陈真　SY2103801

NLP 第 4 次大作业

1.题目：

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.背景知识：

准备写第四个大作业，在搜集基础知识的时候发现了一个宝藏博客，感觉很好，以后尽量也学习下这种记录的风格。

> @王威廉：Steve Renals算了一下icassp录取文章题目中包含deep learning的数量，发现有44篇，而naacl则有0篇。有一种说法是，语言（词、句子、篇章等）属于人类认知过程中产生的高层认知抽象实体，而语音和图像属于较为底层的原始输入信号，所以后两者更适合做deep learning来学习特征。
> 2013年3月4日 14:46

引用博主的一句话：“第二句我很认同，不过我也有信心以后一定有人能挖掘出语言这种高层次抽象中的本质。不论最后这种方法是不是 Deep Learning，就目前而言，Deep Learning 在 NLP 领域中的研究已经将高深莫测的人类语言撕开了一层神秘的面纱。我觉得其中最有趣也是最基本的，就是“词向量”了。”

词向量的方式是将深度学习算法引入到 NLP 领域的一个核心技术。这里根据博客的内容，简单概括下**词向量的历史**。

2.1.词向量的定义

词向量是将自然语言符号化的一种。一种常见的表示是 one-hot representation。这种方式配个最大熵、SVM、CRF 能够完成 nlp 领域各种主流的任务，但是有一个缺点，那就是“词汇鸿沟”：任意两个词都是独立的。光从这两个词的词向量无法看出两个词是否有关系。

深度学习中一般用到的词向量并不是上面的 one-hot representation。而是用 distributed representation 表示的一种低实数维度的向量。维度以 50 和 100 维比较常见。这种向量的表示不是唯一的，后文会提到计算方法。这种形式的最大贡献在于让相关或者相似的词，在距离上更加接近了。

2.2.词向量的来历

Distributed representation 最早是 Hinton 在 1986 年的论文《Learning distributed representations of concepts 》中提出的。虽然这篇文章中并没有说要将词做 distributed representation ，但是这种先进的思想从 2000 年开始逐渐被人重视。Distributed representation 用来表示词，就是常说的“word representation”或者“word embedding”，中文俗称词向量。

如果用传统的稀疏表示法表示词，在解决某些任务的时候会造成维数灾难，所以使用低纬度的词向量显得很合理。同时从计算实践上来看，高维的特征如果要套用 dl，其复杂度会是难以接受的，因此用低纬度的词向量在这里也饱受追捧。

2.3.词向量的训练

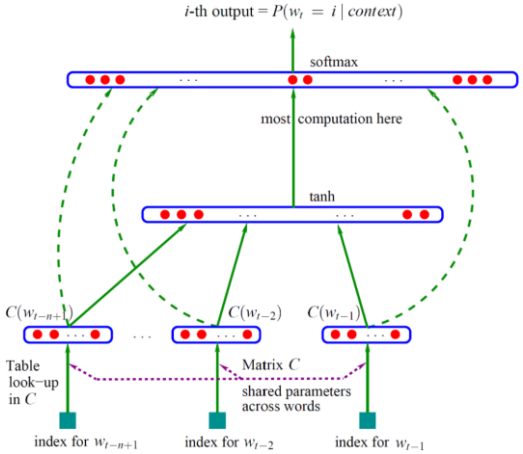
词向量的训练不得不提到语言模型。到目前为止，我所了解到的所有词向量的训练方法都是在训练语言模型的同时，顺便得到的。

对于这一点可以理解为：要从一段无标注的自然文本中学习出一些东西，无非就是统计出词频、词的共现、词的搭配。要从自然语言文本中统计并建立一个语言模型，是一个比较精确的任务。在建立任务的时候，对于文本的统计与分析都是建立在词之上，因此，引出语言模型对应的词向量。

下面介绍的工作都是无监督学习。词向量的训练最经典的有 3 个工作。C&W 2008、M&H 2008、Mikolov 2010。当然在说这些工作之前，还得介绍一下这一系列中 Bengio 的经典之作。

