Bilibili_Danmaku Data Crawling and Analysis

Background: Bilibili, known as the bullet-screen comments also called B-station, is an ACG-related bullet-screen comments video sharing website in China. The video part is mainly containing bilibili live, animation, fan, music, dance and so on. Bullet-screen (so called Danmaku from Japanese), as a key indicator of user activity level, is also a representative product of B-station which has already developed as the listed company successfully. Bilibili is providing the following three common danmaku modes: the rolling danmaku, the top danmaku, and the bottom danmaku. Users with certain permissions can post advanced danmaku comment, visitors and users below the second level cannot send danmaku and visitors cannot comment and leave messages on the bottom of the video page.

(1) Select two different videos to collect the danmaku data;
(2) Try to use web scraper to collect data and analyze;
(3) Compare these two kinds of video, summarize their features.

*web scraper only crawls incomplete data due to the limitation of web scroll bar, so using python to implement.

1. open the xml file, known the meaning of each item.

```
▼<i>
   <chatserver>chat.bilibili.com</chatserver>
   <chatid>113607463</chatid>
   <mission>0</mission>
   <maxlimit>1000</maxlimit>
   <state>0</state>
   <real_name>0</real_name>
   <source>e-r</source>
   <d p="153.49100,1,25,16777215,1566985527,0,ff09637d,20889391619112964">秀儿</d>
   <d p="11.28400,1,25,16777215,1566985628,0,418c3def,20889444645076992">哈哈哈</d>
   <d p="277.45600,1,25,16777215,1566985694,0,314ef924,20889479244939266">打广告就没有激励的钱了</d>
   <d p="152.20400,1,25,16777215,1566985696,0,3d03967,20889480361148416">好有道理</d>
   <d p="34.13300,1,25,16777215,1566985737,0,3fcf6985,20889501813964800">带数学家</d>
   <d p="77.48000,1,25,16777215,1566985824,0,16a6a3a9,20889547142856704">我全都要</d>
   <d p="114.80500,1,25,16777215,1566985836,0,3fcf6985,20889553364582400">工地英语</d>
   <d p="259.42300,1,25,16777215,1566985905,0,49c15780,20889589580824576">活久见，弹幕全是求着打广告的
   <d p="123.67200,1,25,16777215,1566985965,0,96c74878,20889621427126272">卧槽！！！</d>
```

Inside the quotation marks: first one is danmaku occur time, unit in second.
Second one is the mode of danmaku, 1..3 rolling 4 bottom 5 top 6 reverse 7 specific location 8 advanced
Third one is word size, 12 smallest 16 very small 18 small 25 medium 36 big 45 very big 64 biggest
Fourth one is the color of words, following HTML color
Fifth one is the timestamps in unix format, based on 1970-1-1 08：00：00
Sixth one is danmaku pool, 0 normal pool 1 subtitle pool 2 special pool (only for advanced danmaku)
Seventh one is user id, used to shield the sender if needed
Eighth one is the row id, used for “historical danmaku”

2. install the useful packages

```
In  [1]:  !pip install lxml requests beautifulsoup4 jieba # install packages
```

```
Requirement already satisfied: lxml in d:\anaconda\lib\site-packages (4.2.1)
Requirement already satisfied: requests in d:\anaconda\lib\site-packages (2.18.4)
Requirement already satisfied: beautifulsoup4 in d:\anaconda\lib\site-packages (4.6.0)
```

```
n [41]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        import requests
        from bs4 import BeautifulSoup
        import lxml
        import jieba #Chinse word segmenmtation module
```

