Report of Deep Learning for Natural Langauge Processing

Zengchang Qin
zengchang.qin@gmail.com

摘要

本实验利用金庸小说语料（存放于本地文件夹）对词向量进行训练，采用两种模型进行对比：

1.利用 gensim 库训练 Word2Vec 模型；

2.基于 PyTorch 实现的 GloVe 模型（不依赖第三方 glove 库）。

实验流程包括数据加载与预处理（利用 jieba 对中文文本进行分词）、模型训练、定量评估（计算测试词对之间的余弦相似度并输出比较表格）以及聚类分析（利用 KMeans 聚类并计算轮廓系数，结合 PCA 降维后以散点图可视化）。最终，通过对比两种模型的定量结果与聚类效果，分析了不同模型在词向量表达上的差异性与优劣。

简介

词向量作为自然语言处理领域的基础表示方法，能够将离散的文本数据映射到连续的向量空间，使得深度学习模型能够捕获语义信息。Word2Vec 和 GloVe 是两种经典的词向量训练方法，各有其优势。Word2Vec 通过局部上下文窗口建模，而 GloVe 则依赖全局共现信息。在本实验中，我们采用金庸小说作为语料，并实现了两种模型的训练流程。报告主要探讨了：

- 数据预处理流程（中文文本分句与分词）；
- Word2Vec 与基于 PyTorch 的 GloVe 模型训练；
- 两种方法在词向量相似度计算、定量指标（余弦相似度、轮廓系数）及聚类效果上的对比分析。

理论方法

There are models of my research.

M1: Word2Vec Model

本文使用Word2Vec 模型利用神经网络将词映射至低维向量空间，常见的训练方法包括 Skip-Gram 和 CBOW。本实验中采用 gensim 库实现的 Word2Vec，通过设置向量维度、上下文窗口和最小词频等参数进行训练。

M2: GloVe 模型

GloVe（Global Vectors）模型基于词共现矩阵信息，通过构造共现矩阵与加权损失函数训练词向量。本实验使用 PyTorch 自行实现了 GloVe，过程如下：

构建词汇表：统计所有单词出现频率，保留频次大于等于设定阈值的词汇；

构建共现矩阵：遍历每个句子，根据指定窗口以 1/距离 权重累加共现次数；
模型设计：模型包含目标词及上下文词的嵌入矩阵，以及对应的偏置项；
训练过程：利用经典的 GloVe 损失函数，并采用 Adam 优化器完成训练，最终词向量取目标词与上下文词嵌入的和。

实验过程

本报告的整体方法可分为以下几步：

参数配置

1. **数据路径**
   指定金庸小说语料存放的文件夹路径（例如：D:\小说数据集）。
2. **Word2Vec 参数**
   设置向量维度为 100，上下文窗口大小为 5，最小词频 5，训练 10 个 epoch。
3. **GloVe 参数**
   配置词向量维度为 100，窗口大小为 5，学习率 0.05，训练 10 个 epoch，batch size 为 512，并采用 x_max=100、α=0.75 进行权重计算。
4. **聚类参数与测试词**
   聚类使用 KMeans（类别数设为 5）；选取测试词包括 “郭靖”、“黄蓉”、“降龙”、“九阳”、“天下” 等。

数据加载与预处理

采用 Python 的 glob 模块读取指定文件夹下所有 TXT 文件。为应对编码问题，函数尝试依次使用 'utf-8' 与 'gb18030' 编码读取文件，按中文标点（“。！？”）进行句子切分，再利用 jieba 对每个句子进行分词处理。实验输出显示加载的句子数和分词后的句子数均为 274217（示例输出）。

训练 Word2Vec 模型

调用 gensim 的 Word2Vec 接口，将分词后的句子作为训练语料，采用预设参数进行模型训练。训练完成后，模型能够为每个词计算对应的向量表示。

训练 GloVe 模型（PyTorch 实现）

实验流程包括以下步骤：

构建词汇表：利用 Counter 统计分词结果，过滤掉低频词（频次小于 5），词汇表规模约 46282；

构建共现矩阵：遍历分词后的句子，统计每对词在指定窗口内的加权共现次数，共现对数量约为 10452772；

模型定义与训练：采用 PyTorch 构建 GloVe 模型，定义目标词及上下文词嵌入矩阵和偏置向量，通过 Adam 优化器优化权重。训练过程中，每个 epoch 显示平均损失值，10 个 epoch 后的训练损失基本稳定在 60 左右。

定量评估：计算词对余弦相似度

选择预设测试词对（如“郭靖”与“黄蓉”等），分别从 Word2Vec 与 GloVe 模型中提取词向量，计算两者之间的余弦相似度，生成对比表格并保存为 CSV 文件。示例结果部分显示两种模型在词对相似度上的显著差异。

聚类分析与可视化

针对前 500 个高频词的词向量：

聚类：使用 KMeans 进行聚类，并计算轮廓系数。结果显示 Word2Vec 和 GloVe 模型的轮廓系数分别约为 0.0297 与 0.0636；

降维可视化：利用 PCA 将词向量降至二维后绘制散点图，直观展示两种模型的聚类效果。散点图保存在本地文件 clustering_comparison.png 中。

