# 熟悉表

user\_profile\_hbase 存储在hbase，映射到hive内的，因此要启动hbase后，才能看

Hive数据库

## profile数据库

user\_action 用户行为表

user\_article\_basic 对用户行为表的内容进行解析

user\_profile\_hbase 通过关键词对用户进行了一个画像

## article 数据库

article\_data 文章的原始数据信息，sentence中讲channel\_name,title，content进行了合并

article\_profile 文章的画像

article\_vector 文章用一组向量值来代表，通过word2vec计算出来

idf\_keywords\_values 词的idf值

tfidf\_keywords\_values 文章中某个词的tfidf值，存储了文章id，keyword,tfidf值

textrank\_keywords\_values 文章中某个词的textrank值，存储了文章id，keyword,textrank值

## toutiao数据库

news\_article\_basic 新文章的基本信息

news\_article\_content 新文章的id和内容

news\_channel 频道信息

user\_basic 用户阅读文章数目记录信息

user\_profile 用户的描述信息

# 通过flume去读取日志更新

日志格式由算法人员来定

# 曝光的参数，

{"actionTime":"2019-04-10 18:15:35","readTime":"","channelId":0,"param":{"action": "exposure", "userId": "2", "articleId": "[18577, 14299]", "algorithmCombine": "C2"}}

## 到flume的conf目录去增加配置

/root/bigdata/flume/conf/collect\_click.conf

## 开启flume

/root/bigdata/flume/bin/flume-ng agent -c /root/bigdata/flume/conf -f /root/bigdata/flume/conf/collect\_click.conf -Dflume.root.logger=INFO,console -name a1

## 放日志就是模拟用户的操作

echo {\"actionTime\":\"2021-04-11 21:04:39\",\"readTime\":\"\",\"channelId\":18,\"param\":{\"action\": \"click\", \"userId\": \"2\", \"articleId\": \"14299\", \"algorithmCombine\": \"C2\"}} >> userClick.log

## 分区要进行关联，hive里边才有数据

alter table user\_action add partition (dt='2021-08-13') location "/user/hive/warehouse/profile.db/user\_action/2021-08-13/"

# 2.4 离线文章画像计算

启动jupyter的方法

cd /root/toutiao\_project/reco\_sys/offline/full\_cal

执行sj

news\_article\_content 里边有13万篇文章需要计算

article\_data也有13万篇文章

## Textrank

# 2.5 离线增量文章画像计算

### 2.5.1 离线文章画像更新需求

第一次：所有更新，后面增量每天的数据更新26日：1：00~2：00，2：00~3：00，左闭右开,一个小时更新一次

### 2.5.2 定时更新文章设置

* 目的：通过Supervisor管理Apscheduler定时运行更新程序
* 1、更新程序代码整理，并测试运行
* 2、Apscheduler设置定时运行时间，并启动日志添加
* 3、Supervisor进程管理

### 2.6.1 Apscheduler使用

**APScheduler**：强大的**任务调度工具**，可以完成**定时任务**，**周期任务**等

* 配置好定时运行的函数（今天我们只是文章画像的定时更新，可以看到我们的update.sh和update\_article很像）

定义更新逻辑

* 编写APscheduler配置
* 增加打印日志添加(程序问题，离线更新文章画像流程进度，在settings文件夹中)

# 2.7 Word2Vec与文章相似度

### 2.7.1 文章相似度

* 需求
  + 首页频道推荐：每个频道推荐的时候，会通过计算两两文章相似度，快速达到在线推荐的效果，比如用户点击文章，我们可以将离线计算好相似度的文章排序快速推荐给该用户。此方式也就可以解决冷启动的问题
* 方式：
  + 1、计算两两文章TFIDF之间的相似度
  + 2、计算两两文章的word2vec或者doc2vec向量相似度

### 2.7.2 Word2Vec模型介绍

#### 2.7.2.2 词向量是什么

* **词的独热表示**：One-hot Representation
  + 维度过大词汇鸿沟现象：任意两个词之间都是孤立的。
* **词的分布式表示**：Distributed representation
  + **最大的贡献就是让相关或者相似的词，在距离上更接近了**

