**综合实验： 实现推荐系统**

**实验目的:**

（1）掌握Lambda架构

**实验平台:**

操作系统：CentOS 7.6 64bit

**实验内容：**

（1）基于Lambda架构的大数据推荐系统架构概述

Lambda Architecture（LA）最早是由Twitter工程师Nathan Marz提出来的，它是一种大数据软件设计架构，其目的是指导用户充分利用批处理和流式计算任务各自的优点实现一个复杂的大数据处理系统。通过结合这两类技术，LA可以在延时、吞吐量和容错之间找到平衡。

Lambda架构的一个典型应用是推荐系统。在互联网行业，推荐系统被应用在各个领域，包括电子商务、视频、新闻等。推荐系统的设计目的是根据用户的兴趣特点和购买行为，向用户推荐感兴趣的信息和商品。推荐系统是建立在海量数据挖掘上的一种高级商务智能平台，以帮助商家为顾客购物提供完全个性化的决策支持和信息服务。

图1为一个最小化的推荐系统原型。在该原型中，用户产生的数据会被Flume收集，经Kafka传输，被流式计算任务（如Spark Streaming等）和存储组件（HBase）消费。批式处理和流式处理会利用用户产生的数据计算特征与训练预测模型，两者的不同点在于批处理利用HBase中存储的数据（通常为某个时间点之前的所有历史数据），流式处理利用Kafka中最近一段时间内的数据。最后，得到的计算特征与预测模型等都会存储在Redis数据库中，供服务端使用。服务端在接收到客户端的访问服务请求后，会根据Redis存储的特征与预测模型参数，计算最符合用户请求的商品列表，并返回给客户端。客户端再根据用户的行为反馈（比如点击/不点击）向Flume发送日志，形成闭环。同时，在系统后台可以利用Hive/Spark SQL对HBase中的数据进行监控，并进一步完成数据的分析。

