

目 录

第一章	需求分析.....	2
第二章	概要设计.....	3
第三章	详细设计.....	5
3.1	界面设计.....	5
3.2	数据库设计.....	7
3.3	关键技术.....	8
第四章	测试报告.....	12
4.1	系统测试的主要内容.....	12
4.2	系统模块测试.....	12
4.3	系统测试.....	13
4.4	技术指标.....	13
第五章	安装及使用.....	14
5.1	安装环境要求.....	14
5.2	Python3.7 安装.....	14
5.3	启动服务.....	15
5.4	安装本地数据库.....	16
第六章	项目总结.....	21

第一章 需求分析

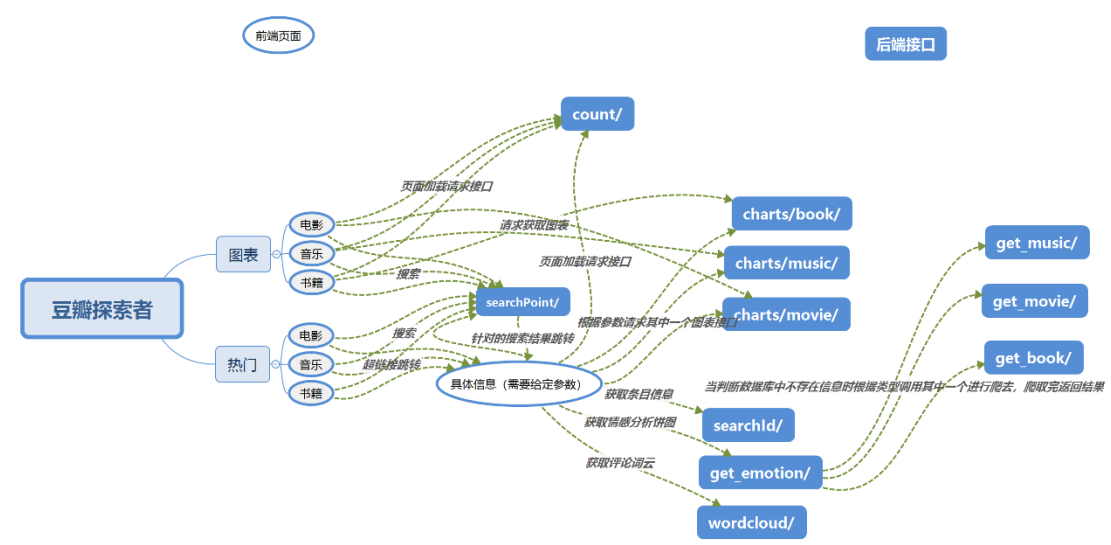
现如今阅读、观影、听歌已成为人们日常生活的一部分，每个人手机里或多或少的都有与这些相关的软件。在每一次的欣赏、聆听的背后，都隐藏着数据的奥秘。比如一部电影每个评分数量的多少，反映了大众对于它的直接评价。每年音乐的发行量，反映了彼时音乐创作人的创作量以及音乐市场的火爆程度。

基于以上认识，我们的项目旨在分析与电影、图书、音乐有关的数据去了解一部电影、一首音乐的市场反响，亦或是对多数电影、音乐等综合分析其背后整个市场的情况。因为需要大量的数据集，由此我们便想到了豆瓣这个平台。

我们这个平台主要是爬取豆瓣平台的信息，通过对数据的爬取、存储、读取、处理与分析、可视化等一系列操作，对数据进行了多维度的分析和展示，同时选取了几个热门条目作为例子，进行单个分析，从中得到了许多有效的信息，这些信息一方面可以帮助大众更好的去了解电影、图书、音乐，另一方面也能够为投资者以及从业者带来便利，更好的为他们展示哪些是符合当前市场的。很多人在学习中也会进行相关的案例分析，但本作品与其他竞品有以下区别：

	本作品	其他竞品
数据集的获取	本作品的数据集均由团队进行爬取	其他竞品中的数据集多来自于教学团队或他人给予
数据集的数量以及质量	本作品的数据集来自豆瓣相对于学习案例的数据集更加丰富且更有实际意义	其他竞品中的数据集相对单一且数据量小
数据集的分析	本作品对数据的分析更加完整，更加多元化，能够对数据进行系统的分析	其他竞品中的数据分析角度较单一
数据分析几个的过程	本作品集成了几个步骤为一体从数据获取直至数据可视化自动完成	其他竞品中一般为分步骤完成

第二章 概要设计



相关调用关系请参考上图，下面为各个文件的介绍。

前端有如下页面：

三个图表页面：index.html（电影）、chart-2.html（音乐）、chart-3.html（书籍）

三个热门页面：movie.html（电影）、book.html（书籍）、music.html（音乐）

一个具体条目信息介绍页面：content.html（具体信息）

后端有如下接口：

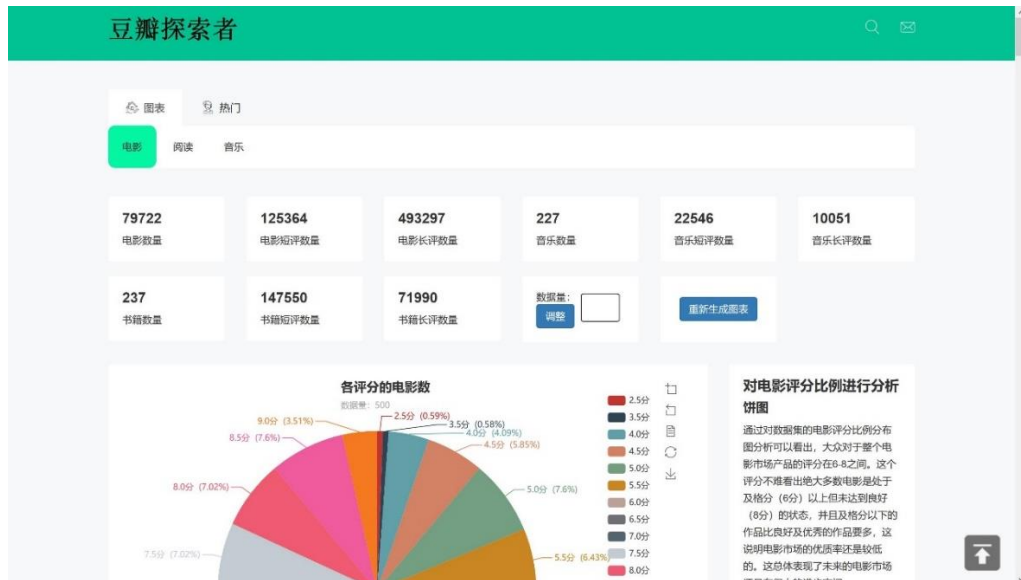
链接	对应文件	调用函数	用途
count/	get/count.py	main()	获取数据库内各个集合的数量
get/	collection/get.py	main()	采集数据合集，需给定 id、type
get_music/	collection/get_music.py	main()	采集音乐数据，需给定 id
get_movie/	collection/get_movie.py	main()	采集电影数据，需给定 id
get_book/	collection/get_book.py	main()	采集书籍数据，需给定 id
wordcloud/	get/wordcloud.py	get_wordcloud()	生成评论词云，需给定 id、type，非必需 no-cache 默认为 0 使用 redis 缓存，为 1 时不使用 redis 缓存
searchPoint/	get/search2.py	get_search()	显示最多 20 条信息，即一次爬取，需给定 q，非必需 cat、start
searchId/	get/search3.py	get_search()	获取作品信息，即详情页，需给定 id、type
get_emotion	get/get_emot	ChartView.	生成情感分析饼图，需给定 id、type、

