2.8 文章相似度增量更新

2.8.1 增量更新需求

- 13万文章
- 用户每天都会发表新文章
  - python 1 万， 自媒体用户A发表 2 机器学习文章
- B用户：python 推荐， 2 机器学习文章
  - 点击两篇新文章
  - 刷新？ 2 机器学习文章 相似文章
- **批量新文章，需要与历史数据进行相似度计算**

离线：一直运行：flume收集， 新闻文章画像、相似度定时更新的程序

3.1 用户画像计算更新

3.1.1 为什么要进行用户画像

而构建用户画像，不仅可以满足根据分析用户进行推荐，更可以运用在全APP所有功能上。

3.1.2 如何构建用户画像

数据源分析

- 静态数据：用户相对稳定的信息，如图所示，主要包括人口属性、商业属性等方面数据
- 动态数据：用户不断变化的行为信息

用户画像的生产

- 用户标签数据收集，通过数据收集工具，如Flume或自己写的脚本程序，把需要使用的数据统一存放到Hadoop集群。
- 用户标签数据清理
- 用户标签数据合并，把用户通过各种数据源提取的特征进行合并对于合并后的结果数据

用户画像构建层次

- 用户画像存储：Hbase, 基于大数据lambda这套框架做推荐, hbase

3.1.2 黑马头条推荐系统用户画像计算设计

黑马用户画像

为了达到更加细致的画像建立，我们给每个频道每个用户都建立标签

用户的兴趣喜好，按照黑马频道区分

3.2 用户行为数据处理与画像计算

3.2.1 增量用户行为日志处理

- userClick.log——>flume——>hdfs (profile, user_action)
- user_action:每一条数据就是用户的一次行为

用户的兴趣喜好？？？？ Python：A 操作行为

目的：首先对用户基础行为日志进行处理过滤

- profile:从user_action—>user_article_basic表
- 1、创建HIVE基本数据表
- 2、读取固定时间内的用户行为日志
  - user_action: hdfs:2019-06-05
  - 增加某个时间分区，跟表进行关联，否则读取不到数据
  - 1、user_action: 没有分区读取不了数据
  - 2、user_action:关联一个日期，但是2019-03-21，在hdfs当中没有文件，user_action读取2019-03-21数据会报错
  - hive与hdfs
- 3、进行用户日志数据处理

思路：按照user_id的行为一条条处理，根据用户的行为类型判别

- 建立：一个用户对应一篇文章， A 10 10行，
  - `sqlDF.rdd.flatMap`与map，mapPartitions？？ 区别
  - 返回新的rdd， flatMap处理一行数据，可以返回多行结果
  - map，mapPartitions：处理一行只能返回一行结果

actionTime	readTime	channelId	articleId	algorithmCombine	action	userId
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 08:31:44		0	[14805, 15196, 44...	C2	exposure	1113004557979353088
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:23:59		0	[1112608068731928...	C2	exposure	4
2019-04-04 14:25:50		3	1112608068731928576	C2	click	4
2019-04-04 14:26:04	13249	3	1112608068731928576	C2	read	4
2019-04-04 14:26:06		3	1112608068731928576	C2	click	4
2019-04-04 14:26:09	2282	3	1112608068731928576	C2	read	4
2019-04-04 14:25:50		3	1112608068731928576	C2	click	4
2019-04-04 14:26:04	13249	3	1112608068731928576	C2	read	4

only showing top 20 rows

- 4、存储到user_article_basic表中
  - 昨天：A 用户: 点击了1文章， clik
  - 今天：A用户：1文章，收藏, collect

- 合并历史数据，按照行
- **mysql: 自动更新，HIVE 支持HIVE终端ACID， update, delete**
- **pypsark: 不支持spark sql 操作HIVE原子性操作**
- 更新过程通过离线计算

```
insert overwrite table     group by user_id, article_id
```

