****

****

**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

1. 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**

1. **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：

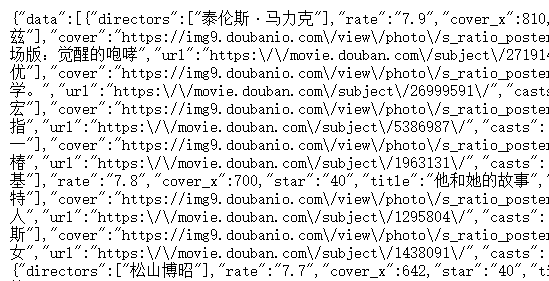
<https://movie.douban.com/j/new_search_subjects?sort=U&range=0,10&tags=%E7%94%B5%E5%BD%B1&start=0>

显然start=0为索引参数。

**第一个页面的数据：**



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：

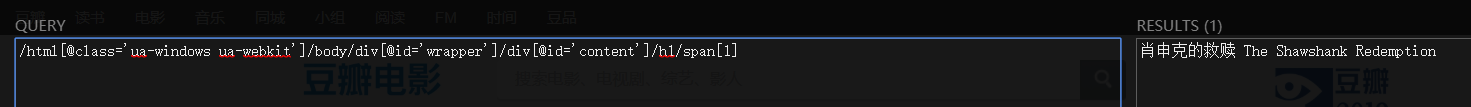


再尝试几次，可以看到依然有数据（如果接口找得不好，可能数据量不够），因此开始分析这些json格式的数据，最后发现有一些特征（主演、编剧、电影时长、上映时间等）不在这些数据里，但是可以通过直接访问电影页面内容爬取，知道了以上信息之后就可以开始编写爬虫程序了。

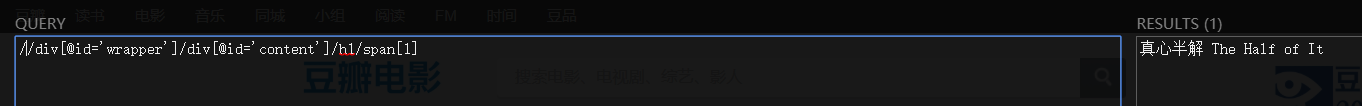
**每部电影的页面分析：**

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。

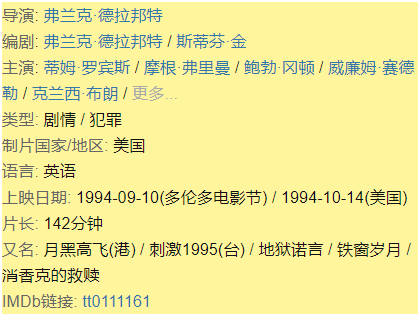


可以看到电影名称成功提取。

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）就是是中午名称，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：



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。



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。

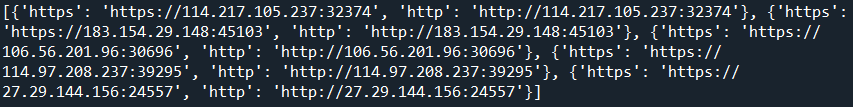
由于有一些电影是用户看过但是前面没有爬取到的，于是再根据用户看过的电影ID构造URL，编写一个爬虫，补充数据集。

**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。



* **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 的用户大都也喜欢物品 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。

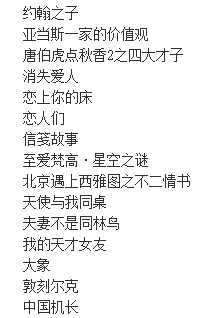
② 根据爬取的用户数据，选出用户比较喜欢（评分大于4）的电影ID列表，并每次随机从中选取5个传入上述函数中，以保证每次推荐的电影不是完全重复。

③ 把返回的待推荐电影数据进行整理，放在推荐窗口上。

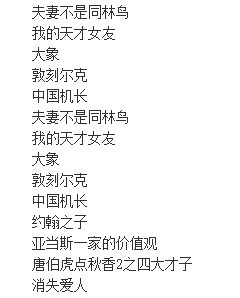
**算法测试：**

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：**

****

**第二次推荐的结果：**

****

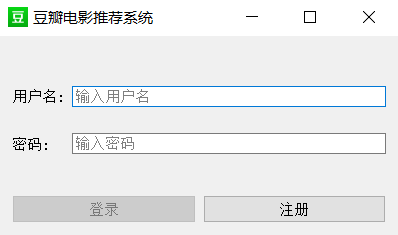
根据两次推荐结果可以看到，虽然还是有一些重复（可能是用户数据不够大），但顺序是不一样的。

