1 运行结果及简述

```
→  NLPhw-1213 git:(main) ✗ python w2v.py
All libraries are satisfied.
['it', 's', 'useful']
50
(['want', 'to', 'go'], 'i')
(['i', 'to', 'go', 'home'], 'want')
(['i', 'want', 'go', 'home'], 'to')
(['i', 'want', 'to', 'home'], 'go')
总Token数: 50
词表大小: 21
[('<unk>', 1), ('want', 6), ('to', 6), ('go', 4), ('i', 3), ('home', 3), ('play', 3), ('like', 3), ('eating', 3), ('he', 3), ('she', 3), ('it'
, 2), ('is', 2), ('we', 2), ('useful', 1), ('awful', 1), ('can', 1), ('read', 1), ('books', 1), ('will', 1), ('now', 1)]
[0. 1. 0. 0.]
[7.80134161e-05 2.12062451e-04 5.76445508e-04 1.56694135e-03
 4.25938820e-03 1.15782175e-02 3.14728583e-02 8.55520989e-02
 2.32554716e-01 6.32149258e-01]
总Token数: 50
词表大小: 21
Epoch 1: 100%|██████████| 50/50 [00:00<00:00, 3781.65it/s, Avg. loss=2.89]
Epoch 2: 100%|██████████| 50/50 [00:00<00:00, 6263.15it/s, Avg. loss=1.54]
Epoch 3: 100%|██████████| 50/50 [00:00<00:00, 6290.01it/s, Avg. loss=1.05]
Epoch 4: 100%|██████████| 50/50 [00:00<00:00, 7344.00it/s, Avg. loss=0.82]
Epoch 5: 100%|██████████| 50/50 [00:00<00:00, 3246.92it/s, Avg. loss=0.76]
Epoch 6: 100%|██████████| 50/50 [00:00<00:00, 3309.90it/s, Avg. loss=0.67]
Epoch 7: 100%|██████████| 50/50 [00:00<00:00, 9243.85it/s, Avg. loss=0.53]
Epoch 8: 100%|██████████| 50/50 [00:00<00:00, 8460.01it/s, Avg. loss=0.54]
Epoch 9: 100%|██████████| 50/50 [00:00<00:00, 6405.47it/s, Avg. loss=0.52]
Epoch 10: 100%|██████████| 50/50 [00:00<00:00, 8745.79it/s, Avg. loss=0.50]
总耗时 0.17s
[('i', 1.0), ('he', 0.9925540605382073), ('she', 0.9663378567626821), ('<unk>', 0.6356992702314611), ('is', 0.39741235376377415)]
[('he', 0.9999999999999999), ('i', 0.9925540605382073), ('she', 0.98580084006031), ('<unk>', 0.6171017925293522), ('is', 0.35823278721958063)]
[('she', 1.0), ('he', 0.98580084006031), ('i', 0.9663378567626821), ('<unk>', 0.5012279262065747), ('is', 0.38698246680231224)]
总Token数: 205068
词表大小: 4000
Epoch 1: 100%|██████████| 205058/205058 [05:09<00:00, 662.53it/s, Avg. loss=6.02]
Recall rate: 7.69%
Epoch 2: 100%|██████████| 205058/205058 [05:07<00:00, 666.59it/s, Avg. loss=5.63]
Recall rate: 12.43%
Epoch 3: 100%|██████████| 205058/205058 [05:06<00:00, 669.64it/s, Avg. loss=5.48]
Recall rate: 13.61%
Epoch 4: 100%|██████████| 205058/205058 [05:05<00:00, 670.94it/s, Avg. loss=5.38]
Recall rate: 15.68%
Epoch 5: 100%|██████████| 205058/205058 [04:58<00:00, 685.86it/s, Avg. loss=5.30]
Recall rate: 16.86%
Epoch 6: 100%|██████████| 205058/205058 [05:13<00:00, 654.35it/s, Avg. loss=5.24]
Recall rate: 17.75%
Epoch 7: 100%|██████████| 205058/205058 [05:11<00:00, 658.52it/s, Avg. loss=5.19]
Recall rate: 18.64%
Epoch 8: 100%|██████████| 205058/205058 [05:14<00:00, 652.61it/s, Avg. loss=5.15]
Recall rate: 18.93%
Epoch 9: 100%|██████████| 205058/205058 [05:05<00:00, 671.00it/s, Avg. loss=5.11]
Recall rate: 18.93%
Epoch 10: 100%|██████████| 205058/205058 [05:03<00:00, 674.73it/s, Avg. loss=5.08]
Recall rate: 18.64%
```

图 1: 全部运行结果

以上1是本次作业的全部运行结果。代码使用了助教提供的默认学习率，即 test1 中的学习率为 1.0，test2 中的学习率为 0.1。可以看到，test1 中最终一个 epoch 的平均 loss 确实约为 0.5，并且“i”、“he”和“she”的相似性较高，表现为三者的数值均接近 1，而前五个最相似的词中，除了“i”、“he”和“she”外，其他词汇的相似性均较低，不超过 0.6. test2 中，最后一个 epoch 平均 loss 确实降至 5.1 左右，并且在同义词上的召回率约为 19%，和助教给出的参考数据比较接近。

2 代码及实现思路简述

工具函数的实现比较直接，注意 softmax 数值上溢出即可，此外，为了避免 softmax 数据取对数的下溢出，我还简单实现了 log-softmax。实际上如果训练参数设置合理，正常收敛的情况，不会发生下溢出。

```python
def one_hot(dim: int, idx: int) -> np.ndarray:
    res = np.zeros(dim)
    res[idx] = 1
    return res

def softmax(x: np.ndarray) -> np.ndarray:
    x = x - np.max(x)
    return np.exp(x) / np.sum(np.exp(x))

def log_softmax(x: np.ndarray) -> np.ndarray:
    x = x - np.max(x)
    return x - np.log(np.sum(np.exp(x)))
```

单步训练的实现，根据 ppt 中给出的公式进行更新即可。这里我直接把目标向量在数据词表中出现 index 的位全部置 1，而不是通过循环遍历累加，实现上简洁一些，也能取得同样的训练效果1.

另外值得一提的是这里的 loss 仅仅用于观察训练是否在收敛，loss 数值对参数更新没有指导作用。所以训练过程正常的情况下，对 softmax 取对数并不会出现下溢出，如果数值异常，可能是训练参数设置的问题。

```python
# 构造输入向量和目标向量
x = np.zeros(len(self.vocab))
x[list(map(lambda x: self.vocab.token_to_idx(x), context_tokens))] = 1
xbar = x / C
# 前向步骤
h = self.U.T@xbar
e = softmax(self.V.T@h)
# 计算loss
y = log_softmax(self.V.T@h)
loss = -1 * y[self.vocab.token_to_idx(target_token)]
# 更新参数
e[self.vocab.token_to_idx(target_token)] -= 1
self.U -= learning_rate * (self.V@e.reshape(-1, 1)@xbar.reshape(1, -1)).T
self.V -= learning_rate * (e.reshape(-1, 1)@h.reshape(1, -1)).T
```