微博舆论分析项目

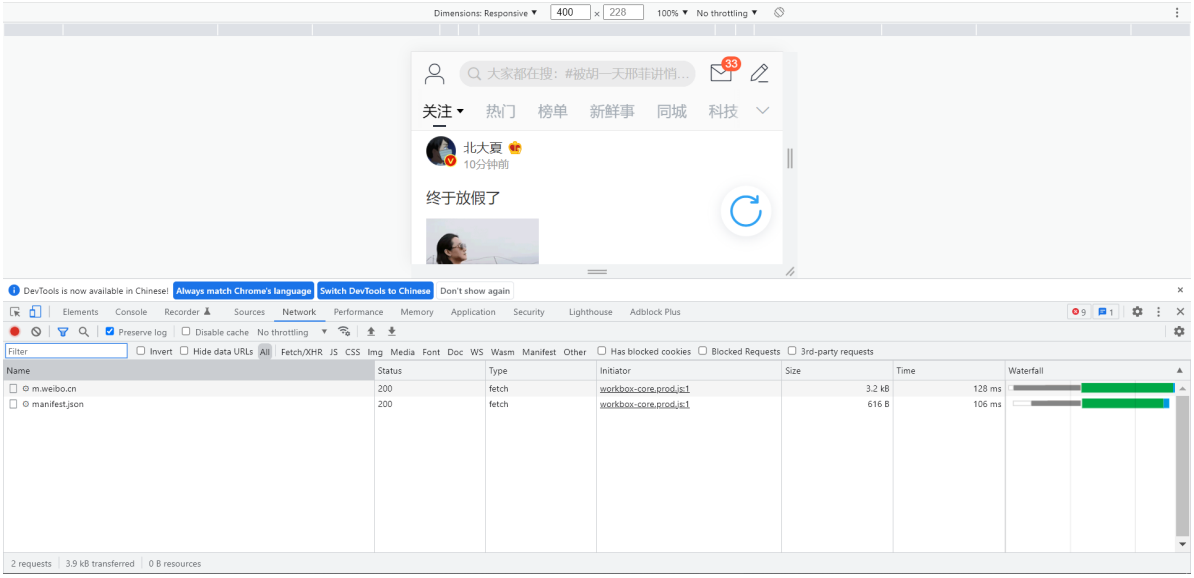
1、数据获取项目

数据获取是通过 crwal.py

这个爬虫程序去获取的

网站是这个手机版网页

然后去获取首先是网页的检查这边去获取对应的id账号



然后把这个位置修改成想要获取到的id账号

```
import re
import pandas as pd

#对应微博该条评论的id号
id1 = '4700724870775299'
```

然后去获取对应的API文件

```
#获取第一页评论的内容
def first_page_comment(url):
    try:
        html = requests.get(url, headers=headers, timeout=20)
        js_con = html.json()
        max_id = js_con['data']['max_id']  # 下一页的max_id
        max = js_con['data']['max']  # 获取最大页数
        comments_list = js_con['data']['data']  # 提取评论内容
        extract_data(comments_list)
        print("已获取第1页的评论")
        return max_id, max, commentLists
    except Exception as e:
        print("遇到异常")
```

根据他的第一页的内容获取下一页的内容

然后重复该过程

```python
if __name__ == '__main__':
    #原始URL
    url = 'https://m.weibo.cn/comments/hotflow?id={}&mid={}&max_id_type=0'.format(id1,id1)
    max_id, max_page, output = first_page_comment(url)

    if len(output) > 0:
        url1 = "https://m.weibo.cn/comments/hotflow?id="
        # 如果结果不只一页，就继续爬
        if (max_page != 1):
            urlNew = 'https://m.weibo.cn/comments/hotflow?id={}&mid={}&max_id={}&max_id_type=0'.format(id1,id1,max_id)
```

最后获取到的内容

```python
#把获取到的内容写入到csv文件中
def write_in(index: str):
    global commentLists
    df = pd.DataFrame()
    df['commentor_name'] = []
    df['comment_text'] = []
    df['create_time'] = []
    df['like_count'] = []
    df['reply_number'] = []
    df.to_csv('{}-{}.csv'.format(id1,index), encoding='utf-8',index=None)
    for obj in commentLists:
        df['commentor_name'] = [obj['commentor_name']]
        df['comment_text'] = [obj['comment_text']]
        df['create_time'] = [obj['create_time']]
        df['like_count'] = [obj['like_count']]
        df['reply_number'] = [obj['reply_number']]
        df.to_csv('{}-{}.csv'.format(id1,index),encoding='utf-8',mode='a+',header=None,index=None)
```

