**用户长期兴趣算法流程笔记**

## 一.引言：主函数的作用

主函数run\_update\_all.sh，猜测是用来更新全部用户的长期兴趣。其中连续运行三个脚本run\_by\_day.sh run\_merge.sh run\_undate\_KV.sh。其中，第一个脚本是通过每天收集到的数据来得到用户的兴趣，第二个脚本是将今日得到的兴趣与之前的兴趣相结合，第三个脚本是用来将结果以字典的形式存储在数据库当中。

## 二.run\_by\_day.sh脚本详细作用

接下来运行的脚本run\_by\_day.sh，这个脚本的输入是信息流日志中那种已经格式化了的信息。里面主要调用了三个脚本，分别是run\_get\_article\_hot\_info.sh、run\_get\_doc\_info.sh、run\_get\_feature\_score.sh，其中，最后一个脚本调用了两次，一次用来计算正兴趣，另一次是负兴趣，据说负兴趣现在已经不再更新，所以咱们把主要精力放在前三个脚本运行中。首先是run\_get\_article\_hot\_info.sh脚本，目测是用来获得文章热度信息的，咱们去看看具体情况。

### 2.1 run\_get\_article\_hot\_info.sh的作用

看了run\_get\_article\_hot\_info.sh中的内容，里面有一组m-r，用处是提取文章的热度，方法是通过信息流日志中标准化的第35个位置doc\_ID来统计。

下一节将介绍下一个被调用的脚本

### 2.2 run\_get\_doc\_info.sh的作用

接下来要运行run\_get\_doc\_info.sh 了，咱们看看这回这个脚本是做什么用的。这个脚本中有一个m-r过程，其作用是针对每一条行为（信息流日志中格式化过的字符串），返回一个字符串，里面有如下的字段：

(id 行为(读 收藏 退出等) 特征(很多，代码中有注释) 阅读时间(除了退出行为外都是0) 文章热度)

具体的m-r.py文件中已经下载到本地并详细注释了功能

下一节将介绍第三个被调用的脚本。

### 2.3 run\_get\_feature\_score.sh的作用

该脚本的作用是根据2.2中的输出，计算每条记录对每个用户的多个关键词产生的分数影响的，具体运作方式如下：代码首先设定路径，并根据需要的时间设置日期时间等（这些技巧现在还不能秒懂，以后要学习），然后运行了get\_feature\_score\_mapper.py和get\_feature\_score\_reducer.py的m-r，其数据输入（处理的文件）是2.2步中的输出

#### 2.3.1 get\_feature\_score\_mapper.py的作用

本py开头定义了通过阅读量和阅读时间计算对应权重的函数，在主函数处理中，根据用户的四类行为：阅读（read, r）、喜欢（favor, f）、分享（share, s）、退出（quit, q）分别采用不同的处理方法，具体来说，首先这四种行为对应的权重r—0.2，f—3，s—5，q—1，如果用户是喜欢或者分享，直接赋予权重3和5并在输出流中输出，如果是阅读，则要乘上文章的阅读量对应的权重，如果是退出，还要乘上文章阅读时长对应的权重。

最后，本mapper输出(mid feat(某kw或者tag等) score time)的模式

#### 2.3.2 get\_feature \_score\_reducer.py的作用

本py的作用就是将2.3.1中的输出收集整理成最终形式并输出，具体而言，首先对输入存储成了一个双重字典，字典是mid:word:score的形式，然后再将字典输出成字符串。

改进思考：感觉不需要用这种方法输出，如果输入是有序的，按照mid为主关键词，word为副关键词排序的话，可以直接计算得到每一个mid+word的分数，然后流式输出即可。

本reducer输出字符串，形式为：

(mid+’\t’+time+’\t’+word1\_score1+’\t’+word2\_score2+’\t’+…)

### 2.4 本章小结

至此，run\_by\_day.sh的运行就结束了，其功能就是通过上面的步骤提取出用户一天的兴趣分数。

## 三.run\_merge.sh脚本的详细作用

本脚本的作用是将今日刚算出的兴趣分数和过去的兴趣分数相结合，其中共运行了两个脚本，首先运行了run\_get\_feature\_score\_merge.sh，然后是run\_normalization\_score.sh，下面分别对这两个脚本进行解说

### 3.1 run\_get\_feature\_score\_merge.sh的具体功能

运行了一组m-r，其中，mapper阶段仅仅是一个管道，reducer阶段的脚本名称是get\_feature\_score\_merge\_reducer.py，其输入形式是2.3.2小节中最后描述的输出形式，其中，不仅仅是第二章中run\_by\_day所处理出来的一天的输出，还包括之前n天处理得到的输出。经过了加权平均之后，输出形式如下："mid+'\t'+TAG\_关晓彤:1.1234##TAG\_鹿晗:0.9876##TAG\_跑男+'\t'+KW\_word1:1.1211##KW\_word2:0.9865"。

