北京工业大学

2020 - 2021 学年 第1学期

信息学部 计算机学院

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称： | 数据结构课程设计 | | |
| 报告性质： | 实验报告 | | |
| 学号： | 18072014 | 姓名： | 刘益涵 |
| 任课教师： | 王众 | 课程性质： | 学科基础必修课 |
| 学分： | 2 | 学时： | 60 |
| 班级： | 180720 | 成绩： |  |
| 教师评语： |  | | |

|  |  |  |
| --- | --- | --- |
| 需求分析 | 根据题中需求，提供功能划分说明 |  |
| 设计 | 逻辑结构、存储结构设计、算法描述 |  |
| 使用说明 | 界面是否友好 |  |
| 总结 | 是否感悟有收获 |  |
| 摘要 | 考察文字抽象能力 |  |
| 格式 | 是否有目录、页号 |  |

2020年 12 月 10 日

目录

[1需求分析 3](#_Toc2548)

[1.1功能概述 3](#_Toc21636)

[1.1.1基础功能 3](#_Toc8209)

[1.1.2拓展功能 3](#_Toc3870)

[1.2需要处理的数据 3](#_Toc4743)

[1.3程序开发运行环境 3](#_Toc1422)

[1.4用户界面设计 4](#_Toc10209)

[2数据结构设计 5](#_Toc13603)

[2.1主要数据结构定义 5](#_Toc15707)

[2.1.1逻辑结构 5](#_Toc11947)

[2.1.2存储结构 5](#_Toc12038)

[2.2整体结构及模块功能描述 5](#_Toc6785)

[2.3各模块功能描述 5](#_Toc5658)

[2.3.1电影节点类（movie） 5](#_Toc7718)

[2.3.2电影链表类（Movielist） 5](#_Toc13998)

[2.4核心算法流程图 6](#_Toc14209)

[3详细设计 6](#_Toc118)

[3.1各模块设计 6](#_Toc824)

[3.1.1 movie 6](#_Toc25148)

[3.1.2 Movielist 7](#_Toc20165)

[3.2各模块核心函数 8](#_Toc28190)

[1.1、交换值函数 8](#_Toc7212)

[1.2添加评论函数 8](#_Toc31971)

[2.1判断链表是否为空 9](#_Toc11400)

[2.2清空链表 9](#_Toc12522)

[2.3访问节点 9](#_Toc4033)

[2.4尾插入 9](#_Toc32116)

[2.4遍历搜索 10](#_Toc27431)

[2.5快速排序 11](#_Toc15705)

[4用户手册及测试结果 12](#_Toc30160)

[1、排序演示 13](#_Toc31860)

[2、添加评论 14](#_Toc24800)

[3、 交叉查询演示 15](#_Toc28990)

[4、 访问网页 16](#_Toc21294)

[5、初始化 17](#_Toc20594)

[5、 修改原数据文件 17](#_Toc6874)

[5总结提高 18](#_Toc12982)

[5.1体会与收获 18](#_Toc3284)

[5.2问题与解决 18](#_Toc17559)

[5.3评价 18](#_Toc2406)

[6参考文献 19](#_Toc31581)

# 1需求分析

## 1.1功能概述

### 1.1.1基础功能

本程序需要实现一个电影检索系统，拥有以下功能：

1、设计实现一个图形界面，能根据不同的分类如豆瓣评分、名著改编、地域、剧情、导演、演员、语言查询到想要的电影，并且各种分类之间要有交叉。

2、设计实现用户可以通过输入电影名，直接查询电影

3、可以按照最新、最热排行查询。

### 1.1.2拓展功能

1、可以进一步完成模糊查找，不用输入全部的电影名就可以查找到电影。

2、可以让用户添加评语

3、可以点击访问外部链接

## 1.2需要处理的数据

针对三个基础功能，对于功能一需要有电影的评分、是否改编、地域、剧情、导演、演员、语言这几种数据，其中剧情和演员包含多个数据。

对于功能二需要有电影名

对于功能三需要有电影的点击率和电影的上映时间

对于功能四添加了评语

对于功能五添加了外部链接

电影文件存储于txt文件中，电影的热度、评论、网页链接为了修改方便各存在单独的txt文件中。其中存储电影的txt文件中含有250部电影，为豆瓣评分前250名。

其中，地域、剧情、语言、改编四项通过按钮点击进行分类查找，演员、导演、电影名通过一个三选一按钮进行搜索，考虑到演员、导演、电影名的数据量巨大，并未像地域语言这种做成按钮点击。

