这六年，一路走来……

——基于时代少年团各成员微博数据分析成员发展状况和发展差异

数据采集与可视化 课程设计

谢曜州 522120910264

也许是个摘要

时代少年团（TNT，Teens in Times），是时代峰峻推出的中国内地流行乐男子团体，由马嘉祺、丁程鑫、宋亚轩、刘耀文、张真源、严浩翔、贺峻霖 7 人组成。对于娱乐明星而言，“咖位”、热度、有没有“过气”一直是一个充满争议又让人无限好奇的话题。而明星作为公众人物，其发送的微博的习惯和方式，也是一个有趣的研究问题。

本课程设计基于 Selenium 方法抓取微博内容和相关数据、基于 Ajax 方法抓取每条微博对应评论内容和相关数据，获得了这七位明星自出道以来的微博数据，包括评论、点赞、转发、文本、@互动、话题、评论内容、评论 IP、发博时间等众多数据，分析了各明星微博的携带话题词云、粉丝属地分析、微博类型、发博方式分析、成员互动分析、微博热度分析、最爱使用表情，希望能提供一些有趣的结论，权当消遣。

目录

数据采集

该部分的重要信息的注释采用红色、倾斜、下划线标注。

1） 爬取前的准备

模块导入

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import csv,time,random,pymysql,requests,re
import json as js
import numpy as np, pandas as pd
from urllib.parse import urlencode
```

获得 cookies

```
#获取 cookies
browser_cookies=webdriver.Edge()
browser_cookies.get('https://m.weibo.cn/u/6177367279')
while(1):
    time.sleep(100)
    cookies=browser_cookies.get_cookies()
    if cookies[0]['domain']=='.weibo.cn': break
    #进入微博验证中心后，domain 会变成'.passport.weibo.com'，如果重新变为'.weibo.cn'则说明验证成功
print(cookies)
browser_cookies.quit()
```

通过团体账号某条微博的@信息 获得各个成员对应主页链接

```
browser=webdriver.Edge()
#根据时代少年团官微的@，获得每位成员的主页，并记录在列表中
#通过这样的方式，提高代码的可移植性，代码现在可以爬取任意一个偶像组合团体的数据
browser.get('https://m.weibo.cn/status/5024842221093499')
members_raw=[]
#使用 Xpath Helper 获得的 Xpath
try:
    for i in range(3,10):
        members_raw.append(browser.find_element(By.XPATH,"/html/body/div[@id='app']/div[@class='lite-page-wrap']/div/div[@class='main']/div[@class='card m-panel c
```

```
ard9 f-weibo']/article[@class='weibo-main']/div[@class='card-wrap']/div[@class='w
eibo-og']/div[@class='weibo-text']/a[{}]".format(i)))
except:
    print(f'The {i}th member has not been found.')

members=[member.get_attribute('href') for member in members_raw]
names=[member.text for member in members_raw]
names=[name[-3:] for name in names]
print(names)
```

2) 基于 Selenium 方法抓取微博内容和相关数据

进入循环并初始化 csv 行

从每一个微博卡片中获得的数据包括，微博的**发送时间**、**发送方式**（如微博视频号、绿洲等）、**微博的文本内容**、**微博的@对象**、**微博带有的话题**（#微博话题#）、**微博中使用的表情**、**微博的类型**（文本型、照片型（照片数量、动图数量、长图数量）、视频型（视频长度、播放量）、明星动态型、卡片型（卡片标题、卡片内容）、转发型（转发的文本内容、转发来源、转发的@对象、转发带有的话题（#微博话题#）、转发中使用的表情、转发的类型（文本型、照片型（照片数量、动图数量、长图数量）、视频型（视频长度、播放量）、卡片型（卡片标题、卡片内容））），**微博的转发量**、**评论量**和**点赞量**）。

```
for i in range(len(members)):
    errorlist=[]
    csvfile=open('{}.csv'.format(names[i]),'wt+',encoding='utf-8',newline='')
    writer=csv.writer(csvfile)
    writer.writerow(['time','from','text','at','topic','url-text','url-icon','img
_count','gif_count','long_image_count','video','video_view','video_length','recen
t-activity','card','card_text','card_subtext','repost', 'repost_text','repost_fro
m', 'repost_at', 'repost_topic', 'repost_url-text', 'repost_url-icon', 'repost_im
g_count', 'repost_gif_count', 'repost_long_image_count', 'repost_video',  'repost
_video_view', 'repost_video_length','repost_recent-activity', 'repost_card', 'rep
ost_card_text', 'repost_card_subtext','forwards','comments','likes'])
    browser.get(members[i])
    #添加 cookies
    for cookie in cookies:
        browser.add_cookie(cookie)
```

通过模拟下拉操作采集完整时段微博数据

```
        #使用页面高度 height 作为采集信息是否完全的判断依据
    height=[0,0]
     # j=0
     # while(j<10):
     while(1):
         height.append(browser.execute_script("return document.body.scrollHeight;
"))
         browser.execute_script("window.scrollBy(0, 5000);")
         time.sleep(2)
         # 使用最后三次下拉的 height 作为判断依据，如果拉到底证明已经采集完成。
        # 由于只使用最后两次的 height 可能导致提前停止，故采用两次机会算法。
        if height[-1]==height[-2] and height[-3]==height[-2]:break
         # j+=1

     #获得每一个微博卡片
    weibos=browser.find_elements(By.CLASS_NAME,'card-main')
     count=0
```

微博卡片分析和存储

```
        #分析每一个微博卡片
    for weibo in weibos:
         newrow=[]
  #出于方便阅读的需要调整了缩进
try:
     count+=1
```

表头数据分析

```
     #表头数据，获得该条微博发送时间
    time_=weibo.find_element(By.CLASS_NAME,'time')
     newrow.append(time_.text)
     #一部分微博发送的时候不显示发送来源
    try:
         from_=weibo.find_element(By.CLASS_NAME,'from')
         from_=from_.text
         newrow.append(from_.replace('来自 ',''))
     except:
         newrow.append('')
```

微博文本内容（weibo-text）分析采集

```
 #拆分出微博内容数据
og=weibo.find_element(By.CLASS_NAME,'weibo-og')

 #拆分处文本数据
text=og.find_element(By.CLASS_NAME,'weibo-text')
 newrow.append(text.text)
 #查找@，一部分微博没有@
 try:
     ats=text.find_elements(By.PARTIAL_LINK_TEXT,'@')
     atss=''
     for at in ats:
         atss+=at.text
     newrow.append(atss)
 except:
     newrow.append('no@')

 #查找微博话题，一部分微博没有##
 try:
     topics=text.find_elements(By.PARTIAL_LINK_TEXT,'#')
     topicss=''
     for topic in topics:
         topicss+=topic.text
     newrow.append(topicss)
 except:
     newrow.append('no#')

 #查找是否有微博链接，一部分微博没有链接
try:
     surls=text.find_elements(By.CSS_SELECTOR, 'span.surl-text')
     surls=list(set(surls+topics+ats)-set(topics)-set(ats))
     surlss=''
     for surl in surls:
         surlss+=surl.text
     newrow.append(surlss)
 except:
     newrow.append('nourl')

 #查找是否有微博表情，一部分微博没有表情
try:
     icons=text.find_elements(By.CSS_SELECTOR, 'span.url-icon')
     alts=''
```

```
        for icon in icons:
            img=icon.find_element(By.CSS_SELECTOR, 'img')
            alts += img.get_attribute('alt')
        newrow.append(alts)
    except:
        newrow.append('noicon')
```

微博媒体内容（类型）分析采集

```
 #content 表示微博文本之后的数据，如照片、视频，一部分微博可能只有文本，但是也会有空的 div，因此不需要 try
    content=og.find_elements(By.CSS_SELECTOR,'div')[1]

    #查找图片
    try:
        imgs=content.find_elements(By.CLASS_NAME,'m-img-box')
        if_single=content.find_elements(By.CLASS_NAME,'single-img')
#由于其他的内容，比如 video 也会有 img_box，因此需要通过更多的步骤判断是否是图片类型的微博
#如果有两张及以上图片的微博，会有两个 class_为 img_box 的 div，这是独特的性质，判断方式是 len(imgs)>1
#如果是只有一张图的微博，会有特殊的 single-img 类名，判断方式是(len(imgs)==1 and len(if_single)==1)
        if (len(imgs)==1 and len(if_single)==1) or len(imgs)>1:
            newrow.append(len(imgs))

            special_imgs=content.find_elements(By.CLASS_NAME,'feed-mark')
            special_img_tags=[special_img.text for special_img in special_imgs]

            gif_count=0
            long_image_count=0

            for special_img_tag in special_img_tags:
                if special_img_tag=='动图':gif_count+=1
                else:
                    if special_img_tag=='长图':long_image_count+=1
                    else:
                        print(special_img_tag)
            newrow.append(gif_count)
            newrow.append(long_image_count)

        else:
```

```
            for k in range(3):
                newrow.append('noimg')
    except:
        for k in range(3):
            newrow.append('noimg')

    #查找视频
    try:
        video_wrap = content.find_element(By.CSS_SELECTOR, 'div.weibo-media-wraps
')
        video = video_wrap.find_element(By.CSS_SELECTOR, 'div.card-video')
        video=video.text
        newrow.append(1)
        #一部分视频只有播放量，没有播放时长，还有一部分两个都没有，这是在爬取过程中发现的新问题
       try:
            newrow.append(video.split()[0])
        except IndexError:
            newrow.append('noview')
        try:
            newrow.append(video.split()[1])
        except IndexError:
            newrow.append('nolength')

    except:
        for k in range(3):
            newrow.append('novideo')

    #查找是否明星动态
    try:
        activity_wrap=content.find_element(By.CSS_SELECTOR, 'div.weibo-media-wrap
s')
        recent_activity=activity_wrap.find_element(By.CSS_SELECTOR,'div.type-bigp
ic')
        newrow.append(1)
    except:newrow.append('noact')
    #是否卡片
    try:
        card_wrap_wrap = content.find_element(By.CSS_SELECTOR, 'div.weibo-media-w
raps')
        card_wrap = card_wrap_wrap.find_element(By.CSS_SELECTOR, 'div.card-wrap')
        card = card_wrap.find_element(By.CSS_SELECTOR, 'div.m-box')
        card_box = card_wrap.find_element(By.CSS_SELECTOR, 'div.m-box-col')

        newrow.append(1)
```

```
            card_texts = card_box.text
            #一部分卡片的话题使用的是空格，只有换行符能作为区分
            card_texts = card_texts.split('\n')
            if len(card_texts) <= 1:
                newrow.append(card_texts[0])
                newrow.append('')
            else:
                cardss=''
                newrow.append(card_texts[0])
                for card in card_texts[1:]:
                    cardss+=card
                newrow.append(cardss)
        except:
            for k in range(3):
                newrow.append('nocard')

        #查找转发
        try:
            repost = weibo.find_element(By.CSS_SELECTOR, 'div.weibo-rp')
            newrow.append(1)
            # 拆分出微博内容数据
            # 拆分处文本数据
            re_text = repost.find_element(By.CLASS_NAME, 'weibo-text')
            newrow.append(re_text.text)
            # 查找@，一部分微博没有@
            try:
                ats = re_text.find_elements(By.PARTIAL_LINK_TEXT, '@')
                #这里特别的是，第一个href指向的是转发内容的发布者
                newrow.append(ats[0].text)
                atss = ''
                for at in ats[1:]:
                    atss += at.text
                newrow.append(atss)
            except:
                newrow.append('noposter')
                newrow.append('no@')

            # 查找微博话题，一部分微博没有##
            try:
                topics = re_text.find_elements(By.PARTIAL_LINK_TEXT, '#')
                topicss = ''
                for topic in topics:
                    topicss += topic.text
                newrow.append(topicss)
```

