网络文本爬取与内容分析报告

# 数据源选择

## 1.1数据源背景

去哪儿网（Qunar.com）是中国领先的一站式旅行平台。去哪儿网站上线于2005年5月，2010年去哪儿旅行App面世。成立至今，去哪儿坚持以低价为核心竞争力，帮助更多用户解锁“人生第一张机票”。截至目前，去哪儿已拥有超两亿交易用户，并以年新增交易用户逾千万的规模持续增长。

选取的旅游地为厦门市，简称“厦”或“鹭”，福建省辖地级市、副省级市、计划单列市、Ⅰ型大城市， 位于中国华东地区、福建省东南部沿海，与漳州市、泉州市相连，地形以滨海平原、台地和丘陵为主，属南亚热带海洋性季风气候，温和多雨。厦门市是国务院批复确定的中国经济特区和东南沿海重要的中心城市、港口及风景旅游城市。厦门市的主要旅游景点包括鼓浪屿、南普陀寺、厦门大学、环岛路等，这些景点每年都吸引了大量的国内外游客，产生了丰富的旅游数据。

## 1.2数据类型与特点

从去哪儿网上获取的厦门市旅游数据主要包括以下几种类型：

攻略数量：数量可以反映该景点的受欢迎程度和游客的关注度。攻略数量较多的景点通常是热门景点，游客体验和分享的意愿较高。

评论数量：评论数量也反映了景点的受欢迎程度，并且评论内容可以用于情感分析和满意度评估。评论数量较多的景点通常有更多的游客反馈信息。

星级：星级是游客对景点整体满意度的直接反映。高星级景点一般意味着游客体验较好，满意度较高；低星级景点可能存在服务或体验上的不足。

排名：排名可以直观反映景点的综合受欢迎程度和在游客心目中的位置。排名靠前的景点通常是游客必去的打卡地。

# 爬取过程与数据处理

## 2.1导入主要库

os：用于操作系统相关功能，例如检查文件是否存在。

hashlib：用于生成MD5哈希值，生成景点名称的唯一键。

pandas：用于处理和操作数据，特别是将数据存储到CSV文件中。

requests：用于发送HTTP请求，从目标网页获取数据。

BeautifulSoup：用于解析HTML文档，提取需要的数据。

time：用于在请求之间添加延时，防止过于频繁的请求。

## 2.1核心函数

该函数用于从指定的URL爬取页面数据，并提取景点信息。

**发送请求和获取响应：**

1. response = requests.get(url=url)
2. page\_text = response.text
3. html = BeautifulSoup(page\_text, 'lxml')

**解析HTML文档：**

1. li\_list = html.find('div', class\_='listbox').find('ul').find\_all('li)

这一步从HTML文档中找到包含景点信息的列表。

**提取景点信息：**

对每一个`li`标签，提取景点的详细信息，包括名称、攻略数量、评论数量、星级、排名、简介、链接和图片。

1. name = li.find('a', class\_='titlink').text
2. strategy\_sum = li.find('div', class\_='strategy\_sum').text
3. comment\_sum = li.find('div', class\_='comment\_sum').text
4. cur\_star = li.find('span', class\_='cur\_star')['style'].split(':')[1].replace('%', '')
5. ranking\_sum = li.find('span', class\_='ranking\_sum').text
6. desbox = li.find('div', class\_='desbox').text
7. link = li.find('a', class\_='imglink')['href']
8. img = li.find('a', class\_='imglink').find('img')['src']

**构建字典并转换为DataFrame：**

1. citydata = {
2. 'key': md5(name.encode('utf-8')).hexdigest(),
3. '城市': city,
4. '景点名称': name,
5. '攻略数量': strategy\_sum,
6. '评论数量': comment\_sum,
7. '星级': cur\_star,
8. '排名': ranking\_sum,
9. '简介': desbox,
10. '链接': link,
11. '图片': img
12. }
13. df = pd.DataFrame(citydata, index=[0])

**将数据追加到CSV文件：**

1. df.to\_csv('data.csv', encoding='utf-8-sig', index=False, header=False, mode='a+')
2. **print**(citydata)

## 2.2主程序

**创建CSV文件**

1. filename = 'data.csv'
2. df = pd.DataFrame(columns=['key', '城市', '景点名称', '攻略数量', '评论数量', '星级', '排名', '简介', '链接', '图片'])
4. **if** **not** os.path.exists(filename):
5. df.to\_csv(filename, encoding='utf-8-sig', index=False)

**定义爬取的城市和URL**

这里定义了厦门市的旅游景点URL，并设置了最大页数（max\_page）。

1. urls = [
2. {'city': '厦门', 'url': 'https://travel.qunar.com/p-cs299782-xiamen-jingdian-1-{page}', 'page': 0, 'max\_page': 3},
3. ]

