北京航空航天大學

BEIHANG UNIVERSITY

深度学习与自然语言处理

作业4

院（系）名称 自动化科学与电气工程学院

学生姓名 潘翔

学生学号 ZY2103707

2022 年 5 月

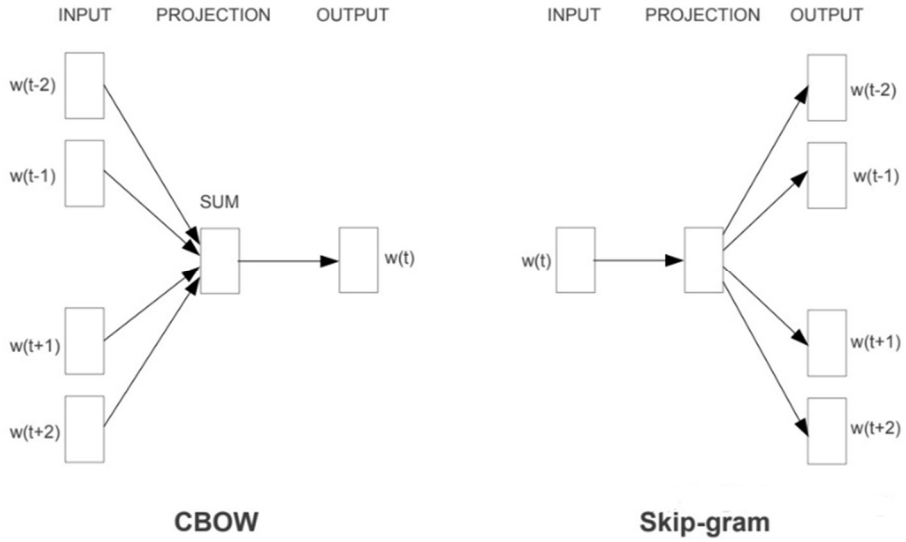
一、问题描述

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。 截至日期：5 月 20 日晚 12 点前

二、试验原理

2.1 Word2Vec

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。CBOW 的方式是在知道词 ω_t 的上下文 $\omega_{t-2},\omega_{t-1},\omega_{t+1},\omega_{t+2}$ 进行预测，如下图所示



2.2 Skip-gram

跳字模型的概念是在每一次迭代中都取一个词作为中心词汇，尝试去预测它一定范围内的上下文词汇。

所以这个模型定义了一个概率分布：给定一个中心词，某个单词在它上下文中出现的概率。我们会选取词汇的向量表示，从而让概率分布值最大化。重要的是，这个模型对于一个词汇，有且只有一个概率分布，这个概率分布就是输出，也就是出现在中心词周围上下词的一个输出。

2.3 CBOW

连续词袋模型与跳字模型类似，最大不同在于连续词袋模型假设基于某中心词在文本序后的背景词来生成该中心词。

例如：‘我’，‘爱’，‘红色’，‘这片’，‘土地’，窗口大小为 2，就是用‘我’，‘爱’，‘这片’，‘土地’这四个背景词，来预测生成 ‘红色’ 这个中心词的条件概率，即：

P（红色|我，爱，这片，土地）

给定一个长度为 T 的文本序列，设时间步 t 的词为 W(t),背景窗口大小为 m。

则连续词袋模型的目标函数（损失函数）是由背景词生成任一中心词的概率。

$$\prod_{t=1}^{T} P\left(w^{(t)} \mid w^{(t-m)},\ldots,w^{(t-1)},w^{(t+1)},\ldots,w^{(t+m)}\right)$$

因为连续词袋模型的背景词有多个，我们将这些背景词向量取平均，然后使用和跳字模型一样的方法来计算条件概率。

三、试验过程和结果

选取的小说为金庸的十六本小说，分别为《白马啸西风》,《碧血剑》,《飞狐外传》,《连城诀》,,《鹿鼎记》,《三十三剑客图》,《射雕英雄传》,《神雕侠侣》,《书剑恩仇录》,《天龙八部》,《侠客行》,《笑傲江湖》,《雪山飞狐》,《倚天屠龙记》,《鸳鸯刀》,《越女剑》，以及科幻小说《三体》

