* **进行用户日志数据处理**

sqlDF.rdd.flatMap(\_compute)

* **合并历史数据，存储到user*article*basic表中**

old.unionAll(data)

* 按照user*id和article*id分组，合并其他信息
  + HIVE目前支持hive终端操作ACID不支持python的pyspark原子性操作，并且开启配置中开启原子性相关配置也不行。---因此我们采用overwrite
  + insert overwrite table user*article*basic

### 3.2.2 用户标签权重计算

存储：

**选择存储在Hbase当中。如果离线分析也想要使用我们可以建立HIVE到Hbase的外部表。---这种设计最佳**

建立HIVE表：

1、存储位置

2、与Hbase表的映射关系

WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,basic:,partial:,env:")

是为了映射时去除前缀

3、映射的表名称

#### 3.2.2.2 Spark SQL关联表读取问题？

依赖包：

* 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**

* 1、读取user*article*basic表，**合并行为表**与**文章画像中的主题词**
  + user*single*topic = user*topics*list.withColumn('topic', F.explode('topics')).drop('topics')
* 2、进行用户权重计算公式、同时落地存储
  + user*single*topic.foreachPartition(*save*weights)
  + **用户标签权重 =( 行为类型权重之和) × 时间衰减**
  + **用户标签权重** :对于某个关键词操作的各种行为分数之和
    - 时间间隔系数，行为权重和
  + 时间衰减系数：根据用户行为的时间距离而定，距离越长，系数越小，此期间行为权重会小一些
    - 距离短，系数越大，此期间操作的行为影响越大

### 3.2.3 基础信息画像更新

Hive toutiao数据库中放着文章，用户的基础信息

* 用户：性别，年龄等信息，更新到画像中

get 'user\_profile', 'user:2'

执行这个操作时，切记后面不要加分号

# 3.3 离线召回与排序介绍

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)

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

#### 3.3.1.1 召回与排序业务流程

### 3.3.2 头条推荐的召回排序设计

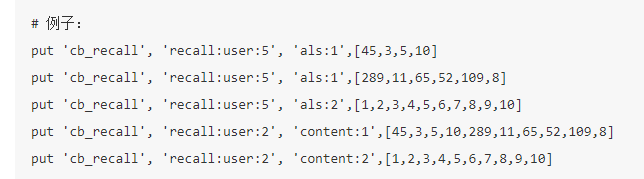
* 在线部分：
  + 非个性化推荐
    - **热门召回**：自定义热门规则，根据当前时间段热点定期更新维护人点文章库
    - **新文章召回**：为了提高新文章的曝光率，建立新文章库，进行推荐
  + 个性化推荐：
    - **基于内容的协同过滤在线召回**：基于用户实时兴趣画像相似的召回结果用于首页的个性化推荐
* 离线部分：
  + 建立用户长期兴趣画像（详细）：包括用户各个维度的兴趣特征
  + 训练排序模型
    - **LR模型、FTRL、Wide&Deep**
  + 离线部分的召回：
    - **基于模型协同过滤推荐离线召回**：ALS
    - **基于内容的离线召回**：或者称基于用户画像的召回

# 3.4 召回表设计与模型召回

### 3.4.1 召回表设计

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

* 将多个召回方式的推荐结果放入一张表中，并且对于每个用户的每个频道会建立多个版本需要，设置一个时间过期
* 召回结果有存储了



上面Als是模型召回，content是内容召回

在HIVE用户数据数据库下建立HIVE外部表,若hbase表有修改，则进行HIVE 表删除更新---这样做的目的是为了做离线的一个数据分析

#### 增加一个历史召回结果表

* + 1、直接在存储召回结果部分进行过滤，比之后排序过滤，节省排序时间
  + **避免重复推荐看过的文章**

### 3.4.2 基于模型召回集合计算

#### 3.4.2.1 用户日志信息处理

步骤：

* 数据类型转换,clicked
* 用户ID与文章ID处理

#### 3.4.2.2 ALS 模型训练与推荐---矩阵分解的推荐

* 目标：处理成ALS模型所需数据类型和格式---重新建立索引即可

Als的坏处，没有行为日志的用户，不会进行推荐

获取每个文章对应的频道，推荐给用户时按照频道存储

* **按照用户和频道存储，所以需要得到每个文章所属频道**

#### 3.4.2.3 推荐结果处理

#### 3.4.2.4 召回结果存储

* hbase

# 3.5 离线用户基于内容召回集

### 3.5.1 基于内容召回实现

* Item协同:看过的文章相似的文章(指的是通过文章一些属性，得到相类似的文章)
* 内容召回：点击过或者看过的文章，**文章内容**进行向量化的相似文章，推荐相似的其他文章
  + article\_similar
* 目的：实现定时离线更新用户的内容召回集合
  + 1、过滤用户点击的文章
    - clicked=True
    - filter('clicked=True')
  + 2、用户每次操作文章进行相似获取并进行推荐
    - 注意相似文章Hbase表的读取以及筛选过滤操作
    - article*dict = article*similar.row(str(116724).encode(), columns=[b'similar'])

# 3.6 离线用户召回定时更新