文章编号:2096-1472(2023)-06-59-04
DOI:10.19644/j.cnki.issn2096-1472.2023.006.013

基于 Spark 的电影推荐系统的设计与实现

刘 念,蔡春花

(黄河交通学院,河南 焦作 454000)
✉ n135691@foxmail.com; 786803383@qq.com

摘 要:针对大数据时代电影资源快速增长而出现的信息过载问题,采用 Spark 分布式计算平台,结合 MongoDB(分布式文档存储数据库)业务数据库、Redis(远程字典服务)缓存数据库,以及 Flume(日志收集系统)、Kafka(开源流处理平台)和 Spring 开源框架等工具设计并实现了一个电影推荐系统。系统主要包括数据采集模块、离线推荐模块、实时推荐模块和综合业务模块,其中离线推荐模块采用交替最小二乘(ALS)算法训练隐语义模型求得预测评分,实时推荐模块通过 Spark Streaming 流实时处理用户在可视化界面对电影的评分操作。经功能测试及兼容性测试表明,系统可依据用户行为偏好,快速准确地完成电影推荐,可用性较高。

关键词:离线推荐;电影推荐;Spark
中图分类号:TP391 **文献标识码**:A

Design and Implementation of Spark-based Movie Recommendation System

LIU Nian, CAI Chunhua

(*Huanghe Jiaotong University, Jiaozuo 454000, China*)
✉ n135691@foxmail.com; 786803383@qq.com

Abstract: To address the problem of information overload caused by the rapid growth of movie resources in the era of big data, this paper proposes to design and implement a movie recommendation system by using Spark distributed computing platform, combined with MongoDB (distributed document storage database) business database, Redis (remote dictionary service) cache database, Flume (log collection system), Kafka (open source stream processing platform) and Spring open source framework. The system mainly includes data collection module, offline recommendation module, real-time recommendation module and integrated business module. The offline recommendation module uses the Alternating Least Squares (ALS) algorithm to train the implicit semantic model for predicting ratings, and the real-time recommendation module processes users' movie ratings in the visual interface in real time through Spark Streaming. Function and compatibility tests show that the proposed system can quickly and accurately complete movie recommendations based on users′ behavior preferences, with high usability.

Keywords: offline recommendation; movie recommendation; Spark

1 引言(Introduction)

20 世纪 90 年代以来,个性化推荐系统受到了全球各大在线平台的关注和研究。与此同时,逐渐出现了各种不同的推荐系统及推荐算法。2014 年,RAMEZANI 等[1]提出了一种基于用户兴趣模式,寻找邻居用户的方法,该方法改善了传统推荐算法在稀疏性和计算上的问题。2019 年,KIM 等[2]为降

基金项目:黄河交通学院校级一流课程建设项目“Linux 原理及应用”(HHJTXY-2021ylkc13);黄河交通学院校级教学资源库建设项目“NoSQL 数据库原理”(HHJTXY-2022kczyk103).

低评分相关数据的稀疏现象提出了基于免疫网络的推荐算法，该算法采用收缩评分矩阵的方法，在降低数据稀疏性方面的效果显著。

随着互联网时代的推进，用户对推荐系统的需求呈现多样化的趋势，只有单一推荐算法的推荐系统在推荐质量和计算效率上已经无法满足时代的需求。国内学者对推荐系统的研究从未停止，宋文君等[3]提出的改进混合推荐算法，应用在 Netflix 数据集上的推荐结果准确性平均提高了 4.22%，而且降低了大规模数据所引发的计算复杂度。王全民等[4]提出的综合推荐法，是利用人口统计资料与评分资料求得系统内物品的相似性；综合推荐算法表现出较好的推荐质量。

