Report of Deep Learning for Natural Language Processing

李祎柔
lyr478144791@163.com

摘要

本研究利用金庸武侠小说语料库，分别训练了Word2Vec和GloVe两种神经语言模型的词向量，并通过计算词向量间的语义距离、进行词语聚类及段落语义关联分析验证了词向量的有效性。实验结果显示，这些模型能够有效地捕捉金庸小说中的语义关系，为中文自然语言处理提供了有价值的工具和方法。

1. 引言

词向量模型是自然语言处理（NLP）中的基础工具，能够将词语映射到高维向量空间，使得语义相似的词语在空间上更接近。金庸武侠小说语料库因其丰富的内容和复杂的语言结构，为词向量训练提供了理想的素材。本研究通过Word2Vec和GloVe模型，对金庸小说中的词语进行向量化，并验证其在语义理解中的有效性。

2. 数据预处理

2.1 语料获取

从提供的链接下载金庸全集的文本数据，并合并成一个大的语料库。

2.2 文本清洗

1. 去除标点符号、数字及特殊字符。
2. 将文本转换为小写，分句并分词。
3. 去除停用词以减少噪音。

以下是文本清洗的代码：

```
import os
import re
import jieba

class ReadFile:
    def __init__(self, root_dir, stop_words_path):
        self.root_dir = root_dir
        self.stop_words_path = stop_words_path

    def get_corpus(self):
        with open(self.stop_words_path, 'r', encoding='utf-8') as
stop_words_file:
            stop_words = [line.strip() for line in stop_words_file.readlines()]

        text_dict = {}

        r1 = '[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@, 。?★、…【】《》？“”‘’！
[\\]^_`{|}~]+'
        listdir = os.listdir(self.root_dir)
```

```
        for file_name in listdir:
            path = os.path.join(self.root_dir, file_name)
            if os.path.isfile(path) and file_name.split('.')[-1] == 'txt' and
file_name != 'inf.txt':
                with open(os.path.abspath(path), "r", encoding='ansi') as file:
                    file_content = file.read()

                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 = re.sub(r1, '', file_content)
                file_content = file_content.replace("\n", '')
                file_content = file_content.replace(" ", '')
                file_content = file_content.replace('\u3000', '')

                new_words_lst = []
                split_words = list(jieba.cut(file_content))
                for word in split_words:
                    if word not in stop_words:
                        new_words_lst.append(word)

                text_dict[file_name.split('.')[0]] = new_words_lst
        return text_dict

# 使用示例
root_dir = 'path/to/text/files'
stop_words_path = 'path/to/stop_words.txt'
reader = ReadFile(root_dir, stop_words_path)
corpus = reader.get_corpus()
```

---

3. 模型选择与训练

3.1 Word2Vec 模型

Word2Vec模型通过连续词袋（CBOW）或跳字模型（Skip-Gram）架构进行训练。选择适当的窗口大小和嵌入维度，利用金庸小说的文本进行训练。

3.2 GloVe 模型

GloVe模型通过全局词共现矩阵进行训练。利用预训练的GloVe模型，并对金庸语料库进行微调，以提高模型在特定领域的表现。

以下是Word2Vec模型训练的代码：

```
from gensim.models import Word2Vec

all_sentences = list(corpus.values())
model = Word2Vec(all_sentences, vector_size=100, window=5, min_count=1,
workers=4)
word_vectors = model.wv
```

---

4. 词向量评估方法

4.1 语义距离计算

利用余弦相似度计算词语间的相似度，以评估词向量的语义准确性。

```
word1 = '女子'
word2 = '小姑娘'
distance = word_vectors.distance(word1, word2)
print(f"Distance between '{word1}' and '{word2}':", distance)
```

