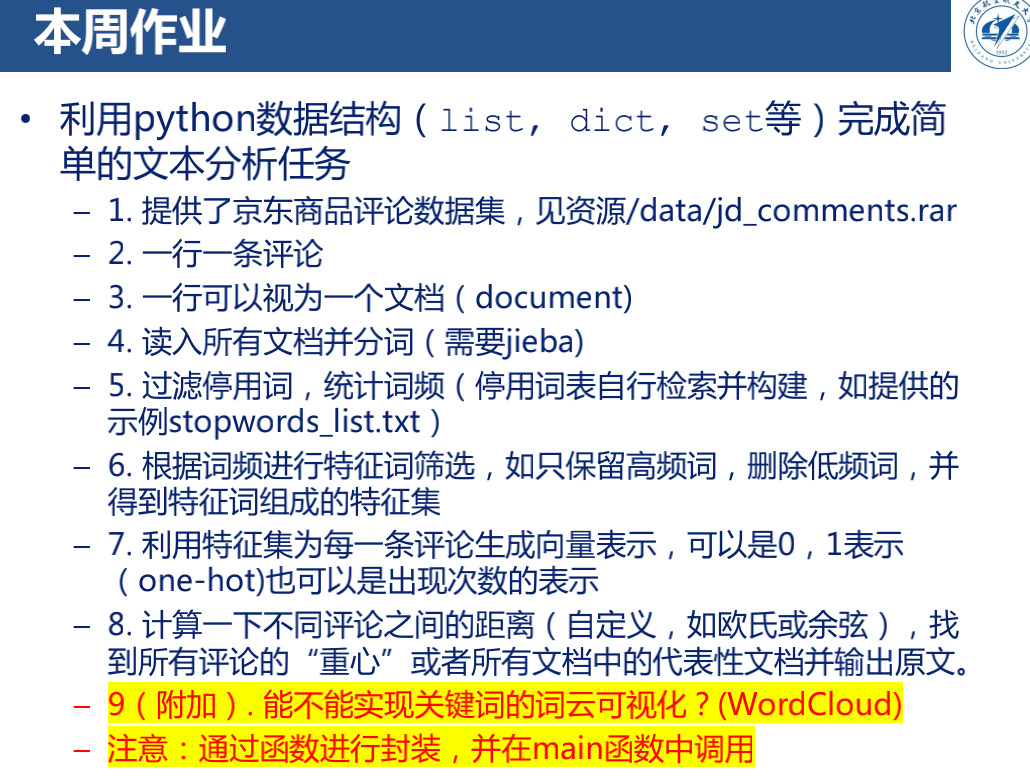
现代程序作业·第一次

1. 作业要求：



1. 实现过程：

对文本进行读取分词，并读取停用词表，将分词后的结果进行进一步的筛选

with open(r'jd\_comments/stopwords\_list.txt','r',encoding='utf-8') as s:  
 stopwords = s.read()  
 stopwords\_list = stopwords.split('\n')  
  
with open(r'jd\_comments/jd\_comments.txt','r',encoding='utf-8') as f:  
 text\_line = f.read()  
 text\_list = text\_line.split('\n')  
 print(text\_list[0])  
  
words\_count = {}  
for i in range(len(text\_list)):  
 words = jieba.analyse.textrank(text\_list[i],topK=20,withWeight=False,allowPOS=('n','v','a','d'))  
 #words = jieba.analyse.textrank(text\_list[i], topK=20, withWeight=False)  
 for word in words:  
 if word not in stopwords\_list:  
 if word not in words\_count:  
 words\_count[word] = 1  
 else:  
 words\_count[word] +=1  
  
words\_count\_sorted = collections.OrderedDict(sorted(words\_count.items(), key=lambda dc:dc[1],reverse=True))  
  
  
words\_count\_list = []  
for key in words\_count\_sorted:  
 if words\_count\_sorted[key] >= 30:  
 words\_count\_list.append({'word': key, 'count': words\_count\_sorted[key]})  
print(words\_count\_list)  
  
words\_sorted\_list = []  
for j in range(len(words\_count\_list)):  
 words\_sorted\_list.append(words\_count\_list[j]['word'])  
print(words\_sorted\_list)  
  
print('text\_list 长度',len(text\_list))

