3. item-based CF的Spark实现

Lance LanceDaily 2016-08-01

本文基于前面两篇的推荐算法调研实现。由于线上为Spark环境，故将调研中使用的算法从Python环境中迁移到了Spark。

系统分4个模块：读取数据、训练、获取结果、写回。

一、读取数据：

训练集选取前60天（不含当天）的购买数据。我们要的数据有三个：user_id、UPC码、购买数量。

UPC码即物品的条形码，对于不同商家的相同物品（例如不同门店的600ml可口可乐），其item_id可能不同，但条形码相同。使用UPC码可实现跨门店的推荐，也防止了“某门店相似的物品为其他门店相同物品”的尴尬场景。

购买数量用于衡量用户对于某物品的感兴趣程度。本次统计的是在时间段内某用户对某物品的购买总量。如用户A在前60天内共购买了3次600ml可口可乐，第一次购买1瓶，第二次购买2瓶、第三次购买1瓶。则矩阵中用户和物品对应的值为4。

数据源为Hive，涉及的上游表有：

mart_waimai.fact_ord_arranged;

mart_waimai.fact_wm_order_detail;

origin_waimai.waimai_poi__wm_food.

其中mart_waimai.fact_ord_arranged是用户(user_id)和订单(ord_id)之间的关联表，mart_waimai.fact_wm_order_detail是订单(ord_id)和物品(item_id)之间的关联表，origin_waimai.waimai_poi__wm_food为物品(item_id)和UPC码(upc_code)之间的关联表。将以上三个表关联并加上日期字段可得到日期、用户ID、物品ID、UPC码、购买数量的元组。实际使用中为方便查看数据加上了物品名称这一字段。

由于mllib中限制，item_id字段必须为整形，而UPC码有13位，故训练时仍然采用item_id。

在训练中poi_id选择了299897。

日期的处理使用了java中的Calendar和SimpleDateFormat。传入参数的格式为"YYYY-MM-DD"，而数据库中格式为"YYYYMMDD"。将日期按前者格式转化成datekey，进行前溯60天处理后转化为后者格式。

二、训练

训练直接调用了spark.mllib.recommendation.ALS的train方法。ALS全称是Alternating Least Square（交替最小二乘法）。最小二乘法不用过多说明了。交互式最小二乘法解决的是二元的最优化问题。当一个表达式中有两个变量时，将一个变量固

定，求另一个变量的最优值；再将后者固定，求前者的最优值。如此交替，最终求得表达式的最优值。

三、获取结果

最终需求为一个由若干个形如（原物品，推荐物品，推荐度）的元组组成的集合。而训练后的结果为物品的描述向量，即上文中提到的item_factor。问题又转化成求向量的最近邻问题。

这个问题的求解使用了LSH(Locality Sensitive Hashing, 局部敏感哈希)。LSH的基本思想是，求解n维向量的最近邻时，将所有向量n个维度的值映射到n个hashmap上。与当前向量最近邻的向量很大几率在hashmap中相同或邻近的cell中。这样极大的减少了求解范围，降低了时间复杂度。

LSH的实现方式上，选择了soundcloud的开源实现。https://github.com/soundcloud/cosine-lsh-join-spark

四、写回

需要写回的有4个字段。除上面提到的（原物品，推荐物品，推荐度）外，还有推荐的日期。另外，还要将item_id转化成UPC_code。

喜欢此内容的人还喜欢

九盆给女孩子的恋爱小冷水
每日豆瓣

《唐探3》，到此为止也挺好
新闻哥