西安交通大學

XI'AN JIAOTONG UNIVERSITY

新闻与新媒体学院

python 编程语言课程报告

共青团中央微博数据分析

姓名：杨忠强

学号：2215310366

2023 年 7 月 25 日

摘要

微博作为快速便捷的社交媒体平台，在网民舆论思潮中发挥着重要的信息引导作。本次作业分析微博平台共青团中央账号的发微情况。以 2019 年至今其所发布的所有微博作为主要研究对象，统计分析其发微规律与时间构成情况，同时利用 jieba 分词提取所发微博中的高频词汇，分词后通过 LDA 主题模型分析其所发微博的几大主题构成，最后通过 pyecharts 制作可视化展现相应的结果提高交互性。

关键词: 微博，共青团中央，数据分析，可视化

目录

摘要 **I**

第 1 章 目标简介 **1**
1.1 数据爬取 1
1.2 数据分析可视化 1
1.3 运行环境 2

第 2 章 数据爬取 **3**
2.1 爬取对象 3
2.2 爬虫实现 3
2.3 数据采集结果 5

第 3 章 数据预处理 **6**
3.1 数据去重与格式转换 6
3.2 文本内容清洗 7

第 4 章 数据分析与可视化 **8**
4.1 逐年发微热力图 8
4.2 转赞评变化 10
4.3 发微时间统计 12
4.3.1 逐年统计 12
4.3.2 每年逐月统计 13
4.3.3 每月逐天统计 14
4.3.4 每周逐天统计 15
4.3.5 每天逐时统计 15
4.4 周时热力统计 16
4.5 高频词统计 17
4.6 LDA 主题分析 18

第 5 章 总结展望 **21**

第 1 章　目标简介

本报告目的是抓取社交媒体平台数据，并进行计算分析，而后对所爬取数据进行分析和可视化展示。

1.1　数据爬取

网络爬虫的主要目的是将互联网上的网页下载到本地形成一个或联网内容的镜像备份，通过对内容的解析之后格式化处理进行存储。本报告使用爬虫项目所用框架为 scrapy。

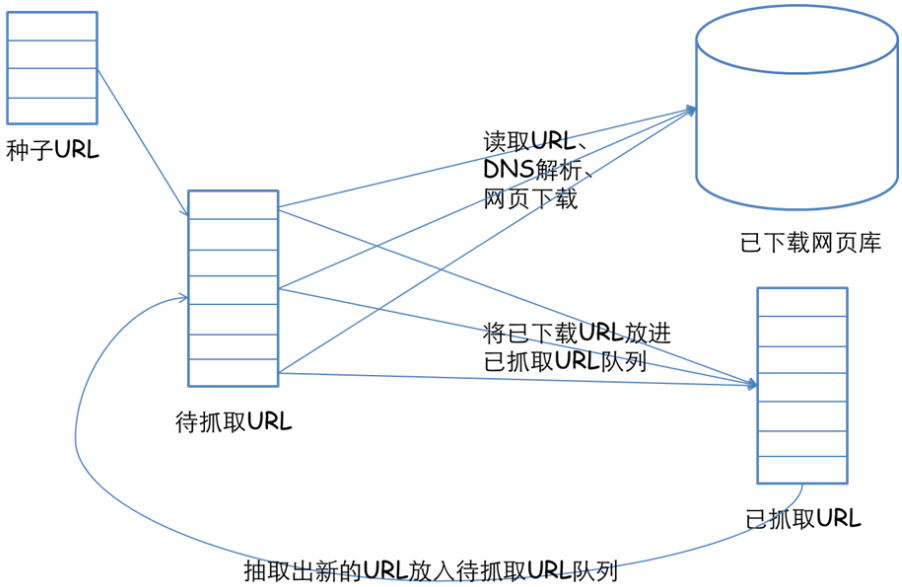

图 1.1: 爬虫示意图

1.2　数据分析可视化

本报告使用 pandas、jieba 等库进行数据处理，具体方案将在后文阐述。可视化工具使用为 pyecharts1.2，可视化数据生成 html 文件后能够较好实现交互。

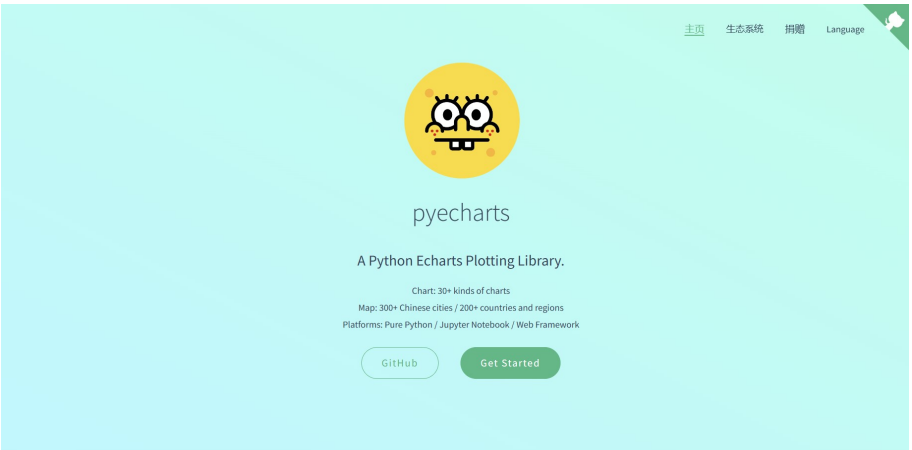

图 1.2: pyecharts

1.3 运行环境

本报告在 pycharm（Community Edition 2022.2.2）的 windows 版本下运行，环境配置为 python3.10，界面如图 1.3 所示。