#### 2.7.2.3 词向量原理

* 统计语言模型：把语言（词的序列）看作一个随机事件，并赋予相应的概率来描述其属于某种语言集合的可能性

N-Gram

一元模型(unigram model)：假设某个出现的概率与前面所有词无关

二元模型(bigram model)：假设某个出现的概率与前面一个词相关

* P(s) = P(w1)P(w2|w1)P(w3|w2)…P(w*i|w*i-1)

三元模型(trigram model)：假设某个出现的概率与前面两个词相关

* P(s) = P(w1)P(w2|w1)P(w3|w1,w2)…P(w*i|w*i-2,w\_i-1)

#### 2.7.2.4 词向量计算得出

* 通过一个三层神经网络得出，由约书亚.本吉奥（Yoshua Bengio）提出word2vec模型

### 2.7.3 文章词向量训练

* 目的：通过大量历史文章数据，训练词的词向量
* 由于文章数据过多，在开始设计的时候我们会**分频道进行词向量训练，每个频道一个词向量模型**
* 25个词向量模型

步骤：

* 1、根据频道内容，读取不同频道号，获取相应频道数据并进行分词
* 2、Spark Word2Vec训练保存模型

### 2.7.4 增量更新-文章向量计算

有了词向量之后，我们就可以得到一篇文章的向量了，**为了后面快速使用文章的向量，我们会将每个频道所有的文章向量保存起来。**

* 目的：保存所有历史训练的文章向量
  + 1、加载某个频道模型，得到每个词的向量
    - 18号频道所有文章训练模型：3000个词
  + 2、获取频道的文章画像，得到文章画像的关键词(接着之前增量更新的文章article\_profile)
  + 3、计算得到文章**每个词的向量**, **计算得到文章的平均词向量即文章的向量**

### 2.7.5 文章相似度计算

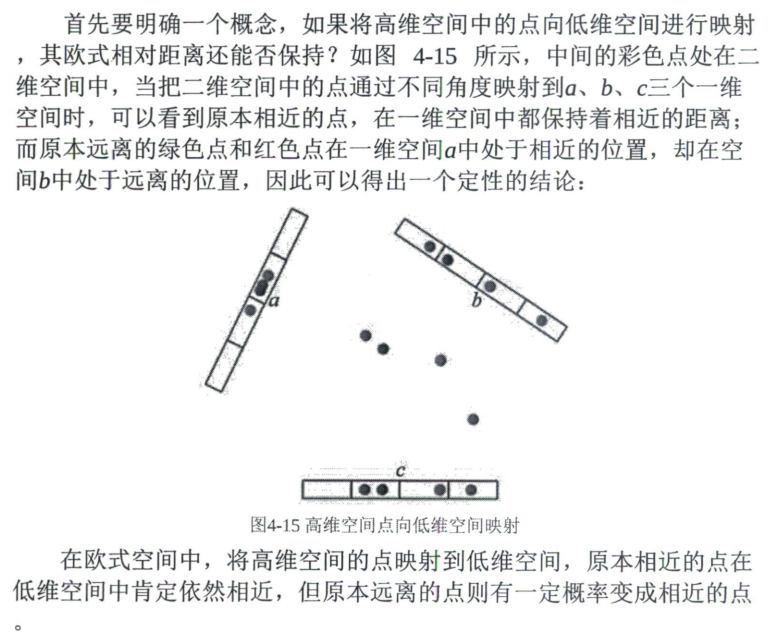
* 目的：计算每个频道两两文章的相似度，并保存
* 分析问题：
  + 1、是否需要某频道计算所有文章两两相似度？
  + 1，2，3，4，5
  + 4+3+2+1 = 10

#### 每个频道的文章先进行聚类

* + - * 1+3 = 4

#### 局部敏感哈希LSH(Locality Sensitive Hashing)

* **LSH算法基于一个假设，如果两个文本在原有的数据空间是相似的，那么分别经过哈希函数转换以后的**它们也具有很高的相似度
  + 离得越近的对象，发生冲突的概率越高
  + 离得越远的对象，发生冲突的概率越低
  1. 如果d(O1,O2)<r1，那么Pr[h(O1)=h(O2)] ≥ p1
  2. 如果d(O1,O2)>r2，那么Pr[h(O1)=h(O2)] ≤ p2

