目录

[1. 引言 0](#_Toc517380826)

[2. 系统开发平台以及相关技术 0](#_Toc517380827)

[2.1. 系统开发环境 0](#_Toc517380828)

[2.2. 相关技术 0](#_Toc517380829)

[2.2.1. 基于用户的协同过滤算法 0](#_Toc517380830)

[2.2.2. K-近邻算法 1](#_Toc517380831)

[2.2.3. 爬虫 1](#_Toc517380832)

[2.2.4. sqlite 2](#_Toc517380833)

[2.2.5. PYQT5 2](#_Toc517380834)

[2.2.6. QtDesigner 2](#_Toc517380835)

[2.3. 团队 2](#_Toc517380836)

[2.3.1. 成员 2](#_Toc517380837)

[2.3.2. 分工 2](#_Toc517380838)

[3. 电影推荐系统的设计与实现 3](#_Toc517380839)

[3.1. 系统设计思路 3](#_Toc517380840)

[3.2. 系统的功能设计与实现 4](#_Toc517380841)

[3.2.1. 电影展示 4](#_Toc517380842)

[3.2.2. 注册 6](#_Toc517380843)

[3.2.3. 登录 7](#_Toc517380844)

[3.2.4. 个人信息修改 9](#_Toc517380845)

[3.2.5. 电影评论 10](#_Toc517380846)

[3.2.6. 推荐功能 12](#_Toc517380847)

[4. 总结与展望 13](#_Toc517380848)

[4.1. 总结 13](#_Toc517380849)

[4.2. 展望 13](#_Toc517380850)



基于用户的电影推荐系统

# 引言

随着互联网的发展，电子商务网站迅猛发展，网络商品信息量不断膨胀。用户要在众多的商品中挑选出自己真正需要的商品难度越来越大。面对海量的信息空间，Web上大部分的信息可能对于大部分的用户是毫无用处的，这导致了用户搜索信息的时间成本越来越高，出现了信息过载（Information Overload）的问题，并导致了系统故障的问题。为了解决信息过载的问题，出现了个性化推荐系统。个性化推荐系统为广大消费者在众多的商品中找到自己所需要的商品提供了有利条件。体验过个性化推荐系统的服务的用户们越来越青睐个性化推荐系统，因此对个性化推荐的研究也越来越热门和必需。

个性化推荐同样也是电影推荐的一种重要手段，能为电子商务电影网站带来巨大的经济利益。个性化电影推荐的目标是根据具有相似偏好的用户的意见向目标用户推荐他/她没有关注到但可能喜欢的电影。优秀的个性化推荐系统能够发掘出用户潜在喜欢的电影。从用户角度而言，如果一打开并登陆网站，不需要花费很多时间翻看网页就能找到自己喜欢的电影，那么用户对这样的网站一定会倍加青睐。这样就更好地建立起了用户与网站之间的关联，锁定用户群，从而提高电影推荐网站的市场竞争力。

由此，我们做了一个基于用户的电影推荐系统。

# 系统开发平台以及相关技术

## 系统开发环境

* 系统开发环境：Eclipse,Anaconda3，pycharm
* 系统开发语言：python
* 系统后台数据库：sqlite
* UI设计库：PYQT5
* UI设计工具：Qtdesigner

## 相关技术

### 基于用户的协同过滤算法

协同过滤（Collaborative Filtering）是现今推荐系统中应用最为成熟的一个推荐算法系类，其中使用比较广泛有基于物品的协同过滤，基于用户的协同过滤，它利用兴趣相投、拥有共同经验之群体的喜好来推荐使用者感兴趣的资讯，个人透过合作的机制给予资讯当程度的回应（如评分）并记录下来以达到过滤的目的进而帮助别人筛选资讯。

基于用户的协同过滤，它根据系统用户的基本信息来发现用户之间的相关程度，然后将相似用户喜爱的其他物品推荐给当前用户。

系统首先会根据用户的属性建模，比如用户的年龄、性别、兴趣等信息。根据这些特征计算用户间的相似度。例如系统通过计算，发现用户A和C比较相似，于是就把A喜欢的物品A推荐给用户C。