结果分析

词对余弦相似度对比：

	Word1	Word2	Cosine_Sim_Word2Vec	Cosine_Sim_GloVe
0	郭靖	黄蓉	0.868962	0.055814
1	郭靖	降龙	-0.153624	-0.035973
2	郭靖	九阳	0.136275	-0.079265
3	郭靖	天下	-0.162967	-0.033873
4	黄蓉	降龙	-0.249989	0.130562
5	黄蓉	九阳	0.044457	-0.002160
6	黄蓉	天下	-0.236394	-0.077756
7	降龙	九阳	0.579174	0.010138
8	降龙	天下	0.333870	-0.152374
9	九阳	天下	0.230825	-0.028683

图 1：实验结果

Word2Vec 轮廓系数：0.0297
GloVe 轮廓系数：0.0636
Word2Vec 聚类结果（前10行）：

	word	cluster
0	，	1
1	的	4
2	：	3
3	了	3
4	“	3
5	”	1
6		1
7	他	4
8	是	3
9	道	2

GloVe 聚类结果（前10行）：

	word	cluster
0	，	2
1	的	2
2	：	2
3	了	2
4	“	2
5	”	2
6		2
7	他	2
8	是	2
9	道	2

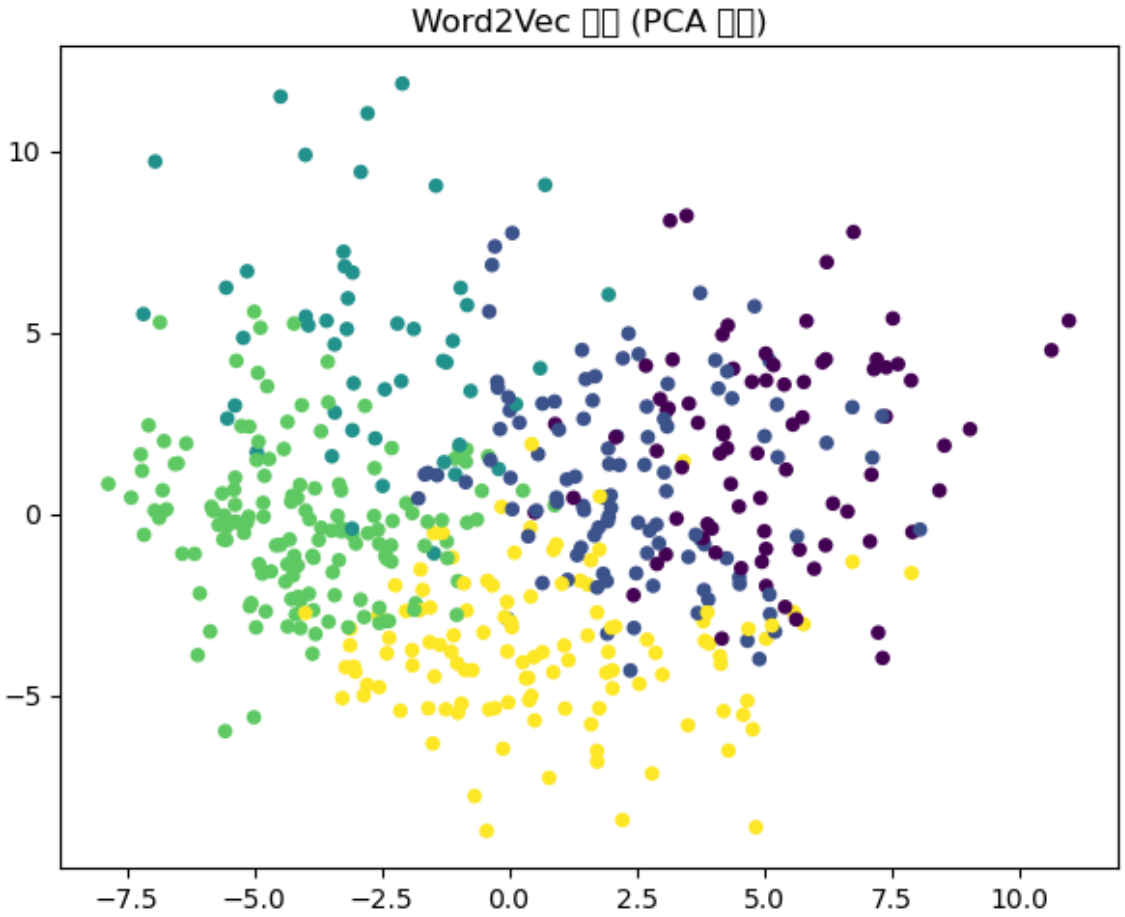

图 3：Word2Vec聚类效果

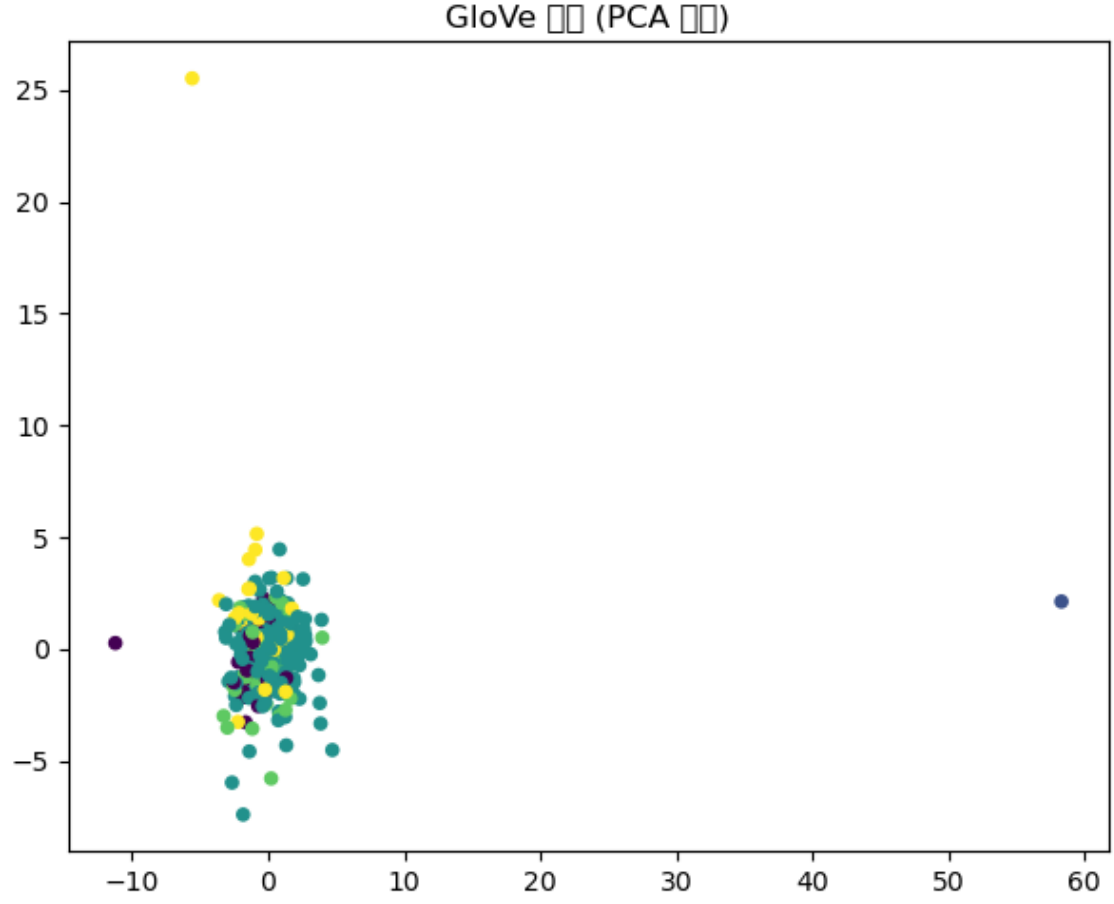

图 4：GloVe聚类效果

由实验结果可以看出，在本实验中，我们主要从词对余弦相似度比较和聚类分析两个方面对比了 Word2Vec 与基loTorch实现的 GloVe 模型的词向量

