GIS Visualization Homework 2

Homework2(1). Basic requirements

1) Python Spider

In [1]:

```
import requests
from bs4 import BeautifulSoup
import random

#User-Agent列表，这个可以自己在网上搜到，用于伪装浏览器的User Agent
USER_AGENTS = [
        "Mozilla/5.0 (Windows; U; Windows NT 5.2) Gecko/2008070208 Firefox/3.0.1"
        "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-us) AppleWebKit/534.50 (KHTML, like
        "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0.1) Gecko/20100101 Firefox/4.
        "Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0.1",
        "Opera/9.80 (Macintosh; Intel Mac OS X 10.6.8; U; en) Presto/2.8.131 Version/11.1
        "Opera/9.80 (Windows NT 6.1; U; en) Presto/2.8.131 Version/11.11",
        "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_0) AppleWebKit/535.11 (KHTML, like G
        "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR
        "Opera/9.80 (Windows NT 5.1; U; zh-cn) Presto/2.9.168 Version/11.50",
        "Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0",
        "Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.11 (KHTML, like Gecko) Chrom
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome
        "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .N
        "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; SV1; QQDownload
        "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.
        "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0)",
        "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2)",
        "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)",
        "Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)",
        "Mozilla/5.0 (Windows; U; Windows NT 5.2) Gecko/2008070208 Firefox/3.0.1",
        "Mozilla/5.0 (Windows; U; Windows NT 5.1) Gecko/20070309 Firefox/2.0.0.3",
        "Mozilla/5.0 (Windows; U; Windows NT 5.1) Gecko/20070803 Firefox/1.5.0.12 "
        ]
#IP地址列表，用于设置IP代理
IP_AGENTS = [
        "http://58.240.53.196:8080",
        "http://219.135.99.185:8088",
        "http://117.127.0.198:8080",
        "http://58.240.53.194:8080"
]

#设置IP代理
proxies={"http":random.choice(IP_AGENTS)}
#设置cookie
Cookie='bid=lOl_O8fsv-Q; douban-fav-remind=1; __utmz=30149280.1632215122.1.1.utmcsr=(dire

def getHTML(url):
    headers={'User-agent':random.choice(USER_AGENTS),
#             'Cookie':Cookie,
#             'Connection':'keep-alive',
#             'Accept':'application/json, text/plain, */*',
#             'Accept-Encoding':'gzip, deflate, br',
#             'Accept-Language':'zh-CN,zh;q=0.9'
            }
    r = requests.get(url,headers=headers,proxies=proxies)
    r.raise_for_status()
    # print("get html successfully")
    r.encoding = 'utf-8'
    # print(r.text)
```

```
    return r.text

def parseHTML(html):
    try:
        soup = BeautifulSoup(html,"html.parser")
        A = soup.find_all('span',attrs = {'class':'short'})
        B = []
        for i in A:
            B.append(i.get_text())
        return B
    except:
        return []
def main1():
    discuss = []
    text=''
    a = 0
    for i in range(0,100,20):
        url = 'https://movie.douban.com/subject/26100958/comments?start='+ str(i) +'&limi
        HTMLpage = getHTML(url)
#         print(HTMLpage)
        for t in parseHTML(HTMLpage):
            discuss.append(t)
    for i in discuss:
        text+=i
#         print(str(a) + ':' + i)
#         a = a + 1
    return text
text=main1()
# print(text)
```