```
        except:
            newrow.append('no#')

        # 查找是否有微博链接，一部分微博没有链接
        try:
            surls = re_text.find_elements(By.CSS_SELECTOR, 'span.surl-text')
            surls = list(set(surls + topics + ats) - set(topics) - set(ats))
            surlss = ''
            for surl in surls:
                surlss += surl.text
            newrow.append(surlss)
        except:
            newrow.append('nourl')

        # 查找是否有微博表情，一部分微博没有表情
        try:
            icons=re_text.find_elements(By.CSS_SELECTOR, 'span.url-icon')
            alts=''
            for icon in icons:
                img=icon.find_element(By.CSS_SELECTOR, 'img')
                alts += img.get_attribute('alt')
            newrow.append(alts)
        except:
            newrow.append('noicon')
```

 # content 表示微博文本之后的数据，在转发的时候，一部分转发内容由于被删除（显示为：“转发的内容已被删除或不可见”，此时的 div 定位会出现问题，因此添加 try

```
        try:
            re_content = repost.find_elements(By.CSS_SELECTOR, 'div')[2]
            #这里的 1 变成了 2，是因为查找的起始位置不同

          # 查找图片
          try:
                imgs = re_content.find_elements(By.CLASS_NAME, 'm-img-box')
                if_single = re_content.find_elements(By.CLASS_NAME, 'single-img')
                if (len(imgs) == 1 and len(if_single) == 1) or len(imgs) > 1:
                    newrow.append(len(imgs))

                    special_imgs = re_content.find_elements(By.CLASS_NAME, 'feed-
mark')
                    special_img_tags = [special_img.text for special_img in speci
al_imgs]

                    gif_count = 0
```

```
                    long_image_count = 0

                    for special_img_tag in special_img_tags:
                        if special_img_tag == '动图':
                            gif_count += 1
                        else:
                            if special_img_tag == '长图':
                                long_image_count += 1
                            else:
                                print(special_img_tag)
                    newrow.append(gif_count)
                    newrow.append(long_image_count)

                else:
                    for k in range(3):
                        newrow.append('noimg')
            except:
                for k in range(3):
                    newrow.append('noimg')

            # 查找视频
        try:
                video_wrap = re_content.find_element(By.CSS_SELECTOR, 'div.weibo-
media-wraps')
                video = video_wrap.find_element(By.CSS_SELECTOR, 'div.card-video
')
                video = video.text
                newrow.append(1)
                try:
                    newrow.append(video.split()[0])
                except IndexError:
                    newrow.append('noview')
                try:
                    newrow.append(video.split()[1])
                except IndexError:
                    newrow.append('nolength')
            except:
                for k in range(3):
                    newrow.append('novideo')

            # 查找是否明星动态
        try:
                activity_wrap = re_content.find_element(By.CSS_SELECTOR, 'div.wei
bo-media-wraps')
```

```
                recent_activity = activity_wrap.find_element(By.CSS_SELECTOR, 'di
v.type-bigpic')
                newrow.append(1)
            except:
                newrow.append('noact')
            # 是否卡片
          try:
                card_wrap_wrap = re_content.find_element(By.CSS_SELECTOR, 'div.we
ibo-media-wraps')
                card_wrap = card_wrap_wrap.find_element(By.CSS_SELECTOR, 'div.car
d-wrap')
                card = card_wrap.find_element(By.CSS_SELECTOR, 'div.m-box')
                card_box = card_wrap.find_element(By.CSS_SELECTOR, 'div.m-box-col
')

                newrow.append(1)
                card_texts = card_box.text
                card_texts = card_texts.split('\n')
                if len(card_texts) <= 1:
                    newrow.append(card_texts[0])
                    newrow.append('')
                else:
                    cardss=''
                    newrow.append(card_texts[0])
                    for card in card_texts[1:]:
                        cardss+=card
                    newrow.append(cardss)
            except:
                for k in range(3):
                    newrow.append('nocard')

        except:
            for k in range(10):
                newrow.append('originerror')
    except:
        for k in range(17):
            newrow.append('norepost')
```

微博媒体转发、评论、点赞量采集

```
#获得转发、评论、点赞数据
footers=weibo.find_element(By.CSS_SELECTOR, 'footer.m-ctrl-box')
 footers=footers.text
```

```
    for footer in footers.split():
        newrow.append(footer)

    writer.writerow(newrow)

except:
    errorlist.append(count)


    print(errorlist)
    csvfile.close()
```

3) 基于 Ajax 方法抓取每条微博对应评论内容和相关数据

使用 Ajax 方法的

设置 headers

```
headers={
    'Accept':'application/json, text/plain, */*',
    'Mweibo-Pwa':'1',
    'Referer':'https://m.weibo.cn/',
    'Sec-Ch-Ua':'"Microsoft Edge";v="125", "Chromium";v="125",
"Not.A/Brand";v="24"',
    'Sec-Ch-Ua-Mobile':'?0',
    'Sec-Ch-Ua-Platform':'"Windows"',
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 Edg/125.0.0.0',
    'X-Requested-With':'XMLHttpRequest',
    'X-Xsrf-Token':'f2d20b'
}
```

自动获得 containerid

```
#黄金部分，必看，简直天才般的设想
#黄金部分，必看，简直天才般的设想
#黄金部分，必看，简直天才般的设想
for i in range(len(members)):
    #每一个 value（用户 Id)会有对应的特定的 containerid，这个 containerid 也需要获取
    #由于需要爬取多个成员的 containerid，一个一个打开对应的页面过于耗时，绞尽脑汁想出来获取
containerid 的另一种方式
    #通过模拟实际打开页面的 request 方式，获得 containerid
```

```
    response = requests.get(url='https://m.weibo.cn/', headers=headers)
    time.sleep(2)
    response = requests.get(url=members[i], headers=headers)

    #获得的members是一个中文的链接，譬如"n/时代少年团队长-马嘉祺",需要转化为数字id
    url=str(response.url)
    url=re.match('https://m.weibo.cn/u/(\d*)',url)
    try:
        value=url.group(1)
    except AttributeError:
        print(url)

    response = requests.get(url='https://m.weibo.cn/api/config', headers=headers)

response=requests.get('https://m.weibo.cn/api/container/getIndex?type=uid&value={
}'.format(value),headers=headers)
    if response.status_code == 200:
        json=response.json()

    # 这里对应有若干个containerid,分别对应的是profile,weibo,video,super topic和album
对应的container
    # 我们需要爬取的是weibo对应的containerid
    containerid=json['data']['tabsInfo']['tabs'][1]['containerid']
    params={
        'type':'uid',
        'value':value,
        'containerid':containerid
    }
```

初始化

```
    #初始化
   page_base_url='https://m.weibo.cn/api/container/getIndex?'
    comment_base_url='https://m.weibo.cn/comments/hotflow?'
    ids=[]
    mids=[]
    since_ids=[0]
    comments=[]
    errorlist=[]
    #基于当前的since_id，获取更多的since_id和当前since_id对应的微博的id和mid
```

get_page 函数

```
    def get_page(since_id):
        if since_id!=0:
            params['since_id']=since_id
        url=page_base_url+urlencode(params)
        try:
            response=requests.get(url, headers=headers)
            if response.status_code == 200:
                json=response.json()
                since_ids.append(json['data']['cardlistInfo']['since_id'])
                for card in json['data']['cards']:
                    if card['card_type']==9:
                        ids.append(int(card['mblog']['id']))
                        mids.append(int(card['mblog']['mid']))
                return json
        except KeyError as e:
            errorlist.append(f'{e} when getting page {url}')
            print(url)
            #设置循环等待，直到成功采集到，以保证采集数据的连贯性
          retry=0
            while (json['ok']==0 and retry<=3):
                #重新模拟一次访问主页的过程
             time.sleep(random.randint(5,10))
                response = requests.get(url=members[i], headers=headers)
                response = requests.get(url='https://m.weibo.cn/api/config',
headers=headers)
                response =
requests.get(url='https://m.weibo.cn/api/container/getIndex?type=uid&value={}'.fo
rmat(value),headers=headers)
                response = requests.get(url=url, headers=headers)
                json = response.json()
                retry+=1
                if retry==3:
                    retry = input('输入 retry，如果 retry=0 则继续尝试')
                    retry = int(retry)
```

get_comment 函数

```
    #基于某条微博的 id 和 mid，获取对应的评论
    def get_comment(id,mid):
        #max_id_type 类似于 weibo_page 的 since_id，这里设置只爬取第一个 comment_page 的评
论以避免数据量过大
        max_id_type=0
```

```
url=comment_base_url+urlencode({'id':id,'mid':mid,'max_id_type':max_id_type})
        try:
            response=requests.get(url, headers=headers)
            if response.status_code == 200:
                if response.json()['ok']==1:
                    return response.json()
                else:
                    print(url)
                    while response.json()['ok']==0:
                        time.sleep(5)
                        response = requests.get(url=url, headers=headers)
        except requests.exceptions.RequestException as e:
            errorlist.append(f'{e} when getting comments {url}')
            print(url)
            comments.append({})
            time.sleep(20)
```

循环采集和存储

```
#核心循环程序，count 用于计算跑了多少
count=0
 while since_ids:
     since_id=since_ids.pop(0)
     text_json=get_page(since_id)
     print(f'{count}\t',end='')
     count+=1
     while ids:
         id=ids.pop(0)
         mid=mids.pop(0)
         comments.append(get_comment(id,mid))
          #为了避免爬取过多的数据账号被封，每一次爬完一条之后随机休息一段时间
       time.sleep(random.randint(1,5))

 #存储爬取到的评论数据
jsfile=open('{}.json'.format(names[i]),'wt+',encoding='utf-8')
 js.dump(comments,jsfile,indent=2,ensure_ascii=False)

 print(i)
```

在循环采集的过程中，通过每采集完成一个微博页面，计数器+1 输出的方式来显示采集进度。

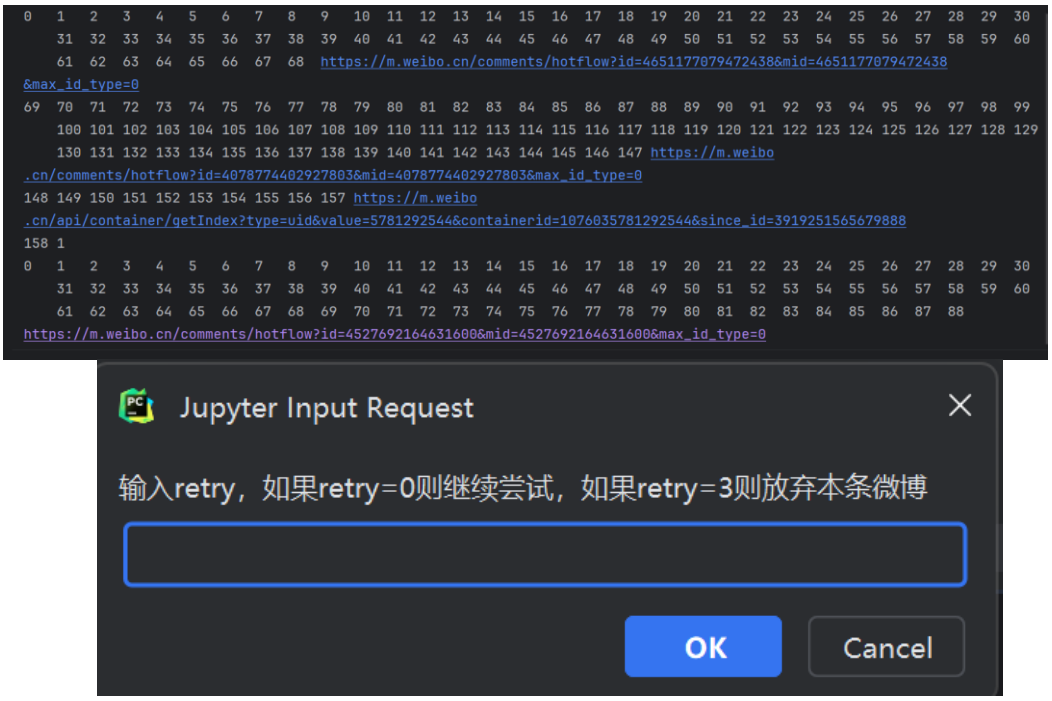
对于那些采集过程中出错的页面，不直接选择跳过，而是使用输出错误链接+循环等待的方式解决采集过程中的问题。在解决问题的过程中，可以点击输出的错误链接在浏览器中访问界面，从而确

认错误来源。常见的错误来源包括：已经到达了第一条微博、太久没有 request 'https://m.weibo.cn/api/config'、一些内容需要登陆才能访问以及一些意外的情况。

其中，对于已经到达了第一条微博的报错，一般需要人工手动验证确认后，输入 3 来结束该成员的采集；对于太久没有 request 'https://m.weibo.cn/api/config'报错和一些内容需要登陆才能访问报错，已经在错误检查程序中添加了 request 代码和 cookies。

通过这种方式，减少了采集过程中的数据损失，保证了采集数据的连贯性；同时避免了因为一次报错重新采集的风险，增强了程序的健壮性。

代码运行过程中的报错界面截图和输出如图所示。



数据预处理和数据存储

1） 数据预处理

Time:将多种时间表示转化为 yyyy-mm-dd 的形式