## 1.3程序开发运行环境

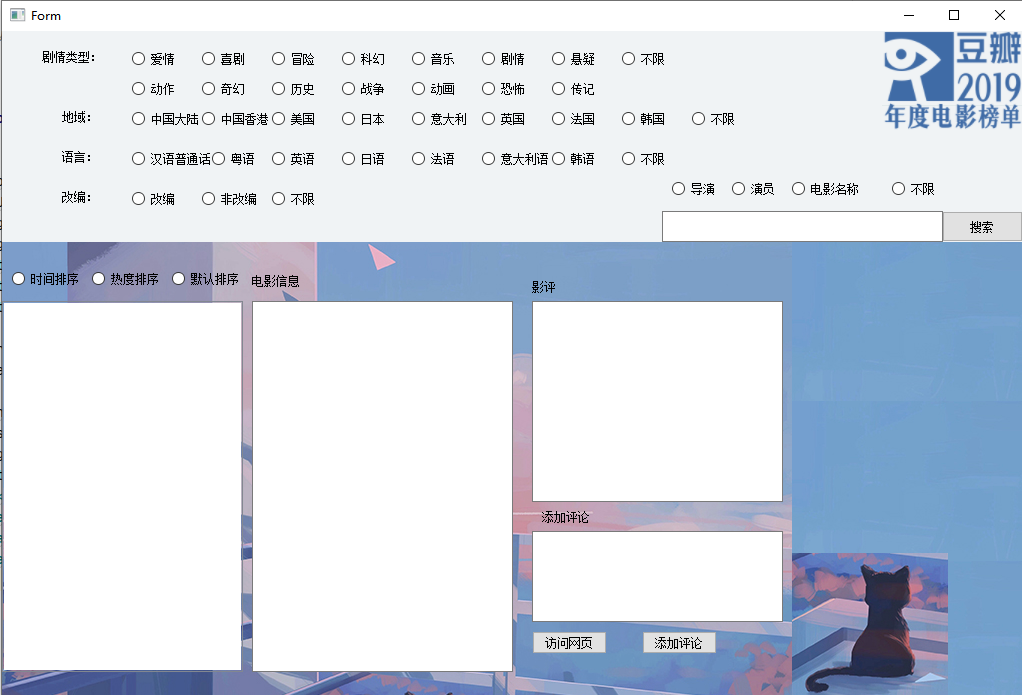
开发环境：

PyCharm Community Edition

运行环境：

* Windows 10 版本 1507 或更高版本：家庭版、专业版、教育版和企业版
* Windows Server 2016：Standard 和 Datacenter
* Windows 8.1：核心板、专业版和企业版
* Windows Server 2012 R2：Essentials、Standard、Datacenter
* Windows 7 SP1：家庭高级版、专业版、企业版、旗舰版

## 1.4用户界面设计



在基础功能1的查找中，将剧情类型、地域、语言、改编这四大类做成了按钮，对导演、演员、电影名称的搜索做出了改变，放置了一个三选一的按钮，可以在导演搜索、演员搜索电影名称搜索之间切换。

在按钮下方，是三块显示部分，最左边为搜索结果，搜索结果可以通过上方按钮进行不同的排序；中间是电影信息，显示了电影的各种信息；最右侧是评论部分，可以在下面输入评论添加进去，并且可以通过访问网页按钮访问网页链接。最右侧的上方是显示图片的部分。

# 2数据结构设计

## 2.1主要数据结构定义

### 2.1.1逻辑结构

电影检索系统中的每一个电影，两两之间并没有什么逻辑关系，因此选择的是线性结构。

### 2.1.2存储结构

电影存储于链表中，选择链表的原因是逻辑简单、操作方便（差删等等），但是在后续中发现链表并不是最优的选择，因为并没有过多的差删操作，更多的访问操作，因此顺序表会更好一点，不过针对链表，在一些算法上做了优化。

为了方便进行快速排序，选择的双向链表。

## 2.2整体结构及模块功能描述

改电影查询系统又电影节点类和电影链表类组成，节点类主要用于存储数据和对数据操作，链表类主要用于实现各种功能。

## 2.3各模块功能描述

### 2.3.1电影节点类（movie）

链表中的节点，负责储存电影的信息。

数据：电影名（name），豆瓣评分（score），名著改编（original），地域（country），语言（language），导演（director），演员（actors），剧情（plot）。

此外为了实现另外两个基础功能，又加入了热度（hot），上架时间（time），两个数据，为了方便后续排序、热度文件写入，又加入了number（按链表顺序从0到n，方便快速排序指针移动，判断指针重合），filenumber（按原电影文件顺序从0-n，用于热度写入）。