Homework2(2). Real-time Operation

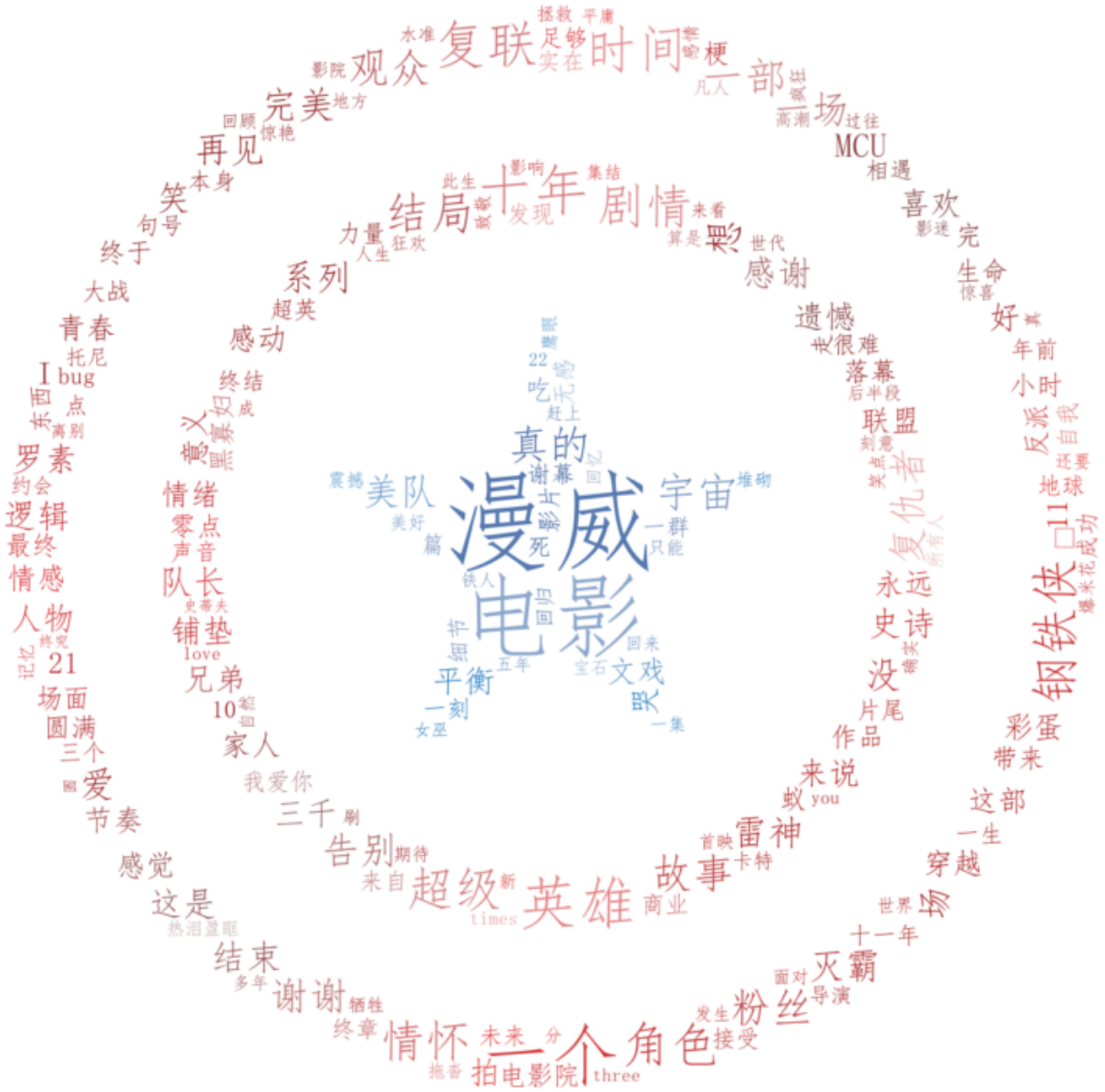
In [53]:

```
# 主要功能自定义设置
# Analysis_text = 'Data/复联.txt'         # 分析文档
userdict = 'Data/用户词典.txt'              # 用户词典
StopWords = 'Data/停用词库.txt'             # 停用词库
number = 200                            # 统计个数
Output = 'Data/词频.txt'                    # 输出文件
background = 'Data/词频背景.png'            # 词频背景
background='Data/图片2.png'
# 导入扩展库
import re                               # 正则表达式库
import jieba                            # 结巴分词
import jieba.posseg                     # 词性获取
import collections                      # 词频统计库
import numpy                            # numpy数据处理库
from PIL import Image                   # 图像处理库
import wordcloud                        # 词云展示库
import matplotlib.pyplot as plt         # 图像展示库（这里以plt代表库的全称）

# 读取文件
# # fn = open(Analysis_text,'r',encoding = 'UTF-8')  # 打开文件
# string_data = fn.read()                           # 读出整个文件
# fn.close()                                        # 关闭文件

# 不进行文件读取，爬取数据的实时读写
string_data=text
# 文本预处理
pattern = re.compile(u'\t|\n|\.|-|:|;|\)|\(|\?|"') # 定义正则表达式匹配模式（空格等）
string_data = re.sub(pattern, '', string_data)     # 将符合模式的字符去除

# 动态调整词典
jieba.suggest_freq('钢铁侠', True)     #True表示该词不能被分割，False表示该词能被分割
```

```
# 添加用户词典
jieba.load_userdict(userdict) #和上面动态调整词典的作用是一样的

# 文本分词
seg_list_exact = jieba.cut(string_data, cut_all=False, HMM=True)    # 精确模式分词+HMM模
object_list = []

# 去除停用词（去掉一些意义不大的词，如标点符号、嗯、啊等）
with open(StopWords, 'r', encoding='UTF-8') as meaninglessFile:
    stopwords = set(meaninglessFile.read().split('\n'))
stopwords.add(' ')
for word in seg_list_exact:          # 循环读出每个分词
    if word not in stopwords:        # 如果不在去除词库中
        object_list.append(word)     # 分词追加到列表

# 词频统计
word_counts = collections.Counter(object_list)       # 对分词做词频统计
word_counts_top = word_counts.most_common(number)    # 获取前number个最高频的词
# 英文词性转中文词性字典：详细版
En2Cn_Pro = {
    'a'     : '形容词',
    'ad'    : '形容词-副形词',
    'ag'    : '形容词-形容词性语素',
    'al'    : '形容词-形容词性惯用语',
    'an'    : '形容词-名形词',
    'b'     : '区别词',
    'bl'    : '区别词-区别词性惯用语',
    'c'     : '连词',
    'cc'    : '连词-并列连词',
    'd'     : '副词',
    'e'     : '叹词',
    'eng'   : '英文',
    'f'     : '方位词',