### K-近邻算法

KNN是通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中K通常是不大于20的整数。KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

KNN算法的思想是：就是在训练集中数据和标签已知的情况下，输入测试数据，将测试数据的特征与训练集中对应的特征进行相互比较，找到训练集中与之最为相似的前K个数据，则该测试数据对应的类别就是K个数据中出现次数最多的那个分类，其算法的描述为：

1）计算测试数据与各个训练数据之间的距离；

2）按照距离的递增关系进行排序；

3）选取距离最小的K个点；

4）确定前K个点所在类别的出现频率；

5）返回前K个点中出现频率最高的类别作为测试数据的预测分类。

本系统在推荐功能中使用了该算法，主要是针对新用户数据的情况下。

### 爬虫

网络爬虫是一个自动提取网页的程序，它为搜素引擎从万维网上下载网页，是搜索引擎的重要组成。传统爬虫从一个或若干初始网页的URL开始，获得初始网页上的URL，在抓取网页的过程中，不断从当前页面上抽取新的URL放入队列,直到满足系统的一定停止条件。聚焦爬虫的工作流程较为复杂，需要根据一定的网页分析算法过滤与主题无关的链接，保留有用的链接并将其放入等待抓取的URL队列。然后，它将根据一定的搜索策略从队列中选择下一步要抓取的网页URL，并重复上述过程，直到达到系统的某一条件时停止。另外，所有被爬虫抓取的网页将会被系统存贮，进行一定的分析、过滤，并建立索引，以便之后的查询和检索；对于聚焦爬虫来说，这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。

### sqlite

SQLite，是一款轻型的数据库，是遵守ACID的关系型数据库管理系统，它包含在一个相对小的C库中。它是D.RichardHipp建立的公有领域项目。它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。它能够支持Windows/Linux/Unix等等主流的操作系统，同时能够跟很多程序语言相结合，比如 Tcl、C#、PHP、Java等，还有ODBC接口，同样比起Mysql、PostgreSQL这两款开源的世界著名数据库管理系统来讲，它的处理速度比他们都快。不像常见的客户-服务器范例，SQLite引擎不是个程序与之通信的独立进程，而是连接到程序中成为它的一个主要部分。所以主要的通信协议是在编程语言内的直接API调用。这在消耗总量、延迟时间和整体简单性上有积极的作用。整个数据库(定义、表、索引和数据本身)都在宿主主机上存储在一个单一的文件中。它的简单的设计是通过在开始一个事务的时候锁定整个数据文件而完成的。

### PYQT5

PyQt是一个创建GUI应用程序的工具包。它是python编程语言和QT库的成功融合。Qt库是目前最强大的库之一。PyQt是由Phil Thompson 开发。

**PyQt**实现了一个Python模块集。它有超过300类，将近6000个函数和方法。它是一个多平台的工具包，可以运行在所有主要操作系统上，包括UNIX，Windows和Mac。 **PyQt**采用双许可证，开发人员可以选择GPL和商业许可。在此之前，GPL的版本只能用在Unix上，从**PyQt**的版本4开始，GPL许可证可用于所有支持的平台。

### QtDesigner

熟悉Qt的人或多或少的都会使用**Qt Designer**(以下简称Designer)，在Qt中，提供了一个用于可视化设计软件的界面的过程，使用Designer，你会减少使用代码来设计界面的过程，当然更多的时候是结合着代码以及Designer两者，从而实现较好的界面，对于熟悉MFC的人，对于生成的窗体上拖拽控件应该是非常的熟悉的。

# 电影推荐系统的设计与实现

这一部分是结合上述的技术，设计并实现电影推荐系统。这里主要介绍电影推荐系统的设计思想与实现过程。

## 系统设计思路

1. 数据的预处理阶段：对老师给定的数据集（ml-100k）进行预处理得到” 用户表.txt”并且生成用户信息表（user\_table）。
2. 实现基于用户的电影推荐算法：这一部分分为两种情况。①对于新用户登录，我们选用knn算法为他进行喜欢的电影进行预测；②对于数据库存在的老用户，使用协同过滤算法对他们进行电影推荐。推荐功能的实现主要是Functions.py。
3. 爬虫：这一部分视为了丰富电影系统，就在维基百科上爬取了电影的海报和部分电影简介。海报保存在image文件夹中，电影简介保存在books文件夹中。

