教育部高等学校大学计算机课程教学指导委员会

中国大学生计算机设计大赛



花满蹊作品文档说明书

作品编号：　　　　　　　 83806

作品名称： 花满蹊

作　　者：　　 王帅帅、梁焯炫、何雨飞

版本编号：　　　　　　 V1.0.0

填写日期：　　 2020年6月1日

目录

[第一章 需求分析 3](#_Toc43470678)

[第二章 概要设计 4](#_Toc43470679)

[第三章 详细报告 5](#_Toc43470680)

[3.1 界面设计 5](#_Toc43470681)

[3.1.1界面展示 5](#_Toc43470682)

[3.1.2使用流程介绍 6](#_Toc43470683)

[3.2 数据库设计 7](#_Toc43470684)

[3.2.1 关系型数据库 8](#_Toc43470685)

[3.2.2 非关系型数据库 8](#_Toc43470686)

[3.3 搜索服务设计 9](#_Toc43470687)

[3.3.1 Elasticsearch简介 9](#_Toc43470688)

[3.3.2 Elasticsearch集群设计 9](#_Toc43470689)

[3.4 个性化推荐模块架构设计 11](#_Toc43470690)

[3.4.1个性化推荐模块架构设计总览 11](#_Toc43470691)

[3.4.2 基于Flume的数据采集 14](#_Toc43470692)

[3.4.3 基于HDFS的数据存储 16](#_Toc43470693)

[3.4.4 基于Spark的数据处理 16](#_Toc43470694)

[3.4.5 基于Redis的实时数据存储 17](#_Toc43470695)

[3.4.6基于MongoDB的负样本存储及召回集存储 17](#_Toc43470696)

[3.5 个性化推荐模块的功能实现 18](#_Toc43470697)

[3.5.1相似古诗 18](#_Toc43470698)

[3.5.2每日推荐 22](#_Toc43470699)

[3.5.3 猜你喜欢 23](#_Toc43470700)

[3.6 关键技术 23](#_Toc43470701)

[3.6.1召回模型SVD的训练与使用 23](#_Toc43470702)

[3.6.2 排序模型Xgboost+LR的训练与使用 27](#_Toc43470703)

[3.6.3 词嵌入模型Word2Vec 35](#_Toc43470704)

[第四章 主要测试 36](#_Toc43470705)

[第五章 安装及使用 38](#_Toc43470706)

[5.1 数据库数据导入 38](#_Toc43470707)

[5.2 Elasticsearch的安装配置 38](#_Toc43470708)

[5.3 系统后台程序安装及使用流程 38](#_Toc43470709)

[5.4 个性化推荐模块安装及使用流程 39](#_Toc43470710)

[5.4.1个性化推荐模块的安装 39](#_Toc43470711)

[5.4.2个性化推荐模块的使用 39](#_Toc43470712)

[第六章 项目总结 42](#_Toc43470713)

# 需求分析

本次参赛，我们设计并实现了一款融合智能分析与个性推荐的跨平台诗词学习系统，名为“花满蹊”，取自诗圣杜甫的《江畔独步寻花·其六》中的诗句“黄四娘家花满蹊，千朵万朵压枝底”。

赛题的选取立足于弘扬中华优秀传统文化。中国古诗词作为中华文化的重要组成部分，是中华文化之瑰宝，几千年来历久弥新，需要年轻的我们去体会和感悟。通过本系统，希望能够利用大数据架构和数据分析方法，将优秀的中国古诗词作品呈现给世人。本系统的操作逻辑符合当今人们的学习习惯，适合所有想学习古诗词的人们，并且能够使用户获得良好的学习体验。

“花满蹊”包含web网页端和微信小程序端，以这种跨平台的方式尽可能的涵盖终端使用范畴；主要功能包括每日推荐、猜你喜欢、相似古诗、诗词搜索、收藏等；功能和性能测试结果显示，该系统能够有效满足诗词的学习和品鉴需求。

为了从多个维度突出本作品的功能特点，现选取已有的作品进行对比，图1是对比分析表格：

|  |  |  |
| --- | --- | --- |
|  | 花满蹊 | 古诗文网 |
| 是否具有小程序端 | 是 | 是 |
| 是否支持诗词搜索 | 是 | 是 |
| 界面是否简洁 | 是 | 否 |
| 是否支持古诗播放 | 是 | 是 |
| 是否支持用户反馈收录 | 是 | 是 |
| 是否可以收藏古诗 | 是 | 是 |
| 是否针对用户推荐诗词 | 是 | 否 |

图1 同已有产品的对比

以上就是和古诗文网的对比，古诗文网是一个很好的古诗词学习网站，并且深耕多年，有很多值得我们学习的地方，在之后的更新中我们也会虚心吸取各个类似作品的优点，不断更新完善，努力提升用户体验！

# 概要设计

本小节将会对花满蹊的主要功能模块进行介绍，通过这些功能模块能够快速的了解本系统的核心功能，具体介绍如下：

1. 登录注册模块：提供用户的登录注册，以及管理员的登录功能。
2. 内容搜索模块：基于ElasticSearch实现，用户可根据内容和类别搜索诗词、诗人、名句来获得想要得到的内容。
3. 智能推荐模块：用户登录后花满蹊会记录用户所浏览的诗词诗人信息，根据这些信息进行计算得出用户可能喜欢的的诗词，进而实现猜你喜欢、每日推荐功能；通过对一首诗的类型朝代等信息进行计算分析，得出与该诗相似的诗词，进而实现相似古诗功能。
4. 个人中心模块：用户在这里可查看自己收藏的诗词和诗人，并对自己的账号进行管理。
5. 数据管理模块：管理员在这里可以对本系统的诗词诗人数据进行管理，并且也可以对用户的反馈信息，如诗词收录完善，诗人完善信息进行管理

模块之间的关系及层次结构图如图1所示：

花满蹊

登录注册模块

数据管理模块

个人中心模块

智能推荐模块

内容搜索模块

用户登录

管理员登录

用户注册

诗人搜索

名句搜索

诗词搜索

每日推荐

相似古诗

猜你喜欢

收藏的诗人

账号管理

收藏的诗词

诗词完善管理

诗人完善管理

收录管理

图2 花满蹊系统模块

# 详细报告

## 3.1 界面设计

本小节将会介绍花满蹊程序的界面设计及使用流程，花满蹊有用户端和管理员端，其中用户端又分为web网页端和微信小程序端，花满蹊的界面是我们团队的原创作品，界面典雅大方、风格富有中国古典美感、内容呈现位置合理美观。

### 3.1.1界面展示

下图分别展示了web网页端和微信小程序端的界面，其中图3-图6展示的是用户的web网页端界面，图8和图9展示的是用户微信小程序端界面，图9和图10是管理员端界面。

|  |  |
| --- | --- |
| 图3网页端每日推荐 | 图4网页端诗词界面 |
| 图5网页端诗人界面 | 图6网页端个人中心界面 |
| 图7每日推荐 | 图8诗词界面 |
| 图9收录管列表 | 图10完善界面 |

### 3.1.2使用流程介绍

图11为用户使用流程图，该流程图适用于web网页端和微信小程序端，用户进入花满蹊如果没有登录就是处在游客模式下，该状态下花满蹊中的诗词、诗人、名句及其详情都是可以查看的，并且搜索功能也可以使用，不过由于该状态下花满蹊不会记录用户使用记录，所以智能推荐功能是不能用的。

当用户点击收藏或者个人空间时会有提示先登录，如果您是新用户可点击注册进行注册，否则可直接登录，登录之后花满蹊会记录用户浏览的每一条记录，并实时保存到Redis和MySQL供计算使用，通过这些数据进行计算分析从而实现猜你喜欢、每日推荐功能。

用户模式下用户可以收藏自己喜欢的诗词和诗人，在诗词详情里用户也可收听该诗音频，以及检测该诗背诵情况。在个人中心里用户可以查看自己收藏的诗词和诗人，还可以对自己的账号进行管理。

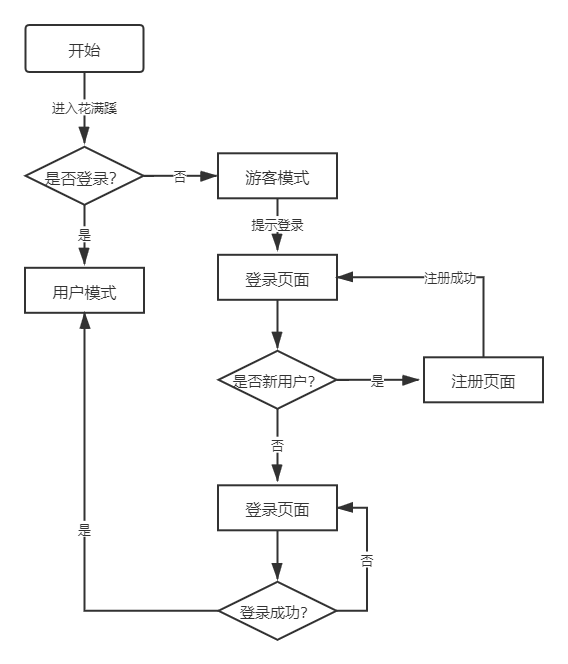


图11用户使用流程图

## 3.2 数据库设计

花满蹊所使用的数据库包括关系型数据库和非关系型数据库，关系型数据库里存放本程序的页面内容数据，如诗词、名句、诗人等和用户数据作为所有数据的存放地，并作简单的查询，关系型数据库使用的是MySQL；非关系型数据库使用的有Redis、MongoDB，关系型数据库和非关系型数据库配合使用，作为本项目数据来源，实现本项目功能。下面分别介绍一下：

### 3.2.1 关系型数据库

花满蹊使用MySQL作为关系型数据库，在所建立的数据库中使用了16张表， 其中实现用户功能主要的六张表为诗词表poem、作者表author、名句表rhesis、用户表user、用户诗词表userandpoem、用户作者表userandauthor，ER图如图12所示：

