Report of Deep Learning for Natural Langauge Processing

Xintong Zhang
zhangxintong0810@icloud.com

Abstract

本次作业利用给定金庸语小说语料库，利用 Word2Vec 模型训练词向量，以计算不同词向量之间的语意距离和某一类词语的聚类关系，最终通过以上方法验证了词向量的有效性。

Introduction

Word2Vec 模型

词向量是用于表示词意义的向量，并且还可以被认为是词的特征向量或表示。将词映射到实向量的技术称为词嵌入。起初，我们使用 One-hot 向量来表示词，每个词都被表示为一个长度为 N 的向量，可以直接由神经网络使用。但是由于任意两个不同词的 One-hot 向量之间的余弦相似度为 0，因此独热向量不能对词之间的相似度进行编码。word2vec 工具是为了解决上述问题而提出的。它将每个词映射到一个固定长度的向量，这些向量能更好地表达不同词之间的相似度和类比关系。

Word2Vec 是一种用于生成词向量的神经网络模型，能够将词语映射到低维稠密向量（如 300 维），利用向量间的几何关系捕捉语义和语法关系。其核心思想是通过词语的上下文信息进行自监督学习，主要包含两种模型：连续词袋模型（CBOW，Continuous Bag of Words）和跳元模型 Skip-Gram。它们的训练依赖条件概率，条件概率可以被看作使用语料库中的一些词来预测另一些词。

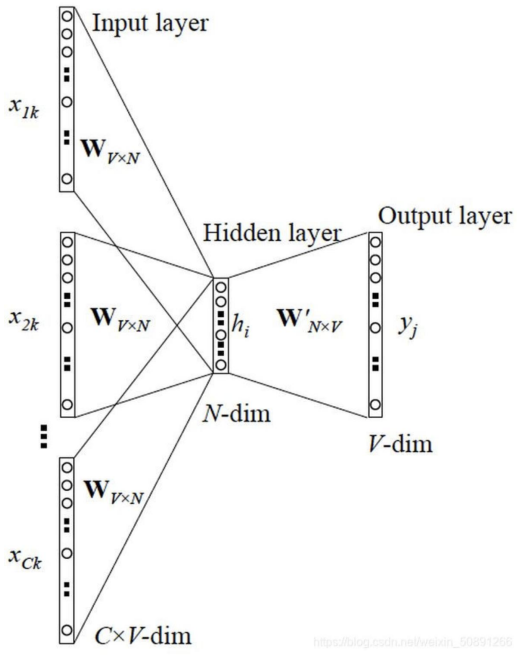

图 1　Skip-gram 网络结构

Skip-gram 的网络结构如所示图 1, Skip-gram 模型假设一个词可以用来在文本序列中生成其周围的词。x 是中心词的 one-hot encoder 形式的输入，通过将中心词映射为词向量以预测上下文输出概率，每个上下文词通过 Softmax 生成概率分布，损失函数为多个上下文词预测损失的均值。

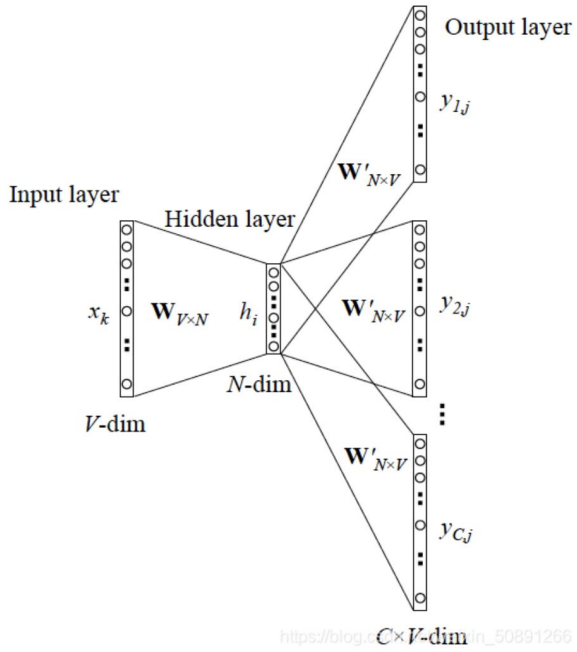

图 2　CBOW 网络结构

连续词袋模型（CBOW）类似于跳元模型，其网络结构如图 2 所示。与跳元模型的主要区别在于，连续词袋模型假设中心词是基于其在文本序列中的周围上下文词生成的。CBOW 的网络结构如下,与 Skip-gram 的模型并联不同，这里是输入变成了多个单词，所以要对输入处理为将每个上下文词的 one-hot 向量与输入权重矩阵相乘，得到词向量（Embedding），再对多个上下文词的词向量取平均，生成隐藏层向量。