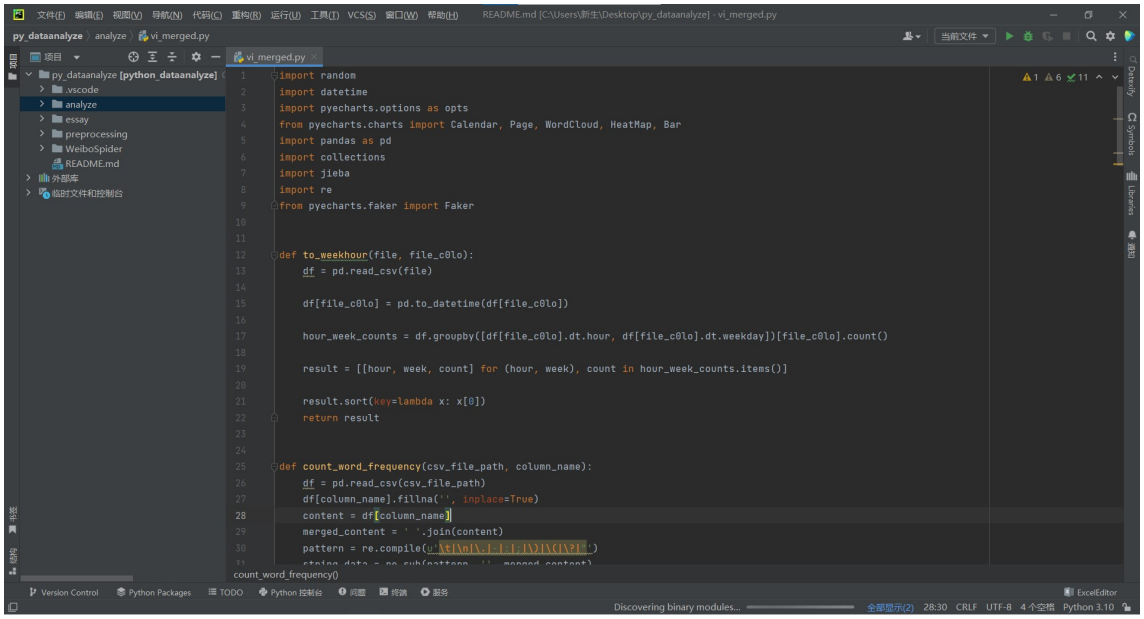

图 1.3: pycharm 运行环境示意图

第 2 章　数据爬取

2.1　爬取对象

本报告选取共青团中央的官方微博如图 2.1，爬取时间范围为 2019 年 1 月 1 日到 2023 年 7 月 16 日。

图 2.1: 共青团中央官方微博

2.2　爬虫实现

本报告数据爬取使用 Github 基于 weibo.com 的新版 API 构建的爬虫项目，项目地址为：https://github.com/nghuyong/WeiboSpider，项目主页如图2.2所示。主要使用该项目下的 tweet.py 基于用户 id 进行爬虫，项目具体使用可参照其 README 文件。

图 2.2: 项目主页示例

该项目将爬取数据解析后存储为 json 格式，其单条数据示例如下所示。


```
{
    "_id": "4762810834227120",
    "mblogid": "LqlZNhJFm",
    "created_at": "2022-04-27 10:20:54",
    "geo": null,
    "ip_location": null,
    "reposts_count": 1890,
    "comments_count": 1924,
    "attitudes_count": 12167,
    "source": "三星Galaxy S22 Ultra",
    "content": "生于乱世纵横四海，义之所在不计生死，孤勇者陈恭一生当如是。#风起陇西
                今日开播# #风起陇西#  今晚，恭候你！"
                ,
    "pic_urls": [],
    "pic_num": 0,
    "isLongText": false,
    "user": {
        "_id": "1087770692",
        "avatar_hd": "https://tvax1.sinaimg.cn/crop.0.0.1080.1080.1024/
                     40d61044ly8gbhxwgy419j20u00u0goc.
                     jpg?KID=imgbed,tva&Expires=
                     1682768013&ssig=r1QurGoc2L",
        "nick_name": "陈坤",
        "verified": true,
        "mbrank": 7,
        "mbtype": 12,
        "verified_type": 0
    },
    "video": "http://f.video.weibocdn.com/o0/CmQEWK1ylx07VAm0nrxe01041200YDIc0E010.
              mp4?label=mp4_720p&template=1280x720.
```

```
                                                    25.0&ori=0&ps=1CwnkDw1GXwCQx&Expires=
                                                    1682760813&ssig=26udcPSXFJ&KID=
                                                    unistore,video",
25      "url": "https://weibo.com/1087770692/LqlZNhJFm",
26      "crawl_time": 1682757213
27  }
```