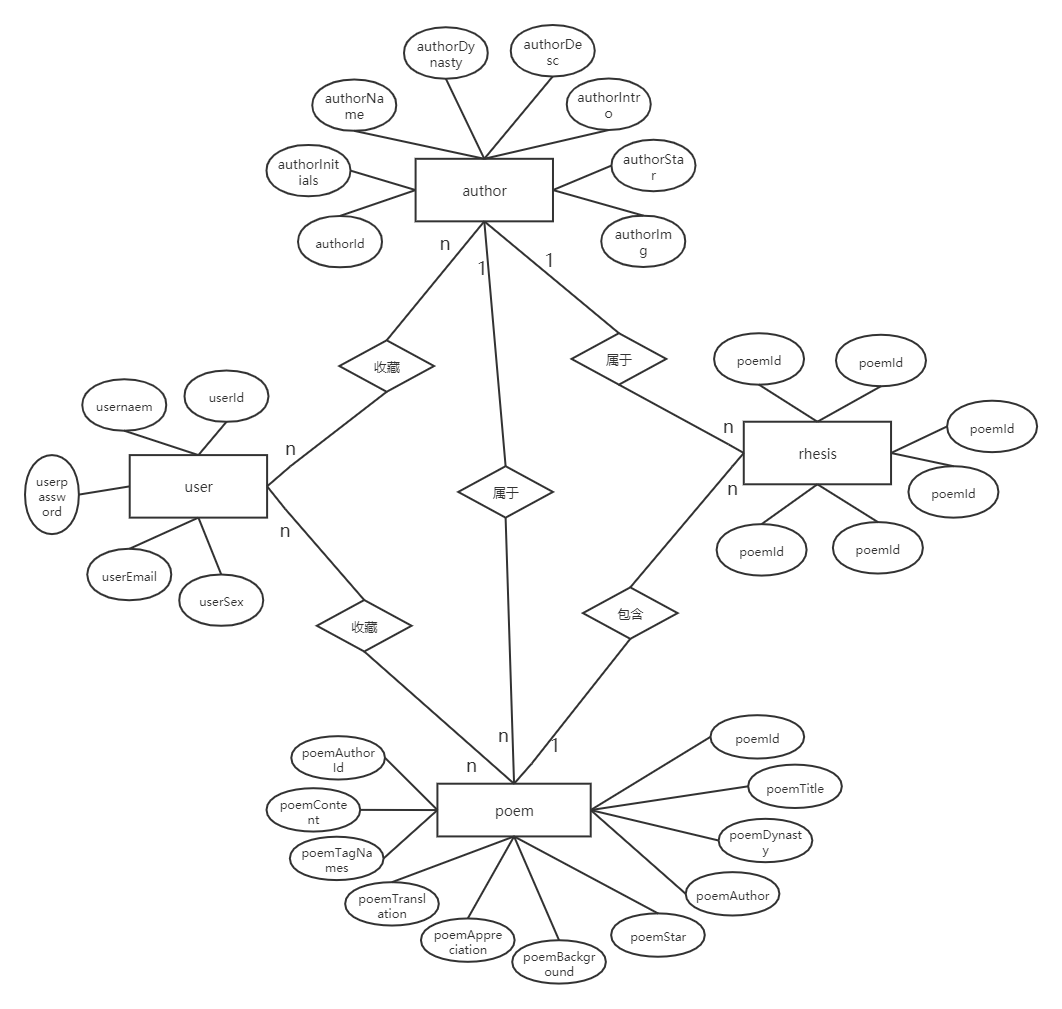


图12

### 3.2.2 非关系型数据库

本作品非关系型数据库采用Redis、和MongoDB，其各自都有其特定的作用，下面介绍一下其承担的主要作用：

1. Redis用于存储用户部分操作信息，比如查看的诗词和诗人等，用该信息作为分析用户使用习惯的数据；
2. MongoDB数据库用作推荐系统中召回候选数据的存储和实时采集的负样本的存储。

## 3.3 搜索服务设计

### 3.3.1 Elasticsearch简介

Elasticsearch（ES）是一个基于Lucene构建的开源、分布式、RESTful接口的全文搜索引擎。Elasticsearch还是一个分布式文档数据库，其中每个字段均可被索引，而且每个字段的数据均可被搜索，ES能够横向扩展至数以百计的服务器存储以及处理PB级的数据。可以在极短的时间内存储、搜索和分析大量的数据。并且ES天生就是分布式的，它知道如何管理多个节点来完成扩展和实现高可用性，所以部署时只需修改其配置文件即可实现集群部署。

### 3.3.2 Elasticsearch集群设计

花满蹊采用Elasticsearch提供搜索服务，部署三个Elasticsearch服务节点构成集群，建立poem、author、rhesis三个索引库，每个索引有五个主分片，每个分片有一个复制，最终每个索引就有十个分片，这样能很好地实现故障转移，保证搜索功能的高可用性。下面分别从逻辑结构和物理结构两方面介绍一下集群中的索引、分片、复制之间的联系（以poem索引为例）。

1. 逻辑结构：

每个索引下面都有五个分片，这些分片将一个索引下的数据按一定规则分成五份，并且每个分片都有一个复制分片，这样可以横向的扩展索引的存储量。逻辑结构图如图13所示。

0

4

3

2

1

0

4

3

2

1

逻辑结构

索引主分片

每个主分片的复制

图13 逻辑结构

1. 物理结构

每个索引的五个主分片以及其每个分片的复制分布在我们所搭建的三台Elasticsearch服务节点上，并且主分片和其复制分片不在同一个Elasticsearch服务节点上，这样可以实现故障转移，即使一台Elasticsearch服务节点出问题其整个搜索服务还是可以运行的。物理结构图如图14所示。

Node01

Node02

Node03

0

1

2

3

4

2

0

3

4

1

物理结构

每一个Node都是一个Elasticsearch服务节点

图14 物理结构

## 3.4 个性化推荐模块架构设计

### 3.4.1个性化推荐模块架构设计总览

为了提升用户在本古诗学习软件的使用体验，帮助用户更快、更准确的发现喜欢的古诗，个性化推荐模块，是我们重点制作的部分。当前是大数据的时代，为了满足在系统用户数量量大、用户实时产生的行为数据量大的情况下提供快速、可靠的个性化推荐服务，本古诗学习系统中的个性化推荐模块是采用大数据架构进行设计的，架构设计是围绕“如何更好地利用和处理用户实时产生的行为数据”搭建的。

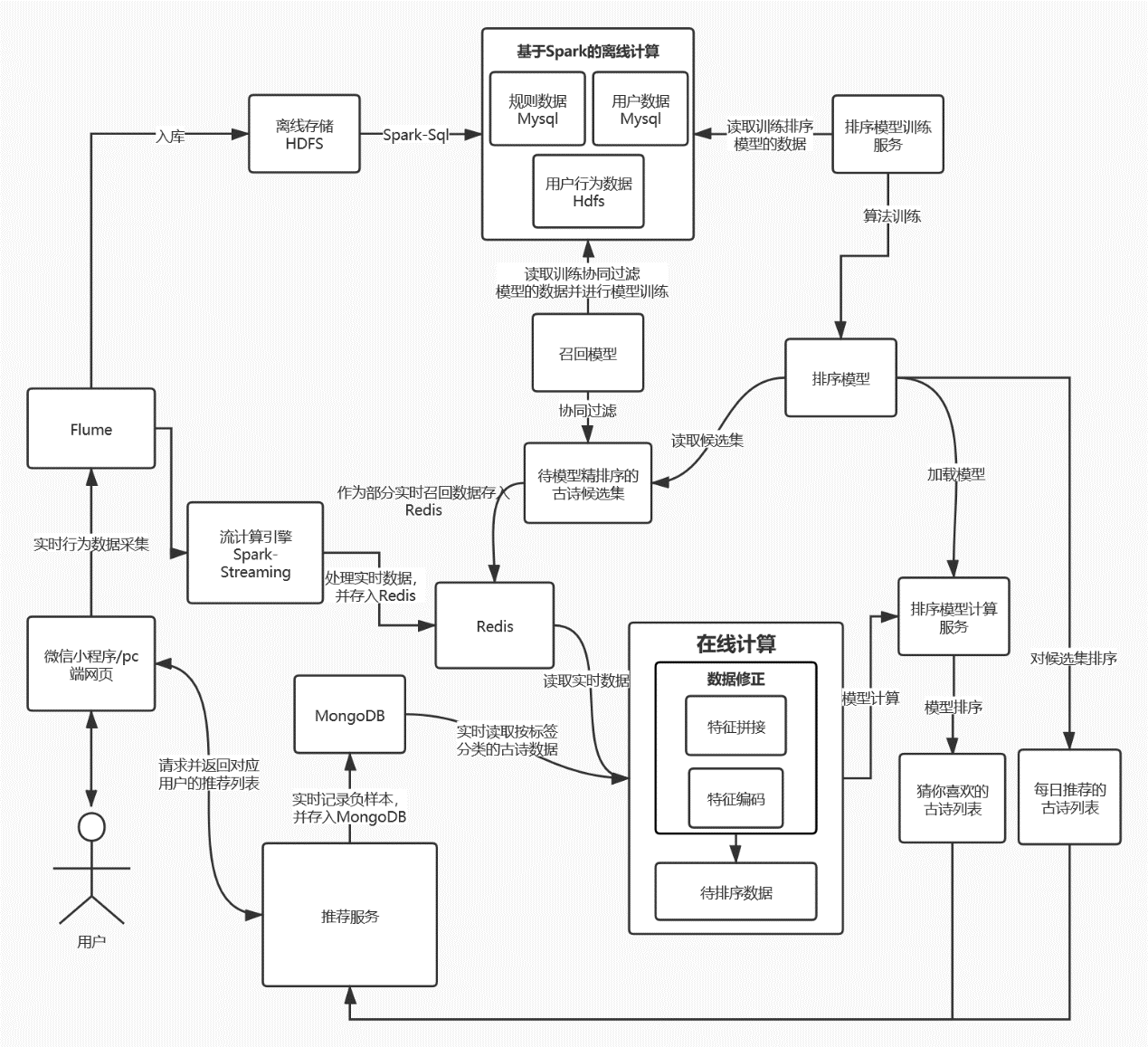


图15 推荐系统整体架构

图15是个性化推荐模块的系统架构图。我们先从图15的左下角开始说明，就是如图16的位置。当用户与系统进行交互时，会实时地产生用户行为数据，后台系统会截取这些数据，以Json的数据格式发送到分布式日志收集系统Flume中。

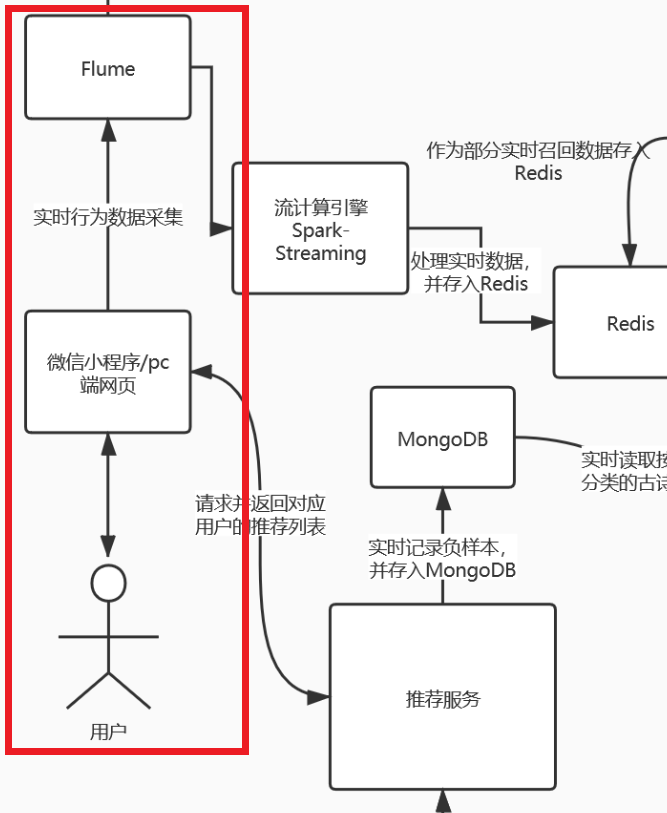


图16用户实时产生的行为数据传入Flume

推荐系统的架构可以分为离线层和在线层，Flume会把web端发送过来的用户行为数据分别传输给离线层和在线层，如图17。

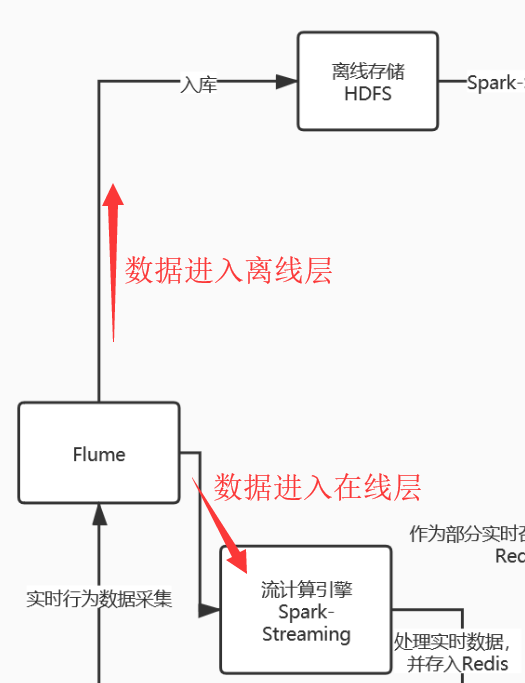


图17 由Flume进行数据的分发

进入离线层的数据会被存入HDFS，这些数据会定期地被Spark程序读取，结合Mysql中的数据进行数据处理，将数据处理为可供模型训练的训练数据，后台使用这些训练数据训练召回和排序模型，用于推荐服务，如图18。