Methodology

1. 导入所需的库

导入代码需要的相关的库，其中 jieba 用于分词，tqdm 用于打印进度条

```
import os
import jieba
import chardet
from tqdm import tqdm
import torch
import torch.nn as nn
import torch.nn.functional as F
import random
import re
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
```

2. 数据读取与定义预处理的函数

首先，定义读入函数，由于 mac 存在一些格式不兼容问题，这里选择自适应文本编码信息的导入方式读取数据。

其次，对读入数据进行预处理，定义预处理函数。由于这里的预处理操作不能像之前预处理一样，将标点符号都去除，因为对于文本生成模型来说，标点符号也在中文文本中具有

其存在的意义，可以表示停顿——例如句号和逗号，也可以表示情感——例如问号表示疑问、感叹号表示惊讶，也可以代表说的话——例如双引号和冒号。所以不能随意地删除。

而观察射雕英雄传的文本可以发现，存在许多不属于中文文本中应该有的符号，这些符号是预处理阶段应该删除的，包括回车符、制表符、空格，还有一些乱码，如□。同时，射雕英雄传里，有些地方的双引号和单引号以「」『』的形式表示，这需要替换成正常中文文本中的双引号和单引号。同时有些不属于射雕英雄传内容的中文也应该删除。

将所有文本分词并显示每一本书分词后的行数和总行数，通过一个嵌套列表存储，每一句为列表中一个元素，每一句又由分好的词构成一个列表，这也是 word2vec 训练时需要输入的格式。

```
def read_text(path):
    """
    用于 Word2Vec 模型：读取文件、自动检测编码、清洗文本，
    按中文句号分句，并对每个句子进行分词，返回分词后句子的列表。
    """
    abs_path = os.path.abspath(path)
    with open(abs_path, "rb") as raw_file:
        raw_data = raw_file.read()
    detected = chardet.detect(raw_data)
    encoding = detected.get("encoding", "utf-8")
    confidence = detected.get("confidence", 0)
    if encoding is None or confidence < 0.7:
        encoding = 'gb18030'
    try:
        with open(abs_path, "r", encoding=encoding) as file:
            file_content = file.read()
    except UnicodeDecodeError:
        with open(abs_path, "r", encoding='gb18030') as file:
            file_content = file.read()

    # 去除无关广告及乱码
    file_content = file_content.replace("本书来自www.cr173.com免费txt小说下载站", '')
    file_content = file_content.replace("更多更新免费电子书请关注www.cr173.com", '')
    file_content = file_content.replace("---- 〖新语丝电子文库(www.xys.org)〗", '')
    file_content = file_content.replace("\t", '')
    file_content = file_content.replace(" ", '')
    file_content = file_content.replace('\u3000', '')
    file_content = file_content.replace('」', '”')
    file_content = file_content.replace('「', '“')
    file_content = file_content.replace('□', '')
    file_content = file_content.replace('』', '’')
    file_content = file_content.replace('『', '‘')

    # 按句号分句，再对每个句子分词
    sentences = file_content.split("。")
    tokenized_sentences = []
    for sent in sentences:
        sent = sent.strip()
        if sent:
            tokens = list(jieba.cut(sent))
            tokenized_sentences.append(tokens)
    print(f"文件 {os.path.basename(path)} 分句后句数：", len(tokenized_sentences))
    return tokenized_sentences
```

由于语料库比较庞大，16 篇小说如果全部用于训练的话，训练过程过于漫长，于是我只选用射雕英雄传、倚天屠龙记、 天龙八部、神雕侠侣、笑傲江湖这 5 篇作为语料库。

```
# 定义小说列表及数据所在目录（请确认文件名与小说名称一致）
novels = ["倚天屠龙记", "天龙八部", "射雕英雄传", "神雕侠侣", "笑傲江湖"]
data_dir = "./jyxstxtqj_downcc.com/"
# 读取所有小说的文本数据
from gensim.models import Word2Vec

all_sentences = []
for novel in novels:
    path = os.path.join(data_dir, novel + ".txt")
    tokenized_sentences = read_text(path)
    all_sentences.extend(tokenized_sentences)
```

