

学 期 2022-2023

****

深度学习与自然语言处理

词向量模型与类聚

|  |  |
| --- | --- |
| 院（系）名称 | 自动化科学与电气工程学院 |
| 专业名称 | 电子信息 |
| 学生姓名 | 苏士鹏 |
| 学号 | ZY2103306 |

2022年5月

目录

[一，作业要求 3](#_Toc103975293)

[二，实验背景 3](#_Toc103975294)

[2.1，词向量 3](#_Toc103975295)

[2.2，Word2Vec模型 3](#_Toc103975296)

[三，实验过程 4](#_Toc103975297)

[3.1预处理 4](#_Toc103975298)

[3.2模型训练 4](#_Toc103975299)

[3.3聚类分析 5](#_Toc103975300)

[四，实验结果 5](#_Toc103975301)

[附录 7](#_Toc103975302)

# 一，作业要求

利用给定语料库（或者自选语料库），利用神经语言模型（如：Word2Vec， GloVe等模型）来训练词向量，通过对词向量的聚类或者其他方法来验证词向量的有效性。

# 二，实验背景

## 2.1，词向量

词向量（Word embedding），又叫Word嵌入式自然语言处理（NLP）中的一组语言建模和特征学习技术的统称，其中来自词汇表的单词或短语被映射到实数的向量。 从概念上讲，它涉及从每个单词一维的空间到具有更低维度的连续向量空间的数学嵌入。而无论是是机器学习还是深度学习在处理不同任务时都需要对对象进行向量化表示，词向量所做的事就是将词表中的单词映射为实数向量。而在单词映射时，常见的编码主要有One-Hot Representation 和 Distributed Representation。

其中One-Hot编码是一种简单的词向量编码方式，就是用一个长向量表示一个词，向量的长度为词典的大小，向量中只有一个 1 ，其他全为 0 ,1 的位置对应该词在词典中的位置而Distributed Representation最早是Hinton于1986年提出的，其基本想法是：通过训练将某种语言中的每一个词映射成一个固定长度的短向量所有这些向量构成一个词向量空间，而每一个向量则可视为 该空间中的一个点，在这个空间上引入“距离”，就可以根据词之间的距离来判断它们之间的语法、语意的相似性。

## 2.2，Word2Vec模型

关于Word2Vec模型的概念，其是一群用来产生词向量的相关模型。这些模型为浅而双层的神经网络，用来训练以重新建构语言学之词文本。网络以词表现，并且需猜测相邻位置的输入词，在word2vec中词袋模型假设下，词的顺序是不重要的。训练完成之后，word2vec模型可用来映射每个词到一个向量，用来表示词对词之间的关系。其采用的就是上一小节中的Distributed Representation的词向量。

结构上来说Word2Vec是轻量级的神经网络，其模型仅仅包括输入层、隐藏层和输出层，模型框架根据输入输出的不同，主要包括CBOW和Skip-gram模型，如图2.1所示CBOW主要方式为知道词的上下文的情况预测当前词，.而Skip-gram主要为在知道了词的情况下对词的上下文进行预测。本次实验主要采用CBOW模型。

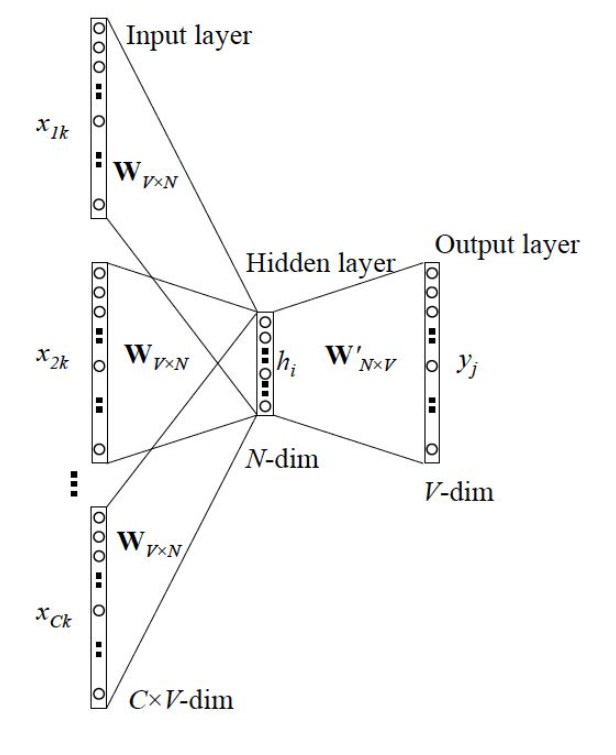
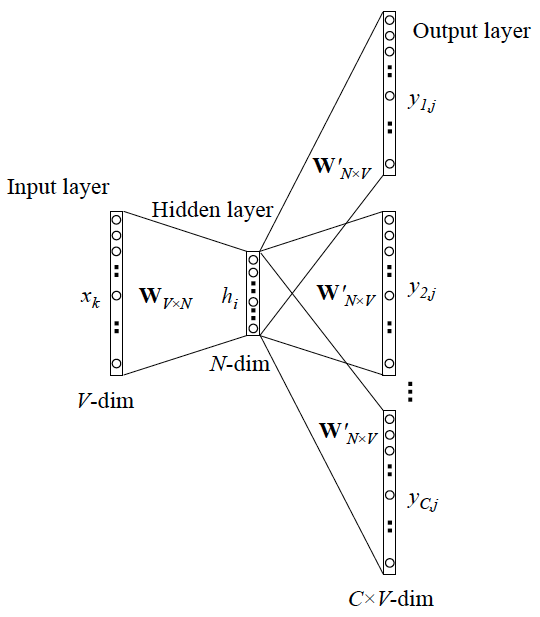
 