^Bengio 的经典之作

Bengio 在 2001 年最早发表在 NIPS 上的文章《A Neural Probabilistic Language Model》用了一个三层的神经网络来构建语言模型，同样也是 n-gram 模型。



网络第一层(输入层)是将 w_{t-n+1}，…，w_{t-2}，w_{t-1}，这 n-1 个向量首尾相接拼起来，形成一个(n-1)m 维的向量，下面记为 x。

网络第二层（隐藏层）就如同普通的神经网络，直接使用 d+Hx 计算得到。d 是一个偏置项。在此之后，使用 tanh 作为激活函数。

网络的第三层（输出层）一共有|V|个节点，每个节点 y_i 表示下一个词为 i 的未归一化 log 概率。最后使用 softmax 激活函数将输出值 y 归一化成概率。

现在万事俱备，用随机梯度下降法把这个模型优化出来就可以了。需要注意的是，一般神经网络的输入层只是一个输入值，而在这里，输入层 x 也是参数（存在 C 中），也是需要优化的。优化结束之后，词向量有了，语言模型也有了。

补充一个很有意思的事情：在其 JMLR 论文中的未来工作一段，他提了一个能量函数，把输入向量和输出向量统一考虑，并以最小化能量函数为目标进行优化。后来 M&H 工作就是以此为基础展开的。他提到一词多义有待解决，9 年之后 Huang 提出了一种解决方案。他还在论文中随口（不是在 Future Work 中写的）提到：可以使用一些方法降低参数个数，比如用循环神经网络。后来 Mikolov 就顺着这个方向发表了一大堆论文，直到博士毕业。所以一定要多读大牛的论文！

^C&W 的 SENNA

Ronan Collobert 和 Jason Weston 在 2008 年的 ICML 上发表的《A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning》里面首次介绍了他们提出的词向量的计算方法。实际上 C&W 这篇论文主要目的

并不是在于生成一份好的词向量，甚至不想训练语言模型，而是要用这份词向量去完成 NLP 里面的各种任务，比如词性标注、命名实体识别、短语识别、语义角色标注等等。他们没有去近似地求 P(w_t | w_1, w_2, ..., w_{t-1})，而是直接去尝试近似 P(w_1, w_2, ..., w_t)。在实际操作中，他们并没有去求一个字符串的**概率**，而是求窗口连续 n 个词的**打分** f(w_{t-n+1}, ..., w_{t-1}, w_t)。打分 f 越高的说明这句话越是正常的话；打分低的说明这句话不是太合理；如果是随机把几个词堆积在一起，那肯定是负分（差评）。打分只有相对高低之分，并没有概率的特性。有了对 f 的假设，C&W 就直接使用 pair-wise 的方法训练词向量。

^M&H 的 HLBL

Andriy Mnih 和 Geoffrey Hinton 在 2007 年和 2008 年各发表了一篇关于训练语言模型和词向量的文章。2007 年发表在 ICML 上的《Three new graphical models for statistical language modelling》表明了 Hinton 将 Deep Learning 战场扩展到 NLP 领域的决心。2008 年发表在 NIPS 上的《A scalable hierarchical distributed language model》则提出了一种层级的思想替换了 Bengio 2003 方法中最后隐藏层到输出层最花时间的矩阵乘法，在保证效果的基础上，同时也提升了速度。

^Mikolov 的 RNNLM

循环神经网络的最大优势在于，可以真正充分地**利用所有上文信息**来预测下一个词，而不像前面的其它工作那样，只能开一个 n 个词的窗口，只用前 n 个词来预测下一个词。从形式上看，这是一个非常"终极"的模型，毕竟语言模型里能用到的信息，他全用上了。可惜的是，循环神经网络形式上非常好看，使用起来却非常难优化，如果优化的不好，长距离的信息就会丢失，甚至还无法达到开窗口看前若干个词的效果。Mikolov 在 RNNLM 里面只使用了最朴素的 BPTT 优化算法，就已经比 n-gram 中的 state of the art 方法有更好的效果，这非常令人欣慰。如果用上了更强的优化算法，最后效果肯定还能提升很多。