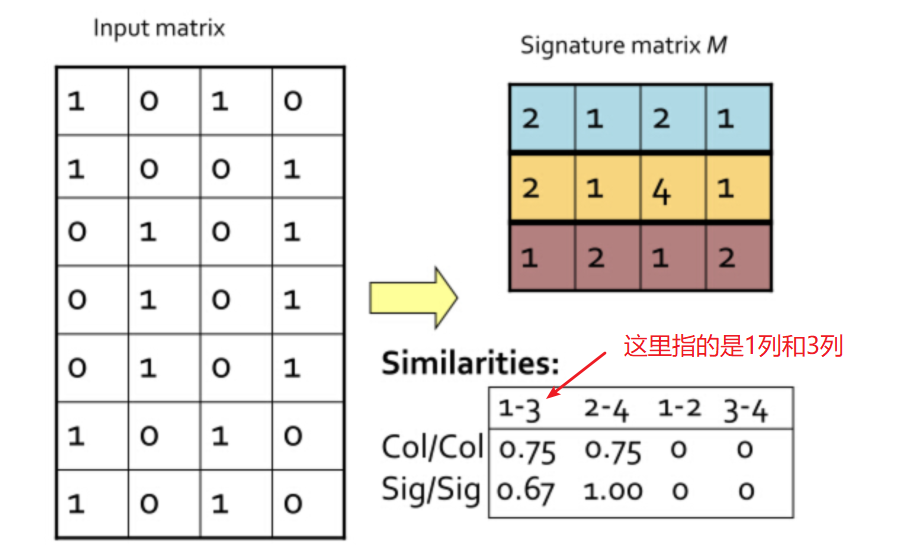


#### mini hashing

1、Minhash的定义为:\*\* 特征矩阵按行进行一个随机的排列后,第一个列值为1的行的行号。

* 一次：1，1，2，1
* 二次：2，3，2，1

上述重复操作的过程的结果：**签名向量 M**，上述过程重复N次随机排列得到的



会发现原矩阵input matrix 文档1和文档3的相似度是0.75，得到签名矩阵的相似度是0.67，非常相近

2、对Signature每行分割成若干brand（**一个brand若干行**）,每个band计算hash值，我们需要将这些hash值做处理，使之成为事先设定好的hash桶的tag，然后把这些band“扔”进hash桶中。

比如：M [L，4] ， L分成B个brand，每个brand 若干行

L / B = r, 5个brand都会哈希到捅当中，r是每一行有多少个签名向量，b是有多少个r

3、最终分配到同一个bucket的概率：1−(1−s^r)^b，s是相似度，先求r次幂，再求b次幂，不用掌握这个公式如何推导的

r=5, b=20时候，效果

* 当s=0.8时，两个文档被映射到同一个哈希桶的概率是
  + Pr(LSH(O1)=LSH(O2))=1−(1−0.8^5)^20=0.9996439421094793
* 当s=0.2时，两个文档被映射到同一个哈希桶的概率是：
  + Pr(LSH(O1)=LSH(O2))=1−(1−0.2^5)^20=0.0063805813047682

总结：通过签名向量矩阵M，来达到离得越近的对象，发生冲突的概率越高，离得越远的对象，发生冲突的概率越低

<https://zhuanlan.zhihu.com/p/46164294> 有很好的r，s值变化的图形变化曲线

https://en.wikipedia.org/wiki/Locality-sensitive\_hashing#Bit\_sampling\_for\_Hamming\_distance

#### Random Projection

总结：通过降维(投影)之后的结果，进行哈希分桶，来达到离得越近的对象，发生冲突的概率越高，离得越远的对象，发生冲突的概率越低

* 2、相似度结果数值如何保存？

### 2.7.4.2 相似度计算

* 目的：计算18号Python频道的文章之间相似度
  + 1、读取数据（保存到表当中向量），进行类型处理(数组到Vector)
  + 2、BRP进行训练模型

