作业 25 Word2Vec 调研报告

自然语言处理（NLP）的核心挑战是如何让计算机理解人类语言的含义。传统方法使用 one-hot 编码表示词汇，但面临维度灾难和无法捕获语义关系的问题。Word2Vec 模型通过将词汇表示为**稠密**向量，有效解决了这些问题，成为现在的 NLP 的基石。

1 Word2Vec 基本思想

基于上下文的词汇表示：一个词的意思不是凭空产生的，是由它经常出现在什么词旁边（上下文）决定的。

分布式假设： 想象一下，“苹果”这个词可能出现在“吃苹果”、“红苹果”、“苹果手机”这些不同的上下文中。“香蕉”呢？常出现在“吃香蕉”、“黄香蕉”、“香蕉皮”中。“苹果”和“香蕉”经常出现在类似的位置（比如“吃 XXX”、“XX 颜色的 XXX”），那么 Word2Vec 就会认为这两个词在语义上是有相似之处的，它们对应的向量在数值上也就应该比较接近。

低维稠密向量：低维意思是相对于词汇表大小（可能几万到几百万）来说，向量维度（通常 50-300 维）非常小，有效压缩了信息。稠密意思是向量里的大部分数值都不是 0（不像 One-Hot 向量只有一个 1，其他全是 0）。

语义相似的词，向量就更接近，在这个向量空间中，两个词的向量夹角越小，距离越近，它们通常就越相似。相似性可以是语义的（如“国王”和“王后”，“男人”和“女人”），也可以是语法的（如“run”和“running”，“quick”和“quickly”）。

Word2Vec 有两种主要训练方式，目标都是通过上下文关系来学习词向量。

CBOW 是给上下文 -> 猜中心词，即给你一句话中间缺失一个词的周围几个词（比如"猫 坐在 __ 上"），模型的目标是预测中间缺失的那个词（“垫子”或“沙发”）；Skip-gram 是给中心词 -> 猜上下文，即给你一个中心词（比如“猫”），模型的目标是预测它周围可能出现的词（如“可爱的”、“坐在”、“垫子”、“上”）。

2 向量计算

（1）相似度计算

目标： 衡量两个词语在语义上的相似程度（比如，“汽车”和“卡车”很相似，“汽车”和“苹果”就不太相似），最常用的方法是余弦相似度。每个词的向量是空间中一个带方向的箭头。余弦相似度衡量的是两个向量方向的一致性。方向越接近（夹角越小），余弦值越接近 1，表示词语越相似。方向垂直（夹角 90 度），余弦值为 0，表示词语几乎无关。方向完全相反（夹角 180 度），余弦值为 -1，表示词语存在强烈的某种对立。

对于两个词语的向量 vec_A 和 vec_B。首先计算它们的点积，(vec_A[0]*vec_B[0]) + (vec_A[1]*vec_B[1]) + ... + (vec_A[n]*vec_B[n]) ；然后计算各自的模： $||vec\_A|| = sqrt(vec\_A[0]^2 + vec\_A[1]^2 + ... + vec\_A[n]^2)$ ；最后余弦相似度 = (点积) / (||vec_A|| * ||vec_B||)

（2）语义关系类比

目标： 捕捉词语之间特定的关系类型（如“国家-首都”、“部分-整体”），并利用这种关系进行类比推理，方法是利用向量加减运算。

词语之间的某种语义关系在训练过程中被编码成了向量空间中的一种特定的方向或位移向量。例如：king 和 queen 的区别主要在于性别这个维度。假设要计算 king - man + woman ≈ queen，king 向量代表了“男性君主”这个概念。man 向量代表了“普通男性”这个概念。那么 (king - man) 这个向量运算，就近似地剔除了“男人”的通用属性，保留了“王”这个特殊地位和权力属性，得到的向量代表“王权”。woman 向量代表了“普通女性”这个概念。把代表“君主身份”的向量 (king - man) 加到代表“女性”的向量 woman 上，就组合成了“女性君主”这个概念，也就是 queen 所代表的含义。所以，整个运算 king - man + woman 的结果向量在空间中非常接近 queen 的向量。

3 基于 Gensim 的实践

代码如下：

```python
import gensim
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from gensim import downloader


print("下载text8语料库...")
corpus = downloader.load('text8')  # 维基百科文本清洗后的语料
print("下载完成")

# 配置模型参数
model = Word2Vec(
    sentences = corpus,
    vector_size=100,
    window=3,
    min_count=1,
    sg=1,
    workers=4,
    epochs=10
)