^Huang 的语义强化

Huang 认为 C&W 的工作只利用了"局部上下文（Local Context）"。C&W 在训练词向量的时候，只使用了上下文各 5 个词，算上自己总共有 11 个词的信息，这些局部的信息还不能充分挖掘出中间词的语义信息。Huang 直接使用 C&W 的网络结构计算出一个得分，作为"局部得分"。

然后 Huang 提出了一个"全局信息"，这有点类似传统的词袋子模型。词袋子模型是把文章中所有词的 One-hot Representation 加起来，形成一个向量（就像把词全都扔进一个袋子里），用来表示文章。Huang 的全局模型是将文章中所有词的词向量求个加权平均（权重是词的 idf），作为文章的语义。他把文章的语义向量和当前词的词向量拼接起来，形成一个两倍长度的向量作为输入，之后还是用 C&W 的网络结构算出一个打分。

2.4. word2vec 语言模型

接着上述背景：神经语言模型构建完成之后，就是训练参数了，这里的参数包括：词向量矩阵 C；神经网络的权重；偏置等参数。训练数据就是大堆大堆的语料库。训练结束之后，语言模型得到了：通过" wt−(n+1) ,..., wt−1"去预测第 t 个词是 wt 的概率，但有点意外收获的是词向量" wt−(n+1) ,..., wt−1"也得到了。换言之，词向量是这个语言模型的副产品。当然，这个模型的缺点就是速度问题，因为词汇表往往很大，几十万几百万，训练起来就很耗时，Bengo 仅仅训练 5 个 epoch 就花了 3 周，这还是 40 个 CPU 并行训练的结果。因此才会有了后续好多的优化工作，除了上述介绍的几种之外，word2vec 也是其中一个。

word2vec 是 Google 研究团队里的 Tomas Mikolov 等人于 2013 年的《Distributed Representations ofWords and Phrases and their Compositionality》以及后续的《Efficient Estimation of Word Representations in Vector Space》两篇文章中提出的一种高效训练词向

量的模型，基本出发点是上下文相似的两个词，它们的词向量也应该相似，比如香蕉和梨在句子中可能经常出现在相同的上下文中，因此这两个词的表示向量应该就比较相似。

对于 Word2vec 而言，词向量矩阵的意义就不一样了，因为 Word2Vec 的最终目的不是为了得到一个语言模型，也不是要把 f 训练得多么完美，而是只关心模型训练完后的副产物：词向量矩阵。

这里我们简单·介绍一下代码部分用到的 Word2vec 模式下的 CBOW 模型的训练步骤。

（1）当前词的上下文词语的独热编码输入到输入层。

（2）这些词分别乘以同一个矩阵 W1 后分别得到各自的 1*N 向量。

（3）将这些 1*N 向量取平均为一个 1*N 向量。

（4）将这个 1*N 向量乘矩阵 W2，变成一个 1*V 向量。

（5）将 1*V 向量 softmax 归一化后输出取每个词的概率向量 1*V。

（6）将概率值最大的数对应的词作为预测词。

（7）将预测的结果 1*V 向量和真实标签 1*V 向量（真实标签中的 V 个值中有一个是 1，其他是 0）计算误差，一般是交叉熵。

（8）在每次前向传播之后反向传播误差，不断调整 W1 和 W2 矩阵的值。

3.结果分析：

1. **相似性分析：**

a. **分析发现：对于一个词语的相似词，其在词性上，都是一致的。发现大多数都是一个故事情节中的词语，并且关系很紧密。**

b. **分析还发现：虽然窗口的数量仅为 5，但是还是能发现一些跨小说的词语具有较高的相似性，比如：令狐冲和张无忌，葵花宝典和王重阳。猜想可能是由于在金庸的小说中有一些关联的名词，比如令狐冲和张无忌都和全真教、武当山等会有联系；而葵花宝典虽然和王重阳没有直接联系，但是葵花宝典和九阴真经都是绝世秘籍，九阴真经和王重阳有关联。从这个角度上看，语言模型得到的词向量具有较强的上下文关联，或者说能够挖掘出文本内部的一些信息。**