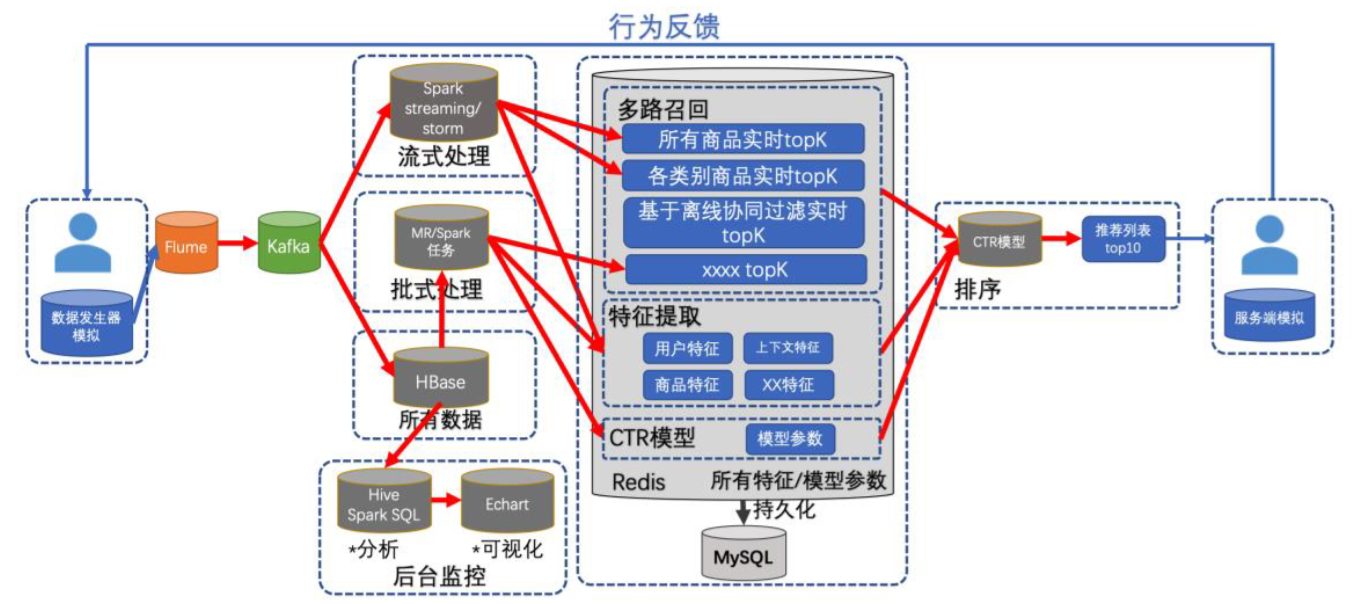


图 1 基于Lambda架构的推荐系统

推荐系统的核心是推荐算法，推荐算法通常会根据用户的兴趣特点和历史行为数据构建推荐模型，以预测用户可能感兴趣的信息和商品，进而推荐给用户。推荐算法通常可以划分为两个阶段①召回阶段②排序阶段，如图2所示。召回阶段就是从所有商品中，粗筛选出一些候选的商品，例如根据用户的兴趣标签TopK、社区热门商品TopK、某个算法模型（如协同过滤模型）的推荐列表TopK。从多种途径获取候选列表也称为多路召回。排序阶段就是根据召回阶段粗筛选出来的候选商品，将之前批式处理、流式处理计算好的特征输入到训练好的点击率预测模型（也称为CTR模型）中，得到用户点击每个候选商品的概率并进行排序，最后将点击率最高的TopK个商品返回给客户端。

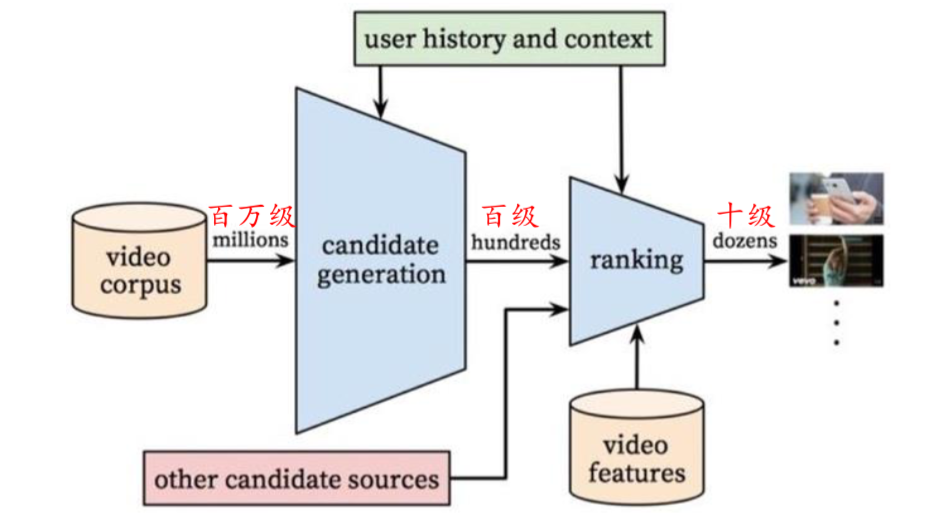


图 2 一个视频推荐算法流程（图片来自网络）

下面将分别对每个组件所需要完成的功能进行讲解。

（2）使用Flume、Kafka实现数据的收集

由于一些原因，本次实验将略去Flume组件。启动一个Kafka服务器，并创建名为“movie\_rating\_records”的Topic。

（3）使用HBase实现原始数据的存储

在HBase中创建一个表并命名为“movie\_records”，数据格式在附加说明（2）部分。编写一个Kafka消费者，定期从“movie\_rating\_records”中获取数据并写入HBase。

（4）使用Redis作为缓存数据库实现特征/模型参数的存储

Redis是一个开源的、可基于内存也可持久化的Key-Value的NoSQL数据库。其性能极高，读写速度可达10W/qps。同时，Redis在2015年的redis3.0+版本中加入了 redis cluster，可以实现分布式集群的架构，将原本单机的数据进行分区，分到若干个子集中。

Redis不仅可以实现高效读写，还可以控制数据的过期时间，因此可被用于大数据组件之间中间结果的缓存。在本次实验中，将使用Redis作为批式、流式、服务等组件之间的数据交换中间件。例如，将批式计算的得到的特征写入Redis中，等待模型训练时再从Redis读取特征。

