2.5.3 文章画像结果

- 关键词以及权重
  - 1、首先取出词的逆文档表中数据IDF
    - withColumnRenamed：修改列名称
  - 2、5篇文章textrank * IDF
    - withColumn：新建一列数据，做计算
  - 3、合并到字典结构
- 主题词：
  - 合并 所有文章， TFIDF20 inner join TextRank 20，共同出现的词

2.5 离线增量文章画像计算

- 历史文章：所有频道 13万
  - 来了新文章？？
  - 新增的文章拿过来
  - article_data:文章数据的合并
  - tfidf_keywords_values, (idf_keywords_values)：新增文章的tfidf
    - ??? 13万，20000词，idf， 5篇文章, 5万
      - a: log(130000/100)
  - textrank_keywords_values：新增文章的textrank
  - **article_profile**:新增的文章画像
- 定时运行添加步骤, apscheduler, supervisor
  - update_article_profile: 编写一个apscheduler程序，定时添加update_article_profile任务，一个小时
  - supervisor管理apscheduler程序(进程)，可以进行启动和关闭
  - 日志打印，自定义日志，offline, /root/logs/offline.log
- 离线画像业务流
- 文章标签构建的技术
- 代码实现：API使用 技术框架API， DF与RDD一些API使用

2.7 Word2Vec与文章相似度

2.7.1 文章相似度

- word2vec：是什么
  - 原理？深度学习？
- **将字词表示为唯一的离散ID还会导致数据稀疏性**

2.7.2.2 词的分布式表示（ distributed representation）

定义：将**文字**通过一串数字向量表示

- **词的独热表示**：One-hot Representation
  - 采用稀疏方式 存储，简单易实现
  - 1、向量的维度会随着句子的词的数量类型增大而增大；
  - 2、任意两个词之间都是孤立的，根本无法表示出在语义层面上词语词之间的相关信息，而这一点是致命的。
- 分布式表示：
  - 它是一种低维实数向量，这种向量
  - 长度？数字(实数)
  - **最大的贡献就是让相关或者相似的词，在距离上更接近了**：可以量化词的距离

2.7.2.3 词的分布式方式

基于神经网络的分布表示，名称可以叫做**词向量**、**词嵌入（ word embedding）**或**分布式表示（ distributed representation）**

- 实现方式：神经网络模型，如CBOW、Skip-gram
  - **word2vec词向量，指的是用word2vec训练出来的词向量**

计算文章相似度

- TFIDF与TEXTRANK :关键词提取 word2vec：向量相似度计算
- 词向量：一个词[,,,,,,]
- [a, d, c,f]：A平均词向量代表这篇文章向量
- [g, f, r, d]：B平均词向量代表这篇文章向量

实现相似度计算分解为一下几个步骤：

- 1、训练文章词向量模型
- 2、利用模型得到文章的平均词向量，存储到article_vector中
- 3、使用相似度计算工具进行文章之间的相似度计算

2.7.3 文章词向量训练

- 文章数据过多：百万，亿文章量，训练成本非常大
- 按照频道去进行训练：1~25频道，Python 2万文章，训练一个模型
  - 得到25个模型
- 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
  - 3、Spark Word2Vec训练保存模型
    - Word2Vec(vectorSize=100, inputCol='words', outputCol='model')
    - vectorSize=100

```
channelInfo = {
```

```
            1: "html",
            2: "开发者资讯",
            3: "ios",
            4: "c++",
            5: "android",
            6: "css",
            7: "数据库",
            8: "区块链",
            9: "go",
            10: "产品",
            11: "后端",
            12: "linux",
            13: "人工智能",
            14: "php",
            15: "javascript",
            16: "架构",
            17: "前端",
            18: "python",
            19: "java",
            20: "算法",
            21: "面试",
            22: "科技动态",
            23: "js",
            24: "设计",
            25: "数码产品",
        }
```

- 25个频道都需要训练

2.7.4 增量更新-文章向量计算

- 文章—>向量
- 目的：保存所有历史训练的文章的向量结果，article_vector
  - 分频道进行计算
  - 18号
- 18号模型，得到Python所有文章不同的词的词向量(词，词向量)
  - Word2VecModel

```
+------------------+--------------------+
|              word|              vector|
+------------------+--------------------+
|                广义|[0.28907623887062...|
|                钟爱|[-0.0529650673270...|
```

- 18号频道所有文章的关键词得到，A文章[a, d, c,f]，求一个平均值
  - 取出每篇文章的画像关键词(20个关键词)
  - LATERAL VIEW explode(keywords) AS keyword, weight