**循环遍历城市列表并爬取数据**

对于每一个城市，循环遍历每一页，调用get\_page函数进行数据爬取，并在每次请求后等待5秒钟。

1. **for** url **in** urls:
2. **while** url['page'] <= url['max\_page']:
3. url['page'] += 1
4. get\_page(url['url'].format(page=url['page']), url['city'])
5. **print**('\r[{city}] 第{page}页/共{max\_page}页'.format(city=url['city'], page=url['page'], max\_page=url['max\_page']), end=' ')
6. time.sleep(5)

## 2.3爬取结果



# 3.内容分析

## 3.1数据文本预处理

定义两个函数 tokenize 和 remove\_stopwords，分别用于分词和去除停用词。

使用 jieba 进行分词，并将结果保存在 tokenized\_intro 列中。

从 stopwords.txt 文件中加载停用词，并去除分词结果中的停用词，保存为 cleaned\_intro 列。

1. **import** pandas as pd
2. **import** jieba
3. **from** collections **import** Counter
4. **from** wordcloud **import** WordCloud
5. **import** matplotlib.pyplot as plt
6. **from** sklearn.feature\_extraction.text **import** CountVectorizer
7. **from** sklearn.decomposition **import** LatentDirichletAllocation
8. **from** snownlp **import** SnowNLP
10. # 读取数据
11. data = pd.read\_csv('data.csv')
13. # 检查和清理数据
14. data.drop\_duplicates(inplace=True)
15. data.dropna(inplace=True)
16. # 文本预处理
17. **def** tokenize(text):
18. **return** ' '.join(jieba.cut(text))
20. **def** remove\_stopwords(text, stopwords):
21. **return** ' '.join([word **for** word **in** text.split() **if** word **not** **in** stopwords])
23. # 分词
24. data['tokenized\_intro'] = data['简介'].apply(tokenize)
26. # 加载停用词
27. with open('stopwords.txt', 'r', encoding='utf-8') as f:
28. stopwords = set(f.read().splitlines())
30. # 去停用词
31. data['cleaned\_intro'] = data['tokenized\_intro'].apply(**lambda** x: remove\_stopwords(x, stopwords))

## 3.2主要模型与情感分析

使用 CountVectorizer 将文本转换为词频矩阵（Document-Term Matrix, DTM），设置 max\_df 和 min\_df 来过滤掉太常见或太少见的词。

使用 LatentDirichletAllocation (LDA) 进行主题模型分析，设置主题数为5。

定义 display\_topics 函数，输出每个主题的前10个关键词。

使用 SnowNLP 对每条简介进行情感分析，结果保存在 sentiment 列中。SnowNLP 返回一个介于0到1之间的值，表示情感的积极程度，越接近1表示情感越积极。

1. # 主题模型分析
2. vectorizer = CountVectorizer(max\_df=0.95, min\_df=2)
3. dtm = vectorizer.fit\_transform(data['cleaned\_intro'])
5. lda = LatentDirichletAllocation(n\_components=5, random\_state=0)
6. lda.fit(dtm)
8. **def** display\_topics(model, feature\_names, no\_top\_words):
9. **for** topic\_idx, topic **in** enumerate(model.components\_):
10. **print**(f"Topic {topic\_idx}:")
11. **print**(" ".join([feature\_names[i] **for** i **in** topic.argsort()[:-no\_top\_words - 1:-1]]))
13. display\_topics(lda, vectorizer.get\_feature\_names\_out(), 10)
15. # 情感分析
16. data['sentiment'] = data['简介'].apply(**lambda** x: SnowNLP(x).sentiments)

# 6.总结

通过主题分析，明确了各景点的特色和游客关注点，有助于精准营销。情感分析结果为景点改进提供了依据，提升游客满意度。社会网络模型揭示了景点之间的连接关系，优化建议包括保护关键节点和提升弱连接的景点。

本项目利用Python和Gephi对厦门市旅游数据进行了全面分析，通过主题和情感分析以及社会网络建模，得出了对厦门旅游景点的深入理解和优化建议。这些结果不仅为厦门旅游管理部门提供了数据支持，也为未来的旅游推广和服务改进指明了方向。

未来将会持续收集和更新旅游数据，扩大数据来源以覆盖更多景点。进一步优化主题和情感分析模型，提升分析精度和深度。探索使用更多先进的数据分析技术，如机器学习和深度学习，提高分析能力和效果。

通过不断的研究和改进，本项目旨在为厦门市旅游业的发展提供持续的数据支持和优化方案，提升整体旅游体验。