, the loss value was acceptable, and the computational time was significantly lower compared to 15 and 20 epochs. Therefore, I selected 10 epochs as the optimal choice. Below is a screenshot showing the loss changes, with the loss printed every 1000 intervals.

Figure 1: Screenshot of loss changes.



```
Epoch: 1, Iteration: 1000, Loss: 2.010782638549805
Epoch: 1, Iteration: 2000, Loss: 2.0100665407180784
Epoch: 1, Average Loss: 2.0259
Epoch: 2, Iteration: 1000, Loss: 1.770556055366993
Epoch: 2, Iteration: 2000, Loss: 1.64625939899683
Epoch: 2, Average Loss: 1.6147
Epoch: 3, Iteration: 1000, Loss: 1.3638754822015762
Epoch: 3, Iteration: 2000, Loss: 1.30261158233881
Epoch: 3, Average Loss: 1.2864
Epoch: 4, Iteration: 1000, Loss: 1.1717284883260728
Epoch: 4, Iteration: 2000, Loss: 1.131898232139647
Epoch: 4, Average Loss: 1.1238
Epoch: 5, Iteration: 1000, Loss: 1.080151588305831
Epoch: 5, Iteration: 2000, Loss: 1.0468807932063937
Epoch: 5, Average Loss: 1.0414
Epoch: 6, Iteration: 1000, Loss: 1.0243878598213196
Epoch: 6, Iteration: 2000, Loss: 0.9944616944789887
Epoch: 6, Average Loss: 0.9893
Epoch: 7, Iteration: 1000, Loss: 0.9851802599430084
Epoch: 7, Iteration: 2000, Loss: 0.9565217160135507
Epoch: 7, Average Loss: 0.9511
Epoch: 8, Iteration: 1000, Loss: 0.959200322419405
Epoch: 8, Iteration: 2000, Loss: 0.929400283023715
Epoch: 8, Average Loss: 0.9226
...
Epoch: 9, Average Loss: 0.9002
Epoch: 10, Iteration: 1000, Loss: 0.9291644724309445
Epoch: 10, Iteration: 2000, Loss: 0.8920039448365569
Epoch: 10, Average Loss: 0.8797
```

2. Trying out different hyperparameters

In this section, we have tried out the skip gram model with different hyperparameters. Based on the results from the experiments with different hyperparameters, the combination of embedding size = 50, negative sample number k = 2, and window size = 1 appears to be the best configuration. This configuration achieved the lowest loss of 0.8785, which is significantly better than most other combinations, while also maintaining a relatively low computational time of 34.44 seconds. Although the embedding size of 100 with k = 2 and window size = 1 yielded a slightly lower loss of 0.8798, the improvement is marginal compared to the increased computational time of 39.63 seconds. Additionally, increasing the window size or the number of negative samples generally led to higher losses and significantly longer training times, suggesting that smaller window sizes and fewer negative samples are more effective for this dataset. Therefore, the combination of emb_size = 50, k = 2, and window_size = 1 strikes the best balance between model performance and computational efficiency.

Table 1: Loss for Each Embedding Configuration

Embedding File	Loss
embeddings_emb50_k2_win1.txt	0.8785
embeddings_emb50_k2_win3.txt	1.3515
embeddings_emb50_k5_win1.txt	1.4490
embeddings_emb50_k5_win3.txt	2.0554
embeddings_emb100_k2_win1.txt	0.8798
embeddings_emb100_k2_win3.txt	1.2845
embeddings_emb100_k5_win1.txt	1.4620
embeddings_emb100_k5_win3.txt	1.9823

Table 2: Computational Time for Each Embedding Configuration

Embedding File	Time (seconds)
embeddings_emb50_k2_win1.txt	34.44
embeddings_emb50_k2_win3.txt	108.32
embeddings_emb50_k5_win1.txt	40.39
embeddings_emb50_k5_win3.txt	112.10
embeddings_emb100_k2_win1.txt	39.63
embeddings_emb100_k2_win3.txt	106.00
embeddings_emb100_k5_win1.txt	39.28
embeddings_emb100_k5_win3.txt	110.82

3. Plot and Compare

In this section, I have included the plots of the 8 embedding results below. Among all of the embedding results, the one with the combination of embedding size = 50, negative sample number k = 2, and window size = 1 have been chosen to be compared with the plot from Lab 4 (LSA). The main observation is that ‘学’ and ‘习’ are very far apart from each other in Skip-Gram model but very close to each other in LSA model.

Figure 2: Plots of 8 embedding results.