1. **GUI的实现**

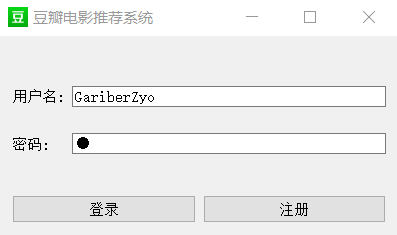
在该项目中使用PyQt5实现GUI。

**3.1 登录注册界面**

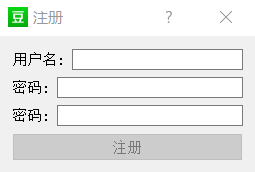
登录注册界面是经典的用户图形界面，在QQ等平台都有类似的界面，使用2个QLabel、2个QLineEdit、2个QPushButton，再使用嵌套布局即可实现界面。



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。



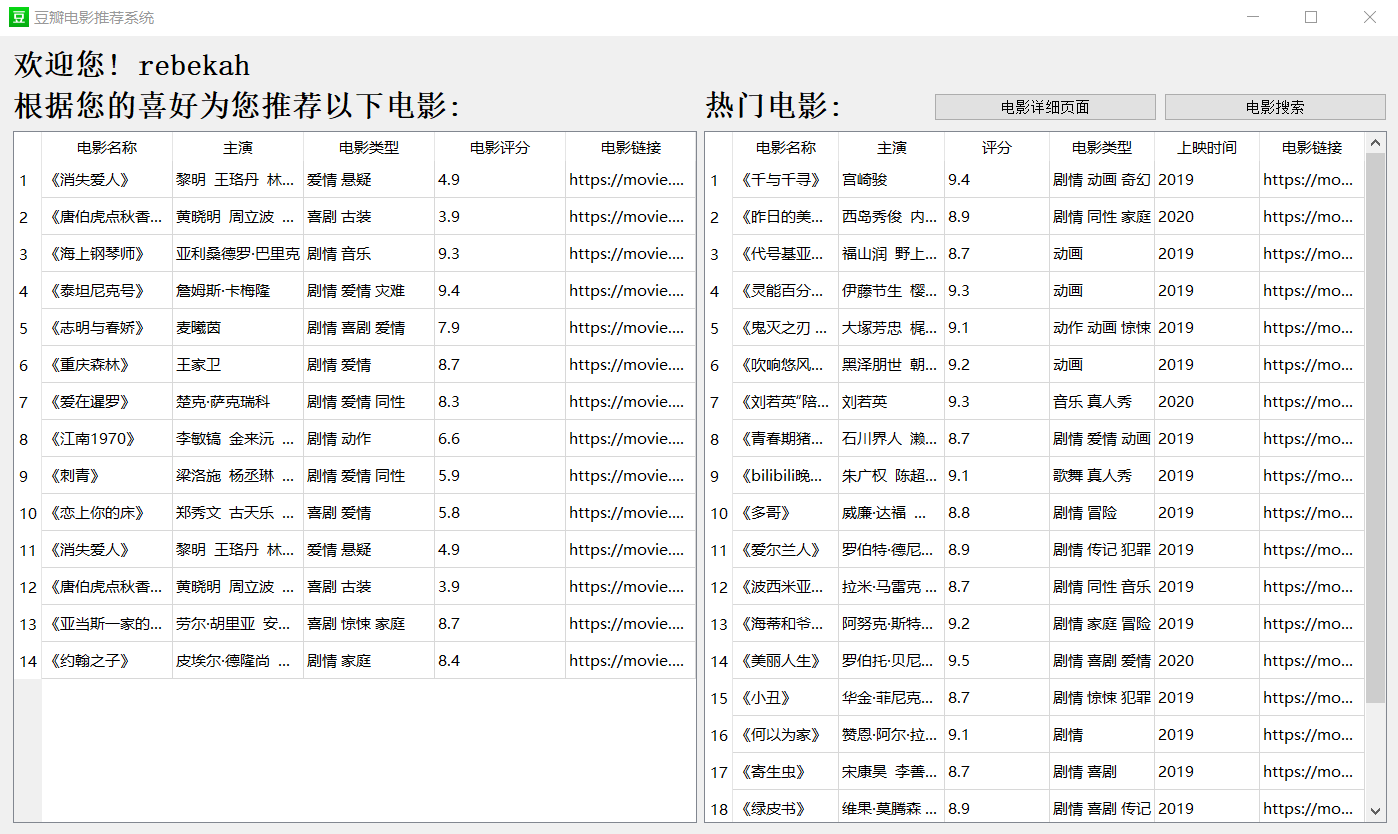
**该登录注册界面实现以下异常处理：**

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。



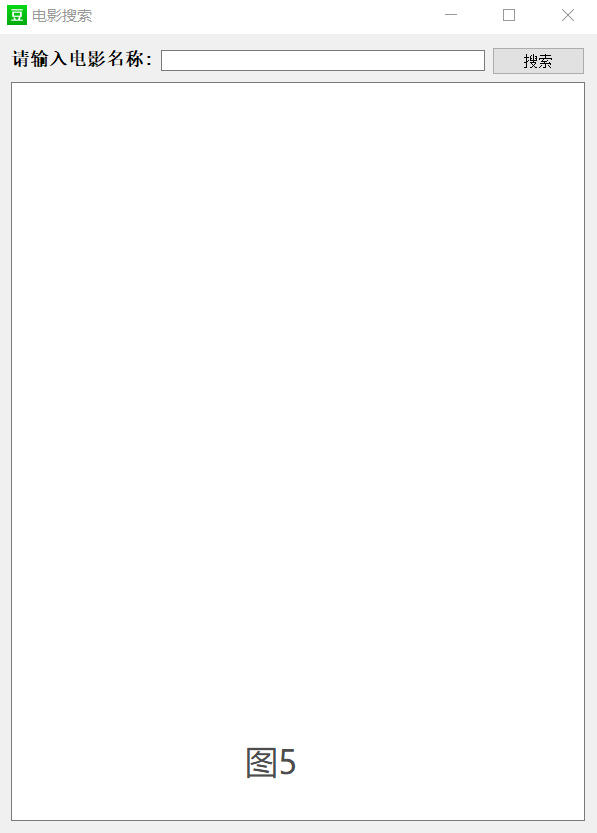
