《Spark大数据运算》

项目报告

**项目制作人：Fan, Flemming Linmao**

**日 期：2019/07/01**

*Spark*项目报告

**摘　要**

本项目主要实现了电影评分可视化，将大众用户对电影的评分评价以图标的形式展现。本次项目采用SpringBoot+Spark+Vue架构，以MySQL为数据库，建立基于基于B/S模式的电影评分可视化平台。我们将数据上传至HDFS，基于RDD实现对movies.csv数据文件的过滤，用户在此平台上可以通过输入电影名称，电影类型进行精确查找或者模糊查找，同时了解电影的所有用户评分，用户评论情况。基于Spark Streaming技术，动态增加打分文件，实现将最新的统计数据用折线图的方式展现到页面上，并能看到折线图实时变化，让用户以图的形式感受数据和数据的实时变化，更加直观。我们将数据导入Mysql，通过Spark SQL调用访问Mysql，按性别、电影的类型的平均分进行数据展示，将男女性别对同类电影的平均分进行对比，采用柱状图显示在网页上。让用户从宏观上了解电影，可以更清楚自己的观影倾向。

**关键词：Mysql**；SpringBoot；Spark；Vue;

Spark project summary

This project realizes the visualization of film rating, showing the public users' rating and evaluation of films in the form of graphics. This project adopts the Spring Boot+Spark+Vue architecture and takes MySQL as the database to establish a visual platform for film scoring based on B/S mode. We uploaded the data to HDFS, and realized the filtering of movies.csv data files based on RDD. On this platform, users can conduct accurate or fuzzy lookup by inputting movie name and movie type, and at the same time understand all user ratings and user comments of the movie. Based on Spark Streaming technology, score files are added dynamically to show the latest statistics to the page by means of broken line graph, and real-time changes of broken line graph can be seen, allowing users to feel the data and real-time changes of data in the form of graph, which is more intuitive. We imported the data into Mysql, accessed Mysql through Spark SQL, and displayed the data according to the average scores of gender and movie types, compared the average scores of male and female against similar movies, and displayed them on the web page by bar chart. Let the user understand the movie from the macro, can be more clear about their own viewing tendency.

**Key words:** Mysql；SpringBoot；Spark；Vue;

目录

[1实验题目 4](#_Toc13513860)

[**1.1** **实验题目** 4](#_Toc13513861)

[**1.1.1** **实验数据** 4](#_Toc13513862)

[**1.1.2** **实验内容** 4](#_Toc13513863)

[**1.1.3** **实验要求** 4](#_Toc13513864)

[**1.2** **小组成员和分工** 5](#_Toc13513865)

[**1.3** **研究方法** 5](#_Toc13513866)

[2开发工具和运行环境 5](#_Toc13513867)

[**2.1** **开发工具** 5](#_Toc13513868)

[**2.2** **运行环境** 6](#_Toc13513869)

[3技术选型说明、框架说明 6](#_Toc13513870)

[**3.1** **技术选型说明** 6](#_Toc13513871)

[**3.2** **框架选型** 7](#_Toc13513872)

[4数据库设计 8](#_Toc13513873)

[**4.1** **temp\_analysis表E-R图** 9](#_Toc13513874)

[**4.2** **temp\_analysis表结构设计** 9](#_Toc13513875)

[**4.3** **temp\_analysis表数据说明** 9](#_Toc13513876)

[5数据分析、处理过程 9](#_Toc13513877)

[**5.1** **Spark RDD** 9](#_Toc13513878)

[**5.2** **Spark Streaming** 12](#_Toc13513879)

[**5.3** **Spark SQL** 13](#_Toc13513880)

[6页面展示 16](#_Toc13513881)

[**6.1** **Spark RDD** 16](#_Toc13513882)

[**6.2** **Spark Streaming** 16](#_Toc13513883)

[**6.3** **Spark SQL** 18](#_Toc13513884)

[结 束 语 ……………………………………………………………………………………20](#_Toc13513885)

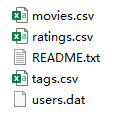
[参考文献……………………………………………………………………………………..21](#_Toc13513886)

[致谢…………………………………………………………………………………………..22](#_Toc13513887)

# **实验题目**

## **实验题目**

### **实验数据**



### **实验内容**

1） 将数据上传至HDFS，基于RDD实现对movies.csv数据文件的过滤，用户可能通过在网页上输入关键字，实现对文件进行过滤（前匹配、后匹配、模糊过滤），将结果展示在页面上。

加分项:在上述基础之上，与其他文件进行结合，形成更多、更灵活的数据过滤方案的实现

2） 基于Spark Streaming技术，动态增加打分文件，实现将最新的统计数据用拆线图的方式展现到页面上，并能看到折线图实时变化。

加分项：能够通过界面进行打分文件的动态生成。

3） 将数据导入Mysql，通过Spark SQL调用访问Mysql，按性别、电影的类型的平均分进行数据展示，将男女性别对同类电影的平均分进行对比，采用柱状图显示在网页上。

加分项：可通过诸如年龄段、职业、地区等方面进行统计和展示

### **实验要求**

使用Web开发技术（后台技术不限，可用JavaEE、.Net、PHP等均可），将页面、Spark、Hadoop相关技术进行整合，形成整体框架，可以进行传值和调用。(该题目只限定大方向，可根据题目大意适当添加自己的一些想法)

