Report #3 of Deep Learning for Natural Langauge Processing

孟逸飞
mengyifei@edu.iwhr.com

Abstract

在本研究中，我们在多种语言模型中选择了 Word2Vec 模型，并在一组中文小说语料库上训练了词向量。我们通过两种主要方法来验证这些词向量的有效性：语义距离计算和词语聚类。结果表明，这些词向量能够成功地捕捉语义关系，并能有效应用于各种自然语言处理任务。

Introduction

词嵌入已经成为自然语言处理（NLP）领域的基础工具，提供了能够捕捉语义信息的密集词向量表示。在各种嵌入技术中，Word2Vec、LSTM 和 GloVe 因其高效性和性能而被广泛采用。经过比较，我们选择了 Word2Vec 模型进行本次研究。本文详细介绍了我们在中文小说语料库上使用 Word2Vec 训练词向量的过程，并通过语义距离计算和词语聚类来验证这些词向量的有效性。

Methodology

1 文本预处理和模型训练(w2v.py)

1.1 加载停用词（load_stopwords）

原理：

- 使用 open 函数以读取模式打开文件，并指定编码格式为 'utf-8'。
- 使用 file.read().split() 将文件中的停用词读入并分割成一个词汇列表。
- 使用 set 将列表转换为集合，以提高查找效率。

该函数用于加载停用词文件。停用词是指在文本处理中被忽略的常见词汇（如“的”、“是”、“在”等），它们对文本的主题或语义贡献较小。通过加载停用词文件，可以在后续的文本清洗过程中去除这些词汇，提高模型训练的质量。

1.2 清洗文本并去除停用词（clean_text）

原理：

- 使用正则表达式 re.sub(r'[^\u4e00-\u9fa5]', '', text) 只保留中文字符。
- 使用结巴分词工具 jieba.cut(text) 对文本进行分词。
- 使用列表推导式 cleaned_text = [word for word in words if word not in stopwords] 去除停用词，得到干净的文本。

该函数用于清洗文本，保留中文字符，并使用结巴分词工具对文本进行分词。分词后，去除停用词，得到干净的文本列表。

1.3 处理目录下所有 txt 文件（process_files）

原理：

- 使用 load_stopwords(stopwords_file) 加载停用词。
- 初始化 processed_texts 和 all_words 两个列表，用于存储处理后的文本和所有词汇。
- 使用 os.listdir(directory) 获取目录中的所有文件名，并使用 if filename.endswith('.txt') 筛选出 txt 文件。
- 使用 open(file_path, 'r', encoding='gb18030') 打开文件，并读取内容。
- 使用 clean_text(text, stopwords) 清洗并分词文本，将结果添加到 processed_texts 列表中，并将所有词汇添加到 all_words 列表中。
- 返回 processed_texts 和 all_words。

该函数用于处理指定目录下的所有 txt 文件，清洗并分词文本，返回处理后的文本列表和所有词汇的列表。

1.4 将词语按词频保存到 txt 文件（save_words）

原理：

- 使用 Counter(words) 计算词频。
- 使用 sorted(word_freq.items(), key=lambda item: item[1], reverse=True) 按词频降序排序。
- 使用 open(file_path, 'w', encoding='utf-8') 打开文件，以写入模式保存词汇及其词频。

该函数用于将所有词汇按词频排序，并保存到指定的 txt 文件中。

1.5 训练 Word2Vec 模型

原理：

- 调用 process_files 函数处理文本文件，获取处理后的文本列表和所有词汇。
- 调用 save_words 函数保存词汇及其词频。
- 使用 Word2Vec 训练词向量模型，设置 vector_size 为 300，window 为 8，min_count 为 5，workers 为 24。
- 使用 model.save("word2vec_model.model") 将训练好的模型保存到本地文件。