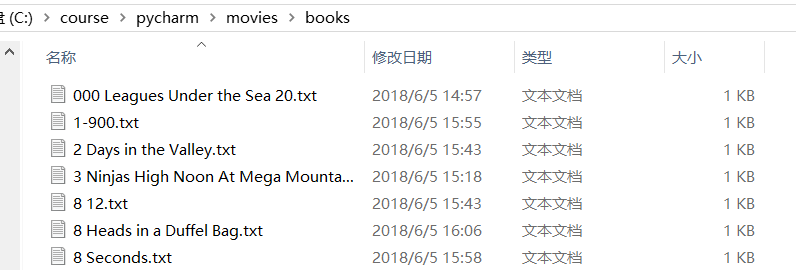


图3-1：部分电影简介文档

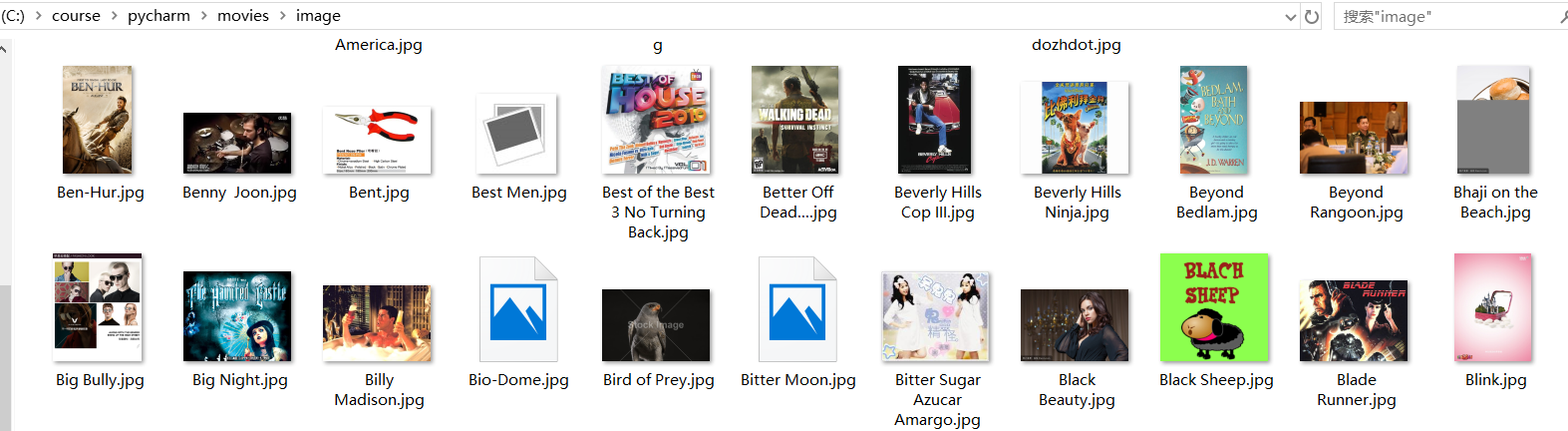


图3-2：部分海报

1. 该系统需要完成的功能有：注册，登录，修改个人信息，对电影的评价，搜索电影等主要功能。（具体实现见3.2）
2. 数据库设计：①电影评论（user\_review）：用户对电影的评价存入数据库中；②用户信息表（user\_table）：存入的老用户信息，即数据集中已有的信息，新注册的用户信息也存入进去。