## **研究方法**

本次实验是以MySQL为数据库，在Spark +SpringBoot+Vue架构下建立的Web应用。主要在Intellij IEDA下进行开发，编程语言是Java语言。

# **开发工具和运行环境**

## **开发工具**

1) 系统开发工具如表2‑1所示。

表2‑1 开发工具表

|  |  |
| --- | --- |
| 开发工具名称 | 软件用途 |
| Eclipse | 开发平台 |
| Intellij IDEA | 开发平台 |
| Tomcat | 服务器 |
| Microsoft Visio 2003 | 系统设计，创建流程图等 |
| Microsoft Word 2013 | 文档编写 |
| MySQL | 系统后台数据库 |
| Spark | 开发框架 |
| JDK | 开发工具 |
| vue | 前端框架 |
| ECharts | 前端js库 |

## **运行环境**

2）运行环境如表2-2所示

表 3 运行环境

|  |  |
| --- | --- |
| 运行环境名称 | 用途 |
| Linux | 系统环境 |
| Hadoop | Hadoop集群环境、提供HDFS文件系统 |

# **技术选型说明、框架说明**

## **技术选型说明**

1）Java开发。

Java语言历时十多年，已发展成为人类计算机史上影响深远的编程语言，从某种程度上来看，它甚至超出了编程语言的范畴，成为一种开发平台，一种开发规范。更甚至于：Java已成为一种信仰，Java语言所崇尚的开源、自由等精神，吸引了全世界无数优秀的程序员。事实是，从人类有史以来，从来没有一门编程语言能吸引这么多的程序员，也没有一门编程语言能衍生出如此之多的开源框架。

Java语言是一门非常纯粹的面向对象编程语言，它吸收了C++语言的各种优点，又摒弃了C++里难以理解的多继承、指针等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，很好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程开发。

不仅如此，Java语言相关的Java EE规范里包含了时下最流行的各种软件工程理念，各种先进的设计思想总能在Java EE规范、平台以及相关框架里找到相应实现。从某种程度上来看，学精了Java语言的相关方面，相当于系统地学习了软件开发相关知识，而不是仅仅学完了一门编程语言。

时至今日，大部分银行、电信、证券、电子商务、电子政务等系统或者已经采用Java EE平台构建，或者正在逐渐过渡到采用Java EE平台来构建，Java EE规范是目前最成熟的，也是应用最广的企业级应用开发规范。

选用Java是因为较为熟悉Java，使用方便。

2）MySQL：

MySQL是一个[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "_blank)，由瑞典MySQL AB 公司开发，目前属于 [Oracle](https://baike.baidu.com/item/Oracle) 旗下产品。MySQL 是最流行的[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "_blank)之一，在 WEB 应用方面，MySQL是最好的 [RDBMS](https://baike.baidu.com/item/RDBMS/1048260) (Relational Database Management System，关系数据库管理系统) 应用软件之一。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是[开放源码](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E7%A0%81/7176422" \t "_blank)这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

3) JDBC：

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用[Java语言](https://baike.baidu.com/item/Java%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/_blank)编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序，同时，JDBC也是个商标名。

选择JDBC是为了Spark同MySQL数据库的交互。

## **框架选型**

1）SpringBoot:

SpringBoot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者。

2）Spark:

为了解决MapReduce及各种处理框架所带来的问题，加州大学伯克利分校推出了Spark统一大数据处理框架。Spark是一种与Hadoop MapReduce类似的开源集群大数据计算分析框架。Spark基于内存计算，整合了内存计算的单元，所以相对于Hadoop的集群处理方法，Spark在性能方面更具优势。Spark启用了弹性内存分布式数据集，除了能够提供交互式查询外，还可以优化迭代工作负载。

从另一角度来看，Spark可以看作MapReduce的一种扩展。MapReduce之所以不擅长迭代式、交互式和流式的计算工作，主要因为它缺乏在计算的各个阶段进行有效的资源共享，针对这一点，Spark创造性地引入了RDD（弹性分布式数据集）来解决这个问题。RDD的重要特性之一就是资源共享。Spark基于内存计算，提高了大数据处理的实时性，同时兼具高容错性和可伸缩性，更重要的是，Spark可以部署在大量廉价的硬件之上，形成集群。

选择Spark是为了方便的处理电影评分的大量数据。

3）Vue:

Vue.js是一个用来开发Web界面的前端库，是个非常轻量级的工具。Vue.js本身具有响应式编程和组件化的特点。相比较Angularjs和ReactJS, Vue.js一直以轻量级，易上手被称道。MVVM的开发模式也使前端从原先的DOM操作中解放出来，我们不再需要在维护视图和数据的统一上花大量的时间，只需要关注于data的变化，代码变得更加容易维护。虽然社区和插件并没有一些老牌的开源项目那么丰富，但满足日常的开发是没有问题的。Vue.js 2.0也已经发布了beta版本，渲染层基于一个轻量级的virtual-DOM实现，在大多数场景下初始化渲染速度和内存消耗都提升了2～4倍。而阿里也开源了weex（可以理解成ReactJS-Native和ReacJS的关系），这也意味着Vue.js在移动端有了更多的可能性。