```
    #将日期数据拆分
     df['date']=df['time'].str.split(' ',expand=True)[0]
     print(df.info())
     print(df.head())
     df['is_yyyy_mm_dd'] = df['date'].str.contains(r'\d*-\d*-\d*')

df.loc[df['is_yyyy_mm_dd']==True,'day']=df[df['is_yyyy_mm_dd']==True]['date']
.str.split('-',expand=True)[2]

df.loc[df['is_yyyy_mm_dd']==True,'month']=df[df['is_yyyy_mm_dd']==True]['date
'].str.split('-',expand=True)[1]

df.loc[df['is_yyyy_mm_dd']==True,'year']=df[df['is_yyyy_mm_dd']==True]['date'
].str.split('-',expand=True)[0]

df.loc[df['is_yyyy_mm_dd']==False,'day']=df[df['is_yyyy_mm_dd']==False]['date
'].str.split('-',expand=True)[1]

df.loc[df['is_yyyy_mm_dd']==False,'month']=df[df['is_yyyy_mm_dd']==False]['da
te'].str.split('-',expand=True)[0]
     df.loc[df['is_yyyy_mm_dd']==False,'year']=2024
     df.drop(['is_yyyy_mm_dd'],axis=1)
     df['day']=df['day'].astype(int)
     df['month']=df['month'].astype(int)
     df['year']=df['year'].astype(int)
     df['formatted_date']=pd.to_datetime(df['year'].astype(str) + '-' +
df['month'].astype(str))
```

Type:将微博分类

```
df.loc[df['img_count'].str.contains(r'\d+'),'type']='img'
df.loc[df['video'].str.contains('1'),'type']='video'
df.loc[df['recent-activity'].str.contains('1'),'type']='recent_activity'
df.loc[df['card'].str.contains('1'),'type']='card'
df.loc[df['repost'].str.contains('1'),'type']='repost'
```

```
#都不是即是文本类微博
df.loc[df['type'].isnull(),'type']='text'
```

Forwards/Comments/Forwards:全部转化为整数数据

```
# 将转发、评论、点赞量数据归为int
#      如果是100万+,则视为100万
    for bottom_index in ['forwards','comments','likes']:
        df.loc[df[bottom_index].str.contains(r'万\+'),bottom_index]=1000000
        print(df[bottom_index])
        mask=df[bottom_index].str.contains('万',na=False)
        # df.loc[mask,bottom_index]= df.loc[mask,bottom_index].str.split('万')[0]
        df.loc[mask,bottom_index]=df.loc[mask,bottom_index].str.split('万
',expand=True)[0]
        df.loc[mask,bottom_index]=df.loc[mask,bottom_index].astype('float')
        df.loc[mask,bottom_index]=df.loc[mask,bottom_index]*10000
        df[bottom_index] = pd.to_numeric(df[bottom_index], errors='coerce')

        avg = df.groupby('formatted_date')[bottom_index].mean()

        avg.plot(x='formatted_date',y=bottom_index,kind='line')
        plt.xlabel('Time')
        plt.ylabel(f'Average')

    plt.title('Average forwards,comments,likes of {}'.format(name))
    plt.legend()
    plt.show()
```

at/topic 将分隔符删去并计数

```
    df = df.iloc[10:,:].copy()
    icons={}
    df['split_icon']=df['url-icon'].str.split("[")
    # print(df['split_icon'].head())
    for splits in df[df['split_icon'].notnull()]['split_icon']:
        for split in splits[1:-1]:
            # 使用的是[做区分，则需要删除]
            if split!='':
                split=split.replace(']','')
                if split in icons:
                    icons[split]=icons[split]+1
                else:
                    icons[split]=1
    # print(icons)
    for key,value in icons.items():
```

```
        if key not in heat.columns:
            heat[key]=0
        heat.loc[name,key]=value
```

2) 将处理好的 csv 部分转换成 sql 存储

```
conn = pymysql.connect(host='localhost', user="root",passwd='JackeyLove2568*',
charset='utf8mb4')
cur = conn.cursor()
cur.execute('CREATE DATABASE IF NOT EXISTS weibo CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;')
cur.execute('use weibo')

for name in names:
    cur.execute('CREATE TABLE IF NOT EXISTS {}(id bigint(10) AUTO_INCREMENT, text
TEXT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci, wtype VARCHAR(200),
forwards INT, comments INT, likes INT, PRIMARY KEY(id)) DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_unicode_ci;'.format(
            name))
    df=pd.read_csv(name+'.csv',encoding='utf-8')
    count=0
    sql=f'insert into {name} (text,wtype,forwards,comments,likes)
values(%s,%s,%s,%s,%s)'
    for bottom_index in ['forwards','comments','likes']:
        df.loc[df[bottom_index].str.contains(r'万\+'),bottom_index]=1000000
        # print(df[bottom_index])
        mask=df[bottom_index].str.contains('万',na=False)
        # df.loc[mask,bottom_index]= df.loc[mask,bottom_index].str.split('万')[0]
        df.loc[mask,bottom_index]=df.loc[mask,bottom_index].str.split('万
',expand=True)[0]
        df.loc[mask,bottom_index]=df.loc[mask,bottom_index].astype('float')
        df.loc[mask,bottom_index]=df.loc[mask,bottom_index]*10000
        df.loc[df[bottom_index].astype(str).str.contains(r'赞|评论|转发
'),bottom_index]=0
        df[bottom_index].astype('int')
    df.loc[df['img_count'].str.contains(r'\d+'),'type']='img'
    df.loc[df['video'].str.contains('1'),'type']='video'
    df.loc[df['recent-activity'].str.contains('1'),'type']='recent_activity'
    df.loc[df['card'].str.contains('1'),'type']='card'
    df.loc[df['repost'].str.contains('1'),'type']='repost'
    #都不是即是文本类微博
   df.loc[df['type'].isnull(),'type']='text'

