深度学习与自然语言处理第四次作业

SY2106318 孙旭东

代码链接

1.作业内容

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。

2.相关知识

2.1 词向量

在自然语言处理中最细力度是词语，词语组成句子，句子再组成段落、篇章、文档，因此最先要处理词语，需要将自然语言交给机器学习中的算法来处理。词语是人类的抽象总结，是符号形式的（比如中文、英文、拉丁文等等），而机器只能接受数值型输入，所以需要把词语转换成数值形式。词向量就是用来将语言中的词进行数学化的一种方式，顾名思义，词向量就是把一个词表示成一个向量。 我们都知道词在送到神经网络训练之前需要将其编码成数值变量，常见的编码方式有两种：One-Hot Representation 和 Distributed Representation。

2.1.1 One-Hot Representation

最简单的也最容易想到的词表示方法是 One-hot Representation，这种方法把每个词表示为一个很长的向量。向量的长度为词典的大小，向量中只有一个1，其他为0，1的位置对应词在词典中的位置。举例：我是人，转换为One-Hot编码为，我[0 0 1]，是[0 1 0]，人[1 0 0]。这种One-Hot编码如果采用稀疏方式存储，会是非常的简洁：也就是给每个词分配一个数字ID。比如上面的例子中，我记为1，人记为3。但这种表示方法有几个缺点：

1. 容易受维度灾难的困扰，当词数量达到1千万的时候,词向量的大小变成了1千万维，假设你使用一个bit来表示每一维，那么仅一个单词大概就需要0.12GB的内存；
2. 任意两个词之间都是孤立的，无法表示语义层面上词汇之间的相关信息；
3. 强稀疏性，只有一位是1，而其他位都是0，这就导致向量中有效的信息非常少。