观看电影已经成为一种重要的娱乐方式，人们在闲暇时间会在视频软件中搜索热播的影视节目。由于现有的视频软件电影分类不够健全，因此有必要设计了一个根据用户偏好和通过简单操作就能获得电影推荐的系统。基于此，本文设计一个基于 Spark[5]的电影推荐系统，该系统根据用户推荐需求，采用基于模型和内容的推荐算法，使用 MongoDB 业务数据库、Redis 缓存数据库，以及 Flume、Kafka、Spring 等工具，实现电影推荐功能。

2 系统设计(System design)

2.1 总体架构设计

本节主要介绍系统的总体架构，本设计采用大数据框架中最流行的 Spark 分布式计算引擎，结合 MongoDB 数据库、Flume、Kafka 及 Spring 等工具搭建一个支持处理海量数据的离线和实时的电影推荐系统。系统总体架构设计如图 1 所示。

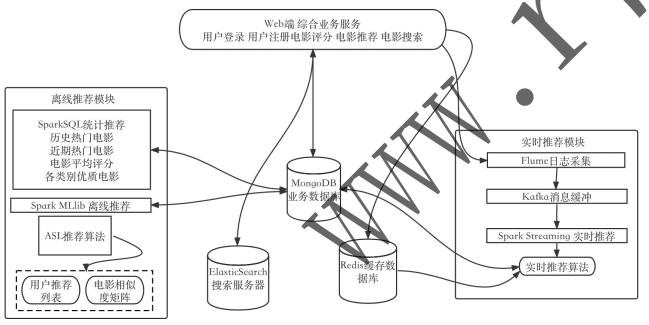

图 1 系统架构图

Fig. 1 System architecture diagram

该推荐系统是基于 Web 开发的 B/S 系统，其业务逻辑架构是由 Web 前端响应后台业务请求的 Spring 框架创建的，采用 MongoDB 管理业务数据库；采用 Elastic Search 作为检索服务器实现匹配查询操作；采用 Redis 作为缓存数据库实现实时推荐模块里对数据的提取。系统基于 Spark 平台，运用到了 Spark 部分生态组件。采用 Flume 进行日志收集业务以及 Kafka 进行消息缓冲操作，使用 Spark Streaming 将 Kafka 处理后的实时数据合并更新到 MongoDB 数据库。

如图 1 所示，系统的 Web 端综合业务服务模块包括用户登录、用户注册、电影评分、电影推荐和电影搜索功能，主要负责将后台业务数据，如图片和文字等展现给用户，为用户展示推荐结果，实现用户与系统的交互。

后台的业务除了响应前端的数据处理，还应把数据写入对应的数据库。系统使用的核心业务数据库是 MongoDB，原始数据全部存在 MongoDB 中，然后从里面读数据，读完数据做分析计算，计算后得到的推荐结果也写入 MongoDB。系统采用 Elastic Search 的匹配查询功能和 Redis 缓存数据库存放用户最近的几次电影评分，可使系统在实时推荐时快速获取所需数据。

数据采集使用 Flume 技术，将用户在商业平台上对某一影片的评分行为及其数据进行收集，并将其实时传输至 Kafka 集群。本系统使用 Kafka 作为流数据缓存部件，用于接收 Flume 的数据收集，并把资料传送至实时推荐系统。使用 Redis 作为缓存数据库的目的是为了支持实时推荐系统中的数据快速采集。

离线推荐模块，首先进行离线的统计推荐服务，由于需要加载的数据量较大，Spark SQL 会对所获得的数据进行一系列的数据预处理，与 MongoDB 和 Elastic Search 有对应的连接方式，只需把定义好的数据直接写入，保存在数据库中供系统调用，这就是数据加载的过程。写入数据后做离线统计服务，这部分涉及四个常见的统计量，包括历史热门电影、近期热门电影、电影平均评分和各类别优质电影，这部分的数据从 MongoDB 读取并做处理，对应生成各自的推荐列表，最后写回 MongoDB，由于统计推荐为用户推荐当前主流电影和经典电影，所有用户看到的推荐列表都是一样的，所以又叫做非个性化的离线统计推荐。然后进行个性化的离线推荐服务，这也是一个协同过滤的推荐，利用 Spark MLlib 库的 ALS(Alternating Least Squares，交替最小二乘)算法对用户评分数据通过隐语义模型做预测评分，然后通过预测评分求电影之间的相似度，得到的结果是用户推荐列表，最后把结果写回业务数据库。