和前面试验的处理方法一致，先对小说文本进行处理，对特殊符号进行删除，利用 jieba 库进行词分类相关操作，由于前面以及进行过多次，所以不过多赘述。

代码如下

```
outfilename_1 = "./train_jieba.txt"
if not os.path.exists('./train_jieba.txt'):
    outputs = open(outfilename_1, 'w', encoding='UTF-8')
    datasets_root = "./datasets1"
    catalog = "inf.txt"

    test_num = 10
    test_length = 20
    with open(os.path.join(datasets_root, catalog), "r", encoding='utf-8') as f:
        all_files = f.readline().split(",")
        print(all_files)

    for name in all_files:
        with open(os.path.join(datasets_root, name + ".txt"), "r", encoding='utf-8') as f:
            file_read = f.readlines()
            train_num = len(file_read) - test_num
            choice_index = np.random.choice(len(file_read), test_num + train_num, replace=False)
            train_text = ""
            for train in choice_index[0:train_num]:
                line = file_read[train]
                line = re.sub('\s', '', line)
                line = re.sub('[\u0000-\u4DFF]', '', line)
                line = re.sub('[\u9FA6-\uFFFF]', '', line)
                if len(line) == 0:
                    continue
                seg_list = list(jieba.cut(line, cut_all=False))  # 使用精确模式
                line_seg = ""
                for term in seg_list:
                    line_seg += term + " "
                    # for index in range len(line_seg):
                outputs.write(line_seg.strip() + '\n')
    outputs.close()
    print("得到训练集！！！")
```

利用 Word2Vec 模型进行训练

由于 gensim 库里面已经封装好，因此直接使用即可，word2vec 的主要参数为：

sentences：可以是一个 list，对于大语料集，建议使用 BrownCorpus,Text8Corpus 或 lineSentence 构建。

size：是指词向量的维度，默认为 100。这个维度的取值一般与我们的语料的大小相关，如果是不大的语料，比如小于 100M 的文本语料，则使用默认值一般就可以了。如果是超大的语料，建议增大维度。大的 size 需要更多的训练数据,但是

效果会更好. 推荐值为几十到几百。

window：窗口大小，即词向量上下文最大距离，这个参数在我们的算法原理篇中标记为 c。window 越大，则和某一词较远的词也会产生上下文关系。默认值为 5。在实际使用中，可以根据实际的需求来动态调整这个 window 的大小。如果是小语料则这个值可以设的更小。对于一般的语料这个值推荐在[5,10]之间。个人理解应该是某一个中心词可能与前后多个词相关，也有的词在一句话中可能只与少量词相关（如短文本可能只与其紧邻词相关）。

min_count: 需要计算词向量的最小词频。这个值可以去掉一些很生僻的低频词，默认是 5。如果是小语料，可以调低这个值。可以对字典做截断， 词频少于 min_count 次数的单词会被丢弃掉。

negative：即使用 Negative Sampling 时负采样的个数，默认是 5。推荐在[3,10]之间。这个参数在我们的算法原理篇中标记为 neg。

cbow_mean: 仅用于 CBOW 在做投影的时候，为 0，则算法中的为上下文的词向量之和，为 1 则为上下文的词向量的平均值。在我们的原理篇中，是按照词向量的平均值来描述的。个人比较喜欢用平均值来表示，默认值也是 1，不推荐修改默认值。

iter: 随机梯度下降法中迭代的最大次数，默认是 5。对于大语料，可以增大这个值。

alpha: 是初始的学习速率，在训练过程中会线性地递减到 min_alpha。在随机梯度下降法中迭代的初始步长。算法原理篇中标记为 η，默认是 0.025。

min_alpha: 由于算法支持在迭代的过程中逐渐减小步长，min_alpha 给出了最小的迭代步长值。随机梯度下降中每轮的迭代步长可以由 iter，alpha， min_alpha 一起得出。这部分由于不是 word2vec 算法的核心内容，因此在原理篇我们没有提到。对于大语料，需要对 alpha, min_alpha,iter 一起调参，来选择合适的三个值。

max_vocab_size: 设置词向量构建期间的 RAM 限制，设置成 None 则没有限制。

sample: 高频词汇的随机降采样的配置阈值，默认为 1e-3，范围是(0,1e-5)。

seed：用于随机数发生器。与初始化词向量有关。

workers：用于控制训练的并行数。

代码如下所示

```
fr = open('./train_jieba.txt', 'r', encoding='utf-8')
train = []
for line in fr.readlines():
    line = [word.strip() for word in line.split(' ')]
    train.append(line)

num_features = 300  # Word vector dimensionality
min_word_count = 10  # Minimum word count
num_workers = 16  # Number of threads to run in parallel
context = 10  # Context window size
downsampling = 1e-3  # Downsample setting for frequent words
sentences = word2vec.Text8Corpus("train_jieba.txt")

model = word2vec.Word2Vec(sentences, workers=num_workers,
                          vector_size=num_features, min_count=min_word_count,
                          window=context, sg=1, sample=downsampling)
model.init_sims(replace=True)
# 保存模型
model.save("model_px.model")
```