    df.loc[df['text'].isnull(),'text']=''
```

```
    for index,row in df.iterrows():
        # print (row['text'], row['type'], row['forwards'], row['comments'],
row['likes'])
        try:
            cur.execute(sql, (row['text'], row['type'], row['forwards'],
row['comments'], row['likes']))
            conn.commit()
        except:
            conn.rollback()
# cur.execute('select * from {}'.format(name))
# print(cur.fetchall())
cur.close()
conn.close()
```

3) 数据存储

微博内容存储的数据结构

以 csv 格式存储以下微博内容，人工手动操作将每一位成员对应的 csv 汇总成了 gathered.xlsx 便于查看，但是后续数据处理的时候使用的是每一位成员对应的 csv 文件。

微博的发送时间		yyyy-mm-dd/mm-dd/mm-dd tt:tt 等多种类型
微博的发送年份		int 20XX
微博的发送月份		int
微博的发送天		int
微博的发送方式		NAN or string
微博的文本内容		string
微博的@对象		('@'+string)*
微博带有的话题（#微博话题#）		#string#*
微博中使用的表情		NAN or string
微博的类型		string
文本型		如果以下所有类型都为 no 则为文本型
照片型	照片数量	noimg or int
	动图数量	noimg or int
	长图数量	noimg or int
视频型	视频长度	novideo or int:int
	播放量	novideo or int+'万'
明星动态型		noact or 1
卡片型	卡片标记	nocard or 1
	卡片标题	nocard or string
	卡片内容	nocard or string
转发型	转发标记	norepost or 1
	转发的文本内容	norepost or string
	转发来源	norepost or '('@'+string)*
	转发的@对象	norepost or '('@'+string)*

	转发带有的话题	norepost or NAN or #string#*
	转发中使用的表情	norepost or NAN or string
	转发的类型	norepost or NAN or string
	照片型	和微博对应相同
	视频型	
	明星动态型	
	卡片型	
微博的转发量		int
微博的评论量		int
微博的点赞量		int