2.3 数据采集结果

本报告共采集到共青团中央从 2019 年 1 月 1 日至 2023 年 7 月 16 日发微数据 25244 条，采集结果如图2.3所示。

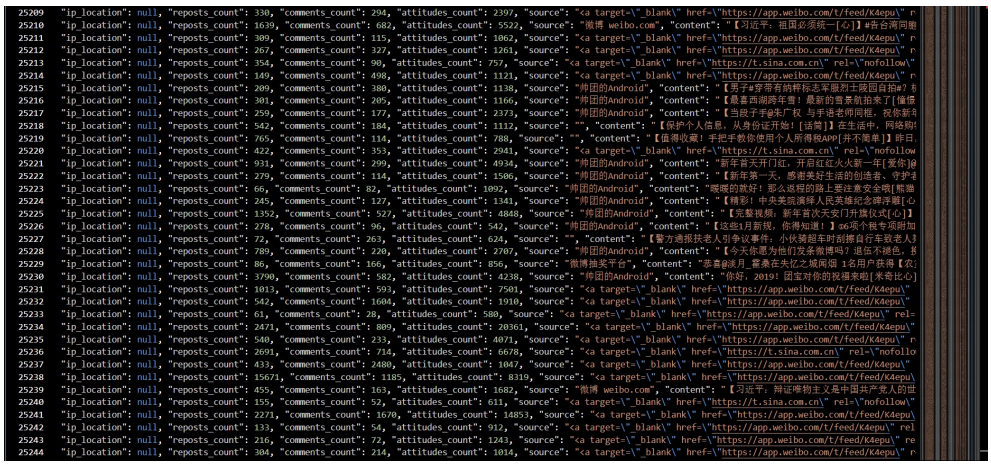

图 2.3: 数据采集结果

第 3 章　数据预处理

与一般的数据处理全流程类似，本报告的实现过程主要包括数据的爬取、数据预处理、计算结果分析和数据的可视化展示等几个主要步骤。

3.1　数据去重与格式转换

首先，将 json 文件转换为 csv 格式，先写入表头，后循环读取每一行的数据的 key 值所对应的 value 后输出，对应函数实现如下。

```
1  def json_to_csv(input_file, output_file):
2      with open(input_file, 'r', encoding='utf-8') as json_file, open(output_file, 'w
                                                ', encoding='utf-8',
3                                                                    newline='') as
                                                                                 
                                                                                 
                                                                                 
4          writer = csv.writer(csv_file)
5          writer.writerow(['user_id', 'mblogid', 'created_at', 'reposts_count', '
                                                  comments_count', 'attitudes_count
                                                  ',
6                           'content', 'pic_urls', 'pic_num', 'isLongText', 'url', '
                                                                  crawl_time'])
7  
8          for line in json_file:
9              data = json.loads(line)
10             user_id = data['_id']
11             mblogid = data['mblogid']
12             created_at = data['created_at']
13             reposts_count = data['reposts_count']
14             comments_count = data['comments_count']
15             attitudes_count = data['attitudes_count']
16             content = data['content'].replace('\n', ' ')  # 将换行符替换为空格，避
                                                    免输出时自动换行
17             pic_urls = data['pic_urls']
18             pic_num = data['pic_num']
19             islongtext = data['isLongText']
20             url = data['url']
21             crawl_time = data['crawl_time']
22 
23             writer.writerow([user_id, mblogid, created_at, reposts_count,
                                                    comments_count,
                                                    attitudes_count,
24                                 content, pic_urls, pic_num, islongtext, url,
                                                                  crawl_time])
```

此外，使用 pandas 的内置函数 drop_duplicates 进行去重处理并使用 sort_value 实现按照'created_at' 列的时间降序排列，核心代码如下。

```
df = pd.read_csv('gqt_output.csv')
df.drop_duplicates(inplace=True)
df = df.sort_values('created_at', ascending=False)
df.to_csv('gqt_output.csv', index=False)
```

3.2 文本内容清洗

定义正则表达式匹配去除 content 中的无效文本。

```
def clean_text(text):
    text = re.sub(r'【[^】【]+】', '', text)
    text = re.sub(r'\[[^\]\[]+\]', '', text)
    text = re.sub(r"(回复)?(//)?\s*@\S*?\s*(:|：|$)", " ", text)  # 去除正文中的@和回复/转发中的用户名
    text = re.sub(r'\([^)(]+\)', '', text)
    text = re.sub(r'\（[^）（]+\）', '', text)
    text = re.sub(r"#([^#]+)#", "", text)
    text = re.sub(r"   ", "", text)
    text = text.replace("\n", " ")
    text = re.sub(r"↓↓", "", text)
    text = re.sub(r"(\s)+", r"\1", text)
    link_regex = r'http://t\.cn/[\w]+(,)?(?=[^\w]|,|$)'
    text = re.sub(link_regex, '', text)
    stop_terms = ['展开', '全文', '展开全文', '一个', '网页', '链接', '?【', 'ue627', 'c【', '10', '一下', '一直',
                  'u3000', '24', '12',
                  '30', '?我', '15', '11', '17', '?\\', '显示地图', '原图']
    for x in stop_terms:
        text = text.replace(x, "")
    return text.strip()
```

第 4 章　数据分析与可视化

本报告使用可视化库为 pyecharts，能够在 web 端实现数据交互查看, 最终 html 文件位于 ananlyze 目录下，预览动态交互可运行 vi_merged.html 可查看，以下部分仅展示静态图片。

4.1　逐年发微热力图

统计分析每年中月日发微博频率，主要代码如下。

```
df = pd.read_csv('cleaned_text.csv', parse_dates=['created_at'])
date_counts = df['created_at'].dt.date.value_counts()
cal_result = [(date, count) for date, count in date_counts.items()]
page = Page(page_title='共青团中央微博分析', layout=Page.DraggablePageLayout)
for year in range(2019, 2024):
    calendar = (
        Calendar()
        .add("",
             cal_result,
             calendar_opts=opts.CalendarOpts(range_=str(year)),
             daylabel_opts=opts.CalendarDayLabelOpts(name_map="cn"),
             monthlabel_opts=opts.CalendarMonthLabelOpts(name_map="cn"),
             )
        .set_global_opts(
            title_opts=opts.TitleOpts(title=f"{year} 年每天发微频率"),
            visualmap_opts=opts.VisualMapOpts(
                max_=55, min_=5, orient="horizontal", is_piecewise=True,
                                                      pos_top="230px",
                                                      pos_left="100px",
            ),
        )
    )
```