用pandas写入csv文件里面

2、数据预处理

先把全部的内容汇集到一个dataframe文件里面

]:
```python
import pandas as pd
```

]:
```python
data1 = pd.read_excel('./data/数据(1).xlsx')
data2 = pd.read_excel('./data/数据(2).xlsx')
data3 = pd.read_excel('./data/数据(3).xlsx')
data4 = pd.read_excel('./data/数据(4).xlsx')
data5 = pd.read_excel('./data/数据(5).xlsx')
data6 = pd.read_excel('./data/数据(6).xlsx')
data7 = pd.read_excel('./data/数据(7).xlsx')
```

]:
```python
sum_data = pd.concat([data1,data2,data3,data4,data5,data6,data7])
sum_data
```

然后去除空值和重复项内容

```python
#删除空值和重复项
comment_text = sum_data['comment_text']
comment_text.dropna(how='any', inplace=True)
comment_text.drop_duplicates(keep='first', inplace=True)
comment_text
```

删除完之后从 4万多条数据变成1万多条数据

47075 rows × 5 columns

```
#删除空值和重复项
comment_text = sum_data['comment_text']
comment_text.dropna(how='any', inplace=True)
comment_text.drop_duplicates(keep='first', inplace=True)
comment_text
```

```
0                                    恭喜EDG！！冠军🏆！
1                                         恭喜EDG！
2                                    我们！是！世界冠军🏆！
3                                 EDG太棒了，LPL是冠军！
4                          恭喜电宝！！！！！！！！！！！！！！！！！！
                          ...
68                                 恭喜EDG。  网页链接
70                                        竞圈公狗
71     我看到了非常恶心的事因为喊了句DK牛逼，而发生了校园暴力，还是大规模群欧，真牛逼
73              人家是冠军，你还是一猥琐男，冷静一点儿别麻烦医生护士消防员警察
74                             今晚真的打开眼界，不要秀下限了好吗
Name: comment_text, Length: 15816, dtype: object
```

然后再删除一些标点符号

```
#删除标点符号
import re
def clear_characters(text):
    return re.sub('\W', '', text)
```

```
comment_text = comment_text.astype(str)
comment_text = comment_text.apply(clear_characters)
comment_text
```

```
0                                    恭喜EDG冠军
1                                      恭喜EDG
2                                    我们是世界冠军
3                                 EDG太棒了LPL是冠军
4                                      恭喜电宝
                          ...
68                                 恭喜EDG网页链接
70                                      竞圈公狗
71     我看到了非常恶心的事因为喊了句DK牛逼而发生了校园暴力还是大规模群欧真牛逼
73              人家是冠军你还是一猥琐男冷静一点儿别麻烦医生护士消防员警察
74                             今晚真的打开眼界不要秀下限了好吗
Name: comment_text, Length: 15816, dtype: object
```

然后采用机械压缩把文本内容进行压缩

```
#定义机械压缩函数
def yasuo(st):
    for i in range(1,int(len(st)/2)+1):
        for j in range(len(st)):
            if st[j:j+i] == st[j+i:j+2*i]:
                k = j + i
                while st[k:k+i] == st[k+i:k+2*i] and k<len(st):
                    k = k + i
                st = st[:j] + st[k:]
    return st
```

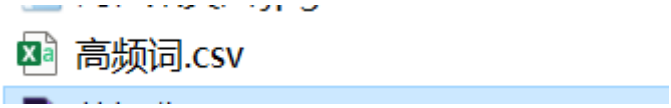
```
comment_text = comment_text.apply(yasuo)
comment_text
```

```
0                                   恭喜EDG冠军
1                                     恭喜EDG
2                                   我们是世界冠军
3                              EDG太棒了LPL是冠军
4                                     恭喜电宝
                    ...
68                                恭喜EDG网页链接
70                                     竞圈公狗
71    我看到了非常恶心的事因为喊了句DK牛逼而发生了校园暴力还是大规模群欧真牛逼
73            人家是冠军你还是一猥琐男冷静一点儿别麻烦医生护士消防员警察
74                          今晚真的打开眼界不要秀下限了好吗
Name: comment_text, Length: 15816, dtype: object
```