图2.1 CBOW，Skip-gram网络结构图

# 三，实验过程

## 3.1预处理

和前三次实验一样，第一步都是预处理，不同的是，本次实验为了能够让结果更加准确在预处理时额外删除了类似‘他’，‘她’等词，具体程序见附录1，将处理后的结果输出到一个新的文件夹中。

## 3.2模型训练

采用Gensim库提供的Word2vec模型做训练，根据python手册可得到函数的输入参数，根据需要的输入参数依次输入参数并开始训练模型，得到模型的输出。代码见附录一。

* sentence：语料词，是一个可迭代对象，本实验由LineSentence函数得到
* min\_counts：指定了需要训练的词语最小出现次数，本实验为10
* max\_vocab\_size：最大词汇数，防止内存溢出，本实验取默认值
* size：词向量维度为200
* window：滑动窗口大小取5
* sg：训练模型（0：CBOW；1：skip-gram）0采用CBOW模型
* hs：word2vec两个解法的选择，为0， 则是Negative Sampling，为1并且负采样个数negative大于0， 则是Hierarchical Softmax。本实验取1
* epochs：迭代次数本实验取100

## 3.3聚类分析

最后做聚类分析，为了能够更好的看模型训练效果，这里我采用我最了解的5本小说，分别是射雕三部曲和倚天屠龙记，笑傲江湖，通过对主角名和门派名做类聚得到结果以此验证模型训练的好坏。具体代码见附录1.

# 四，实验结果

首先是倚天屠龙记，主角没有争议就是张无忌，门派出现的最多的就是武当和明教，这里选用明教，结果储存到倚天屠龙记.txt中，如表4.1所示，与张无忌联系最大的是周芷若，这很正常因为赵敏出场较晚而周芷若从张无忌小时候就和他一起，所以联系性高一点而张翠山是因为每次提到张无忌时都会说他是张翠山之子，后面更是他的义父，母亲，而对于明教，在小说里明教中人大多叫本教其他门派叫魔教，可以看到模型较为准确。

表4.1倚天屠龙记关联结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 张无忌 | 关联度 | 明教 | 关联度 |
| 周芷若 | 0.4633382260799408 | 本教 | 0.43261751532554626 |
| 张翠山 | 0.4331585168838501 | 魔教 | 0.3609812557697296 |
| 赵敏 | 0.4283623695373535 | 周颠 | 0.24741247296333313 |
| 谢逊 | 0.3890618681907654 | 阳 | 0.23926053941249847 |
| 殷素素 | 0.38561564683914185 | 统率 | 0.23037508130073547 |
| 鹿杖客 | 0.374381422996521 | 兵法 | 0.2299850732088089 |
| 殷离 | 0.3645256459712982 | 掌力 | 0.2278500348329544 |
| 宋青书 | 0.3558835983276367 | 有个 | 0.2245110422372818 |
| 师父 | 0.350943386554718 | 曾阿牛 | 0.22397123277187347 |
| 周颠 | 0.3344482183456421 | 汝阳王 | 0.2207714170217514 |