（5）使用MapReduce或Spark实现定时启动的批式计算任务

使用MapReduce/Spark启动定时任务，定时时间为5分钟（5\*60\*1000ms），计算以下两部分内容：

①历史特征的计算

从HBase中读取所有历史数据，并计算以下五个批式特征。

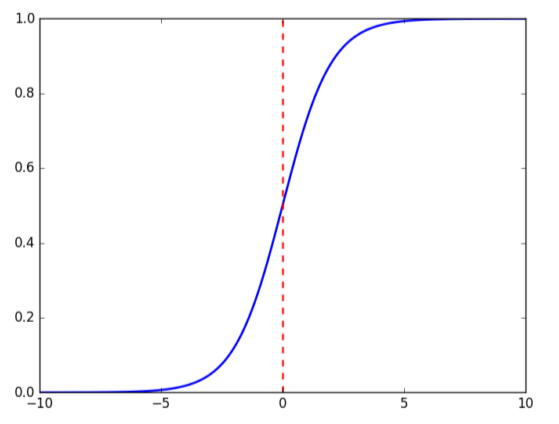
|  |  |
| --- | --- |
| batch2feature\_userId\_rating1\_{userId} | 某用户历史正反馈次数 |
| batch2feature\_userId\_rating0\_{userId} | 某用户历史负反馈次数 |
| batch2feature\_userId\_to\_genresId\_{userId}\_{genreId} | 某用户历史点击某分类的比例 |
| batch2feature\_movieId\_rating1\_{movieId} | 某电影历史被正反馈次数 |
| batch2feature\_movieId\_rating0\_{movieId} | 某电影历史被负反馈次数 |

②CTR预测模型的训练

Logistic Regression算法是机器学习常用的有监督学习算法。可以将Spark MLLib中的Logistic Regression模型作为本次实验的CTR模型。

线性回归是监督学习中最简单的模型之一，该模型通过训练一条直线来拟合样本数据。其公式如下：

其中，表示样本的第个特征，表示该特征在模型中的参数，表示偏置值，表示线性回归模型对于该样本的预测值。在训练阶段，通过已知的样本和真实值计算与。在预测阶段，对于未知真实值的样本，使用模型参数与计算预估样本的真实值。但线性回归没办法完成二分类任务（比如CTR模型中的点击与不点击就是一个二分类问题）。要解决这个问题只需要找到一个单调可微的函数将线性预测的结果映射到（0，1）区间上。例如sigmoid函数，它的函数图像如下：



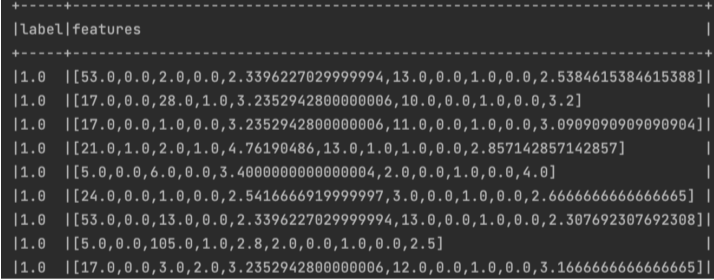
将线性回归的预测值再进过sigmoid函数，即，就得到了逻辑回归。

Spark MLlib中的spark.ml.LogisticRegression类接受的训练数据的形式为一个DataFrame，其中包括两列：

label：DoubleType（表示数据的标签）

features：VectorUDT（表示数据的特征）

如下图所示



特征可以划分为批式特征和流式特征。需要注意的是，在模型训练阶段，流式特征从历史数据（HBase）模拟得到（例如从某条记录的前1000条记录中提取该记录的流式特征），而在模型预测阶段，从Redis中获取流式处理得到的流式特征。

训练结束后，可将模型参数（lrModel.coefficients、lrModel.intercept）取出并保存在Redis中，以供服务端使用。

（6）使用Spark Streaming或Flink实现流式计算任务

不同于批式计算，流式计算直接从Kafka中消费“movie\_rating\_records”Topic，定时时间为30秒（30\*1000ms）并完成以下两部分内容的计算：