然后我们再用停用词和分词进行高频词统计

```
import jieba

#可视化库
import stylecloud
from IPython.display import Image
# 定义分词函数
def get_cut_words(content_series):
    # 读入停用词表
    stop_words = []

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

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

    # 条件筛选
    word_num_selected = [i for i in word_num if i not in stop_words and len(i)>=2]
    return word_num_selected
```

```
#分词，寻找高频词
ditc = {}
list_word = []
list_count = []
for t in text1:
    ditc[t] = ditc.get(t, 0) + 1
ls = list(ditc.items())
ls.sort(key=lambda x: x[1], reverse=True)
for i in range(len(ls)):
    word, count = ls[i]
    list_word.append(word)
    list_count.append(count)

df1 = pd.DataFrame()

df1['word'] = list_word
df1['count'] = list_count
df1.to_csv('高频词.csv',encoding='gbk')
```



3、数据分析

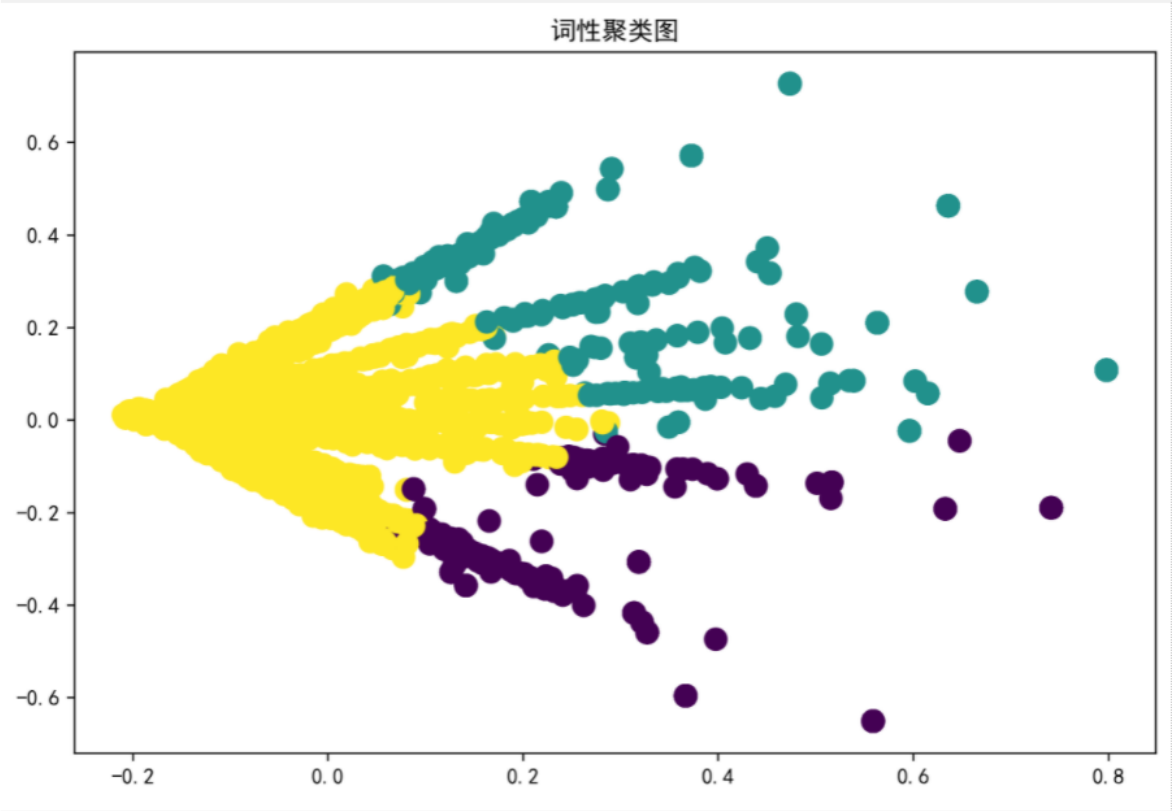
我们来计算它的tf-idf的值，进行聚类可视化来查看

先计算好它的tf-idf

```
#第一步 计算TFIDF
# 文档预料 空格连接
corpus = []

# 读取预料 一行预料为一个文档
for line in open('fenci.txt', 'r',encoding='utf-8').readlines():
    corpus.append(line.strip())
# 将文本中的词语转换为词频矩阵 矩阵元素a[i][j] 表示j词在i类文本下的词频
vectorizer = CountVectorizer()

# 该类会统计每个词语的tf-idf权值
transformer = TfidfTransformer()

# 第一个fit_transform是计算tf-idf 第二个fit_transform是将文本转为词频矩阵
tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
# 获取词袋模型中的所有词语
word = vectorizer.get_feature_names()

# 将tf-idf矩阵抽取出来 元素w[i][j]表示j词在i类文本中的tf-idf权重
weight = tfidf.toarray()

# 打印特征向量文本内容
print('Features length: ' + str(len(word)))
```