3. data crawling and analysis

collect data from web

```
7]: #video1_food: ”点外卖备注“我有200万粉丝，吃播你家外卖”店主的操作我傻眼了！”
    url=r'http://comment.bilibili.com/113607463.xml'
    r1=requests.get(url)
    r1.encoding='utf-8' # download data and saved in r1
```

```
3]: soup=BeautifulSoup(r1.text,'lxml')
    Danmaku_food=soup.find_all('d') # parse web page, and find all 'd' tag, then save in Danmaku_food
```

```
[79]: import datetime
      Danmaku_food_list=[] # create a blank list
      n=0
      for d in Danmaku_food:
          n+=1
          Danmaku_food={} # put single info into dict
          Danmaku_food['content']=d.text
          Danmaku_food['url']=url
          Danmaku_food['occur_time']=float(d.attrs['p'].split(',')[0])
          Danmaku_food['sending_time']=datetime.datetime.fromtimestamp(int(d.attrs['p'].split(',')[4]))
          Danmaku_food['mode']=d.attrs['p'].split(',')[1]
          Danmaku_food['color']=d.attrs['p'].split(',')[3]
          Danmaku_food['pool']=d.attrs['p'].split(',')[5]
          Danmaku_food_list.append(Danmaku_food) # put all dicts into list
          if n % 100 == 0:
              print('get %i data' %n)
      print('done, get %i data in total' %n)

      get 100 data
      get 200 data
      get 300 data
      get 400 data
      get 500 data
      get 600 data
      get 700 data
      get 800 data
      get 900 data
      get 1000 data
```

```
30]: df_food=pd.DataFrame(Danmaku_food_list) # convert list to dataframe
```

```
6]: df_food.to_csv('Danmaku_food.csv',encoding='utf_8_sig')
```

```
7]: df_food.head()
```

7]:

	color	content	mode	occur_time	pool	sending_time	url
0	16777215	实在是高	1	178.664	0	2019-08-28 19:35:53	http://comment.bilibili.com/113607463.xml
1	16777215	鬼才	1	141.639	0	2019-08-28 19:38:35	http://comment.bilibili.com/113607463.xml
2	16777215	鬼才	1	162.876	0	2019-08-28 19:39:55	http://comment.bilibili.com/113607463.xml
3	16646914	看到这里终于笑了	1	142.136	0	2019-08-28 19:40:00	http://comment.bilibili.com/113607463.xml
4	16777215	哈哈哈哈丑了?	1	346.568	0	2019-08-28 19:40:06	http://comment.bilibili.com/113607463.xml

2]:
```
#video2_game:"现在吃鸡卖外挂的都这么硬核得吗！？各种内幕爆料！【绝地求生】"
url=r'http://comment.bilibili.com/111831214.xml'
r2=requests.get(url)
r2.encoding='utf-8'
```

3]:
```
soup2=BeautifulSoup(r2.text,'lxml')
Danmaku_game=soup2.find_all('d')
```

84]:
```
Danmaku_game_list=[]
n=0
for d in Danmaku_game:
    n+=1
    Danmaku_game={}
    Danmaku_game['content']=d.text
    Danmaku_game['url']=url
    Danmaku_game['occur_time']=float(d.attrs['p'].split(',')[0])
    Danmaku_game['sending_time']=datetime.datetime.fromtimestamp(int(d.attrs['p'].split(',')[4]))
    Danmaku_game['mode']=d.attrs['p'].split(',')[1]
    Danmaku_game['color']=d.attrs['p'].split(',')[3]
    Danmaku_game['pool']=d.attrs['p'].split(',')[5]
    Danmaku_game_list.append(Danmaku_game)
    if n % 100 == 0:
        print('get %i data' %n)
print('done, get %i data in total' %n)
```

```
get 100 data
get 200 data
get 300 data
get 400 data
get 500 data
get 600 data
get 700 data
get 800 data
get 900 data
```

5]:
```
df_game=pd.DataFrame(Danmaku_game_list)
```

3]:
```
df_game.to_csv('Danmaku_game.csv',encoding='utf-8')
```

4]:
```
df_game.head()
```

4]:

	color	content	mode	occur_time	pool	sending_time	url
0	16777215	哈哈哈哈哈哈哈	1	155.404	0	2019-08-21 01:28:00	http://comment.bilibili.com/111831214.xml
1	16777215	哈哈哈哈哈	1	254.736	0	2019-08-21 01:29:43	http://comment.bilibili.com/111831214.xml
2	16777215	局长哈哈哈哈	1	415.359	0	2019-08-21 01:32:53	http://comment.bilibili.com/111831214.xml
3	16777215	@橙子	1	449.812	0	2019-08-21 01:41:24	http://comment.bilibili.com/111831214.xml
4	16777215	29杀呢	1	442.592	0	2019-08-21 01:43:05	http://comment.bilibili.com/111831214.xml