```

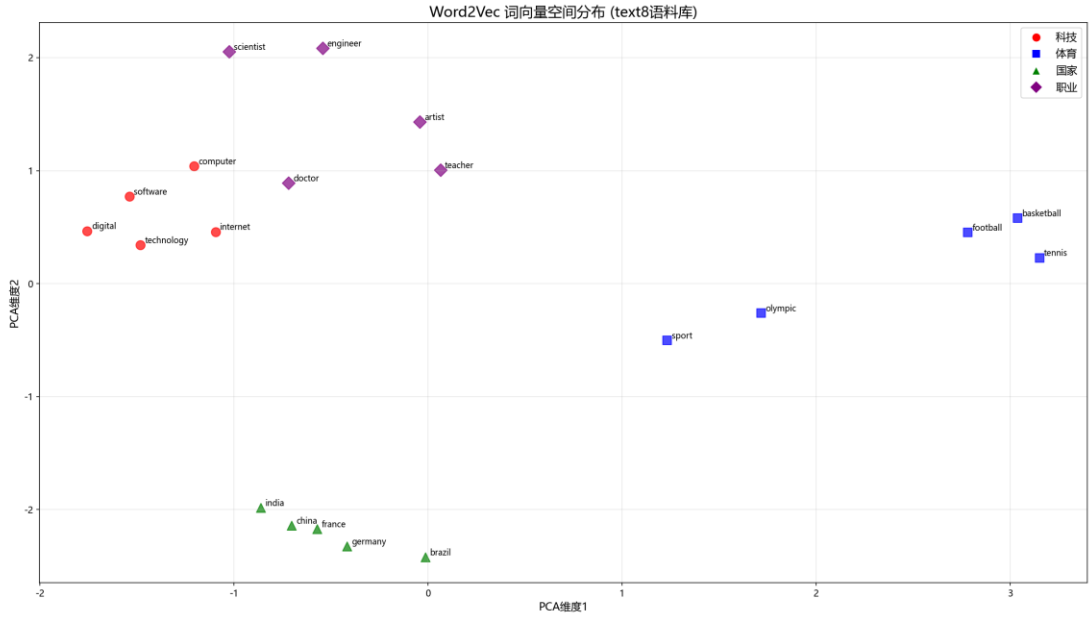
```python
# 保存和加载模型
model.save("word2vec_text8.model")
model = Word2Vec.load("word2vec_text8.model")

# 词语相似度查询
for word in ['computer', 'engine', 'language', 'sport']:
    print(f"\n与'{word}'最相似的词：")
    print(model.wv.most_similar(word, topn=5))

# 类比推理
print("\n类比推理: king - man + woman =")
result = model.wv.most_similar(positive=['woman', 'king'], negative=['man'], topn=3)
for word, score in result:
    print(f"{word}: {score:.3f}")

# 选择要可视化的词汇
categories = {
    "科技": ["computer", "internet", "software", "technology", "digital"],
    "体育": ["football", "basketball", "tennis", "sport", "olympic"],
    "国家": ["france", "germany", "china", "india", "brazil"],
    "职业": ["doctor", "engineer", "teacher", "scientist", "artist"]
}

words = []
for cat in categories.values():
    words.extend(cat)

# 提取词向量
vectors = model.wv[words]

# PCA降维
pca = PCA(n_components=2)
result = pca.fit_transform(vectors)
```

```python
# 绘制可视化
plt.figure(figsize=(14, 10))
colors = {'科技': 'red', '体育': 'blue', '国家': 'green', '职业': 'purple'}
markers = {'科技': 'o', '体育': 's', '国家': '^', '职业': 'D'}


for i, word in enumerate(words):
    category = next(cat for cat, words in categories.items() if word in words)
    plt.scatter(result[i, 0], result[i, 1],
                c=colors[category],
                marker=markers[category],
                s=100, alpha=0.7)
    plt.annotate(word, xy=(result[i, 0], result[i, 1]),
                 xytext=(5, 2),
                 textcoords='offset points')

from matplotlib.lines import Line2D
legend_elements = [Line2D([0], [0], marker=markers[cat], color='w', label=cat,
                          markersize=10, markerfacecolor=colors[cat])
                   for cat in categories]
plt.legend(handles=legend_elements, loc='best', fontsize=12)
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  # 指定中文字体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
plt.title('Word2Vec 词向量空间分布 (text8语料库)', fontsize=16)
plt.xlabel('PCA维度1', fontsize=12)
plt.ylabel('PCA维度2', fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

(parameter) xytext: Sequence[float]

运行程序，得到的结果如下：

```
(E:\conda_envs\AIMath) PS E:\conda_python\AIMath> python .\question25.py
下载text8语料库...
下载完成

与'computer'最相似的词：
[('computers', 0.8190619945526123), ('hardwired', 0.7775139212608337), ('computing', 0.7677469253540039), ('hardware', 0.7655975818634033), ('pdas', 0.7570258975028992)]

与'engine'最相似的词：
[('engines', 0.8749544024467468), ('turbofan', 0.7730128765106201), ('vulcain', 0.7655909061431885), ('turbine', 0.7578784227371216), ('turbojet', 0.7498461008071899)]

与'language'最相似的词：
[('languages', 0.8808875679969788), ('patois', 0.7722416520118713), ('agglutinative', 0.7626668810844421), ('fusional', 0.7588006854057312), ('vocabulary', 0.758547842502594)]

与'sport'最相似的词：
[('sports', 0.7879763841629028), ('competitions', 0.7706478834152222), ('biathlon', 0.7705586552619934), ('soccer', 0.7703060507774353), ('golf', 0.7690773010253906)]

类比推理: king - man + woman =
sigismund: 0.658
queen: 0.653
dowager: 0.634
(E:\conda_envs\AIMath) PS E:\conda_python\AIMath>
```


Word2Vec 词向量空间分布 (text8语料库)
科技
体育
国家
职业
scientist
engineer
artist
computer
teacher
doctor
software
digital
internet
technology
basketball
football
tennis
olympic
sport
india
china
france
germany
brazil
PCA维度1
PCA维度2