实时推荐模块的流程为用户在系统电影详情页面进行电影评分，用户请求发送至业务后台，电影数据先写入 MongoDB，使用 Flume 技术从综合业务后台日志环境里收集日志，然后推送到 Kafka 集群上做消息队列的缓冲，待 Kafka 做完处理后，Spark Streaming 从 Kafka 主题拉取 MongoDB 和 Redis 收集的用户行为数据，对预处理后得到评分数据进行实时推荐，结果写回 MongoDB。对应的搜索服务要结合 MongoDB 和 Elastic Search 检索服务器，推荐结果的查询需要从数据库获取数据，最后返回前台。

2.2 系统功能设计

电影推荐系统的核心是针对不同的用户进行不同类型的电影推荐，那么相应的系统功能需求也是围绕着推荐业务服

务展开，具体要具备的功能要求如图 2 所示。

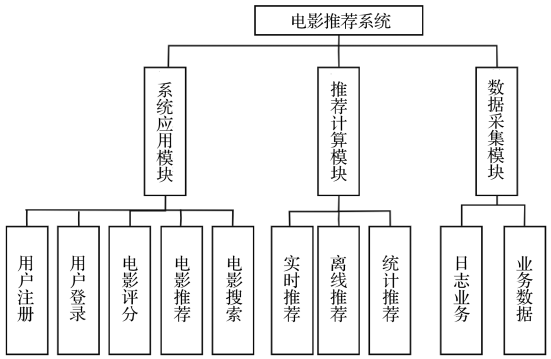

图 2　用户功能模块框架图

Fig. 2 Framework of user function module

2.3　数据库设计

电影推荐系统使用了三种数据库，分别是 MongoDB 业务数据库、Redis 缓存数据库和 Elastic Search 检索服务器。其中，MongoDB 是系统的核心数据库，数据源可以以文档的形式存储在 MongoDB 数据库中，也可以自定义数据结构，适合海量数据存储，除了存储数据源，还能存储用户信息表、各种推荐指标的电影推荐列表和通过推荐算法得到的推荐矩阵列表，例如电影信息表(Movie)、电影评分表(Rating)和电影标签表(Tag)。电影信息表(表 1)主要记录每部电影的关键信息以及影响用户选择的多种因素，进而为实现个性化推荐提供数据支持，同时可以完成统计类推荐。电影评分表(表 2)用来记录用户在过去一段时间内的评分行为信息、具体评分时间和行为内容。该表的数据来源于日志数据。电影标签表(表 3)用来记录每个用户关注的电影集合及其类别，数据源主要是为个性化的推荐服务提供支持。

Redis 缓存数据库用于存放用户近期对影片的评分数据，作为支撑实时推荐算法的数据源，当用户对电影进行评分时，后台服务记录数据后，将数据推送至 Redis 缓存数据库用于计算电影的预测评分。

Elastic Search 检索服务器主要存放一些搜索业务需求产生的数据，系统利用其强大的匹配查询能力实现基于内容的推荐服务。

表 1　电影信息表

Tab.1 Movie table

字段名	字段类型	是否允许为空	约束条件	字段描述
Mid	VARCHAR	否	主键	电影 ID
Name	VARCHAR	否	外键	电影名称
Descry	VARCHAR	否		电影描述
Timelong	VARCHAR	否		电影时长
shoot	VARCHAR	否		电影拍摄时间
Issue	VARCHAR	否		电影发布时间
language	VARCHAR	否		电影语言