[8]:
```
df_food.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
color           1000 non-null object
content         1000 non-null object
mode            1000 non-null object
occur_time      1000 non-null float64
pool            1000 non-null object
sending_time    1000 non-null datetime64[ns]
url             1000 non-null object
dtypes: datetime64[ns](1), float64(1), object(5)
memory usage: 54.8+ KB
```

:
```
df_game.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
color           1500 non-null object
content         1500 non-null object
mode            1500 non-null object
occur_time      1500 non-null float64
pool            1500 non-null object
sending_time    1500 non-null datetime64[ns]
url             1500 non-null object
dtypes: datetime64[ns](1), float64(1), object(5)
memory usage: 82.1+ KB
```

```
]: # combine the two dataframe for later analysis
   df=df_food.append(df_game).reset_index(drop=True)
```

```
]: df['url']=df['url'].replace('http://comment.bilibili.com/113607463.xml','food')
```

```
]: df['url']=df['url'].replace('http://comment.bilibili.com/111831214.xml','game')
```

```
]: df=df.rename(columns={'url':'cat'}) #mark a new column:category
```

```
]: df.info()
```

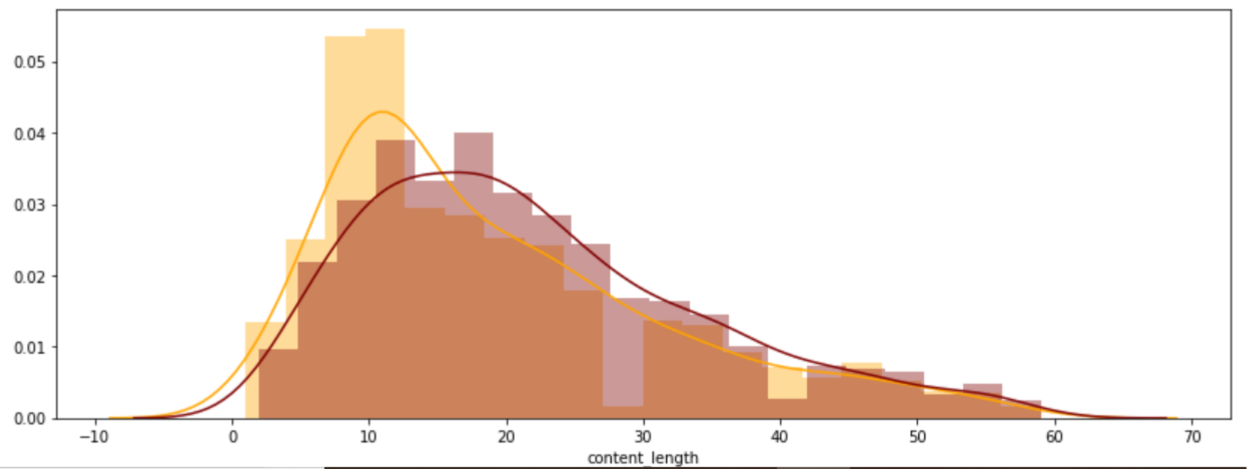
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 8 columns):
Unnamed: 0      2500 non-null int64
color           2500 non-null int64
content         2500 non-null object
mode            2500 non-null int64
occur_time      2500 non-null float64
pool            2500 non-null int64
sending_time    2500 non-null int64
cat             2500 non-null object
dtypes: float64(1), int64(5), object(2)
memory usage: 156.3+ KB
```

Danmaku length analysis

```
17]: df['content_length']=df['content'].str.len()
     df=df[df['content_length']<60]
```

```
47]: plt.figure(figsize=(14,5))
     sns.distplot(df['content_length'][df.cat=='food'],bins=20,color='orange')
     sns.distplot(df['content_length'][df.cat=='game'],bins=20,color='maroon')
```

```
47]: <matplotlib.axes._subplots.AxesSubplot at 0x10d7e470>
```



It can be seen from the figure that the length of the danmaku of video (food) is about 7-10 words, and the length of the game video is around at 15-20 words. The user watching game video like to talk more because communication when playing games is very important.

Danmaku frequency-list analysis

