***import pandas as pd***

***import gensim***

***from gensim.parsing.preprocessing import STOPWORDS***

***from nltk.stem import WordNetLemmatizer, SnowballStemmer***

***from nltk.stem.porter import \****

***import numpy as np***

***np.random.seed(2018)***

***import nltk***

***import re***

***from nltk.corpus import stopwords***

***from gensim import corpora, models***

***import jieba***

***from sklearn.cluster import KMeans***

***from sklearn.metrics import silhouette\_score***

***import emoji***

***# 读取CSV文件***

***file\_path = r'D:\VsFile\Python\啊\_comment.csv'***

***data = pd.read\_csv(file\_path)***

***# 加载中文停用词***

***def load\_stopwords(filepath):***

***with open(filepath, 'r', encoding='utf-8') as file:***

***stopwords = set(word.strip() for word in file.readlines())***

***return stopwords***

***# 清洗评论内容***

***def clean\_text(text):***

***# 仅保留中文字符、空格，并去除表情符号***

***pattern = re.compile("[^\u4e00-\u9fa5]")  # 匹配非中文字符***

***text = emoji.demojize(text)***

***text = re.sub(pattern, "", text)***

***# 分词***

***tokens = jieba.lcut(text)***

***# 去除停用词***

***stop\_words = load\_stopwords(r'D:\VsFile\Python\baidu\_stopwords.txt')***

***# 添加单字作为停用词***

***single\_chars = set([t for t in tokens if len(t) == 1])***

***stop\_words.update(single\_chars)***

***cleaned\_tokens = [token for token in tokens if token not in stop\_words and token.isalpha()]***

***# 如果清洗后为空，则保留一个特殊标记***

***if not cleaned\_tokens:***

***cleaned\_tokens = ['<empty>']***

***return cleaned\_tokens***

***# 删除重复的行，保留首次出现的行***

***data = data.drop\_duplicates()***

***# 应用清洗函数***

***data['cleaned\_content'] = data['用户评论'].apply(clean\_text)***

***print('清洗数据结束')***

***# 针对每个景点提取主题***

***scenic\_areas = data['景点'].unique()***

***all\_topic\_keywords = {}***

***all\_lda\_models = {}***

***# 对每个景点提取主题***

***for scenic\_area in scenic\_areas:***

***print(f"Processing {scenic\_area}...")***

***scenic\_area\_data = data[data['景点'] == scenic\_area]***

***# 创建词典***

***dictionary = corpora.Dictionary(scenic\_area\_data['cleaned\_content'])***

***# 创建语料库***

***corpus = [dictionary.doc2bow(text) for text in scenic\_area\_data['cleaned\_content']]***

***# 设置LDA模型参数 - 调整主题数量为2***

***chunksize = 2000***

***passes = 20***

***iterations = 400***

***eval\_every = None***

***# 训练LDA模型***

***lda = gensim.models.ldamodel.LdaModel(***

***corpus=corpus,***

***id2word=dictionary,***

***num\_topics=2,  # 调整为2个主题***

***random\_state=100,***

***update\_every=1,***

***chunksize=chunksize,***

***passes=passes,***

***alpha='auto',***

***per\_word\_topics=True,***

***iterations=iterations,***

***eval\_every=eval\_every***

***)***

***all\_lda\_models[scenic\_area] = lda***

***# 获取主题关键词***

***topic\_keywords = []***

***for i in range(2):  # 只获取2个主题***

***keywords = [kw[0] for kw in lda.show\_topic(i, topn=10)]***

***topic\_keywords.append(keywords)***

***# 返回主题关键词***

***all\_topic\_keywords[scenic\_area] = topic\_keywords***

***# 初始化一个新的 DataFrame 用于存储每个景区的信息***

***scenic\_area\_topics\_df = pd.DataFrame(columns=['景区名称', '主题1关键词', '主题2关键词'])***

***# 将结果合并到新的 DataFrame***

***for scenic\_area, keywords in all\_topic\_keywords.items():***

***# 创建一个临时字典，用于存储当前景区的行数据***

***row\_data = {'景区名称': scenic\_area,***

***'主题1关键词': keywords[0],***

***'主题2关键词': keywords[1]}***

***# 将字典转换为 DataFrame 并追加到 scenic\_area\_topics\_df***

***scenic\_area\_topics\_df = pd.concat([scenic\_area\_topics\_df, pd.DataFrame(row\_data, index=[0])], ignore\_index=True)***