3. Word2Vec 参数设定与模型训练

Word2Vec 中需要定义的参数有以下：sentences 是输入的语料，即分词得到的嵌套列表。size 是训练得到的词向量的维数。window 为滑窗大小，也就是训练词与上下文词最远的距离。min_count 是指过滤掉一些低词频的词。sg 为选择 CBOW 模型还是 skip-gram 模型，0 为 CBOW 模型，1 为 skip-gram 模型，默认为 CBOW 模型。

```
# 训练 Word2Vec 模型，参数可根据需要调整（vector_size: 词向量维数；window: 上下文窗口；min_count: 过滤低频词）
from gensim.models import Word2Vec
w2v_model = Word2Vec(sentences=all_sentences, vector_size=100, window=5, min_count=5, workers=4)
print("Word2Vec 模型训练完成！")
```

4. 聚类分析与模型测试

通过分析词向量之间的语义距离、某一类词语的聚类情况等测试模型的精确程度。具体代码编写和测试输出结果见下一部分内容。

Experimental Studies

1. 词向量之间的语义距离

（1）词向量加减

杨过+黄蓉-郭靖 ≈ 小龙女，其中杨过与小龙女是夫妻/师徒关系，黄蓉与郭靖是夫妻关系。由此可以反映出词向量基于金庸武侠小说训练，模型可以捕捉到人物之间的复杂关系：

```
try:
    result = w2v_model.wv.most_similar(positive=["杨过", "黄蓉"], negative=["郭靖"], topn=1)
    print("示例（词向量加减）：杨过 + 黄蓉 - 郭靖 得到词：", result[0][0], "（相似度：{:.4f}）".format(result[0][1]))
except Exception as e:
    print("Word2Vec 词向量加减运算错误：", e)
```

```
示例（词向量加减）：杨过 + 黄蓉 - 郭靖 得到词： 小龙女 （相似度：0.8200）
```

与这个问题相似的问法是寻找对应关系（寻找 X1 之于 X2 等于 Y1 之于什么），如郭靖之于黄蓉对应杨过之于谁，回答结果同上。

```
try:
    analogy_result = w2v_model.wv.most_similar(positive=["杨过", "黄蓉"], negative=["郭靖"], topn=1)
    print("类比关系：郭靖之于黄蓉 对应 杨过之于：", analogy_result[0][0], "（相似度：{:.4f}）".format(analogy_result[0][1]))
except Exception as e:
    print("Word2Vec 类比运算异常：", e)
```

```
类比关系：郭靖之于黄蓉 对应 杨过之于： 小龙女 （相似度：0.8200）
```

（2）寻找人物与武功之间相似度

如寻找与段誉相关的武功相似度排名。从结果来看，段誉使用或被提及到的武功的频率：

一阳指 > 独孤九剑 >九阴真经 > 打狗棒法 > 降龙十八掌

一阳指的相关性大于其他几个，这么看来段誉的一阳指功夫还是相当到位的。这个排序和五部小说中主人公出现频率也呈正相关，所以得到的结果还是比较符合认知的。

```
# 任务1.4：人物与特定武功的相似度分析（以段誉为例）

martial_arts = ["降龙十八掌", "打狗棒法", "九阴真经", "一阳指","独孤九剑"]

# 基于词向量相似度计算
sim_scores = {}
print("基于词向量相似度，段誉相关武功排名：")
for ma in martial_arts:
    try:
        sim = w2v_model.wv.similarity("段誉", ma)
        sim_scores[ma] = sim
        print("  {} : {:.4f}".format(ma, sim))
    except KeyError:
        print("武功 '{}' 不在词汇表中。".format(ma))

# 排序并输出最终排名
if sim_scores:
    sim_sorted = sorted(sim_scores.items(), key=lambda x: x[1], reverse=True)
    print("\n最终排序：")
    for ma, sim in sim_sorted:
        print("  {} : {:.4f}".format(ma, sim))
else:
    print("\n未找到任何有效的武功与 '段誉' 的词向量相似度。")
```

✓ 0.0s

```
基于词向量相似度，段誉相关武功排名：
  降龙十八掌 : 0.2376
  打狗棒法 : 0.2549
  九阴真经 : 0.3043
  一阳指 : 0.3790
  独孤九剑 : 0.3158

最终排序：
  一阳指 : 0.3790
  独孤九剑 : 0.3158
  九阴真经 : 0.3043
  打狗棒法 : 0.2549
  降龙十八掌 : 0.2376
```