```
33]:
import jieba.analyse
pd.set_option('max_colwidth',500)
#read data again
rows=pd.read_csv('Danmaku_food.csv', header=0,encoding='utf-8',dtype=str)

segments = []
for index, row in rows.iterrows():
    content = row[2] #including the index column, so content is the third column
    #TextRank keywords collection
    words = jieba.analyse.textrank(content, topK=50,withWeight=False,allowPOS=('ns', 'n', 'vn', 'v'))
    splitedStr = ''
    for word in words:
        # record overall words
        segments.append({'word':word, 'count':1})
        splitedStr += word + ' '
dfSg = pd.DataFrame(segments)

# word frequency count
dfWord = dfSg.groupby('word')['count'].sum()
dfWord.to_csv('keywords.csv',encoding='utf-8')
```

```
37]:
dfWord.sort_values(ascending=False)[:10]
```

```
37]:
word
鬼才    47
逻辑    35
空气    32
店主    14
广告    14
重量    12
人家     9
物理     8
店家     7
粉丝     6
Name: count, dtype: int64
```

```
38]:
pd.set_option('max_colwidth',500)
rows=pd.read_csv('Danmaku_game.csv', header=0,encoding='utf-8',dtype=str)

segments = []
for index, row in rows.iterrows():
    content = row[2]
    words = jieba.analyse.textrank(content, topK=50,withWeight=False,allowPOS=('ns', 'n', 'vn', 'v'))
    splitedStr = ''
    for word in words:
        # record overall words
        segments.append({'word':word, 'count':1})
        splitedStr += word + ' '
dfSg = pd.DataFrame(segments)

# word frequency count
dfWord = dfSg.groupby('word')['count'].sum()
dfWord.to_csv('keywords2.csv',encoding='utf-8')
```

```
]: dfWord.sort_values(ascending=False)[:10]
```

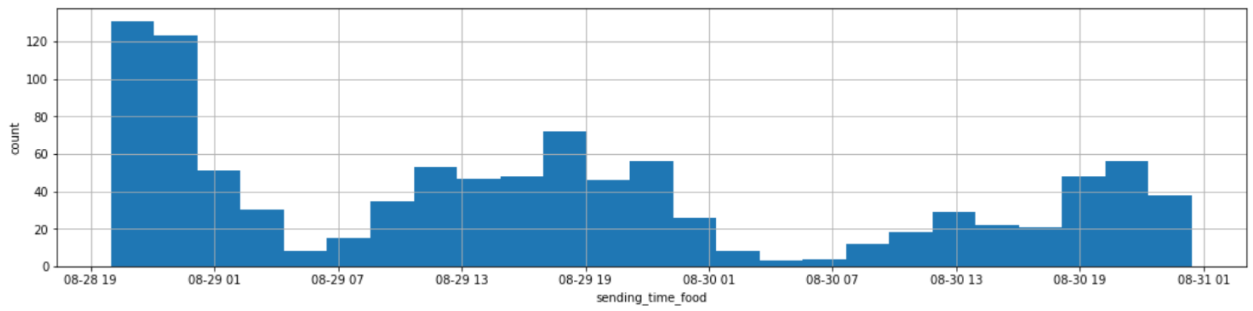
```
]: word
   开挂    36
   没有    20
   卖挂    18
   声音    16
   笑声    12
   游戏    10
   好像     8
   神仙     8
   裤头     7
   被盗     7
   Name: count, dtype: int64
```

Compare the words with most frequent occurrences of the video, found that the people who watched the food program liked to praise the host and concerned more on the restaurant providing the food; the vocabulary used by the audience of the game program mostly comment on the game, conform more to the video theme like "外挂"(means plug-in).

Danmaku sending_time analysis

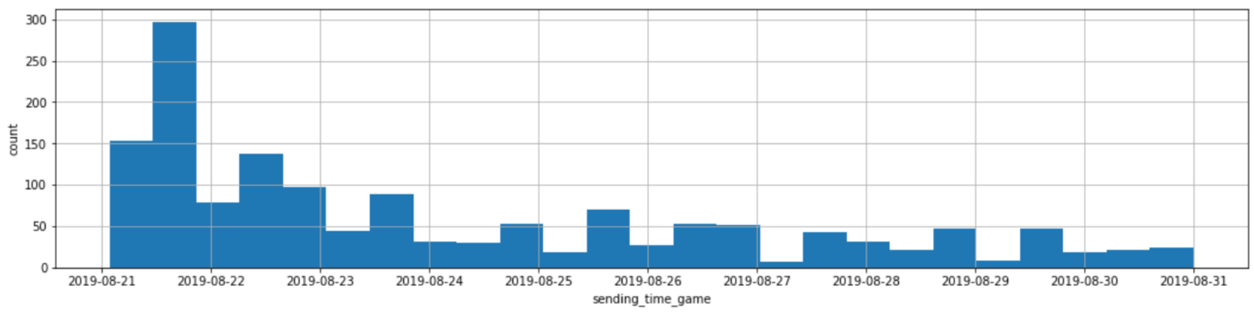
```
]: # investigate the whole distribution over several days' period
   plt.figure(figsize=(18,4))
   df_food['sending_time'].hist(bins=25)
   plt.xlabel('sending_time_food')
   plt.ylabel('count')
```