为了实现拓展功能，后续又加入了评论（pinglun[]），网页链接（url）

功能 ：

1、交换两个节点数值（用于排序）

2、更新热度

3、添加评论

### 2.3.2电影链表类（Movielist）

链表本体

功能：

1. 按类型查找，各类有交叉
2. 按照时间、热度、默认排序

## 2.4核心算法流程图

输入排序类型

按时间、热度、默认排序

输出查询结果

判断是否继续查询

访问电影，热度+1

结束

开始

从文件读取电影信息

将结果存入查询链表

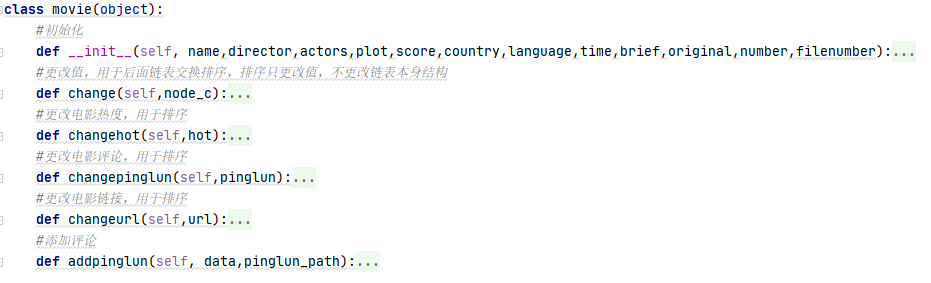
输入插叙类别

判断是否继续查询

# 3详细设计

## 3.1各模块设计

### 3.1.1 movie



### 3.1.2 Movielist



## 3.2各模块核心函数

### 1.1、交换值函数

#更改值，用于后面链表交换排序，排序只更改值，不更改链表本身结构

def change(self,node\_c):

self.name = node\_c.name

self.score = node\_c.score

self.country = node\_c.country

self.language = node\_c.language

self.director = node\_c.director

self.actors = node\_c.actors

self.plot = node\_c.plot

self.original = node\_c.original # 改编

self.brief = node\_c.brief # 剧情简介

self.hot = node\_c.hot # 记录点击量

self.time = node\_c.time

self.filenumber = node\_c.filenumber

self.pinglun = node\_c.pinglun

self.url = node\_c.url

#更改电影热度，用于排序

def changehot(self,hot):

self.hot = hot

#更改电影评论，用于排序

def changepinglun(self,pinglun):

self.pinglun = pinglun

#更改电影链接，用于排序

def changeurl(self,url):

self.url = url

该类函数作用是在快速排序和搜索电影功能中。用于快速排序是以为对于链表，交换值的效率要比交换链表的节点效率要高。用于搜索电影是因为每次搜索都会把结果存入一个链表中，下一次搜索会从这个新表中搜索，在创建新表时要用其赋值。

### 1.2添加评论函数

def addpinglun(self, data,pinglun\_path):

if self.pinglun == '':

self.pinglun = list(self.pinglun)

self.pinglun.append(data)

f = open(pinglun\_path, 'r+',encoding='utf-8')

flist = f.readlines()

#newpinglun = str(self.pinglun)

flist[self.filenumber] = data + "/" +flist[self.filenumber]

f = open(pinglun\_path, 'w+',encoding='utf-8')

f.writelines(flist)

f.close()

该函数的作用是添加评论到节点的pinglun[]中，并将pinglun[]写入txt文件中

### 2.1判断链表是否为空

def is\_empty(self):

if self.header == None:

return True

else:

return False

判断链表是否为空，用于插入节点

### 2.2清空链表

def clearlist(self):

self.header = None

self.tail = None

self.length = 0

清空删除表，用于在查询结束后删除查询结果

### 2.3访问节点

def fangwen(self,node,hot\_path):

node.hot = node.hot +1

f = open(hot\_path, 'r+')

flist = f.readlines()

newhot = str(node.hot)

flist[node.filenumber] = newhot+'\n'

f = open(hot\_path, 'w+')

f.writelines(flist)

f.close()

访问、点击电影，热度+1，存入储存热度的文件

### 2.4尾插入

def addend(self,node):

current\_node = self.header

if self.is\_empty():

self.header = node

self.tail = node

else:

while current\_node.next != None:

current\_node=current\_node.next

current\_node.next = node

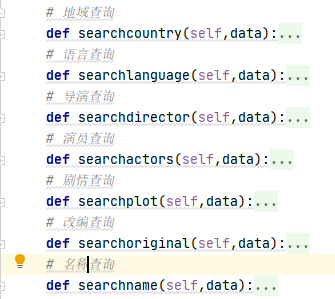
node.before = current\_node

self.tail = node

self.length += 1

在链表结尾插入节点

### 2.4遍历搜索



def searchcountry(self,data):

current\_Node = self.header # 头指针，用于遍历当前链表

empty\_list = MovieList() # 空表，存储这次检索出的信息

#data = input("请输入剧情地域:")

len = 0

while len <= self.length:

if data in current\_Node.country:

# node用于暂存符合检索条件的current\_node的信息，但是其next=None

node = movie(current\_Node.name, current\_Node.director, current\_Node.actors, current\_Node.plot,

current\_Node.score, current\_Node.country, current\_Node.language, current\_Node.time,

current\_Node.brief, current\_Node.original,current\_Node.number,current\_Node.filenumber)

node.changehot(current\_Node.hot)

node.changepinglun(current\_Node.pinglun)

node.changeurl(current\_Node.url)

empty\_list.addend(node) # 将符合检索信息的节点加入空链表

current\_Node = current\_Node.next # 检索当前表下一个节点

else:

current\_Node = current\_Node.next # 检索当前表下一个节点

len += 1

print("查询结束")

self.length = empty\_list.length

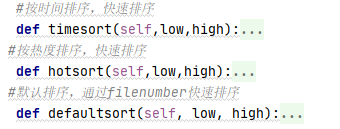
self.header = empty\_list.header

self.tail = empty\_list.tail

遍历搜索包括遍历搜索地域、语言、导演、演员、剧情、名字、是否改编，每个搜索函数的机构类似，只是更改了搜索的数据类型。

此处以遍历搜索地域为例子。其逻辑为遍历链表，当当前节点的值等于搜索的Data值时，将改节点的值复制到新链表中，并在最后将这个新链表代替当前链表进行下一次的搜索。

### 2.5快速排序



def timesort(self,low,high):

key = movie('abc', 'abc', 'abc', 'abc', 'abc', 'abc', 'abc', 'abc', 'abc','no',-1,-1)

if not low or not low.next:

return

if low.number != high.number:

p=low

q=high

key.change(p)

while p.number != q.number:

while p.number != q.number and q.time <= key.time:

q = q.before

p.change(q)

while p.number != q.number and p.time >= key.time:

p = p.next

q.change(p)

q.change(key)

if low.time != p.time:

self.timesort(low,p.before)

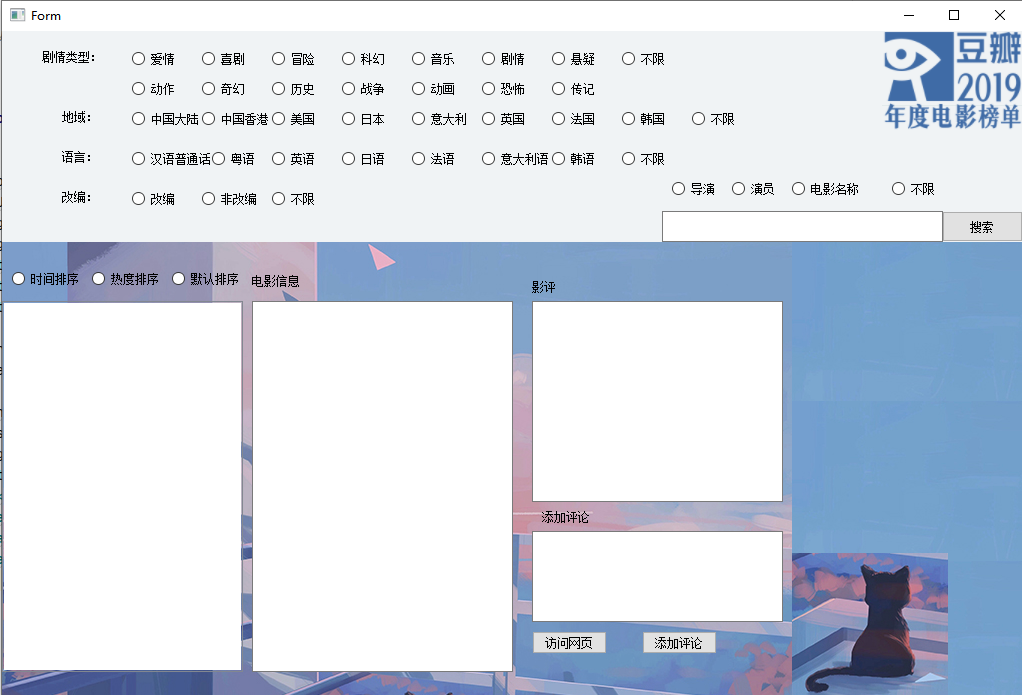
if p.time != high.time:

self.timesort(p.next,high)

排序分为时间排序、热度排序、默认排序，采用的都是快速排序的思路（O (nlogn)）。两个指针，p是头、q是尾，在交换节点时，不对链表本身进行改动，只是交换数据（调用节点中的change（）函数），这样稍微弥补了一些链表本身交换位置效率低的缺陷。