    'g'     : '语素',
    'h'     : '前缀',
    'i'     : '成语',
    'j'     : '简称略语',
    'k'     : '后缀',
    'l'     : '习用语',
    'm'     : '数词',
    'mq'    : '数量词',
    'n'     : '名词',
    'ng'    : '名词-名词性语素',
    'nl'    : '名词-名词性惯用语',
    'nr'    : '名词-人名',
    'nr1'   : '名词-汉语姓氏',
    'nr2'   : '名词-汉语名字',
    'nrf'   : '名词-音译人名',
    'nrfg'  : '名词-人名',
    'nrj'   : '名词-日语人名',
    'nrt'   : '名词',
    'ns'    : '名词-地名',
    'nsf'   : '名词-音译地名',
    'nt'    : '名词-机构团体名',
    'nz'    : '名词-其他专名',
    'o'     : '拟声词',
    'p'     : '介词',
    'pba'   : '介词-“把”',
    'pbei'  : '介词-“被”',
    'q'     : '量词',
    'qt'    : '量词-动量词',
    'qv'    : '量词-时量词',
    'r'     : '代词',
    'rg'    : '代词-代词性语素',
    'rr'    : '代词-人称代词',
    'rz'    : '代词-指示代词',
    'rzs'   : '代词-处所指示代词',
```

```
    'rzt'   : '代词-时间指示代词',
    'rzv'   : '代词-谓词性指示代词',
    'ry'    : '代词-疑问代词',
    'rys'   : '代词-处所疑问代词',
    'ryt'   : '代词-时间疑问代词',
    'ryv'   : '代词-谓词性疑问代词',
    's'     : '处所词',
    't'     : '时间词',
    'tg'    : '时间词-时间词性语素',
    'u'     : '助词',
    'ude1'  : '助词-“的”“底”',
    'ude2'  : '助词-“地”',
    'ude3'  : '助词-“得”',
    'udeng' : '助词-“等”“等等”“云云”',
    'udh'   : '助词-“的话”',
    'uguo'  : '助词-“过”',
    'ule'   : '助词-“了”“喽”',
    'ulian' : '助词-“连”',
    'uls'   : '助词-“来讲”“来说”“而言”“说来”',
    'usuo'  : '助词-“所”',
    'uyy'   : '助词-“一样”“一般”“似的”“般”',
    'uz'    : '助词-“则”',
    'uzhe'  : '助词-“着”',
    'uzhi'  : '助词-“之”',
    'v'     : '动词',
    'vd'    : '动词-副动词',
    'vf'    : '动词-趋向动词',
    'vg'    : '动词-动词性语素',
    'vi'    : '动词-不及物动词（内动词）',
    'vl'    : '动词-动词性惯用语',
    'vn'    : '动词-名动词',
    'vshi'  : '动词-“是”',
    'vx'    : '动词-形式动词',
    'vyou'  : '动词-“有”',
    'w'     : '标点符号',
    'wb'    : '标点符号-百分号千分号，全角：％ ‰ 半角：%',
    'wd'    : '标点符号-逗号，全角：，  半角：,',
    'wf'    : '标点符号-分号，全角：；  半角：  ;',
    'wj'    : '标点符号-句号，全角：。',
    'wh'    : '标点符号-单位符号，全角：￥ ＄ ￡ ° ℃ 半角 $',
    'wkz'   : '标点符号-左括号，全角：（ 〔 ［ ｛ 《 【 〖 〈 半角：( [ { <',
    'wky'   : '标点符号-右括号，全角：） 〕 ］ ｝ 》 】 〗 〉 半角：  ) ] { >',
    'wm'    : '标点符号-冒号，全角：：  半角：  :',
    'wn'    : '标点符号-顿号，全角：、',
    'wp'    : '标点符号-破折号，全角：—— －－ ———  半角：—',
    'ws'    : '标点符号-省略号，全角：…… …',
    'wt'    : '标点符号-叹号，全角：！  半角：!',
    'ww'    : '标点符号-问号，全角：？  半角：?',
    'wyz'   : '标点符号-左引号，全角：“  ‘  『',