# **数据集及数据库设计**

本次实验所使用的数据库表只有一张，是temp\_analysis表。该DB数据在项目中src/java/resources/db/temp\_analysis.sql。与此同时，还需要用到linux系统的hadoop集群，集群的hdfs上需要有实验题目中所展示的数据集文件。

## **temp\_analysis表E-R图**



图 4-1 temp\_analysis表E-R图

## **temp\_analysis表结构设计**

表 4-2 temp\_analysis表结构设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 是否为空 | 是否为主键 | 备注 |
| gender | text | 0 |  |  | 性别 |
| age | text | 0 |  |  | 年龄 |
| genres | text | 0 |  |  | 电影类型 |
| rating | text | 0 |  |  | 电影评分 |

## **temp\_analysis表数据说明**

temp\_analysis表所存储的数据是利用spark sql读取ratings文件后，处理完成的部分数据保存到数据库。spark sql部分显示这张表的数据。

# **数据分析、处理过程**

## **Spark RDD**

先将movies.csv文件上传到HDFS文件系统中，然后利用Spark上下文读取该文件，得到Java RDD，利用Java8的lamda表达式提取数据，过滤数据，最后返回数据。

实验中主要实现两个功能-电影名称查询、电影类型查询，实验核心代码分两种方式编写。

第一种方式代码：

|  |
| --- |
| /\*\*  \* 过滤查询  \* **@param** filePath 本地文件路径或HDFS路径  \* **@param** movieName 电影名称  \* **@param** movieType 电影类型  \* **@return**  \*/  **public** List<Movie> filterSearch(SparkSession spark, String filePath, String movieName,String movieType) {  List<Movie> list = **new** ArrayList<Movie>();  //用SparkSession直接可以读取csv格式类容，自动识别表头。  spark.read().csv(filePath)  .cache()  .collectAsList()  .forEach(row-> {  **if**(movieType != **null** || !movieType.equals("")) {//如果电影类型名不为空  //同时查询电影名和电影类型  **if**(row.getString(1).contains(movieName) && row.getString(2).contains(movieType)) {  //将过滤的内容添加到list并返回  list.add(**new** Movie(row.getString(0),row.getString(1),row.getString(2)));  }  }  **if**(movieType == **null** || movieType.equals("")) {//如果电影类型为空，则查询全部  **if**(row.getString(1).contains(movieName)) {  list.add(**new** Movie(row.getString(0),row.getString(1),row.getString(2)));  }  }  });    **return** list;    } |

第二种方式代码

|  |
| --- |
| /\*\*  \* 查询指定类型的电影  \* @param filePath  \* @param typename  \* @return  \*/  public List<Movie> findMoviesByTypeNames(JavaSparkContext jsc, String filePath, String typeName) {  JavaRDD<String> textFile = jsc.textFile(filePath).cache();  String firstline = textFile.first();  List<Movie> collect = textFile.filter(line -> !line.contentEquals(firstline))  .map(s -> {  String[] str = s.split(",");  return new Movie(str[0], str[1], str[2]);  })  .filter(movie -> movie.getGenres().contains(typeName))  .collect();  return collect;  }    /\*\*  \* 根据电影名称查询电影  \* @param filePath  \* @param filmName  \* @return  \*/  public List<Movie> findMoviesByFilmName(JavaSparkContext jsc, String filePath, String filmName){  JavaRDD<String> textFile = jsc.textFile(filePath).cache();  String firstline = textFile.first();  //先过滤掉第一行  List<Movie> collect = textFile.filter(line -> !line.contentEquals(firstline))  .map(s -> {  String[] str = s.split(",");  return new Movie(str[0], str[1], str[2]);  })  .filter(movie -> {  String newName = filmName.trim().replaceAll("\\s\*", "").toLowerCase();//将输入的电影名去空格转换为小写匹配  return movie.getTitle().replaceAll("\\s\*", "").toLowerCase().contains(newName);  })  .collect();  return collect;  } |

## **Spark Streaming**

利用 Streaming Context对象 对HDFS的文件夹进行监听，每20秒监听一次，如果有文件上传，则分析计算，然后提取数据，通过Web socket向客户端不断的发送数据，客户端不断的显示数据，并动态显示出来。

实验核心代码如下

|  |
| --- |
| /\*\*  \* 开启Stream任务  \* **@param** server  \* **@param** listenerDirectory 要监听的文件夹  \*/  **public** **void** startStreamTask(StreamingConsumer server, String listenerDirectory) {  streamingContext = **new** JavaStreamingContext(javaSparkContext, Durations.*seconds*(20));  JavaDStream<String> lines = streamingContext.textFileStream(listenerDirectory);  //提取rating  lines.map(line -> line.split(",")[2])  .foreachRDD(rdd -> {  List<String> collect = rdd.collect();  **for** (String d : collect) {  server.sendMessageToCient(d);  }  //rdd.saveAsTextFile("");  });    streamingContext.start();  **try** {  streamingContext.awaitTermination();  streamingContext.close();  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  /\*\*  \* 手动关闭Stream  \*/  **public** **void** destroyStreamTask() {  **if**(streamingContext != **null**) {  streamingContext.stop();    }  } |