csv 格式存储的数据使用 python 进行数据可视化。

微博评论存储的数据结构

以 json 格式存储了完整的微博评论页面 return 回的 json 数据（偷懒 ver. 主要是爬的时候没有想好可能从微博评论中获得什么样的结论）。其中重要信息包括"created_at"、"text"、"floor_number"、"source"等，分别表示评论的发送时间、评论的文本内容、评论的发送层数（第几个发送的评论）、评论的 IP 属地。访问路径均为：

json[i]['data']['data']['created_at(text/floor_number/source)']

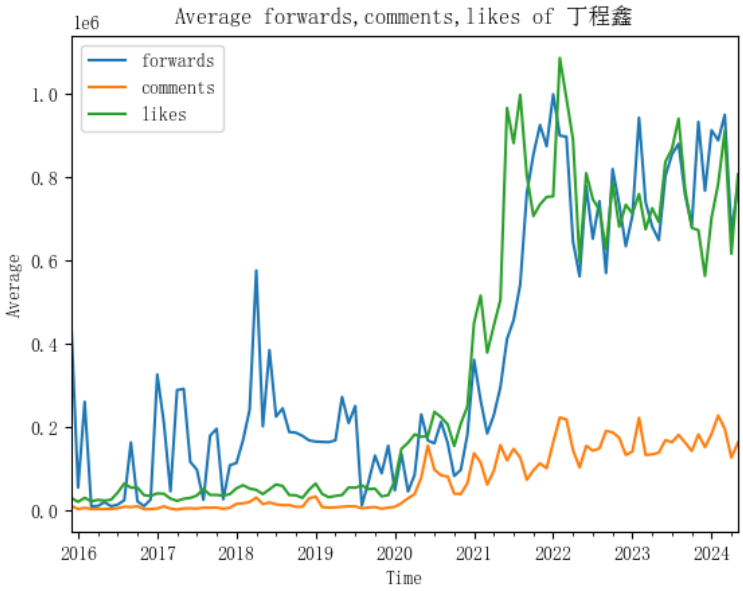
json 格式存储的数据使用 Tableau 进行数据可视化。

可视化和结论

1） 单一成员的微博数据分析

由于各成员之间分析模式大致相同，这里仅以最具代表性的丁程鑫为例。其他成员的可视化图表放在可视化第三部分，不做分析。

时间序列分析

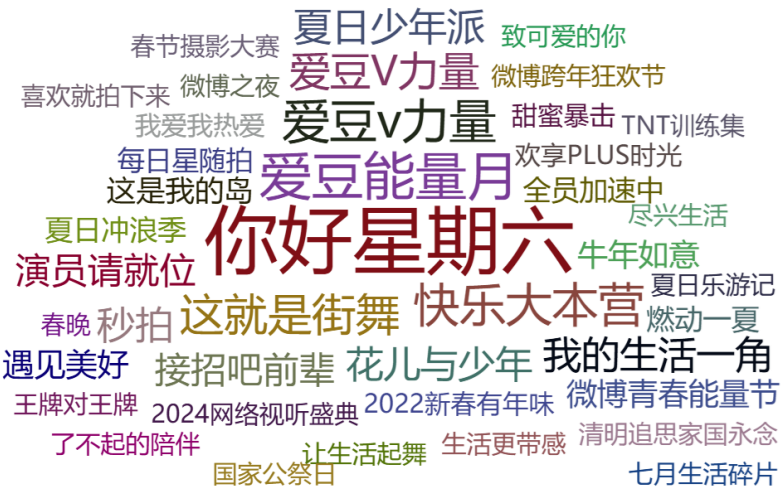


2020 年下半年，是时代少年团爆发的一年。由于日积月累，粉丝增多、同行竞争团又出事，时代少年团的资源大幅度增加，逐步出现在了大众面前。在成员的微博数据上，可以很明显看到 2020 年，丁程鑫的三项微博数据指标均有了明显的提升乃至飙升。整体看来，丁程鑫的星途璀璨，目前热度良好。

携带话题词云

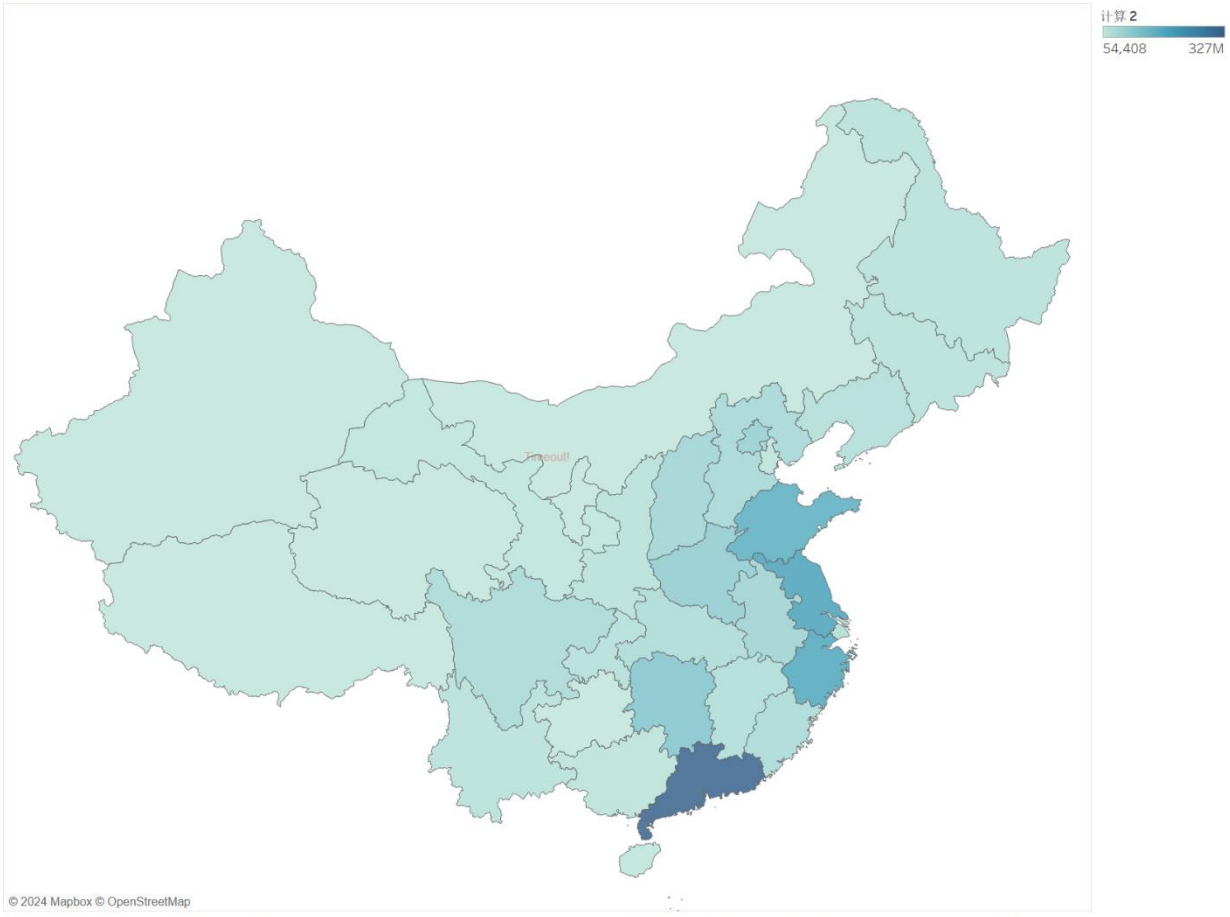
其中需要说明的是，之所以有两个“爱豆 v 力量”，是因为使用了大小写的区别。通过对于携带话题词云的筛选可以从中获得丁程鑫参加的综艺和艺人活动的信息，如综艺节目“演员请就位”、“王牌对王牌”、“快乐大本营”、“这就是街舞”等；艺人活动“爱豆能量月”、“你好星期六"等。通过词云图的回顾，可以看到那些活动艺人长期发展和参与了，哪些可能有所参与但并不长久。丁程鑫参与较久较多的活动话题有“你好星期六”、“爱豆能量月”、” 爱豆 V 力量”、“快乐大本营”等。

The Word Cloud of Topics of 丁程鑫



粉丝属地分析

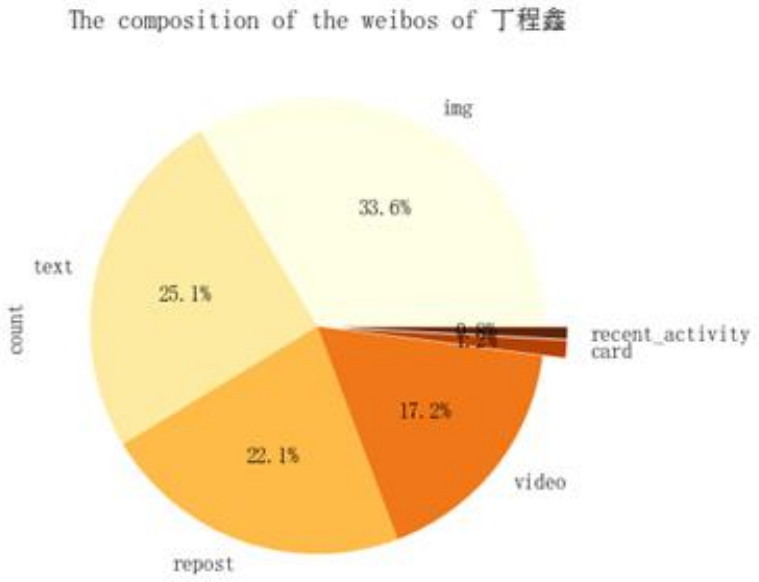
丁程鑫粉丝IP来源分布



基于 经度(自动生成) 和 纬度(自动生成) 的地图。 颜色显示 计算 2 总和。 为 Source - 拆分 1 显示了详细信息。 视图按 Source - 拆分 1 进行筛选，这会保留 36 个成员(总共 49 个)。

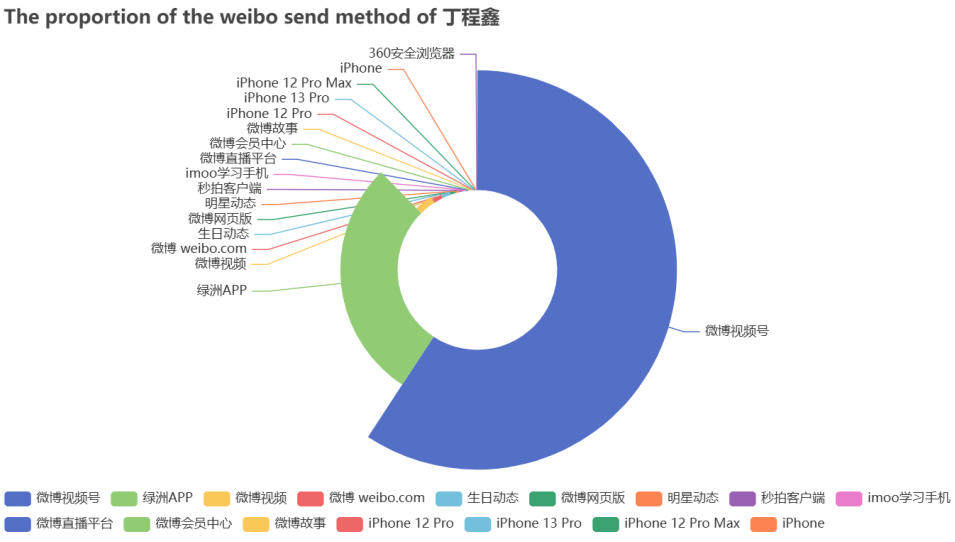
通过对评论区中 IP 属地的查询，对于中国大陆地区的用户制作 IP 来源分布地图。可以看出，丁程鑫的粉丝主要分布在沿海一带，以广东为首，江浙鲁紧随其后。

微博类型分析