训练完成后，在模型中读取和关键词最近的十个词
代码如下

```
from gensim.models import word2vec
model=word2vec.Word2Vec.load("./model_px.model")
print(f'-----乔峰的词向量为----')
print(model.wv['乔峰'])
print(f'-----张无忌的词向量为----')
print(model.wv['张无忌'])
print(f'-----郭靖的词向量为----')
print(model.wv['郭靖'])
print(f'-----罗辑----')
print(model.wv['郭靖'])
print(f'-----叶文洁----')
print(model.wv['叶文洁'])

print(model.wv.most_similar("乔峰",topn=10))
print(model.wv.most_similar("张无忌",topn=10))
print(model.wv.most_similar("郭靖",topn=10))
print(model.wv.most_similar("罗辑",topn=10))
print(model.wv.most_similar("叶文洁",topn=10))
```

提供关键词，利用得到的词向量，并且使用 KMeans 方法进行聚类分析，并由此判断 Word2Vec 训练结果的准确性。

为了便于判断训练结果的准确性，直接抽取 6 本小说的部分关键词列表如下所示，小说包括《天龙八部》,《三体》,《雪山飞狐》,《神雕侠侣》,《笑傲江湖》,《倚天屠龙记》，其中选取的关键词如下所示：

段誉，萧峰，张无忌，周芷若，宋青书，灭绝师太，赵敏，叶文洁，程心，令狐冲，任盈盈，岳不群，林平之，胡斐，袁紫衣，程灵素，东方不败，杨过，小龙女，郭靖，黄蓉，郭襄，陆无双，胡一刀，苗人凤，虚竹，段正明，段正淳，执剑人，三体，罗辑，黑暗森林

对应代码如下所示

```
model = word2vec.Word2Vec.load('./model_px.model')
names=[]
for line in open("name.txt","r",encoding='utf-8'):
    line = line.strip('\n')
    names.append(line)
names = [name for name in names if name in model.wv]
name_vectors = [model.wv[name] for name in names]

n=6
label = cluster.KMeans(n).fit(name_vectors).labels_
print(label)
print("类别1：")
for i in range(len(label)):
    if label[i]==0:
        print(names[i],end=" ")
print("\n")
print("类别2：")
for i in range(len(label)):
    if label[i]==1:
        print(names[i],end=" ")
print("\n")
print("类别3：")
for i in range(len(label)):
    if label[i]==2:
        print(names[i],end=" ")

print("\n")
print("类别4：")
for i in range(len(label)):
    if label[i]==3:
        print(names[i],end=" ")
print("\n")
print("类别5：")
for i in range(len(label)):
    if label[i]==4:
        print(names[i],end=" ")

print("\n")
print("类别6：")
for i in range(len(label)):
    if label[i]==5:
        print(names[i],end=" ")
```

3.1 对关键词进行 KMeans 聚类分析的结果如下所示

类别 1：
段誉 萧峰 虚竹 段正明 段正淳
类别 2：
叶文洁 程心 执剑人 三体 罗辑
类别 3：
杨过 小龙女 郭靖 黄蓉 郭襄 陆无双
类别 4：
令狐冲 岳不群 林平之 东方不败
类别 5：
胡斐 袁紫衣 程灵素 胡一刀 苗人凤
类别 6：
张无忌 周芷若 宋青书 灭绝师太 赵敏

根据结果可以看出，绝大多数分类结果是非常准确的，但是是因为我从六本小说中选取关键词，然后将结果分为六类，因此非常准确，如果将其分为 4 类结果如

下所示

类别 1：

叶文洁 程心 执剑人 三体 罗辑

类别 2：

张无忌 周芷若 宋青书 灭绝师太 赵敏

类别 3：

段誉 令狐冲 岳不群 林平之 东方不败 段正明 段正淳

类别 4：

萧峰 胡斐 袁紫衣 程灵素 杨过 小龙女 郭靖 黄蓉 郭襄 陆无双 胡一刀 苗人凤 虚竹

结果分析：

从两个分类的结果可以看出，根据词向量进行分类时候，基本能够将同一本小说的关键词分在一起，并且即使在分类器的分类数量小于真实类别的时候能够保证大多数关键词分类准确，仍然没有将差异很大的三体和金庸的小说错误的分在一起，此处也表现出词向量的正确性，即不同小说的关键词向量之间差异很大。