2. 某一类词语的聚类

（1）人物亲近关系分析

寻找与一个人物关联度最高的五个人物，如寻找张无忌关系最接近的十个人名。由这个相关度的结果来看，在张无忌心中这几个女人在心中的排行顺序应该为：

赵敏>周芷若>小昭>殷离

其他人物的相关度高应该是由于都是主角的相似度原因。

```
# 任务2：人物亲近关系分析
characters = ["张无忌", "赵敏", "周芷若", "小昭", "殷离", "郭靖", "黄蓉", "杨过", "小龙女", "段誉", "令狐冲", "任盈盈"]
if "张无忌" in w2v_model.wv:
    target_sim = {}
    for word in characters:
        # 排除自己
        if word != "张无忌" and word in w2v_model.wv:
            sim = w2v_model.wv.similarity("张无忌", word)
            target_sim[word] = sim
    # 按相似度降序排序
    sorted_sim = sorted(target_sim.items(), key=lambda x: x[1], reverse=True)
    print("与 '张无忌' 关系最接近的十个人物：")
    for word, sim_score in sorted_sim[:10]:
        print("  {} : {:.4f}".format(word, sim_score))
else:
    print("词 '张无忌' 未出现在 Word2Vec 模型中。")
```

✓ 0.0s

```
与 '张无忌' 关系最接近的十个人物：
  赵敏 : 0.6929
  周芷若 : 0.6928
  段誉 : 0.6892
  令狐冲 : 0.6819
  小昭 : 0.5976
  殷离 : 0.5209
  杨过 : 0.4025
  小龙女 : 0.3827
  黄蓉 : 0.3728
  郭靖 : 0.3638
```

（2）武功关系分析

寻找与降龙十八掌相似度最高的十个词，可以看出所有输出均为武功，且基本均为拳法、掌法等技能型武功。

```
# （2）武功关系分析：寻找与“降龙十八掌”最相似的武功
martial_list = ["降龙十八掌", "一阳指", "六脉神剑", "北冥神功", "凌波微步", "左右互搏", "打狗棒法", "辟邪剑法"]
if "降龙十八掌" in w2v_model.wv:
    similar_martial = w2v_model.wv.most_similar("降龙十八掌", topn=10)
    print("与 '降龙十八掌' 相似度最高的武功：")
    for word, score in similar_martial:
        print("  {} : {:.4f}".format(word, score))
else:
    print("词 '降龙十八掌' 未出现在 Word2Vec 模型中。")
```

✓ 0.0s

```
与 '降龙十八掌' 相似度最高的武功：
  打狗棒法 : 0.8996
  拳 : 0.8853
  拳法 : 0.8810
  神通 : 0.8681
  掌法 : 0.8674
  六脉 : 0.8582
  一阳指 : 0.8553
  七伤 : 0.8515
  玉女 : 0.8495
  空明拳 : 0.8447
```

（3）门派关系分析

寻找与峨嵋派相似度最高的十个派别，可以看到输出均为相似名声度很高的大派。

```
# （3）门派关系分析：寻找与“峨嵋派”最相似的门派（选取包含“派”的词）
factions = [word for word in w2v_model.wv.key_to_index if "派" in word]
if "峨嵋派" in w2v_model.wv:
    faction_sim = {}
    for faction in factions:
        if faction == "峨嵋派":
            continue
        try:
            faction_sim[faction] = w2v_model.wv.similarity("峨嵋派", faction)
        except:
            continue
    faction_sim_sorted = sorted(faction_sim.items(), key=lambda x: x[1], reverse=True)
    print("与 '峨嵋派' 相似度最高的十个门派：")
    for faction, score in faction_sim_sorted[:10]:
        print("  {} : {:.4f}".format(faction, score))
else:
    print("词 '峨嵋派' 未出现在 Word2Vec 模型中。")
```

✓ 0.0s

```
与 '峨嵋派' 相似度最高的十个门派：
  华山派 : 0.9326
  青城派 : 0.9012
  本派 : 0.8959
  武当派 : 0.8731
  正派 : 0.8573
  贵派 : 0.8391
  派众 : 0.7871
  古墓派 : 0.7580
  五派 : 0.7579
  逍遥派 : 0.7480
```