电影内容通过QTableWidget进行布置。对每张表格通过setColumnCount和setHorizontalHeaderLabels设置列数和列名之后，再将每部电影的信息通过setItem方法写入表格。

**3.3 电影分类搜索界面**

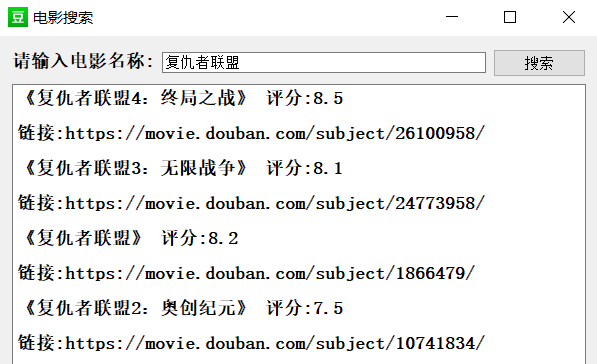
进入用户主界面之后，通过点击“电影搜索”按钮（图4），可以进入电影搜索界面（图5）：



电影搜索界面：



该搜索界面支持模糊搜索，例如：



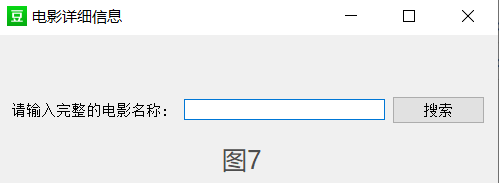
可以看到复仇者联盟这一系列的电影。

**3.4 电影详情界面**

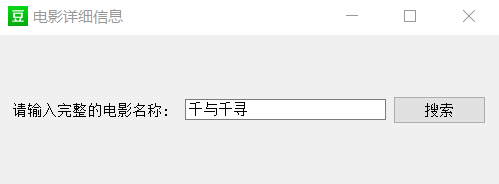
在用户主界面中，通过点击“电影详细页面”按钮（图6），可以进入电影详细信息的搜索界面（图7）：