微博构成方面，丁程鑫的微博呈现较为平均的四开，除了少有的微博动态和转发卡片内容之外，绝大多数内容都较为平均地分布于图片、文本内容、转发和视频中。对于一个一人而言，塑造一个具有多媒体形象的微博主页是艺人基本功中较为必需的一环，在这一方面，丁程鑫表现良好，既没有“用力过猛”，也没有“平平无奇”。

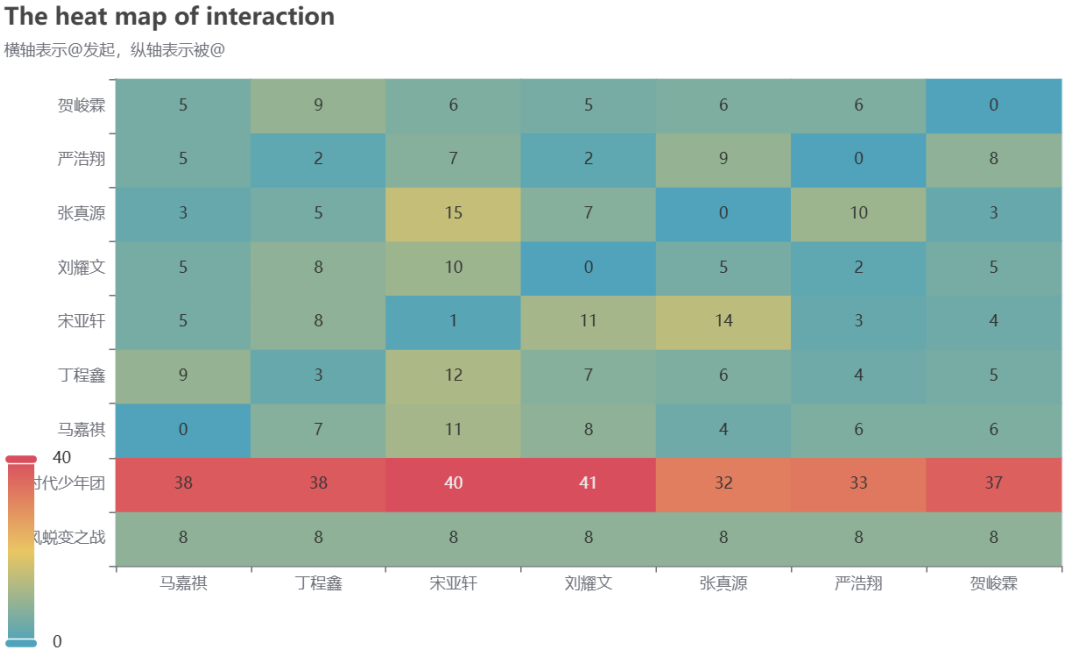
发博方式分析



在所有可以获得发博方式的微博内容中，主要是微博视频号和绿洲 APP 为主。这两个发博方式占主导的原因是由于微博自身机制导致（发视频必须要使用微博视频号）。在剩余的几个部分中，丁程鑫比较独特的发博方式是“imoo 学习手机”，回看当时的微博，发现他接到了该品牌手机的代言。发播方式体现的艺人信息的信息熵较低。

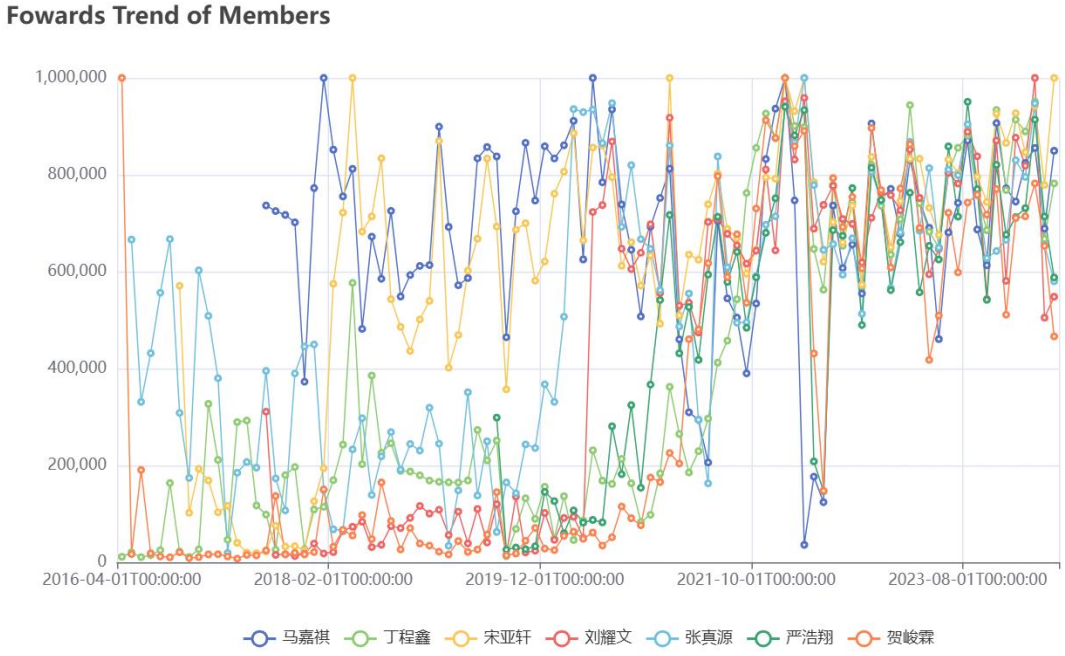
2) 成员间微博数据比较分析

成员互动分析

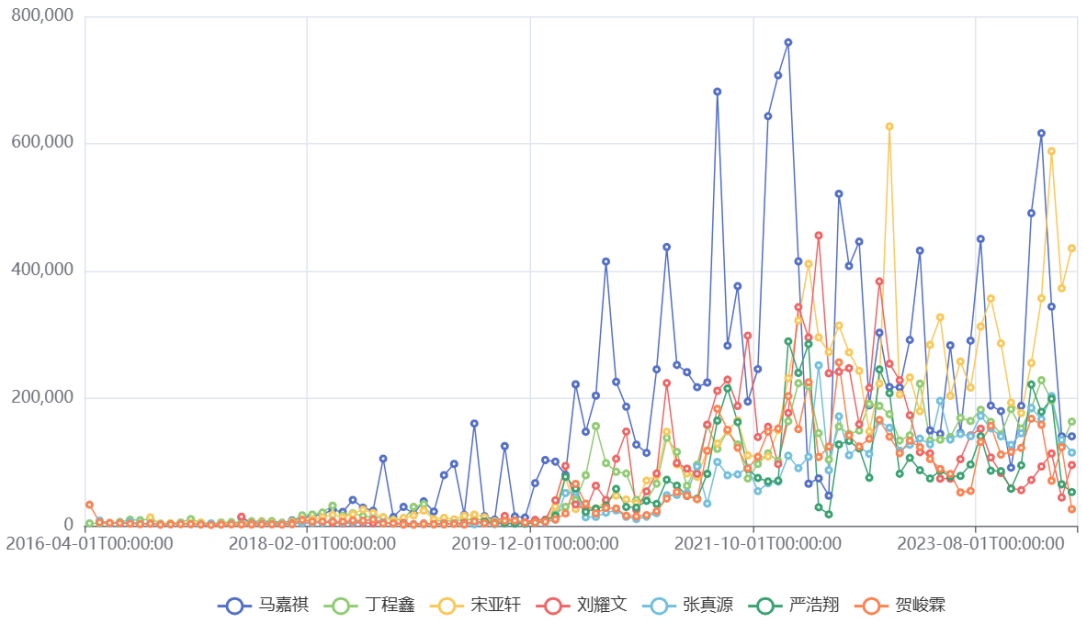


对所有成员的@互动情况绘制热图，可以明显看出，张真源和宋亚轩、严浩翔和张真源、刘耀文和宋亚轩、宋亚轩和丁程鑫关系较好。而宋亚轩又是一个比较喜欢@人的人，也正因如此，他也收获了很多人的@。

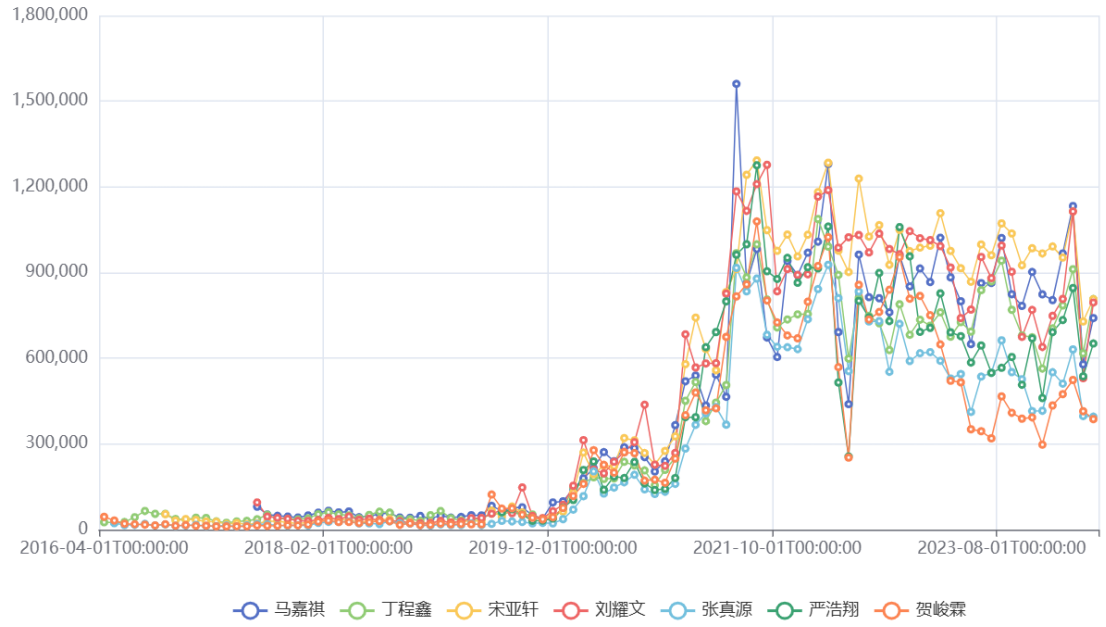
微博热度分析



Comments of Members



Likes of Members



成员微博热度分析再一次验证了“2020 年下半年，是时代少年团爆发的一年。”的结论。2020 年后，成员的转发、评论和点赞数量都有了较为明显的中枢抬升。

转发层面看，宋亚轩和马嘉祺在大火之前就已有不错的微博转发数据，展现出不错的发展潜力，也的确在大火之后的数据表现拔得头筹——当然，也不排除购买转发数据造假的嫌疑。大火之后，各成员见转发数据基本接近，波动趋势相同。少有的一次下挫发生在 2021 年下半年，那一年发生的最大事件是“#马嘉祺 我没有到高考录取分数线#”这一件事情。与转发数量下挫的同时（甚至是更早）是评论量的蹿升，达到了历史极值。这可能是由于粉丝对于偶像形象光环的崩塌

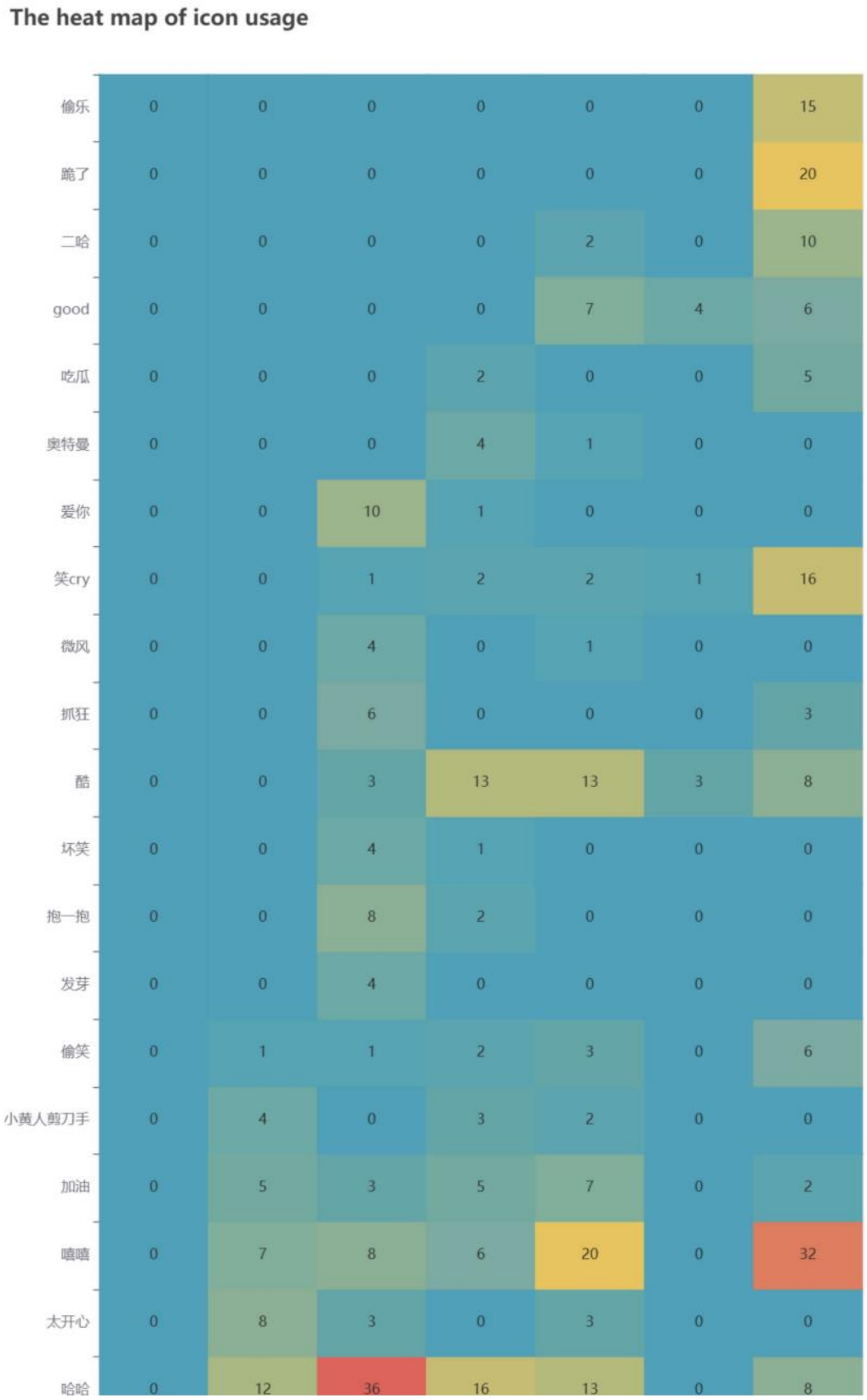
所导致的脱粉、不愿意给他转发点赞的缘故；也不排除马嘉祺艺人经济团队为了避风头，刻意控制了艺人微博热度有关。

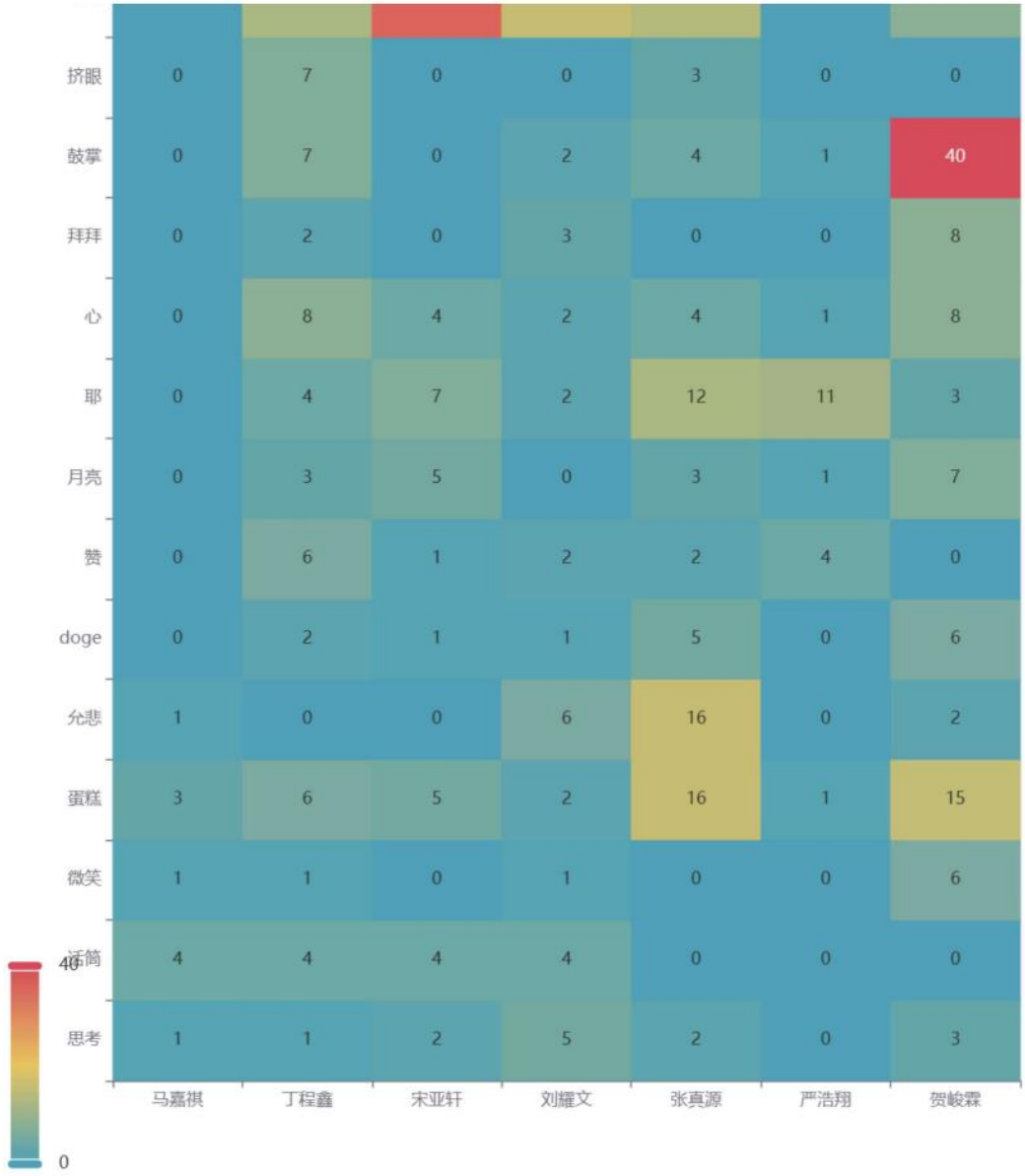
评论方面的结论和转发类似。除了出道以来一直争议不断的马嘉祺的评论波动巨大，宋亚轩长期以来的评论数一直领跑团队，展现出良好的发展势头，其余几位成员则相对而言较为接近，没有特别明显的差异。

点赞方面，2022 年之前，各成员之间的点赞数量相对接近，保持着一致趋同发展的势头。2022 年后出现了一定程度的分化，贺峻霖、张真源出现了一定程度的下跌趋势，相对于其他成员保持着一定均值水平波动，略有下降而言，他们俩位的下降趋势更加明显，一定程度上提示“过气”风险。

最爱使用表情

这里仅仅计算包括在微博 url-icon 中的表情，不包含计入文本中的 emoji，计入规则为发送数量>=3。