## **Spark SQL**

将数据上传到HDFS中，然后通过SparkSession 或者 SparkContext 读取csv文件和dat文件，后台读取多个文件将读取出来的所有数据像数据库一样，联合查询，分析得到想要的数据。将这些数据存入一部分到数据库，最后，从利用Spark SQL从数据库读取这些分析数据，然后提取过滤，返回到客户端

连接查询在数据库建立新表的代码

|  |
| --- |
| /\*\*  \* 将数据源的类容，连接查询后写入数据库中  \* **@param** sparkSession  \*/  **public** **static** **void** joinAfterWriteMySQL(SparkSession sparkSession) {    //随机切片权重比例  **double**[] weights = **new** **double**[] {0.1, 0.4, 0.5};    //获取数据源  Dataset<Row>[] ratingscsv = sparkSession.read().format("csv").option("header", "true").load(*ratings*).randomSplit(weights);  Dataset<Row> usersjdbc = sparkSession.read().jdbc(*commonDBConURL*, "users", *connectionProperties*);  Dataset<Row> moviescsv = sparkSession.read().jdbc(*commonDBConURL*, "movies", *connectionProperties*);    //连接查询得到新的数据集  Dataset<Row> newtable = ratingscsv[0].join(moviescsv,  ratingscsv[0].col("movieId").equalTo(moviescsv.col("movieId")))  .join(usersjdbc,  ratingscsv[0].col("userId").equalTo(usersjdbc.col("userId")))  .select(usersjdbc.col("gender"),usersjdbc.col("age"),moviescsv.col("genres"),ratingscsv[0].col("rating"));  //将新的数据集写入本地数据库中  newtable.write().jdbc(*commonDBConURL*, "temp\_analysis", *connectionProperties*);    } |

连接数据库获取查询数据的核心代码

|  |
| --- |
| /\*\*  \* 获取类型平均值  \* **@return** Map<类型，平均值>  \* **@throws** Exception  \*/  **public** Map<String, Double> catogoryAvg(String gender, String age) **throws** Exception {  String[] types = **new** String[] {"Action",  "Adventure","Animation","Children","Comedy","Crime",  "Documentary","Drama","Fantasy","Film-Noir",  "Horror","Musical","Mystery","Romance","Sci-Fi",  "Thriller","War","Western","(no genres listed)"};    Map<String, Double> map = **new** HashMap<String,Double>();    **if**(!"".equals(gender)) {  gender = "gender = '"+ gender +"' AND ";  }  **if**(!"".equals(age)) {  age = "age = '" + age +"' AND ";  }    String sqlText = "select AVG(rating) as average from temp\_analysis WHERE " + gender + age;  **for** (String type : types) {  String sql = sqlText + "genres LIKE '%"+ type.trim() +"%'";  //Dataset<Row> result = SparkSqlUtils.readMySQL(sql);  ResultSet result = jdbc.doQuery(sql, **null**);//不用预编译  **while**(result.next()) {  map.put(type, result.getDouble(1));  }  }  jdbc.close();  **return** map;  } |