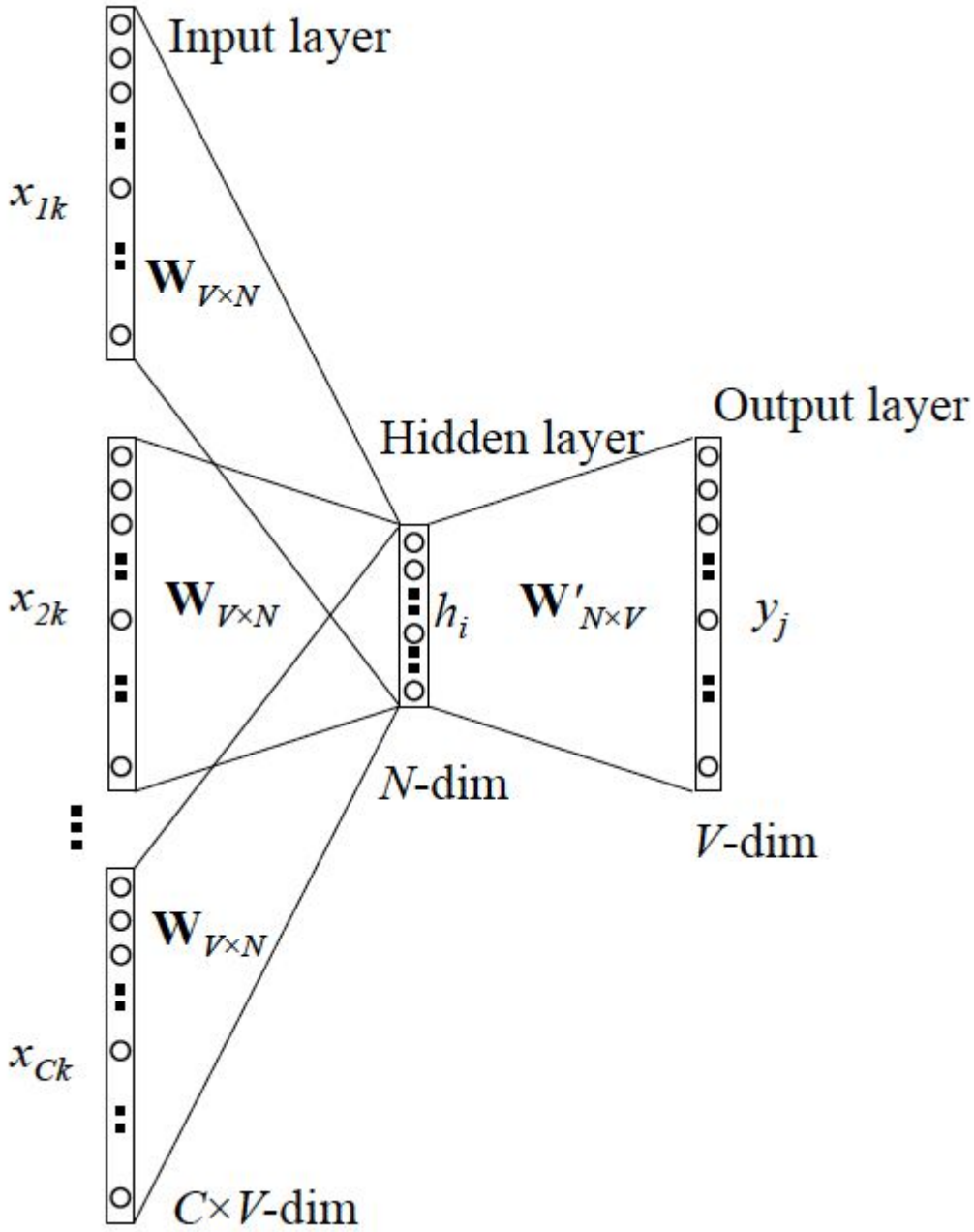
2.1.2 Distributed Representation

Distributed Representation最早是Hinton于1986年提出的，可以克服One-Hot Representation的上述缺点。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、语义上的相似性了。Word2Vec中采用的就是这种Distributed Representation的词向量。

2.2 Word2Vec

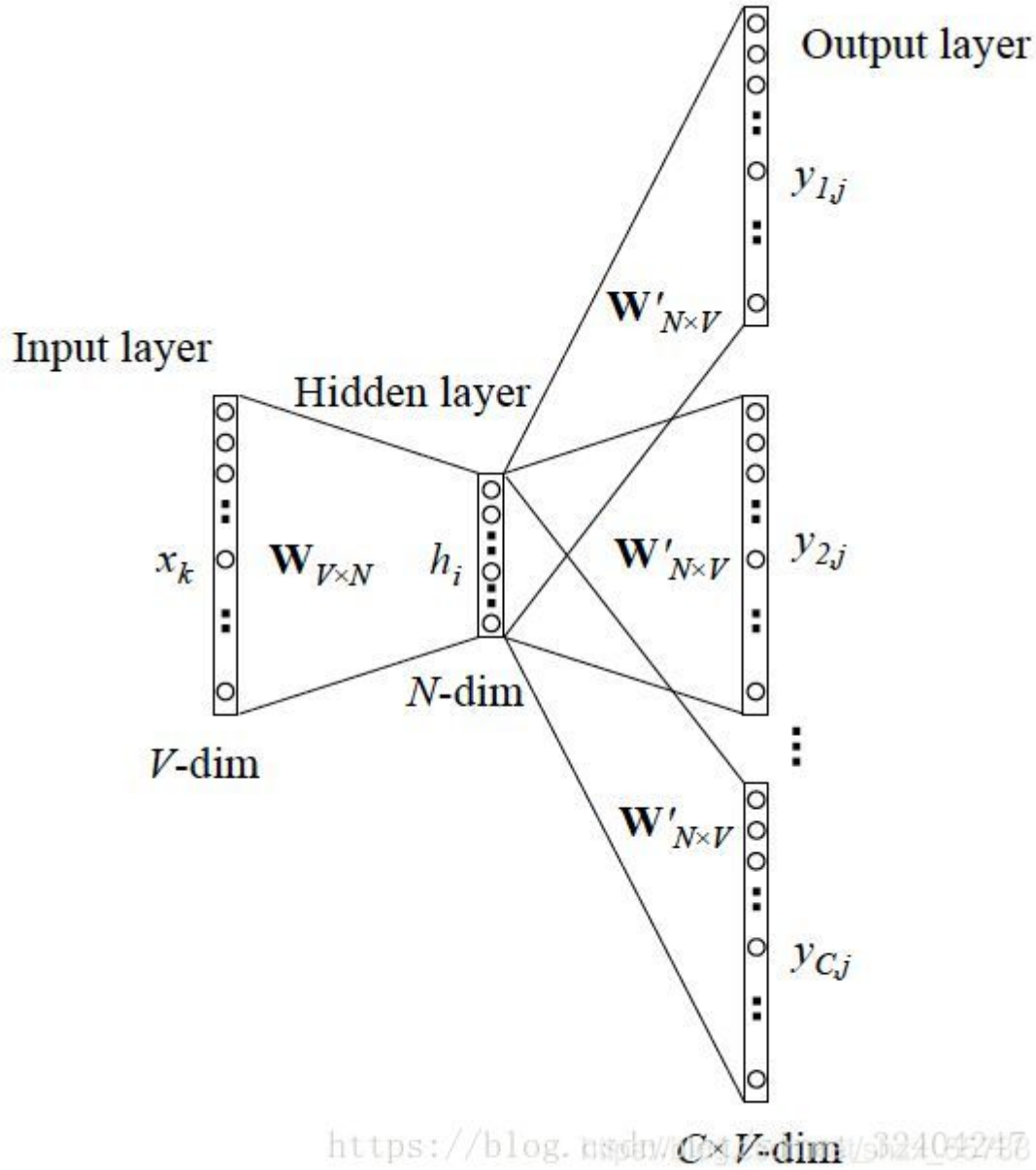
Word2Vec是轻量级的神经网络，其模型仅仅包括输入层、隐藏层和输出层，模型框架根据输入输出的不同，主要包括CBOW和Skip-gram模型。CBOW模型是在知道一个词的上下文的情况下预测当前词，而Skip-gram是在知道一个词的情况下，对该词的上下文进行预测，两种情况分别如下：