    'wyy'   : '标点符号-右引号，全角：”  ’  』',
    'x'     : '字符串',
    'xu'    : '字符串-网址URL',
    'xx'    : '字符串-非语素字',
    'y'     : '语气词',
    'z'     : '状态词',
    'zg'    : '量词-数量词',
    'un'    : '未知词',
}

# 输出至工作台，并导出“词频.txt”文件
#print ('\n词语\t词频\t词性')
# print ('————————————')
fileOut = open(Output,'w',encoding='UTF-8')      # 创建文本文件；若已存在，则进行覆盖
fileOut.write('词语\t词频\t词性\n')
fileOut.write('————————————\n')
count = 0
Wordlist=[]
```

```
Frelist=[]
POSlist=[]
for TopWord,Frequency in word_counts_top:                          # 获取词语和词频
    for POS in jieba.posseg.cut(TopWord):                          # 获取词性
        if count == number:
            break
        #print(TopWord + '\t',str(Frequency) + '\t',list(En2Cn_Pro.values())[list(En2Cn_P
        fileOut.write(TopWord + '\t' + str(Frequency) + '\t' + list(En2Cn_Pro.values())[
        Wordlist.append(TopWord)
        Frelist.append(Frequency)
        POSlist.append(list(En2Cn_Pro.values())[list(En2Cn_Pro.keys()).index(POS.flag)])
        count += 1
fileOut.close()                                                    # 关闭文件

# 词频展示
# print ('\n开始制作词云……')                                        # 提示当前状态
mask = numpy.array(Image.open(background))                         # 定义词频背景
wc = wordcloud.WordCloud(
    font_path = 'C:/Windows/Fonts/simfang.ttf', # 设置字体（这里选择“仿宋”）
    background_color='white',                   # 背景颜色
    mask = mask,                                # 文字颜色+形状（有mask参数再设定宽高是无
    max_words = number,                         # 显示词数
    max_font_size = 150                         # 最大字号
)

wc.generate_from_frequencies(word_counts)                                          # 从字典
wc.recolor(color_func=wordcloud.ImageColorGenerator(mask))                         # 将词云
plt.figure(figsize=(10,10))
# plt.figure('词云')                                                               # 弹框
plt.subplots_adjust(top=0.99,bottom=0.01,right=0.99,left=0.01,hspace=0,wspace=0) # 调整边
plt.imshow(wc, cmap=plt.cm.gray, interpolation='bilinear')                         # 处理词
plt.axis('off')                                                                    # 关闭坐
# print ('制作完成！')                                                             # 提示
plt.show()
```