```
]: Text(0,0.5,'count')
```



```
]: plt.figure(figsize=(18,4))
   df_game['sending_time'].hist(bins=25)
   plt.xlabel('sending_time_game')
   plt.ylabel('count')
```

```
]: Text(0,0.5,'count')
```



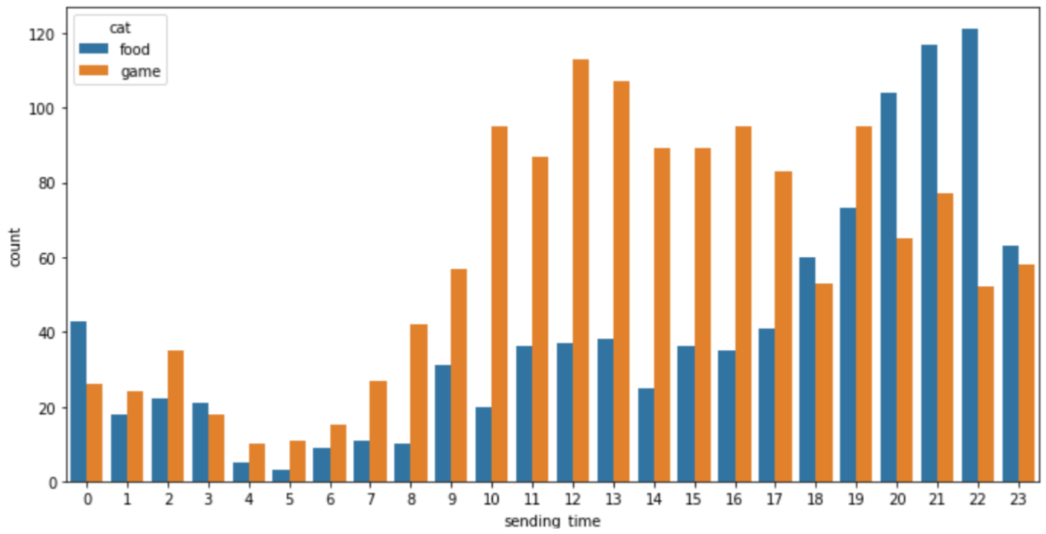
It can be seen that the game video kept a longer time of sending danmaku comments, and the amount of the comment is gradually reduced for nearly 10 days. The video of the eating lasted for 4 days, but the amount of the danmaku per day has a peak, indicating that people really love eating.

```
]: # check the danmaku comment number distribution in one day
   import datetime
   df_food['sending_time'] = pd.to_datetime(df_food['sending_time'])
   df_game['sending_time'] = pd.to_datetime(df_game['sending_time'])
```

```
]: df_food['sending_time']=df_food['sending_time'].apply(lambda x:x.hour)
   df_game['sending_time']=df_game['sending_time'].apply(lambda y:y.hour)
```