计算好之后用聚类的可视化来进行查看

```
print('Start Kmeans:')
from sklearn.cluster import KMeans

clf = KMeans(n_clusters=3)
print(clf)
pre = clf.fit_predict(weight)
print(pre)

# 中心点
print(clf.cluster_centers_)
print(clf.inertia_)

#第三步 图形输出 降维

from sklearn.decomposition import PCA

pca = PCA(n_components=3)  # 输出两维
newData = pca.fit_transform(weight)  # 载入N维

x = [n[0] for n in newData]
y = [n[1] for n in newData]
plt.figure(figsize=(9,6),dpi = 300)
plt.rcParams['font.sans-serif'] = ['SimHei']  # 支持中文
plt.rcParams['axes.unicode_minus'] = False
plt.scatter(x, y, c=pre, s=100)
plt.title("词性聚类图")
plt.savefig('词性聚类图.jpg')
plt.show()
```



然后我们再用snownlp来进行数据分类，把正面和负面的评论进行区分出来

顺便把它们的训练集和测试集区分出来

```
from snownlp import SnowNLP
from snownlp import sentiment
from snownlp.sentiment import Sentiment
from snownlp import sentiment
from tqdm import tqdm
import re
# 按文章的说法，初步筛选语料，大于0.8的归入积极，小于0.3的归入消极。
def train_model(texts):
    for comm in tqdm(texts):
        if comm != "":
            text = re.sub(r'(?:回复)?(?://)?@[\w\u2E80-\u9FFF]+:?|\[\w+\]', ',',comm)
            score = SnowNLP(text)
            if score.sentiments > 0.8:
                with open('train_pos.txt', mode='a', encoding='utf-8') as g:
                    g.writelines(comm +"\n")
            elif score.sentiments < 0.3:
                with open('train_neg.txt', mode='a', encoding='utf-8') as f:
                    f.writelines(comm + "\n")
            else:
                pass
```

```
def test_model(texts):
    for comm in tqdm(texts):
        if comm != "":
            text = re.sub(r'(?:回复)?(?://)?@[\w\u2E80-\u9FFF]+:?|\[\w+\]', ',',comm)
            score = SnowNLP(text)
            if score.sentiments > 0.8:
                with open('test_pos.txt', mode='a', encoding='utf-8') as g:
                    g.writelines(comm +"\n")
            elif score.sentiments < 0.3:
                with open('test_neg.txt', mode='a', encoding='utf-8') as f:
                    f.writelines(comm + "\n")
            else:
                pass
```

```
%%time
comment_text = comment_text.astype(str)
train_model(comment_text[0:int(len(comment_text)*0.7)])
test_model(comment_text[int(len(comment_text)*0.7):len(comment_text)])
```

test_neg.txt	2022/1/27 14:40
test_pos.txt	2022/1/27 14:40
train_neg.txt	2022/1/27 14:39
train_pos.txt	2022/1/27 14:39

然后我们用机器学习中的贝叶斯进行数据分类

```
train_words_list1, train_labels1 = loadfile('train_pos.txt', '1')
train_words_list2, train_labels2 = loadfile('train_neg.txt', '0')

train_words_list = train_words_list1 + train_words_list2
train_labels = train_labels1 + train_labels2

test_words_list1, test_labels1 = loadfile('test_pos.txt', '1')
test_words_list2, test_labels2 = loadfile('test_neg.txt', '0')

test_words_list = test_words_list1 + test_words_list2
test_labels = test_labels1 + test_labels2

train_comments_new = [deal_text(comment, "stopwords_cn.txt") for comment in train_words_list]
test_comments_new = [deal_text(comment, "stopwords_cn.txt") for comment in test_words_list]


stop_words = open('stopwords_cn.txt','r',encoding='utf-8').read()
stop_words = stop_words.encode('utf-8').decode('utf-8-sig') #列表头部\ufeff处理
stop_words = stop_words.split('\n') #根据分隔符分隔
```

```
#计算单词权重
tf = TfidfVectorizer(stop_words=stop_words,max_df=0.5)

train_features = tf.fit_transform(train_comments_new)
#上面fit过了,这里transform
test_features = tf.transform(test_comments_new)

#多项式贝叶斯分类器
clf = MultinomialNB(alpha=0.001).fit(train_features,train_labels)
predicted_labels = clf.predict(test_features)
#计算准确率
print('准确率：',accuracy_score(test_labels,predicted_labels))
```

```
准确率： 0.881351689612015
```