用两种方法得到的结果以及关联性如下：

c. **利用 CBOW 得到的几个最相关名词结果如下：**

袁承志 [('青青', 0.59722501039505), ('韦小宝', 0.5577995181083679), ('焦宛儿', 0.5557922124862671), ('多隆', 0.5554175972938538), ('洪胜海', 0.5463778376579285), ('穆人清', 0.5451043844223022), ('完颜洪烈', 0.5366737246513367), ('胡斐', 0.4971747100353241), ('张召重', 0.49419283866882324), ('陈近南', 0.4913315176963806)]

黄蓉 [('杨过', 0.8117994070053101), ('欧阳锋', 0.7547358274459839), ('郭靖', 0.7529341578483582), ('黄药师', 0.7192626595497131), ('郭襄', 0.6925807595252991), ('洪七公', 0.6906004548072815), ('周伯通', 0.6844973564147949), ('小龙女', 0.6589915156364441), ('陆无双', 0.653498113155365), ('武三通', 0.6315143704414368)]

杨过 [('黄蓉', 0.8117994666099548), ('小龙女', 0.7985276579856873), ('郭靖', 0.7877666354179382), ('李莫愁', 0.7470971941947937), ('欧阳锋', 0.7414225935935974), ('黄药师', 0.7262893319129944), ('法王', 0.6763430237770081), ('郭襄', 0.6707555651664734), ('武三通', 0.6649258732795715), ('柯镇恶', 0.6374181509017944)]

令狐冲 [('岳不群', 0.7352237105369568), ('盈盈', 0.6372964382171631), ('左冷禅', 0.6267445683479309), ('俞岱岩', 0.6058960556983948), ('石破天', 0.6009785532951355), ('岳夫人', 0.5999348759651184), ('田伯光', 0.5675665736198425), ('岳灵珊', 0.5667975544929504), ('任我行', 0.5586450695991516), ('张无忌', 0.5428802371025085)]
韦小宝 [('康熙', 0.7718302011489868), ('索额图', 0.734282910823822), ('皇上', 0.7072708606719971), ('施琅', 0.6905516982078552), ('乾隆', 0.6694779992103577), ('费要多罗', 0.6342082619667053), ('吴三桂', 0.6282054781913757), ('公主', 0.6138565540313721), ('吴之荣', 0.6119030117988586), ('多隆', 0.6045823693275452)]
峨嵋派 [('本派', 0.7223036885261536), ('华山派', 0.7149252891540527), ('武当派', 0.6774415373802185), ('峨嵋', 0.6711875200271606), ('贵派', 0.6608545184135437), ('门下', 0.6408249735832214), ('青城派', 0.6358729600906372), ('一派', 0.6178311705589294), ('第四代', 0.612822949886322), ('韦陀门', 0.606776773929596)]
葵花宝典 [('九阴真经', 0.7423869967460632), ('成就', 0.7393814921379089), ('上代', 0.7170454263687134), ('秘奥', 0.6997765302658081), ('宝典', 0.6960001587867737), ('王重阳', 0.6907806992530823), ('秘诀', 0.6798321008682251), ('神通', 0.6770796775817871), ('才智', 0.674254834651947), ('钻研', 0.6726574897766113)]
九阴真经 [('经文', 0.7613109350204468), ('真经', 0.7424441576004028), ('葵花宝典', 0.7423869967460632), ('法门', 0.7373106479644775), ('秘笈', 0.7012843489646912), ('练成', 0.7007570862770081), ('滚瓜烂熟', 0.6990230083465576), ('宝典', 0.6961445212364197), ('秘诀', 0.6949171423912048), ('修习', 0.6889116764068604)]