```
]: plt.figure(figsize=(12,6))
   sns.countplot(x='sending_time',hue='cat',data=df)
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x11379940>
```



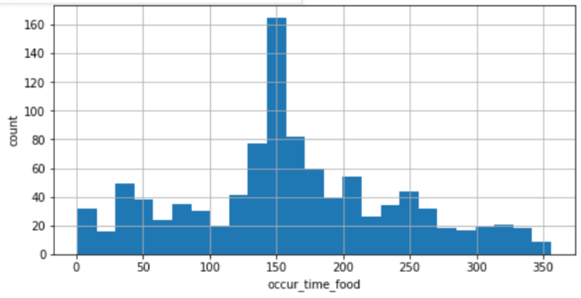
The peak of the number of comments on the game program is from 10:00 am to 5:00 pm, which is the daytime period, lasting for a long time; the peak of the food program is from 8:00 to 10:00 pm, which is shorter for the night time period (everyone enjoy watching delicious food in the middle of the night.)

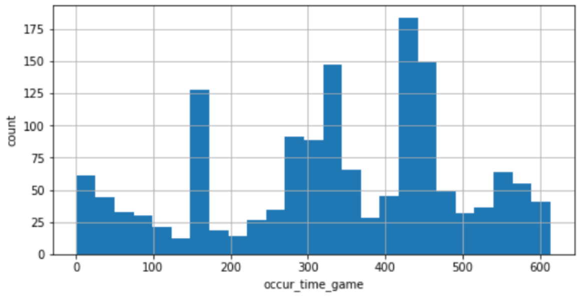
Danmaku occur_time analysis

```
[72]: plt.figure(figsize=(18,4))
      plt.subplot(1,2,1)
      df_food['occur_time'].hist(bins=25)
      plt.xlabel('occur_time_food')
      plt.ylabel('count')
      plt.subplot(1,2,2)
      df_game['occur_time'].hist(bins=25)
      plt.xlabel('occur_time_game')
      plt.ylabel('count')
```

```
t[72]: Text(0,0.5,'count')
```

click to expand output; double click to hide output



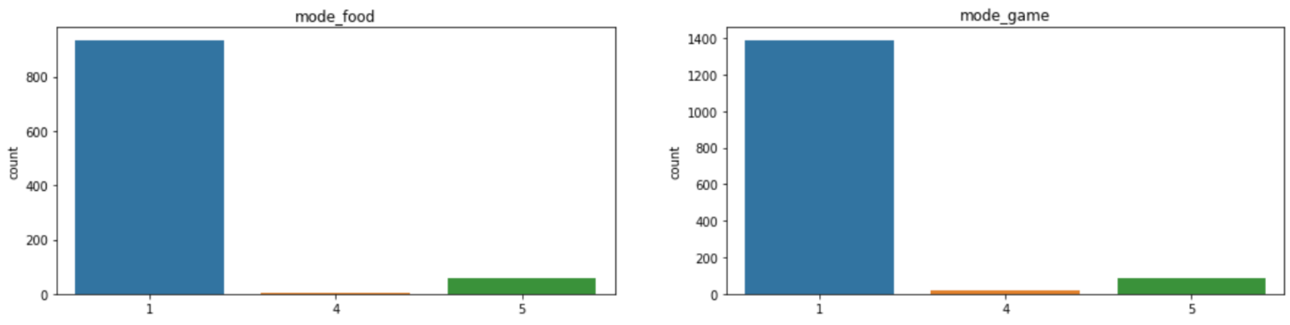


Compared with the occur time of the danmaku during the two videos' playing, danmaku appeared in the middle of the food video, so perhaps the most interesting content of the video is at about the 150th second; while there's several peaks of the danmaku appearance in game video, which exactly showing the uncertainty and diversity during the game.

Danmaku mode analysis

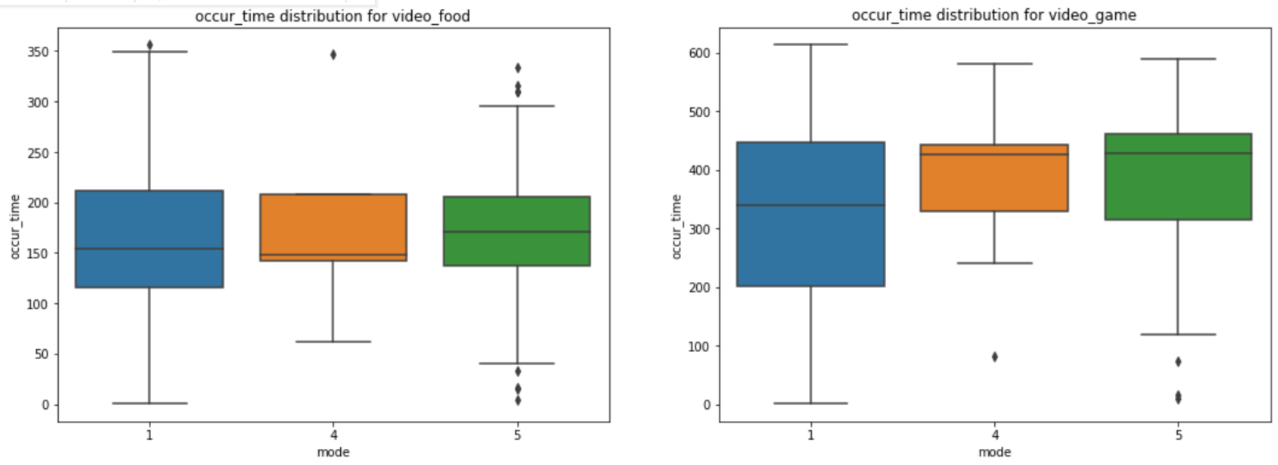
```
]: plt.figure(figsize=(18,4))
   plt.subplot(1,2,1)
   sns.countplot(x='mode',data=df_food)
   plt.title('mode_food')
   plt.subplot(1,2,2)
   sns.countplot(x='mode',data=df_game)
   plt.title('mode_game')
```