***# 保存结果***

***scenic\_area\_topics\_df.to\_csv('scenic\_area\_topics.csv', index=False)***

***print('主题关键词提取完成')***

***# 用于存储每个景点的平均主题分布***

***scenic\_area\_topic\_distributions = {}***

***# 为每个景点提取平均主题分布***

***for scenic\_area in scenic\_areas:***

***scenic\_area\_data = data[data['景点'] == scenic\_area]***

***dictionary = corpora.Dictionary(scenic\_area\_data['cleaned\_content'])***

***lda = all\_lda\_models[scenic\_area]***

***corpus = [dictionary.doc2bow(text) for text in scenic\_area\_data['cleaned\_content']]***

***topic\_distributions = [lda.get\_document\_topics(doc) for doc in corpus]***

***# 创建一个空的二维数组，形状为 (文档数量, 主题数量)***

***num\_topics = lda.num\_topics***

***num\_docs = len(topic\_distributions)***

***topic\_distributions\_array = np.zeros((num\_docs, num\_topics))***

***# 填充数组***

***for i, doc\_topics in enumerate(topic\_distributions):***

***for topic\_id, prob in doc\_topics:***

***topic\_distributions\_array[i, topic\_id] = prob***

***# 计算平均主题分布***

***avg\_topic\_distribution = np.mean(topic\_distributions\_array, axis=0)***

***scenic\_area\_topic\_distributions[scenic\_area] = avg\_topic\_distribution***

***# 在转换为可用于聚类的矩阵前打印scenic\_area\_topic\_distributions的长度***

***print(len(scenic\_area\_topic\_distributions))***

***# 转换为可用于聚类的矩阵***

***X = np.array(list(scenic\_area\_topic\_distributions.values()))***

***# 打印X矩阵的形状***

***print(X.shape)***

***# 应用K-Means聚类***

***n\_clusters = 5  # 可以调整簇的数量***

***kmeans = KMeans(n\_clusters=n\_clusters, random\_state=0).fit(X)***

***labels = kmeans.labels\_***

***# 在应用K-Means聚类后打印labels的长度***

***print(len(labels))***

***# 计算轮廓系数评估聚类质量***

***# 打印scenic\_area\_topics\_df的行数***

***print(len(scenic\_area\_topics\_df))***

***silhouette\_avg = silhouette\_score(X, labels)***

***print("The average silhouette\_score is :", silhouette\_avg)***

***# 将聚类标签添加到结果DataFrame中***

***# 确保聚类标签的数量与景点数量一致***

***scenic\_area\_topics\_df['聚类标签'] = list(labels)***

***# 保存结果***

***scenic\_area\_topics\_df.to\_csv('scenic\_area\_topics\_with\_clusters.csv', index=False)***

***print('主题关键词聚类完成')***

这是完整代码，我想要通过评论获得主题，然后通过主题对景区进行聚类分析，也就是每一个景区对应一个plot，同时，这个代码有报错  
Traceback (most recent call last):

File "d:\VsFile\Python\关键词提取.py", line 110, in <module>

scenic\_area\_topics\_df = pd.concat([scenic\_area\_topics\_df, pd.DataFrame(row\_data, index=[0])], ignore\_index=True)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

ruction.py", line 503, in dict\_to\_mgr

return arrays\_to\_mgr(arrays, columns, index, dtype=dtype, typ=typ, consolidate=copy)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Users\simple\.conda\envs\D2024-7-9\Lib\site-packages\pandas\core\internals\construction.py", line 119, in arrays\_to\_mgr

arrays, refs = \_homogenize(arrays, index, dtype)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Users\simple\.conda\envs\D2024-7-9\Lib\site-packages\pandas\core\internals\construction.py", line 630, in \_homogenize

com.require\_length\_match(val, index)

File "C:\Users\simple\.conda\envs\D2024-7-9\Lib\site-packages\pandas\core\common.py", line 573, in require\_length\_match

raise ValueError(

ValueError: Length of values (10) does not match length of index (1)  
如何修改保证我能完整目标并且修复