热度排序与默认排序与时间排序结构相同，只是排序所根据的值换成了hot和filenumber（文件中的顺序号）

# **4用户手册及测试结果**

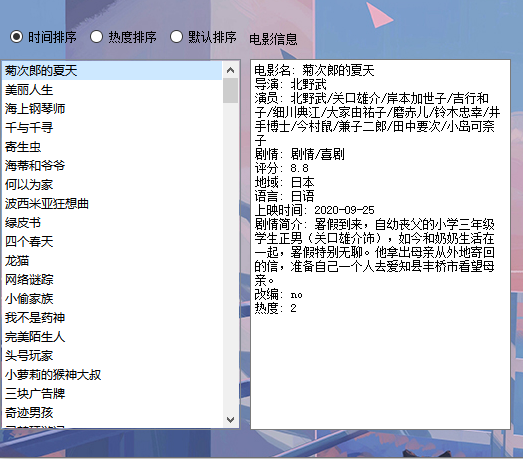


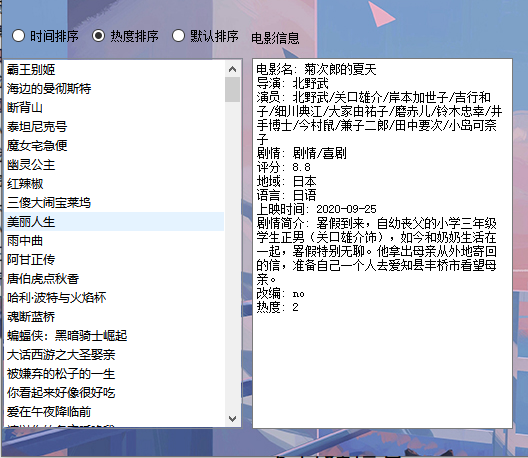
全部选择不限，再点击搜索，可以看到全部250部电影的信息，并可以点击查看详情



## 1、排序演示

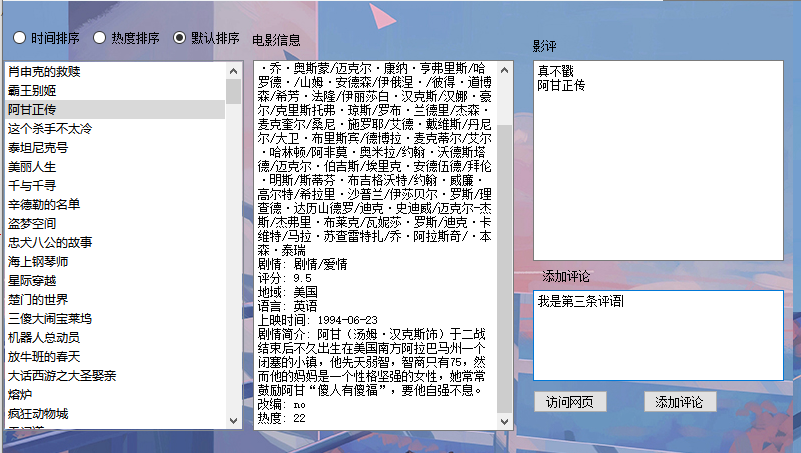
