Python数据预处理从入门到入葬

文本数据预处理可以分为读取数据源、文本分词、文本清洗以及数据可视化展示
接下来会使用代码实现完整的预处理流程

1.文本分词

Jieba库

Jieba是强大的Python中文分词库，主要功能是做中文分词，可以进行简单分词、并行分词、命令行分词

```
import jieba
```

补充-对比两种分词函数

cut()与lcut()方法

i) .cut生成的是一个生成器，generator，可以通过for循环来取里面的每一个词，通过print输出得到的是封装的数据
ii) .lcut 直接生成的就是一个list，通过print输出得到的是一个列表

```
# 参数传入 需要分词的字符串 是否采用全模式
seg_list = jieba.cut("中国上海是一座美丽的国际性大都市", cut_all=True)
```

```
list(seg_list)
```

```
['中国', '上海', '是', '一座', '美丽', '的', '国际', '国际性', '大都', '大都市', '都市']
```

```
# 对比是否采用全模式的区别
seg_list = jieba.cut("中国上海是一座美丽的国际性大都市", cut_all=False)
list(seg_list)
```

```
['中国', '上海', '是', '一座', '美丽', '的', '国际性', '大都市']
```

添加自定义词典

自定义的词汇的方法： jieba.load_userdict(file_name) 参数是文本文件，txt、csv都可以。
自定义词典文件的词汇格式是一个词占一行，每一行分三部分：词语、词频（可省略）、词性（可省略），用空格隔开，顺序不可颠倒

```
蒂法
最终幻想7
爱丽丝
米德加
萨菲罗斯
神罗战士
第五街区
赛特拉
约定之地
反神羅组织
魔晄
破坏剑
尼布尔海姆
杰诺瓦
重制版
扎克斯
```

```
# 载入词典
jieba.load_userdict("自定义分词.txt")
seg_list = jieba.cut("游戏最终幻想7，男主角为克劳德，游戏是双女主，分别是蒂法和爱丽丝。")
"/ ".join(seg_list)
```

```
'游戏/ 最终幻想7/ ，/ 男主角/ 为/ 克劳德/ ，/ 游戏/ 是/ 双/ 女主/ ，/ 分别/ 是/ 蒂法/ 和/ 爱丽丝/ 。'
```

修改jieba的正则表达式

由于jieba在分词时会把它们空格左右的词语进行分词
这种方式有时并不符合我们的需求。我们可以通过改jieba包init.py中几个正则表达式来解决这个问题

```
import re

jieba.re_han_default = re.compile("([\u4E00-\u9FD5a-zA-Z0-9+#&\._% -]+)", re.U)
```

2.词频统计

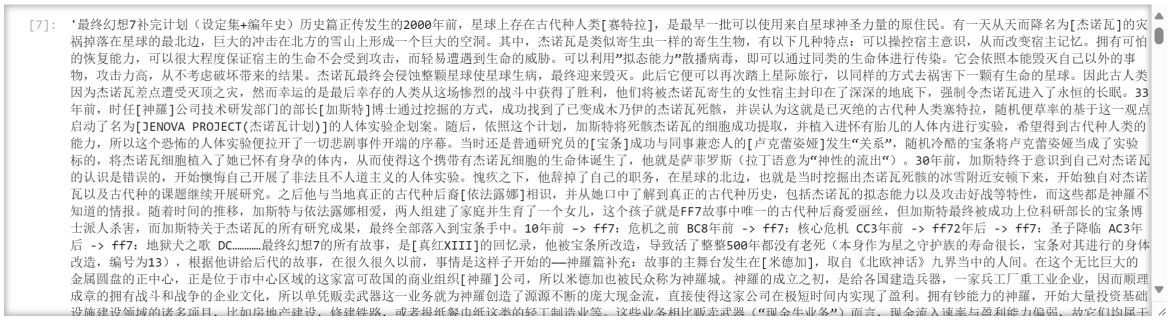
任务1.从本地word文档爬取正文内容存储为Python中的字符串
任务2.进行文本分词
任务3.将分词后的词段进行词频统计

docx库

使用Document(docx文件文件名)来读取word文档。调用.paragraphs()方法可以读取每一段内容，但该内容是封装的，还需要用.text()方法才能显示出文本内容

```
from docx import Document

docx = Document("FF7补完计划.docx")
# 将docx文档中每个段落的内容拼接到字符串中
contents = "".join(para.text for para in docx.paragraphs)
contents
```

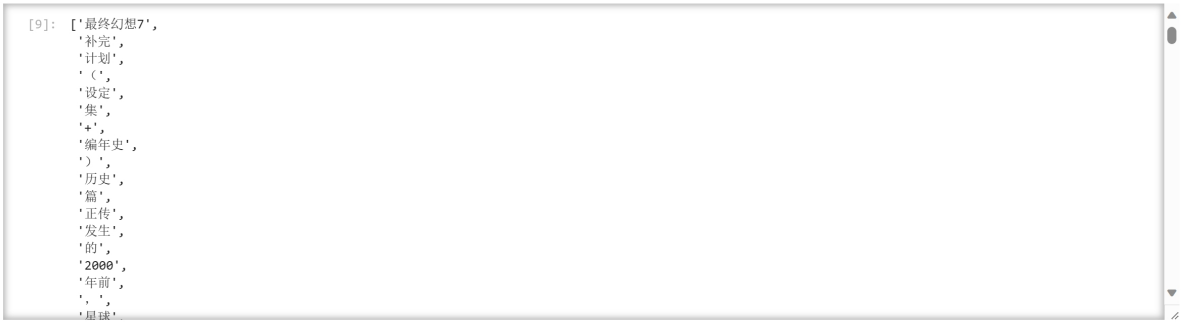


进行分词并储存为列表形式

```
seg_list = jieba.lcut(contents)
```

通过strip方法将字符串中的转义字符和空格符剔除

```
words = []
for word in seg_list:
    if word.strip() != "":
        words.append(word.strip())
words
```



collections库

使用collections库中的Counter()模块，调用most_common()实现词频统计

```
from collections import Counter

top_10 = Counter(words).most_common(10)  # 筛选分词后词频前10名的数据
print(*top_10)
```

```
('的', 1078) (', ', 1035) ('。', 500) ('了', 415) ('在', 169) ('扎克斯', 165)
('是', 139) ('他', 137) ('神羅', 112) ('[', 103)
```

不难发现，这些分词大部分都是我们不希望得到的词语。所以还需要做一步停用词的过滤

3.停用词过滤

下载停用词库

停用词一般储存在txt文件中，每一行代表一个停用词
首先在网上下载了中文停用词库、四川大学机器智能实验室停用词库、哈工大停用词表、百度停用词表
并把它们整理为一个新的txt文件

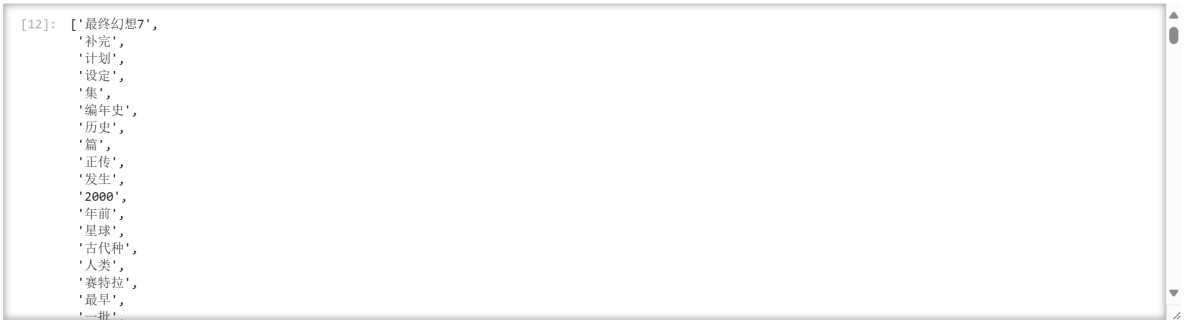
```
“
”
、
。
《
》
一
一些
一何
一切
一则
一方面
一旦
一来
一样
一般
一转眼
万一
上
上下
```

考虑到这四个停用词表可能会有重复的停用词
因此还需要定义一个集合用来去重

```
# 加载停用词库
with open("停用词库.txt", encoding="utf-8") as f:
    contents = f.readlines()
    stop_words = set()
    for word in contents:
        word = word.replace("\n", "")  # 去掉读取每一行数据的换行符
        stop_words.add(word)
```

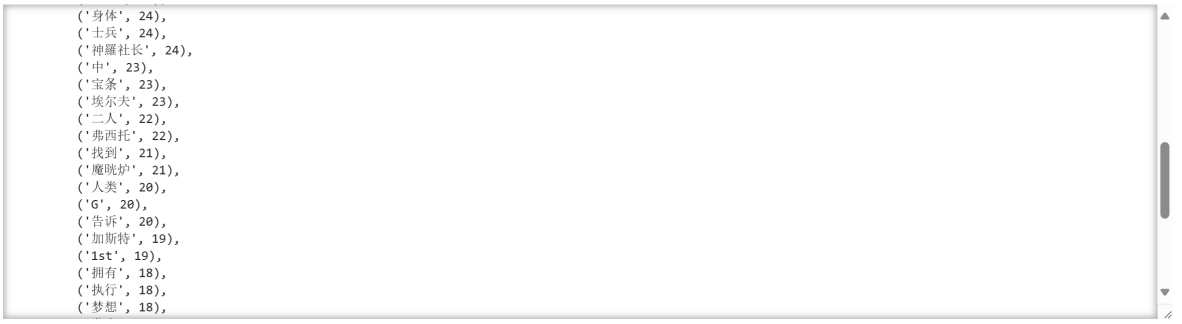
```
# 读取完停用词库后，去除停用词

data = []
jieba.load_userdict("自定义分词.txt")
for word in words:
    for word in jieba.lcut(word):
        if word not in stop_words:
            data.append(word)
data
```




```
# 再次查看排名前60的词条

top_60 = Counter(data).most_common(60)
top_60
```



添加自定义停用词

依旧有一些我们不想要的词条，需要进行二次清洗
即定义一个自定义停用词库，把那些无效的词条放进去

```
最终
便
回到
发生
S
身体
部门
中
里
年
…
...
均
没
没想到
年前
新
方式
```

```
with open("自定义停用词.txt", encoding="utf-8") as f:
    contents = f.readlines()
    stop_words = set()
    for content in contents:
        content = content.replace("\n", "")
        stop_words.add(content)
```

```
filtered_data = []
for word in data:
    if word not in stop_words:
        filtered_data.append(word)
# 查看排名前60的词条
top_60 = Counter(filtered_data).most_common(60)
top_60
```

```
[14]: [('扎克斯', 165),
       ('神罐', 112),
       ('萨菲罗斯', 84),
       ('杰内西斯', 82),
       ('塔克斯', 74),
       ('安吉尔', 73),
       ('杰诺瓦', 66),
       ('克劳德', 63),
       ('特种兵', 54),
       ('雪崩', 53),
       ('细胞', 46),
       ('维尔多', 41),
       ('魔晄', 39),
       ('爱丽丝', 35),
       ('杰诺瓦细胞', 33),
       ('魔石', 33),
       ('公司', 32),
       ('赫兰德', 28),
```

完成数据清理后，就可以实现文本数据的可视化了。这里使用的第三方库是Pyecharts
具体的模块和方法参考官网：
https://gallery.pyecharts.org/#/README
https://pyecharts.org/#/zh-cn/intro

4.绘制词频柱状图

制作一个柱状图查看词频排名前30的词条

Pyecharts库

需要用到Pyecharts库中的bar模块
options模块用于修改全局配置

要想在Jupyter_lab中显示出图表，还需要在顶部声明 Notebook 类型，必须在引入 pyecharts.charts 等模块前声明

```
from pyecharts.globals import CurrentConfig, NotebookType
CurrentConfig.NOTEBOOK_TYPE = NotebookType.JUPYTER_LAB
```

```
from pyecharts import options as opts
from pyecharts.charts import Bar
from pyecharts.globals import ThemeType

# 定义两个列表分别代表x轴和y轴数据
x_axis, y_axis = [], []
# 只展示排名前30的分词
count = 1
for part_tuple in top_60:
    if count == 30:
        break
    x_axis.append(part_tuple[0])
```

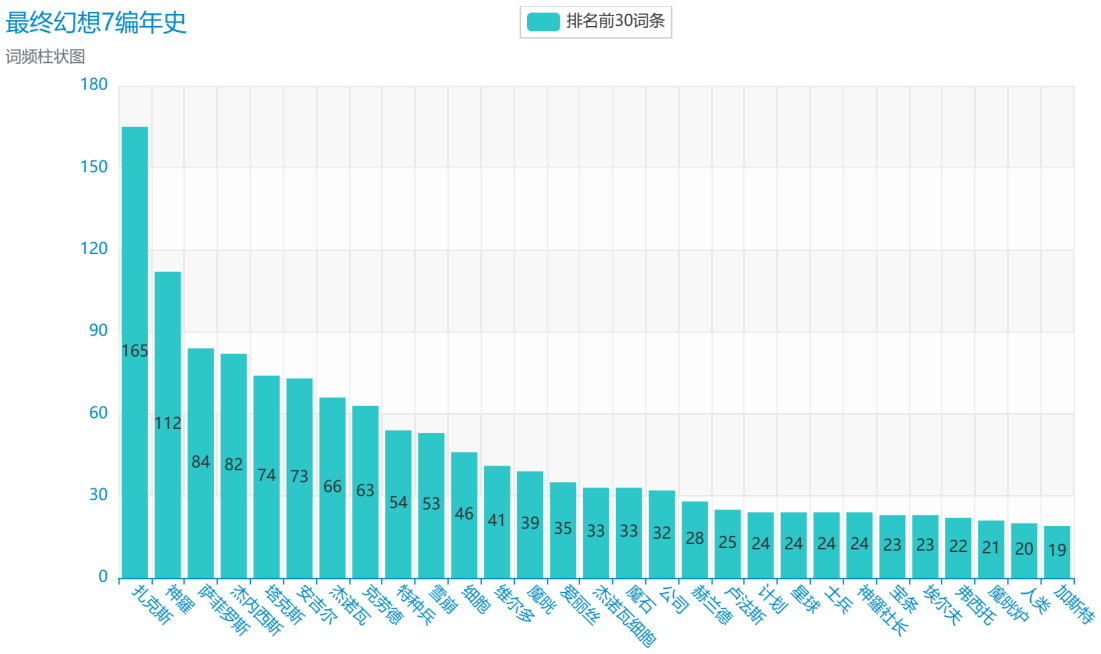
```
    y_axis.append(part_tuple[1])
    count += 1

bar = (
    Bar(init_opts=opts.InitOpts(theme=ThemeType.MACARONS))
    .add_xaxis(x_axis)
    .add_yaxis("排名前30词条", y_axis)
    .set_global_opts(
        # 防止x轴数据名太长，使其向下旋转45度
        xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=-45)),
        title_opts=opts.TitleOpts(title="最终幻想7编年史", subtitle="词频柱状图"),
    )
)
# 在第一次渲染时加载javascript文件
bar.load_javascript()
```

```
<pyecharts.render.display.Javascript at 0x1ee11d2f6d0>
```

注意：在.load_javascript()和.render_notebook两种方法要在不同的cell中调用，否则图片还是不会显示出来。

```
bar.render_notebook()
```




```
# 保存到本地
bar.render("词频图.html")
```

5.绘制词云图

需要用到pyecharts中的wordcloud模块

```
# 使用pyecharts实现词频统计可视化
import pyecharts.options as opts
from pyecharts.charts import WordCloud

cloud = (
    WordCloud()
    .add(series_name="词云图", data_pair=top_60, word_size_range=[10, 60],
shape="circle")
    .set_global_opts(
        title_opts=opts.TitleOpts(
            title="最终幻想7编年史-词云图",
title_textstyle_opts=opts.TextStyleOpts(font_size=30)
        ),
        tooltip_opts=opts.TooltipOpts(is_show=True),
    )
)
```

```
cloud.load_javascript()
```

```
<pyecharts.render.display.Javascript at 0x1ee17bc9120>
```

```
cloud.render_notebook()
```

最终幻想7编年史-词云图


```
# 保存到本地
cloud.render("词云图.html")
```

完成!