效果。通过计算测试词对的余弦相似度，我们发现两种模型在捕捉词语语义联系上存在明显差异。以“郭靖”与“黄蓉”为例：**Word2Vec 模型**计算出的相似度约为 0.87，这表明该模型能够较好地捕获这对词在局部上下文中的相似性；而**GloVe 模型**在同一词对上的相似度仅约 0.06，显示出两者在词向量分布上存在较大不同。

这种差异可能归因于两种模型的训练机制：Word2Vec 更倾向于利用局部上下文信息，而 GloVe 则依赖于全局共现矩阵。在实际应用中，不同的语义需求可能会对两者的表现提出不同的要求。

针对前 500 个高频词的词向量，我们利用 KMeans 进行了聚类分析，并通过计算轮廓系数评估聚类效果：**Word2Vec** 模型的聚类轮廓系数为 0.0297，表明该模型的词向量在聚类空间中较为分散；**GloVe** 模型则获得轮廓系数 0.0636，显示其词向量在聚类时呈现出更高的紧凑性和区分性。

同时，利用 PCA 将高维词向量降维到二维空间后绘制的散点图也直观地反映了这一现象，在散点图中，可以观察到 GloVe 模型的词向量在某些区域内呈现明显聚类趋势，而 Word2Vec 模型的词向量分布则较为分散。

综上所述：

- **Word2Vec 模型**在局部上下文中的相似度捕捉方面表现较好，但在全局词向量聚类效果上存在一定不足；
- **GloVe 模型**则依托全局共现信息，虽在部分测试词对上显示较低的相似度，但其词向量在聚类任务中表现出更好的紧凑性；

References

[1] Zenchang Qin and Lao Wang (2023)，How to learn deep learning? Journal of Paper Writing, Vol. 3: 23: pp. 1-12.

[2] LDA主题模型及Python实现_python lda-CSDN博客