在未选择排序时，默认为默认排序，可以在左侧点击切换排序。



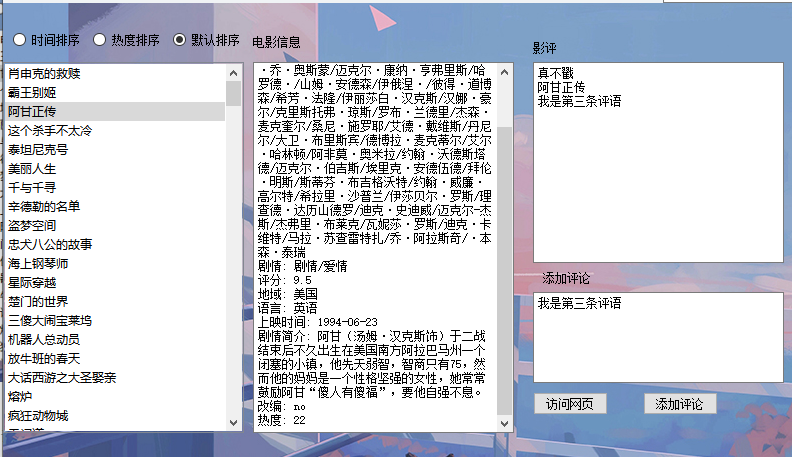


## 2、添加评论

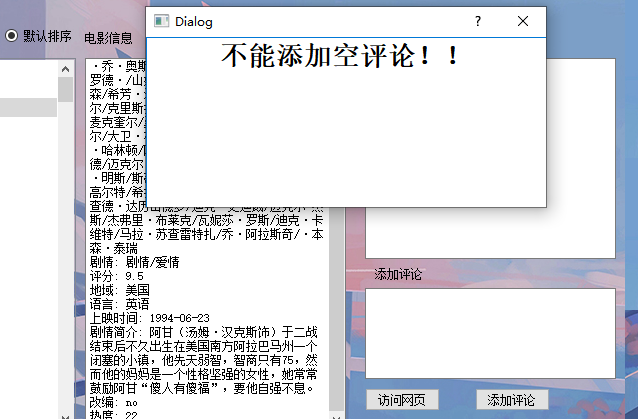
之后可以为阿甘正传添加评语，在右下角添加评论框中输入评语



点击添加，成功显示

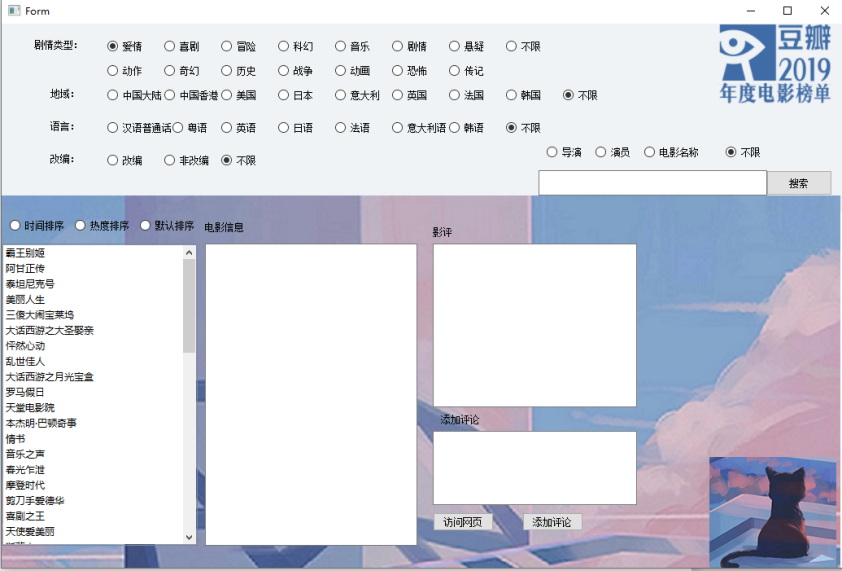


当评论为空时，会提示错误

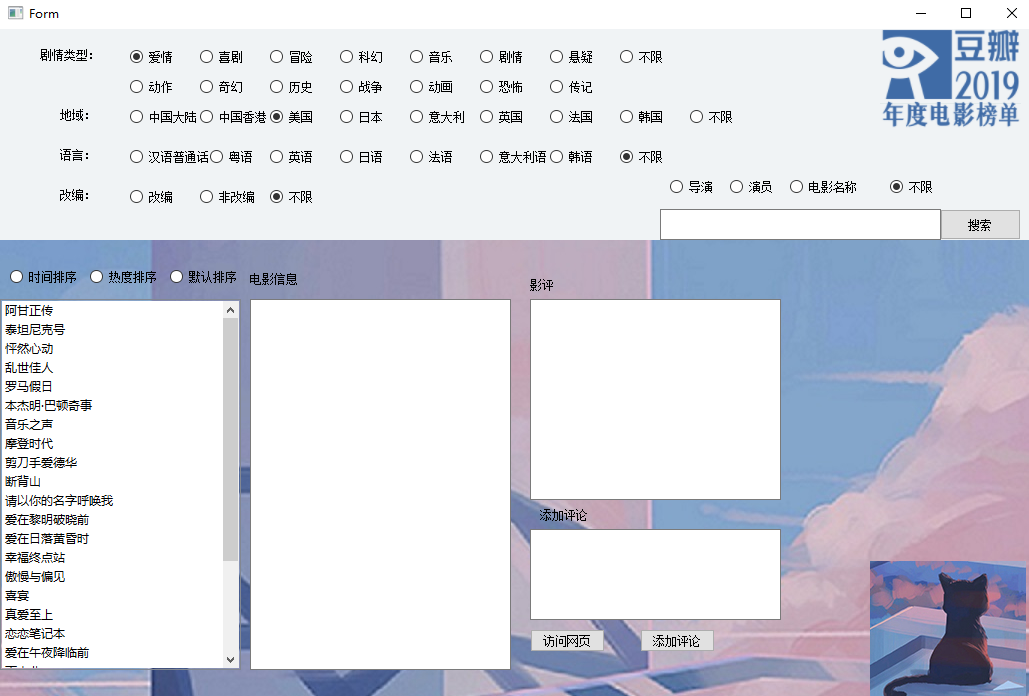