# **页面展示**

## **Spark RDD**

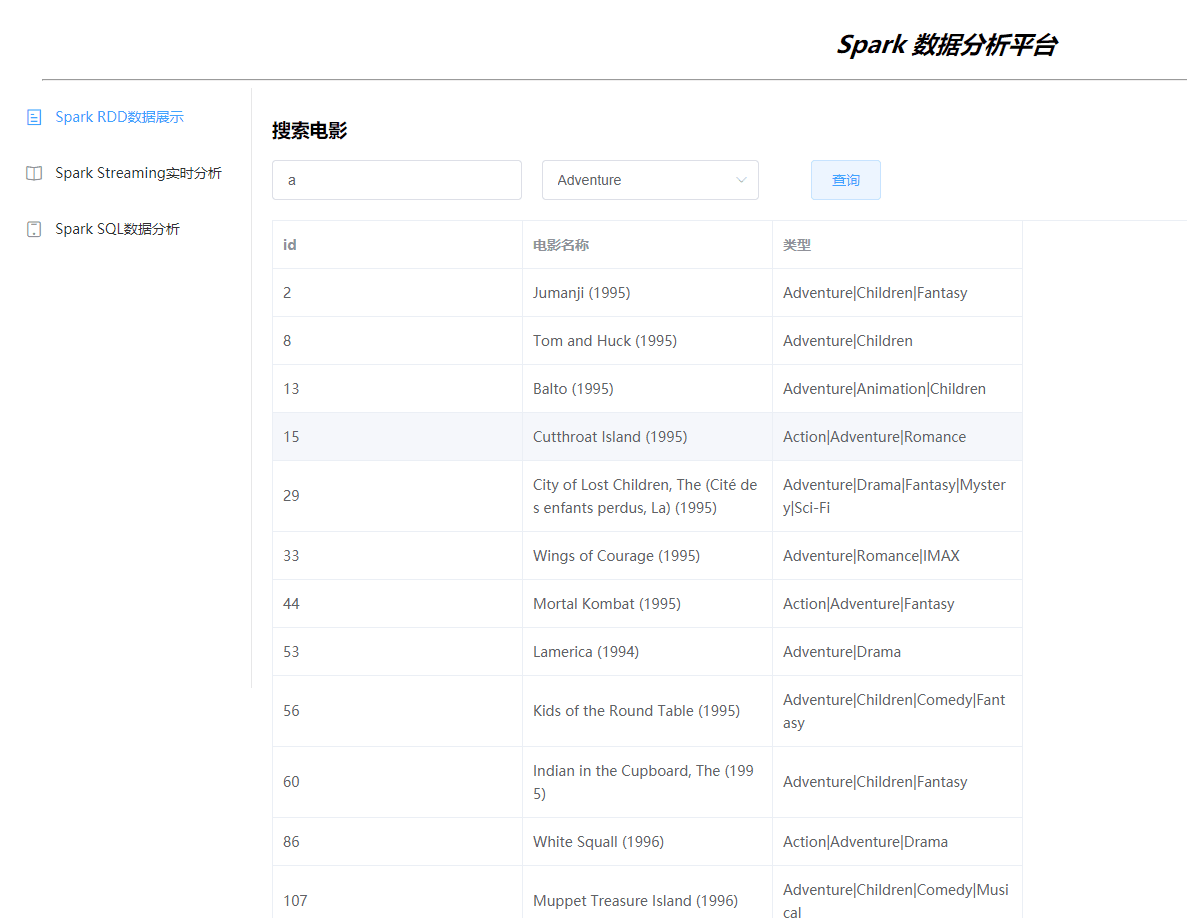


图6-1-1 电影类型查询

## **Spark Streaming**

无数据界面如下图6-2-1所示

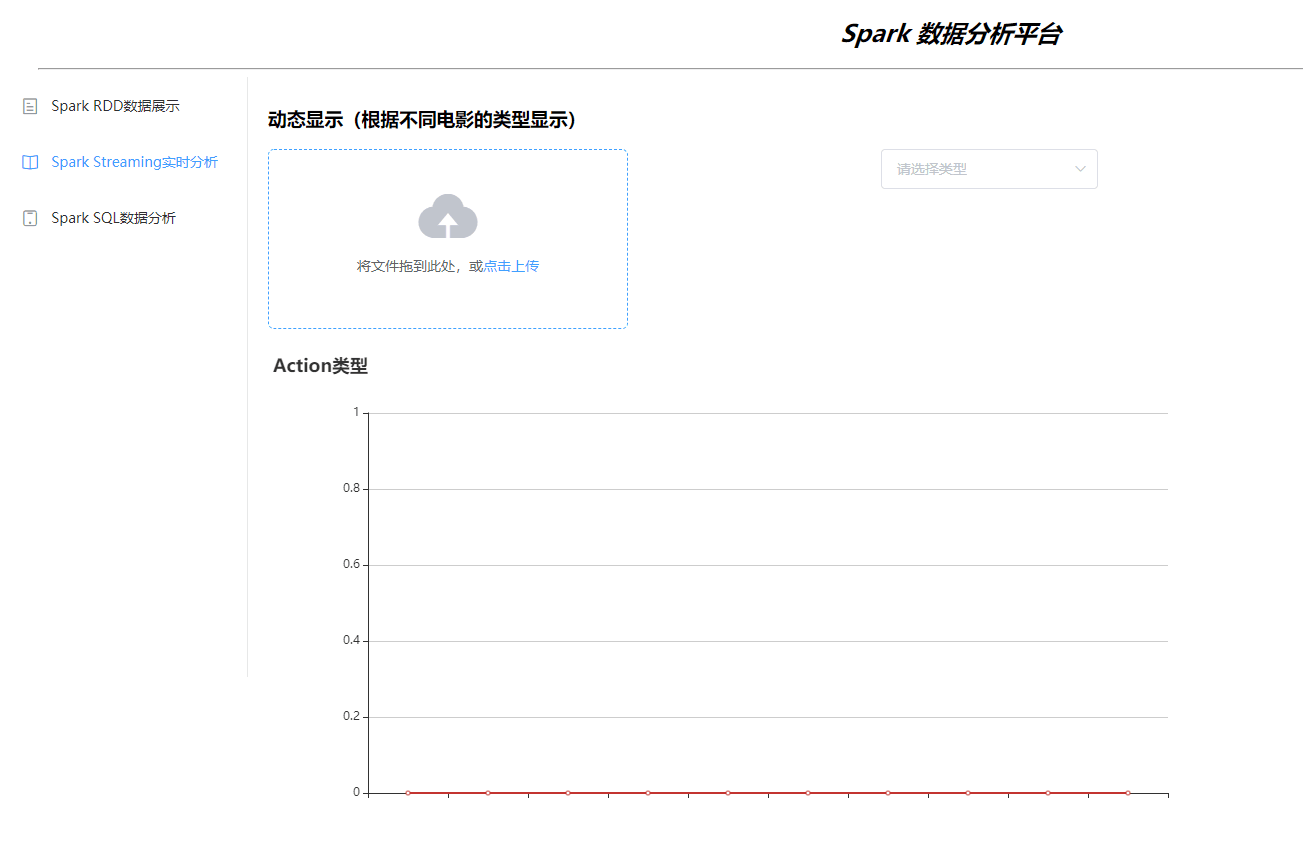


图 6-2-1 无数据页面