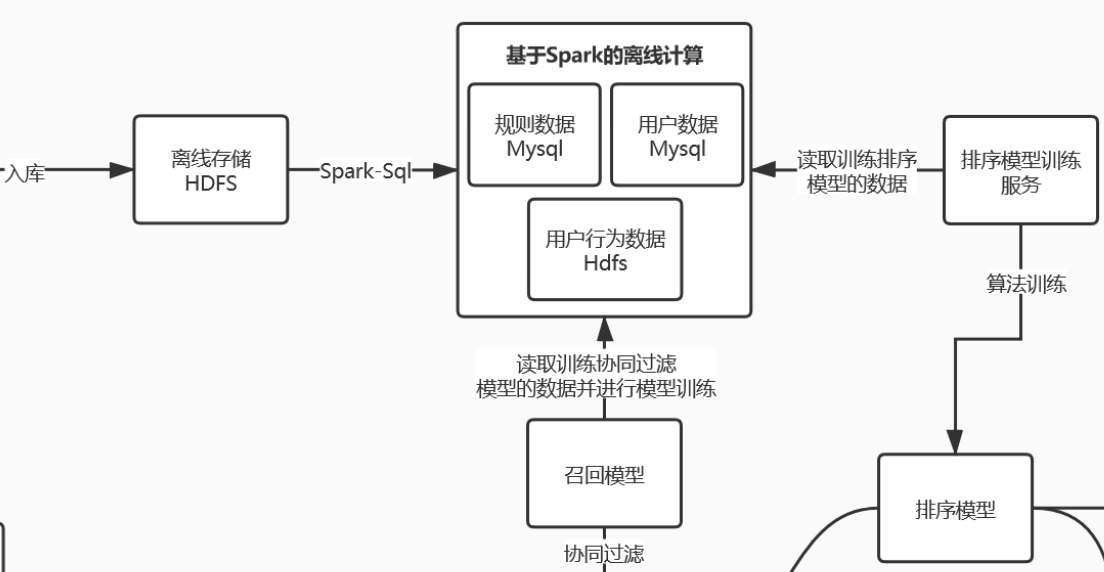


图18离线层架构

进入在线层的数据会被传输给Spark-Streaming，Spark-Streaming对数据里面的信息进行提取并存入Redis，后台会根据Redis里面存储的信息，从Mongodb里面筛选出要进行排序的古诗候选集，后台对古诗候选集的数据进行特征拼接等处理，得到排序模型的输入数据，使用排序模型排序得到要推荐的古诗列表，如图19。

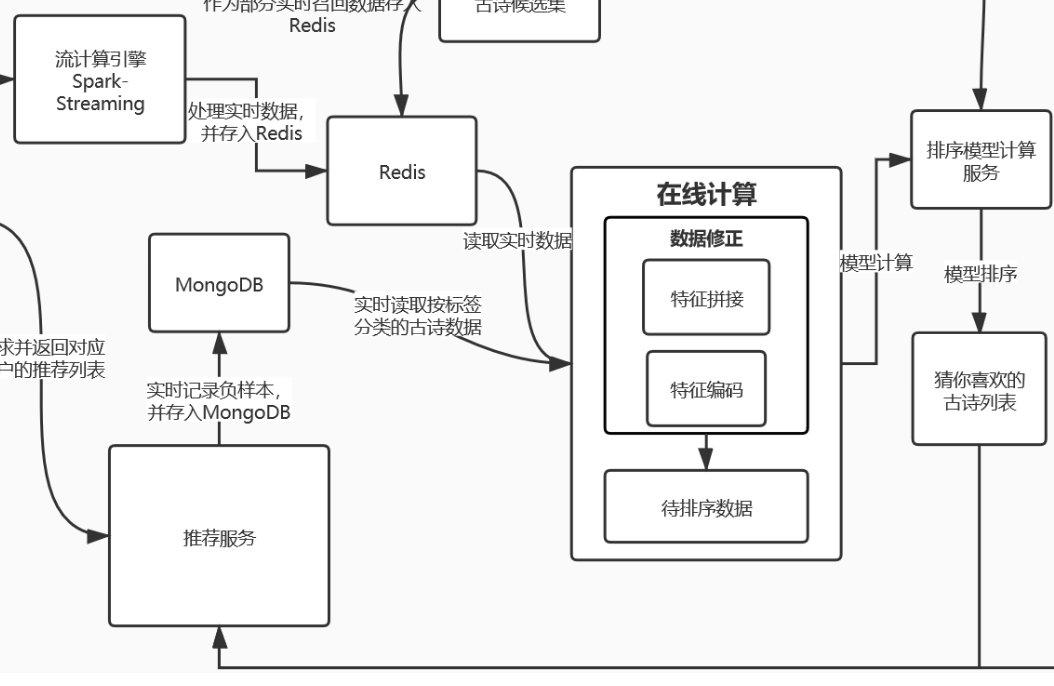


图19 在线层架构

### 3.4.2 基于Flume的数据采集

Apache Flume是Cloudera提供的一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的框架。

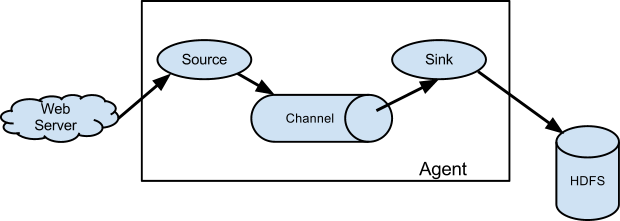


图 20 Flume的运行机制

Flume的运行机制如图20。Flume中的核心是Agent，Agent本身是一个Java进程，它相当于一个数据传递员，由Source、Channel和Sink 三个部分组成。Agent要做的事情就是把数据从数据源（Source）收集过来，将收集到的数据传输到指定的地方（Sink），为确保数据的可靠传输，会将数据放进缓存队列（Channel），待数据成功传输，Flume在删除自己缓存的数据。

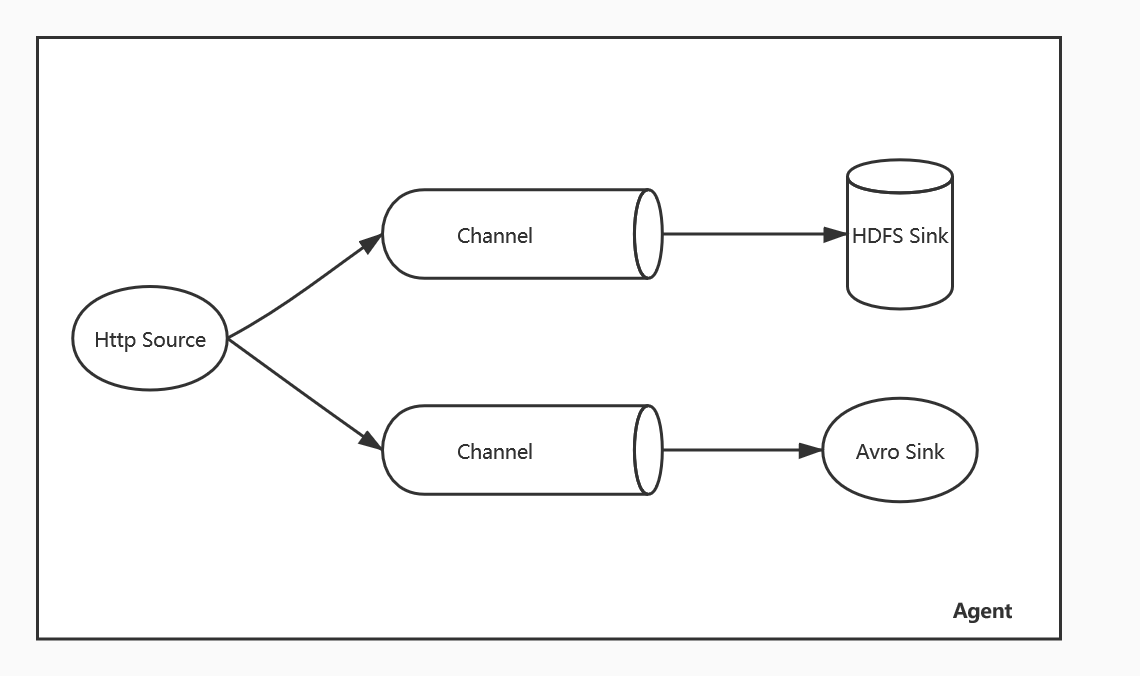


图 21 Flume的结构设计

本系统中Flume的结构设计如图21，使用的Source是Http Source，接收web端发送过来的json数据；使用了两个Sink，分别是HDFS Sink和Avro Sink，HDFS Sink 是定时进行事件回滚，将这个时间段内产生的事件进行存储。Avro Sink用于接收Flume 事件，将Flume事件转化为Avro事件，Spark-Streaming用Flume-Push的方式消费Avro事件。



图 22 传输的Json格式

Http Source 需要接收的Json 格式如图22，这个Json数据的格式是由Flume的org.apache.flume.source.http.JSONHandler方法指定的。这个Json是一个数组，数组里包含多个元素，每个元素有键”headers”和键”body”。键”headers”对应的内容是用于实时推荐的数据，后面会通过Spark-Streaming进行处理并存入Redis；键”body”对应的内容将文本文件的形式存入HDFS。Flume配置文件如图23，系统在运行Flume的时候将根据里面的配置运行。

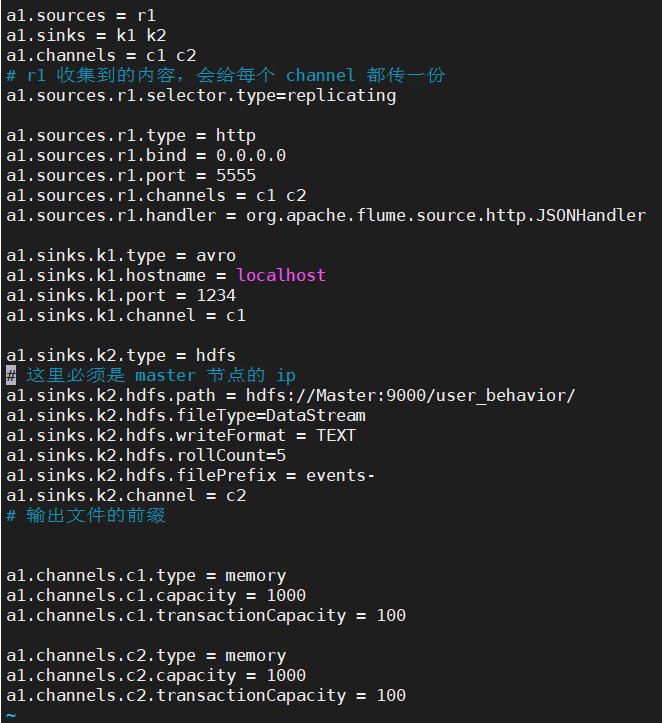


图 23 Flume运行的配置文件

### 3.4.3 基于HDFS的数据存储

分布式文件管理系统，是一种用来管理不同机器上的文件的技术，HDFS就是分布式文件管理系统的一种。HDFS(Hadoop Distributed File System)，用于存储文件，通过目录树来定位文件，它是分布式的，由很多机器联合起来实现其功能，集群中的机器扮演着各自的角色。

在本推荐系统中，要存储的日志数据是结构化的数据，为了适应系统搭建初期，用户使用时间短的情况，HDFS中按照批次进行数据存储，每5条数据存储一个文件，以文件I/O时的时间戳作为文件的名字。这些文件在HDFS中的数据存储情况如图24。

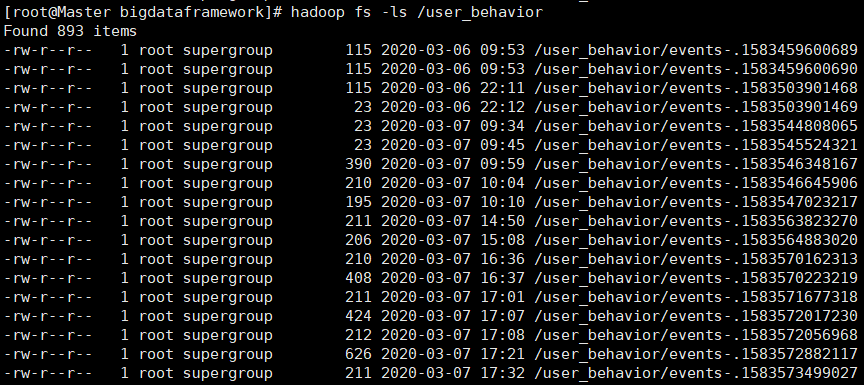


图 24 HDFS数据存储展示图

### 3.4.4 基于Spark的数据处理

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎，它是基于内存的，计算速度非常快，Spark主要由 Spark-Core、Spark-SQL、Spark-Streaming等部分组成。Spark-Core包含Spark的基本功能，内部定义了RDD（弹性分布式数据集）是Spark里面最基本的可操作对象，Spark-Core提供了很多APIs来操作这些RDD；Spark-SQL是Spark用来操作结构化数据的组件，可以用DataFrame、SQL语句等方式来操作结构化数据；Spark-Streaming是实时数据流处理组件，提供了API来操作实时流数据。

在本系统中，除了Spark-Streaming部分的是使用Scala语言编写，其他操作Spark的部分均使用Python语言编写，具体的处理步骤，会在功能实现部分进行详细说明。