2.2.1 CBOW



输入层输入的为单词上下文的向量表示，输入层到输出层中间的隐层中间有一个权重矩阵W，隐藏层得到的值是由输入乘上权重矩阵得到的，隐藏层到输出层也有一个权重矩阵W'，最终的输出需要经过softmax函数，将输出向量中的每一个元素归一化到0-1之间的概率，概率最大的，就是预测的词。

2.2.2 Skip-gram



Skip-gram model是通过输入一个词去预测多个词的概率。与CBOW不同的是，损失函数变成了几个词损失函数的总和，但权重矩阵W'还是共享的。

3.实验过程

3.1数据准备

使用到的数据为金庸的16本武侠小说，对数据进行预处理的代码如下：

```
def get_single_corpus(file_path):
    """
    获取file_path文件对应的内容
    :return: file_path文件处理结果
    """
    corpus = ''
    # unuseful items filter
```

```
    r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@, 。?★、…【】《》？“”‘’！[\\]^_`{|}~「」『』
() ]+'
    with open('../stopwords.txt', 'r', encoding='utf8') as f:
        stop_words = [word.strip('\n') for word in f.readlines()]
        f.close()
    # print(stop_words)
    with open(file_path, 'r', encoding='ANSI') as f:
        corpus = f.read()
        corpus = re.sub(r1, '', corpus)
        corpus = corpus.replace('\n', '')
        corpus = corpus.replace('\u3000', '')
        corpus = corpus.replace('本书来自免费小说下载站更多更新免费电子书请关注', '')
        f.close()
    words = list(jieba.cut(corpus))
    return [word for word in words if word not in stop_words]
```

在以utf8编码格式读取文件内容后，删除文章内的所有非中文字符，以及和小说内容无关的片段，得到字符串形式的语料库，然后使用jieba分词进行分词，并使用**百度停用词表**进行停用词的过滤，最终返回小说的分词列表。

因为后续需要分析词语之间的相关性，因此本次实验选取了《射雕英雄传》，《神雕侠侣》，《天龙八部》，《笑傲江湖》，《倚天屠龙记》五本小说，将分词列表按照每行50词保存在txt格式文件中，方便后续使用。

3.2 训练Word2Vec模型

这里直接使用python库gensim中的Word2Vec类进行模型的训练，并选取5本小说中的代表性人物，分析训练后与该人物相关性最强的10个词。

```
test_name = ['郭靖', '杨过', '段誉', '令狐冲', '张无忌']
model = Word2Vec(sentences=PathLineSentences(DATA_PATH), hs=1, min_count=10, window=5,
                 vector_size=200, sg=0, workers=16, epochs=200)
for name in test_name:
    print(name)
    for result in model.wv.similar_by_word(name, topn=10):
        print(result[0], '{:.6f}'.format(result[1]))
```