上传文件part-0005.txt后折线图发生变化

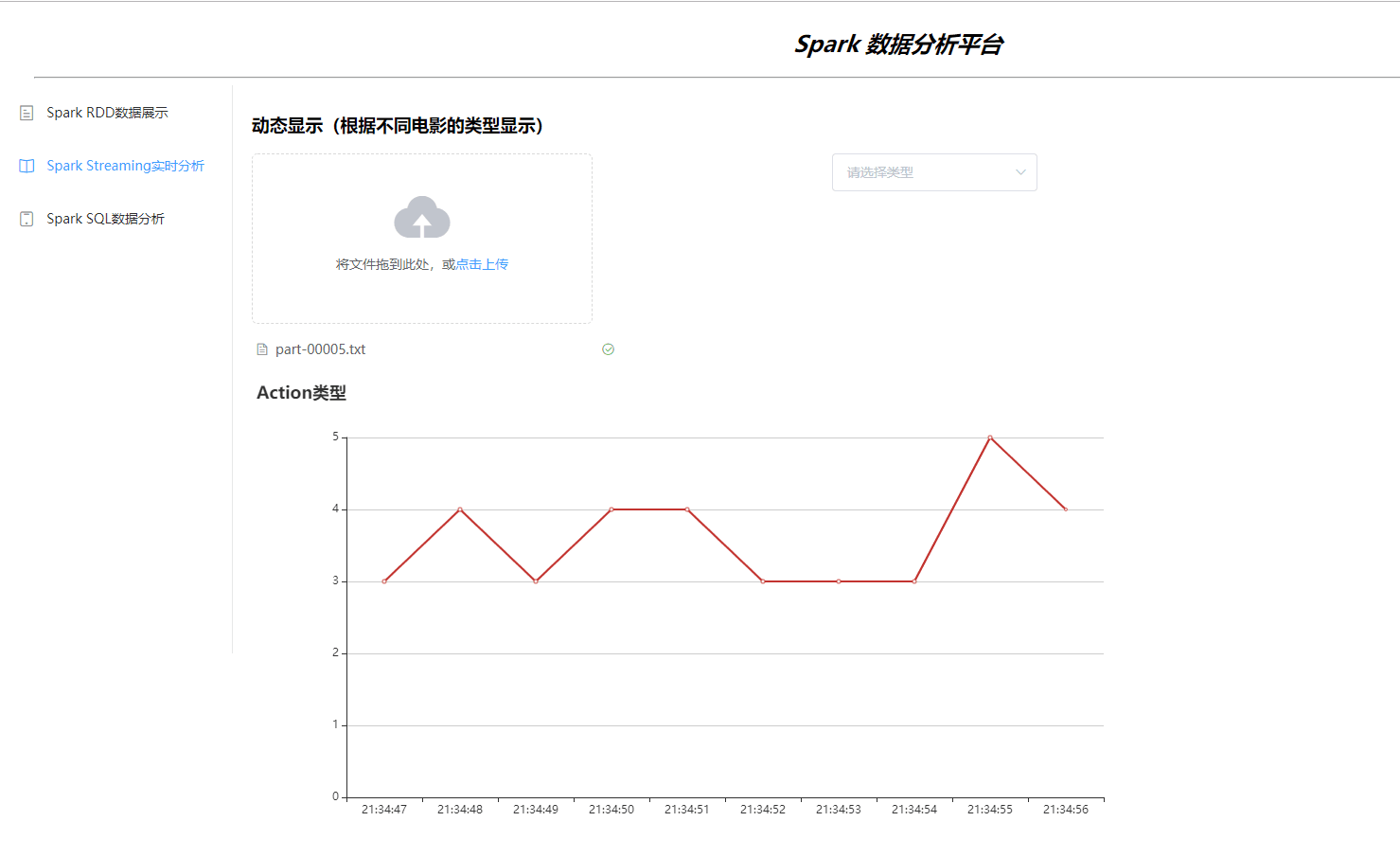


图 6-2-2 实时数据显示1

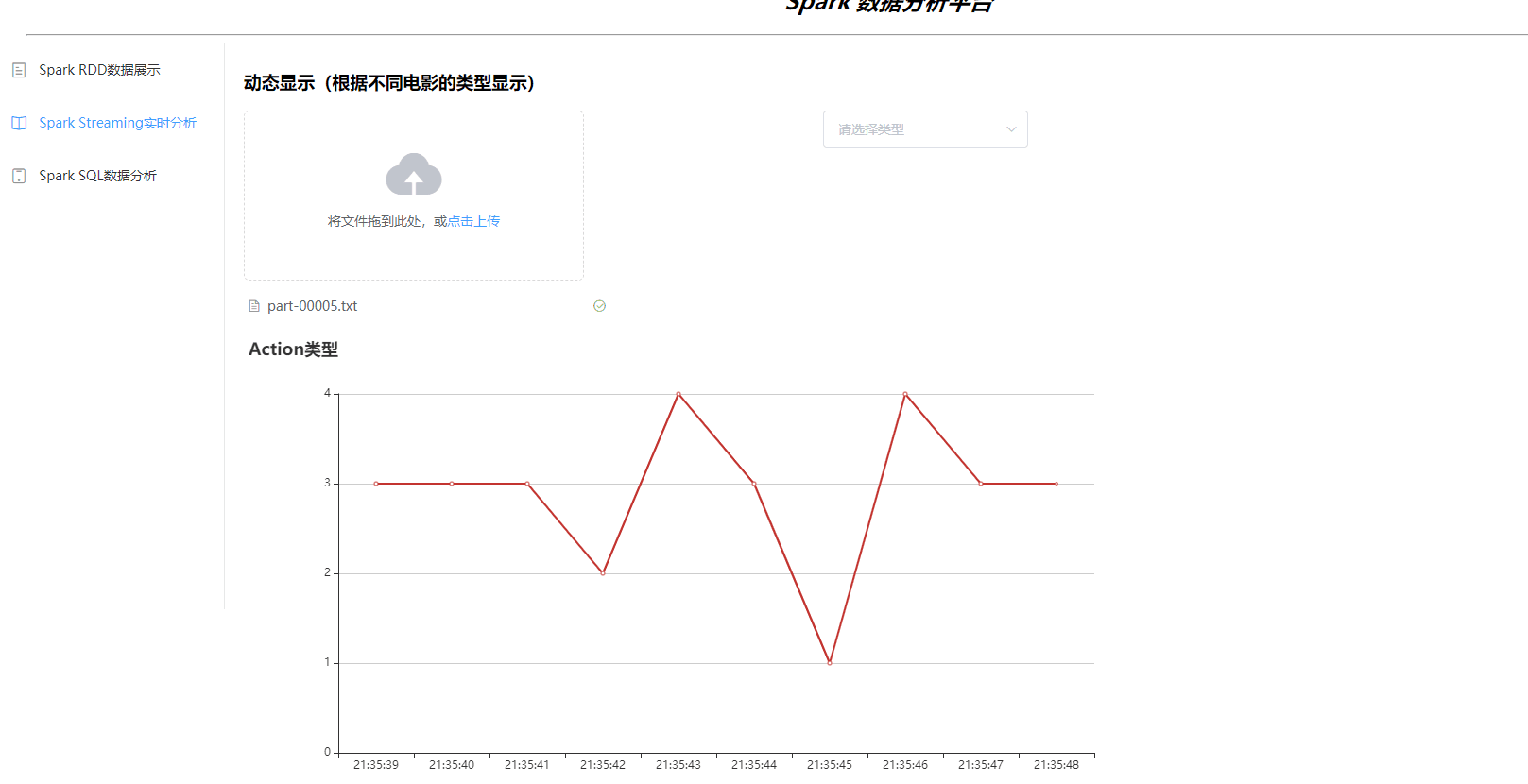


图 6-2-3 实时数据显示2

## **Spark SQL**

筛选后存储的数据存储在temp\_analysis表中

