下面是一段逐行注释和整体框架的详细讲解，包括对一些英语单词的中文解释以及每段代码的作用，让你对这份程序的执行流程一目了然。

整体结构与思路

1. 导入依赖库（import statements）

• requests：负责网络请求

• BeautifulSoup：解析网页 HTML

• jieba：中文分词

• re：正则表达式处理

• matplotlib.pyplot：绘图库，用来展示图像

• wordcloud：生成词云的库

2. 定义 cookies 和请求头 headers

• 用于模拟登录微博或绕过微博的反爬虫限制

3. 发送网络请求并获取响应 response

• 使用 requests.get() 访问微博热搜页面

4. 解析 HTML

• 用 BeautifulSoup 将 HTML 文本转换为可检索的对象

• 通过 soup.select() 提取热搜标题

5. 打印热搜榜前 10 条

• 验证数据是否抓取正确

6. 定义并使用分词函数 clean\_text()

• 用正则表达式 re.sub() 去除非中文字符

• 用 jieba.cut() 进行中文分词

• 过滤掉停用词和长度小于 2 的词

7. 定义并使用生成词云函数 generate\_wordcloud()

• 用 WordCloud() 设置词云属性，如字体、尺寸、背景色等

• 用 plt.imshow() 展示词云

• 用 wordcloud.to\_file() 保存生成的词云图

8. 调用 generate\_wordcloud()

• 将清理后的文本生成词云并可视化

代码分段解读

import requests

from bs4 import BeautifulSoup

import jieba

import re

import matplotlib.pyplot as plt

from wordcloud import WordCloud

1. import requests

• requests：网络请求库，用来发送 HTTP 请求并获取响应。

• “request” 的中文意思是 “请求”。

2. from bs4 import BeautifulSoup

• BeautifulSoup：将 HTML 文本转换为可以搜索、提取的 Python 对象。

• “soup” 这里类比 “汤”，表示把网页内容“煮”成可解析的结构。

3. import jieba

• jieba：中文分词工具。

4. import re

• re：正则表达式库，用于字符串匹配和替换。

• “regular expression（regex）” 中文是 “正则表达式”。

5. import matplotlib.pyplot as plt

• matplotlib：Python 的 2D 绘图库，可以画图表或展示图片。

• “pyplot” 相当于一个类似 MATLAB 的绘图接口。

6. from wordcloud import WordCloud

• WordCloud：词云生成库，可以把文字变成词云图形。

# 你的微博 cookies

cookies = {

"SUB":"\_2AkMQlHMIf8NxqwFRmfAUxWvgbo12yAjEieKmyILTJRMxHRl-yT9kqnAPtRB6OxRd53t0PwazLoN2IL93tBYMS5STSZqN",

"SUBP":"0033WrSXqPxfM72-Ws9jqgMF55529P9D9WWlBqWC8an9MTpxmSbq5fWw"

}

• cookies：存储微博登录或访问时所需的身份信息，帮助我们绕过微博的 “Sina Visitor System”。

• “cookie” 在网络里指 “会话标识”，用来跟踪用户登录状态。

# 发送请求

url = "https://s.weibo.com/top/summary"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"

}

1. url = "..."

• 要爬取的微博热搜榜链接。

2. headers = {...}

• User-Agent：告诉对方服务器，你的访问来自什么浏览器或客户端。

• 这可以伪装成真实浏览器，减少被反爬虫拦截的几率。

response = requests.get(url, headers=headers, cookies=cookies)

• requests.get(...)：对指定 url 发起 GET 请求，并返回响应对象。

• headers=headers 和 cookies=cookies：附加请求头和 cookie 信息，模拟正常访问。

• response：包含网页的 HTML、状态码等信息。

# 输出返回内容的前1000字符，检查是否成功获取数据--weibo热搜榜HTML数据

# print(response.text[:1000])

• 这里注释掉了打印语句，之前用它来调试，查看获取到的 HTML 前 1000 个字符是否正确。

# 解析 HTML

soup = BeautifulSoup(response.text, "html.parser")