### 3.4.5 基于Redis的实时数据存储

Redis是完全开源免费的，用C语言编写的，遵守BSD开源协议，是一个高性能的(key/value)分布式内存数据库，基于内存运行,并支持持久化的NoSQL数据库，它也通常被称为数据结构服务器，因为值（value）可以是字符串(String), 哈希(Map), 列表(list), 集合(sets) 和有序集合(sorted sets)等类型。

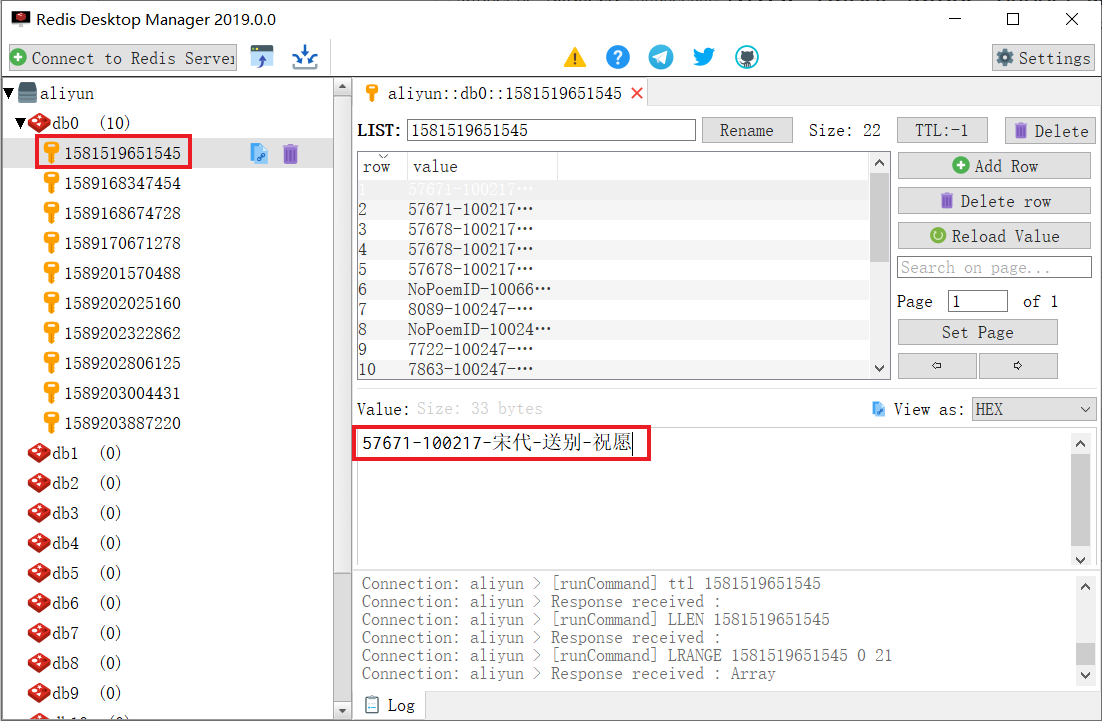


图25 Redis数据展示

在本系统中，使用列表数据结构进行存储。使用用户id作为键，值使用列表作为的存储结构，列表里面每一个元素是一个字符串，这个字符串记录的是一个用户的一次行为对应的古诗的标签，这个标签包括古诗的朝代、诗人id、古诗的类型。这里以用户id为1581519651545的数据作为展示，其对应的值里面的第一个元素为字符串” 57671-100217-宋代-送别-祝愿”，如图25所示。

### 3.4.6基于MongoDB的负样本存储及召回集存储

MongoDB是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。在高负载的情况下，添加更多的节点，可以保证服务器性能。MongoDB旨在为WEB应用提供可扩展的高性能数据存储解决方案。MongoDB将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB文档类似于JSON对象。字段值可以包含其他文档，数组及文档数组。

在本系统中，MongoDB用作召回候选数据的存储和实时采集的负样本的存储。召回候选集是经过文本相似度筛选出来的，与每首古诗文本相似度最高的100首古诗和根据古诗标签分组筛选出来的每个标签对应的所有古诗。实时采集的负样本是将实时推荐的古诗列表全部记录下来。

## 3.5 个性化推荐模块的功能实现

个性化推荐模块具有三个功能，分别是相似古诗、每日推荐和猜你喜欢。这里将对这些功能的实现进行介绍。

### 3.5.1相似古诗

相似古诗推荐的列表，是根据某一首古诗，计算与当前古诗文本相似度最高的10首古诗。由于计算量大，我们会先根据古诗的诗人、朝代、标签先筛选出一部分古诗，以所有古诗的标题和正文作为语料库，对古诗进行分词，并训练Word2Vec模型，计算每一个词的 TFIDF值，使用Word2Vec模型 计算每一个词的词向量，对没首古诗所拥有的词语对应的词向量进行加权求和（当前词的TFIDF值 \* 当前词的词向量）得到每一首古诗对应的古诗向量，并使用余弦相似度计算与当前古诗文本相似度最高的前100首古诗，将每首古诗与其文本相似度最高的100首古诗存入Mongodb数据库，除了步骤(8)，其他步骤均使用Spark进行处理。功能实现步骤如下：