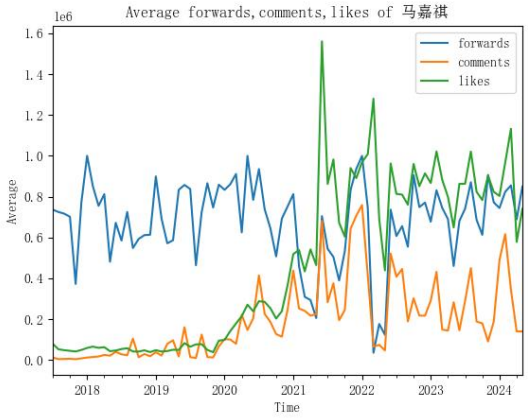


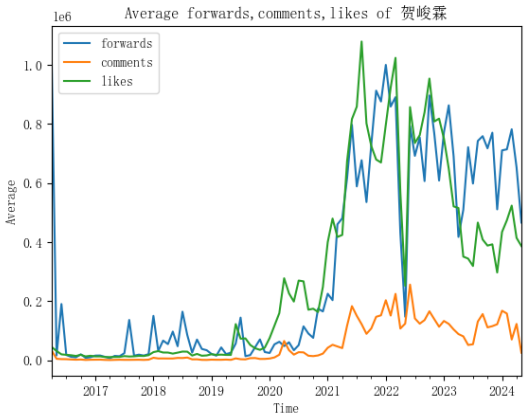


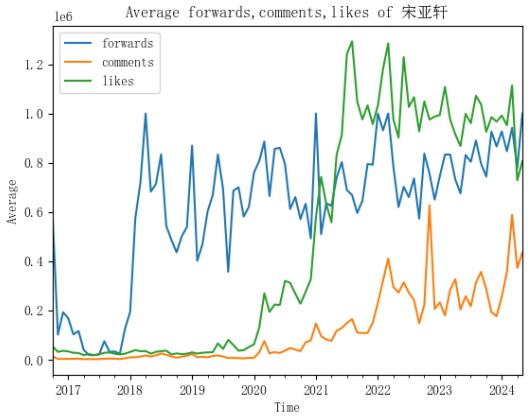
从该热图中可以明显看到各个成员之间不同的微博表情使用习惯。贺峻霖、张真源相对而言更喜欢使用微博表情，而马嘉祺则不太使用微博表情。其他成员几乎不怎么使用的[鼓掌]表情，贺峻霖一个人使用了 40 次。[嘻嘻]、[哈哈]、[酷]、[蛋糕]等一些比较常见的日常使用表情或者表示心情的表情也如预期，获得了较为普遍的使用。

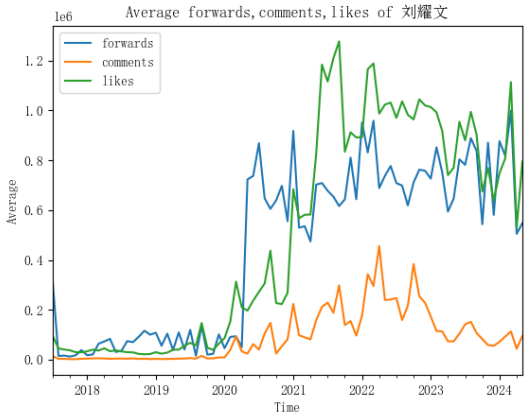
3） 附表：其他成员微博数据分析

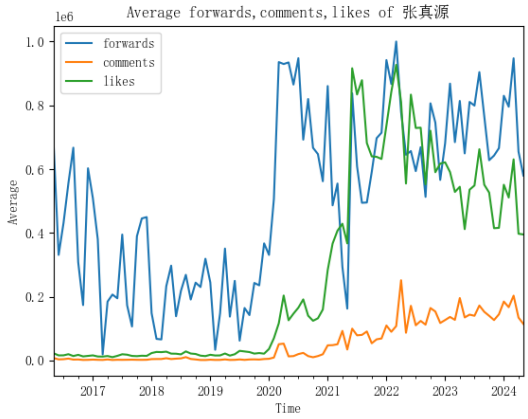
时间序列分析

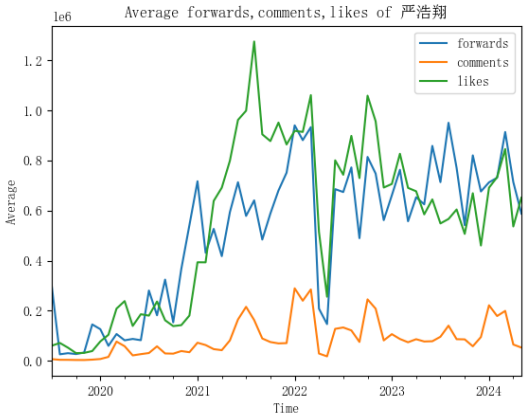






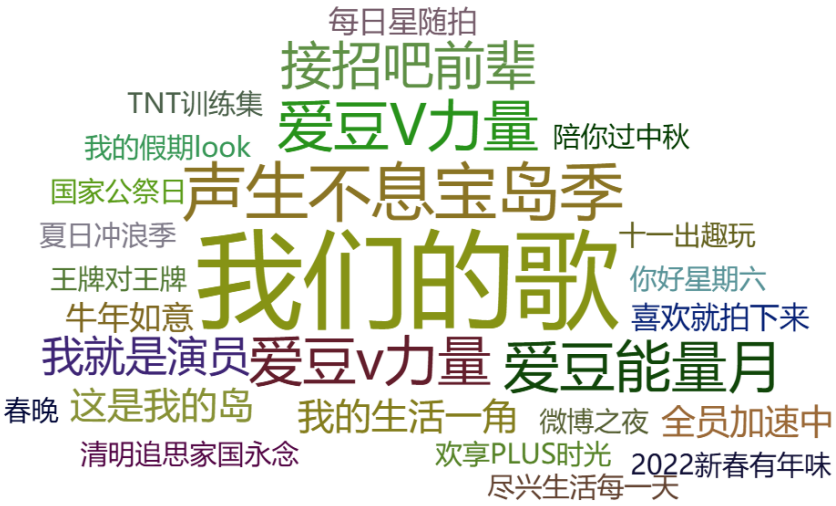




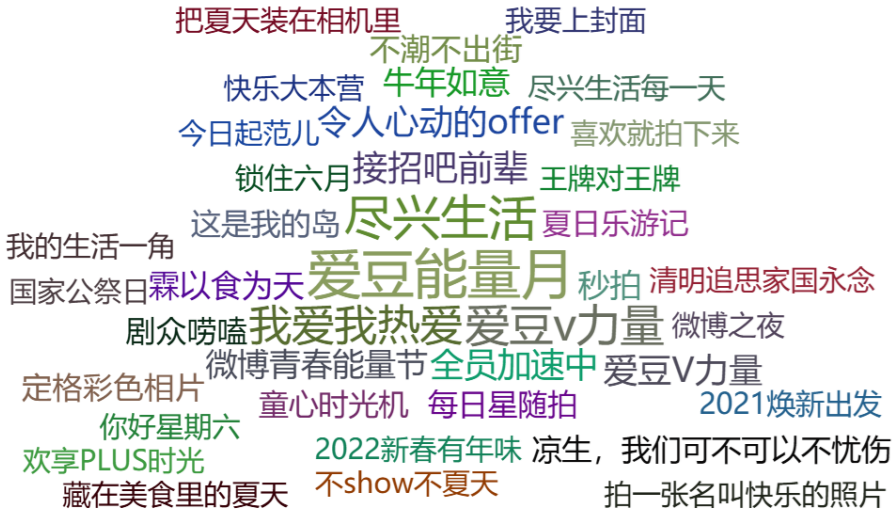


携带话题词云

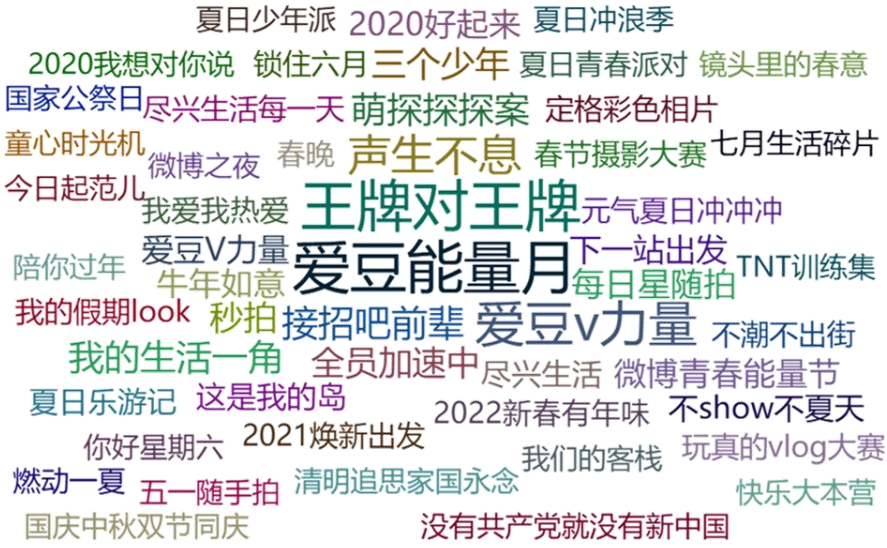
The Word Cloud of Topics of 马嘉祺



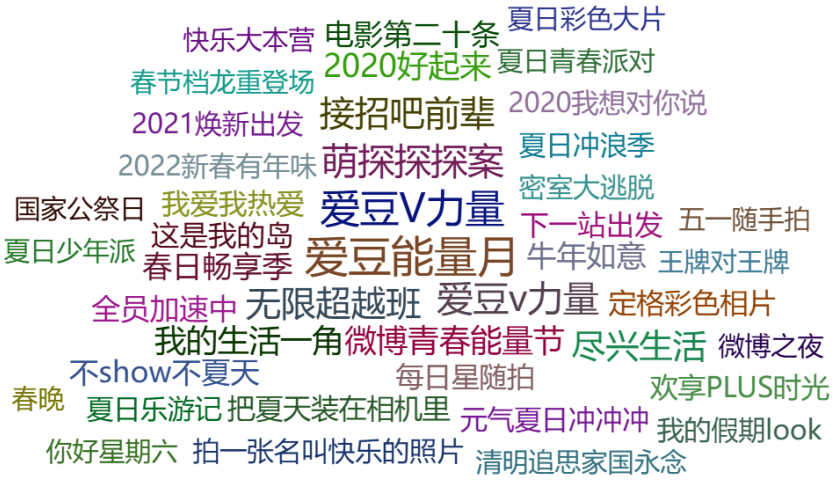
The Word Cloud of Topics of 贺峻霖



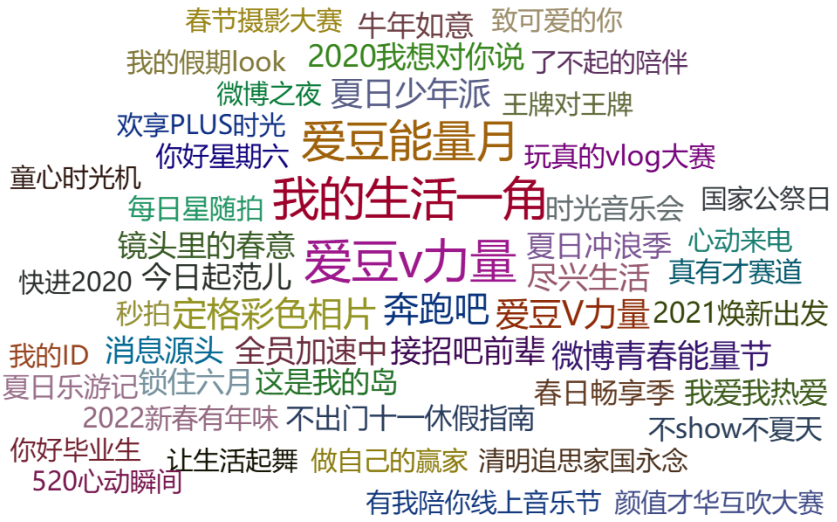
The Word Cloud of Topics of 宋亚轩



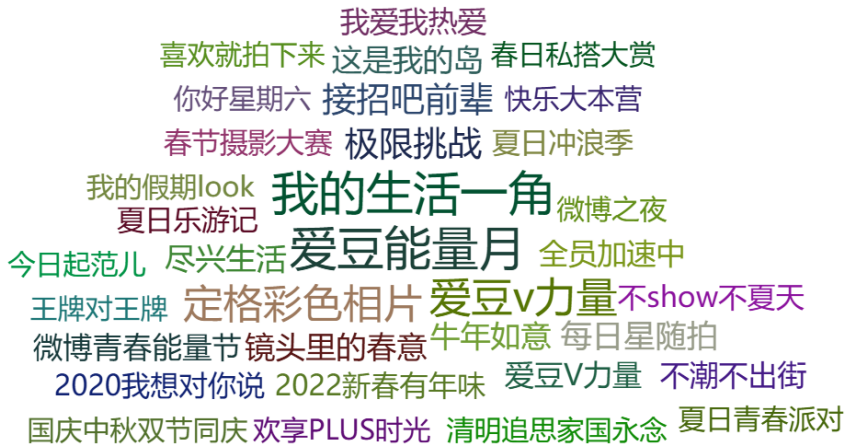
The Word Cloud of Topics of 刘耀文



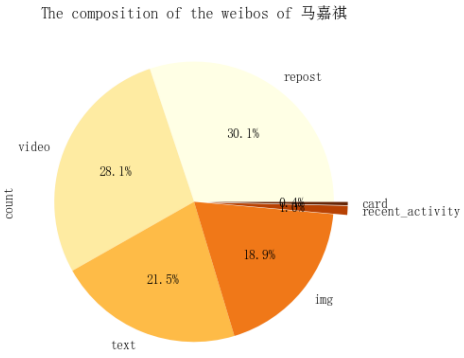
The Word Cloud of Topics of 张真源

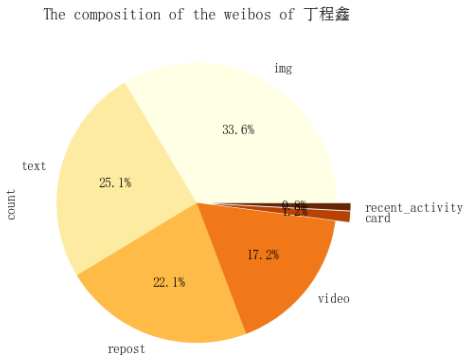


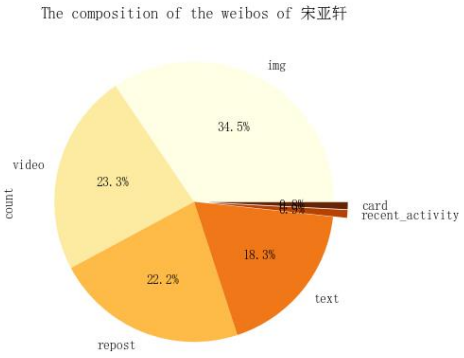
The Word Cloud of Topics of 严浩翔

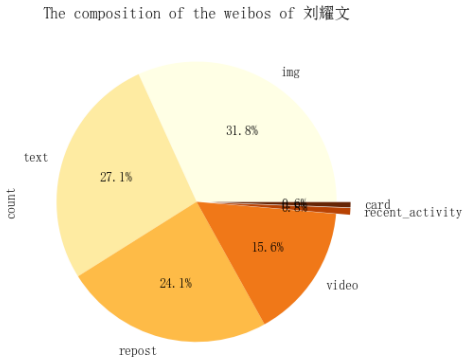


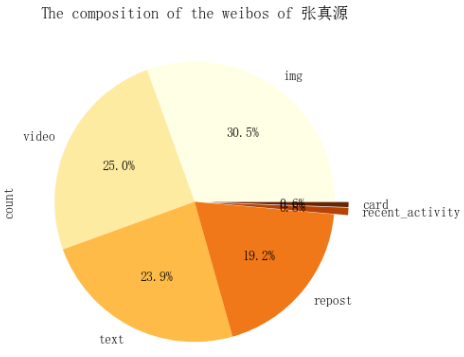
微博类型分析

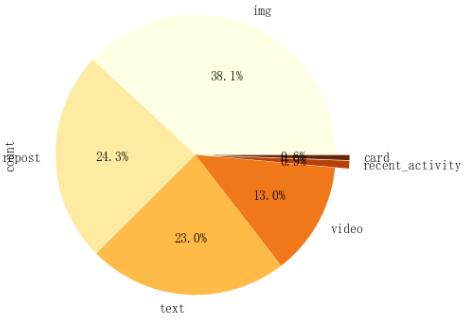

The composition of the weibos of 马嘉祺
repost
video
30.1%
28.1%
0.4%
card
recent_activity
18.9%
21.5%
img
text
count