预测出来的准确率为0.88，准确率一般在0.8以上都是处于较好的训练模型，所以这个模型已经算是比较可以的了

数据可视化

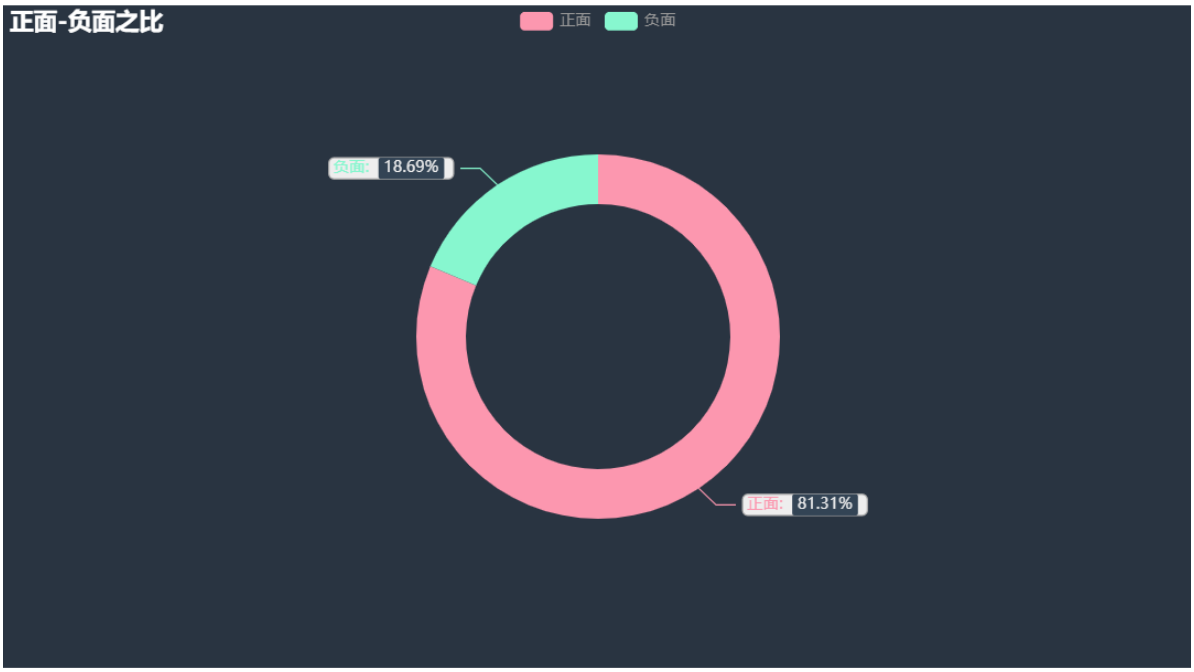
最后我们根据我们上面处理的数据进行数据可视化

词云图

```
from pyecharts import options as opts
from pyecharts.charts import WordCloud
from pyecharts.globals import ThemeType
from pyecharts.globals import SymbolType
x_data = list_word[0:100]
y_data = list_count[0:100]

data_pair = [(x,int(y)) for x,y in zip(x_data,y_data)]

c = (
    WordCloud(init_opts=opts.InitOpts(theme=ThemeType.CHALK))
    .add("", data_pair, word_size_range=[20, 100], shape=SymbolType.DIAMOND)
    .set_global_opts(title_opts=opts.TitleOpts(title="高频词-词云图"))
    .render("wordcloud.html")
)
```