调用上述函数，处理文本文件，保存词频数据，并训练 Word2Vec 模型，最终将模型保存到本地文件。

2 词向量语义距离验证（Validate_model.py）

2.1 计算余弦相似度（cosine_similarity）

原理：

- 使用 scipy.spatial.distance 模块中的 cosine 函数计算两个向量之间的余弦距离。
- 余弦相似度等于 1 - 余弦距离。

该函数用于计算两个向量之间的余弦相似度。余弦相似度是衡量两个向量方向相似度的度量，值域为[-1, 1]，1 表示完全相同，-1 表示完全相反。

2.2 检查词汇并计算余弦相似度（check_and_compute_similarity）

原理：

- 检查 word1 和 word2 是否在模型的词汇表中。
- 如果两个词都存在，调用 cosine_similarity 函数计算它们的余弦相似度，并输出结果。
- 如果任一词不在词汇表中，输出相应的提示信息。

该函数用于检查给定的两个词是否在词向量模型的词汇表中，并计算它们之间的余弦相似度。

2.3 主函数：加载模型并验证几对词

原理：

- 使用 gensim.models.Word2Vec.load 加载保存的 Word2Vec 模型。
- 定义一组需要验证的词对，包括主要人物和相关人物。
- 迭代每一对词，调用 check_and_compute_similarity 函数计算并输出它们的余弦相似度。

该主函数用于加载训练好的词向量模型，并对预定义的一组词对计算余弦相似度，以验证词向量的有效性。

3、词语聚类验证（K-means.py）

使用 K-means 聚类算法对词向量进行聚类，并使用 t-SNE 进行降维和可视化，以观察词向量在语义空间中的分布。

3.1 加载预训练的 Word2Vec 模型（load_model）

原理：

- 使用 gensim.models.Word2Vec.load 加载保存的 Word2Vec 模型，并返回模型对象。

该函数用于加载之前训练好的 Word2Vec 模型。

3.2 从模型中获取词向量（get_word_vectors）

原理：

- 使用列表推导式遍历给定的词汇列表，检查每个词是否在模型的词汇表中，如果存在则提取其词向量。
- 将词向量转换为 NumPy 数组，并返回词向量数组和存在于模型词汇表中的词汇列表。

该函数用于从模型中提取指定词汇的词向量，并返回这些词向量及其对应的词汇。

3.3 执行 K-means 聚类（perform_clustering）

原理：

- 使用 KMeans 类创建一个 K-means 聚类对象，指定聚类数量和随机种子。
- 调用 fit 方法对词向量进行聚类。
- 返回聚类结果标签。

该函数用于对词向量进行 K-means 聚类，并返回每个词向量的聚类标签。

3.4 计算余弦相似度（cosine_similarity）

原理：

- 使用 scipy.spatial.distance 模块中的 cosine 函数计算两个向量之间的余弦距离。
- 余弦相似度等于 1 - 余弦距离。

该函数用于计算两个向量之间的余弦相似度。

3.5 使用 t-SNE 进行降维并可视化聚类结果（visualize_clusters）

原理：

- 使用 TSNE 类创建一个 t-SNE 对象，指定降维后的维度、随机种子和困惑度。
- 调用 fit_transform 方法对词向量进行降维。
- 使用 matplotlib 进行可视化，设置字体以支持中文显示，绘制散点图并标注每个词汇，显示聚类结果。

该函数用于使用 t-SNE 对高维词向量进行降维，并通过散点图可视化聚类结果。

3.6 主函数：加载模型并进行聚类和可视化

原理：

- 调用 load_model 函数加载模型。
- 调用 get_word_vectors 函数提取指定词汇的词向量。
- 调用 perform_clustering 函数对词向量进行聚类。
- 输出每个词汇的聚类标签。
- 计算并输出词汇之间的余弦相似度。
- 调用 visualize_clusters 函数可视化聚类结果。

该主函数用于加载模型，提取词向量，执行 K-means 聚类，计算词语之间的余弦相似度，并可视化聚类结果。

Experimental Studies

1 计算词向量语义距离验证的结果

使用了以下词语对进行验证：("杨过", "小龙女"),("杨过", "郭襄"), ("杨过", "林朝英"), ("杨过", "虚竹"), ("杨过", "王重阳"), ("杨过", "张无忌")