1. 导入数据

通过pyspark将数据文件 source.csv导入到内存，转换成pyspark.sql.DataFrame。

1. 将每首古诗的标题与内容进行合并

使用pyspark.sql.functions.concat\_ws方法合并古诗标题与古诗内容，代码及输出结果如图20。

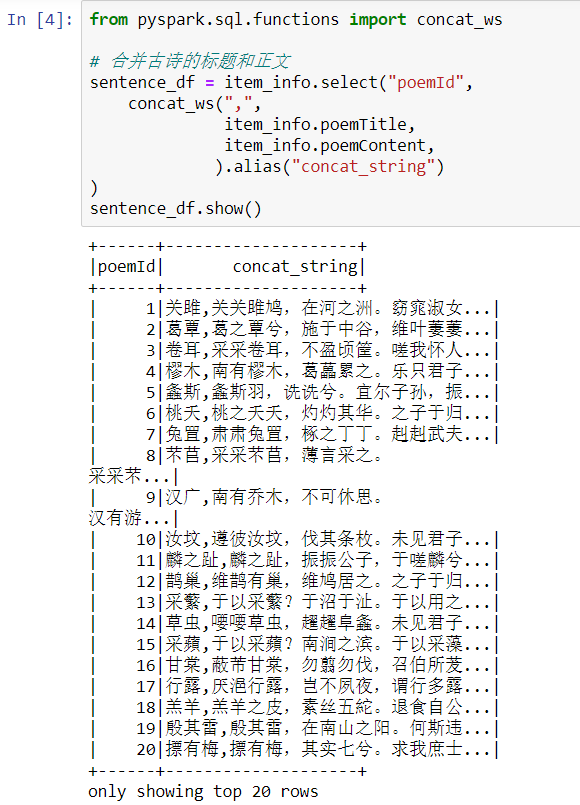


图 26 合并古诗标题和内容

1. 对每首古诗合并的内容进行分词

* 使用正则表达式去除古诗中除文字外的标点符号
* 使用jieba库对每首古诗进行分词

代码及输出结果如图26、图27。

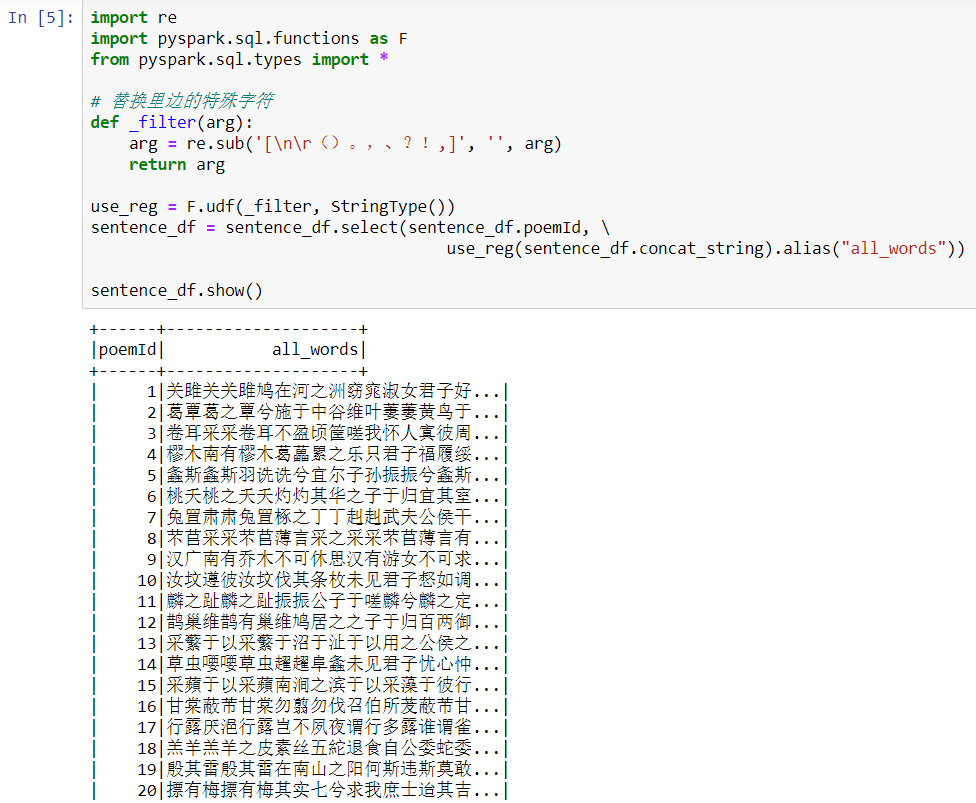


图 27 使用正则表达式去除古诗中除文字外的标点符号

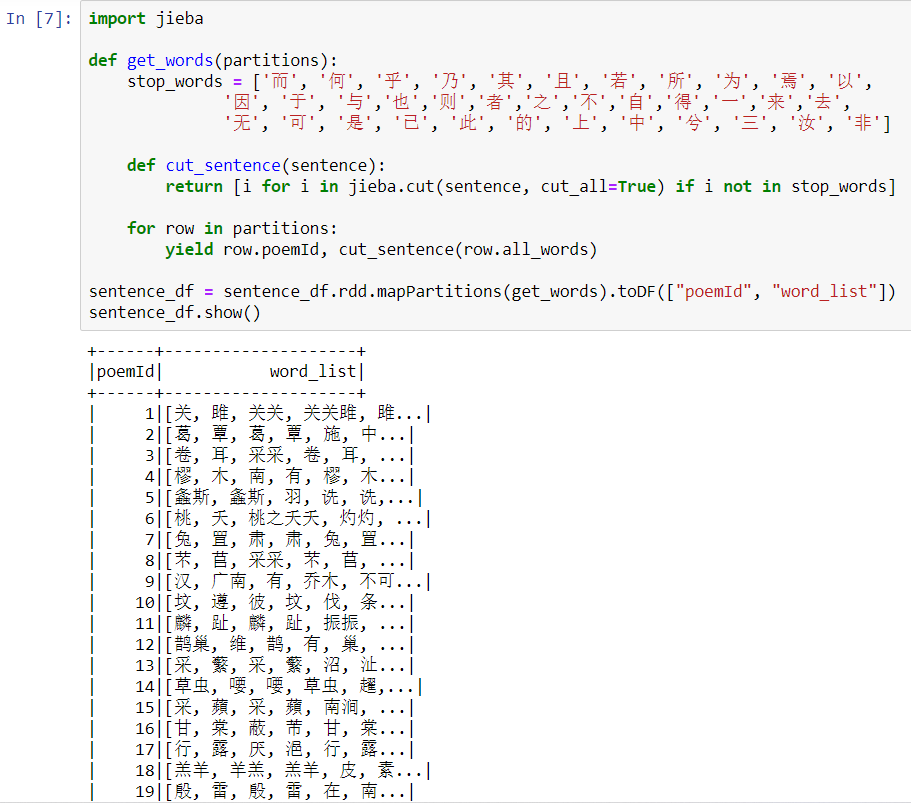


图 28 使用jieba库对每首古诗进行分词

1. 计算每个词的TFIDF值

根据TDIDF值的计算公式，计算得到每个词的TFIDF值，将结果按照TFIDF排序，核心代码如图29。

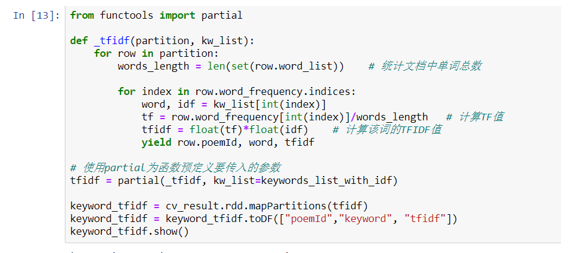


图 29 计算每个词的TFIDF值

1. 把每首古诗分词之后的数据作为训练集，训练Word2Vec模型，得到每个词的词向量

设置词向量的向量长度为10，作为Word2Vec的模型参数，传入每首诗分词后的数据训练模型，代码及输出结果如图30。

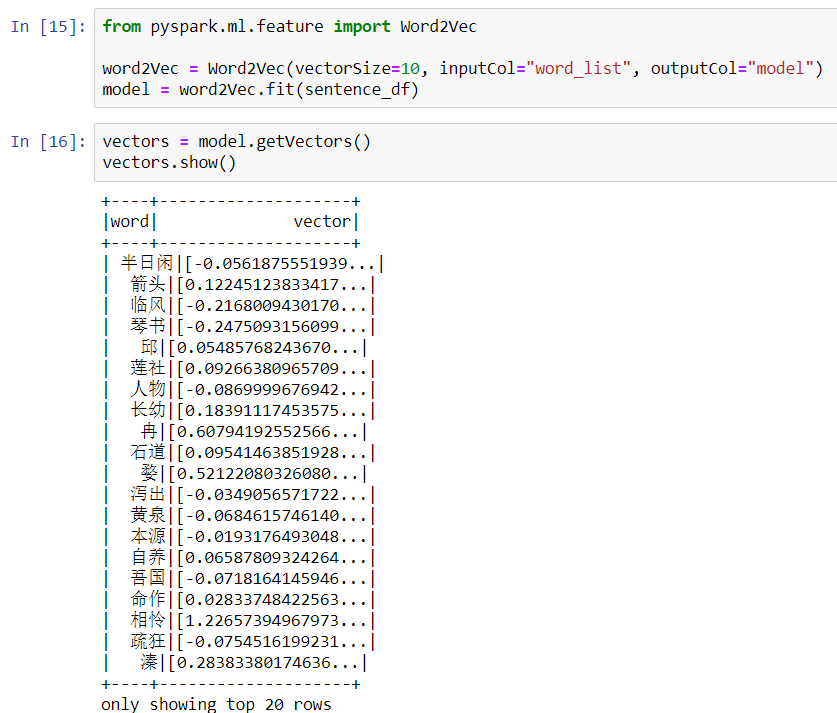


图30 训练Word2Vec模型

1. 用每个词的TFIDF值与该词对应的词向量相乘，得到每个词对应的加权词向量

对于每个词，使用(4)计算得到的该词的TFIDF值，与(5)计算得到的该词的词向量相乘，得到其加权词向量，代码及输出结果如图31。



图31 计算加权词向量

1. 求出每首古诗的词向量的均值

将每首古诗里面分词得到的词的加权词向量求均值，讲这个值作为每首古诗对应的古诗向量。

1. 使用每首古诗的词向量的均值计算两两古诗之间的文本相似度，并得到每首古诗与其文本相似度最高的100首古诗

使用余弦相似度计算两两古诗之间的文本相似度，假设古诗A对应的古诗向量为a，古诗B对应的古诗向量为b，余弦相似度计算公式为：

使用余弦相似度计算得到每首古诗与其文本相似度最高的100首古诗，这100首古诗里面的前10首古诗作为相似古诗推荐的内容。

### 3.5.2每日推荐

后台会在每天00:00点调已经封装好的Spark程序读取HDFS里面格式化的用户行为数据，对数据进行清洗和处理，使用处理好的数据训练召回模型SVD和排序模型 Xgboost+LR，使用召回模型对数据（经过Spark特定处理的数据）进行粗排序，得到每个用户对应的古诗推荐候选集（Top-100），使用排序模型对每个用户对应的古诗推荐候选集进行精排序，得到每个用户每日推荐将要推荐的古诗列表（Top-4）功能实现过程如下：

* 1将从Flume收集到Hdfs的数据进行整合，并转换为userlogs表
* 2读入user表、poem表
* 3对这些表进行处理，得到训练召回模型和排序模型的数据
  + 3.1处理userlogs表
    - 3.1.1去除关于搜索的日志
    - 3.1.2根据用户的行为（点击、收藏、点赞）进行分组
    - 3.1.3对每个分组进行处理
      * 3.1.3.1使用正则表达式提取parameter字段里的古诗id等信息
      * 3.1.3.2对行为进行打分
    - 3.4以（用户id，古诗id）为分组键进行分组
      * 3.4.1根据评分进行聚合求和处理得到dataframe1
      * 3.4.2向dataframe1中加入负样本
      * 3.4.3对数据进行修正,得到dataframe1’
  + 3.2处理user表
    - 3.2.1仅留下用户的id和用户的性别
  + 3.3处理poem表
    - 3.3.1对古诗标签、古诗朝代、诗人这几个特征进行独热编码
    - 3.3.2对poemStar进行归一化
  + 3.4合并多表作为训练排序模型的数据集
    - 3.4.1 userlogs表和poem表根据古诗id进行主键合并得到dataframe2
    - 3.4.2 dataframe2和user表根据用户id进行主键合并得到dataframe3
    - 3.4.3特征创建
* 4训练召回模型，并得到每个用户对应的推荐候选集
* 5训练排序模型，使用排序模型对上一步得到的推荐候选集进行排序，得到Top-4

### 3.5.3 猜你喜欢

后台实时的根据Redis里面记录的，用户之前操作的古诗对应标签，实时地召回古诗候选集，并使用训练好的排序模型对古诗候选集进行排序，得到猜你喜欢的推荐列表。具体过程如下：

1. 推荐系统的在线层实时地获取用户与系统交互产生的行为数据

Flume实时地收集道后台发送过来的用户行为数据，并传给Spark-Streaming，由Spark-Streaming对这些数据提取信息并存入Redis。

1. 实时地召回待排序的推荐候选集

后台根据用户发送的请求，实时地从Redis里面召回该用户对应的发生行为的古诗的属性（诗人、朝代、古诗标签），用这些属性从Mongodb中搜索属性对应的古诗作为候选集，并向搜索与用户发生行为的古诗文本相似度最高的10首古诗加入到候选集，得到最终的推荐候选集。

1. 对推荐候选集进行排序

使用训练好的Xgboost+LR模型对推荐候选集进行排序，得到要进行实时推荐的10首古诗。

## 3.6 关键技术

### 3.6.1召回模型SVD的训练与使用

1. 模型介绍

业界的推荐系统中，一般分为召回+排序两个阶段。协同过滤又称召回，召回相当于是粗排序，从海量的用户物品对中筛选出一部分，后面在使用排序模型对这部分数据进行精排序。此处使用的召回算法是SVD算法，SVD是一个基于矩阵分解MF（Matrix Factorization）的算法，所以先介绍一下MF算法。

在推荐系统中，要根据用户的历史行为（点击、收藏、评分等），挖掘用户感兴趣的物品进行推荐，根据评分矩阵里面已有的评分，预测评分矩阵里面空缺的评分，然后对于每个用户，为其对应的物品向量的评分进行排序找出 Top-N，即可完成推荐。

这里设要得到的用户-物品矩阵为 ,这里的中心问题是：如何根据 用户-物品 评分矩阵里面的非空值，预测出该矩阵里面空缺位置应该填的值？我们可以进行矩阵分解，将 矩阵分解成两个矩阵相乘的形式 ，假设要将 矩阵分解成 、，其中 是模型要学习的隐因子的个数。根据矩阵相乘的性质，两个矩阵的维度分别为、，因 ，所以这里的问题转化为求出 和 。求出了 和 之后， 就可以将 用户-物品矩阵空缺的地方填满，即得到未知的用户-物品对的预测评分。SVD算法是MF算法的改进算法，在原来的基础上加上了偏置项(bias)。

1. 模型训练预评估

* 将Spark处理好的[数据对象](#_2.4.2向dataframe1中加入负样本)spark.sql.Dataframe.dataframe转换为 pandas.DataFrame对象，作为训练数据
* 使用算法库Surprise进行SVD和MF模型的训练
* 使用[RMSE](#_3.4.1模型介绍)作为模型的评估指标
* 使用交叉验证的方式进行模型的训练与评估
* 使用网格搜索找出最优的参数组合

1. 使用交叉验证训练与评估模型

交叉验证(Cross Validation)：将数据平均切分成k份，每次使用其中的k-1份作为训练集，1份作为测试集进行模型的评估。

此处使用5折交叉验证，根据是否加入负样本（保持正负样本之比为1:1）和是否增加偏置项（SVD和MF的区别）作为对比，具体可看图32、33、34、35。

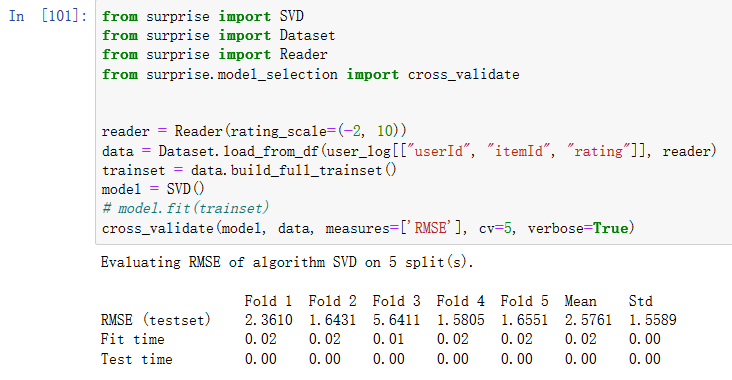


图 32 加入负样本的SVD进行5折交叉验证

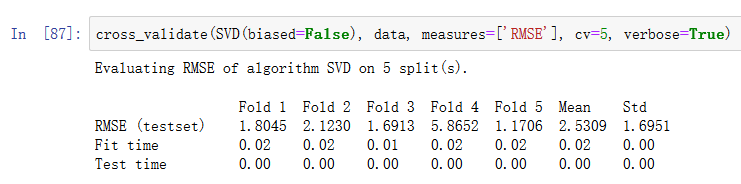


图 33 加入负样本的MF进行5折交叉验证

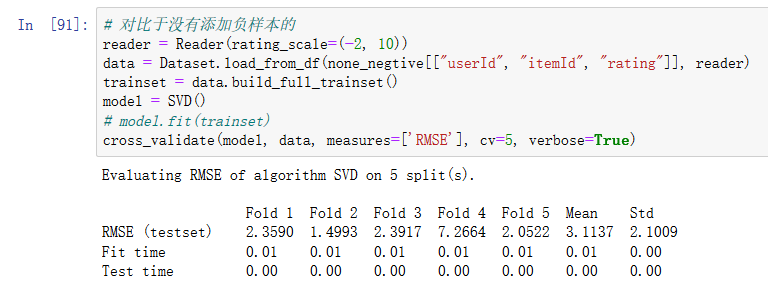


图 34 只含正样本的SVD进行5折交叉验证

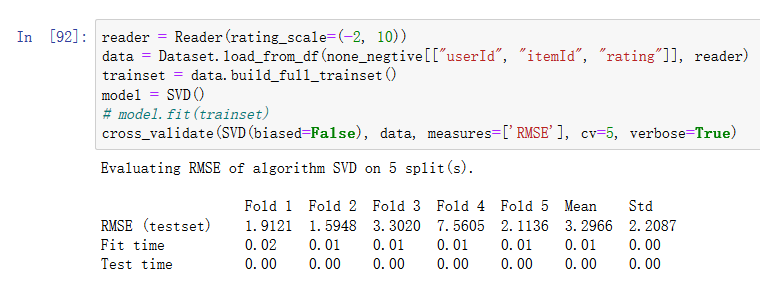


图 35 只含正样本的MF进行5折交叉验证

结果如下：

图 36 不同模型做5折交叉验证的平均RMSE对比

可以发现，保持训练集的正负样本之比是1:1进行训练的SVD算法的RMSE的均值是最小的。因此我们选用SVD模型，并在训练集中保持正负样本之比为1:1。

1. 使用网格搜索找出模型最优的参数组合

网格搜索(GridSearch)：使用暴力搜索的方法，找出使得模型评估指标最优的参数组合。

SVD模型的主要参数介绍如表1，模型进行网格搜索的参数范围如表2。

|  |  |
| --- | --- |
| 参数名 | 含义 |
| n\_factors | 隐因子的数量 |
| n\_epochs | 梯度下降最大迭代次数 |
| lr\_all | 的学习率 |
| reg\_all | 的正则项系数 |