分词统计结果：

[{'word': '不错', 'count': 301}, {'word': '电脑', 'count': 195}, {'word': '非常', 'count': 181}, {'word': '速度', 'count': 172}, {'word': '没有', 'count': 166}, {'word': '开机', 'count': 143}, {'word': '问题', 'count': 131}, {'word': '满意', 'count': 126}, {'word': '客服', 'count': 125}, {'word': '感觉', 'count': 119}, {'word': '很快', 'count': 117}, {'word': '收到', 'count': 107}, {'word': '包装', 'count': 89}, {'word': '真的', 'count': 89}, {'word': '有点', 'count': 89}, {'word': '喜欢', 'count': 86}, {'word': '使用', 'count': 85}, {'word': '外观', 'count': 83}, {'word': '系统', 'count': 82}, {'word': '物流', 'count': 81}, {'word': '键盘', 'count': 81}, {'word': '效果', 'count': 77}, {'word': '比较', 'count': 75}, {'word': '快递', 'count': 72}, {'word': '值得', 'count': 70}, {'word': '运行', 'count': 69}, {'word': '价格', 'count': 69}, {'word': '游戏', 'count': 69}, {'word': '性价比', 'count': 66}, {'word': '屏幕', 'count': 66}, {'word': '流畅', 'count': 65}, {'word': '耐心', 'count': 65}, {'word': '硬盘', 'count': 65}, {'word': '性能', 'count': 64}, {'word': '购买', 'count': 63}, {'word': '机器', 'count': 61}, {'word': '推荐', 'count': 61}, {'word': '鼠标', 'count': 61}, {'word': '知道', 'count': 59}, {'word': '方便', 'count': 58}, {'word': '总体', 'count': 58}, {'word': '希望', 'count': 57}, {'word': '安装', 'count': 54}, {'word': '清晰', 'count': 54}, {'word': '好评', 'count': 50}, {'word': '卖家', 'count': 48}, {'word': '配置', 'count': 48}, {'word': '特别', 'count': 48}, {'word': '质量', 'count': 46}, {'word': '购物', 'count': 46}, {'word': '办公', 'count': 45}, {'word': '笔记本', 'count': 45}, {'word': '固态', 'count': 44}, {'word': '声音', 'count': 43}, {'word': '发货', 'count': 43}, {'word': '服务态度', 'count': 43}, {'word': '需要', 'count': 43}, {'word': '已经', 'count': 43}, {'word': '评价', 'count': 43}, {'word': '态度', 'count': 42}, {'word': '玩游戏', 'count': 41}, {'word': '内存', 'count': 39}, {'word': '打开', 'count': 39}, {'word': '好看', 'count': 38}, {'word': '做工', 'count': 38}, {'word': '起来', 'count': 38}, {'word': '朋友', 'count': 36}, {'word': '应该', 'count': 35}, {'word': '支持', 'count': 35}, {'word': '店家', 'count': 34}, {'word': '散热', 'count': 34}, {'word': '机子', 'count': 34}, {'word': '完美', 'count': 34}, {'word': '发现', 'count': 33}, {'word': '不会', 'count': 33}, {'word': '很好', 'count': 33}, {'word': '终于', 'count': 33}, {'word': '时间', 'count': 33}, {'word': '软件', 'count': 33}, {'word': '风扇', 'count': 31}, {'word': '手感', 'count': 31}, {'word': '漂亮', 'count': 31}, {'word': '觉得', 'count': 31}, {'word': '显卡', 'count': 31}, {'word': '适合', 'count': 30}, {'word': '很大', 'count': 30}, {'word': '电池', 'count': 30}, {'word': '视频', 'count': 30}]

分词内容列表

['不错', '电脑', '非常', '速度', '没有', '开机', '问题', '满意', '客服', '感觉', '很快', '收到', '包装', '真的', '有点', '喜欢', '使用', '外观', '系统', '物流', '键盘', '效果', '比较', '快递', '值得', '运行', '价格', '游戏', '性价比', '屏幕', '流畅', '耐心', '硬盘', '性能', '购买', '机器', '推荐', '鼠标', '知道', '方便', '总体', '希望', '安装', '清晰', '好评', '卖家', '配置', '特别', '质量', '购物', '办公', '笔记本', '固态', '声音', '发货', '服务态度', '需要', '已经', '评价', '态度', '玩游戏', '内存', '打开', '好看', '做工', '起来', '朋友', '应该', '支持', '店家', '散热', '机子', '完美', '发现', '不会', '很好', '终于', '时间', '软件', '风扇', '手感', '漂亮', '觉得', '显卡', '适合', '很大', '电池', '视频']