3.2.2 用户标签权重计算

- 用户喜好标签计算
- 黑马的用户画像存储方案：存在Hbase数据库，建立HIVE到Hbase的外部表
  - 用户画像，实时推荐；
  - 数据存在Hbase， HIVE 当中不存在数据
- 如果存到HIVE，建立HBASE关联过去，删除Hive表对HBase没有影响，但是先删除HBase表Hive就会报TableNotFoundException
- HBase中的有同样的主键的行会被更新成最新插入的。可以依靠hbase来 新增/修改单条记录， 然后利用hive这个外表来实现hbase数据统计

```
create 'user_profile', 'basic','partial','env'
```

```
put 'user_profile', 'user:2', 'partial:{channel_id}:{topic}': weights
```

- 数据分析：业务熟悉，pyspark sql分析

注：spark sql 要操作external table user_profile_hbase，

- 拷贝/root/bigdata/hbase/lib/下面hbase-*.jar 到 /root/bigdata/spark/jars/目录下
- 拷贝/root/bigdata/hive/lib/h*.jar 到 /root/bigdata/spark/jars/目录下

3.2.2.3 用户画像频道关键词获取与权重计算

- 目标：**获取用户1~25频道(不包括推荐频道)的关键词，并计算权重**
  - 用户：1~25频道
- 1、读取user_article_basic表，**合并行为表与文章画像中的主题词**
  - 用户日志：推荐，HTML ，Python。。。。。人工智能频道
    - 1， 2， 25
- 2、进行用户权重计算公式、同时落地存储
  - **用户标签权重 =(行为类型权重之和) × 时间衰减**
  - 用户的喜好不是一成不变，**时间衰减**:1/(log(t)+1) ,t为时间发生时间距离当前时间的大小。
  - 点击行为越远，时间衰减系数越小(靠近0)， 用户标签权重计算的时候结果就回偏小
  - 点击行为越近，时间衰减系数越大(靠近0)， 用户标签权重计算的时候结果就回偏大

3.2.3 基础信息画像更新

- 黑马：文章偏好关键词，其它基础信息不多，直接同步基础资料

3.2.4 用户画像增量更新定时开启

3.3 离线召回与排序介绍

3.3.1 大数据的离线与在线架构

3.3.2 召回与排序介绍

召回：从海量文章数据中得到若干候选文章召回集合(数量较多)

- 离线部分去召回：A 用户：[1,.....100] Hbase存储

排序：从召回集合中读取推荐文章，构建样本特征进行排序过滤筛选

- 对召回结果排序之后，推荐给用户

3.3.2.1 推荐算法复习

3.3.2.2 召回模块

- 召回模块组成：多路召回池，召回缓存

3.3.2.3 排序模块

排序模块：

- 主要就是模型的训练以及部署
    - 排序spark 解决模型训练
    - TensorFlow以及相关方案
- 模型在线召回结果排序
    - 调用Serving服务进行实时排序