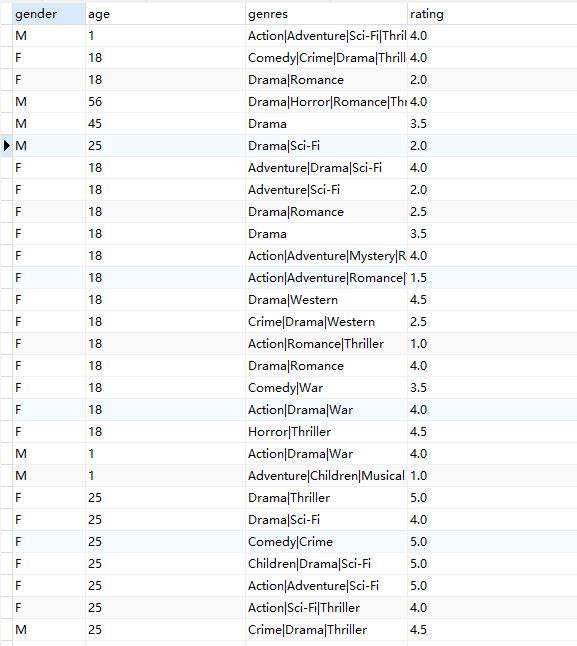


图 6-3-1 temp\_analysis表数据

实验结果显示如图

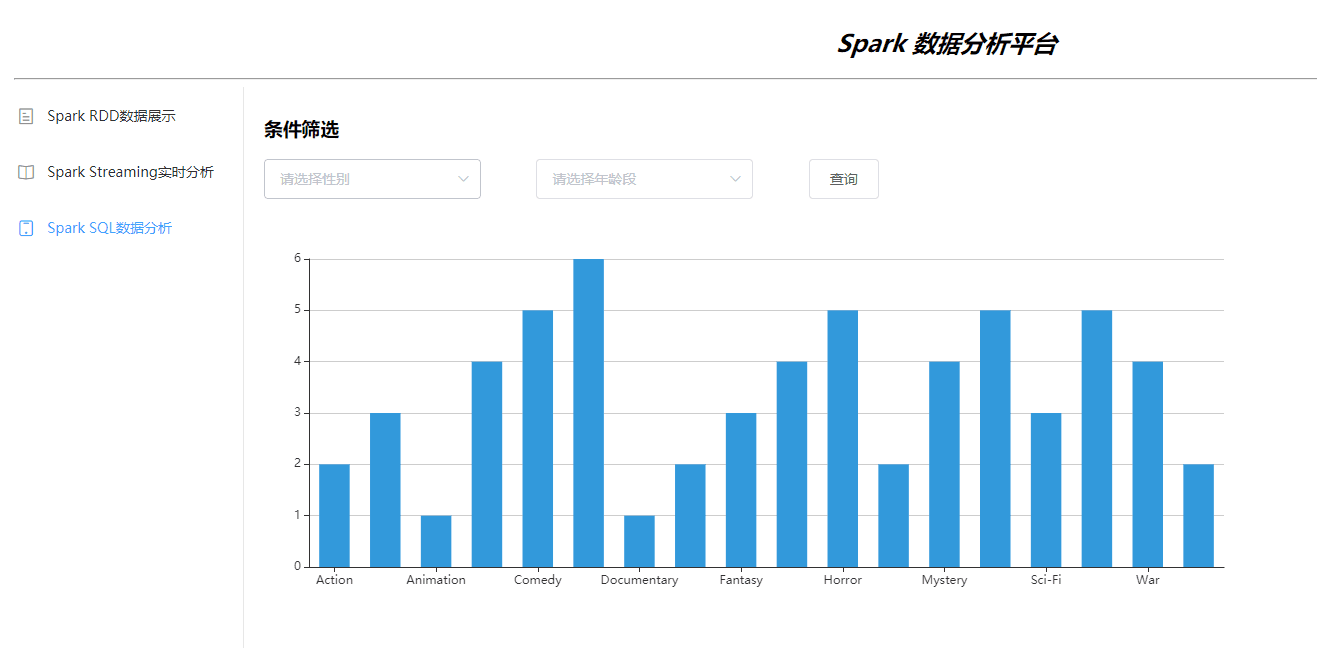


图 6-3-2 无条件显示

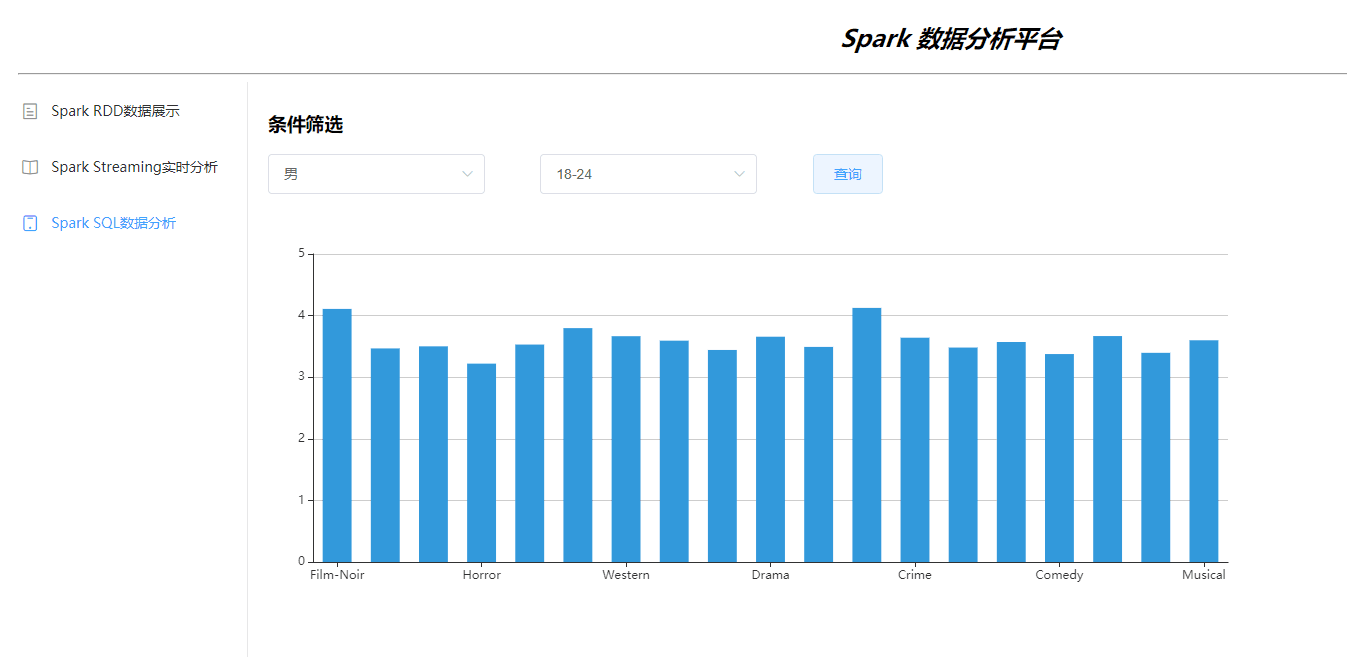


图 6-3-3 条件筛选

结 束 语

本次项目要求是使用Web开发技术，将页面、Spark、Hadoop相关技术进行整合，形成整体框架，实现电影评分可视化。

参考文献

[1] Holden Karau，Andy Konwinski，《Spark快速大数据分析》，中国工信出版设

[2] Spark官方文档

[3] Spark API

致谢

感谢给予帮助的同学和广大网友!

感谢小组各成员相互配合，坚持不懈!

最后向在百忙之中评审本文的老师表示衷心的感谢！