表 2　电影评分表

Tab.2 Rating table

字段名	字段类型	是否允许为空	约束条件	字段描述
Uid	VARCHAR	否	主键	用户 ID
Mid	VARCHAR	否	外键	电影 ID
score	VARCHAR	否		电影分值
timestamp	TIMESTAMP	否		评分时间

表 3　电影标签表

Tab.3 Movie tag table

字段名	字段类型	是否允许为空	约束条件	字段描述
Uid	VARCHAR	否	主键	用户 ID
Mid	VARCHAR	否	外键	电影 ID
Tag	VARCHAR	是		电影标签
timestamp	TIMESTAMP	否		评分时间

3　系统实现(System implementation)

3.1　系统环境介绍

本系统主要在 Windows 和 Linux 操作系统上运行。系统的开发环境为 CentOS7.0、JDK1.8 版本、ZooKeeper 分布式服务管理框架 3.4 版本、Flume 日志采集工具 1.7.0 版本、Kafka 分布式消息队列 2.11 版本、MongoDB 分布式文件存储数据库 5.0.4 版本、Key-Value 数据库 Redis4.0.2 版本、Elastic Search 搜索服务器 5.6.2 版本。主要开发工具为 IntelliJ IDEA，推荐系统模块的开发语言主要为 Java 和 Scala。物理机的硬件配置为 16 GB 内存、500 GB 硬盘，CPU 为 Intel Core i7-11800H。

3.2　Spark 集群搭建

本系统基于 Spark 平台对海量数据进行存储和分析，但受经济条件的约束，实际开发过程中只配置了一台物理机，所以采取 Spark 的伪分布式集群安装模式，在一台物理机上模拟多机，通过虚拟机技术实现小规模的集群，其中 Spark 由 4 个虚拟节点组成(1 个 Master 节点和 3 个 Slave 节点)。以下为伪分布式 Spark 集群的搭建步骤。

(1)首先在物理机上安装 VMware 虚拟机软件，然后在虚拟机中安装 CentOS 操作系统的镜像文件，在 Linux 系统中安装 JDK，配置其环境变量，主机名为 hadoop102，重复上述步骤，在虚拟机中再次安装、运行 2 个 CentOS 系统，主机名分别设置为 hadoop103、hadoop104。

(2)在集群中配置 ssh，生成公钥，实现 3 台客户机的相互免密码登录、修改对应配置文件及加入集群节点的主机名。配置 Spark 运行的 IP 地址、端口号、JDK 位置和集群对应的 Master 节点等信息。

(3)以独立部署模式 Standalone 为例，使用 Spark 自身节点运行集群模式，执行脚本命令 sbin/start-all.sh，查看 3 台

服务器运行进程，可以看到 1 个 Master 节点和 3 个 Slave 节点均正常运行。Spark 运行图如图 3 所示。

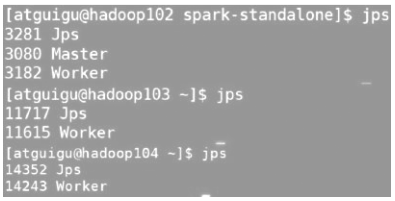

图 3 Spark 运行图

Fig. 3 Spark operation diagram

3.3 数据采集模块

电影推荐系统采用 Fulme 作为实时日志数据采集引擎，该引擎包括三个组件，即 Source、Channel 和 Sink，Source 是指数据的来源和方式，Channel 是一个数据的缓冲池，Sink 定义了数据输出的方式和目的地[6]。使用 Flume 将日志数据更新到 Kafka 队列，实现流式的日志处理，使用 Spark Streaming 流式实时处理技术，从而完成日志实时解析的目标。Flume 组件结构如图 4 所示。