①实时特征的计算

窗口从Kafka中读取实时数据，并计算以下五个流式特征，具体如下表所示。

|  |  |
| --- | --- |
| streaming2feature\_userId\_rating1\_{userId} | 某用户最近正反馈次数 |
| streaming2feature\_userId\_rating0\_{userId} | 某用户最近负反馈次数 |
| streaming2feature\_userId\_to\_genresId\_{userId}\_{genreId} | 某用户最近点击某分类的比例 |
| streaming2feature\_movieId\_rating1\_{movieId} | 某电影最近被正反馈次数 |
| streaming2feature\_movieId\_rating0\_{movieId} | 某电影最近被负反馈次数 |

②实时TopK的计算

窗口从Kafka中读取实时数据，计算出现此处TopK的movieId和每个电影类别中出现TopK的movieId，并存储Redis中。

|  |  |
| --- | --- |
| popular\_movies\_all | 所有电影TopK |
| popular\_movies\_genreId\_{genreId} | 某类别电影TopK |

（7）实现服务应答

服务端根据客户端的请求，从Redis中读取多种召回方式得到的候选列表、批式特征、流式特征、CTR模型参数，根据CTR模型参数与特征，为列表中每个电影打分，并将评分最高的TopK部电影返回给客户端。

（8）（选做内容1）实现简易后台监控（20分）

一个系统通常还需要有后台监控和分析能力。可以从以下两方面考虑：

①使用Hive/SparkSQL实现简易后台分析统计（10分）

定期统计：不同类别的电影的记录数

②将后台分析统计结果可视化（10分）

将①的结果可视化，使用条形图等进行可视化展示，并定期更新图表。

（9）（选做内容2）提升推荐系统的推荐效果（20分）

①增加更多的召回（10分）

使用多种方式进行召回，如spark.mllib中的协同过滤模型，对每个userId产生单独的召回列表。

②使用更多特征（10分）

进一步从数据中进行挖掘，例如使用上电影的年份等信息。

（10）（选做内容3）Redis的改进（20分）

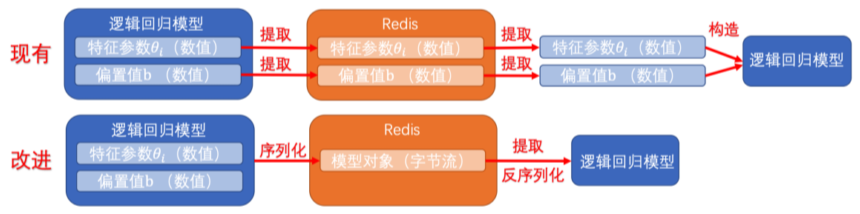
①分布式Redis的部署（6分）

Redis支持集群化部署，通过增加Redis集群的结点可以增加对Redis的读取访问速度。本改进要求将Redis改为集群部署。

②对于存储在Redis中的特征和数据，设置一个合理的过期时间并说明自己的思考（6分）

③CTR模型序列化后直接存储在Redis中（8分）

本次实验中使用LR模型，由于Redis不支持对象存储，想要在Redis中存储该模型，可以通过将模型参数，（都为数值型）从模型中提取出来，再分别存储到Redis模型中，然后服务端从Redis中提取这些模型参数，并根据这些模型参数重新构造模型用于预测。但对于拥有较多模型参数的算法模型来说，这种做法过于繁琐。更一般的做法是，基于Redis支持二进制字节流存储的特性，将整个模型对象序列化后直接存到Redis中，在服务端使用时再从Redis中提取并反序列化模型对象，如下图所示。



（11）（提高部分）（20分）

经过多年的发展，lambda架构逐步稳定，可以满足大部分需求，但lambda架构具有一定的复杂性，主要是由于：批处理端和流处理端都需要不同的代码库，需对其进行维护并保持同步。然而，试图将代码库抽象到单个框架中会使许多专门的批处理和实时生态系统工具实现过程较为复杂。同时，随着数据的不断膨胀，批式处理也面临一定的压力。在这样的背景下，产生了许多新的大数据软件架构。

kappa架构使用单一代码库的纯流式方法。在关于采用纯流式处理方法的优点的技术讨论中，有人指出，使用灵活的流处理框架(如Apache Samza)可以提供一些与批处理相同的优点，而且没有延迟。这样的流式框架可以收集和处理任意数量的数据，适应阻塞和处理状态。

delta架构则引入了一个表存储层，它处理流存储和表存储，并通过单个代码库进行访问。典型的实现是Delta Lake，它是一个开源的基于云对象存储的ACID表存储层。