表1 SVD模型主要参数介绍

|  |  |
| --- | --- |
| 参数名 | 搜索范围 |
| n\_factors | 20到100，步长为10 |
| n\_epochs | 10到28，步长为2 |
| lr\_all | 0.002,0.005,0.01 |
| reg\_all | 0.02, 0.1, 0.6, 1, 2, 3, 4, 5 |

表2 SVD模型进行调参的参数搜索范围

最终得到最优的模型的rmse为 2.495，网格搜索的代码以及搜索结果如图37

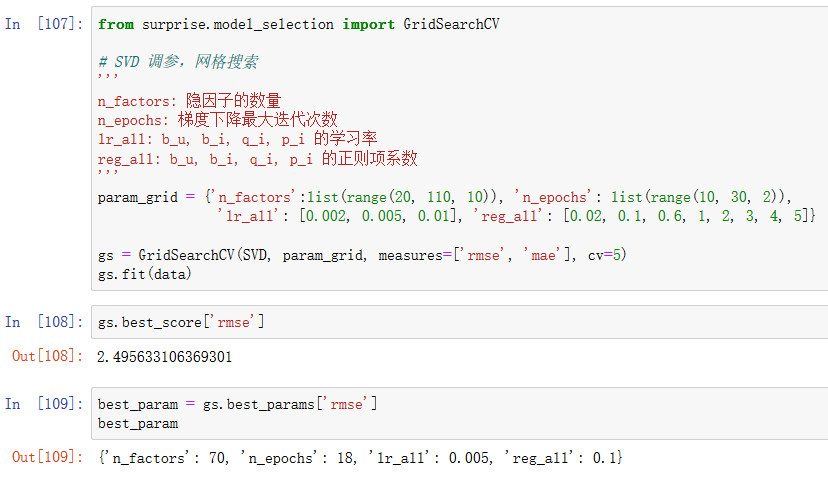


图 37 SVD使用网格搜索调参

1. 模型使用

为了降低进行精排序的计算量，本推荐系统中使用SVD模型对处理好的用户行为数据，进行粗排序，对每个用户推荐候选集的筛选，并保存这些用户推荐候选集供排序模型进行精排序。

### 3.6.2 排序模型Xgboost+LR的训练与使用

一般的推荐系统都会分为召回、排序两个阶段，在排序阶段需要使用精确的模型对召回阶段筛选出来的候选物品进行精排序，这些排序模型又称CTR预估模型。在这部分，我将对比和评估逻辑回归模型LR(Logistic Regression)、因子分解机FM(Factorization Machines)、极限梯度提升树Xgboost与LR的融合模型、极限梯度提升树Xgboost和FM的融合模型的性能，并使用最优的模型对上一部分得到的每个用户对应的Top-100数据进行排序。

#### 3.6.2.1模型介绍

1. 逻辑回归

逻辑回归是一个常用的二分类线性模型，常常被用来进行CTR预估，它通过学习特征向量对应的权重向量，通过对和做点积运算的结果进行变换，将模型的输出值转换为一个0到1之间的类概率值，便可以进行CTR预估。

1. FM

FM模型是LR模型的优化模型，在面对高维稀疏数据的时候，LR模型并不能学习到组合特征的权重，就不能利用好组合特征。

对于特征组合来说，最简单的就是二阶特征组合了。给LR的线性部分加上两两的特征组合的形式如下：

1. Xgboost

Xgboost(Extreme Gradient Boosting)是一个集成模型，其基模型通常使用cart决策树，使用梯度提升的方法进行模型的训练，是当今十分常用且分类效果很优秀的有督促分类模型。

点击率预估模型涉及的训练样本一般是上亿级别的，样本量大，模型通常采用训练速度较快的LR。但是LR是线性模型，学习能力有限，需要大量的特征工程作为基础，十分消耗人力，Xgboost算法可以用来发掘有区分度的特征、特征组合，减少特征工程中的人力成本。这种融合模型在当今是被广泛使用的。

1. Xgboost+LR

Xgboost+LR 是使用Xgboost来对训练集提取特征作为LR模型训练特征的输入数据，LR模型则作为新的训练数据的分类器。

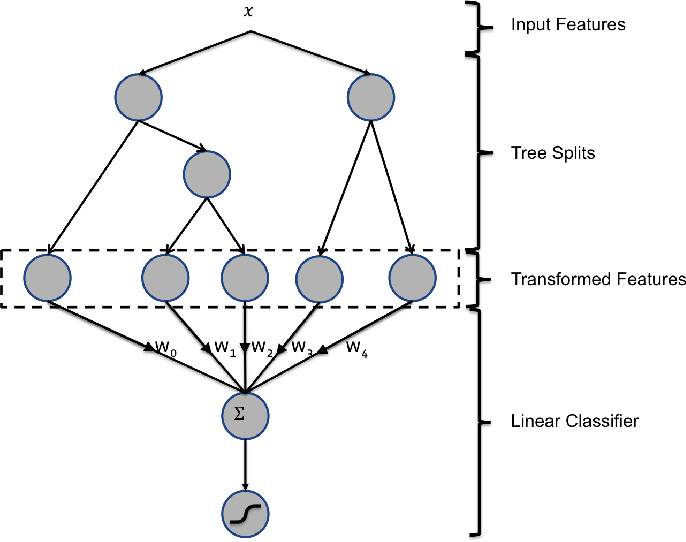


图 38 Xgboost+LR模型结构图

1. Xgboost+FM

FM模型相比于LR模型，可以更好地刻画长尾，长尾就是稀疏的，没有出现过的特征，FM可以算出没有出现过的特征组合。和Xgboost+LR一样，也是先使用Xgboost进行特征编码，将编码的后的数据作为FM模型的训练数据，进行训练。

#### 3.6.2.2模型评估指标

1. AUC

对于样本，类别为1称为Positive，类别为0称为Negative，模型预测正确的样本为True，预测错误的样本为False，对这些概念进行组合，便可以得到混淆矩阵。

根据混淆矩阵便可以得到 True Positive Rate、False Positive Rate

* 表示：在所有真实类别为1的样本中，预测类别为1的比例
* 表示：在所有真实类别为0的样本中，预测类别为1的比例

1. ROC曲线

横轴为FPRate，纵轴为TPRate所绘制的曲线，设置一个概率的阈值，大于这个阈值的样本会被视为正样本，小于这个阈值则被视为负样本，把这个阈值依次取0到1上的有限个值，如0.1, 0.2, 0.3…1.0，每取一个值，便可计算对应的FPRate和TPRate，描点连线，得到的便是ROC曲线，AUC就是ROC曲线下的面积。当TPRate和FPRate相等的时候，AUC取得最小值0.5，表示对于不论真实类别是1还是0的样本，分类器预测为1的概率是相等的。最理想情况下，既没有真实类别为1而错分为0的样本TPRate一直为1，也没有真实类别为0而错分为1的样本，FPRate一直为0，AUC取得最大值1。

1. Accuracy

准确率，表示模型预测正确的样本数占总样本数的比例。

1. Log Loss

Log loss又称为交叉熵损失，用来评估模型输出的概率值，对于真实标签为 并且概率估计函数 ，在模型给定样本真实标签的情况下，每个样本的Log loss定义为：

根据Log loss的计算公式可知，对于正样本，模型预测的概率越接近于1，log loss越小；对于负样本，模型预测概率越接近于0，log loss越小。所以log loss是对于模型对样本预测概率置信度的评估。

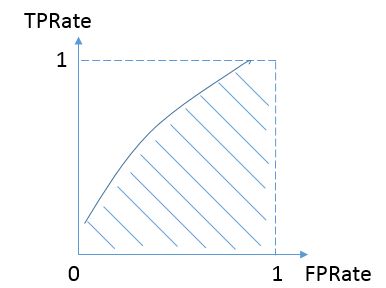


图 39 ROC曲线示例图

#### 3.6.2.3模型训练与评估

此处只对Xgboost+LR模型的训练进行介绍。

* 1.0训练Xgboost模型，以AUC为指标将模型调到最优
  + 1.1 max\_depth 和 min\_weight
  + 1.2 gamma
  + 1.3 subsample和colsample\_bytree
  + 1.4 reg\_alpha
  + 1.5降低学习率
  + 1.6 增加模型的基模型数量
* 2.0使用Xgboost对进行特征编码
  + 2.1使用apply方法得到样本所落在每一颗决策树的叶子节点的位置
  + 2.2对样本落在叶子节点的位置进行独热编码
* 3.0使用编码后的数据训练LR模型，以AUC为指标进行模型调优
  + 3.1正则项系数 C
  + 3.2最大迭代次数 max\_iter

使用网格搜索，对每一组参数的模型训练使用5折交叉验证，评估指标使用AUC，得到表现最优的模型对应的参数。

##### 1.0训练Xgboost模型，以AUC为指标将模型调到最优

在Xgboost模型训练最开始的时候，随机化初始模型参数进行训练，使用评估指标AUC进行模型评估，得到的AUC为 0.681（用于对比使用最优参数的Xgboost模型）。

Xgboost模型训练过程中，使用网格搜索进行调参，各个参数的搜索范围如表3，最终得到的最优参数的取值如表4，我们从中抽取了几个参数的调优代码进行展示，如图40,41,42。

|  |  |
| --- | --- |
| 参数名 | 参数搜索范围 |
| max\_depth | 1到10，步长为1 |
| min\_weight | 1到6，步长为1 |
| gamma | 0.1到0.9，步长为0.1 |
| subsample | 0.4到0.7，步长为0.5 |
| colsample\_bytree | 0.7到0.95，步长为0.5 |
| reg\_alpha | 0到1之间随机取多个数 |
| learning\_rate | 0.01到 0.1之间随机取多个数 |
| n\_estimators | 50到400，步长为50 |

表3 Xgboost模型进行调参的参数搜索范围

|  |  |
| --- | --- |
| 参数名 | 最优参数取值 |
| max\_depth | 4 |
| min\_weight | 1 |
| gamma | 0.3 |
| subsample | 0.75 |
| colsample\_bytree | 0.9 |
| reg\_alpha | 0.05 |
| learning\_rate | 0.01 |
| n\_estimators | 300 |

表3 Xgboost模型的最优参数

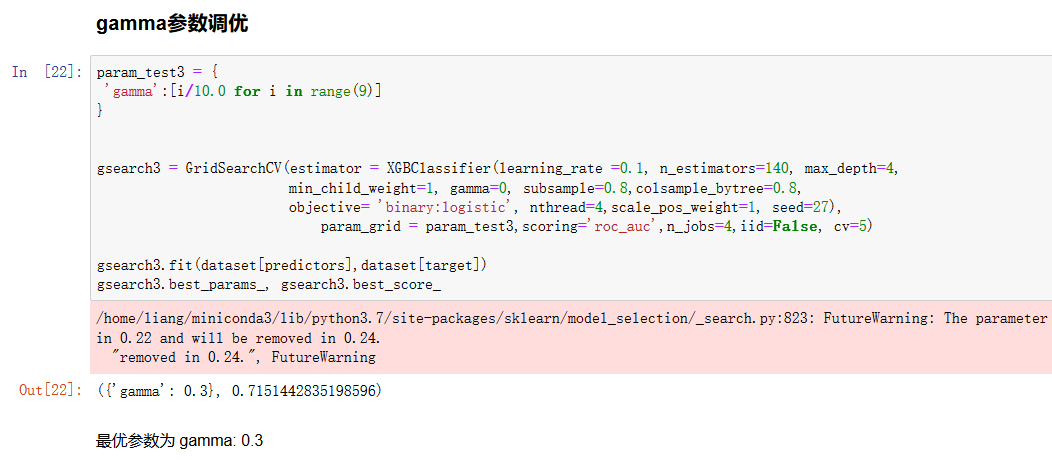
****

图40 gamma参数调优



图41 subsample和colsample\_bytree参数调优

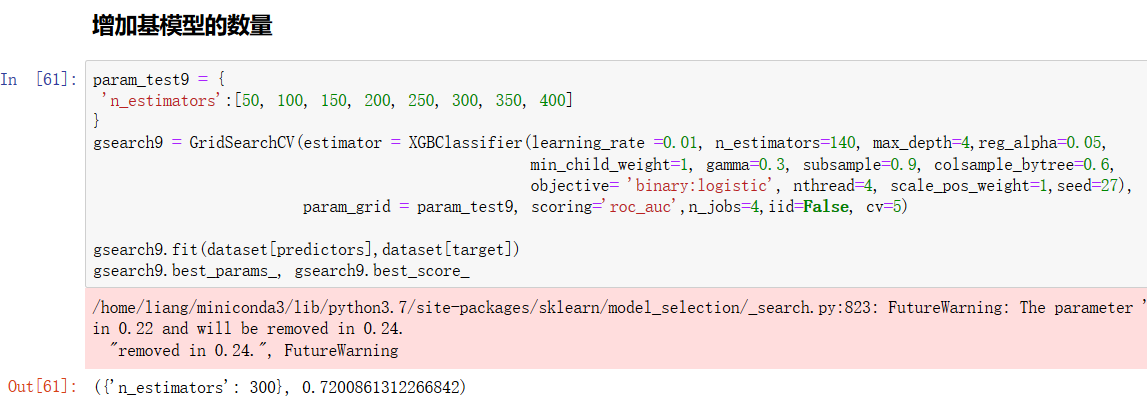


图42 基模型数量参数调优

最终得到调完参数后的最优模型的AUC为 0.720，相比于模型最初的参数提升了3~4个百分点。

##### 2.0使用Xgboost进行特征编码

Xgboost模型有一个apply方法，可以得到每个样本落在每个基模型（决策树）所在叶子节点的位置。用one-hot 编码对上一步得到的数据进行编码，便可以得到用作LR训练的数据，代码如图43。