图3-3：user\_review部分预览

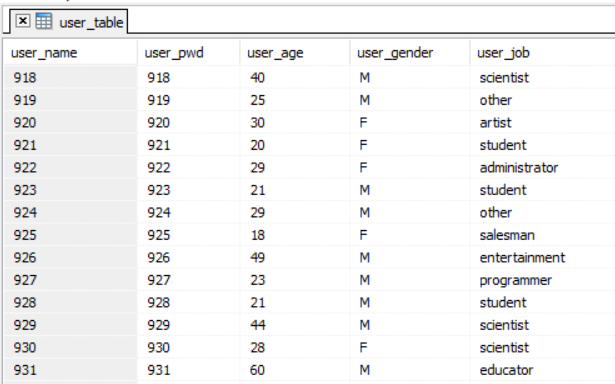


图3-4：user\_table部分预览

1. UI设计：使用QtDesigner设计。

## 系统的功能设计与实现

### 电影展示

* **实现：**这一部分的电影展示分三种情况：①未登陆的游客进入首页后，首页会随机推荐四部电影；②如果游客登陆后，则会基于用户进行个性化推荐，这一部分的实现会在登录功能中详述；③如果用户是新注册用户，则会对用户使用knn算法预测出新用户喜欢的电影类型，这个新用户只有在登录后才会进行电影推荐，所以电影的展示也是在系统登录进行的。三种情况的代码详见下面。
* **函数代码部分：**