d. 利用 skip-gram 得到的几个最相关名词结果如下：

袁承志 [('青青', 0.8321207761764526), ('何铁手', 0.7563806176185608), ('焦宛儿', 0.73509681224823), ('洪胜海', 0.7163307070732117), ('阿九', 0.6859924793243408), ('程青竹', 0.6679292321205139), ('何惕守', 0.6641486287117004), ('沙天广', 0.6169764399528503), ('崔秋山', 0.6154423356056213), ('穆人清', 0.6067624092102051)]
黄蓉 [('郭靖', 0.9048688411712646), ('洪七公', 0.8639653921127319), ('欧阳锋', 0.8508039116859436), ('周伯通', 0.8259311318397522), ('黄药师', 0.808832585811615), ('欧阳克', 0.7710778117179871), ('靖哥哥', 0.7596354484558105), ('蓉', 0.7566204071044922), ('黄蓉道', 0.7423366904258728), ('穆念慈', 0.7249600887298584)]
杨过 [('小龙女', 0.8149077892303467), ('郭襄', 0.7964937686920166), ('金轮法王', 0.7764447927474976), ('法王', 0.7746132016181946), ('李莫愁', 0.7444475889205933), ('朱子柳', 0.6897915601730347), ('陆无双', 0.6883792877197266), ('周伯通', 0.6840124130249023), ('武三通', 0.681389331817627), ('黄药师', 0.6733368039131165)]
令狐冲 [('盈盈', 0.8319492340087891), ('任我行', 0.7623758316040039), ('冲虚', 0.7604200839996338), ('岳不群', 0.7555186748504639), ('左冷禅', 0.7412029504776001), ('恒山', 0.730137825012207), ('方证', 0.711450457572937), ('

岳先生', 0.6937503814697266), ('向问天', 0.6870913505554199), ('桃谷六仙', 0.6869032979011536)]

韦小宝 [('施琅', 0.772824227809906), ('索额图', 0.7612042427062988), ('佟国纲', 0.736808180809021), ('康熙', 0.72219318151474), ('吴之荣', 0.7088249325752258), ('郑克爽', 0.7081042528152466), ('皇上', 0.7016462087631226), ('台湾', 0.6995344758033752), ('苏荃', 0.6889874935150146), ('陈近南', 0.6853499412536621)]

峨嵋派 [('灭绝师太', 0.7778224945068359), ('峨嵋', 0.7689250111579895), ('武当派', 0.7356485724449158), ('本派', 0.734597384929657), ('武当', 0.6947146654129028), ('别派', 0.6861841678619385), ('郭襄郭', 0.6860575675964355), ('第二代', 0.6852330565452576), ('少林', 0.6799653768539429), ('宋夫人', 0.6766288876533508)]

葵花宝典 [('宝典', 0.8030036091804504), ('残本', 0.7978923320770264), ('手录', 0.7899168133735657), ('那宝典', 0.7295091152191162), ('这训', 0.7274553775787354), ('习之', 0.7256235480308533), ('所录', 0.7168220281600952), ('想夺', 0.7144618630409241), ('武经', 0.7067800164222717), ('这卷', 0.7040396928787231)]

九阴真经 [('真经', 0.8245108723640442), ('经文', 0.8131584525108337), ('这卷', 0.8089039921760559), ('下卷', 0.7961553931236267), ('总纲', 0.7881062030792236), ('滚瓜烂熟', 0.782174825668335), ('上卷', 0.7499107122421265), ('视为至宝', 0.7419134974479675), ('功诀', 0.7387032508850098), ('上下卷', 0.7376535534858704)]

2. 聚类分析:

挑选 4 本小说中的人物如下:

sdyx=["郭靖","黄药师","黄蓉","杨康","穆念慈","欧阳锋","段智兴","周伯通","梅超风","柯镇恶","欧阳克","王重阳","完颜洪烈","郭啸天","包惜弱"]