### 2.7.4.3 问题3

对于计算出来的相似度，是要在推荐的时候使用。那么我们所知的是，HIVE只适合在离线分析时候使用，因为运行速度慢，所以只能将相似度存储到HBASE当中

* hbase

### 2.7.5 文章相似度存储

目的：将所有文章对应相似度文章及其相似度保存

* **foreachPartition不同于map和mapPartition，主要用于离线分析之后的数据落地**
* 代码：
  + table.put(str(row.datasetA.article\_id).encode(),  
     {b"similar:%d" % row.datasetB.article\_id: b"%0.4f" % row.EuclideanDistance})

文章相似度相对于LSH更好的处理

IMG_256https://www.jianshu.com/p/b72af2d27235

# 2.8 文章相似度增量更新

### 2.8.1 增量更新需求

### 2.8.2 增量更新文章向量与相似度

总结：1小时，业务数据库中取出这一个小时的新文章，1、合并文章三个标结果到sentence，2、计算TFIDF与TextRank， 3、计算文章画像 4、计算新文章的向量，计算新文章相似的文章以及相似度

去使用文章全量代码定时任务时，注意上传/root/modelsbak

# 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 用户画像计算设计

* 用户画像标签建立
  + 用户：每个频道这个用户的关键词和权重， 基本信息的结果

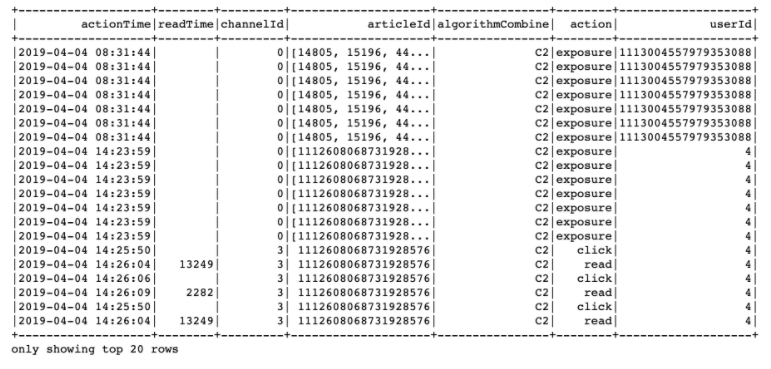
# 3.2 用户画像增量更新

### 3.2.1 增量用户行为日志处理

比如clicked/exposure就是点击率

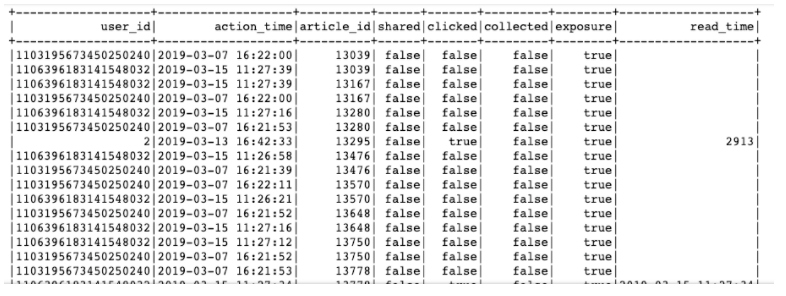
* 目的：**首先对用户基础行为日志进行处理过滤，解析参数，从user *action—>user* article\_basic表。**
* 1、创建HIVE基本数据表
* 2、读取固定时间(第一次所有历史行为数据)内的用户行为日志
  + user\_action固定日期
  + 关联表与Hadoop历史日期目录---这个已经给大家关联好了
  + 定量进行更新：
  + **读取固定时间内的用户行为日志**
    - 注意每天有数据都要关联一次日期文件与HIVE表
* 3、进行用户日志数据处理

原始数据是什么时间做了什么事情



我们要把它转为对应用户是否做过某个操作

这里注意spark sql执行hive sql需要hive-hcatalog-core-1.1.0-cdh5.13.1.jar



* 4、用户行为做了透视，存储到user *article* basic表中

缺少pyhdfs，需要pip install pyhdfs