Ps.为了得到这个输出，py文件中对列表的各种操作看了1h才明白。。。其中的关键节点相关注释已经写进了本地的该文件当中。

改进思考：py文件中使用了type作为变量，而type是py的固有关键字

### 3.2 run\_normalization\_score.sh的具体功能

运行了一组m-r，其中，reducer阶段直接输出，mapper阶段的脚本名称是normalization\_mapper.py。该程序对每一个用户的kw、tag、topic分别进行归一化，也即是说，每一个用户的最高分都是1。归一化的方法很简单，就是最普通的线性映射。

## 四. run\_undate\_KV.sh脚本的详细作用

本脚本首先运行了get\_updated\_mid.py这个程序，然后是run\_get\_data.sh脚本，最后是update\_into\_KV.py程序。接下来咱们一个一个看。

### 4.1 get\_updated\_mid.py的作用

本程序用来提取出所有今天更新的用户id，事实上，只有今天产生行为的用户才需要更新，所以本程序的方法就是在本日的计算日志中（2.3.2节的输出）提取全部的用户id放在一个集合中，并在最后存储在文件当中。

### 4.2 run\_get\_data.sh的作用

本脚本中执行了一个m-r过程，实际上只是一个管道，是将正兴趣和负兴趣合并的一个操作。脚本中的for循环将待处理文件一个一个接在inputs变量后面，然后echo将其字符串化（将转义字符/变量等转化为本意），之后放到m-r过程当中。

### 4.3 update\_into\_KV.py的作用

本程序的作用是将之前计算出来的日志写入数据库，具体的操作并不是太明白，用到了一些专门的模块，暂时不研究了。以后有需要再研究效率会高一些。

注1：所采用的兴趣提取方法，大致流程如下：对于每一个用户user阅读了一篇文章doc的行为，我们认为这篇doc里出现的全部关键词(kw)/话题(topic)/推送文章的账号(account)和标签(tag)都增加了一个相同的分数，这个分数是score=w1\*w2\*w3，其中，w1和文章本身的热度相关，越热的文章其权重越低；w2是用户阅读的时长；w3是用户的具体行为，对于收藏、分享、浏览这几种不同的行为，分别给予一个不同的权重。

注2：上述步骤文件所在路径./test/long\_term\_interest/positive\_interest/

Tips1：在代码中处理流的时候，要注意各种严谨的小处理，即使是已经格式化的数据，也需要对没有成功格式化（出现奇怪的空字段）进行判断，当然最好可以try-except这种py风格的写法。作为面向对象的语言，或者说要上线的代码，一旦出现字段不是预期中的让程序崩溃就不好弄了

## 五.想讨论的一些问题

### **5.1关于权重的一些思考**

1.阅读时长的权重该如何计算。现有的计算方式是按照绝对时长，但是文章的长度并不相同，这么计算是否足够科学

2.阅读的权重是否过高，对于用户，如果他有阅读的行为，（猜测，可以进行统计）几乎一定对应一条退出行为，根据本程序内的函数逻辑，咱们可以认为阅读产生了大约0.2\*0.5=0.1的分数，退出产生了1\*0.5\*0.8=0.4的分数，加起来就是0.5

3.FAVOR的权重是否应该比SHARE低，就我个人的经验而言，SHARE某东西有不低的几率是社交配合，FAVOR一样东西几乎一定是真喜欢。

4.不知道有没有相关文献。

### 5.2 程序架构中的一些问题

1.处理中间步骤的时候，基本都是在hadoop上m-r生成.lzo压缩文件，然后下载到linux本地解压再处理，甚至还要再push上去。感觉略有一点违背m-r的初衷，如果将来的用户点击量暴涨，感觉计算的压力会比较大。压缩是必要的么？

2.还是压缩的问题，每日计算得到的原始分数在之后的分数合成中是有用的，但是如果将其压缩，每次计算分数合成都需要对上百天的分数文件进行解压，并且，似乎是需要拿到本地进行解压？我不知道这样的计算量相比较于不压缩多占空间这个选择而言，是不是利好，感觉是亏了。

3.

### 5.3 py脚本中的更改

1.删去了get\_feature\_score\_merge\_reducer.py中我认为一些逻辑冗余的部分，比如对score\_time\_list列表的排序。

2.增加了一些脚本中对数据格式的判断，使用try\_except将字段不符合所要求格式的数据略过，之前的代码会直接引发异常中断。

3.将get\_article\_hot\_info\_reducer.py中的字典输出法更改成了流式输出法

4.将get\_feature\_score\_merge\_reducer.py中与python默认保留字段冲突的变量名type更换为label。

5.极其少量的变量名字更改。

6.文件名更改，文件名中全部加入了”yj”作为责任人标签，这些脚本的责任在我