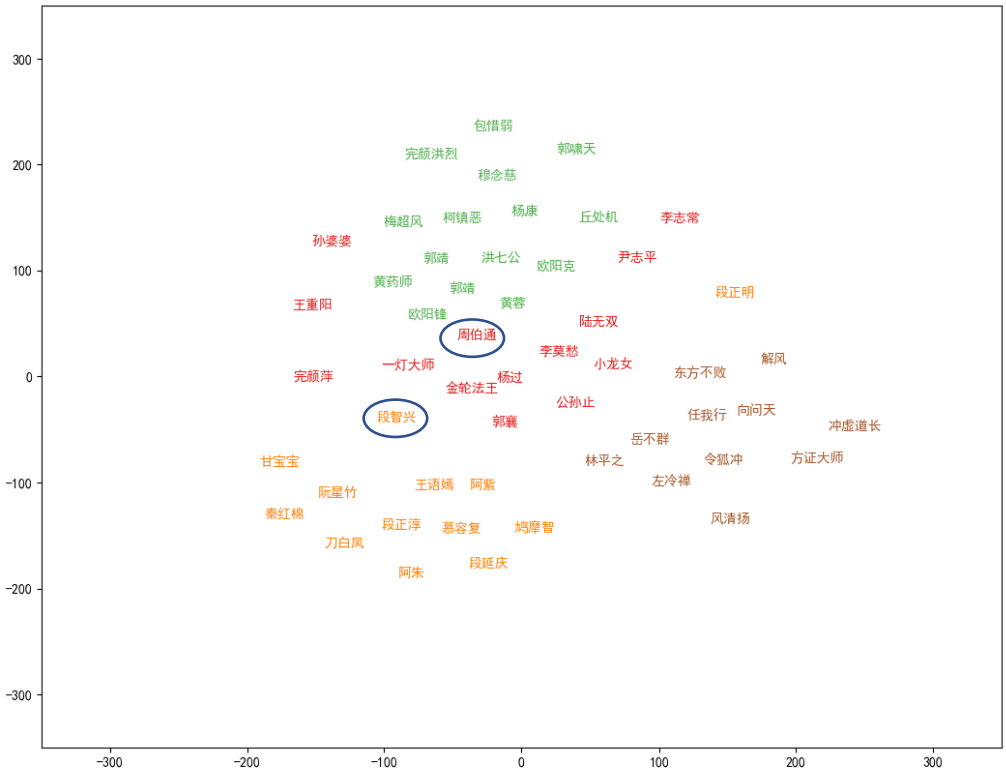
tlbb=["段正明","段正淳","段延庆","刀白凤","秦红棉","甘宝宝","阮星竹","王语嫣","阿朱","阿紫",'慕容复','鸠摩智']

sdxl=["一灯大师","小龙女","尹志平","丘处机","公孙止","孙婆婆","李莫愁","李志常","完颜萍","陆无双","杨过","金轮法王","洪七公","郭襄"]

xajh=["东方不败","风清扬","方证大师","任我行","令狐冲","冲虚道长","岳不群","林平之","左冷禅","解风","向问天"]

利用词向量进行聚类，得到的聚类二维可视化如下：用颜色表示聚类效果

可以发现，除了两个人物有错误，其余人物都聚类正确，并且在有错误的人物中，我们发现：聚类结果将周伯通归为神雕侠侣，我们通过小说发现，确实周伯通在神雕侠侣中和射雕英雄传中都有大量出现。由此可见词向量以及聚类效果都与实际相互验证。



代码如下：

#语料处理：

```
# coding=gbk
import jieba
import gensim
import openpyxl
from gensim.models import word2vec
from gensim.models import Word2Vec
def load stopword():
    '''
    加载停用词表
    :return: 返回停用词的列表
    '''
    f stop = open('cn stopwords.txt', encoding='utf-8')
    sw = [line.strip() for line in f stop]
    f stop.close()
    return sw
def load text(stop words,f):
    texts = []
    for line in f:
        cd = jieba.lcut(line)
        cddd = []
        for dd in cd:
            if dd not in stop words:
                cddd.append(dd)
        texts.append(cddd)
    return texts
# 加载停用词表
stop words = load stopword()
# 读入语料库
f = open('preprocess.txt', "r", encoding='utf-8')
```

```
texts=load text(stop words,f)
f.close()
file = open('op.txt','w',encoding='utf-8')
for line in texts:
    for w in line:
        file.write(w+' ')
    file.write('\n')
```

#模型训练与保存

```
from gensim.models import KeyedVectors
sentences=word2vec.Text8Corpus('op.txt')
model=word2vec.Word2Vec(sentences,sg=1,window=5,min count=2,negative=
3,sample=0.001,hs=1,workers=4)
word vectors = model.wv
word vectors.save("word2vec1.wordvectors")
wv = KeyedVectors.load("word2vec1.wordvectors", mmap='r')
```