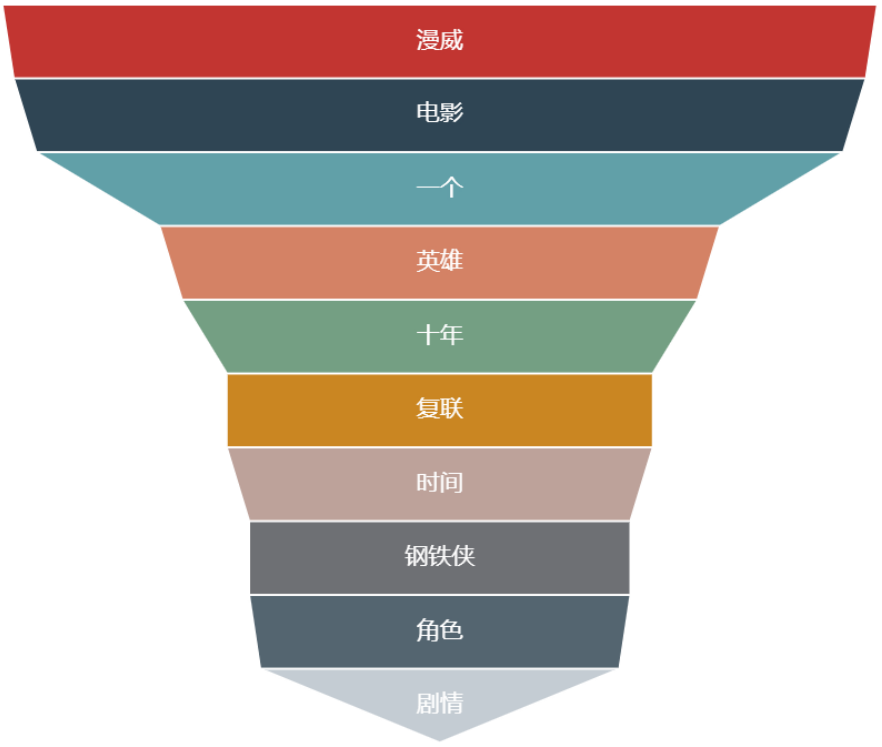
Homework2(3). Statistic

1) Funnel

In [3]:

```
from pyecharts import options as opts
from pyecharts.charts import Funnel
from pyecharts.faker import Faker
# 将频率前10的词作漏斗图
Wordlist10=Wordlist[0:10]
Frelist10=Frelist[0:10]

(
    Funnel(init_opts=opts.InitOpts(width="610px"))
    .add(
        "word",
        [list(z) for z in zip(Wordlist10, Frelist10)],
        label_opts=opts.LabelOpts(position="inside"),
    )
    .set_global_opts(title_opts=opts.TitleOpts())
    .render_notebook()
)
```

Out[3]:



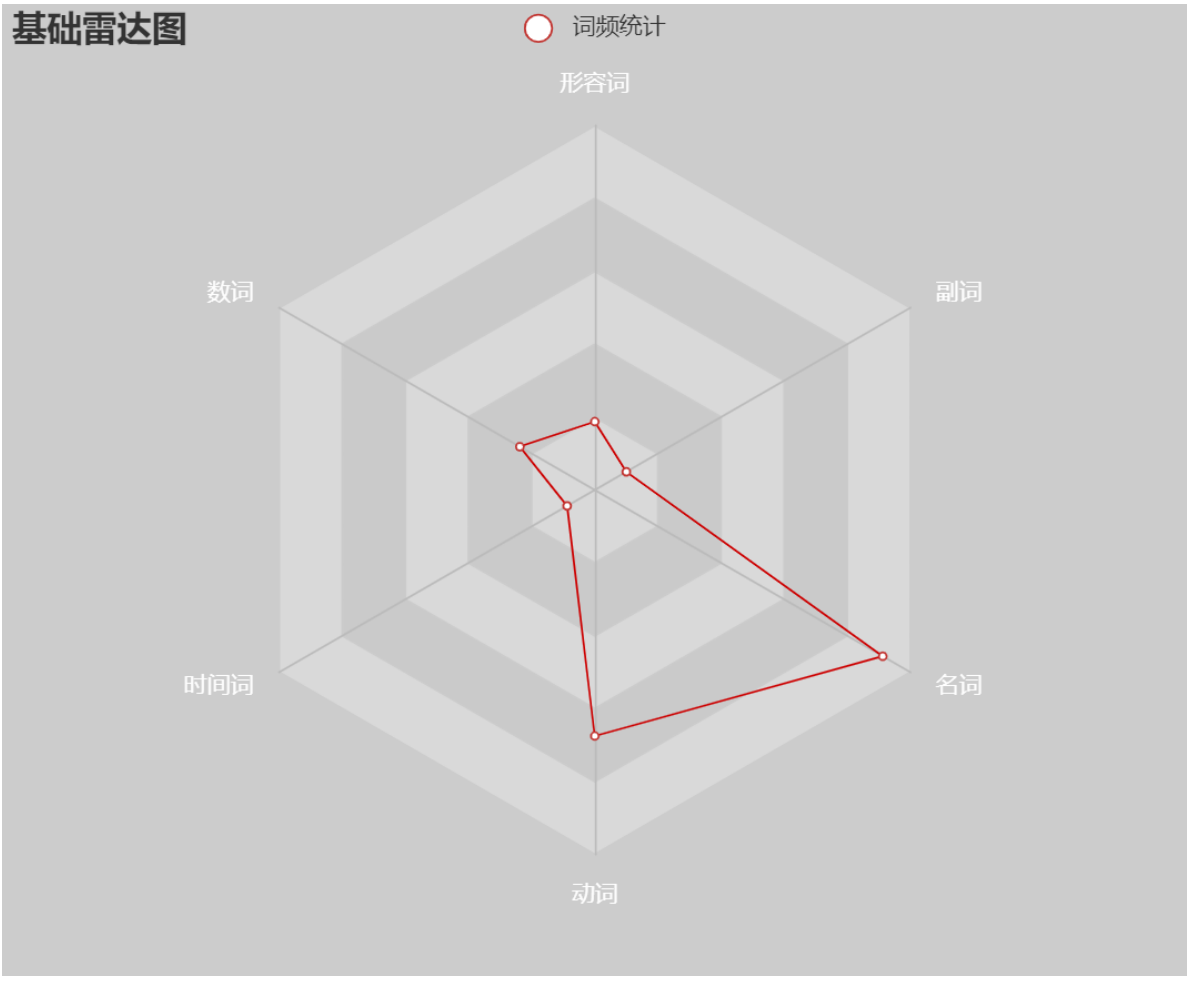
2) Radar

In [54]:

```
from pyecharts.charts import Radar
import random
# 统计词性，做雷达图
sum_POSlist=[]
# 由于部分词性细分了，这里将其还原成总词性
for i in POSlist:
    pos=i.find('-')
    if pos>0:
        i=i[0:pos]
        sum_POSlist.append(i)
    else:
        sum_POSlist.append(i)
# 使用counter统计词频
POS_counter=collections.Counter(sum_POSlist)
POS_child_counter=collections.Counter(POSlist)
POS_counter_top = POS_counter.most_common(6)
# 打乱从高到低的默认顺序
random.shuffle(POS_counter_top)
# 各个词性的频数列表
POS_value=[]
for i in POS_counter_top:
    POS_value.append(i[1])
POS_value=[POS_value]

(
    Radar(init_opts=opts.InitOpts(width="610px", bg_color="#CCCCCC"))
    .add_schema(
        schema=[
            opts.RadarIndicatorItem(name=POS_counter_top[0][0], max_=80),
            opts.RadarIndicatorItem(name=POS_counter_top[1][0], max_=80),
            opts.RadarIndicatorItem(name=POS_counter_top[2][0], max_=80),
            opts.RadarIndicatorItem(name=POS_counter_top[3][0], max_=80),
            opts.RadarIndicatorItem(name=POS_counter_top[4][0], max_=80),
            opts.RadarIndicatorItem(name=POS_counter_top[5][0], max_=80),
        ],
```

```
        splitarea_opt=opts.SplitAreaOpts(
            is_show=True, areastyle_opts=opts.AreaStyleOpts(opacity=1)
        ),
        textstyle_opts=opts.TextStyleOpts(color="#fff"),
    )
    .add(
        series_name="词频统计",
        data=POS_value,
        linestyle_opts=opts.LineStyleOpts(color="#CD0000"),
    )
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(
        title_opts=opts.TitleOpts(title="基础雷达图"), legend_opts=opts.LegendOpts()
    )
    .render_notebook()
)
```

Out[54]:



3) Sunburst

In [57]:

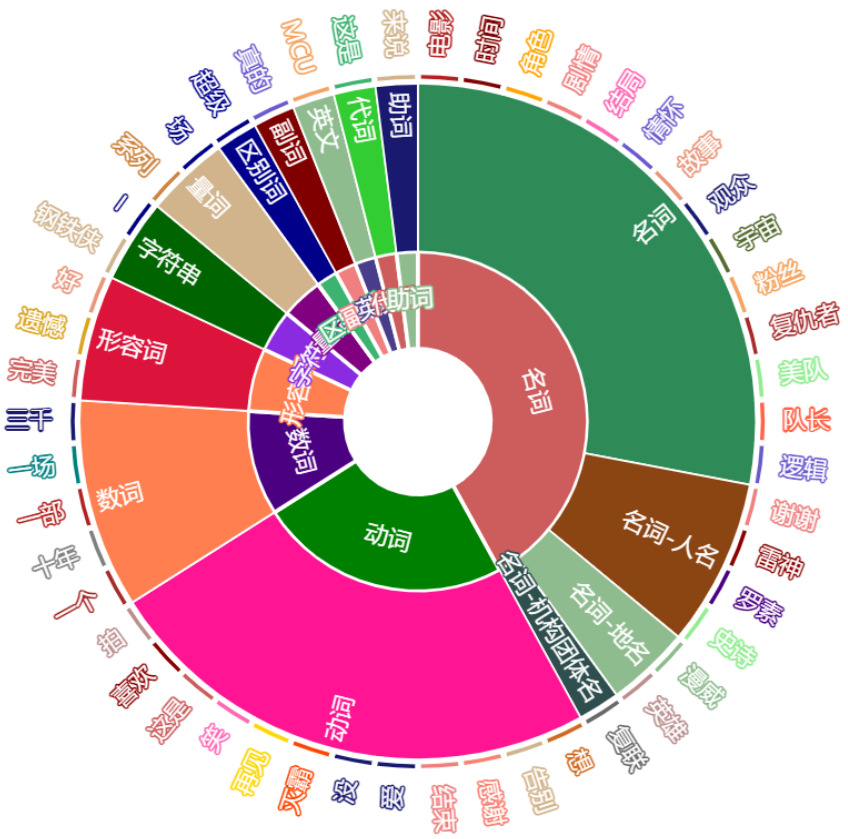
```
import json
from pyecharts.charts import Sunburst

colorlist=["#DC143C", "#DB7093", "#FF69B4", "#FF1493", "#C71585", "#DA70D6", "#8B008B", "

############################创建数据字典树 child指子节点，minchild指叶节点
# POSlist词性列表（子节点），Wordlist词列表（叶节点）
# print(POSlist)
# print(Wordlist)
# 做一下截断，不然太多了数据可视化效果不好
POSlist=POSlist[0:50]
Wordlist=Wordlist[0:50]

#子节点顺序列表，用于索引，在一次建树中使用
POS_child_Order=[]
```

```
#根节点顺序列表，用于索引，在二次建树中使用
POS_Order=[]
#叶节点对应的子节点索引，在一次建树中使用
MinChild_Label=[]
#子节点对应的根节点索引，在二次建树中使用
Child_Label=[]

#########一次建树阶段
#初始化POS_child_Order，MinChild_Label
for i in range(len(Wordlist)):
    if POSlist[i] in POS_child_Order:
        ind=POS_child_Order.index(POSlist[i])
        MinChild_Label.append(ind)
    else:
        POS_child_Order.append(POSlist[i])
        ind=POS_child_Order.index(POSlist[i])
        MinChild_Label.append(ind)
# print(POS_child_Order)
# print(MinChild_Label)

# 为子节点，叶节点创建词典列表
dict_child=[]
dict_minchild=[]
# 对每个叶节点创建词典，并加入列表
for i in range(len(Wordlist)):
    dic={
        'name':Wordlist[i],
        'value':1,
        'itemStyle':{"color":random.choice(colorlist)}
    }
    dict_minchild.append(dic)

# 对每个子节点创建词典，并加入列表
# 对i个子节点遍历
for i in range(len(POS_child_Order)):
    # tmp词典为当前子节点的叶节点集合
    tmp_dictlist=[]
    # 对所有叶节点便利
    for j in range(len(MinChild_Label)):
        if i==MinChild_Label[j]:
            tmp_dictlist.append(dict_minchild[j])
    dic={
        'name':POS_child_Order[i],
        'itemStyle':{"color": random.choice(colorlist)},
        'children':tmp_dictlist
    }
    dict_child.append(dic)

#########二次建树阶段
# 初始化POS_Order, Child_Label
for i in range(len(POSlist)):
    pos=POSlist[i].find('-')
    if pos>0:
        i=POSlist[i][0:pos]
        if i in POS_Order:
            ind=POS_Order.index(i)
            Child_Label.append(ind)
        else:
            POS_Order.append(i)
            ind=POS_Order.index(i)
            Child_Label.append(ind)
    else:
        i=POSlist[i]
        if i in POS_Order:
            ind=POS_Order.index(i)
            Child_Label.append(ind)
        else:
```

```
            POS_Order.append(i)
            ind=POS_Order.index(i)
            Child_Label.append(ind)
# print(Child_Label)
# print(POS_Order)

# 为根节点创建词典列表
dict_all=[]

# 对每个根节点创建词典，并加入列表
for i in range(len(POS_Order)):
    # tmp词典为当前根节点的子节点集合
    tmp_dictlist=[]
    # 对所有子节点遍历
    for j in range(len(POS_child_Order)):
        pos=POS_child_Order[j].find('-')
        if pos>0:
            s=POS_child_Order[j][0:pos]
            if s==POS_Order[i]:
                tmp_dictlist.append(dict_child[j])
        else:
            s=POS_child_Order[j]
            if s==POS_Order[i]:
                tmp_dictlist.append(dict_child[j])
    dic={
        'name':POS_Order[i],
        'itemStyle': {"color": random.choice(colorlist)},
        'children':tmp_dictlist
    }
    dict_all.append(dic)

# 最终转化为json格式
with open('Data/word.json','w') as dumpf:
    json.dump(dict_all,dumpf, indent=2)
# data=json.dumps(dict_all, indent=2,ensure_ascii=False)

data1=[]
with open('Data/word.json','rb') as loadf:
    data1=json.load(loadf)

(
    Sunburst(init_opts=opts.InitOpts(width="610px"))
    .add(
        "",
        data_pair=data1,
        highlight_policy="ancestor",
        radius=[0, "95%"],
        sort_="null",
        levels=[
            {},
            {
                "r0": "15%",
                "r": "35%",
                "itemStyle": {"borderWidth": 2},
                "label": {"rotate": "tangential"},
            },
            {"r0": "35%", "r": "70%", "label": {"align": "right"}},
            {
                "r0": "70%",
                "r": "72%",
                "label": {"position": "outside", "padding": 3, "silent": False},
                "itemStyle": {"borderWidth": 3},
            },
        ],
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="Sunburst-官方示例"))
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}"))
```

```
    .render_notebook()
)
```