逐年可视化热力图如下。

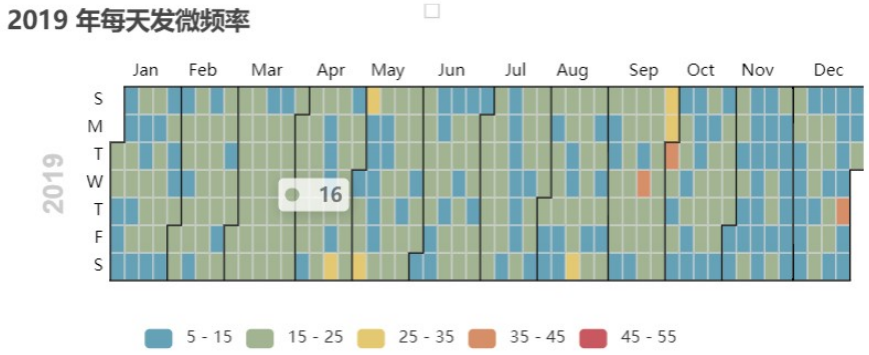

图 4.1: 2019 热力图

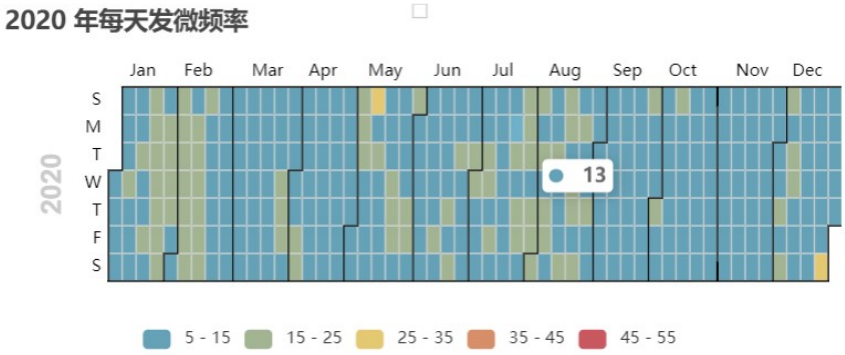

图 4.2: 2020 热力图

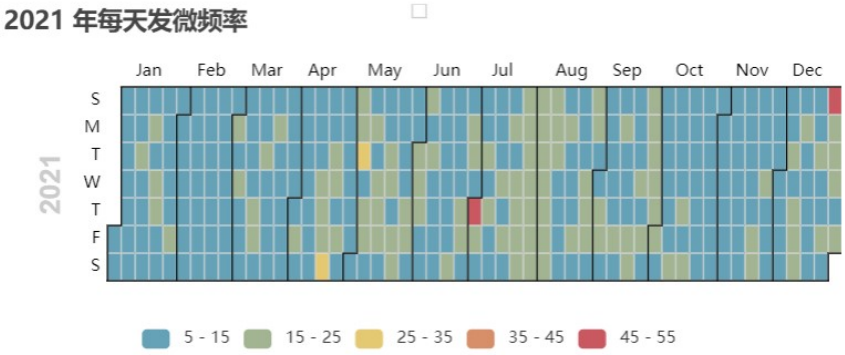

图 4.3: 2021 热力图

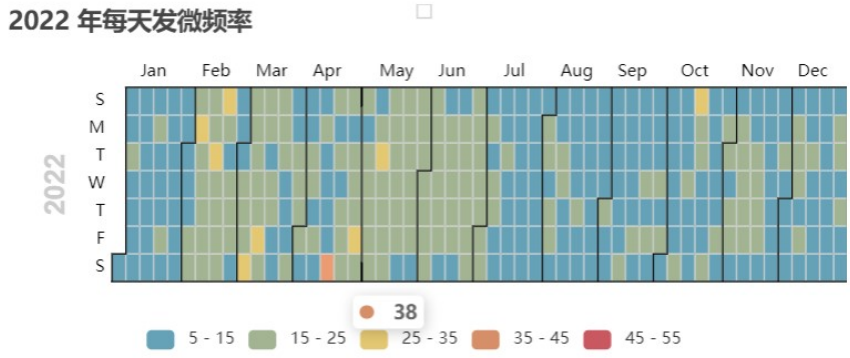

图 4.4: 2022 热力图

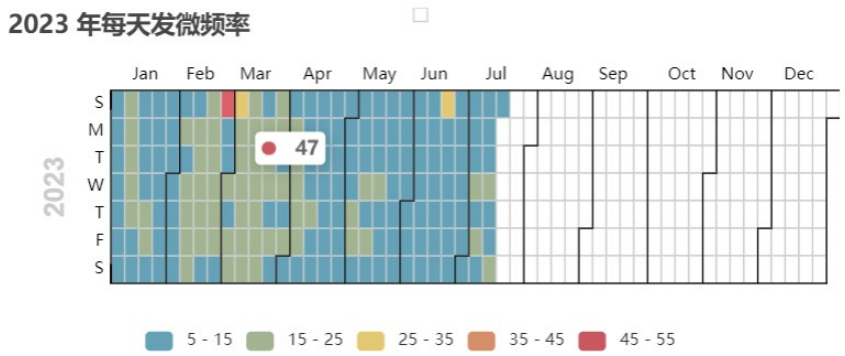

图 4.5: 2023 热力图

4.2 转赞评变化

统计 2019 年 1 月至 2023 年 7 月，共青团中央所发微博转发、评论、点赞数量随时间的变化情况，并配置 datazoom_opts 实现时间拖动缩放。

```
df = pd.read_csv('cleaned_text.csv')
df['created_at'] = pd.to_datetime(df['created_at'])
timeData = df['created_at'].dt.strftime('%Y-%m-%d %H:%M:%S').tolist()
repostsData = df['reposts_count'].tolist()
commentsData = df['comments_count'].tolist()
attitudesData = df['attitudes_count'].tolist()
line = (
    Line()
    .add_xaxis(xaxis_data=timeData)
    .add_yaxis(
        series_name="转发",
        y_axis=repostsData,
        linestyle_opts=opts.LineStyleOpts(width=1.5),
    )
    .add_yaxis(
        series_name="评论",
        y_axis=commentsData,
        linestyle_opts=opts.LineStyleOpts(width=1.5),
    )
    .add_yaxis(
        series_name="点赞",
        y_axis=attitudesData,
        linestyle_opts=opts.LineStyleOpts(width=1.5),
    )
    .set_global_opts(
        tooltip_opts=opts.TooltipOpts(trigger="axis"),
        xaxis_opts=opts.AxisOpts(
            type_="category",
            axispointer_opts=opts.AxisPointerOpts(
                is_show=True, label=opts.LabelOpts(formatter=JsCode("function(
```

```
                                                                    params) {return
                                                                    params.value; }"))
31                  ),
32              ),
33              yaxis_opts=opts.AxisOpts(name="数量"),
34              legend_opts=opts.LegendOpts(pos_top='5%'),
35              title_opts=opts.TitleOpts(title="转发、评论、点赞数量随时间变化"),
36              datazoom_opts=[opts.DataZoomOpts()],
37          )
38      )
```