n/	ion.py	as_view()	nochache, 非必需 n 为 review 个数, 默认 1 个
charts/movie/	charts/movie.py	ChartView.as_view()	生成电影分析图表, 需给定 id、type、typ, typ 为图表类型, 非必需 num 分析的数据量, 默认为 5000, nochache 默认为 0 使用 redis 缓存, 为 1 时不使用 redis 缓存
charts/book/	charts/book.py	ChartView.as_view()	生成书籍分析图表, 需给定 id、type、typ, typ 为图表类型, 非必需 num 分析的数据量, 默认为 5000, nochache 默认为 0 使用 redis 缓存, 为 1 时不使用 redis 缓存
charts/music/	charts/music.py	ChartView.as_view()	生成音乐分析图表, 需给定 id、type、typ, typ 为图表类型, 非必需 num 分析的数据量, 默认为 5000, nochache 默认为 0 使用 redis 缓存, 为 1 时不使用 redis 缓存

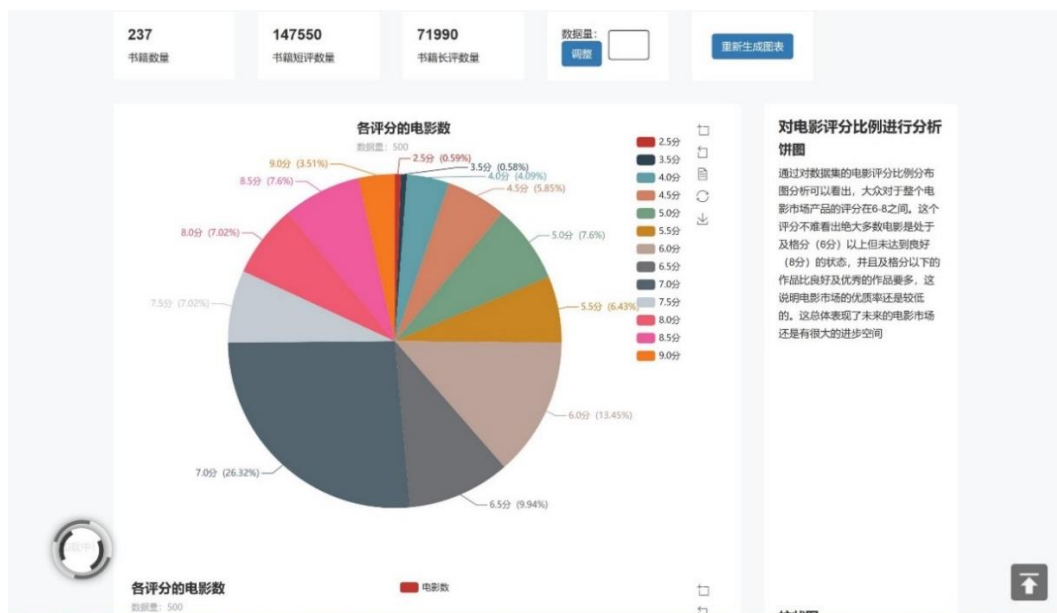
第三章 详细设计

3.1 界面设计

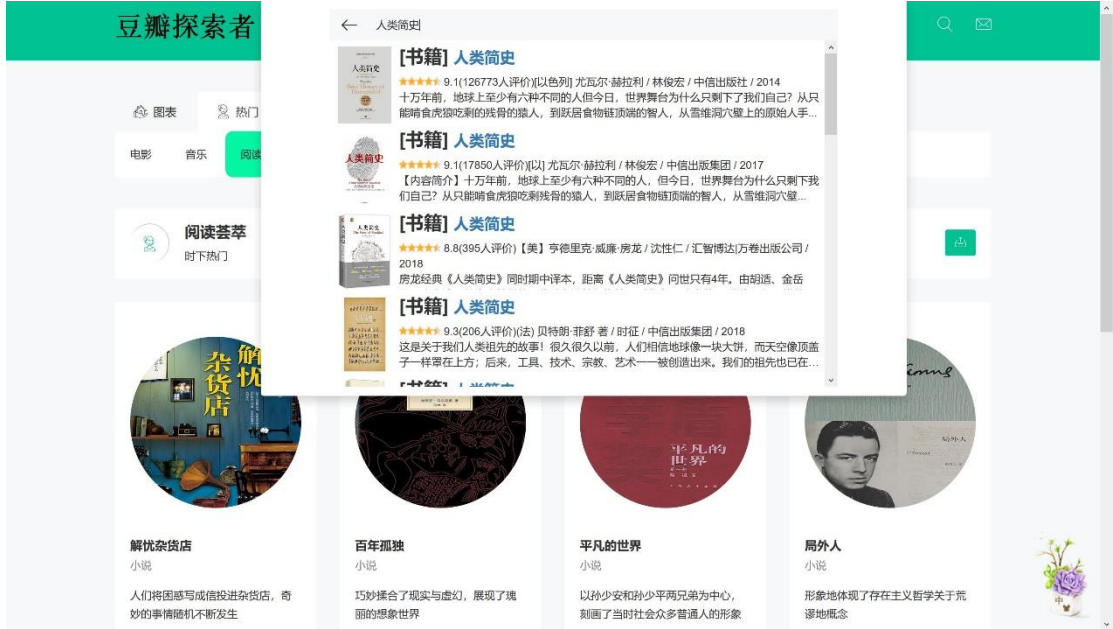
顶部是我们的作品名以及搜索框，在下面就是各个页面切换的超链接，中间显示的就是我们当前 MongoDB 数据库各个集合里所存储的数据量，每次加载都是实时统计。除了数据之外，我们还加了两个按钮，分别用来调整图表分析的数据量以及重新生成图表，数据量默认取的是 500 条且为随机取出。



我们以电影图表最上面的饼图为例，将鼠标放置图表上，也会有具体的信息，右边倒数第二个 Label 可以控制饼图是否显示该数据，最右侧的工具栏也很丰富，缩放工具可以在柱状图、散点图、折线图中使用，用查看具体的范围内的数据，同时还有数据视图功能和图表下载功能。左下角有页面加载状况，右下角有置顶功能，这两个功能在有图表的页面都会有。



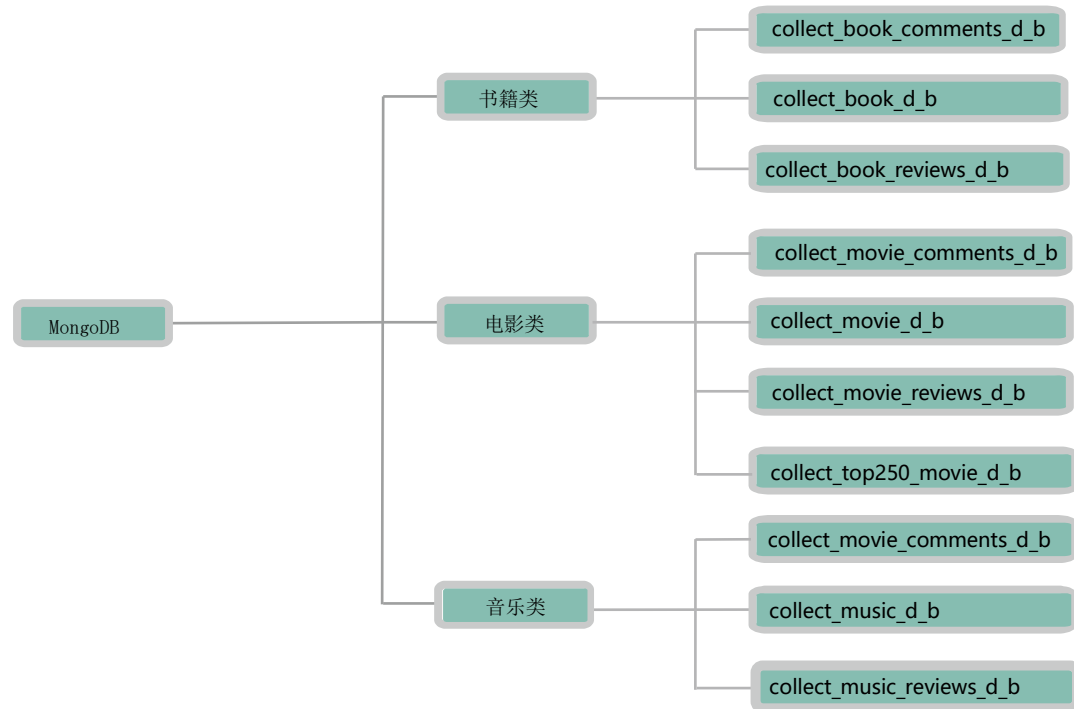
这是电影的热门页面，这类页面由我们选出的 6 个作为代表展示（可调用豆瓣热门电影接口获取替换），点击单个条目后可跳转到 content.html 具体内容介绍页面。顶上放是点击搜索按钮后的效果。（每个结果都可跳转到详细内容页面，若跳转过去的是数据库不存在的条目，那后台将进行自动进行爬取、数据预处理、数据存取等工作，爬去完成后再读取、处理和显示图表，爬取的时长由条目评论的数量决定。）



这是详细页面中的上部分内容，主要为该条目的文字详情信息介绍，在其下面还有五张针对于该条目所生成的图表。



3.2 数据库设计



CollectMovieDB			CollectBookDB			CollectMusicDB		
字段名	类型	备注	字段名	类型	备注	字段名	类型	备注
movie_id	IntField	唯一标识 (索引、唯一性约束)	book_id	IntField	唯一标识 (索引、唯一性约束)	music_id	IntField	唯一标识 (索引、唯一性约束)
original_title	StringField	原始标题	title	StringField	中文标题	title	StringField	中文标题
title	StringField	中文标题	author	StringField	作者	rating	StringField	评分
rating	StringField	评分	origin_title	StringField	原始标题	ratings_count	IntField	评分数
ratings_count	IntField	评分数	pubdate	StringField	出版时间	pubdate	StringField	发行时间
pubdate	StringField	上映日期	images	StringField	封面图片	images	StringField	封面图片
pubdates	StringField	上映日期2	summary	StringField	书简介	summary	StringField	简介
year	IntField	年份	rating	IntField	评分	publisher	StringField	出版者
countries	StringField	制片国家/地区	rating_count	IntField	评分人数	singer	StringField	歌手
mainland_pubdate	StringField	主要上映日期	rating_star	StringField	评价星级	media	StringField	介质
aka	StringField	又名	comments_count	IntField	短评人数	version	StringField	专辑类型
tags	StringField	标签	reviews_count	IntField	长评人数	tags	StringField	标签
durations	StringField	时长	reading_count	IntField	在读人数	rating_star	StringField	评价星级
genres	StringField	类型	readed_count	IntField	读过人数	comments_count	IntField	短评人数
videos	StringField	短视频	wish_count	IntField	想读人数	reviews_count	IntField	长评人数
wish_count	StringField	想看	publisher	StringField	出版商	reading_count	IntField	在读人数
reviews_count	IntField	短评数	isbn10	StringField	ISBN10	readed_count	IntField	读过人数
comments_count	IntField	评论数	isbn13	StringField	ISBN13	wish_count	IntField	想读人数
collect_count	IntField	收藏	author_intro	StringField	作者简介	record_time	StringField	记录时间
images	StringField	封面图片	ebook_price	StringField	电子书价格			
photos	StringField	照片	price	StringField	电子书价格			
languages	StringField	语言	series	StringField	系列			
writers	StringField	作者	pages	IntField	页数			
actor	StringField	演员	translator	StringField	翻译人			
summary	StringField	简介	binding	StringField	出版类型			
directors	StringField	导演	tags	StringField	标签			
record_time	StringField	记录时间	record_time	StringField	记录时间			

CollectMovieCommentsDB			CollectBookCommentsDB			CollectMusicCommentsDB			CollectTop250MovieDB		
字段名	类型	备注	字段名	类型	备注	字段名	类型	备注	字段名	类型	备注
comments_id	IntField	短评唯一标识（索引、唯一性约束）	comments_id	IntField	短评唯一标识（索引、唯一性约束）	comments_id	IntField	短评唯一标识（索引、唯一性约束）	movie_id	IntField	唯一标识（索引、唯一性约束）
comments_movid_id	IntField	电影唯一标识（索引）	comments_book_id	IntField	书籍唯一标识（索引）	comments_music_id	IntField	音乐唯一标识（索引）	movie_title	StringField	中文标题
comments_rating	IntField	评分	comments_rating	IntField	评分	comments_rating	IntField	评分	movie_original_title	StringField	原始标题
comments_useful_count	IntField	认为有用的评论数	comments_useful_count	IntField	认为有用的评论数	comments_useful_count	IntField	认为有用的评论数	movie_rating	StringField	评分
comments_content	StringField	评论内容	comments_content	StringField	评论内容	comments_content	StringField	评论内容	movie_year	IntField	年份
comments_author_uid	StringField	评论作者uid	comments_author_uid	StringField	评论作者uid	comments_author_uid	StringField	评论作者uid	movie_pubdates	StringField	上映日期
comments_author_id	IntField	评论作者id	comments_author_id	IntField	评论作者id	comments_author_id	IntField	评论作者id	movie_directors	StringField	导演
comments_author_name	StringField	评论作者昵称	comments_author_name	StringField	评论作者昵称	comments_author_name	StringField	评论作者昵称	movie_genres	StringField	类型
comments_time	StringField	评论时间	comments_time	StringField	评论时间	comments_time	StringField	评论时间	movie_actor	StringField	演员
record_time	StringField	记录时间	record_time	StringField	记录时间	record_time	StringField	记录时间	movie_durations	StringField	时长
									movie_collect_count	IntField	收藏
									movie_mainland_pubdate	StringField	主要上映日期
									movie_images	StringField	封面图片
									record_time	StringField	记录时间

CollectMovieReviewsDB			CollectBookReviewsDB			CollectMusicReviewsDB		
字段名	类型	备注	字段名	类型	备注	字段名	类型	备注
reviews_id	IntField	长评唯一标识（索引、唯一性约束）	reviews_id	IntField	长评唯一标识（索引、唯一性约束）	reviews_id	IntField	长评唯一标识（索引、唯一性约束）
reviews_movid_id	IntField	电影唯一标识（索引）	reviews_book_id	IntField	书籍唯一标识（索引）	reviews_music_id	IntField	音乐唯一标识（索引）
reviews_rating	IntField	评分	reviews_rating	IntField	评分	reviews_rating	IntField	评分
reviews_useful_count	IntField	认为有用的评论数	reviews_rating_text	StringField	评论文字描述	reviews_rating_text	StringField	评论文字描述
reviews_content	StringField	评论内容	reviews_useful_count	IntField	认为有用的评论数	reviews_useful_count	IntField	认为有用的评论数
reviews_author_uid	StringField	评论作者uid	reviews_content	StringField	评论内容	reviews_content	StringField	评论内容
reviews_author_id	IntField	评论作者id	reviews_author_uid	StringField	评论作者uid	reviews_author_id	IntField	评论作者id
reviews_author_name	StringField	评论作者昵称	reviews_author_id	IntField	评论作者id	reviews_author_name	StringField	评论作者昵称
reviews_time	StringField	评论时间	reviews_author_name	StringField	评论作者昵称	reviews_time	StringField	评论时间
reviews_title	StringField	评论标题	reviews_time	StringField	评论时间	reviews_title	StringField	评论标题
reviews_updated	StringField	评论更新时间	reviews_title	StringField	评论标题	reviews_updated	StringField	评论更新时间
reviews_share_url	StringField	评论url	reviews_useless_count	IntField	认为是无用的评论数	reviews_useless_count	IntField	认为是无用的评论数
reviews_summary	StringField	评论简述内容	reviews_comments_count	IntField	评论数	reviews_comments_count	IntField	评论数
reviews_useless_count	IntField	认为是无用的评论数	record_time	StringField	记录时间	record_time	StringField	记录时间
reviews_comments_count	IntField	评论数						
record_time	StringField	记录时间						

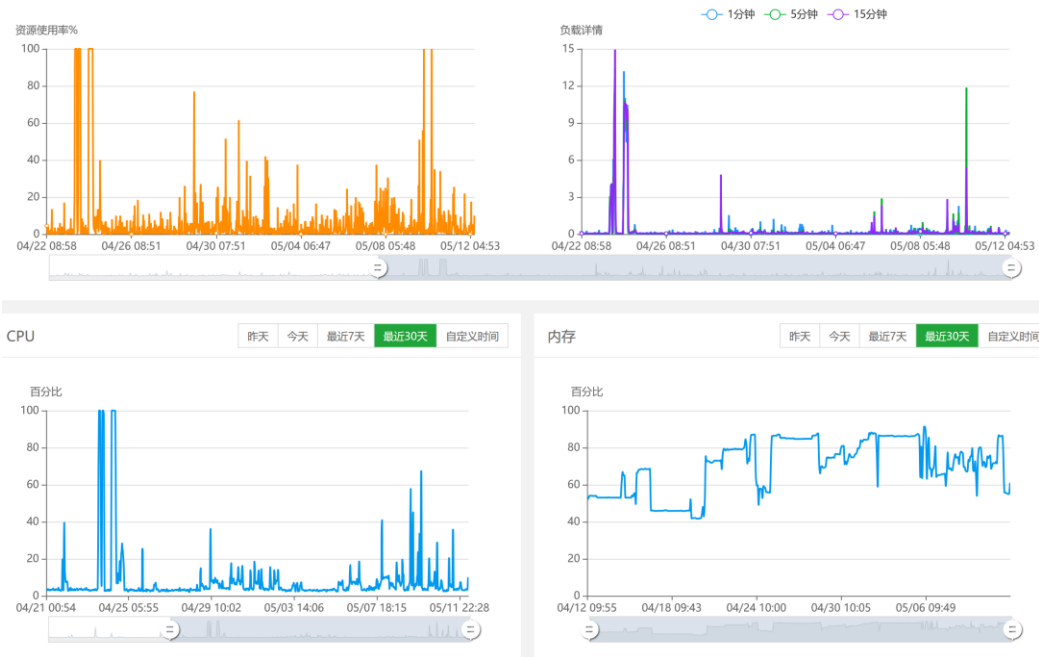
3.3 关键技术

1) 数据爬取：（重点、难点）

在数据爬取时我们要不断地完善修改代码，因为代码在初始编写时是针对单个页面来写的，能适应单个页面不代表能适应所有类型的页面，因为有些页面即使类别相同，所展示的内容不同有多有少，就会导致爬取的时候代码出现问题，因此即使对单个页面的爬取代码也要反复修改多次，来不断能完善。在爬取的代码中，我们同时也为了防止被豆瓣的反爬机制所屏蔽而加入了相关代码，但也相应的造成了爬取时间的延长。

2) 数据存储查询优化：（重点、难点）

为了证明数据存取优化的重要性，下面我放了一张我们在服务器上爬取的几个时间节点的资源消耗图。最左边的一段峰值点时段就是使用未优化过的查询语句所造成的。在完成语句的优化后，后几次爬取就未达到如此之高。



3) 数据处理：（重点、难点）

因为不知道数据库里所存储的是否符合所要数据的规范，任何类型的数据都会出现，以及是否有对异常值等，所以在提取出数据后，我们还需要对其再做相关的处理，如对数据缺失、极端值、数据格式不统一等问题进行处理。（下图就是其中的一个数据处理的例子）

```
#评论时间与评分的关系（单个喜好关系）
result = CollectMovieCommentsDB.objects.aggregate([
    {'$match': {'comments_movid_id': 3541415}},
    {'$sample': {'size': 50}},
    {'$project': {'comments_rating': 1, 'comments_time': 1}},
    {'$sort': {'comments_time': -1}}
]) #从数据库中读取所需数据

rating = list()
times = list()
quarter = list()
for i in result:
    # print(i['comments_rating'])
    rating.append(i['comments_rating'])
    times.append(i['comments_time'][:10])
    year = i['comments_time'][:4]
    month = int(i['comments_time'][5:7])
    # 对异常数据进行处理
    if month >= 1 and month <= 3:
        quarter.append(year+'-'+'1')
    elif month >= 4 and month <= 6:
        quarter.append(year+'-'+'2')
    elif month >= 7 and month <= 9:
        quarter.append(year+'-'+'3')
    elif month >= 10 and month <= 12:
        quarter.append(year+'-'+'4')
# print(rating)
# print(times)
# print(quarter)
index = np.arange(len(rating)) #使用NumPy生成数组
data = pd.DataFrame(data={'评分':rating,'季度':quarter},index=index) # 使用Pandas将数据集成DataFrame型方便于转换处理
data = data.groupby(by="季度",as_index=False).mean() #对数据集进行筛选聚合
# for i in data:
#     print(i)
# data
# data.keys
rt = list(map(float, np.around(data['评分'].values, 2))) # 对数据类型进行转换，生成指定类型数据集输出
qu = data['季度'].values
# print(rt)
```

4) 图表展示和分析：（重点、难点）

选取完要分析的数据关系后，就该选择合适的图表来进行输出表示，确定好图表后，还要考虑图表的展现形式，包括图表中的工具配置等，同时还要对图表进行分析。（下图就是其中的一个数据可视化的例子）

```
l11 = (
    Line()
    .add_xaxis(xaxis_data=qu) # 使用前面输出的数据
    .add_yaxis(
        series_name="评分",
        y_axis=rt, # 使用前面输出的数据
        markpoint_opts=opts.MarkPointOpts( # 创建标记数据项点（最大值最小值）
            data=[
                opts.MarkPointItem(type_="max", name="最大值"),
                opts.MarkPointItem(type_="min", name="最小值"),
            ]
        ),
        markline_opts=opts.MarkLineOpts( # 创建标记数据项线（平均值）
            data=[opts.MarkLineItem(type_="average", name="平均值")]
        ),
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="评论时间季度与评分的关系"), # 设置标题
        tooltip_opts=opts.TooltipOpts(trigger="axis", is_show=True), # 设置提示显示
        axispointer_opts=opts.AxisPointerOpts(
            is_show=True, link=[{"xAxisIndex": "all"}]
        ),
        datazoom_opts=[opts.DataZoomOpts(), opts.DataZoomOpts(type_="inside")], # 配置区域缩放配置项
        toolbox_opts=opts.ToolboxOpts(is_show=True, # 设置工具栏
            orient="vertical",
            pos_left="90%",
            feature={
                "dataZoom": [{"yAxisIndex": "none"}],
                "dataView": {},
                "magicType": {
                    "show": True,
                    "title": "切换",
                    "type": ['line', 'bar'], # 启用的动态类型，包括'line'（切换为折线图），'bar'（切换为柱状图）
                },
                "restore": {},
                "saveAsImage": {},
            },
        ),
        xaxis_opts=opts.AxisOpts(name="评论时间季度", type_="category", boundary_gap=False), # 设置横坐标轴
        yaxis_opts=opts.AxisOpts(name="评分", # 设置纵坐标轴
    )
)
l11.render_notebook()
```



5) Redis 缓存技术：（重点）

一开始在访问网站时，每次生成图表都要再次进行读数据、分析处理数据，这不仅大大增加了对服务器的性能压力，同时也加慢了网页的加载速度，由此，选择了 Redis，当图表第一次生成就让 Redis 存储其结果 24 小时，第二次在访问时就可以直接从 Redis 中获取结果，省去了再次进行读数据、分析处理数据的过程，很好的解决的上述的两个问题。

6) 前端图表的链式调用：（重点）

考虑到如果一加载页面，就一下子请求所有图表，那会对服务器一下子产生很大的压力，容易出错。所以我们想到了使用链式请求的方法，一个一个请求图表，且同时加上 loading 的动态效果，表明当前页面的图表加载状况。

```
function fetchChartAll(type, typ, num, j, nochange) {
    // typ = GetQueryValue("typ");
    var chart = echarts.init(document.getElementById(typ), 'white', {
        renderer: 'canvas'
    });
    ur = BASE_URL + "/charts/" + type + "?id=&typ=" + typ + "&num=" + num + "&nochange=" + nochange;
    if (typ == "get_emotion") {
        ur = BASE_URL + "/get_emotion/?id=&type=" + type + "&nochange=" + nochange;
    } else if (typ == "wordcloud") {
        ur = BASE_URL + "/wordcloud/?id=&type=" + type + "&nochange=" + nochange;
    }
    $.ajax({
        type: "GET",
        url: ur,
        dataType: 'json',
        success: function (result) {
            chart.setOption(result.data);
            j++;
            console.log(j);
            if (j < dt.length) {
                // if(j<2){
                fetchChartAll(dt[j][0], dt[j][1], num, j, nochange); //链式调用
            } else {
                $("#circle").fadeOut(800);
                $("#circle1").fadeOut(1200);
                $("#circletext").text("加载完成! ").fadeOut(1700);
            }
        },
        error: function (xhr, textStatus, errorThrown) {
            $("#circle").fadeOut(800);
            $("#circle1").fadeOut(1200);
            $("#circletext").text("加载完成! ").fadeOut(1700);
            alert("请求失败, 服务器端出错!");
            j++;
            if (j < dt.length) {
                // if(j<2){
                fetchChartAll(dt[j][0], dt[j][1], num, j, nochange); //链式调用
            } else {
                $("#circle").fadeOut(800);
                $("#circle1").fadeOut(1200);
                $("#circletext").text("加载完成! ").fadeOut(1700);
            }
            return
        }
    });
}
```

第四章 测试报告

4.1 系统测试的主要内容

为了确保测试的质量，系统多次进行了软件测试，主要包括代码审阅、模块测试、功能测试、安全性测试等内容。

代码审阅：在系统实现完成以后，应先对代码的规范性进行检测，并且测试代码的语法逻辑问题，保证系统能正常运行。

模块测试：对系统的主页模块，页面跳转，搜索框，图表等模块进行测试，确保其工作正常。

主要测试过程：模拟用户访问页面，不断点击跳转页面；模拟用户查找特定案例，进行搜索框搜索；模拟用户查看数据图表。

测试结果：页面能正常跳转、搜索功能可以正常使用。词云模块图表展示不正常，请求本地时正常。

修正过程：因为服务器处理数据能力有限，当前条件下在打开相关页面后修改代码可以达到展示效果。在条件成熟后会选择升级服务器。

4.2 系统模块测试

系统模块测试主要是对系统中各个功能模块进行详细的测试工作，发现问题并处理问题。测试工作是通过手动反复对系统进行操作，观察系统运行的结果，判断该功能模块是否达到应用要求。具体测试如下表所示：

表 5-1 系统模块测试表

测试内容	测试方法	预期结果	测试情况
主页测试	打开 index 主页面	打开成功	通过
页面跳转	在页面上点击所有具有链接功能的模块	所有页面都能成功跳转	通过
搜索框	在搜索框内输入想要查询的电影、音乐或图书	若查询内容存在数据集内或有与查找内容相近内容则显示相应选项，若没有匹配数据则不显示	通过
图表	将鼠标放在图标表上，并在具有缩放功能的图标上使用鼠标滚轮或点击图表内缩放功能	显示鼠标停放位置所对应的相应信息，三维立体图以及具有缩放功能的图表能正常缩放	通过

4.3 系统测试

模块测试的完成只是保证了模块的正常工作,无法保证整个整体工作是否能够正常的运行。所以在模块测试完成以后,要进行系统完整的用例测试,来验证系统是否运行正常。

本系统为基于 Window 10 平台,系统设计分为四部分。测试过程中首先测试主页面展示,确保主页面模块可以正确打开并展示相关内容。然后测试页面跳转模块,能保证页面之间能够互相跳转。然后测试搜索框模块,确保能够进行精确定位查询,展示用户所需要查找的内容。最后测试图表模块,确保图表能正常显示,以及图表内某一块内容能够展示其独有的信息。若图表具有缩放功能,测试其能够利用缩放功能对图表的整体或局部内容进行查看。如果四个模块均正常运行,说明该系统均已正常工作。

4.4 技术指标

通过测试发现,本系统因图表展示较多,运行速度相对较慢达到 6.5s。因为本系统数据来源为豆瓣,因此数据安全性较高。在系统初期阶段没有上线用户登录注册功能,因此没有账号安全性的威胁。系统扩展性较强,耦合性较低内聚性较强。本系统的部署非常简单,方便性极强。可用性很强。

第五章 安装及使用

5.1 安装环境要求

操作系统: Windows 10 或 Centos 7

应用软件环境: Python 3.6 及以上、Redis、MongoDB (后两个可根据需要搭建)

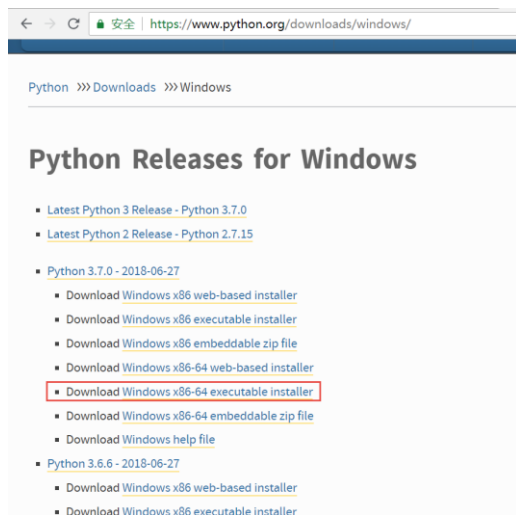
浏览器: Chrome 或 Firefox

5.2 Python3.7 安装

(注: 以下安装过程皆为 Windows 下的, 如需 Linux 安装教程, 请咨询相关服务人员)

去官网下载安装包, 下载链接: <https://www.python.org/downloads/windows/>

进入下载链接后选择对应的版本下载



①点击 customize installation。(自定义安装路径), 也可以选择 Install Now 选择默认路径安装

②勾选 add python 3.7 to PATH (这样可以避免安装后, 还要手动去配置 python 的环境变量)



后续操作默认即可, 安装完成, 点击 close 按钮

win+r, 录入 cmd, 进入如下窗口, 录入: python, 点击 enter, 出现如下内容, 则为安装成功。

```
命令提示符 - python
Microsoft Windows [版本 10.0.18363.778]
(c) 2019 Microsoft Corporation。保留所有权利。

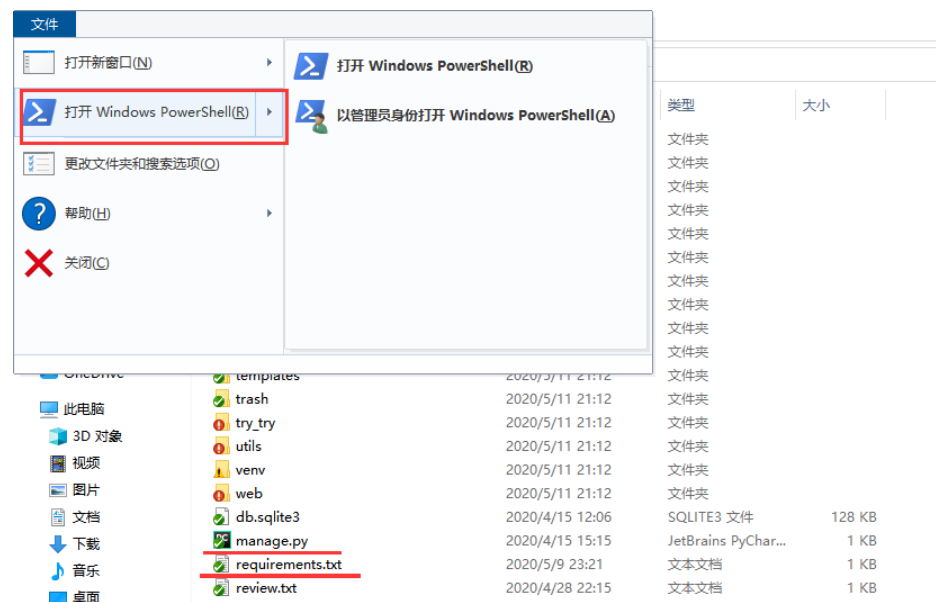
C:\Users\cy>python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

5.3 启动服务

打开 web 文件夹(Django 后端)点击左上角文件按钮, 点击打开 Windows PowerShell (R)



输入命令安装应用程序所需 python 模块

命令: `pip install --index https://mirrors.aliyun.com/pypi/simple/ -r requirements.txt`

```
PS E:\file\web> pip install --index https://mirrors.aliyun.com/pypi/simple/ -r requirements.txt
Looking in indexes: https://mirrors.aliyun.com/pypi/simple/
Collecting gensim==3.8.1
  Downloading https://mirrors.aliyun.com/pypi/packages/09/ed/b59a2edde05b7f5755ea68648487c150c7c742361e9c8733c6d4ca005020/gensim-3.8.1-cp37-cp37m-win_amd64.whl (24.2 MB)
    |#####| 24.2 MB 6.4 MB/s
Collecting matplotlib==3.1.2
  Downloading https://mirrors.aliyun.com/pypi/packages/dd/73/dc25ca27a9960539ef984921b0d42368445b856ae0861c3acba542b9a39c/matplotlib-3.1.2-cp37-cp37m-win_amd64.whl (9.1 MB)
    |#####| 9.1 MB 6.4 MB/s
Requirement already satisfied: requests==2.22.0 in e:\app\anaconda3\lib\site-packages (from -r requirements.txt (line 3)) (2.22.0)
Collecting snownlp==0.12.3
  Downloading https://mirrors.aliyun.com/pypi/packages/3d/b3/37567686662100d3bce62d3b0f2adec18ab4b9ff2b61abd7a61c39343c1
```

输入命令, 启动 Django 服务

命令: `python .\manage.py runserver`

```

PS E:\file\web> python .\manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
May 11, 2020 - 21:25:39
Django version 3.0.3, using settings 'web.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

打开 html 文件夹（前端网页文件），点击 index.html 打开主页面

此电脑 > 软件 (E:) > file > html

名称	修改日期	类型	大小
css	2020/5/11 21:27	文件夹	
fonts	2020/5/11 21:27	文件夹	
images	2020/5/11 21:27	文件夹	
img	2020/5/11 21:27	文件夹	
js	2020/5/11 21:27	文件夹	
book.html	2020/5/11 3:41	Chrome HTML D...	22 KB
chart-2.html	2020/5/11 3:19	Chrome HTML D...	35 KB
chart-3.html	2020/5/11 3:19	Chrome HTML D...	37 KB
content.html	2020/5/11 13:32	Chrome HTML D...	32 KB
index.html	2020/5/11 3:19	Chrome HTML D...	44 KB
movie.html	2020/5/11 3:41	Chrome HTML D...	22 KB
music.html	2020/5/11 3:41	Chrome HTML D...	22 KB

Web 里的是后端服务 (Django)，里面的配置连接的是我们的服务器端 MongoDB 和 Redis 数据库，如需想用自己的数据库，只需更改相应的数据库连接代码。

MongoDB: \web\web\settings.py 第 97 行的参数

Redis: \web\utils\redis_pool.py 第 2 行的参数

(PS: 因涉及数据库账号密码就不贴图了)

如有需求想用自己的数据库，请按照下面教程安装两个数据库，然后改上面两个文件的连接信息。

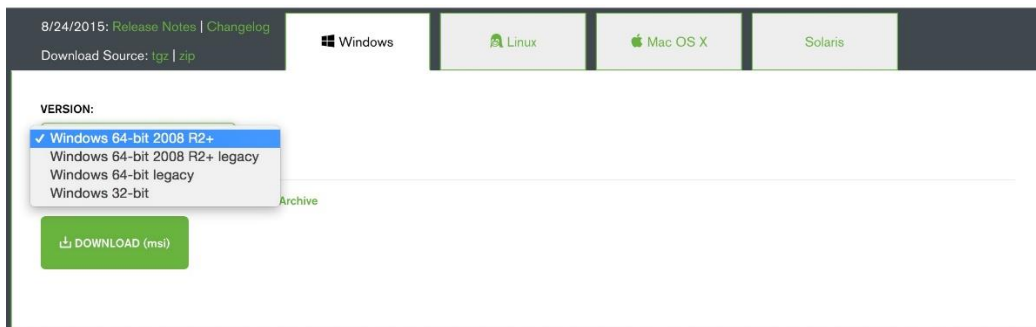
5.4 安装本地数据库

1) 安装 MongoDB:

MongoDB 下载:

MongoDB 提供了可用于 32 位和 64 位系统的预编译二进制包，你可以从 MongoDB 官网下载安装，MongoDB 预编译二进制包下载地址：<http://www.mongodb.org/downloads>

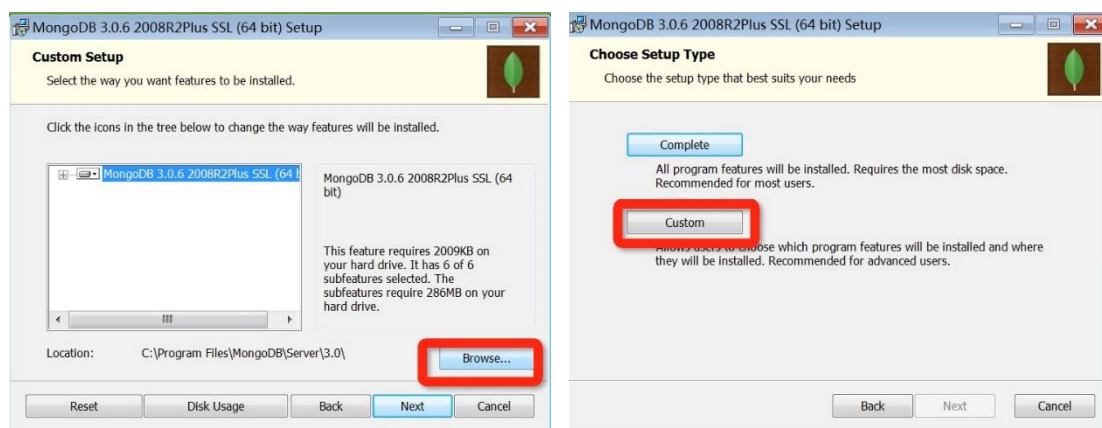
注意：在 MongoDB2.2 版本后已经不再支持 Windows XP 系统。



- **MongoDB for Windows 64-bit** 适合 64 位的 Windows Server 2008 R2, Windows 7, 及最新版本的 Window 系统。
- **MongoDB for Windows 32-bit** 适合 32 位的 Window 系统及最新的 Windows Vista。32 位系统上 MongoDB 的数据库最大为 2GB。
- **MongoDB for Windows 64-bit Legacy** 适合 64 位的 Windows Vista, Windows Server 2003, 及 Windows Server 2008 。

根据你的系统下载 32 位或 64 位的 .msi 文件，下载后双击该文件，按操作提示安装即可。

安装过程中，你可以通过点击 “Custom(自定义)” 按钮来设置你的安装目录。



一直点下一步，直到完成。

创建数据目录：

MongoDB 将数据目录存储在 db 目录下。但是这个数据目录不会主动创建，我们在安装完成后需要创建它。请注意，数据目录应该放在根目录下（如： C:\ 或者 D:\ 等 ）。

在本教程中，我们已经在 C 盘安装了 mongodb，现在让我们创建一个 data 的目录然后在 data 目录里创建 db 目录。

```
c:\>cd c:\  
c:\>mkdir data
```

```
c:\>cd data
c:\data>mkdir db
c:\data>cd db
c:\data\db>
```

命令行下运行 MongoDB 服务器：

为了从命令提示符下运行 MongoDB 服务器，你必须从 MongoDB 目录的 bin 目录中执行 mongod.exe 文件。

```
C:\mongodb\bin>mongod --dbpath c:\data\db
```

如果执行成功，会输出如下信息：

```
2015-09-25T15:54:09.212+0800 I CONTROL Hotfix KB2731284 or later update is not
installed, will zero-out data files
2015-09-25T15:54:09.229+0800 I JOURNAL [initandlisten] journal dir=c:\data\db\j
ournal
2015-09-25T15:54:09.237+0800 I JOURNAL [initandlisten] recover : no journal fil
es present, no recovery needed
2015-09-25T15:54:09.290+0800 I JOURNAL [durability] Durability thread started
2015-09-25T15:54:09.294+0800 I CONTROL [initandlisten] MongoDB starting : pid=2
488 port=27017 dbpath=c:\data\db 64-bit host=WIN-1VONBJOCE88
2015-09-25T15:54:09.296+0800 I CONTROL [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2015-09-25T15:54:09.298+0800 I CONTROL [initandlisten] db version v3.0.6
.....
```

创建配置文件：

创建一个配置文件。该文件必须设置 systemLog.path 参数，包括一些附加的配置选项更好。

例如，创建一个配置文件位于 C:\mongodb\mongod.cfg，其中指定 systemLog.path 和 storage.dbPath。具体配置内容如下：

```
systemLog:
  destination: file
  path: c:\data\log\mongod.log
storage:
  dbPath: c:\data\db
```

安装 MongoDB 服务

```
C:\mongodb\bin>mongod.exe --config "C:\mongodb\mongod.cfg" --install
```

进入 MongoDB 后台管理 Shell，为文档增加索引，加快查询速度：

```
C:\mongodb\bin>mongo.exe
```

进入 shell 后直接输入如下所有命令即可：

```
db.collect_top250_movie_d_b.ensureIndex({"movie_id":1}, {background:true,unique:
true})
db.collect_movie_d_b.ensureIndex({"movie_id":1}, {background:true,unique:true})
db.collect_movie_reviews_d_b.ensureIndex({"reviews_id":1}, {background:true,uniq
ue:true})
db.collect_movie_reviews_d_b.ensureIndex({"reviews_movid_id":1}, {background:tru
e})
db.collect_movie_comments_d_b.ensureIndex({"comments_id":1}, {background:true,un
ique:true})
db.collect_movie_comments_d_b.ensureIndex({"comments_movid_id":1}, {background:t
rue})
```

```
db.collect_music_d_b.ensureIndex({"music_id":1}, {background:true,unique:true})
db.collect_music_reviews_d_b.ensureIndex({"reviews_id":1}, {background:true,uniq
ue:true})
db.collect_music_reviews_d_b.ensureIndex({"reviews_music_id":1}, {background:tru
e})
db.collect_music_comments_d_b.ensureIndex({"comments_id":1}, {background:true,un
ique:true})
db.collect_music_comments_d_b.ensureIndex({"comments_music_id":1}, {background:t
rue})
```

```
db.collect_book_d_b.ensureIndex({"book_id":1}, {background:true,unique:true})
db.collect_book_reviews_d_b.ensureIndex({"reviews_id":1}, {background:true,uniqu
e:true})
db.collect_book_reviews_d_b.ensureIndex({"reviews_music_id":1}, {background:true
})
db.collect_book_comments_d_b.ensureIndex({"comments_id":1}, {background:true,uni
que:true})
db.collect_book_comments_d_b.ensureIndex({"comments_book_id":1}, {background:tru
e})
```

2) 安装 redis 数据库

Window 下安装

下载地址：<https://github.com/MSOpenTech/redis/releases>。

Redis 支持 32 位和 64 位。这个需要根据你系统平台的实际情况选择，这里我们下载 **Redis-x64-xxx.zip** 压缩包到 C 盘，解压后，将文件夹重新命名为 **redis**。

Downloads

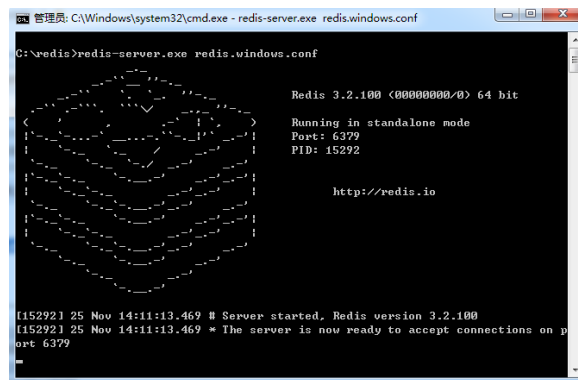
 Redis-x64-3.2.100.msi	5.8 MB
 Redis-x64-3.2.100.zip	4.98 MB
 Source code (zip)	
 Source code (tar.gz)	

打开文件夹，内容如下：

名称	修改日期	类型	大小
dump.rdb	2017/7/20 14:34	RDB 文件	15
EventLog.dll	2016/7/1 16:27	应用程序扩展	1
Redis on Windows Release Notes.docx	2016/7/1 16:07	Microsoft Office...	13
Redis on Windows.docx	2016/7/1 16:07	Microsoft Office...	17
redis.windows.conf	2016/7/1 16:07	CONF 文件	48
redis.windows-service.conf	2016/7/1 16:07	CONF 文件	48
redis-benchmark.exe	2016/7/1 16:28	应用程序	400
redis-benchmark.pdb	2016/7/1 16:28	PDB 文件	4,268
redis-check-aof.exe	2016/7/1 16:28	应用程序	251
redis-check-aof.pdb	2016/7/1 16:28	PDB 文件	3,436
redis-cli.exe	2016/7/1 16:28	应用程序	488
redis-cli.pdb	2016/7/1 16:28	PDB 文件	4,420
redis-server.exe	2016/7/1 16:28	应用程序	1,628
redis-server.pdb	2016/7/1 16:28	PDB 文件	6,916
Windows Service Documentation.docx	2016/7/1 9:17	Microsoft Office...	14

打开一个 cmd 窗口 使用 cd 命令切换目录到 C:\redis 运行：
redis-server.exe redis.windows.conf

如果想方便的话，可以把 redis 的路径加到系统的环境变量里，这样就省得再输路径了，后面的那个 redis.windows.conf 可以省略，如果省略，会启用默认的。输入之后，会显示如下界面：



```
C:\redis>redis-server.exe redis.windows.conf

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 15292

http://redis.io

[15292] 25 Nov 14:11:13.469 # Server started, Redis version 3.2.100
[15292] 25 Nov 14:11:13.469 # The server is now ready to accept connections on port 6379
```

这时候另启一个 cmd 窗口，原来的不要关闭，不然就无法访问服务端了。
切换到 redis 目录下运行，就进入到 redis 命令行模式了：
redis-cli.exe -h 127.0.0.1 -p 6379

第六章 项目总结

本次作品制作开发过程我们团队是从零开始一步一步走向成品的，每次遇到的难点都是一次学习的机会。我们团队每次在一起讨论，一起查阅资料，一起交流都会得到很多的启发，也给相互之间的配合增加了默契。

项目初期，按照计划，我们三个人一个人做前端、一个人做后端还有一个写文档，当然一开始的时候写文档的是比较轻松的，所以也会安排些辅助前后端的工作。项目结束后就会反过来一起写文档，因为有些具体的前后端内容还是要实现的那个来写。

在本项目目中，我们遇到一个典型的困难就是当数据量达到一定程度之后需要对其进行优化。如果不优化，那么读取和运算的性能就会降低。因为我们团队开始查询资料以优化查询语句。最终通过排序查询、限制数量查询以及对字段的个数进行限制等方法优化了查询语句，特别在增加索引后，查询速度得到很大的提示。当然，我们也通过查询资料了解到使用聚合函数查询更优。

限于当前环境以及实力，在未来的提升中我们会搭建集群来使得项目更加完善。同样我们也会加入一个推荐系统，弄上登录注册，根据用户日常喜好收集数据并分析，收集浏览过的页面信息结合大数据为用户提供私人定制推荐服务。同样我们会增加社交功能，为大家互相交流提供便利。

如果未来有机会进行商业推广，我们会在完善系统以及增加新功能以外，会向影院，书店等推荐我们的系统。因为在一些观影淡季可以通过我们的系统发现当前时间段最受用户喜爱的是哪部影片、用户喜欢的什么类型的影片，以帮助影院更好的排表，吸引更多顾客。可以帮助书店在引人入胜的位置放置通过大数据分析得到的顾客喜欢的书籍以及更多同类型的书籍，以增加销售量。