下面是天龙八部，主角为乔峰，门派选用出现最多的逍遥派，最后结果储存在天龙八部.txt中，如表4.2所示，其中萧峰为主角真名，联系度最高，后面则是他的结拜兄弟，对于逍遥派整个文章联系最紧的就是虚竹，其是无崖子弟子同时和童姥联系紧密并且获得了掌门人戒指，所以通过结果看出模型准确。

|  |  |  |  |
| --- | --- | --- | --- |
| 乔峰 | 关联度 | 逍遥派 | 关联度 |
| 萧峰 | 0.519851565361023 | 虚竹 | 0.3801710307598114 |
| 段誉 | 0.505978524684906 | 本门 | 0.3582545816898346 |
| 游坦之 | 0.49938711524009705 | 小僧 | 0.3479800820350647 |
| 段正淳 | 0.49321192502975464 | 师父 | 0.32823362946510315 |
| 慕容复 | 0.4404674768447876 | 段誉 | 0.30930206179618835 |
| 包不同 | 0.39711686968803406 | 苏星河 | 0.30487072467803955 |
| 阿朱 | 0.3905782401561737 | 谭 | 0.2863919734954834 |
| 虚竹 | 0.3869171440601349 | 无崖子 | 0.2818996012210846 |
| 全冠清 | 0.38075491786003113 | 目下 | 0.2725520431995392 |
| 王语嫣 | 0.37298399209976196 | 童姥 | 0.27074313163757324 |

后面的小说不在一一列举，分别放到神雕侠侣.txt，射雕英雄传.txt，笑傲江湖.txt中，从分析结果可以看出，结果都是基本符合认知的，本次实验采用Word2Vec模型对词向量做训练，根据训练得到的模型对五本小说的主角和门派名进行了聚类，最后的结果基本与认知相符，但在观察整个结果时发现，较早的与主角建立联系的往往关联度较高，这可能是因为较早的与主角联系后面就有更大与晚出场的和主角有联系的人产生联系所以加高了概率，以后的学习中会更多留意这一方面。

# 附录

import jieba

import os

import re

from gensim.models import Word2Vec

from gensim.models.word2vec import LineSentence

###########################################################################

# 本程序参考https://blog.csdn.net/shzx\_55733/article/details/116768740?spm=1001.2014.3001.5501

###########################################################################

def read\_novel(path\_in, path\_out): # 读取语料库，和前三次作业不同的是，同时将处理结果输出到另一文件夹中

content = []

names = os.listdir(path\_in)

for name in names:

novel\_name = path\_in + '\\' + name

fenci\_name = path\_out + '\\' + name

for line in open(novel\_name, 'r', encoding='ANSI'):

line.strip('\n')

line = re.sub("[A-Za-z0-9\：\·\—\，\。\“\”\\n \《\》\！\？\、\...]", "", line)

line = content\_deal(line)

con = jieba.cut(line, cut\_all=False) # jieba分词，与作业一方法相同

content.append(" ".join(con))

with open(fenci\_name, "w", encoding='utf-8') as f:

f.writelines(content) #将处理结果写入新的文件夹

return content, names

def content\_deal(content): # 语料预处理，这里与前面相同，不过为了结果更精确在本次实验，多删去一些类似他，我等这些无意义的词

ad = ['本书来自www.cr173.com免费txt小说下载站\n更多更新免费电子书请关注www.cr173.com', '----〖新语丝电子文库(www.xys.org)〗', '新语丝电子文库',

'\u3000', '\n', '。', '？', '！', '，', '；', '：', '、', '《', '》', '“', '”', '‘', '’', '［', '］', '....', '......',

'『', '』', '（', '）', '…', '「', '」', '\ue41b', '＜', '＞', '+', '\x1a', '\ue42b', '她', '他', '你', '我', '它', '这']

for a in ad:

content = content.replace(a, '')

return content

if \_\_name\_\_ == '\_\_main\_\_': ##主函数开始

[data\_txt, files] = read\_novel("dataset", "output") #读取给定语料库，注意，这里只留下了5部小说

test\_name = ['张无忌', '乔峰', '郭靖', '杨过', '令狐冲'] #列出这几本小说主角名

test\_menpai = ['明教', '逍遥派', '全真教', '全真教', '华山派'] #列出各个小说里面出现的主要门派

for i in range(0, 5): ##这里面分别对5本小说做训练并输出结果

name = "output/" + files[i]

print(name) ##输出本次处理的小说名

model = Word2Vec(sentences=LineSentence(name), hs=1, min\_count=10, window=5, vector\_size=200, sg=0, epochs=200)

for result in model.wv.similar\_by\_word(test\_name[i], topn=10):

print(result[0], result[1]) ##输出与主角名联系的关联系数，输出前10

for result in model.wv.similar\_by\_word(test\_menpai[i], topn=10):

print(result[0], result[1]) ##输出与门派名联系的关联系数，输出前10