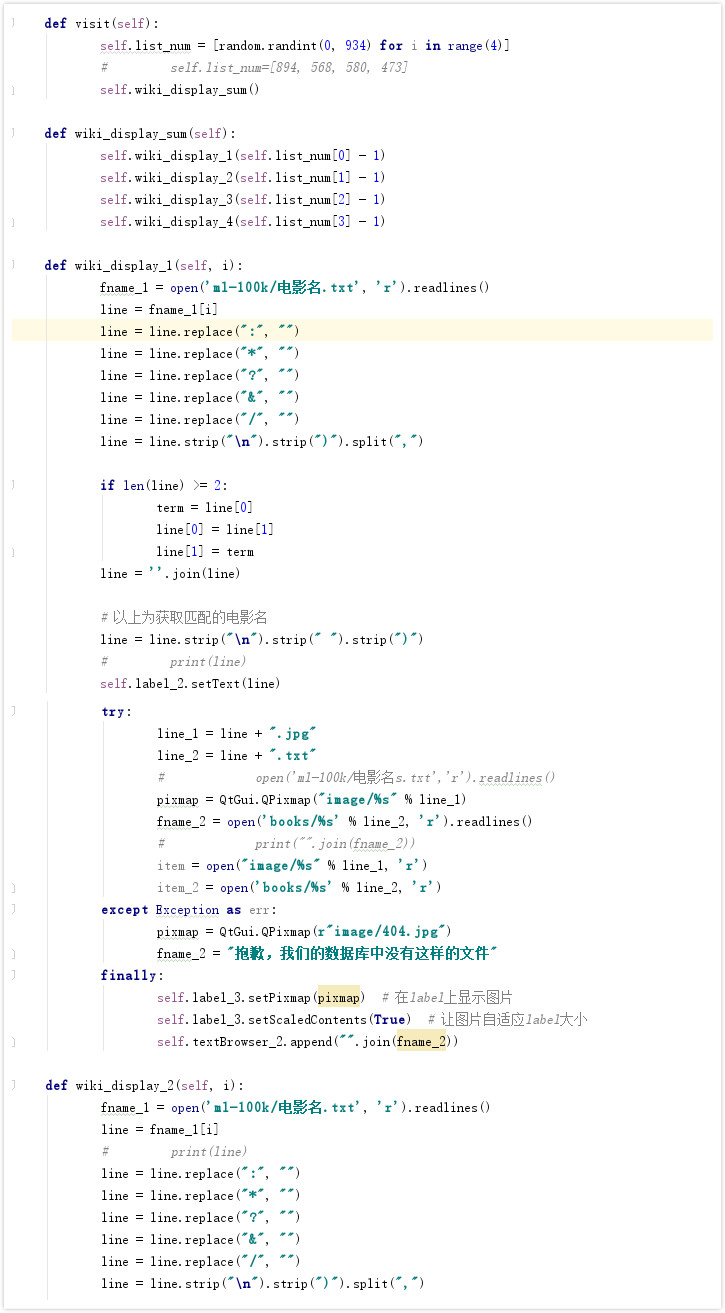


图3-5：未登录首页的电影展示

* **UI展示：**

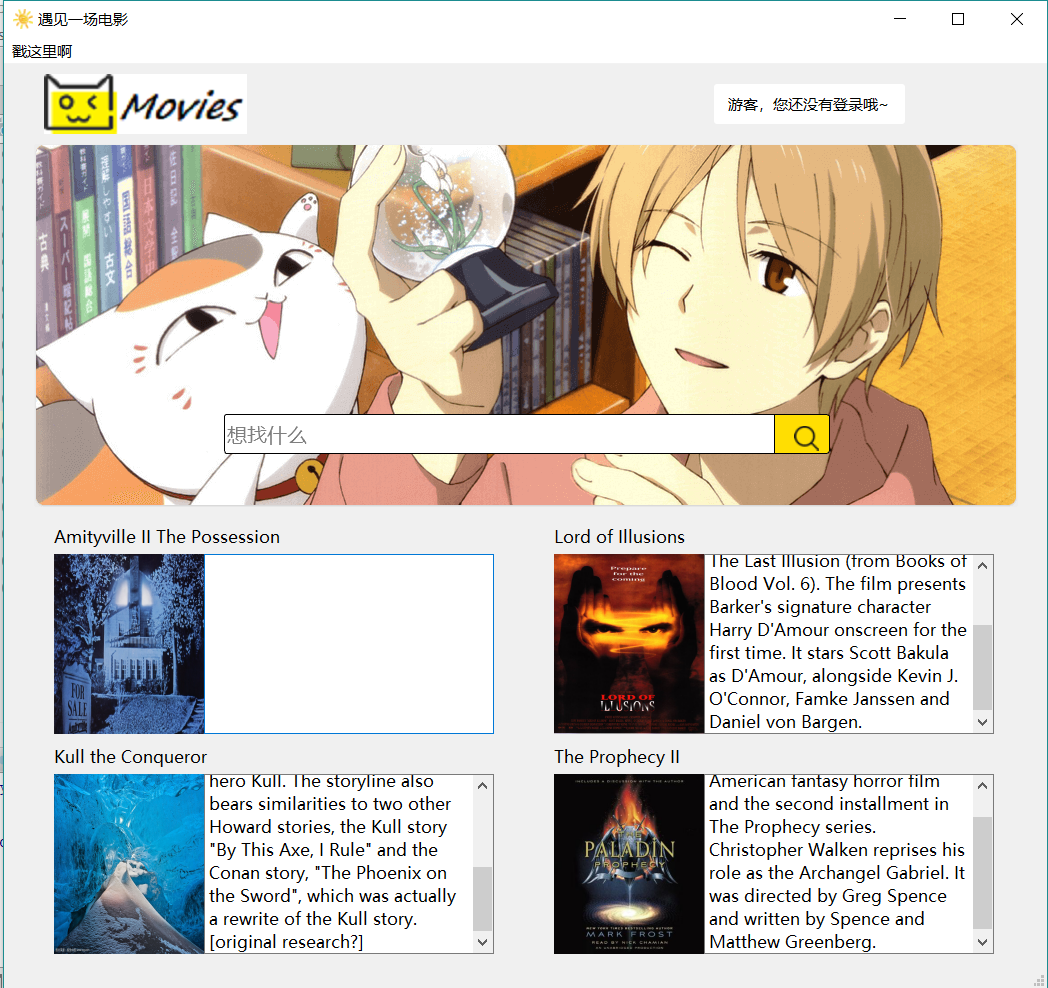
****

图3-6：系统首页

### 注册

* **实现：实现系统的注册功能。设有数据库，可以对账户进行管理。对注册的用户进行用户名是否重复的检测；对登录的用户进行用户名和密码的匹配的检测。**
* **函数代码部分：**