模型的参数释义如下：

- sentences：可以是一个list，此处使用的PathLineSentences为一个文件夹下的所有文件，另外有LineSentence对单个文件生效；
- hs：如果为1则会采用hierarchica·softmax技巧。如果设置为0（defau·t），则negative sampling会被使用；
- min_count：可以对字典做截断. 词频少于min_count次数的单词会被丢弃掉, 默认值为5；
- window：表示当前词与预测词在一个句子中的最大距离是多少；
- vector_size：是指特征向量的维度，默认为100，大的size需要更多的训练数据,但是效果会更好；
- sg：用于设置训练算法，默认为0，对应CBOW算法；sg=1则采用skip-gram算法；
- workers：线程数；
- epoches：训练迭代轮数。

3.3 K-means聚类

为了进一步验证模型的有效性，使用TSNE将训练得到的模型中的词向量进行降维（方便展示效果），并使用K-means算法进行聚类。这里聚类用到的词为5本小说中的代表性人物。最终用散点图进行效果展示，部分代码如下：

```
with open('../name.txt', 'r', encoding='utf8') as f:
    names = f.readline().split(' ')
model = Word2Vec.load('model3.model')
names = [name for name in names if name in model.wv]
name_vectors = [model.wv[name] for name in names]
tsne = TSNE()
embedding = tsne.fit_transform(name_vectors)
n = 5
label = KMeans(n).fit(embedding).labels_
plt.title('kmeans聚类结果')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
for i in range(len(label)):
    if label[i] == 0:
        plt.plot(embedding[i][0], embedding[i][1], 'ro', )
    if label[i] == 1:
        plt.plot(embedding[i][0], embedding[i][1], 'go', )
    if label[i] == 2:
        plt.plot(embedding[i][0], embedding[i][1], 'yo', )
    plt.annotate(names[i], xy=(embedding[i][0], embedding[i][1]), xytext=(embedding[i]
[0]+0.1, embedding[i][1]+0.1))
plt.savefig('cluster3.png')
```

4.实验结果

4.1 相关词分析

在五本小说中各选取一名主角进行相关词语的分析，《射雕英雄传》——郭靖，《神雕侠侣》——杨过，《天龙八部》——段誉，《笑傲江湖》——令狐冲，《倚天屠龙记》——张无忌，分析结果如下图所示。