#数据分析

```
with open("results.txt",'a') as f:
    print("袁承志",wv.most similar(u"袁承志", topn=10),file=f)
    print("黄蓉",wv.most similar(u"黄蓉", topn=10),file=f)
    print("杨过",wv.most similar(u"杨过", topn=10),file=f)
    print("令狐冲",wv.most similar(u"令狐冲", topn=10),file=f)
    print("韦小宝",wv.most similar(u"韦小宝", topn=10),file=f)
    print("峨嵋派",wv.most similar(u"峨嵋派", topn=10),file=f)
    print("葵花宝典",wv.most similar(u"葵花宝典", topn=10),file=f)
    print("九阴真经",wv.most similar(u"九阴真经", topn=10),file=f)
vc=["郭靖","黄药师","黄蓉","杨康","穆念慈","欧阳锋","段智兴","周伯通","梅超风
","柯镇恶","欧阳克","王重阳","完颜洪烈","郭啸天","包惜弱"]
tian=["段正明","段正淳","段延庆","刀白凤","秦红棉","甘宝宝","阮星竹","王语嫣
","阿朱","阿紫",'慕容复','鸠摩智']
shen=["一灯大师","小龙女","尹志平","丘处机","公孙止","孙婆婆","李莫愁","李志
常","完颜萍","陆无双","杨过","金轮法王","洪七公","郭靖","郭襄"]
xiao=["东方不败","风清扬","方证大师","任我行","令狐冲","冲虚道长","岳不群","
林平之","左冷禅","解风","向问天"]
vc+=tian
vc+=shen
vc+=xiao
wb = openpyxl.load workbook("Test.xlsx")
sheet = wb['Sheet1']
for eve in vc:
    name=[eve]
    name+=list(wv[eve])
    sheet.append(name)
wb.save("Test.xlsx")
```

```
#k mean 算法
import pandas as pd

#参数初始化
inputfile = 'Test.xlsx' #销量及其他属性数据
outputfile = 'data type.xlsx' #保存结果的文件名
k = 4#聚类的类别
iteration = 5 #聚类最大循环次数
data = pd.read excel(inputfile, index col = 'name') #读取数据
data_zs = 1.0*(data - data.mean())/data.std() #数据标准化，std()表示求总
```

```
体样本方差(除以 n-1),numpy 中 std()是除以 n
print(data_zs)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = k, max_iter = iteration) #分为 k 类
#model = KMeans(n_clusters = k, n_jobs = 4, max_iter = iteration) #分为 k 类，并发数 4
model.fit(data_zs) #开始聚类

#简单打印结果
r1 = pd.Series(model.labels_).value_counts() #统计各个类别的数目
r2 = pd.DataFrame(model.cluster_centers_) #找出聚类中心
r = pd.concat([r2, r1], axis = 1) #横向连接（0 是纵向），得到聚类中心对应的类别下的数目
print(r)
r.columns = list(data.columns) + [u'类别数目'] #重命名表头
print(r)

#详细输出原始数据及其类别
r = pd.concat([data, pd.Series(model.labels_, index = data.index)], axis = 1)  #详细输出每个样本对应的类别
r.columns = list(data.columns) + [u'聚类类别'] #重命名表头
r.to_excel(outputfile) #保存结果


#用 TSNE 进行数据降维并展示聚类结果
from sklearn.manifold import TSNE
tsne = TSNE()
tsne.fit_transform(data_zs) #进行数据降维,并返回结果
tsne = pd.DataFrame(tsne.embedding_, index = data_zs.index) #转换数据格式
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['font.size'] = 10
plt.rcParams['axes.unicode_minus']=False
#不同类别用不同颜色和样式绘图
d = tsne[r[u'聚类类别'] == 0]     #找出聚类类别为 0 的数据对应的降维结果
num=k
fig = plt.figure()
ax = plt.subplot(111)
plt.axis([-350, 350, -350, 350])
for i in range(num):
    d=tsne[r[u'聚类类别'] == i]
    e=np.array(d)
    print(len(e[:,0]))
    for l in range(0,len(e[:,0])):

plt.text(e[l,0]*3,e[l,1]*3,str(d.index.values[l]),color=plt.cm.Set1(i / k))
plt.show()
```