# word2vec 实验报告

2023200440

2024 年 11 月 28 日

## 1 实验目的

本次实验是完成补充 skip-gram 算法计算损失和梯度的部分，完成模型的训练，得到实验结果和词向量图。从而加深对词向量和 skip-gram 算法的理解。

## 2 实验环境

机器：windows11
解释器：python3.12
编辑器：pycharm

## 3 实验方法

### 3.1 1.sigmoid 函数实现

```
s = 1 / (1.0 + np.exp(-x))
```

### 3.2 2. 计算交叉熵损失:

首先对中心词向量进行维度转化，变为 d*1 的列向量，然后与外部词向量矩阵相乘，计算每一行 (每一个外部词向量) 与中心词向量的点积。然后取出第 outsideWordIdx 个计算交叉熵损失

```
d = centerWordVec.shape[0]          # 词向量长度
```

```
v_c = np.reshape(centerWordVec, (d, 1))     # 词向量维度改为(d, 1)
# 计算外部词向量矩阵和中心词向量的乘积
score = np.dot(outsideVectors, v_c)
y_hat = np.exp(score) / np.sum(np.exp(score))
p = y_hat[outsideWordIdx]
loss = -np.log(p)      # 计算交叉熵损失
```

### 3.3 3. 计算梯度

先构造 one-hot 向量，根据梯度公式分别计算外部词向量梯度和中心词向量梯度，然后将中心词向量梯度转为行向量。

```
N = outsideVectors.shape[0]       # 外部词向量数
y = np.zeros((N, 1))       # one-hot向量
y[outsideWordIdx][0] = 1
gradCenterVec = np.dot(outsideVectors.T, y_hat - y)
gradOutsideVecs = (y_hat - y) @ v_c.T
gradCenterVec = gradCenterVec.flatten()      # 转为行向量
```

### 3.4 4. 模型训练

训练时间：1.5h 左右 训练生成的 word2vec 图：

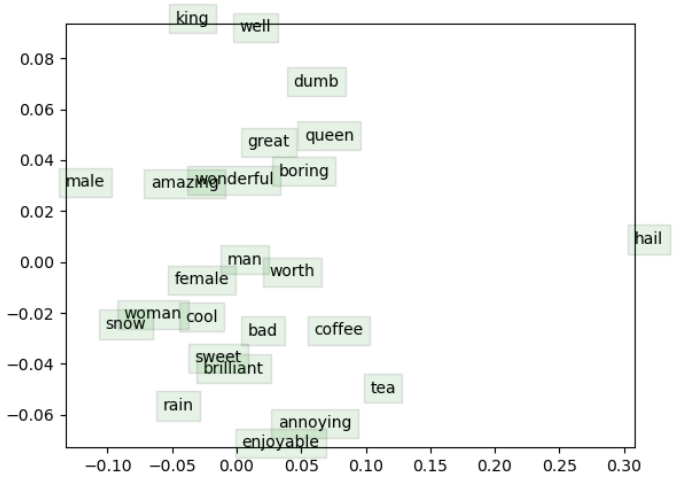


图 1: word2vec

# 4 实验解释

## 4.1 词向量

计算机无法处理人类的语言文字，所以需要将文字映射到计算机可以理解和处理的形式，即词向量。one-hot 是一种经典的表示方法，将每个单词都映射到一个 V 维向量中，不过 one-hot 存在稀疏性，无法捕捉语义相似和上下文信息等问题，特别是维度爆炸问题。word2vec 则可以较好地解决这个问题，能够将单词表示为一个密集的高维词向量，并且语义相近的词在向量空间中比较相近，词向量的相近可以通过余弦相似度计算。词向量的长度为所有单词的个数，不同位置的值代表了相应的词与中心词语义的相近程度。

### 4.1.1 对 word2vec 图的理解

由图 1，可以将每个词与原点的连线看做一个词向量，向量的接近程度代表了两个词语义的相近程度。我们知道，skip-gram 通过中心词来预测上下文单词，容易从图中看出中心词是单词 man，以 man 为中心通过不同大小的窗口去看，可以推测出中心词 man 的上下文。对于其他词也有相似的

效果。我们可以看到 female，woman 等与 man 相近，而 hail，dumb 等与 man 较远，这与直观理解也是很相近的。

## 4.2 skip-gram 理解

word2vec 有 CBOW 和 skip-gram 两种实现模型。原理和实现相似，本次实验则是对 skip-gram 进行学习。 skip-gram 模型通过给定的中心词来预测上下文的单词，预测上下文的范围由窗口的大小决定。skip-gram 是一个比较简单的浅层网络，由向量输入层、隐藏层、输出层，输出结果经过 softmax 处理。其中隐藏层的权重矩阵就是模型学习的 wordvectors，称为词典。我们将输入 one-hot 向量与隐藏层的权重矩阵相乘其实得到的结果就是 one-hot 向量为 1 的下标 one_idx 对应的权重矩阵的第 one_idx 行，这个就是中心词对应的词向量 v_c。然后将 v_c 与输出层权重矩阵相乘 Wv_c，这里的 W 是隐藏层权重矩阵的转置，最后对输出结果做 softmax 就得到了上下文单词的概率。