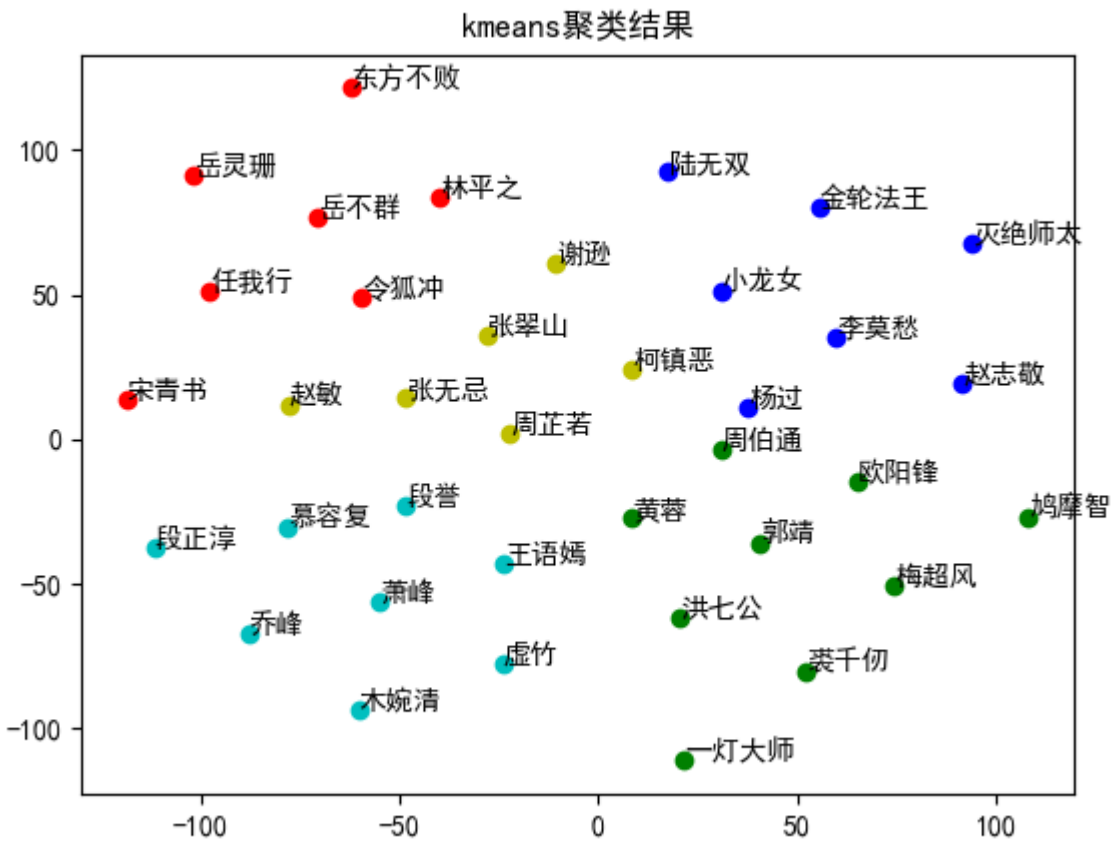
序号	郭靖		杨过		段誉		令狐冲		张无忌	
1	欧阳锋	0.696552	黄蓉	0.657014	萧峰	0.640561	岳不群	0.745456	周芷若	0.7555
2	黄药师	0.681082	小龙女	0.654537	慕容复	0.607331	林平之	0.713134	赵敏	0.680689
3	洪七公	0.658469	黄药师	0.624954	王语嫣	0.606414	岳灵珊	0.642824	张翠山	0.650916
4	黄蓉	0.657095	周伯通	0.624769	木婉清	0.578046	仪琳	0.638547	谢逊	0.630989
5	欧阳克	0.642184	李莫愁	0.609227	鸠摩智	0.572172	田伯光	0.635987	金花婆婆	0.601789
6	梅超风	0.598958	郭靖	0.593917	段正淳	0.567369	岳夫人	0.624675	令狐冲	0.596668
7	穆念慈	0.595278	欧阳锋	0.580477	乔峰	0.564037	任我行	0.611713	灭绝师太	0.573908
8	杨过	0.593917	赵志敬	0.571746	虚竹	0.554444	盈盈	0.611354	蛛儿	0.567661
9	柯镇恶	0.575538	柯镇恶	0.56456	段公子	0.539029	张无忌	0.596668	周颠	0.560676
10	小龙女	0.574317	洪七公	0.559716	游坦之	0.519155	劳德诺	0.5613	殷素素	0.560587

根据结果可以看出，词向量相似度较高的词在小说中也有一定关系，以令狐冲为例，岳不群是令狐冲的师傅；林平之是令狐冲的同门师兄弟；岳灵珊是岳不群的女儿，从小和令狐冲一起长大，是令狐冲的师妹；仪琳喜欢令狐冲……其他人也和令狐冲有一定关系，但排在第9的张无忌和令狐冲，虽然都是金庸小说中的主人公，但本身并无联系，考虑到两者可能经常出现在类似的语境中，因此两者词向量比较相似也容易理解。

其他小说中相关分析的结果也较为合理，此处不做解释。

4.2 聚类分析

使用Kmeans聚类的结果如下图所示：



可以看出，同一本小说中的人物基本被分到了同一类中，但也有极少数划分错我的情况，比如柯镇恶，但效果总体还算不错。

结论

本次作业使用Word2Vec模型对金庸的五本小说进行了词向量的构建和聚类，结果显示与某个词（选取小说主角）相似的词在原著中也有一定的联系，在原著中有联系的一系列词（以人物为例）构建而成的词向量距离较近，使用Kmeans聚类效果良好。可见词向量对后续任务的重要意义。

参考文档

深入浅出Word2Vec原理解析

秒懂词向量Word2vec的本质

word2vec词向量中文语料处理gensim总结

利用Word2Vec模型训练Word Embedding,并进行聚类分析