（4）利用 **K-Means** 聚类分析目标词

利用 **K-Means** 聚类分析一组词之间的聚类结果，结果展示如图 **3** 所示。可以看到，**K-Means** 聚类可以很好的把这组词分为人物、武功、帮派 **3** 种类别，且在其二维投影平面有很好的区分度，由此可知，Word2Vec 模型能够很好的提取词向量间的特征信息。

```python
# 任务2.4：利用 K-Means 聚类分析目标词
target_words = ['郭靖', '黄蓉', '杨过', '小龙女', '张无忌', '赵敏',
                '降龙十八掌', '打狗棒法', '九阴真经', '乾坤大挪移',
                '少林', '武当', '明教', '丐帮']

# 过滤出存在于模型词汇表中的词
valid_words = [word for word in target_words if word in w2v_model.wv]
vectors = [w2v_model.wv[word] for word in valid_words]

# 利用 pypinyin 将中文词转换为英文拼音表示
# 例如：'郭靖' -> 'guojing'
pinyin_words = [''.join(lazy_pinyin(word)) for word in valid_words]

from sklearn.cluster import KMeans
# 使用 K-Means 将这些词分为 3 类（可根据需要调整聚类数量）
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(vectors)
print("K-Means 聚类结果：")
for word, cluster in zip(valid_words, clusters):
    print(f"  {word} -> Cluster {cluster}")

# 使用 PCA 将这些词的向量降到二维，并进行可视化
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
points = pca.fit_transform(vectors)

plt.figure(figsize=(10, 6))
for i, pinyin_word in enumerate(pinyin_words):
    x, y = points[i, 0], points[i, 1]
    plt.scatter(x, y, c=f"C{clusters[i]}", s=100)
    plt.text(x + 0.02, y + 0.02, pinyin_word, fontsize=10)
plt.title("Cluster results (PCA visualization)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

```
K-Means 聚类结果：
  郭靖 -> Cluster 0
  黄蓉 -> Cluster 0
  杨过 -> Cluster 0
  小龙女 -> Cluster 0
  张无忌 -> Cluster 0
  赵敏 -> Cluster 0
  降龙十八掌 -> Cluster 1
  打狗棒法 -> Cluster 1
  九阴真经 -> Cluster 1
  少林 -> Cluster 2
  武当 -> Cluster 2
  明教 -> Cluster 2
  丐帮 -> Cluster 2
```

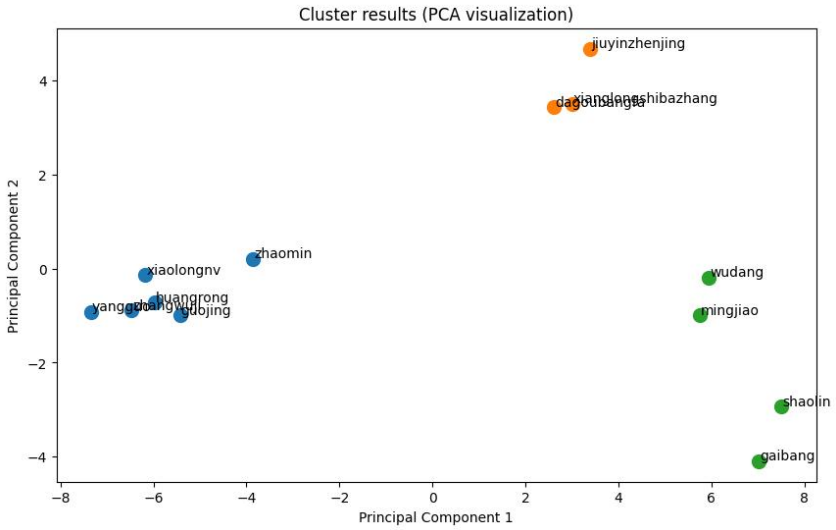

图 3　聚类结果展示

Conclusions

本次研究基于金庸小说语料库，通过 Word2Vec 模型训练词向量，验证了词向量在语义关系捕捉与文本生成任务中的有效性。以下是主要结论：

1. 词向量语义关系的有效性*

Word2Vec 模型成功捕捉了人物间的复杂关系（如“杨过+黄蓉-郭靖≈小龙女”），并通过类比推理（如“郭靖之于黄蓉对应杨过之于小龙女”）展示了词向量的语义关联性。实验结果与小说情节高度吻合，表明模型能够有效编码文本中的逻辑关系。

2. 词语聚类分析的合理性

人物关系分析：张无忌的关联人物排名（赵敏>周芷若>小昭>殷离）与小说情节一致，验证了模型对角色互动的敏感性。

武功与门派关联：降龙十八掌的相似武功多为掌法类（如打狗棒法），峨嵋派的关联门派均为正派大宗（如少林、武当），反映了模型对语义类别的区分能力。

本研究表明，基于 Word2Vec 的词向量方法在金庸小说文本分析中具有显著应用价值，为后续的文本挖掘（如情感分析、情节预测）提供了可靠的基础。未来工作可进一步融合深度语言模型（如 BERT）以捕捉更复杂的语义层次。

Problems and Future Improvements:

1. 引入更细粒度的预处理（如实体对齐、错别字校正）以优化词汇覆盖。
2. 结合注意力机制（如 Transformer）增强长文本生成的逻辑连贯性。
3. 扩展语料库至更多金庸小说，提升低频词和复杂关系的建模能力。

References

[1] Zenchang Qin (2025)，Embeddings of Words.Vol. 3: 23: pp. 1-12.