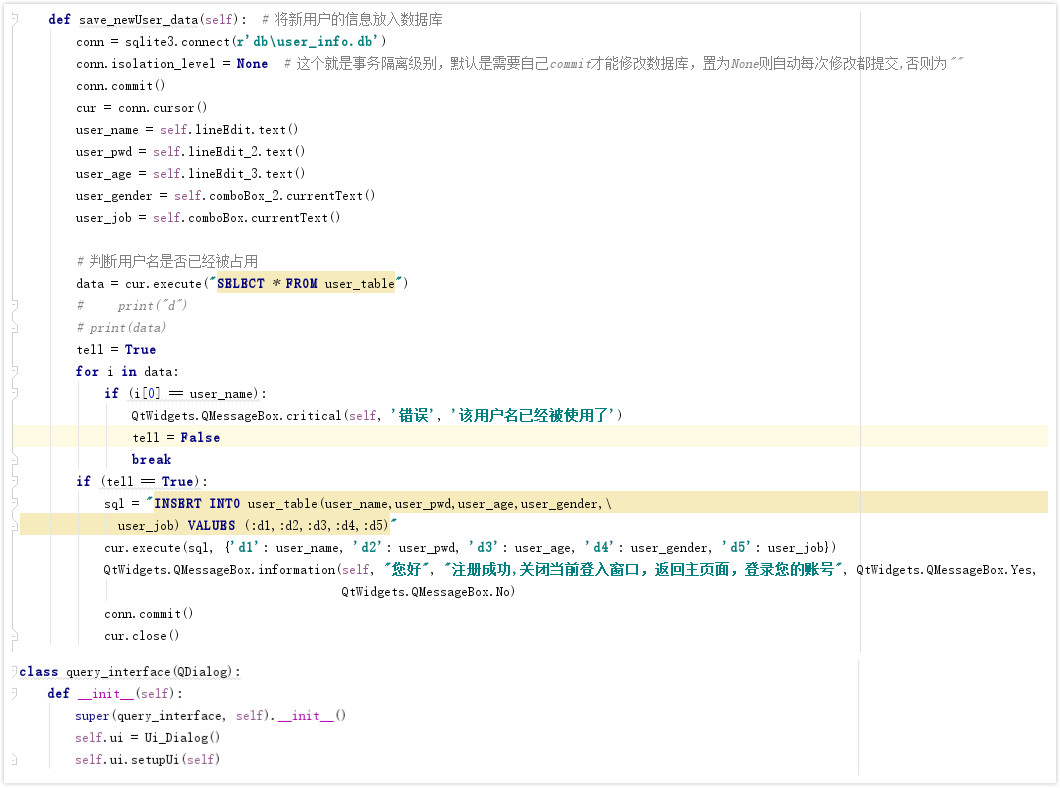


图3-7：“注册”代码

* **UI展示：**



图3-8：“注册”界面

### 登录

* **实现：**用户先进入首页，在首页进行登录并进行信息的更新和电影展示的更新。
* **函数代码部分：**



图3-9：“登录”代码

* **UI展示：**



图3-10：“登录”界面

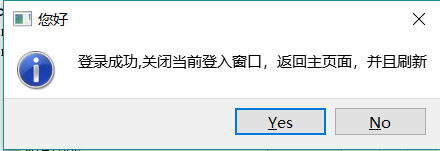


图3-11：“登录”成功提示窗口

### 个人信息修改

* **实现：**个人信息修改部分不能对用户名进行修改，其他的属性均可以进行修改。但是在登陆的用户进行信息修改后，首页的电影推荐是不能刷新的。
* **函数代码部分：**

****

图3-12：“个人信息修改”代码

* **UI展示：**

****

图3-13：“修改个人信息”界面

### 电影评论

* **实现：**这一部分是通过主页面的电影搜索跳转过来。这个页面可以看到电影海报和电影简介，并能自己写评论，提交到数据库。而且可以看到其他人提交的热门评论，只不过这一部分并没有完全实现。
* **函数代码部分：**