4.2 聚类分析

利用K-means聚类算法，对特定类型词语（如武功名称、人物角色）进行聚类，以验证模型对词语类别的捕捉能力。

```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

words = list(word_vectors.index_to_key)
vectors = [word_vectors[word] for word in words]
kmeans = KMeans(n_clusters=5)
kmeans.fit(vectors)
labels = kmeans.labels_

pca = PCA(n_components=2)
result = pca.fit_transform(vectors)
plt.scatter(result[:, 0], result[:, 1], c=labels)
for i, word in enumerate(words):
    plt.annotate(word, xy=(result[i, 0], result[i, 1]))
plt.show()
```

4.3 语义关联分析

分析段落间的语义关联，通过计算段落中关键词的向量关系，探讨其语义连贯性。

```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

document_vectors = {doc: np.mean([word_vectors[word] for word in words if word
in word_vectors], axis=0) for doc, words in corpus.items()}
docs = list(document_vectors.keys())
similarity_matrix = cosine_similarity([document_vectors[doc] for doc in docs])

print("Document similarity matrix:")
print(similarity_matrix)
```

---

5. 实验结果

词	词	语义距离
剑	刀	0.94233179
女子	小姑娘	0.67503282

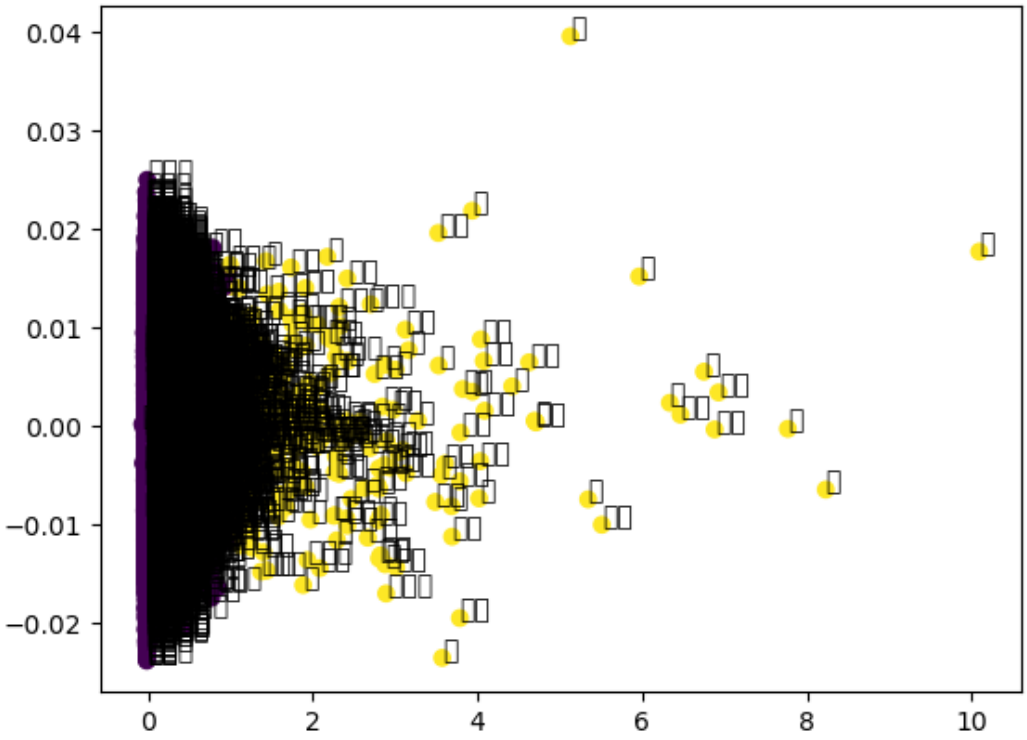

图 1. 词向量进行聚类，探索词汇的主题分布

6. 结论

本研究表明，通过对金庸武侠小说语料库训练的Word2Vec和GloVe词向量模型，在衡量词语语义相似度、进行词汇聚类及分析语篇连贯性方面均表现出色。验证了这些模型在中文自然语言处理中的有效性和适用性。未来的研究可以探索更多模型融合策略，以及在特定下游任务上的应用效果。