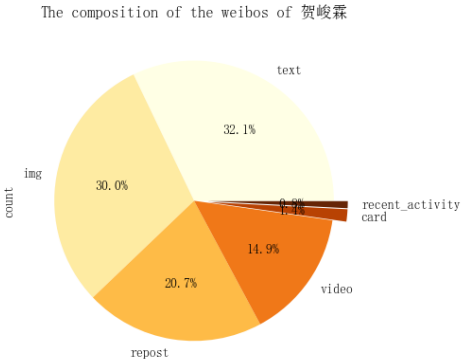

The composition of the weibos of 丁程鑫
img
text
33.6%
25.1%
recent_activity
card
17.2%
22.1%
video
repost
count


The composition of the weibos of 宋亚轩
img
video
34.5%
23.3%
card
recent_activity
18.3%
22.2%
text
repost
count


The composition of the weibos of 刘耀文
img
text
31.8%
27.1%
card
recent_activity
15.6%
24.1%
video
repost
count

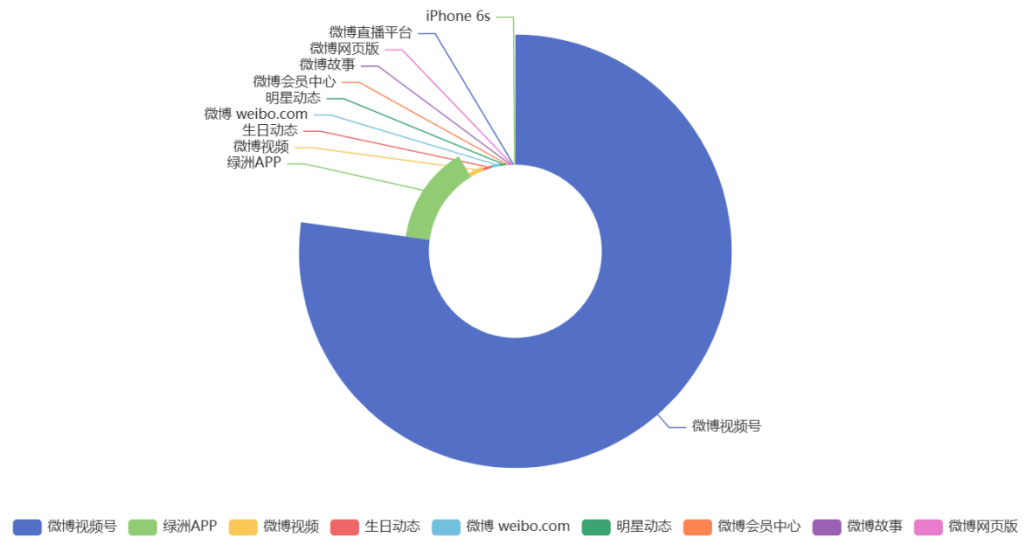

The composition of the weibos of 张真源
img
video
30.5%
25.0%
card
recent_activity
19.2%
23.9%
repost
text
count


The composition of the weibos of 严浩翔
img
38.1%
repost
24.3%
card
recent_activity
13.0%
video
23.0%
text
count

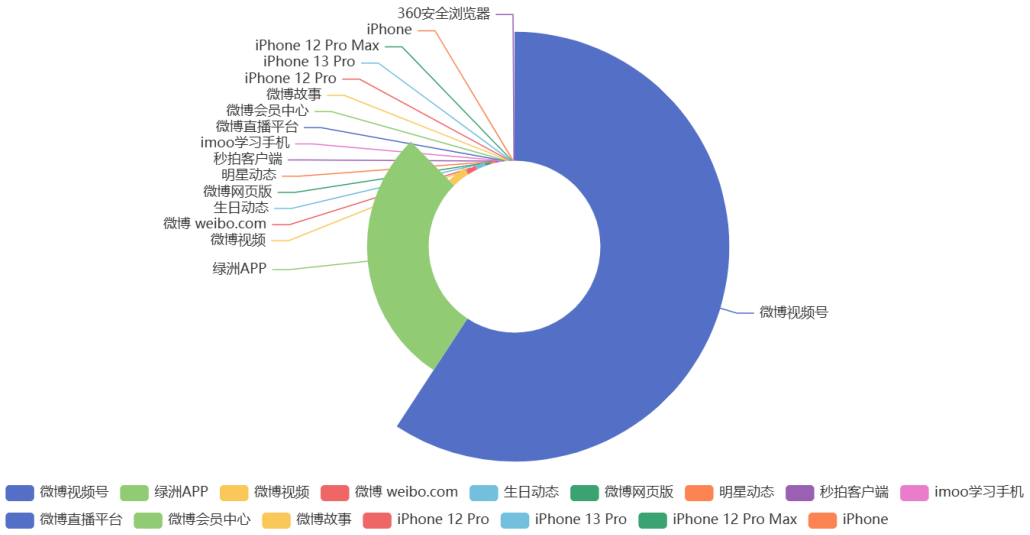

The composition of the weibos of 贺峻霖
text
32.1%
img
30.0%
recent_activity
card
14.9%
20.7%
video
repost
count

发博方式分析

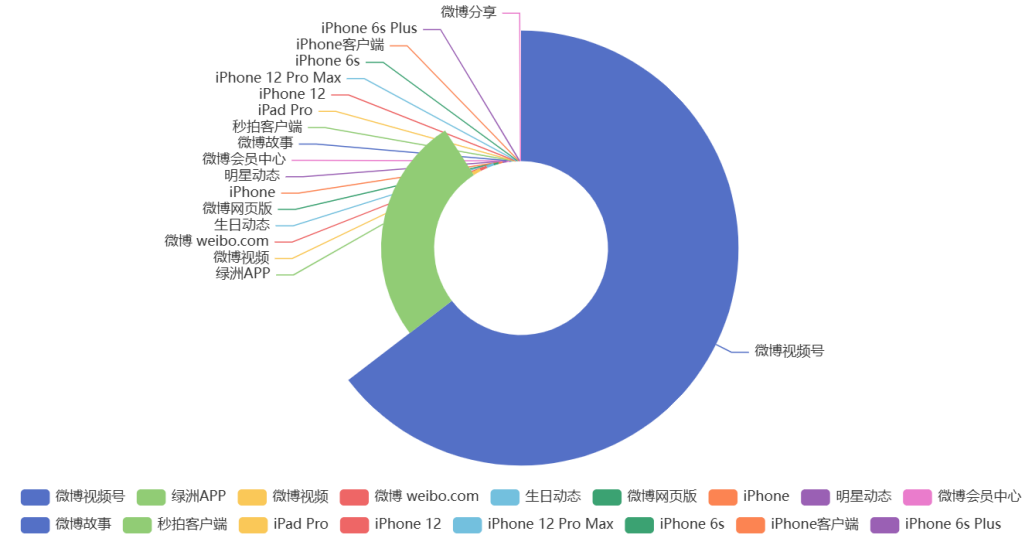
The proportion of the weibo send method of 马嘉祺



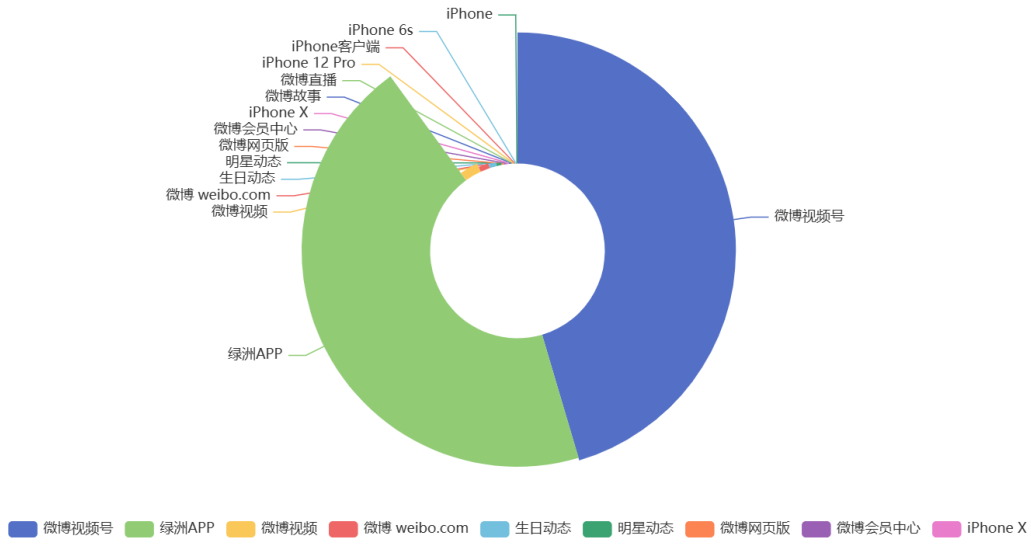
The proportion of the weibo send method of 丁程鑫



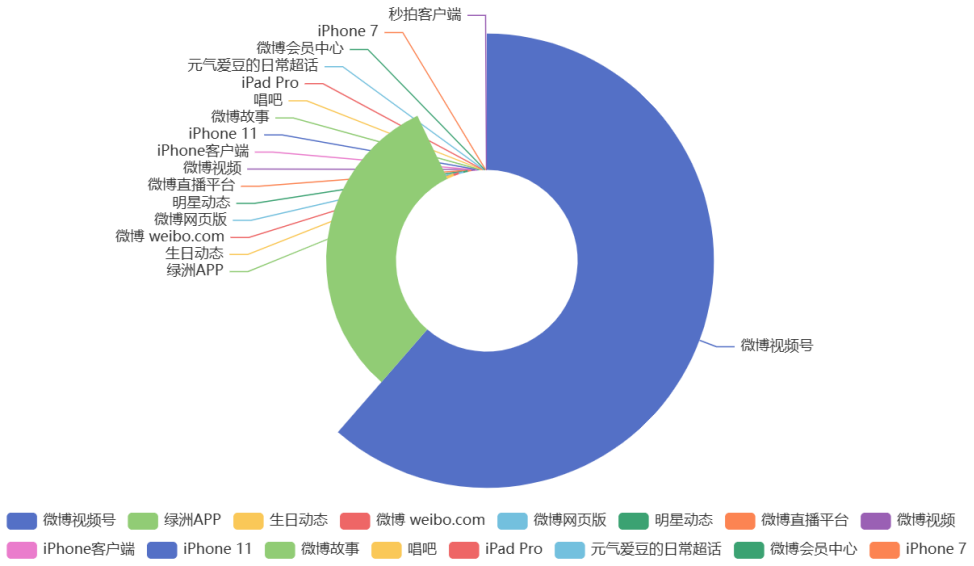
The proportion of the weibo send method of 宋亚轩



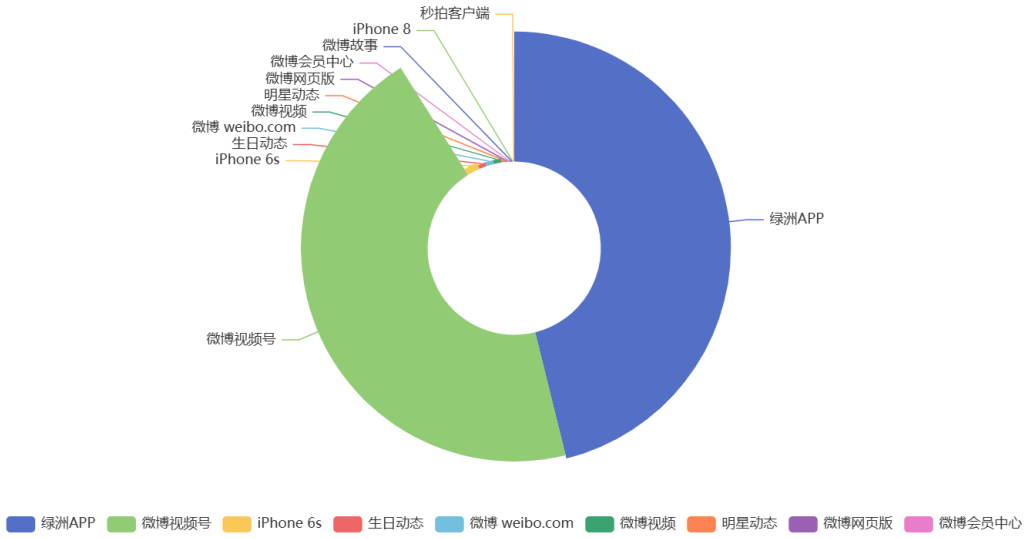
The proportion of the weibo send method of 刘耀文



The proportion of the weibo send method of 张真源



The proportion of the weibo send method of 贺峻霖



The proportion of the weibo send method of 严浩翔