3.2 给定词的最相关的 10 个词的总结

给定乔峰，张无忌，郭靖，罗辑，叶文洁，令狐冲六个词，分别给出其词向量，结果在 mostRelative.txt 文件里面，并且给出最相近的 10 个词如下所示。

乔峰->'阿朱', '谭公', '徐长老','马夫人','单正', '智光', '全冠清', '乔帮主', '谭婆', '萧峰'

张无忌->'赵敏', '周芷若', '小昭','蛛儿', '杨不悔' ,'赵姑娘', '赵敏道', '韦一笑','灭绝师太'，'张公子'

郭靖->'黄蓉', '黄药师', '柯镇恶' , '杨康', '欧阳锋', '拖雷', '蓉','完颜洪烈', '华筝', ,'洪七公'

罗辑->'后罗辑', '乔纳森', '史强','大史','护士', '斐兹罗', '萨伊', '坎特', '柯曼琳', '艾伦'

叶文洁->'杨卫宁', '文洁', '申玉菲', '杨母', '红岸', '志成', '白沐霖', '白蓉', '政委', '斐兹罗'

令狐冲->'盈盈', '岳不群','岳灵珊', '田伯光', '仪琳', '恒山', '向问天','岳夫人', '任我行', '冲虚'

结果分析：

因为对金庸的小说不是特别熟悉，但是在查询了相关资料之后，发现尤其是前三个词在小说当中和选定关键词非常相近。比如阿朱是乔峰的一生所爱，赵敏对张无忌一往情深，郭靖和黄蓉也是有美满爱情，不过让人以外的是在相关性分析时和关键人物最相近的一般都是另外一个关键人物。在没有查阅资料之前并不知道金庸小说的具体内容，查阅之后结合相关性分析的结果，发现虽然是武侠小说，但是金庸小说里面对于情感的描述确实很多，以至于和主角最相关的人物都是他的爱人。

由于对金庸小说的不熟悉，我在训练集里面添加了三体小说，可以看出，和罗辑最接近的十个词分别为后罗辑，乔纳森（三大舰队联席会议特派员,带罗辑参与

最后一次面壁者听证会），史强（警察、地球防务安全部情报人员。在《三体 1 地球往事》中提出古筝行动，攻击“审判日”号 ，使伊文斯死亡从而截取三体世界信息。在《三体 2 黑暗森林》中负责保护罗辑，帮助罗辑找到庄颜，在危机纪元后期主动保护了罗辑），大史（史强）

在叶文洁最相关的十个词当中，杨卫宁（红岸基地总工程师，叶文洁的丈夫），申玉菲（叶文洁和申玉菲史同一派，是科学边界组织的领头人），红岸（叶文洁在红岸基地担任工程师）

而从三体的相关性分析中可以看出，虽然三体属于科幻小说，和金庸的武侠小说在小说类型上截然不同，但是在三体当中和主角最相关的仍然是其他人物。因此可以看出在这有限的十多本小说中，主要还是围绕人与人之间的故事进行展开，这也是小说的核心要素。

四、试验总结和体会

这次试验成功实现了利用 word2vec 模型对金庸的十六本小说和三体进行分析，并且利用训练出来的词向量进行分类和相关性分析，最后得出的结果发现通过词向量的分类符合不同小说之间的关系，词向量的相关性分析满足人物之间的关系。

经过这次试验，我学到了 Word2Vec 的基本原理和 gensim 库中调用 word2vec 的使用方法，进一步提升了自己对处理自然语言的一些基本方法的掌握。在进时试验过程中，因为要使用 python 进行遍程，在编写的过程中通过参考以前的代码，更加快速的理解模型，并且在写的过程中提升了自己使用 python 的熟练度，为以后做其他相关项目打下了坚实的基础。

这是深度学习与自然语言处理的第四次作业，经过这几次的作业，我深刻的体会到了自然语言处理的魅力。对于一个完全陌生的文本，我们可以利用譬如这次作业的模型对其进行分析，获取自己需要的隐含在文本当中的信息。

五、参考资料

https://zhuanlan.zhihu.com/p/114538417
https://blog.csdn.net/Jruo911/article/details/123585597