最终可视化效果实现如下图4.6所示。

图 4.6: 转赞评变化情况

可拖动查看具体时间内的转赞评数量。

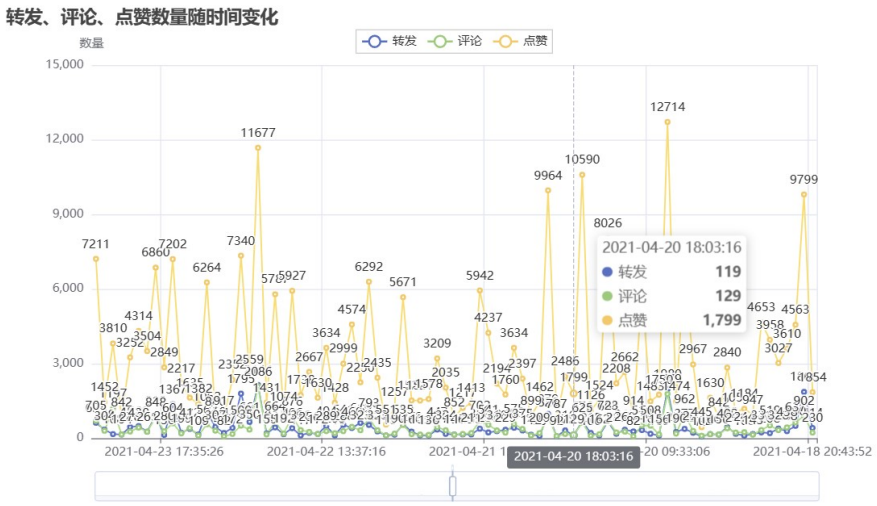

图 4.7: 具体时间缩放

4.3 发微时间统计

使用 pandas 内置函数按照年、月、周、日、时分别统计共青团中央发微时间。

```
df = pd.read_csv('cleaned_text.csv', parse_dates=['created_at'])
dateyear_counts = df['created_at'].dt.year.value_counts()
datemonth_counts = df['created_at'].dt.month.value_counts()
dateweek_counts = df['created_at'].dt.weekday.value_counts()
dateday_counts = df['created_at'].dt.day.value_counts()
datehour_counts = df['created_at'].dt.hour.value_counts()
```

4.3.1 逐年统计

利用 pyecharts 制作逐年统计可视化，结果如图4.8所示。

```
c_year = (
    Bar()
    .add_xaxis([i[0] for i in resultyear])
    .add_yaxis("共青团中央", [i[1] for i in resultyear])
    .set_global_opts(
        title_opts=opts.TitleOpts(title="每年发微频率"),
        yaxis_opts=opts.AxisOpts(name="次数"),
        xaxis_opts=opts.AxisOpts(name="时间"),
    )

)
```

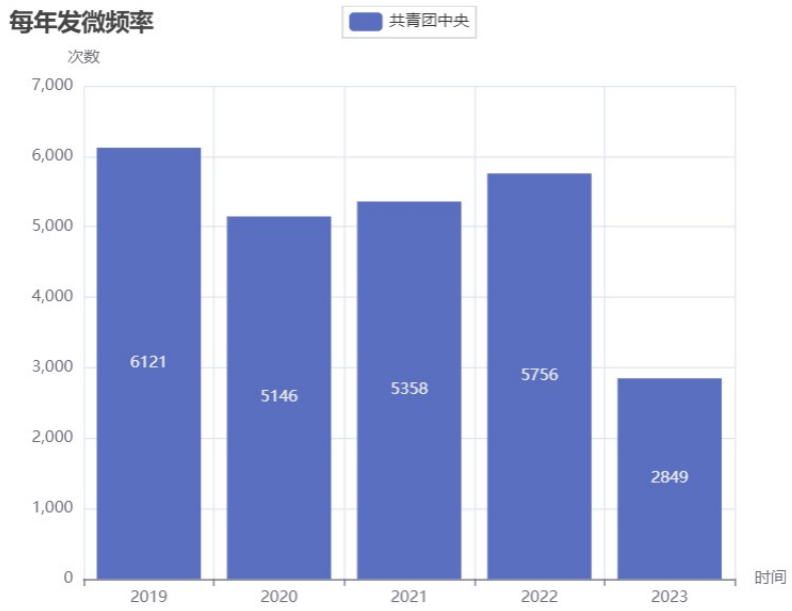

图 4.8: 逐年统计

4.3.2 每年逐月统计

逐月统计可视化。

```
    c_month = (
        Bar()
        .add_xaxis(Faker.months)
        .add_yaxis("共青团中央", [i[1] for i in resultmonth])
        .set_global_opts(
            title_opts=opts.TitleOpts(title="每月发微频率"),
            yaxis_opts=opts.AxisOpts(name="次数"),
            xaxis_opts=opts.AxisOpts(name="时间"),
        )

    )
```