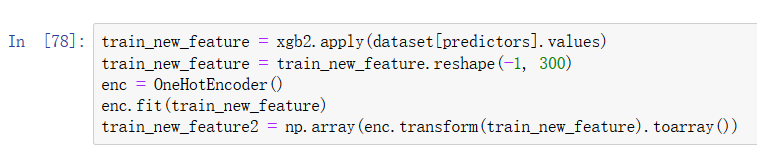


图43 使用Xgboost进行特征编码

##### 3.0使用编码后的数据训练LR模型，以AUC为指标进行模型调优

使用网格搜索进行LR模型的调参，参数搜索范围如表4，最后得到的最优参数如表5，模型调参代码如图42。

|  |  |
| --- | --- |
| 参数名 | 参数搜索范围 |
| C | 0.1到10之间随机取多个数 |
| max\_iter | 50到250，步长为50 |

表4 LR模型参数搜索范围

|  |  |
| --- | --- |
| 参数名 | 最优参数取值 |
| C | 0.1 |
| max\_iter | 100 |

表5 LR模型最优参数

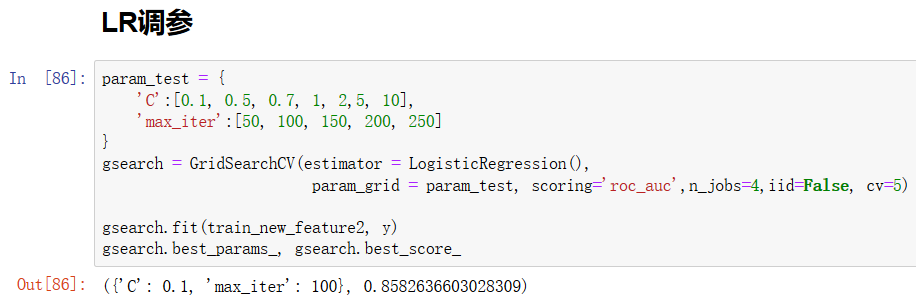


图44 LR模型参数调优

我们看一组训练得到的参数对比：

|  |  |
| --- | --- |
| 模型 | AUC |
| 没调参的Xgboost+没调参的LR | 0.8078 |
| 使用最优参数的Xgboost+没调参的LR | 0.8669 |
| 使用最优参数的Xgboost和LR | 0.8697 |

表6 模型调参前后的AUC对比

根据表6，我们显然可以得知，进行了网格搜索之后，模型的AUC提升显著，模型具有更好的分类效果。

最后，我们看一下，不同模型使用其本身的默认参数进行训练，其AUC、Accuracy、Log Loss 这三个指标的对比，如图45；以及不同模型的ROC曲线的对比，如图46。

图45 不同排序模型评估指标对比

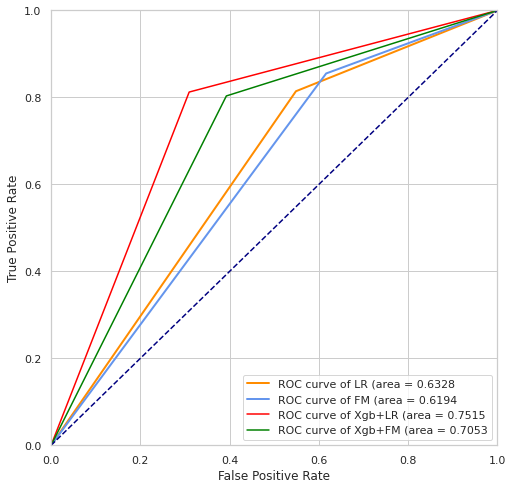
****

图46 不同模型的ROC曲线对比

我们可以发现，对比单个模型和融合模型，融合模型的效果优于单个模型；在单个模型中，FM模型的效果和LR相差不大；对比Xgboost+LR和Xgboost+FM，Xgboost+LR模型的效果好于Xgboost+FM，可以看到Xgboost+LR的模型拥有最高的AUC和最低的LogLoss，本人分析，可能是由于FM模型的隐因子向量的学习导致的过拟合。根据不同模型的评估，本推荐系统的排序模型决定使用Xgboost+LR模型。

### 3.6.3 词嵌入模型Word2Vec

在NLP中，最细粒度的东西是单词，在机器学习或深度学习问题中，我们往往需要将单词表以数学形式表示出来，可以使用one-hot的形式进行表示，但是这样做没有办法区分不同词语之间的含义，所以出现了一种词嵌入(word-Embedding)的方式，将整个预料库看作一个多维的空间，将语义接近的词语的位置嵌入到比较接近的位置。使用Word2Vec模型可以计算出这个语料库中各个词对应的词向量。

这里是Word2Vec模型使用的是Spark ml库里面的Word2Vec模型。数据的预处理步骤和模型的使用在功能实现里的相似古诗部分已经有[详细描述](#_3.4.1相似古诗)。

# 主要测试

本章主要介绍花满蹊的主要功能测试，包括运行速度、安全性、部署方便性和可用性。

（1）运行速度：

由于硬件限制目前花满蹊的性能并没有达到特别高的水准，不过也可以满足日常使用，其中web网页端要比微信小程序端速度稍慢，速度最慢的是web网页端的诗词页面，以此为样本我们做了40次实验，10次为一组每组平均速度如图47所示，总体平均速度为1.71s

图47

（2） 安全性：

对于用户端花满蹊采用Spring拦截器进行用户认证，未登录状态下访问敏感信息会提示先登录，经过测试可以满足需求，提示登录效果如图48所示：

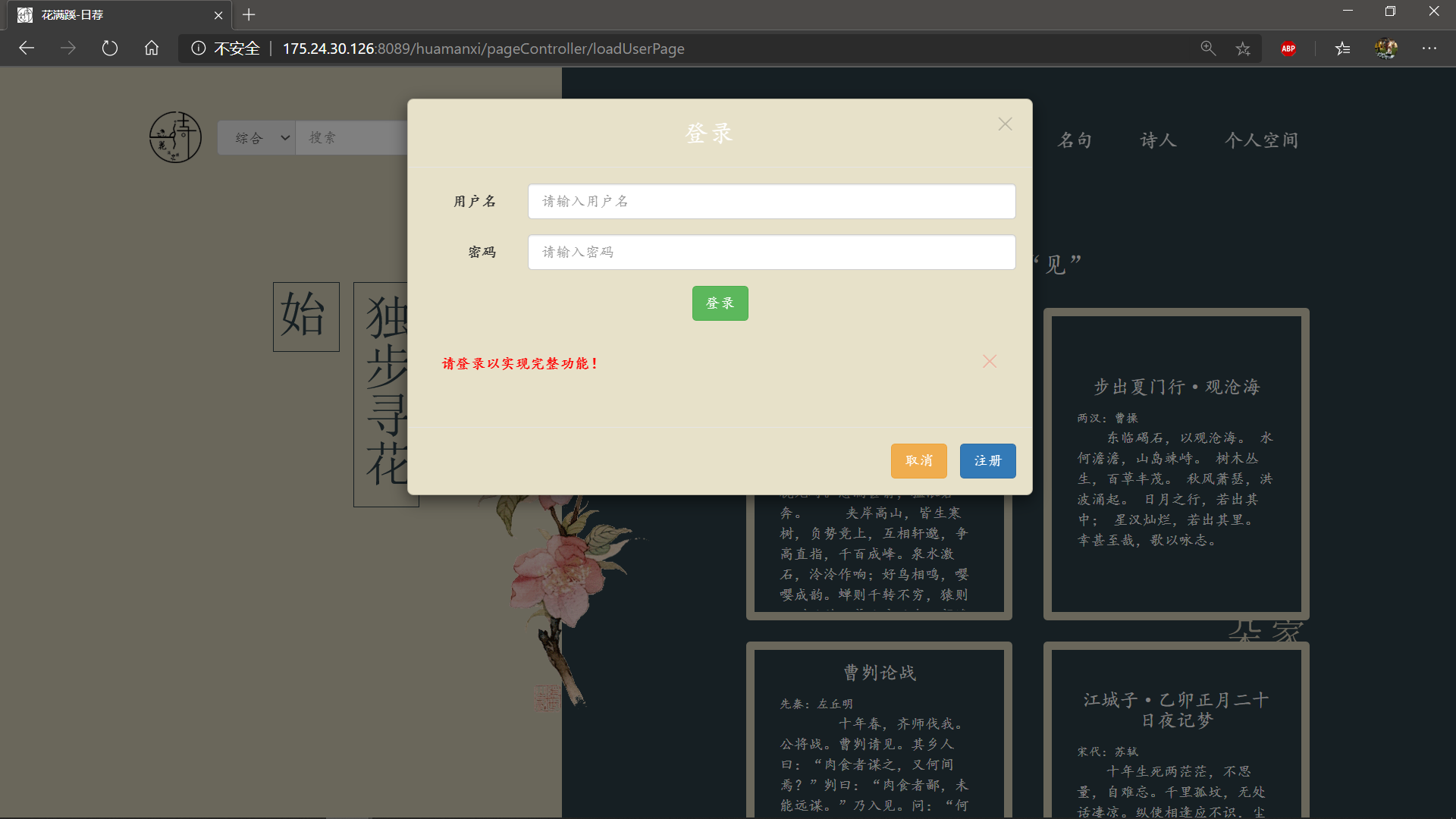


图48

管理员端采用springSecurity框架做用户认证，经过过测试未登录状态下不能使用。

1. 部署方便性：

此项目后台采用SpringBoot框架进行编写，其部署只需打成Jar包即可直接在拥有Java环境下的系统中运行。

1. 可用性：

花满蹊具有web网页端和微信小程序端可用性较强。

# 安装及使用

## 5.1 数据库数据导入

本作品关系型数据库使用的是Mysql，版本为5.5.40，数据库文件的导入方式可通过我们在素材文件中所提供的sql文件，利用可视化工具执行sql语句即可将本作品数据库文件导入。

## 5.2 Elasticsearch的安装配置

本作品全文检索使用的技术是Elasticsearch，版本为6.5.4，安装及使用流程如下：

① 将Elasticsearch的安装包解压三份至自己的服务器，修改配置文件，可参照所提供的配置文件

② 分别进入加压后的bin目录，启动服务，开启集群

③ 安装对应版本的logstash，执行所提供的author、poem、rhesis配置文件，注意将里面的数据库以及Elasticsearch集群ip和端口改成自己服务器的ip和端口，将诗词、诗人、名句信息导入Elasticsearch供查询使用

④ 完成安装、启动、导入数据即可使用

## 5.3 系统后台程序安装及使用流程

1. 安装环境：CentOS（带有java环境）、Windows（带有java环境）
2. web服务端安装流程：

web服务端采用Java语言编写，并且使用springBoot框架，安装流程如下：

1. 在集成开发工具IDEA中利用Maven将其打成jar包
2. 在CentOS系统中安装JDK并配置Java环境
3. 之后将jar包上传到CentOS系统中
4. 创建日志目录文件
5. 执行‘nohup java -jar 项目名 >日志文件名 2>&1 &’命令开启服务

