-

July 1, 2020

0.1 Python 201865

[17]:
```
#
import pandas as pd
import numpy as np
import re
import jieba
import pyecharts
```

[18]:
```
#
df_all = pd.read_csv("danmu2.csv", header= 0,index_col=0,encoding='utf-8-sig')
df = df_all.copy()

#
df = df.reset_index(drop=True)
df.head()
```

[18]:

	tv_name	uid	contentsId	contents	likeCount
0	01	1354611638	1592838562920007156		19
1	01	1290387715	1592308702474009744		78
2	01	1304178541	1592317555873002245		9
3	01	1791773843	1593139392541001130		2
4	01	1581452867	1593150522156003497		0

[19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201865 entries, 0 to 201864
Data columns (total 5 columns):
tv_name       201865 non-null object
uid           201865 non-null int64
contentsId    201865 non-null int64
contents      201865 non-null object
likeCount     201865 non-null int64
dtypes: int64(3), object(2)
memory usage: 7.7+ MB
```

0.1.1

[20]:
```
#
danmu_counts = df.groupby('uid')['contentsId'].count().sort_values(ascending =␣
↪False).reset_index()
danmu_counts.columns = [' id','    ']
danmu_counts.head()
```

[20]:

	id	
0	1810351987	2561
1	1319063154	146
2	2244033948	131
3	2407948956	106
4	1488898523	104

[21]:
```
df_top1 = df[df['uid'] == 1810351987].sort_values(by="likeCount",ascending =␣
↪False).reset_index()
df_top1.head(10)
```

[21]:

	index	tv_name	uid	contentsId	contents	likeCount
0	48926	03	1810351987	1592396281667005291		125
1	18370	01	1810351987	1592310435456006922		96
2	48276	03	1810351987	1592401371493007155		94
3	52807	03	1810351987	1592398904483005951		81
4	97350	06	1810351987	1592483605611004816		81
5	53405	03	1810351987	1592399003845007165		77
6	71351	04	1810351987	1592398032950002734		74
7	78328	05	1810351987	1592483501326004084		73
8	71459	04	1810351987	1592401777744000259		70
9	65411	04	1810351987	1592396346696002728		66

[22]:
```
data_top1 = df_top1.groupby('tv_name')['contentsId'].count()
```

[23]:
```
from pyecharts.charts import Bar
from pyecharts import options as opts

bar0 = Bar(init_opts=opts.InitOpts(width='960px', height='500px'))
bar0.add_xaxis(data_top1.index.tolist())
bar0.add_yaxis("",data_top1.values.tolist())
bar0.set_global_opts(title_opts=opts.TitleOpts(title='   '))
bar0.set_series_opts(
        label_opts=opts.LabelOpts(is_show=False),
        markline_opts=opts.MarkLineOpts(
            data=[opts.MarkLineItem(y=213.4, name="yAxis=213.4")]
        ))
bar0.render_notebook()
```

[23]:
```
<pyecharts.render.display.HTML at 0x2429fea5888>
```

0.1.2

[24]:
```
df.head()
```

[24]:

	tv_name	uid	contentsId	contents	likeCount
0	01	1354611638	1592838562920007156		19
1	01	1290387715	1592308702474009744		78
2	01	1304178541	1592317555873002245		9
3	01	1791773843	1593139392541001130		2
4	01	1581452867	1593150522156003497		0

[25]:
```
df_like = df[df.groupby(['tv_name'])['likeCount'].rank(method="first",␣
↪ascending=False)==1].reset_index()[['tv_name','contents','likeCount']]
df_like.columns = [' ',' ',' ']
df_like
```

[25]:

0	01		8305
1	02		8889
2	03		8526
3	04		8451
4	05	6	8472
5	06		7452
6	07		3387
7	08	5601	
8	09		4533
9	10		4521
10	11		3097
11	12		6174

0.1.3

[26]:
```
a = {'  ':' | |  ',
     '  ':'  ',
     ' ':'  ',
     ' ':'  ',
     '  ':'  ',
     '  ':' |  ',
     ' ':'  ',
     ' ':' |  ',
     '  ':'  | |  ',
     ' ':' |   ',
     '  ':' | ',
     '  ':'  ',
     '  ':'  '}
```

```
for key, value in a.items():
    df[key] = df['contents'].str.contains(value)
staff_count = pd.Series({key: df.loc[df[key], 'contentsId'].count() for key in␣
 ↪a.keys()}).sort_values()
print(staff_count)
```

```
        2
       22
       47
       99
      125
      153
      583
      818
     1462
     1942
     5075
     5188
     5734
dtype: int64
```

[27]:
```
from pyecharts import options as opts
from pyecharts.charts import Polar
a = ['  ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ']
b = [22,47,99,125,153,583,818,1462,1942,5075,5188,5734]
Polar = Polar(init_opts=opts.InitOpts(width='960px', height='500px'))

Polar.add_schema(angleaxis_opts=opts.AngleAxisOpts(data=a, type_= "category"))
Polar.add("", b, type_="bar")
Polar.set_global_opts(title_opts=opts.TitleOpts(title="     "))
#word1.render("     .html")
Polar.render_notebook()
```

[27]:
```
<pyecharts.render.display.HTML at 0x2429fa95ec8>
```

0.1.4

[12]:
```
def get_cut_words(content_series):
    #
    import jieba
    stop_words = []

    with open("stop_words.txt", 'r', encoding='utf-8') as f:
        lines = f.readlines()
        for line in lines:
            stop_words.append(line.strip())
```

```
    #
    my_words = ['  ', '  ', '1 ', '   ', '   ', '  ', '   ']
    for i in my_words:
        jieba.add_word(i)

    #
    my_stop_words = [' ', ' ',]
    stop_words.extend(my_stop_words)

    #
    word_num = jieba.lcut(content_series.str.cat(sep=' '), cut_all=False)

    #
    word_num_selected = [i for i in word_num if i not in stop_words and␣
↪len(i)>=2]

    return word_num_selected
```

[13]:
```
text1 = get_cut_words(content_series=df.contents)
text1[:5]
```

```
Building prefix dict from the default dictionary …
Loading model from cache C:\Users\Administrator\AppData\Local\Temp\jieba.cache
Loading model cost 1.111 seconds.
Prefix dict has been built successfully.
```

[13]:
```
[' ', ' ', ' ', '  ', ' ']
```

[14]:
```
import stylecloud
from IPython.display import Image

stylecloud.gen_stylecloud(text=' '.join(text1), collocations=False,
                          font_path=r'C:\Windows\Fonts\msyh.ttc',
                          icon_name='fas fa-play-circle',size=400,
                          output_name='   - .png')
Image(filename='   - .png')
```

[14]:


孩子
严良
普普
爬山
演技
妈妈
爸爸
喜欢
已经
知道
张东升
厉害
真实
有点
一起
不错
儿子
女儿
这是
剧情
女孩
可爱
相机
导演
现在
后妈
太好了
哈哈哈
音乐
演员
朋友
女人
问题
衣服
老师
张老师
王瑶
聪明
机会
秦昊
弟弟
盒饭
片尾曲
小女孩
看到
报警
好像
这种
正常
男人
细节
叔叔
小白船
花钱
这部
学霸
晶晶
善良
两个
三个
一集
害怕
东西
声音
人头
时长
智商
眼神
头发
他爸
长得
重要
有钱
欣欣
学习
童话
简单
配乐
为啥
相信
发现
家庭
懂事
开心
直接
是不是
好好
心机
几个
哈哈哈哈