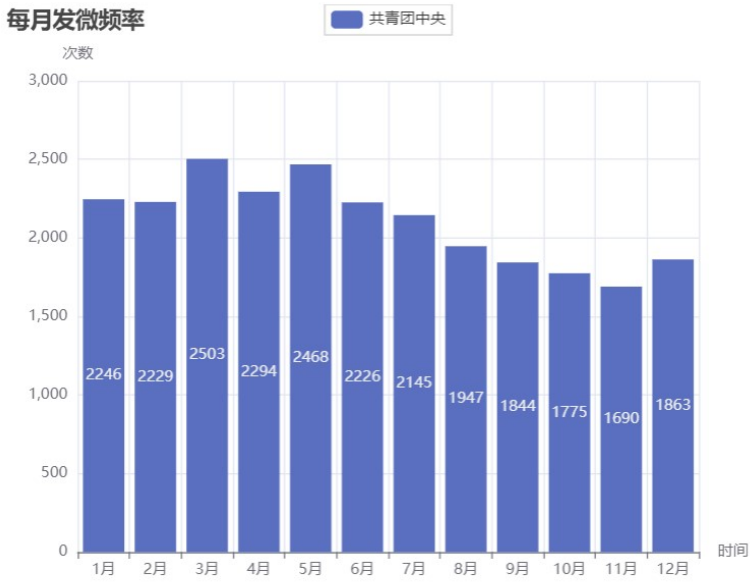

图 4.9: 逐月统计

4.3.3 每月逐天统计

本小节后续部分与上述实现过程相似，不再赘述，仅展示最终可视化结果。

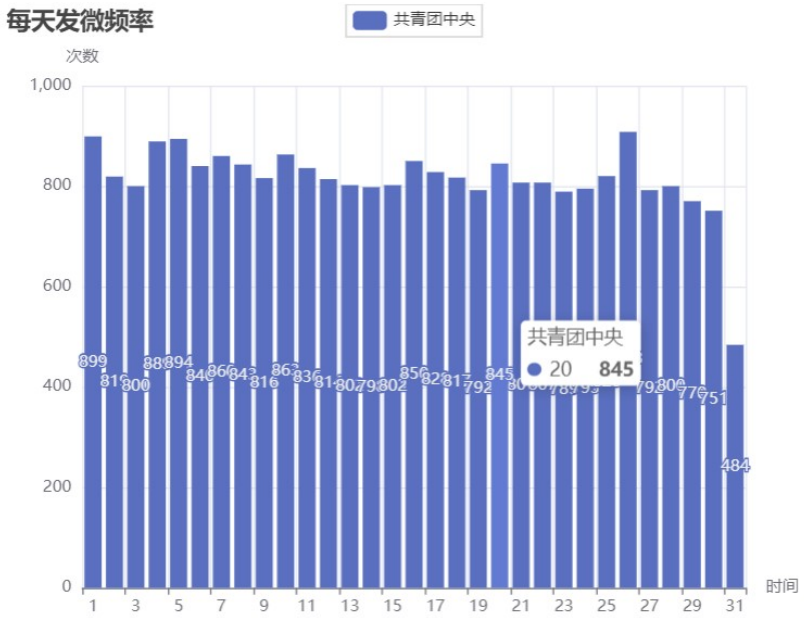

图 4.10: 逐天统计

4.3.4 每周逐天统计

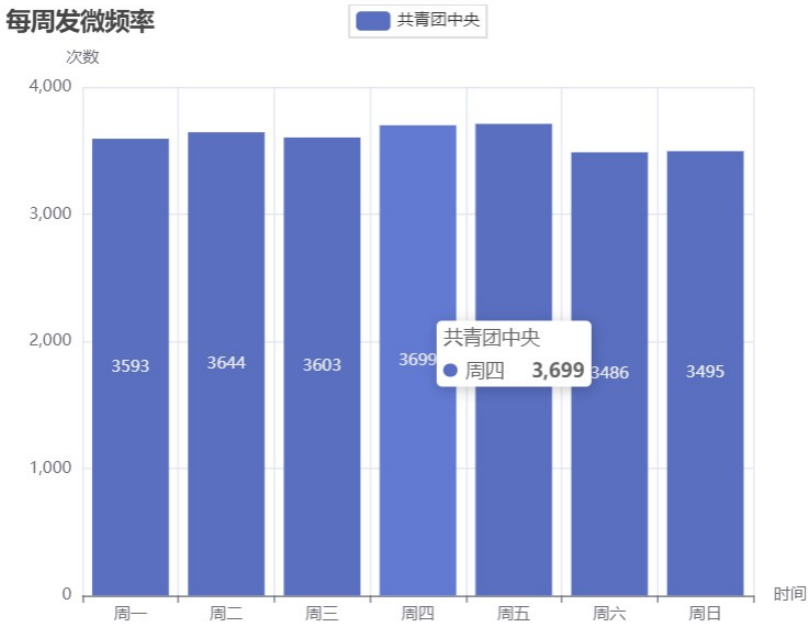

图 4.11: 逐周统计

4.3.5 每天逐时统计

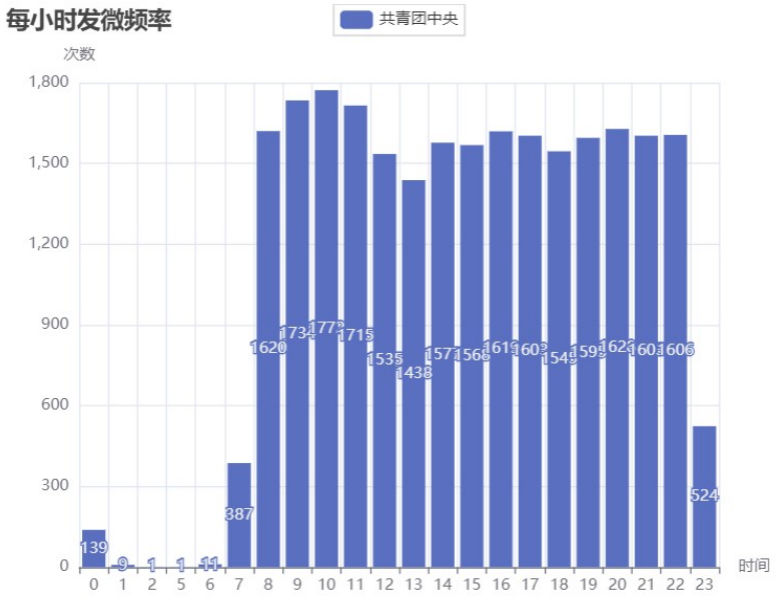

图 4.12: 逐时统计

4.4 周时热力统计

使用 pandas 的 groupby 多列分组先按照 hour 来分组。每组内，再按 weekday 来分组, 并统计每个周时数量返回。

```
def to_weekhour(file, file_colm):
    df = pd.read_csv(file)
    df[file_colm] = pd.to_datetime(df[file_colm])
    hour_week_counts = df.groupby([df[file_colm].dt.hour, df[file_colm].dt.weekday]
                                  )[file_colm].count()
    result = [[hour, week, count] for (hour, week), count in hour_week_counts.items
                                  ()]
    result.sort(key=lambda x: x[0]) #按照第一个排序
    return result
```

调用 to_weekhour 函数将分类统计结果传回 pyecharts 进行可视化绘图，核心代码如下。

```
    c_heatmap = (
        HeatMap()
        .add_xaxis(Faker.clock)
        .add_yaxis(
            "发微次数", Faker.week, to_weekhour('cleaned_text.csv', 'created_at'),
            label_opts=opts.LabelOpts(position="middle")
        )
        .set_global_opts(
            title_opts=opts.TitleOpts(title="共青团中央发微周时热力图"),