3.3.3 黑马头条推荐的召回排序设计

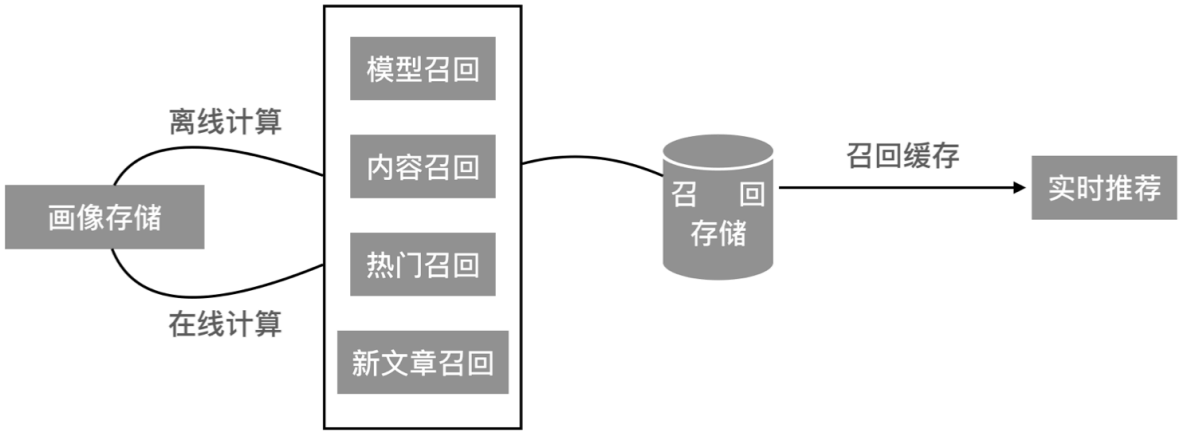
用户冷启动（前期点击行为较少情况）

- 非个性化推荐
    - **热门召回**：自定义热门规则，根据当前时间段热点定期更新维护人点文章库
    - **新文章召回**：为了提高新文章的曝光率，建立新文章库，进行推荐
    - 个性化推荐：
        - **基于内容的协同过滤在线召回**：基于用户实时兴趣画像相似的召回结果用于首页的个性化推荐

后期

- 建立用户长期兴趣画像（详细）：包括用户各个维度的兴趣特征
- 离线部分的召回：
    - **基于模型协同过滤推荐离线召回**：ALS

- **基于内容的离线召回**：或者称基于用户画像的召回
- 训练排序模型
  - **LR模型、FTRL、Wide&Deep**



3.4 召回表设计与模型召回

方案：基于模型与基于内容的召回结果存入同一张表，避免多张表进行读取处理

- `cb_recall`:als, content,online
- TTL=>7776000：过期时间
- VERSIONS=>999999：数据版本
  - A号用户： 3小时，第一次 [23, 45,16] 第二次：[130, 1240, 298, 10]
  - 不能一直存着召回的结果，设置一个过期时间，7776000 黑马三个月
  - 昨天用了APP，18 ,A [23, 45,16], [130, 1240, 298, 10],版本信息
  - [23, 45,16]，[130, 1240, 298, 10]， []，.....
  - VERSIONS=>999999

召回结果表：cb_recall

```
put 'cb_recall', 'recall:user:5', 'als:1',[45,3,5,10]
```

历史召回结果表：history_recall

```
put 'history_recall', 'reco:his:5', 'channel:18',[1,2,3]
put 'history_recall', 'reco:his:5', 'channel:18',[4,5,6]
```

3.4.2 基于模型召回集合计算

3.4.2.1 ALS模型召回实现

用户，物品，评分

- X * Y = [u, k]x [k, l] = [u, l]= y # y 表示真实的矩阵
- 建立两个随机的参数矩阵 [u, k]x [k, l] ~ y' [u, i] 预测矩阵
  - y'与y去计算损失，平均每个评分Loss
  - 梯度下降优化，两个随机的参数矩阵 [u, k]x [k, l] 里面的值
    - 优化好 [u, k]x [k, l] 矩阵分解算法的参数, y模型预测结果
- 步骤：
  - 1、从哪里得到评分矩阵(利用用户，文章点击日志构造)
    - clicked, int
    - from pyspark.ml.feature import StringIndexer from pyspark.ml import Pipeline

      ALS 模型输入要求：不能使用业务的用户ID和文章ID

      ALS需要输入：用户总数N，1~N， 文章总数：M， 1~M
  - 2、ALS模型训练以及推荐
    - recommendations：[[als_article_id, score],....]
  - 3、推荐结果解析处理，存储
    -
    - article_id| user_id|channel_id|
    - 按照用户和频道进行分组，得到每个推荐列表
    - 3.4.2.4 召回结果存储

```
+-------------------+----------+--------------------+
|            user_id|channel_id|        article_list|
+-------------------+----------+--------------------+
|1114864874141253632|        18|[17665, 16005, 44...|
|                 33|        13|            [141431]|
|1114863735962337280|        13|            [141431]|
|1114864237131333632|        13|            [141431]|
```