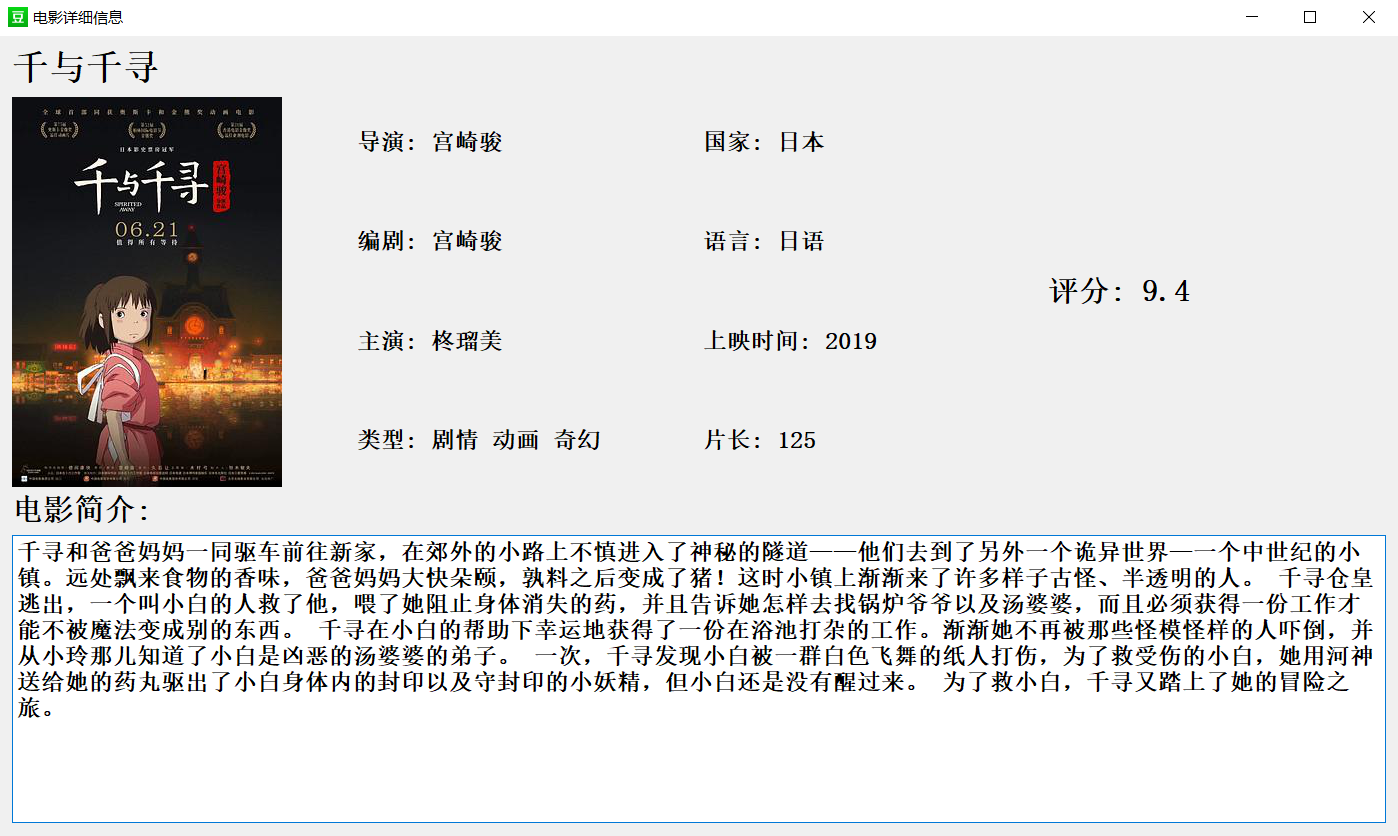
电影详细信息搜索界面：



通过输入完整的电影名称，我们可以了解电影的详细信息：



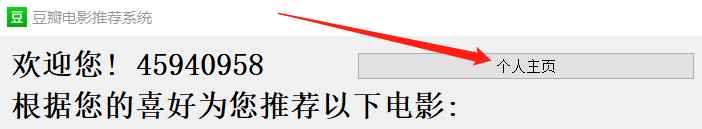
按下“搜索”按钮后，显示如下界面：



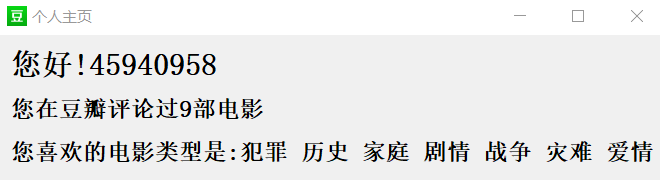
导演、编剧、主演、电影简介等信息都可以看到。

**3.5 用户个人界面**

进入主界面后，有一个“个人主页”按钮

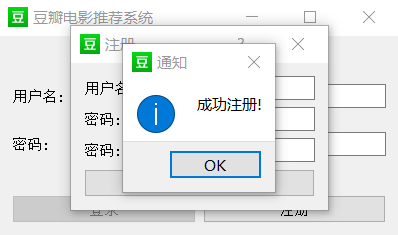