            visualmap_opts=opts.VisualMapOpts(),
        )
    )
```

最终可视化效果如图4.13

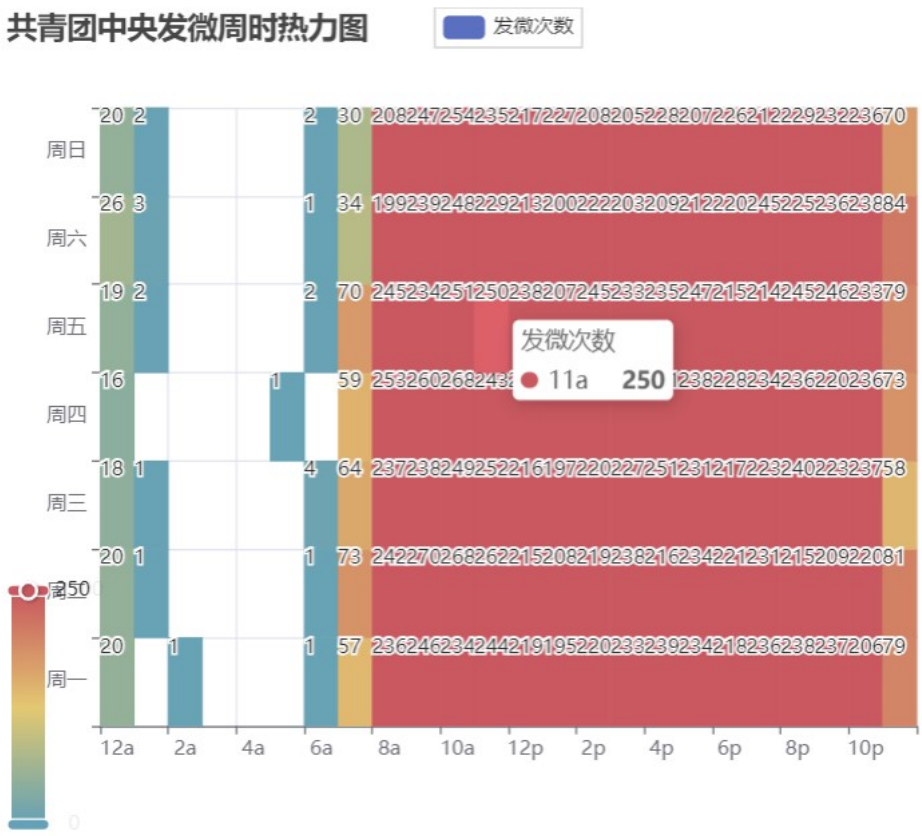

图 4.13: 周时热力图

4.5 高频词统计

定义函数统计词频，添加自定义词典和停用词词典，使用 jieba 库进行分词，jieba 分词工具有三种主要分词模式，本报告选取 **HMM** 模式。

```
def count_word_frequency(csv_file_path, column_name):
    df = pd.read_csv(csv_file_path)
    df[column_name].fillna('', inplace=True)
    content = df[column_name]
    merged_content = ' '.join(content)
    pattern = re.compile(u'\t|\n|\.|-|:|;|\)|\(|\?|"')
    string_data = re.sub(pattern, '', merged_content)
    jieba.suggest_freq('共青团', True)
    jieba.load_userdict('user_words.txt')
    seg_list_exact = jieba.cut(string_data, cut_all=False, HMM=True)  # 分词模式
    object_list = []
    # 去除停用词
    with open('stop_words.txt', 'r', encoding='UTF-8') as meaninglessFile:
        stopwords = set(meaninglessFile.read().split('\n'))
    stopwords.add(' ')
    for word in seg_list_exact:
```

```
        if word not in stopwords:
            object_list.append(word)
    word_counts = collections.Counter(object_list)
    for word in list(word_counts.keys()):
        if len(word) == 1:
            del word_counts[word]
    word_counts_top = word_counts.most_common(300)
    return word_counts_top
```