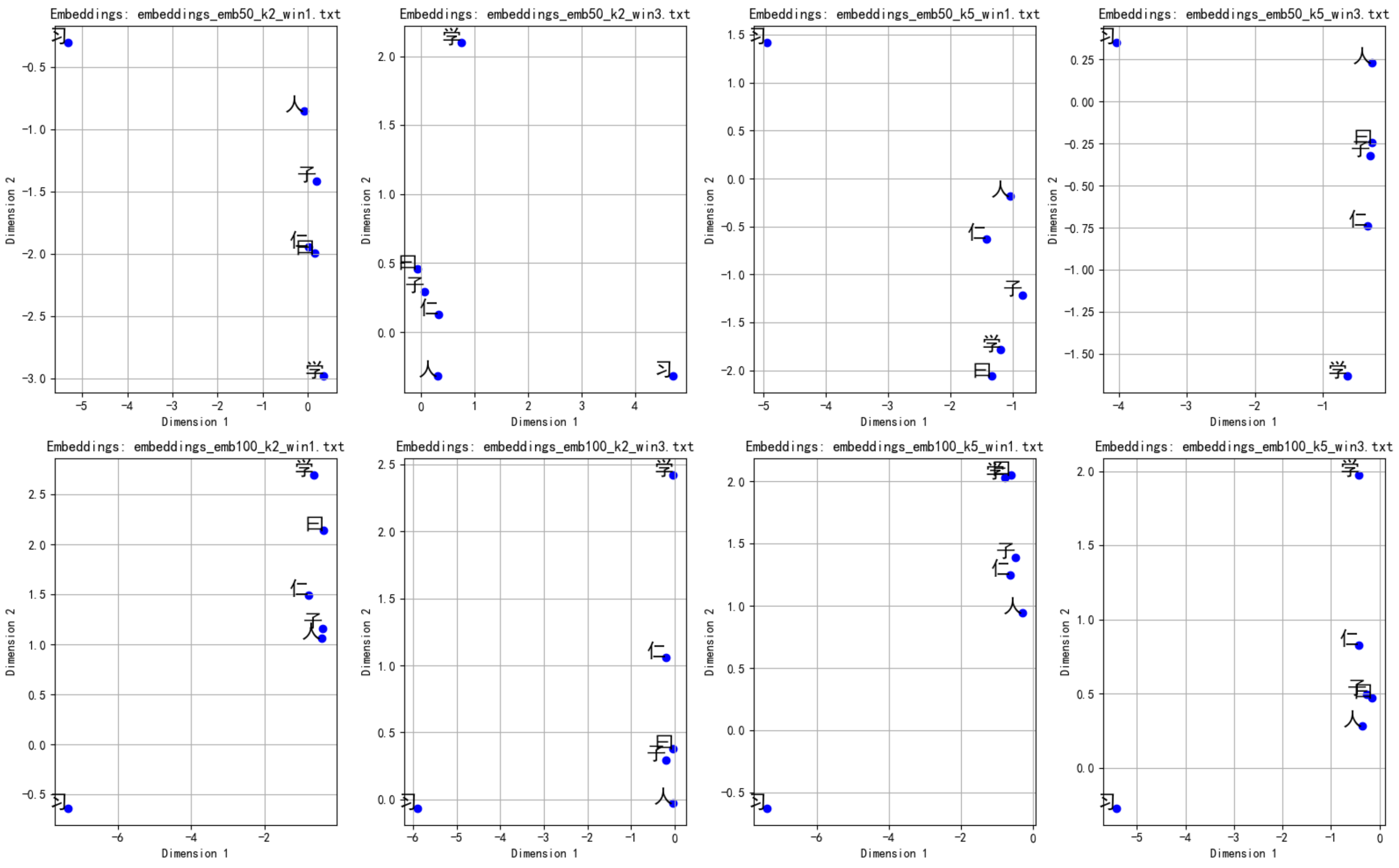
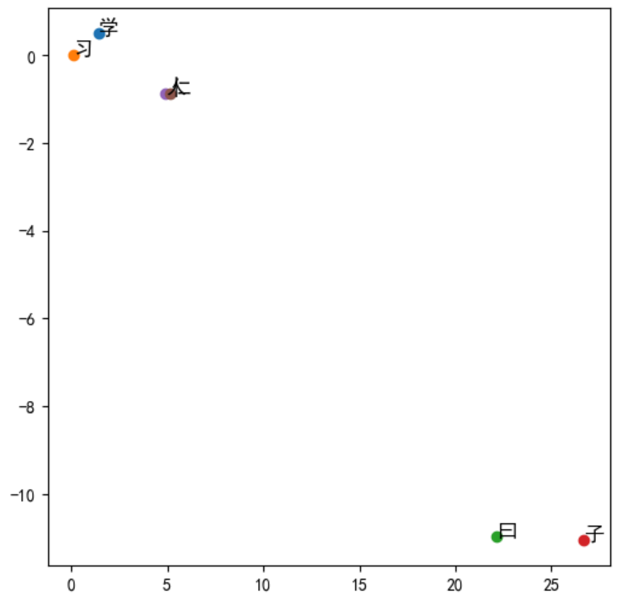


Figure 3: Plot from Lab4 (LSA)



Conclusion

In this assignment, we implemented and trained a Word2Vec model using the Skip-Gram architecture with negative sampling on the text of《论语》. Through experimentation, we determined that 10 training epochs provided an optimal balance between loss reduction and computational efficiency. Additionally, we explored various hyperparameters and found that the combination of embedding size = 50, negative sample number k = 2, and window size = 1 yielded the best performance, achieving the lowest loss of 0.8785 with a reasonable computational time of 34.44 seconds. Visualization of the embeddings revealed notable differences compared to the LSA model, particularly in the semantic relationships between words like ‘学’ and ‘习’. These results highlight the importance of hyperparameter tuning in optimizing model performance and computational efficiency. Overall, the Skip-Gram model demonstrated its effectiveness in capturing semantic relationships, providing valuable insights into the text's structure and meaning.