安装流程图如49所示：



图49

（3）推荐系统安装流程

推荐系统部分的大数据组件的部署要部署到Linux系统上

## 5.4 个性化推荐模块安装及使用流程

### 5.4.1个性化推荐模块的安装

1. 安装Python3.6.0，JDK1.8，Scala2.11.12
2. 安装并配置好Hadoop-2.6.5，配置文件太多此处无法列出全部，参照官网即可使用
3. 安装并配置好Spark-2.2.0，配置文件太多此处无法列出全部，参照官网即可使用
4. 安装并配置好Flume-1.9.0，配置好Flume的路径，要运行的Flume文件如图17
5. 在Maven官网，搜索并下载好如下依赖，放入到某个的文件中，并在Spark的spark-defaults.conf配置文件中设置spark.executor.extraClassPath和spark.driver.extraClassPath的路径为依赖所在的路径，向文件中添加如图50的依赖包

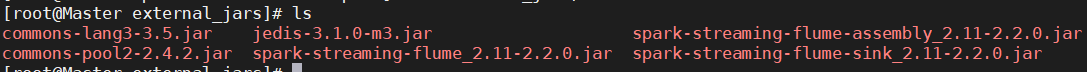


图50 向Spark中添加的依赖包

### 5.4.2个性化推荐模块的使用

1. 启动Redis、MongoDB数据库
   1. 在Redis的bin目录下使用命令：’redis-server redis.conf’ (redis.conf为自定义的redis配置文件)，启动成功如图51

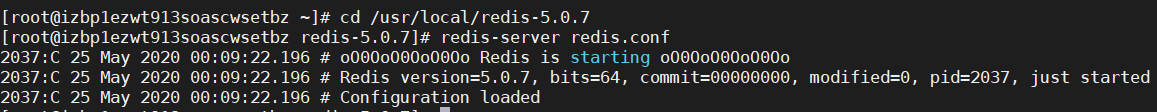


图51 启动Redis

* 1. 在mongodb的bin目录下使用命令：’mongod -f mongodb.conf’（mongodb.conf为自定义的mongodb配置文件），如图52

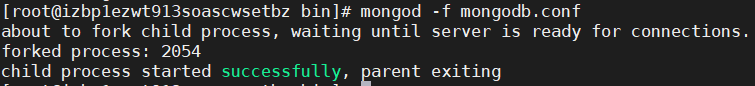


图52 启动MongoDB

* 1. 使用命令 ‘ps -aux | grep mongod’ 和 ’ps -aux | grep redis’ 检查是否成功启动这两个数据库，如图53

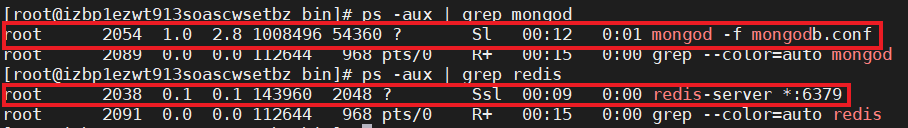


图53 检验Redis和MongoDB是否成功启动

1. 启动Hadoop
   1. 在Hadoop目录下使用命令 ‘sbin/start-all.sh’，如图54

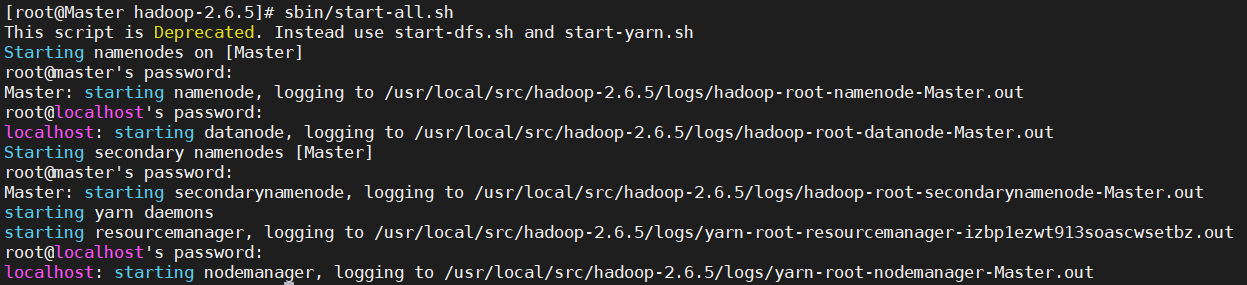


图54 启动Hadoop

* 1. 使用 jps命令查看是否存在 namenode 和 datanode 两个进程，有这两个进程说明Hadoop启动成功，启动成功如图55

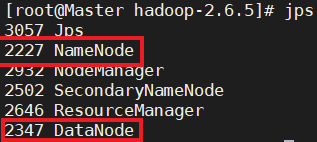


图55 检验Hadoop是否启动成功

1. 启动Spark-Streaming程序
   1. 在 Spark的bin目录下使用命令：’spark-submit --class 包名.类名 Spark-Streaming程序的jar包所在的路径’将打包好的Spark-streaming程序提交到Spark集群上运行，启动成功如图56

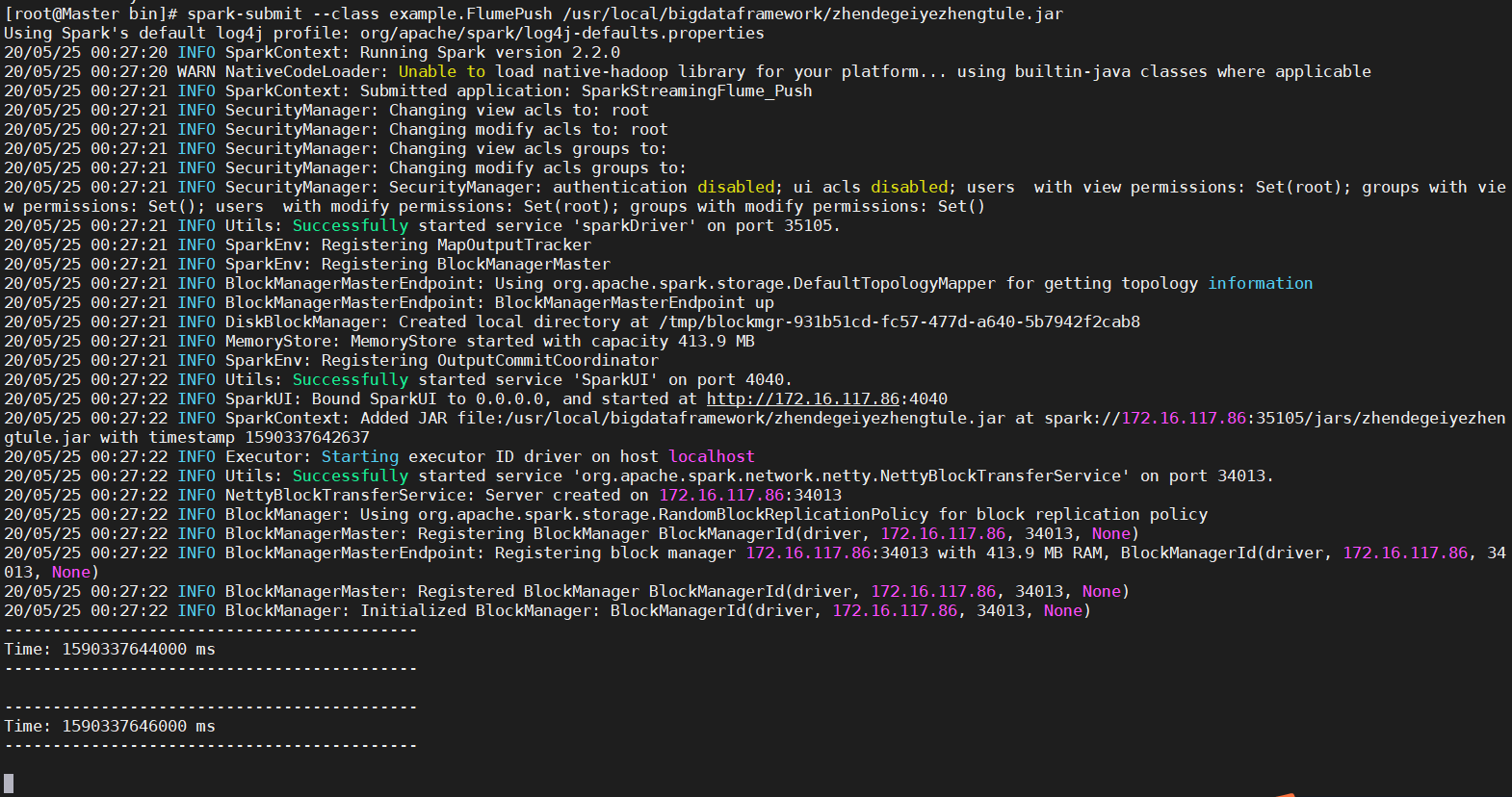


图56 启动Spark-Streaming程序

1. 启动Flume（注意，必须先启动Spark-Streaming）
   1. 使用命令：’bin/flume-ng agent --conf conf --conf-file 配置文件名 --name 配置文件设置的agent名 -Dflume.root.logger=WARN,console’，如图57

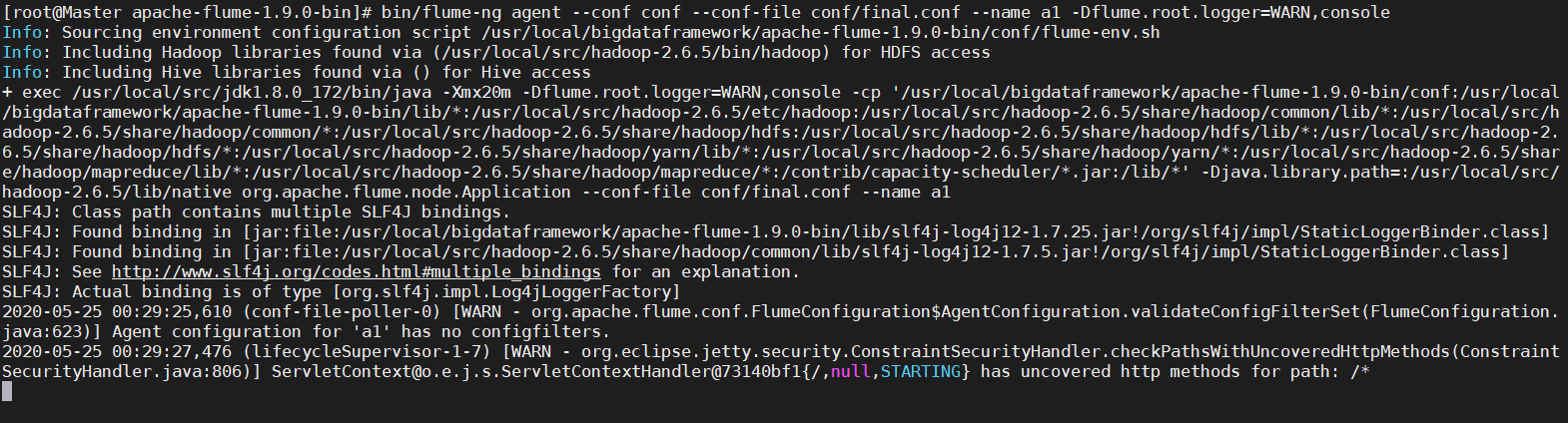


图57 启动Flume

1. 启动Django框架
   1. 在 http\_flume目录下使用命令：’python manage.py runserver 0.0.0.0:8008’

# 第六章 项目总结

本作品是在新冠疫情发生之际做成的，期间遇到到一些困难，比如队员之间的沟通协调远不如在一起方便，项目写好后测试时不方便等，不过在通过我们的努力这些困难都被一一解决了。通过此次比赛我们认识到了团队合作的重要性，意识到了明确的分工可以事半功倍，相信通过此次比赛以后我们在参加其他比赛时能够配合的更好，更加高效率的完成任务。

本作品也存在一些需要升级改进的地方：

1. 首先就是登录这一块可以引入第三方登录，这样可以使用户的体验更好；
2. 再者就是性能方面，web端采用的是MVC三层架构，并不是前后端分离，以后可以升级为前后端分离项目进一步提高性能，由于成本问题我们没有进行集群部署这一点当有足够服务器时也可以进行升级，采用集群部署；
3. 还有小程序部分，小程序的页面可以再加以修改使其更加美观。
4. 推荐模块的排序模型的使用可以使用更先进的深度学习模型，可能会有更好的效果。

感谢各位评委老师拨冗翻阅拙作，敬请斧正。