点击，简单的个人信息显示如下：

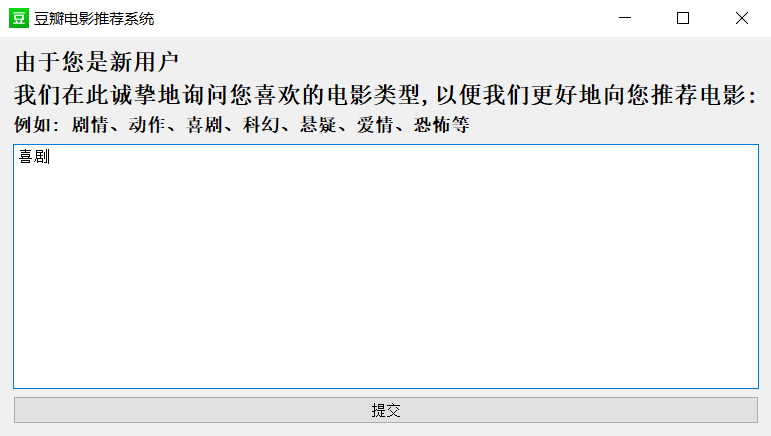


1. **优化**

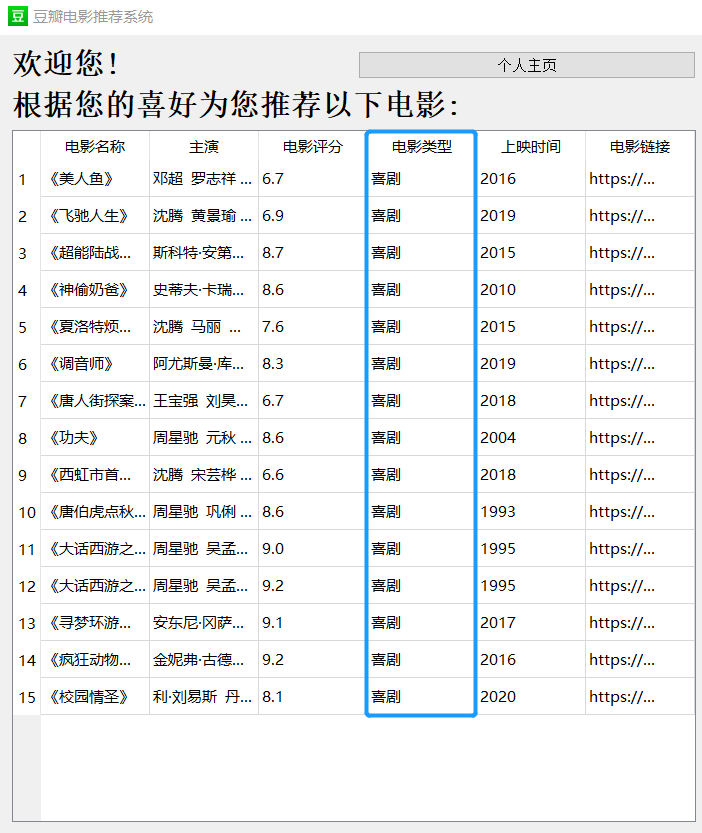
4.1用户的冷启动问题



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：



此处输入“喜剧”进行测试，结果如下：



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。