```
]: Text(0.5,1,'mode_game')
```



```
7]: plt.figure(figsize=(18,6))
    plt.subplot(1,2,1)
    sns.boxplot(x='mode',y='occur_time',data=df_food)
    plt.title('occur_time distribution for video_food')
    plt.subplot(1,2,2)
    sns.boxplot(x='mode',y='occur_time',data=df_game)
    plt.title('occur_time distribution for video_game')
```

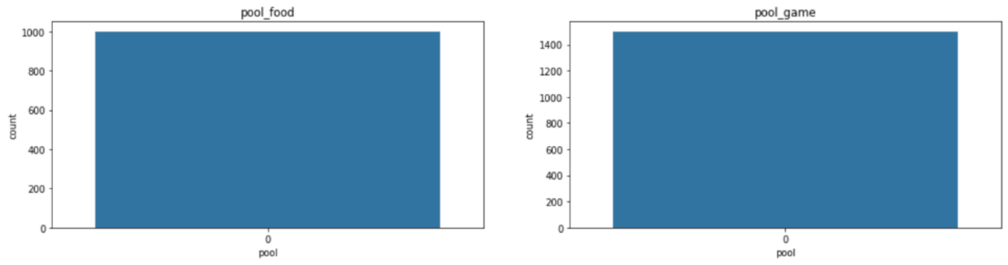
```
]: Text(0.5,1,'occur_time distribution for video_game')
   click to expand output; double click to hide output
```



Comparing the number of the danmaku modes and the occue time in the two videos, it is found that the mode 1 scroll bar is the most popular in both of the two videos, followed by the mode 5 top barrage, and the least used is the bottom mode 4; there is not much difference in the occur time of the video, basically distributed throughout the video. The biggest difference lies in the mode 4, which is concentrated in the early stage of eating video, while more at the end of the game video.

```
)8]:  plt.figure(figsize=(18,4))
      plt.subplot(1,2,1)
      sns.countplot(x='pool',data=df_food)
      plt.title('pool_food')
      plt.subplot(1,2,2)
      sns.countplot(x='pool',data=df_game)
      plt.title('pool_game')

)8]:  Text(0.5,1,'pool_game')
```



Danmaku pool is 0 normal pool both in the two videos, means that most of the audience are normal users rather than vip.

Conclusion: 1. After crawling data need to dispose the name of each field and data type, especially the time
2. When doing word frequency-list analysis, checked more materials and got so many methods, would better choose the most easy-understanding methods for yourself
3. Through this project, got more familiar with crawling code and plot packages in python