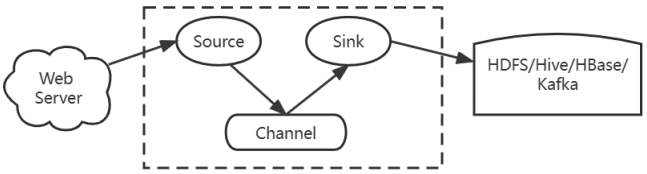

图 4 Flume 组件

Fig. 4 Flume component

Kafka 配置如下：

```
//输入和输出的 topic
string from = "log" ;
String to = "recommender";
//定义 Kafka Streaming 的配置
Properties settings = new Properties();
settings.put(StreamsConfig.APPLICATION_ID_CONFIG,"logFilter");
settings.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, brokers);
settings.put(StreamsConfig.ZOOKEEPER_CONMECT_CONFIG, zookeepers);
//创建 Kafka Stream 配置对象
StreamsConfig config = new StreamsConfig(settings);
//创建一个拓扑建构器
TopologyBuilder builder = new TopologyBuilder();
//定义流处理的拓扑结构
builder.addSource( "SOURCE",from)
        .addProcessor("PROCESSOR",()->new LogProcessor(), "SOURCE")
        .addSink( "SINK",to,"PROCESSOR");
KafkaStreams streams = new KafkaStreams( builder,config );
```

3.4 推荐计算模块

3.4.1 离线推荐

基于模型的协同过滤算法(ModelCF)根据已有的用户评分数据进行学习并训练模型，进而预测空白的用户和物品之间的评分关系。实际上，ModelCF 算法可以称作是基于用户的协同过滤算法(UserCF)和基于物品的协同过滤算法(ItemCF)的混合形式，同时分析了用户和物品两个方面，让其更加节省空间，并且训练和预测速度也更快，推荐结果也更加准确。交替最小二乘(ALS)矩阵分解的协同过滤算法[7]就是典型的基于模型的协同过滤算法。

该矩阵是用户与物品之间的关系矩阵，其数值代表用户对物品的看法，数值可为布尔值。1 代表曾做过的事，例如喜欢、点击、购买，0 代表没有，此矩阵称为“行为矩阵”。此外，矩阵的数值还可以是一个表示回馈的分数，例如 0—5 的数值，这个矩阵叫做“分数矩阵”。由于用户无法对所有项目进行操作，因此该矩阵是稀疏的。基于模型的协同过滤算法是通过训练出一个参数模型描述用户与物品、用户与用户或者物品与物品之间的关系，然后通过优化过程得到模型参数。离线推荐算法的执行流程如图 5 所示。

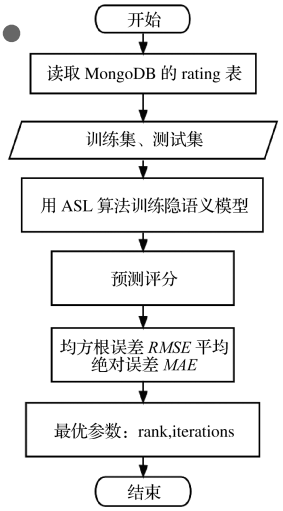

图 5 离线推荐算法流程

Fig. 5 Offline recommendation algorithm flow

3.4.2 实时推荐算法

基于内容的推荐算法[8]的具体流程为加载电影内容信息数据，提取电影 ID(Mid)、电影名称(Name)和电影所属类别(Genres)3 项作为电影原始内容特征；实例化一个分词器，默认按空格分词，用分词器对原始数据进行转换，生成新的数据结构；引入 HashingTF 工具，把一个词语序列转化成对应的词频 TF(Term Frequency)；引入逆文档频率 IDF(Inverse Document Frequency)工具，训练 IDF 模型得到一个词的逆文档频率；用模型对原数据进行处理，得到文档中每个词的 TF-IDF[9]，作为新的特征向量；通过特征向量计算电影相似度矩阵，通过实时推荐算法计算基于内容的实时推荐结果。

(下转第 45 页)