使用 pyecharts 制作词云可视化，此外本报告也采用了自定义功能更强大的 wordcloud 库结合 matplot 等库制作了共青团中央微博内容词云可视化，相关结果为摘要页微博图标背景的词云可视化图片。

```
c_wordcloud = (
    WordCloud()
    .add(
        "",
        count_word_frequency('cleaned_text.csv', 'content'),
        word_size_range=[20, 100],
        textstyle_opts=opts.TextStyleOpts(font_family="cursive"),
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="top200 词 云"))
)
```

使用 pyecharts 实现的可视化效果如图所示，能够实现在网页端点击悬停等交互。

图 4.14: 词频统计词云

4.6 LDA 主题分析

首先对文本内容进行分词，分词程序位于 analyze/LDA/cutword.py，具体实现与词

云处一致调用 jieba 库。分词后，基于已分词文档使用 gensim 库进行 LDA 主题分析，并使用 pyLDAvis 进行可视化展示。思路阐述参考下列代码注释

```
import pyLDAvis.gensim_models
from gensim.corpora.dictionary import Dictionary
from gensim.models.ldamodel import LdaModel
from gensim import corpora, models

if __name__ == '__main__':
    words_ls = []
    with open('outdemowei.txt', 'r', encoding='UTF-8') as f:
        data = f.readlines()
        for line in data:
            words_ls.append(line.split(' '))
    f.close()
    # 构造词典，将每个唯一的词语映射到一个唯一的整数ID
    dictionary = corpora.Dictionary(words_ls)
    # 基于词典，使【词】→【稀疏向量】，并将向量放入列表，形成【稀疏向量集】（即语料库）
    corpus = [dictionary.doc2bow(words) for words in words_ls]
    # tf-idf编码
    tfidf = models.TfidfModel(corpus)
    # 将语料库转换为tf-idf编码形式
    corpus_tf = tfidf[corpus]
    # lda模型，num_topics设置主题的个数
    lda = LdaModel(corpus=corpus_tf, id2word=dictionary, num_topics=4, random_state=100, iterations=50)

    for topic in lda.print_topics(num_words=10):
        print(topic)
    # 用pyLDAvis可视化
    plot = pyLDAvis.gensim_models.prepare(lda, corpus_tf, dictionary)
    pyLDAvis.save_html(plot, 'ldaweibo.html')
```

通过 LDAModel 抽取 4 个主题后，最终可视化效果如图4.15所示，html 可视化源文件位于 analyze/LDA/ldaweibo.html。

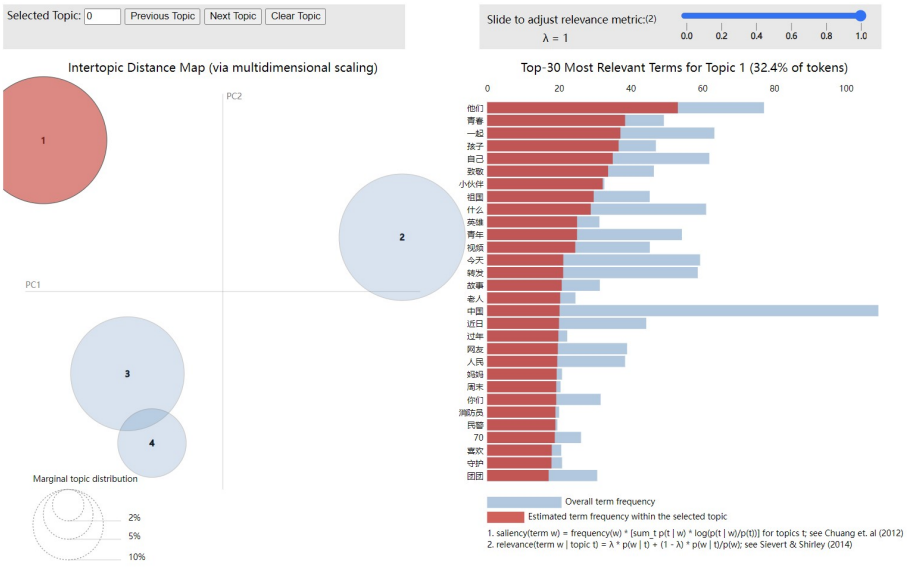

图 4.15: LDA 分析结果

第 5 章　总结展望

本文在 python3.10 环境下，基于对 Github 开源爬虫项目的改动实现数据爬取，调用了 pandas,pyecharts,jieba,gensim 等软件包，对所获得数据进行了统计分析和可视化交互展示。国内新闻传播顶刊有计算传播论文中写到利用 python 进行情感分析，使用 snownlp 进行分析，作为一个封装好的项目，再去调用已经没有难度，基础还是在于循环，判断等语句的应用。本报告并不试图去调用一些可能是用家具评论数据进行有监督机器学习的工具通过一行代码就能实现情感分析来用到自己所爬取的可能和模型预训练预料毫无关系的数据，试问这样的实现难度和简单写个函数让每一条数据调用分析后输出结果可视化有什么不同呢？背后更深的应该是整个模型实现的过程，有什么条件需要满足，能不能用到我的项目上。本报告也存在遗憾，在 LDA 分析的时候 gensim 库是做的无监督，但是确定主题数的时候没能实现一些暴力搜索之类的方法来确定主题数，还是人为尝试了很多数字之后确定的主题数，后续有时间一定解决这部分的遗憾实现编程的严谨。

写完报告主体想到读过的很多新闻传播论文，真诚希望很多文章在使用某个方法非要去赋予概念的时候能够多一点技术理性，首先理解技术逻辑、判断自己的项目是否适合使用该数据分析技术、最终分析社会问题，否则如果仅仅是冠以计算和智能的 tag 毫无判断也无法读懂方法核心为了让自己的研究显得更“高端”、方法更新，在用一个错误的手段分析社会解读社会，这个世界真的会因为所谓学术更好嘛？

老师对待计算机，对待编程，对待技术、对待社会的严谨，从中收获很多。希望国内计算传播越来越好!