饼图

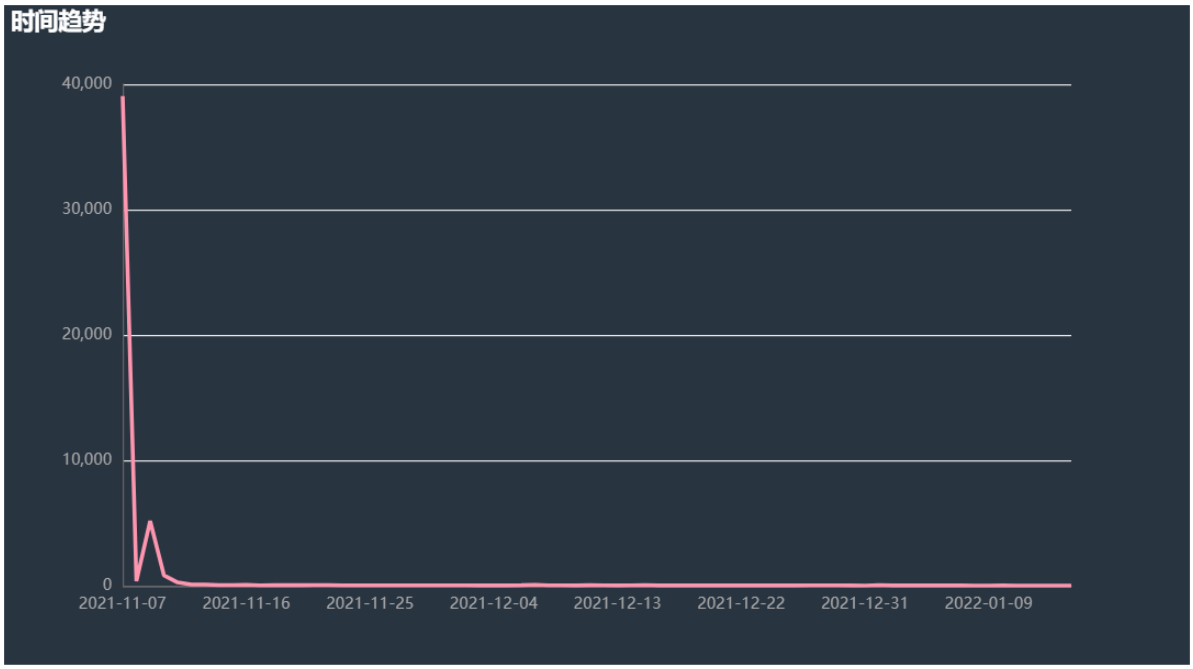
```
from pyecharts import options as opts
from pyecharts.charts import Pie
from pyecharts.faker import Faker

c = (
    Pie(init_opts=opts.InitOpts(theme=ThemeType.CHALK))
    .add(
        "",
        data_pair1,
        radius=["40%", "55%"],
        label_opts=opts.LabelOpts(
            position="outside",
            formatter=" {b}:  {per|{d}%}  ",
            background_color="#eee",
            border_color="#aaa",
            border_width=1,
            border_radius=4,
            rich={
                "a": {"color": "#999", "lineHeight": 22, "align": "center"},
                "abg": {
                    "backgroundColor": "#e3e3e3",
                    "width": "100%",
                    "align": "right",
                    "height": 22,
                    "borderRadius": [4, 4, 0, 0],
                },
                "hr": {
                    "borderColor": "#aaa",
                    "width": "100%",
                    "borderWidth": 0.5
```



查看正面和负面之间的占比情况

时间趋势图

```
c = (

    Line(init_opts=opts.InitOpts(theme=ThemeType.CHALK))
    .add_xaxis(xaxis_data=x_data)
    .add_yaxis(
        series_name="",
        symbol="emptyCircle",
        is_symbol_show=False,
        color="#ADFF2F",
        y_axis=y_data,
        label_opts=opts.LabelOpts(is_show=False),
        linestyle_opts=opts.LineStyleOpts(width=3)
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="时间趋势"),
        tooltip_opts=opts.TooltipOpts(trigger="axis"),
        yaxis_opts=opts.AxisOpts(
            type_="value",
            axistick_opts=opts.AxisTickOpts(is_show=True),
            axislabel_opts=opts.LabelOpts(formatter="{value}"),
            splitline_opts=opts.SplitLineOpts(is_show=True),
        ),
        xaxis_opts=opts.AxisOpts(type_="category", boundary_gap=False,axisline_opts=opts.AxisLi
                is_on_zero=False,
            )),
    )
    .render("line.html")
)
```



人物领袖图

```
from pyecharts.charts import Bar
from pyecharts.globals import ThemeType
x_data1.reverse()
y_data1.reverse()
c = (
    Bar(init_opts=opts.InitOpts(width="1600px", height="800px",theme=ThemeType.CHALK))
    .add_xaxis(x_data1)
    .add_yaxis("",y_data1,label_opts=opts.LabelOpts(is_show=False))
    .reversal_axis()
    .set_global_opts(
        title_opts={"text": "前20影响力最大的人"}
    )
    .render("bar.html")
)
```