## 交叉查询演示

点击爱情，搜索，在左侧显示了全部爱情类型的电影。



进一步选择美国，语言，非改编，可以看到搜索的范围变小了

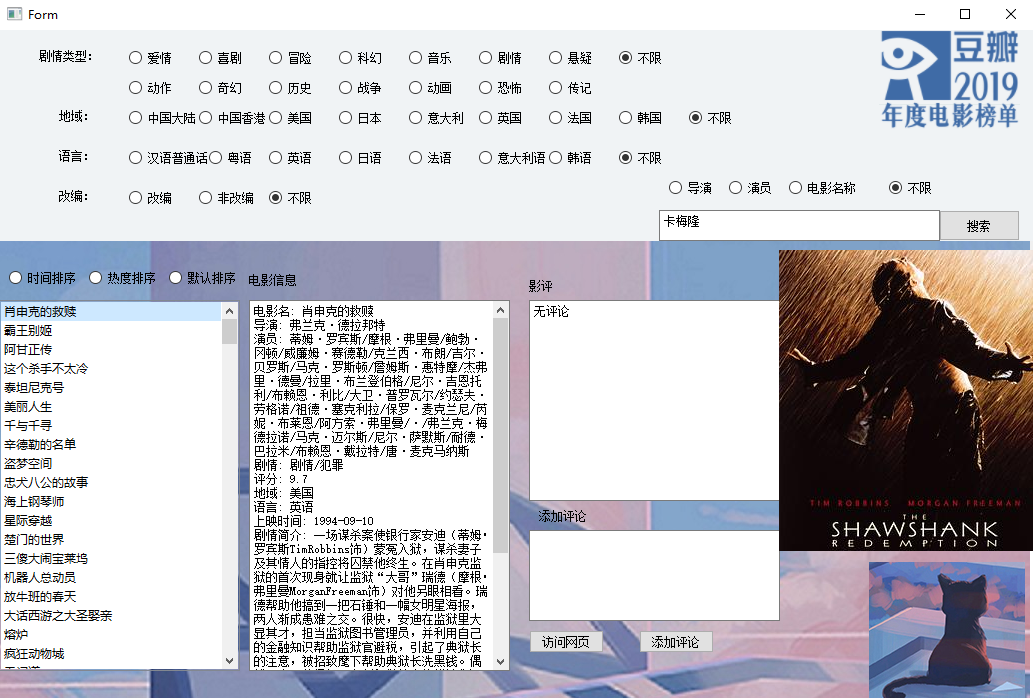


在进一步缩小范围，选择导演，输入卡梅隆，就只剩下泰坦尼克号了



## 访问网页

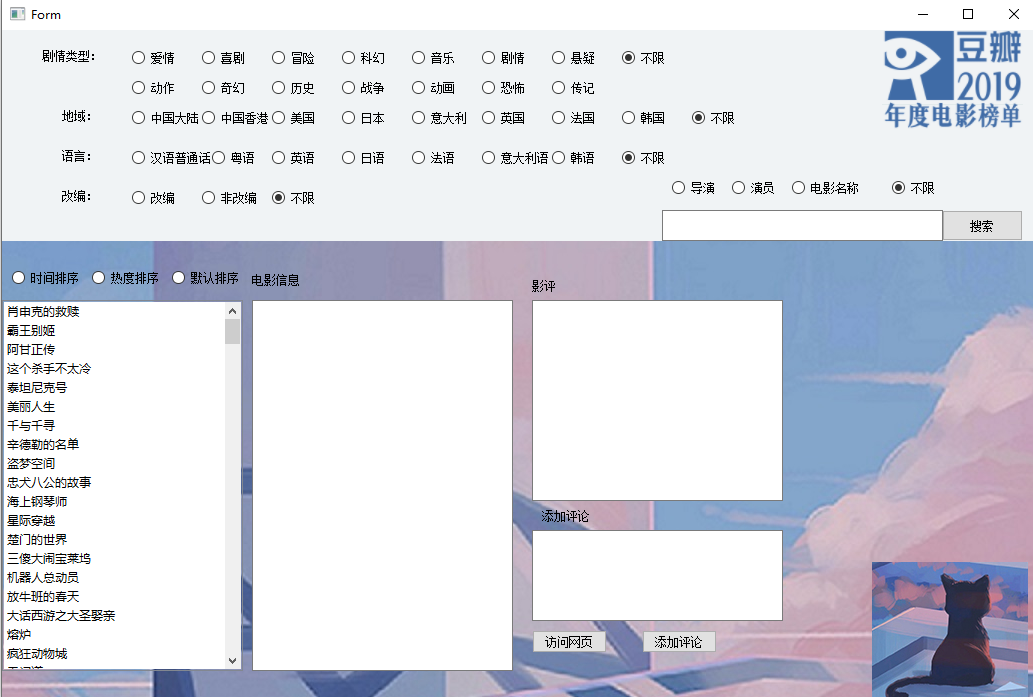
点击访问网页，会自动打开浏览器，弹出该电影的豆瓣网页，可以通过豆瓣观看电影的部分画面（很多电影并没有免费的观看网页，因此统一选择了豆瓣）。