①调研现有的大数据软件架构，并撰写调研报告/综述（需要附上引用）（5分）

②搭建新的架构（包括但不限于上面提到kappa架构和delta架构）来完成本次实验（推荐系统）（15分）

**附加说明：**

（1）数据集说明

包含三个文件：movies.csv,json\_train\_ratings.json,json\_test\_ratings.json。

① movies.csv：存储电影信息，包括电影ID，电影title，电影genres，电影year，以及属于各个类别。在movies.csv中，一共出现了19个类别的电影，具体见数据文件。在系统运行前，通过提供的load\_movie\_redis.py组件将这部分数据导入到 Redis中。

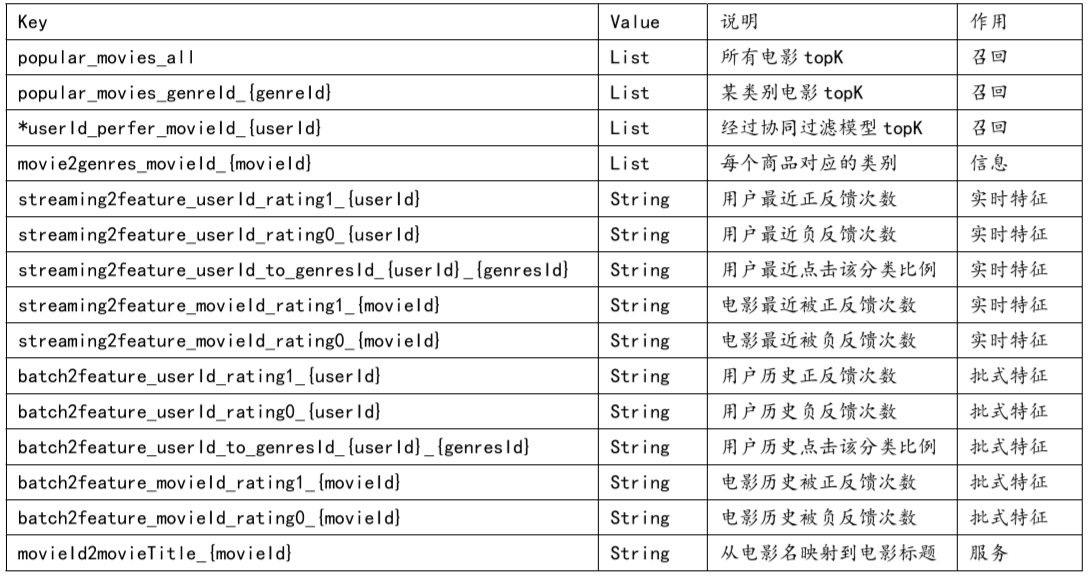
② json\_train\_ratings.json：用于模拟历史数据。在系统运行前，通过提供的load\_train\_rating\_hbase.py组件将这部分数据导入到HBase中。

③ json\_test\_ratings.json：用于模拟实时数据。在系统运行时，通过提供的generatorRecord.py组件发送到Kafka。

（2）HBase中的数据格式

HBase中创建一张表 movie\_records，该表包含一个列族“details”,该列族包含 4 个列，“userId”,“movieId”,“rating”,“timestamp”

（3）Redis中的数据格式



注意事项

1. 本次实验是小组的形式，一个小组不多于2人。
2. 本次实验报告的提交截止时间为第18周，即6月27号前（不包括6月27号）
3. 本次实验的最终分数由以下几部分组成：

* 基础部分（60分）
* 选做部分（20分，最多选做一个内容）
* 提高部分（20分）

1. 本次实验严禁抄袭
2. 本次实验的邮件名与实验报告的命名格式都为“姓名-学号-综合实验报告”。例如，

邮件名：赵四-2018222333-综合实验报告

邮件正文：（空）

邮件附件：赵四-2018222333-综合实验报告.doc

注意命名格式中各字段的相对顺序与使用的分隔符。除此之外，不要出现多余的字眼，如“大数据基础实验”，多余的双引号，多余的班级信息等。不按照要求提交将会扣除部分分数。