深圳大学期末考试试卷

开/闭卷 开卷 A/B 卷 _____

课程编号 1504660001 课序号 01 课程名称 自然语言处理 学分 2.5

命题人(签字)_____ 审题人(签字)_____ _____年____月____日

题号	一	二	三	四	五	六	七	八	九	十	基本题总分	附加题
得分												
评卷人												

座号_____ 学号_____ 姓名_____ 专业_____ 学院_____

(密封线内不答题)

密 封 线

一、随堂测试：词嵌入方法的计算案例 (共 4 题，每道 25 分，共 100 分)

假设我们有三个词：
king
queen
man
写出 python 代码，把这些词表示为向量，以 king - man + woman ≈ queen 为优化方向，构建向量间的语义关系。

1. 将词进行独热编码（One-hot Encoding）表示
king_onehot = np.array([1, 0, 0, 0])
queen_onehot = np.array([0, 1, 0, 0])
man_onehot = np.array([0, 0, 1, 0])
woman_onehot = np.array([0, 0, 0, 1])

2. 初始化参数矩阵
E = [[0.04967142 -0.01382643 0.06476885]
 [0.15230299 -0.02341534 -0.0234137]
 [0.15792128 0.07674347 -0.04694744]
 [0.054256   -0.04634177 -0.04657298]]

此步骤中我的代码使用固定随机种子生成随机参数矩阵：

```
# 2. 初始化参数矩阵
np.random.seed(42)  # 固定随机种子
E = np.random.randn(4, 3) * 0.1  # 随机初始化，4个词×3维
print("初始嵌入矩阵：")
print(E)
```

✓ 0.0s

```
初始嵌入矩阵：
[[ 0.04967142 -0.01382643  0.06476885]
 [ 0.15230299 -0.02341534 -0.0234137 ]
 [ 0.15792128  0.07674347 -0.04694744]
 [ 0.054256   -0.04634177 -0.04657298]]
```

3. 计算词的嵌入。根据上面的独热编码和嵌入矩阵，通过矩阵乘法得到词的嵌入。
king_emb = king_onehot.dot(E)
queen_emb = queen_onehot.dot(E)
man_emb = man_onehot.dot(E)
woman_emb = woman_onehot.dot(E)

king_emb = [0.04967142 -0.01382643 0.06476885]
queen_emb = [0.15230299 -0.02341534 -0.0234137]
man_emb = [0.15792128 0.07674347 -0.04694744]
woman_emb =[0.054256   -0.04634177 -0.04657298]

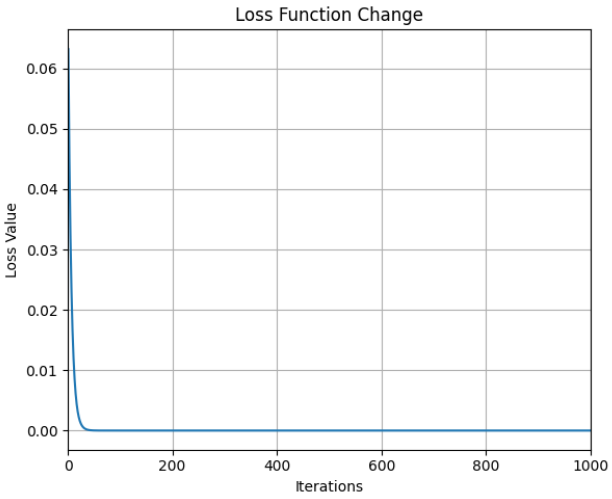
4. 写出损失函数的计算公式
predicted_queen = king_emb - man_emb + woman_emb

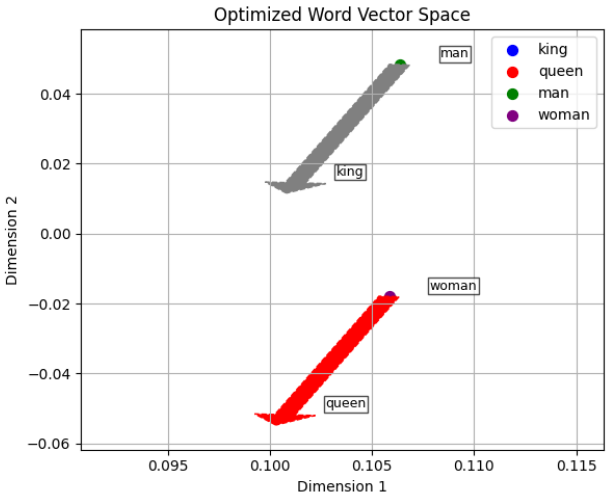
使用梯度下降算法和迭代优化，计算最终的词嵌入和结果:

```
# 梯度下降优化参数设置
learning_rate = 0.01  # 学习率
iterations = 1000     # 迭代次数
loss_history = []     # 记录损失变化

# 计算预测向量和损失
predicted_vec = king_emb - man_emb + woman_emb
diff = predicted_vec - queen_emb
initial_loss = np.sum(diff **2)
loss_history.append(initial_loss)
```

```
# 迭代优化
for i in range(iterations):
    # 计算当前词嵌入向量
    king_emb = king_onehot.dot(E)
    queen_emb = queen_onehot.dot(E)
    man_emb = man_onehot.dot(E)
    woman_emb = woman_onehot.dot(E)

    # 计算预测向量和损失值
    predicted_vec = king_emb - man_emb + woman_emb
    diff = predicted_vec - queen_emb
    loss = np.sum(diff** 2)

    # 计算梯度（对每个词嵌入的偏导数）
    grad_king = 2 * diff
    grad_queen = -2 * diff
    grad_man = -2 * diff
    grad_woman = 2 * diff

    # 构建完整梯度矩阵
    grad_E = np.zeros_like(E)
    grad_E[0] = grad_king  # king对应第0行
    grad_E[1] = grad_queen  # queen对应第1行
    grad_E[2] = grad_man    # man对应第2行
    grad_E[3] = grad_woman  # woman对应第3行

    # 更新嵌入矩阵（梯度下降）
    E -= learning_rate * grad_E

    # 记录损失
    loss_history.append(loss)

    # 每100次迭代打印一次信息
```

通过测试发现在 100 次迭代内就已找到最优迭代结果





优化后结果：

```
king嵌入: [0.10124563 0.01454765 0.0426296 ]
man嵌入: [ 0.10634707  0.04836939 -0.02480819]
woman嵌入: [ 0.10583022 -0.01796769 -0.06871223]
king - man + woman = [ 0.10072877 -0.05178942 -0.00127444]
queen嵌入: [ 0.10072877 -0.05178942 -0.00127444]
最终损失: 0.000000
预测与真实queen的差异: 0.000000
```