- 加入词的权重 * 词的向量 = weights x vector=new_vector
    - 可以选择不进行相乘计算
    - 合并vectors与文章画像表

```
+----------+----------+-------+------------------+------+--------------------
+
|article_id|channel_id|keyword|            weight|  word|
vector|
+----------+----------+-------+------------------+------+--------------------
+
|     12936|        18| strong|  4.705249850191452|strong|
[0.06607607007026...|
|     12936|        18| python| 1.5221131438694515|python|
[-0.0696719884872...|
|     12936|        18|   bool|  1.698618363235006|  bool|
[0.08196849375963...|
```

```
+----------+----------+--------------------+
|article_id|channel_id|      articlevecoter|
+----------+----------+--------------------+
|     12936|        18|[[0.2497335821390...|
|     13206|        18|[[0.1658860594034...|
|     14029|        18|[[-0.058167435228...|
|     14259|        18|[[-0.054562915116...|
|     14805|        18|[[-0.058167435228...|
+----------+----------+--------------------+
```

```
+----------+----------+--------------------+
|article_id|channel_id|       articlevector|
+----------+----------+--------------------+
|     13098|        18|[0.10032696360722...|
|     13248|        18|[0.30715655184843...|
|     13401|        18|[0.13749069944024...|
|     13723|        18|[0.12696191947907...|
|     14719|        18|[-0.0156031135935...|
+----------+----------+--------------------+
```

- 最终文章向量结果保存：
    - **对计算出的"articleVector"列进行处理，该列为Vector类型，不能直接存入HIVE，HIVE不支持该数据类型**
    - `article_vector`
    - [float(i) for i in row.articlevector.toArray()]

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

    - ?? 13万， Python ： 1万， C++：1万
    - 减少计算量
    - 业务按照频道区分
- 1、是否需要某频道计算所有文章两两相似度?
    - Python ： 100万
- 2、相似度结果数值如何保存?
    - 放在Hbase 不放在HIVE

但是事实当文章量达到千万级别或者上亿级别，特征也会上亿级别，计算量就会很大

1 每个频道的文章先进行聚类

- Python：100万， 100个族群， 每个族群10000篇
    - 100个族群：每个族群内做两两相似度计算
    - 但是对于每个频道聚成多少类别这个M是超参数，并且聚类算法的时间复杂度并不小，当然可以使用一些优化的聚类算法二分、层次聚类。

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

- 背景：**或近似最近邻查找（Approximate Nearest Neighbor, ANN）**
    - **K-d tree**
    - **LSH**

什么是局部敏感哈希(Locality Sensitive Hashing，LSH)

总结：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

- 定义：对于两个物体**u**,**v**（可以理解为两个文件、两个向量等），LSH生成的value值的每一bit位相等的概率等于这两个物体的相似度
- 如果d(u,v)相似度小，那么冲突概率Pr[h(u)=h(v)] <= p1
- 如果d(u,v)相似度大，那么冲突概率Pr[h(u)=h(v)] ≥ p2

- 1、MinHash

假设我们有如下四个文档D1,D2,D3,D4D1,D2,D3,D4的集合情况,每个文档有相应的词项,用{w1,w2,..w7}{w1,w2,..w7}表示。若某个文档存在这个词项,则标为1，否则标为0。

- 第一步：
    - （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。
    - 循环上述（1）这步骤，
        - 矩阵按行进行多次置换,每次置换之后统计每一列(对应的是每个文档)第一个不为0位置的行号,这样每次统计的结果能构成一个与文档数等大的向量,我们称之为**签名向量**
    - signature matrix
        - S次打乱，得到S行结果，结果当中都是记录的行号
    - (2)S行 = r * b
        - s =100
        - 100 * 4 = 25行

  比如，令b=20,r=5，s∈[0,1]是这两个文档的相似度，等于给的文当前提条件下：

- 假设原始两个文档相似，如当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−(1−0.85)5=0.9996439421094793
- 假设原始两个文档不相似，如当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−(1−0.25)5=0.0063805813047682

结论：

1、A，B,C,D 四篇文章，mini hash之后，A,B冲突概率高：0.98

求取相似度：AB文档这两个文档相似度 S

2、A，B,C,D 四篇文章， B,C,D之间都不冲突，分到不同的桶，BCD本身就不相似，不需要计算他们之间的相似度

- 2、基于随机投影的方式
  - 基于Stable Distribution的投影

2、基于随机投影的方式(了解)

给定特征向量**v**v，**Hash的每一bit的生成公式为**：

$$h(\boldsymbol{v}) = \left\lfloor \frac{\boldsymbol{x} \cdot \boldsymbol{v} + b}{w} \right\rfloor$$

2.7.4.2 相似度计算

- 1、读取数据，进行类型处理(数组到Vector)
- 2、BRP进行FIT得到距离

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
- 调用foreachPartition
  - **foreachPartition不同于map和mapPartition，主要用于离线分析之后的数据落地，如果想要返回新的一个数据DF，就使用map后者。**
  - 最后分析计算的结果落地存储使用foreachPartition函数