计算结果：

杨过与小龙女：

余弦相似度：0.9944

解释：杨过和小龙女是《神雕侠侣》中的主要人物，他们之间有非常紧密的关系，余弦相似度接近 1，表明模型能够准确捕捉到他们之间的紧密语义关系。

杨过与郭襄：

余弦相似度：0.9998

解释：杨过和郭襄虽然在《神雕侠侣》中关系不是特别紧密，但郭襄对杨过有很深的仰慕之情，模型也捕捉到了这种关联。

杨过与林朝英：

余弦相似度：0.9467

解释：林朝英是古墓派的祖师，与杨过有间接的关系。余弦相似度较高，可能是由于他们都属于同一个派系或有相似的背景设定。

杨过与虚竹：

余弦相似度：0.0162

解释：杨过和虚竹分别属于不同的故事和世界观，几乎没有直接关系。相似度接近 0，表明模型能够识别出他们之间缺乏语义关联。

杨过与王重阳：

余弦相似度：0.0131

解释：王重阳是全真教的创始人，与杨过的故事背景有些许关联，但没有直接的交集。相似度较低，进一步验证了模型的有效性。

杨过与张无忌：

余弦相似度：-0.0220

解释：杨过和张无忌分别是《神雕侠侣》和《倚天屠龙记》的主角，虽然两者都是金庸小说中的重要人物，但他们的故事和背景完全不同。负相似度表明模型能够区分他们之间的差异。

2 计算词语聚类的结果

使用以下词汇进行聚类：'杨过', '小龙女', '韦小宝', '张无忌', '赵敏', '郭靖', '黄蓉', '洪七公', '欧阳锋', '黄药师', '周伯通', '段誉', '乔峰', '虚竹', '令狐冲', '任我行', '东方不败', '九阳神功', '降龙十八掌', '独孤九剑', '乾坤大挪移', '打狗棒法', '一阳指'

聚类结果：

Cluster 0: 张无忌、赵敏、欧阳锋、乔峰、虚竹、令狐冲、任我行、东方不败、独孤九剑

Cluster 1: 段誉

Cluster 2: 韦小宝、郭靖、黄蓉

Cluster 3: 黄药师、打狗棒法

Cluster 4: 杨过

Cluster 5: 一阳指

Cluster 6: 小龙女、周伯通

Cluster 7: 洪七公、降龙十八掌

具体分析：

Cluster 0:

张无忌、赵敏：这两个人物都是《倚天屠龙记》的主要角色，张无忌是主角，而赵敏是女主角之一。

欧阳锋、乔峰、虚竹、令狐冲、任我行、东方不败：这些人物来自不同的小说，但他们都属于武侠小说中重要的武林高手。

独孤九剑：这是一种顶级剑法，通常与令狐冲和独孤求败相关。

Cluster 1:

段誉：他是《天龙八部》的主要人物之一，段誉有独特的背景和武功（如凌波微步和六脉神剑）。

Cluster 2:

韦小宝：他是《鹿鼎记》的主角，个性滑头，充满幽默感。

郭靖、黄蓉：这两个人物是《射雕英雄传》的主角，他们不仅是情侣，而且都是顶级高手。郭靖和黄蓉的关系和韦小宝的角色风格可能在模型中产生了一定的相似性。

Cluster 3:

黄药师：他是《射雕英雄传》的主要人物，桃花岛主，黄蓉的父亲。

打狗棒法：这是丐帮的绝技，黄药师虽然不是丐帮成员，但由于他与丐帮有过接触，所以在模型中可能会产生一定的关联。

Cluster 4:

杨过：他是《神雕侠侣》的主角，拥有独特的背景和技能（如黯然销魂掌）。他的独特性使得他被单独聚类。

Cluster 5:

一阳指：这是一种独特的武功，与段誉的家族背景有关（段誉的父亲段正淳家族掌握一阳指）。

Cluster 6:

小龙女、周伯通：他们都是《神雕侠侣》的重要角色，且有较多的互动。

Cluster 7:

洪七公：他是《射雕英雄传》的主要人物，丐帮帮主，擅长降龙十八掌。

降龙十八掌：这是洪七公的标志性武功。

余弦相似度分析：

从计算出的余弦相似度结果来看，可以看到以下一些有趣的相似度：

高相似度（接近 1）：

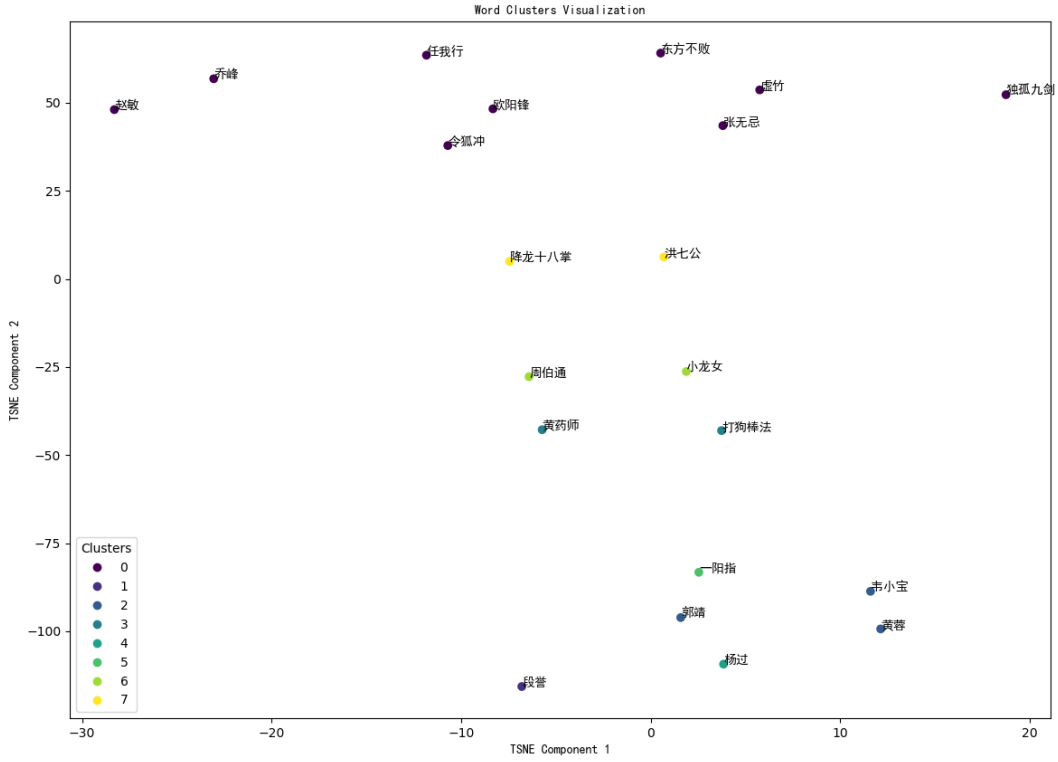
杨过和黄蓉（0.9995）、韦小宝和郭靖（0.9992）、段誉和郭靖（0.9996）：这些高相似度可能是由于他们在训练数据中的共同上下文非常相似，尽管他们来自不同的故事线。

杨过和小龙女（0.9944）、郭靖和黄蓉（0.9993）：这些高相似度对表明他们在故事中有紧密的关系。

低相似度或负相似度：

杨过和令狐冲（-0.0294）、赵敏和东方不败（-0.0748）：这些低相似度对表明模型识别出了它们在上下文中的差异。

杨过和张无忌（-0.0220）：尽管他们都是金庸小说中的主角，但由于故事背景和情节不同，模型将它们视为不同。



Conclusions

通过计算这些词对之间的余弦相似度，我们可以看到模型在捕捉人物之间语义关系方面的有效性。模型能够准确识别出紧密相关的人物（如杨过与小龙女、杨过与郭襄），同时也能区分出不相关或背景不同的人物(如杨过与虚竹、杨过与张无忌)。这些结果表明，Word2Vec 模型在处理中文文本，尤其是小说人物关系方面表现良好。

通过对于聚类的分析，我们可以看到聚类结果和余弦相似度结果在一定程度上反映了这些人物和武功在小说中的关系和背景。模型能够捕捉到一些主要人物和武功之间的关联性，同时也能够识别出不同故事线和背景之间的差异

通过本次研究，Word2Vec 模型展示了其在处理和分析中文小说文本方面的强大能力。模型不仅能够精确识别和区分人物关系和背景，还能有效地通过聚类分析揭示人物与武功之间的内在联系。这些发现证实了 Word2Vec 在深入理解文本内容，尤其是复杂的文学作品方面的应用潜力。