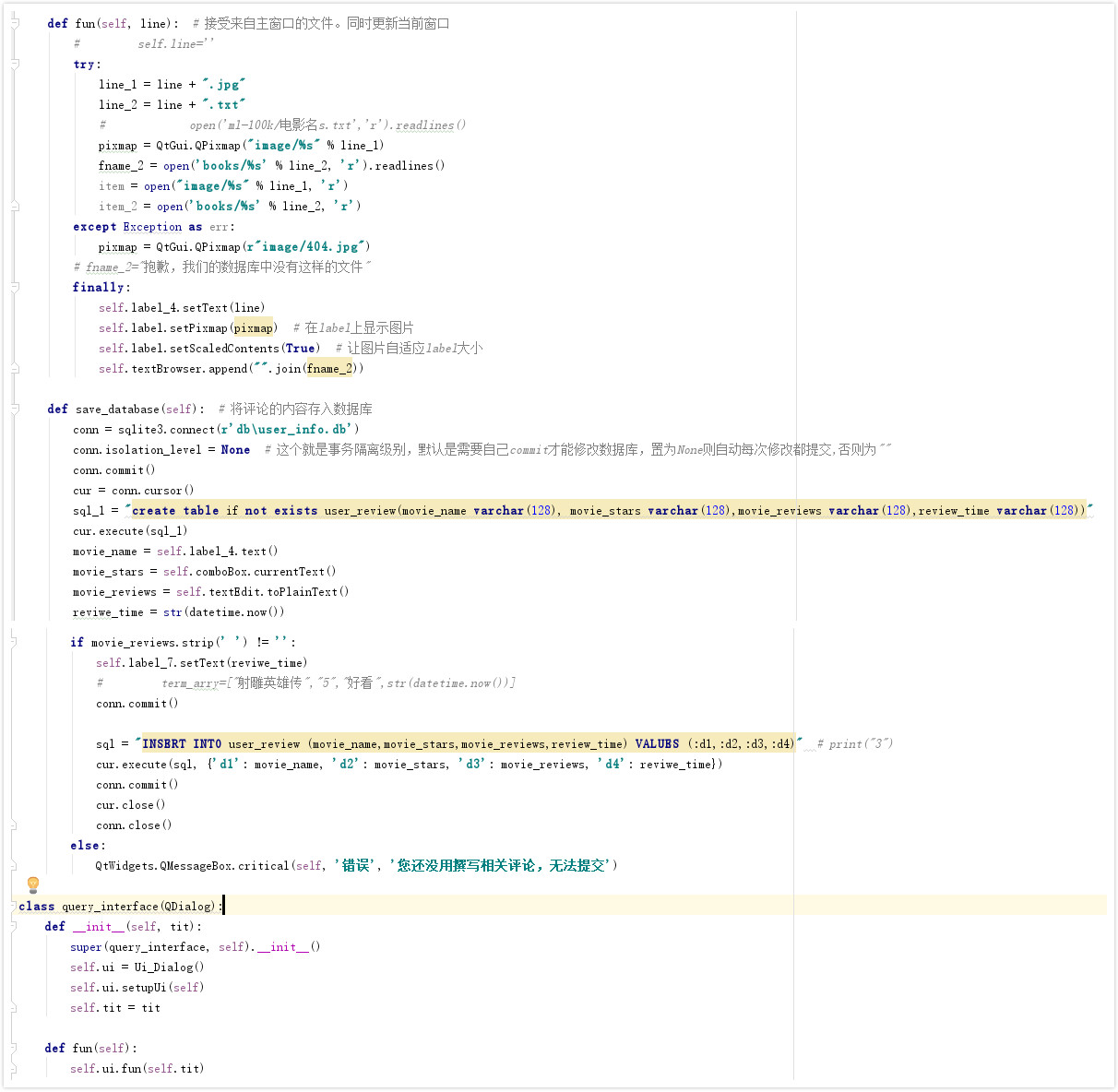


图3-14：“电影评论”代码

* **UI展示：**

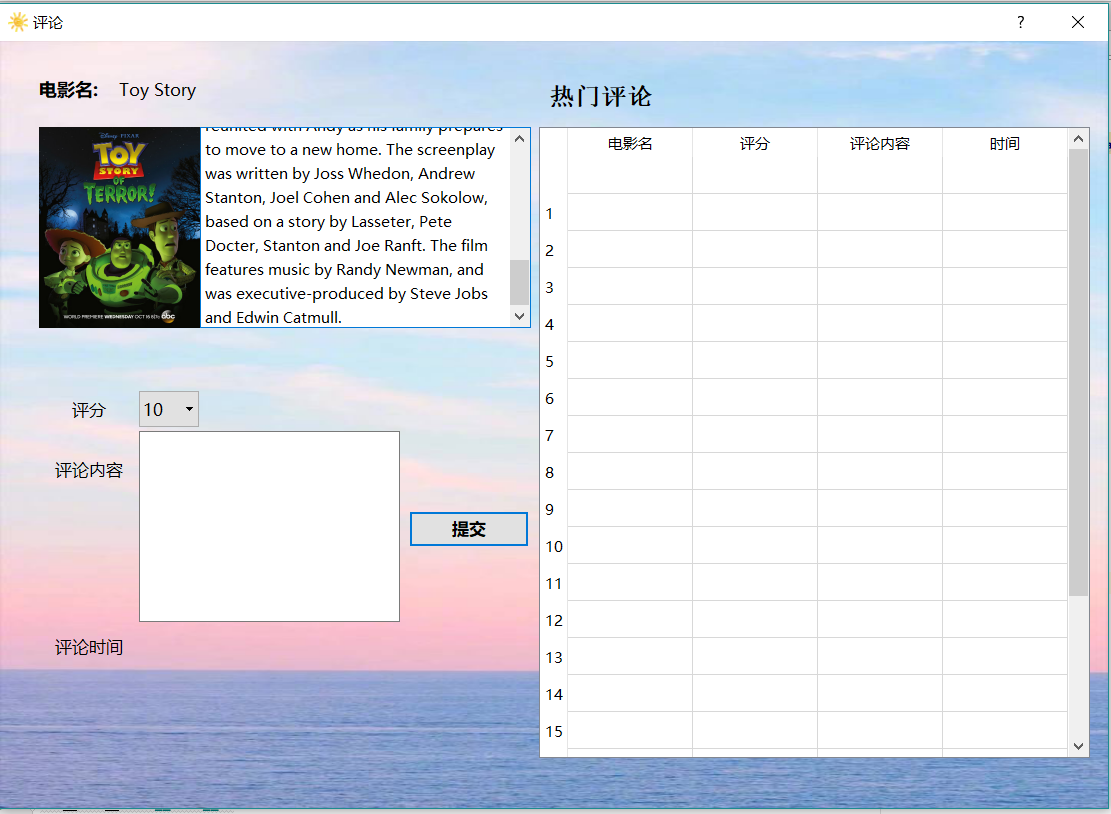
****

图3-15：“电影评论”界面

### 推荐功能



图3-16：“knn算法”预测



图3-17：“spark的ALS算法”

# 总结与展望

## 总结

该系统主要基于用户的推荐系统大概完成了任务，很多方面依旧有许多的不足，但也着实学到了不少。UI的设计，协同过滤算法的应用，knn算法的运用，爬虫等，也算一个有模有样的系统了。如果时间更充足，希望进一步完善。

## 展望

本系统也存在着一些如实时性问题，当数据量比较大时，相似度矩阵的计算比较耗时，相似度矩阵的稀疏性导致浪费掉大量的磁盘和内存空间。当新的评分加入系统时，系统无法及时重新计算所有的相似度信息，新加入的评分就不能迅速反映在推荐结果中。在构造用户模型时使用的基于邻域的模型，使用的是用户的评分数据，这种评分数据是用户的历史数据并不能完全准确反映出用户当前兴趣。就UI设计和各方面都有点 “粗制滥造”，但主要完成了基于用户的电影推荐。