Out[57]: **Sunburst-官方示例**



Homework2(4). Dynamic Webpages

In [52]:

```
import time
HTML='[[{"rating":["9.3","50"],"rank":1,"cover_url":"https://img2.doubanio.com\/view\/pho
HTML=HTML.replace("true","True")
HTML=HTML.replace("false","False")

def parseJSON(html):
    title=[]
    with open('Data/demo.json',mode='w',encoding='utf-8') as f:
        json.dump(html,f,ensure_ascii=False)
    with open('Data/demo.json',mode='r',encoding='utf-8') as f:
        dicts=eval(json.load(f))
        for i in range(len(dicts)):
            title.append(dicts[i]['title'])
    return title

def main2():
    Title=[]
    for i in range(0,100,20):
        url = 'https://movie.douban.com/j/chart/top_list?type=17&interval_id=100%3A90&act
        HTMLpage = getHTML(url)
        HTMLpage=HTMLpage.replace("true","True")
        HTMLpage=HTMLpage.replace("false","False")
        t=parseJSON(HTMLpage)
        Title=Title+t
        time.sleep(1.5)
    print(Title)

main2()
```

['盗梦空间', '星际穿越', '楚门的世界', '机器人总动员', '蝙蝠侠：黑暗骑士', '超感猎杀：完结特别篇', '赛文奥特曼 我是地球人', '新世纪福音战士 第0:0话 诞生之始', '少年骇客：变身之谜', '黑客帝国', '黑镜：圣诞特别篇', '攻壳机动队2：无罪', '大都会', '赛文奥特曼 光荣与传说', '红辣椒', '攻壳机动队', '赛文奥特曼 仿制的男人', '高智能方程式赛车 TV总集篇', 'V字仇杀队', '回忆三部曲', '博士之日', '赛文奥特曼 空中飞舞的大铁块', '阿凡达', '蝴蝶效应', '蝙蝠侠：黑暗骑士崛起', '黑客帝国3：矩阵革命', '终结者2：审判日', '2001太空漫游', '千钧一发', '黑客帝国动画版', '潜行者', '她的回忆', '原始星球', '赛文奥特曼 约定的结果', '头号玩家', '超能陆战队', '人工智能', '回到未来', '正义联盟：闪点悖论', '别了武器', '神秘博士：博士之时', '奈克瑟斯奥特曼剧场版', '剧场版 假面骑士零一 REAL×TIME', '赛文奥特曼 EVOLUTION EPISODE：5 神秘进化记录', '蜘蛛侠：平行宇宙', '疯狂的麦克斯4：狂暴之路', '穿越时空的少女', '黑客帝国2：重装上阵', '发条橙', 'E.T. 外星人', '机器管家', '钢铁巨人', '飞向太空', '霹雳五号', '星际宝贝史迪奇', '红线', '落叶', '星际5555：异星梦系统秘传', '赛文奥特曼 EVOLUTION EPISODE：2 完美的世界', '复仇者联盟4：终局之战', '源代码', '火星救援', '彗星来的那一夜', '变脸', '蝙蝠侠：侠影之谜', '这个男人来自地球', '月球', '暖暖内含光', '银翼杀手', '守望者', '回到未来2', 'K星异客', '回到未来3', '阿基拉', '神秘博士：瑞芙·桑恩的丈夫们', '神秘博士：最后的圣诞', '她', '星球大战', '黑洞频率', '星球大战前传3：西斯的复仇', '超时空接触', '异次元骇客', '星球大战2：帝国反击战', '星球大战3：绝地归来', '星际宝贝', '他人之颜', '星际宝贝：终极任务', '飞出个未来大电影2：万背之兽', '蝙蝠侠超人大电影-最佳搭档', '如此美好的一天', '绿行星', '假面骑士ZO', '飞天小女警Z', '钢铁侠', '后天', '变形金刚', '银翼杀手2049', '第九区', '金刚狼3：殊死一战', '最终幻想7：圣子降临']

In [ ]: