



# 廣東工業大學

**QG 工作室**

**数据挖掘组**

**终期考核详细报告书**

题    目 \_\_\_\_\_电影推荐系统\_\_\_\_\_

学    院 \_\_\_\_\_计算机学院\_\_\_\_\_

专    业 \_\_\_\_\_计算机科学与技术\_\_\_\_\_

年级班别 \_\_\_\_\_19 级 1 班\_\_\_\_\_

学    号 \_\_\_\_\_3119004757\_\_\_\_\_

学生姓名 \_\_\_\_\_许继元\_\_\_\_\_

2019 年 05 月

# 目录

一、项目介绍-----	3
二、实现模块-----	3
2.1 爬虫模块-----	3
爬虫模块遇到的问题-----	7
2.2 电影推荐模块-----	9
ItemCF 算法的实现-----	10
三、GUI 的实现-----	13
3.1 登录注册界面-----	15
3.2 用户主界面-----	16
3.3 电影分类搜索界面-----	17
3.4 电影详情界面-----	19
3.5 用户个人界面-----	21
四、优化-----	22
4.1 用户的冷启动问题-----	22

## 一、项目介绍

电影推荐系统——通过爬取电影数据和用户数据，再利用所爬取的数据设计并实现相关推荐算法对用户进行电影推荐。然后设计出图形用户界面(GUI)进行交互，封装成电影推荐软件，针对数据集中的用户推荐相关电影。

主要分为两大模块：

(1) 爬虫模块

(2) 推荐系统模块

## 二、实现模块

### 2.1 爬虫模块

#### - 获取电影数据

由于需要爬取的数据量是上千条，所以找到一个好的数据接口很重要，经过分析网页，最终选取接口如下：

[https://movie.douban.com/j/new\\_search\\_subjects?sort=U&range=0,10&tags=%E7%94%B5%E5%BD%B1&start=0](https://movie.douban.com/j/new_search_subjects?sort=U&range=0,10&tags=%E7%94%B5%E5%BD%B1&start=0)

显然 start=0 为索引参数。

## 第一个页面的数据：

```
▼ {data: [{directors: ["莱恩·约翰逊"], rate: "8.2", cover_x: 1685, star: "40", title: "利刃出鞘",...},...]}
▼ data: [{directors: ["莱恩·约翰逊"], rate: "8.2", cover_x: 1685, star: "40", title: "利刃出鞘",...},...]
  ► 0: {directors: ["莱恩·约翰逊"], rate: "8.2", cover_x: 1685, star: "40", title: "利刃出鞘",...}
  ► 1: {directors: ["伍思薇"], rate: "8.0", cover_x: 1500, star: "40", title: "真心半解",...}
  ► 2: {directors: ["曾国祥"], rate: "8.3", cover_x: 5906, star: "40", title: "少年的你",...}
  ► 3: {directors: ["弗兰克·德拉邦特"], rate: "9.7", cover_x: 2000, star: "50", title: "肖申克的救赎",...}
  ► 4: {directors: ["宫崎骏"], rate: "9.4", cover_x: 1080, star: "45", title: "千与千寻",...}
  ► 5: {directors: ["拜伦·霍华德", "瑞奇·摩尔", "杰拉德·布什"], rate: "9.2", cover_x: 1418, star: "45", title: "疯狂动物城",...}
  ► 6: {directors: ["詹姆斯·卡梅隆"], rate: "9.4", cover_x: 2015, star: "45", title: "泰坦尼克号",...}
  ► 7: {directors: ["罗伯特·泽米吉斯"], rate: "9.5", cover_x: 1892, star: "50", title: "阿甘正传",...}
  ► 8: {directors: ["吕克·贝松"], rate: "9.4", cover_x: 658, star: "45", title: "这个杀手不太冷",...}
  ► 9: {directors: ["山姆·哈格雷夫"], rate: "7.1", cover_x: 1500, star: "35", title: "惊天营救",...}
  ► 10: {directors: ["涅提·蒂瓦里"], rate: "9.0", cover_x: 4500, star: "45", title: "摔跤吧！爸爸",...}
  ► 11: {directors: ["饺子"], rate: "8.5", cover_x: 5594, star: "45", title: "哪吒之魔童降世",...}
  ► 12: {directors: ["文牧野"], rate: "9.0", cover_x: 2810, star: "45", title: "我不是药神",...}
  ► 13: {directors: ["罗伯·莱纳"], rate: "9.1", cover_x: 986, star: "45", title: "怦然心动",...}
  ► 14: {directors: ["拉吉库马尔·希拉尼"], rate: "9.2", cover_x: 3030, star: "45", title: "三傻大闹宝莱坞",...}
  ► 15: {directors: ["郭帆"], rate: "7.9", cover_x: 1786, star: "40", title: "流浪地球",...}
  ► 16: {directors: ["陈凯歌"], rate: "9.6", cover_x: 600, star: "50", title: "霸王别姬",...}
  ► 17: {directors: ["柯汶利"], rate: "7.7", cover_x: 1429, star: "40", title: "误杀",...}
  ► 18: {directors: ["克里斯托弗·诺兰"], rate: "9.3", cover_x: 2229, star: "45", title: "盗梦空间",...}
  ► 19: {directors: ["彼得·法雷里"], rate: "8.9", cover_x: 2000, star: "45", title: "绿皮书",...}
```

可以看到每个页面有 20 部电影，为了验证数据量的充足，尝试着传入参数 start=6000：

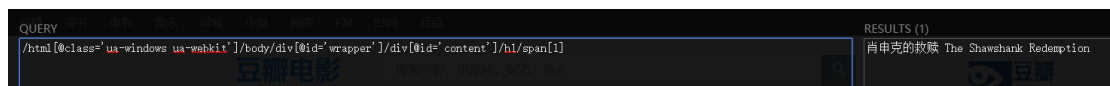
```
{
  "data": [
    {
      "directors": ["泰伦斯·马力克"],
      "rate": "7.9",
      "cover_x": 810,
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/271914",
      "title": "觉醒的咆哮",
      "cast": ["优", "学"],
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/26999591",
      "title": "宏",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/5386987",
      "title": "指",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/1963131",
      "title": "椿",
      "rate": "7.8",
      "cover_x": 700,
      "star": "40",
      "title": "他和她的故事",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/1295804",
      "title": "特",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/1438091",
      "title": "人",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/1438091",
      "title": "斯",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/1438091",
      "title": "女",
      "cover": "https://img9.doubanio.com/view/photo/s_ratio_poster",
      "url": "https://movie.douban.com/subject/1438091",
      "title": "松山博昭"],
      "rate": "7.7",
      "cover_x": 642,
      "star": "40",
      "t"
    }
  ]
}
```

再尝试几次，可以看到依然有数据（如果接口找得不好，可能数据量不够），因此开始分析这些 json 格式的数据，最后发现有一些特征（主演、编剧、电影时长、上映时间等）不在这些数据里，但是可以通过直接访问电影页面内容爬取，知道了以上信息之后就可以开始编写爬虫程序了。

## 每部电影的页面分析：

这里使用了谷歌浏览器的插件 XPath Helper 辅助分析页面，提取每部电影的相关信息。

### 1. 首先是电影名称：



为了 XPath 表达式的普适性，适当地删去前面的语句，在其他电影试试看，如果都能提取到信息，那就可以了。



可以看到电影名称成功提取。

### 2. 电影英文名称：

上述提取的字符串，用 `split` 函数以空格符分割，所得列表的第 1 个元素（索引为 0）就是是中文名称，剩下的就是英文名称（有一些电影没有英文名称）。当然，也可以用正则表达式提取。

### 3. 其他相关信息：

导演: 弗兰克·德拉邦特  
编剧: 弗兰克·德拉邦特 / 斯蒂芬·金  
主演: 蒂姆·罗宾斯 / 摩根·弗里曼 / 鲍勃·冈顿 / 威廉姆·赛德勒 / 克兰西·布朗 / 更多...  
类型: 剧情 / 犯罪  
制片国家/地区: 美国  
语言: 英语  
上映日期: 1994-09-10(多伦多电影节) / 1994-10-14(美国)  
片长: 142分钟  
又名: 月黑高飞(港) / 刺激1995(台) / 地狱诺言 / 铁窗岁月 / 肖香克的救赎  
IMDb链接: [tt0111161](https://www.imdb.com/title/tt0111161)

这块信息可以由 XPath 直接提取，但是得到的内容比较繁多，需要通过字符串分割（split 函数）和列表的元素连接（join 函数）等手段进行提取和整理。

### 4. 电影简介：

同上，很容易提取出来。

其他的相关信息有一些已经在获取的 json 格式的数据中，直接可以提取。

## - 获取用户数据

用户数据的获取采取以下思路：在每部电影页面的短评下，获取五位用户的 ID（图 3），并通过用户 ID 直接进入用户的评论界面，爬取用户看过的电影以及评分等信息。



图3

在用户的主页 URL 后加上 `/reviews` 就是用户的评论页面了。



爬取该页面和上述原理类似。

由于有一些电影是用户看过但是前面没有爬取到的，于是再根据用户看过的电影 ID 构造 URL，编写一个爬虫，补充数据集。



## 爬虫模块遇到的问题：

### - 反爬虫

当爬取的数据太多时，会被豆瓣封 IP，就算换 User-Agent 也没用，因此要使用代理（最好是高匿代理），可以使用代理 IP 的接口进行提取（如下图），每次对网页发出请求的时候，使用 `random.choice()` 随机抽取一个，此处以防万一，创建了一个 User-Agent 列表，每次请求都随机抽取一个。

除此以外，还使用了 `time.sleep()` 降低爬取数据的频率，每爬取固定量的数据后，更新代理 IP。

```
[{'https': 'https://114.217.105.237:32374', 'http': 'http://114.217.105.237:32374'}, {'https': 'https://183.154.29.148:45103', 'http': 'http://183.154.29.148:45103'}, {'https': 'https://106.56.201.96:30696', 'http': 'http://106.56.201.96:30696'}, {'https': 'https://114.97.208.237:39295', 'http': 'http://114.97.208.237:39295'}, {'https': 'https://27.29.144.156:24557', 'http': 'http://27.29.144.156:24557'}]
```

### - 提取数据

有时候一些数据单纯用 XPath 提取不到，需要使用正则表达式以及相关的 Python 语句进行提取。

### - 爬取速度

设计爬虫的时候没有去学多进程爬虫或者异步加载，导致爬虫设计得不够好，爬取速度很慢，只能在睡觉的时候打开爬虫，爬到天亮。但我有做了一些异常处理，所以不会一出错就关闭程序。

# 2.2 电影推荐模块

阅读了《推荐系统实践》的前 2 章，学习了推荐系统的协同过滤算法，其中一种是基于用户的协同过滤（UserCF），一种是基于物品的协同过滤（ItemCF）。在该项目中使用基于物品的协同过滤算法（ItemCF）进行电影推荐。

ItemCF 算法不利用物品的内容属性计算物品之间的相似度，而是通过分析用户的行为记录计算物品之间的相似度。ItemCF 算法认为，物品 A 和物品 B 具有很大的相似度是因为喜欢物品 A 的用户大也都喜欢物品 B。

## 基于物品的协同过滤算法步骤：

- 计算物品之间的相似度。
- 根据物品的相似度和用户的历史行为给用户生成推荐列表。

简单来说，ItemCF 算法给用户推荐那些和他们之前喜欢的物品相似的物品。

## 举个例子：

用户/物品	物品 A	物品 B	物品 C
用户 A	√		√
用户 B	√	√	√
用户 C	√		与物品 A 相似，推荐

基于物品的协同过滤算法可以利用用户的历史行为给推荐结果提供推荐解释，比如给用户推荐《机器学习实战》的解释可以是因为用户之前买过《统计学习方法》。

# ItemCF 算法的实现

为了实现 ItemCF 算法，首先构造用户-电影矩阵，矩阵元素记录用户对该电影的评分。

构造用户-电影矩阵伪代码如下：

```
1. for i in 用户数据.itertuples():
2.     try:
3.         用户-电影矩阵[用户索引][所有电影索引] = 用户对该电影的评分
4.     except:
5.         pass
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 4. 0. 0.]
 [0. 0. 0. ... 4. 5. 0.]
 [0. 0. 0. ... 5. 0. 4.]]
```

接着使用 sklearn 中的 cosine\_similarity 来计算电影余弦相似度矩阵：

```
1. from sklearn.metrics.pairwise import cosine_similarity
2. 电影余弦相似度矩阵 = cosine_similarity(用户-电影矩阵.T)
```

```
[[0.         0.         0.         ... 0.         0.         0.         ]
 [0.         0.         0.         ... 0.         0.         0.         ]
 [0.         0.         0.         ... 0.         0.         0.         ]
 ...
 [0.         0.         0.         ... 1.         0.3866946  0.48336824]
 [0.         0.         0.         ... 0.3866946  1.         0.         ]
 [0.         0.         0.         ... 0.48336824 0.         1.         ]]
```

### 推荐算法实现思路：

- ① 编写一个函数，功能为传入电影 ID，返回与该电影最相似的 k 部电影。
- ② 根据爬取的用户数据，选出用户比较喜欢（评分大于 4）的电影 ID 列表，并每次随机从中选取 5 个传入上述函数中，以保证每次推荐的电影不是完全重复。
- ③ 把返回的待推荐电影数据进行整理，放在推荐窗口上。

## 算法测试：

随机选取一个用户，从该用户喜欢的电影中随机抽取 5 部，进行推荐：

### 第一次推荐的结果：

约翰之子  
亚当斯一家的价值观  
唐伯虎点秋香2之四大才子  
消失爱人  
恋上你的床  
恋人们  
信笺故事  
至爱梵高·星空之谜  
北京遇上西雅图之不二情书  
天使与我同桌  
夫妻不是同林鸟  
我的天才女友  
大象  
敦刻尔克  
中国机长

### 第二次推荐的结果：

夫妻不是同林鸟  
我的天才女友  
大象  
敦刻尔克  
中国机长  
夫妻不是同林鸟  
我的天才女友  
大象  
敦刻尔克  
中国机长  
约翰之子  
亚当斯一家的价值观  
唐伯虎点秋香2之四大才子  
消失爱人

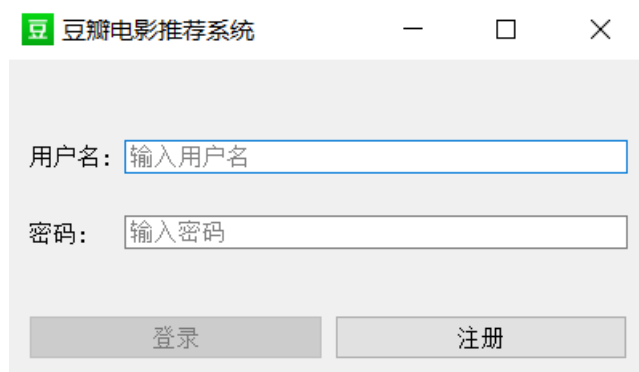
根据两次推荐结果可以看到，虽然还是有一些重复（可能是用户数据不够大），但顺序是不一样的。

## 三、GUI 的实现

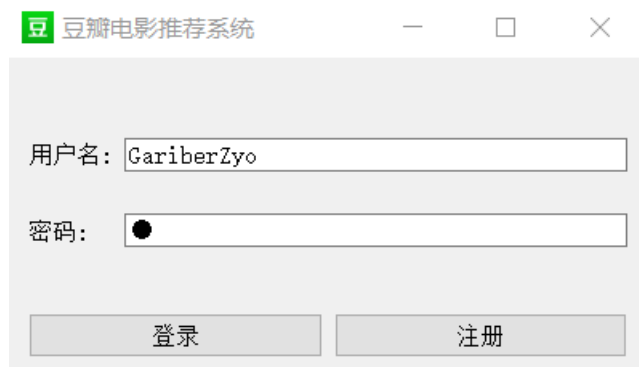
在该项目中使用 PyQt5 实现 GUI。

### 3.1 登录注册界面

登录注册界面是经典的用户图形界面，在 QQ 等平台都有类似的界面，使用 2 个 QLabel、2 个 QLineEdit、2 个 QPushButton，再使用嵌套布局即可实现界面。



此外通过 `setPlaceholderText` 设置占位字符串，再通过 `setEchoMode` 设置密码掩码，且连接检测用户名和密码输入的函数，使得输入用户名或密码为空时，登录按钮无法按下。然后设置注册按钮的 `clicked` 信号连接注册的界面。



接下来是注册界面的实现，和登录界面类似，3 个 QLabel、3 个 QLineEdit、1 个 QPushButton，再加上嵌套布局即可实现。



豆注册 ? ×

用户名:

密码:

密码:

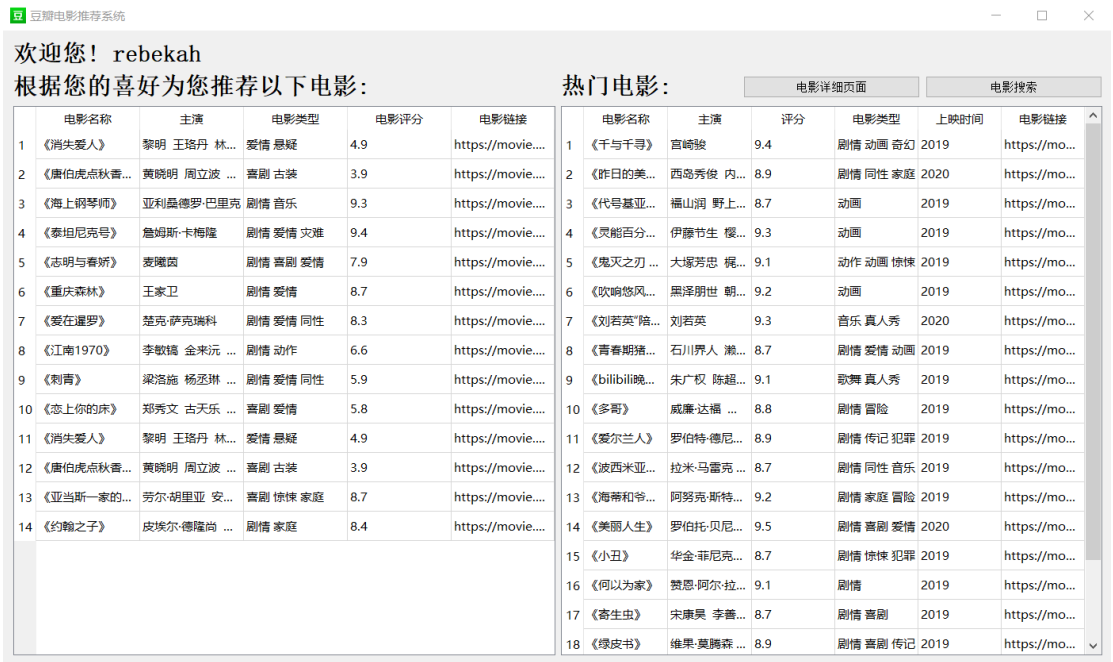
注册

该登录注册界面实现以下异常处理：

- ①用户名无法重复注册
- ②注册时两次密码输入应一致

### 3.2 用户主界面

登录成功后进入主界面，左边是个性化推荐板块，右边是热门电影板块。



电影内容通过 QTableWidgetItem 进行布置。对每张表格通过 setColumnCount 和 setHorizontalHeaderLabels 设置列数和列名之后，再将每部电影的信息通过 setItem 方法写入表格。

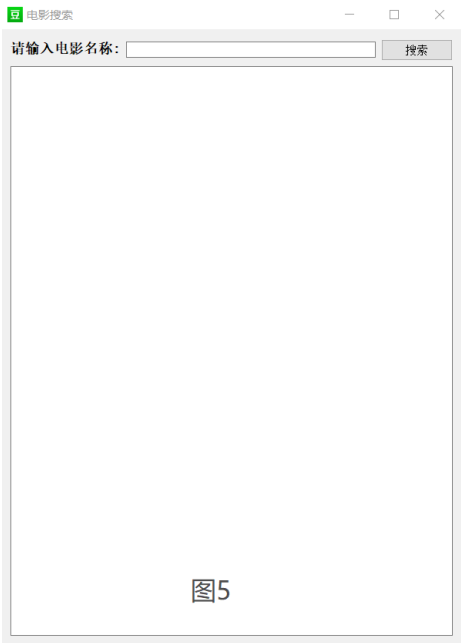


### 3.3 电影分类搜索界面

进入用户主界面之后，通过点击“电影搜索”按钮（图 4），可以进入电影搜索界面（图 5）：



电影搜索界面：



该搜索界面支持模糊搜索，例如：



可以看到复仇者联盟这一系列的电影。

### 3.4 电影详情界面

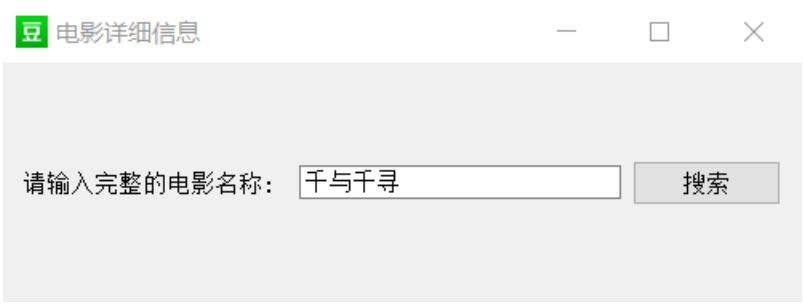
在用户主界面中，通过点击“电影详细页面”按钮（图 6），可以进入电影详细信息的搜索界面（图 7）：



电影详细信息搜索界面：




通过输入完整的电影名称，我们可以了解电影的详细信息：



按下“搜索”按钮后，显示如下界面：

电影详细信息

千与千寻



导演：宫崎骏

国家：日本

编剧：宫崎骏

语言：日语

主演：柊瑠美

上映时间：2019

类型：剧情 动画 奇幻

片长：125

评分：9.4

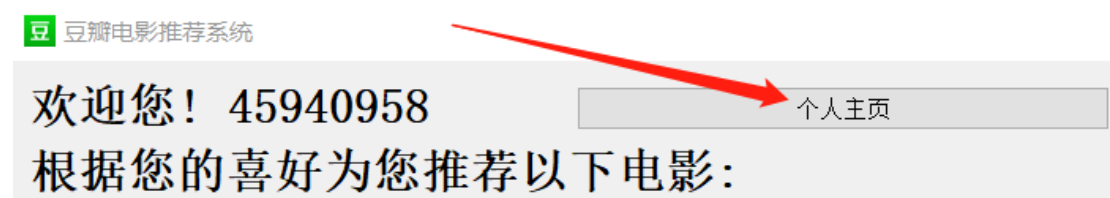
电影简介：

千寻和爸爸妈妈一同驱车前往新家，在郊外的小路上不慎进入了神秘的隧道——他们去到了另外一个诡异世界——一个中世纪的小镇。远处飘来食物的香味，爸爸妈妈大快朵颐，孰料之后变成了猪！这时小镇上渐渐来了许多样子古怪、半透明的人。千寻仓皇逃出，一个叫小白的人救了他，喂了她阻止身体消失的药，并且告诉她怎样去找锅炉爷爷以及汤婆婆，而且必须获得一份工作才能不被魔法变成别的东西。千寻在小白的帮助下幸运地获得了一份在浴池打杂的工作。渐渐她不再被那些怪模怪样的人吓倒，并从小玲那儿知道了小白是凶恶的汤婆婆的弟子。一次，千寻发现小白被一群白色飞舞的纸人打伤，为了救受伤的小白，她用河神送给她的药丸驱出了小白身体内的封印以及守封印的小妖精，但小白还是没有醒过来。为了救小白，千寻又踏上了她的冒险之旅。

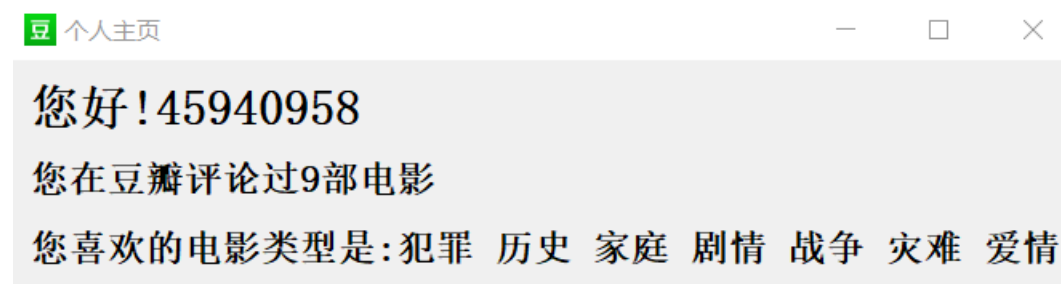
导演、编剧、主演、电影简介等信息都可以看到。

### 3.5 用户个人界面

进入主界面后，有一个“个人主页”按钮

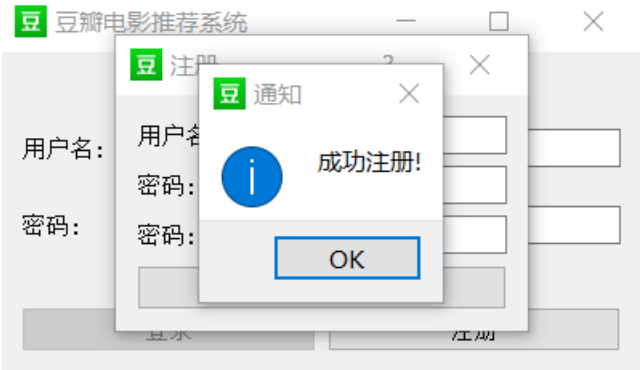


点击，简单的个人信息显示如下：



# 四、优化

## 4.1 用户的冷启动问题



如图，用户新注册时，会弹出一个窗口，询问用户喜欢的电影类型：

豆瓣电影推荐系统

由于您是新用户  
我们在此诚挚地询问您喜欢的电影类型, 以便我们更好地向您推荐电影:  
例如: 剧情、动作、喜剧、科幻、悬疑、爱情、恐怖等

喜剧

提交

此处输入“喜剧”进行测试，结果如下：

豆瓣电影推荐系统

欢迎您！

个人主页

根据您的喜好为您推荐以下电影：

	电影名称	主演	电影评分	电影类型	上映时间	电影链接
1	《美人鱼》	邓超 罗志祥 ...	6.7	喜剧	2016	https://...
2	《飞驰人生》	沈腾 黄景瑜 ...	6.9	喜剧	2019	https://...
3	《超能陆战...	斯科特·安第...	8.7	喜剧	2015	https://...
4	《神偷奶爸》	史蒂夫·卡瑞...	8.6	喜剧	2010	https://...
5	《夏洛特烦...	沈腾 马丽 ...	7.6	喜剧	2015	https://...
6	《调音师》	阿尤斯曼·库...	8.3	喜剧	2019	https://...
7	《唐人街探案...	王宝强 刘昊...	6.7	喜剧	2018	https://...
8	《功夫》	周星驰 元秋 ...	8.6	喜剧	2004	https://...
9	《西虹市首...	沈腾 宋芸桦 ...	6.6	喜剧	2018	https://...
10	《唐伯虎点秋...	周星驰 巩俐 ...	8.6	喜剧	1993	https://...
11	《大话西游之...	周星驰 吴孟...	9.0	喜剧	1995	https://...
12	《大话西游之...	周星驰 吴孟...	9.2	喜剧	1995	https://...
13	《寻梦环游...	安东尼·冈萨...	9.1	喜剧	2017	https://...
14	《疯狂动物...	金妮弗·古德...	9.2	喜剧	2016	https://...
15	《校园情圣》	利·刘易斯 丹...	8.1	喜剧	2020	https://...

可以看到，根据用户喜欢的电影类型给用户进行了个性化推荐。