所有文本数量

text\_list 长度 1003

生成特征向量，利用特征向量生成距离矩阵，计算每条评论与其他评论之间的总距离，将总距离最小的评论作为重心。

feature\_vector\_list = []  
for i in range(len(text\_list)):  
 feature\_vector = np.zeros(len(words\_count\_list))  
 for j in range(len(words\_count\_list)):  
 if words\_count\_list[j]['word'] in text\_list[i]:  
 feature\_vector[j] = 1  
 feature\_vector\_list.append(feature\_vector)  
print(feature\_vector\_list[0])  
  
dist\_matrix(feature\_vector\_list,text\_list)  
  
core\_vector = np.zeros(len(words\_count\_list))  
for i in range(len(text\_list)):  
 core\_vector+=feature\_vector\_list[i]  
core\_vector/=[len(text\_list)]  
print(core\_vector)

生成特征向量中的一个

[1. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.

0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

向量求和的平均值

[0.35194417 0.24725823 0.20538385 0.19541376 0.19441675 0.16251246

0.15852443 0.16550349 0.1445663 0.14556331 0.13060818 0.13958126

0.11465603 0.10169492 0.10169492 0.09770688 0.10767697 0.10368893

0.10867398 0.09571286 0.09272183 0.08574277 0.09371884 0.08175474

0.07976072 0.07676969 0.10069791 0.13858425 0.08973081 0.08374875

0.07876371 0.06979063 0.08574277 0.07577268 0.07078764 0.07377866

0.07577268 0.07577268 0.06979063 0.07976072 0.0777667 0.06480558

0.06480558 0.0777667 0.07078764 0.0558325 0.06380857 0.05982054

0.07078764 0.05682951 0.06081755 0.06979063 0.05682951 0.05284148

0.04985045 0.04586241 0.05284148 0.05882353 0.06281157 0.09371884

0.0448654 0.05982054 0.04685942 0.04685942 0.04785643 0.06281157

0.04785643 0.03788634 0.04187438 0.03788634 0.05084746 0.04187438

0.0448654 0.0448654 0.04586241 0.21435693 0.03988036 0.08175474

0.05084746 0.03888335 0.03190429 0.05882353 0.04386839 0.04187438

0.03589232 0.03589232 0.03788634 0.03888335]

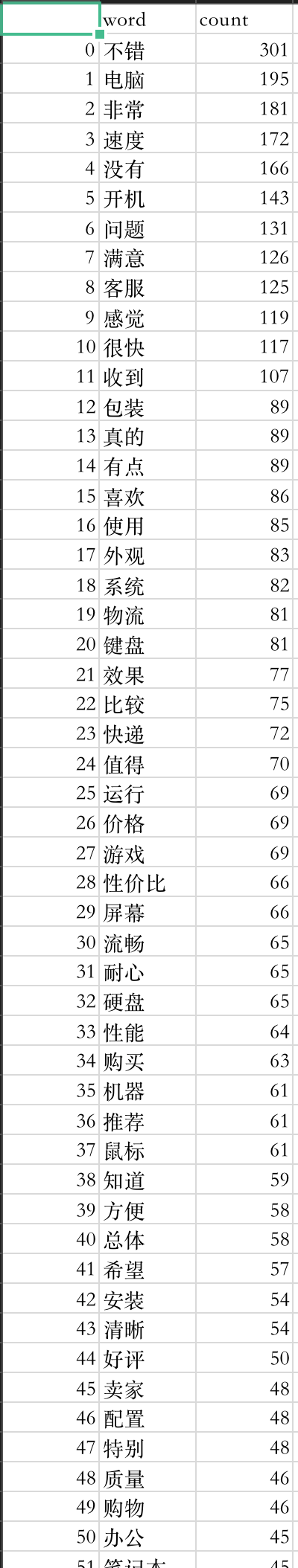
def dist\_matrix(vs\_list, content):  
 length = len(vs\_list)  
 distance = []  
 vector1 = np.array(vs\_list)  
 for i in range(length):  
 vector2 = vs\_list[i]  
 dis = np.sqrt(np.sum(np.square(vector1 - vector2)))  
 distance.append(dis)  
 print("第%d个评论与重心的距离为：%f" %((i+1), distance[i]) )  
 pos = distance.index(min(distance))  
 print("第%d个评论为评论重心" %(pos+1))  
 print("评论内容为：" + content[pos])

第488个评论为评论重心

评论内容为：可以可以，之前一直纠结I7还是AMD，不负所望，京东答应的E卡记得要送哦，还有水冷等着装，测试了CPU温度不高（鲁大师）

输出为csv文件

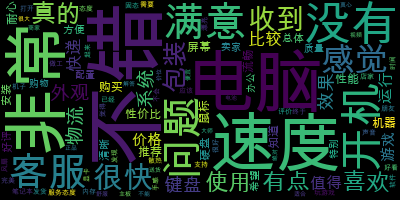
words\_df = pd.DataFrame(words\_count\_list)  
print(words\_df.head())  
words\_df.to\_csv('keyword.csv',encoding='utf-8')



词云图：

def draw\_cloud(read\_name):  
 #image = Image.open('')  
 wc = WordCloud(font\_path = 'Arial Unicode.ttf',background\_color = 'black',max\_words=100)  
 dic = words\_count  
 wc.generate\_from\_frequencies(dic)  
 plt.imshow(wc)  
 plt.axis('off')  
 plt.show()  
 wc.to\_file('词云.png')

draw\_cloud('keyword.csv')



完整代码

import jieba  
import jieba.analyse  
import pandas as pd  
import matplotlib.pyplot as plt  
from wordcloud import WordCloud,ImageColorGenerator  
import collections  
import numpy as np  
from PIL import Image  
  
def draw\_cloud(read\_name):  
 #image = Image.open('')  
 wc = WordCloud(font\_path = 'Arial Unicode.ttf',background\_color = 'black',max\_words=100)  
 dic = words\_count  
 wc.generate\_from\_frequencies(dic)  
 plt.imshow(wc)  
 plt.axis('off')  
 plt.show()  
 wc.to\_file('词云.png')  
  
  
def dist\_matrix(vs\_list, content):  
 length = len(vs\_list)  
 distance = []  
 vector1 = np.array(vs\_list)  
 for i in range(length):  
 vector2 = vs\_list[i]  
 dis = np.sqrt(np.sum(np.square(vector1 - vector2)))  
 distance.append(dis)  
 print("第%d个评论与重心的距离为：%f" %((i+1), distance[i]) )  
 pos = distance.index(min(distance))  
 print("第%d个评论为评论重心" %(pos+1))  
 print("评论内容为：" + content[pos])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with open(r'jd\_comments/stopwords\_list.txt','r',encoding='utf-8') as s:  
 stopwords = s.read()  
 stopwords\_list = stopwords.split('\n')  
  
 with open(r'jd\_comments/jd\_comments.txt','r',encoding='utf-8') as f:  
 text\_line = f.read()  
 text\_list = text\_line.split('\n')  
 print(text\_list[0])  
  
 words\_count = {}  
 for i in range(len(text\_list)):  
 words = jieba.analyse.textrank(text\_list[i],topK=20,withWeight=False,allowPOS=('n','v','a','d'))  
 #words = jieba.analyse.textrank(text\_list[i], topK=20, withWeight=False)  
 for word in words:  
 if word not in stopwords\_list:  
 if word not in words\_count:  
 words\_count[word] = 1  
 else:  
 words\_count[word] +=1  
  
 words\_count\_sorted = collections.OrderedDict(sorted(words\_count.items(), key=lambda dc:dc[1],reverse=True))  
  
  
 words\_count\_list = []  
 for key in words\_count\_sorted:  
 if words\_count\_sorted[key] >= 30:  
 words\_count\_list.append({'word': key, 'count': words\_count\_sorted[key]})  
 print(words\_count\_list)  
  
 words\_sorted\_list = []  
 for j in range(len(words\_count\_list)):  
 words\_sorted\_list.append(words\_count\_list[j]['word'])  
 print(words\_sorted\_list)  
  
 print('text\_list 长度',len(text\_list))  
  
  
 feature\_vector\_list = []  
 for i in range(len(text\_list)):  
 feature\_vector = np.zeros(len(words\_count\_list))  
 for j in range(len(words\_count\_list)):  
 if words\_count\_list[j]['word'] in text\_list[i]:  
 feature\_vector[j] = 1  
 feature\_vector\_list.append(feature\_vector)  
 print(feature\_vector\_list[0])  
  
 dist\_matrix(feature\_vector\_list,text\_list)  
  
 core\_vector = np.zeros(len(words\_count\_list))  
 for i in range(len(text\_list)):  
 core\_vector+=feature\_vector\_list[i]  
 core\_vector/=[len(text\_list)]  
 print(core\_vector)  
  
 words\_df = pd.DataFrame(words\_count\_list)  
 print(words\_df.head())  
 words\_df.to\_csv('keyword.csv',encoding='utf-8')  
  
 draw\_cloud('keyword.csv')