1. BeautifulSoup(response.text, "html.parser")

• 把 response.text（网页 HTML 文本）交给 BeautifulSoup，用 HTML 解析器（parser）进行处理。

• “parse” 的中文意思是 “解析”。

• soup 是一个 BeautifulSoup 对象，可以用 .select()、.find() 等方法搜索 DOM 元素。

# 提取热搜标题

hot\_topics = [tag.text.strip() for tag in soup.select(".td-02 a")]

1. soup.select(".td-02 a")

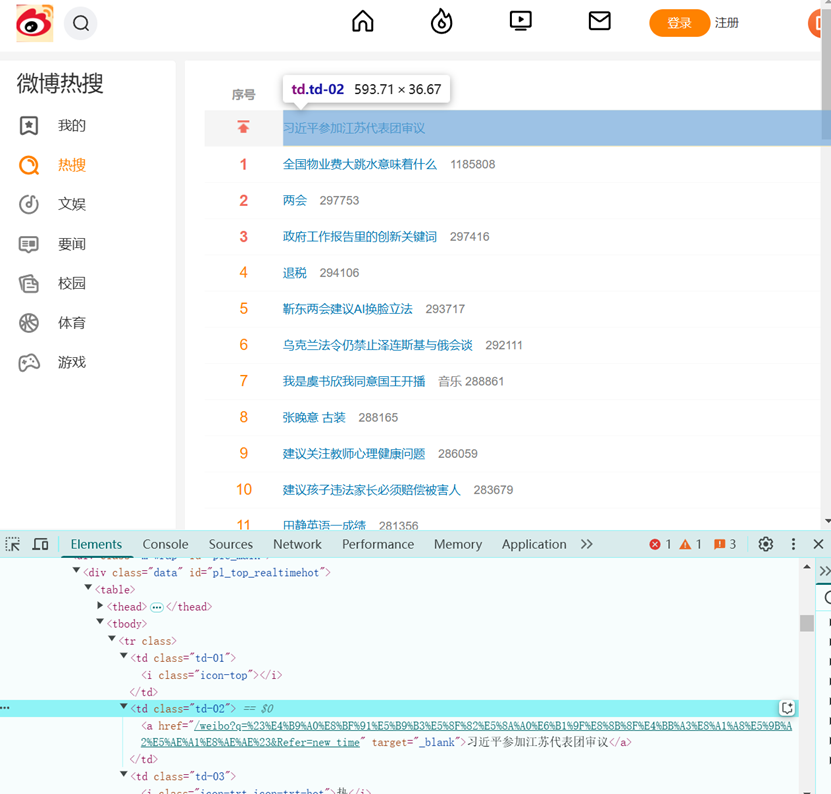
• 在 HTML 里查找所有符合 CSS 选择器 “.td-02 a” 的标签。

• .td-02 表示类名为 td-02 的元素，a 表示超链接标签。

2. [tag.text.strip() for tag in ...]

• 遍历查找到的所有 tag，提取其 .text（标签内文本），并用 .strip() 去除首尾空格。

3. hot\_topics

 • 最终得到一个 列表，里面是所有热搜的标题文本。

# 打印前 10 个热搜标题

print("微博热搜榜：")

for i, topic in enumerate(hot\_topics[:10], 1):

print(f"{i}. {topic}")

1. for i, topic in enumerate(hot\_topics[:10], 1):

• enumerate：将列表组合成 (索引, 元素) 的可迭代对象。

• hot\_topics[:10]：只取前 10 个热搜。

• 1：从 1 开始计数，而不是默认的 0。

2. print(f"{i}. {topic}")

• 用 f-string（格式化字符串）打印出序号和对应的热搜标题。

# 加载停用词表（可以扩展）

STOPWORDS = {"的", "了", "是", "在", "就", "都"}

• STOPWORDS：一个 集合，包含一些常见的无意义词（也叫“停用词”）。

• 例如 “的”、“了”、“是” 等在中文里频繁出现，但对分析没有帮助。

def clean\_text(text\_list):

"""合并文本，去除无关字符，进行分词"""

text = " ".join(text\_list) # 合并热搜标题

text = re.sub(r"[^\u4e00-\u9fa5]", " ", text) # 只保留中文

words = jieba.cut(text) # 分词

words = [w for w in words if len(w) > 1 and w not in STOPWORDS] # 过滤单字和停用词

return " ".join(words)

1. def clean\_text(text\_list):

• 定义一个函数，用来清理和分词微博热搜列表。

2. text = " ".join(text\_list)

• 将所有热搜标题合并成一个长字符串，用空格分隔。

3. re.sub(r"[^\u4e00-\u9fa5]", " ", text)

• 正则替换：将非中文字符替换成空格。

• r"[^\u4e00-\u9fa5]"：匹配所有非中文字符。

4. jieba.cut(text)

• 调用 jieba 库进行中文分词，返回一个可迭代的分词结果。

5. [w for w in words if len(w) > 1 and w not in STOPWORDS]

• 列表推导式：只保留长度大于 1且不在停用词表中的词。

6. " ".join(words)

• 将最终分词后的列表用空格连接，形成一个可供词云使用的分词字符串。

cleaned\_text = clean\_text(hot\_topics)

print("清理后的文本:", cleaned\_text[:100]) # 预览部分结果

1. cleaned\_text = clean\_text(hot\_topics)

• 调用我们定义的 clean\_text() 函数，把热搜标题做清洗和分词处理。

2. print(...[:100])

• 打印前 100 个字符，便于查看结果，不至于输出过多。

def generate\_wordcloud(text, output\_path="weibo\_wordcloud.png"):

"""生成微博热搜词云图"""

wordcloud = WordCloud(

font\_path="simhei.ttf", # 指定中文字体

width=1000, height=600,

background\_color="white",

max\_words=100

).generate(text)

1. def generate\_wordcloud(text, output\_path="weibo\_wordcloud.png"):

• 定义一个函数，用来根据给定的文本生成词云并保存。

2. WordCloud(...)：初始化词云对象，主要参数：

• font\_path="simhei.ttf"：指定中文字体文件，以防止乱码。

• width=1000, height=600：词云图片的宽、高。

• background\_color="white"：背景色为白色。

• max\_words=100：显示最多 100 个词。

3. .generate(text)：将我们的分词文本传入，生成词云数据。

# 显示词云

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off") # 关闭坐标轴

plt.show()

1. plt.figure(figsize=(10, 5))

• 创建一个图像画布，大小是 (10, 5) 英寸。

2. plt.imshow(wordcloud, interpolation="bilinear")

• 在画布上显示词云图。

• “interpolation” 的中文意思是 “插值”，bilinear（双线性插值）让图像更平滑。

3. plt.axis("off")

• 隐藏坐标轴，让图像更干净。

4. plt.show()

• 弹出图像窗口或在 notebook 里显示图像。

# 保存词云图

wordcloud.to\_file(output\_path)

print(f"词云图已保存至: {output\_path}")

• wordcloud.to\_file(output\_path)：将词云图保存为一个 .png 文件。

• 这里默认文件名是 weibo\_wordcloud.png。

# 生成微博热搜词云

generate\_wordcloud(cleaned\_text)

• 调用我们定义的 generate\_wordcloud() 函数，用 cleaned\_text 来生成词云图。

• 生成后会自动显示在窗口，并保存到 weibo\_wordcloud.png 文件。

总结

1. 获取 HTML：利用 requests + cookies 绕过微博限制，获取热搜页面内容。

2. 解析网页：用 BeautifulSoup 查找热搜标题标签，提取文本。

3. 清理分词：用正则表达式去除非中文字符，用 jieba 切分词语，并过滤停用词。

4. 生成词云：使用 WordCloud 将清理好的文本转换成词云图，用 matplotlib 可视化。

到此为止，你的微博热搜词云程序就完成了！如果需要进一步优化，比如定时更新或更多可视化样式，可以再做扩展。祝你学习愉快，玩得开心！