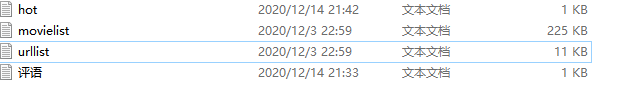


## 5、初始化

全部点击不限，并在此点击搜索，就可以重新搜索（并未找到重置按钮的方法）



## 修改原数据文件





hot.txt 存储电影的热度

urllist.txt 存储电影的网页链接

评语.txt 存储电影的评语

movielist.txt 存储电影的主要数据（不包含热度、网页链接、评语）

热度、网页链接、评语单独存储的目的是方便修改，因为修改数据时是采用全部覆盖的形式，分开存储可以更加效率。

Img文件夹存储图片

# **5总结提高**

同学依据自己的经历写出个性化的总结。内容可以包括：程序开发中的体会与收获，开发中遇到的问题与解决情况，自己对自己完成课设情况的评价等等。

## 5.1体会与收获

每次课设的制作都让我收获很多，他能让我重新巩固以前学习的知识，更加熟练的运用编程语言，尤其是这次使用了以前从未使用的python，对于我来说还是有着很大的挑战的。在最开始的阶段，选择哪种数据结构、选择那些算法才能使效率变高让我苦恼不已，这使得我畏手畏脚，迟迟不敢动手，直到后来检查的时间越来越近，才逼迫自己采用最简单、最容易想到的方法先去入手做数据结构的部分，因此选择了思路简单的链表和遍历搜索。

但是这在后面的制作中为我添了很大的麻烦，比如在排序时，链表的快速排序，尤其是单链表的快速排序相比顺序表更复杂、效率也低，此时很多一开始和我思路相同的同学改用了顺序表的存储结构，而我认为通过链表实现快速排序，也是一种练习，因此继续坚持使用了链表，不过还是做出了改进，改为了双向链表。在与其他同学的交流中，我也了解到了树的存储结构的优缺点，可以说通过这次课设，让我对存储结构有了更深刻的认识和体会。

另一个收获就是让我学会了使用python，以前一直听说python是很方便很容易学习理解的语言，并且功能强大，但是一直没有一个时机去学习它，这次课设给我提供了这个时机。虽然在过程中因为不了解python绕了很多弯路（比如python没有指针），但是当到了后期，我真的体会到了python的便利和强大。

最后一个收获就是让我体会到了编程的魅力，以前对编程一直非常抵触，甚至一度因为不想接触编程而想放弃走IT这条路，但是在这次课设中，每当解决一个问题后，所获得的那种巨大的成就感，让我感受到了编程的魅力，这将鼓励我、支持我继续走下去。

## 5.2问题与解决

这次遇到的问题主要有两个方面

一是对python的不熟悉，让我受了很多苦，比如我一开始并不会使用python中的self，再比如没有指针使得我对链表的操作、以及数据更改上遇到了很大的问题，最让我困扰的就是python和c完全是两种存储方式，比如同样的a=1，这两种语言的逻辑完全不同。克服的办法就是在网上搜索解决办法，更深入了解python，实在不行就用些笨办法，比如电影每次重新搜索我都是将数据全部导入一遍。

二是图形界面，python的图形界面用的是pyqt5，学习使用pyqt5和Qtdesiner就花了不少的时间。在制作的过程中有时候因为用错了某个功能，让我不得不推翻从做，总之在制作界面时花费了超过制作功能模块的时间。

## 5.3评价

在界面的美化上也不够精致，应该吧电影图片和电影名字一起在搜索结果中显示。

# **6参考文献**

1. 《a byte of python》 Swaroop C H

2、《数据结构与算法》（张铭版）

3、PyQt5，弹窗，自定义输入对话框

<https://blog.csdn.net/qq_39315153/article/details/88241517?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.control>

4、PyQt5-按钮控件使用

<https://www.cnblogs.com/ygzhaof/p/10059540.html>

5、QT designer的安装与汉化（pycharm）

https://blog.csdn.net/qq\_36374896/article/details/83513828