推荐系统-Item Based CF代码实例

原创 努力给自己看 码小白TM 2019-01-15

前边我们已经简单介绍了基于内容的推荐系统CB和基于协同过滤的推荐系统CF，今天我们就来看一个基于协同过滤中的基于物品的 Item Based CF 的一个实际实例来帮助大家更好的来了解和掌握以前的知识。

下面我们来看看我们的元数据，数据很简单，每一行由userId（用户ID）、itemId（物品ID）、score（用户打分）组成，之间用”，“分隔。

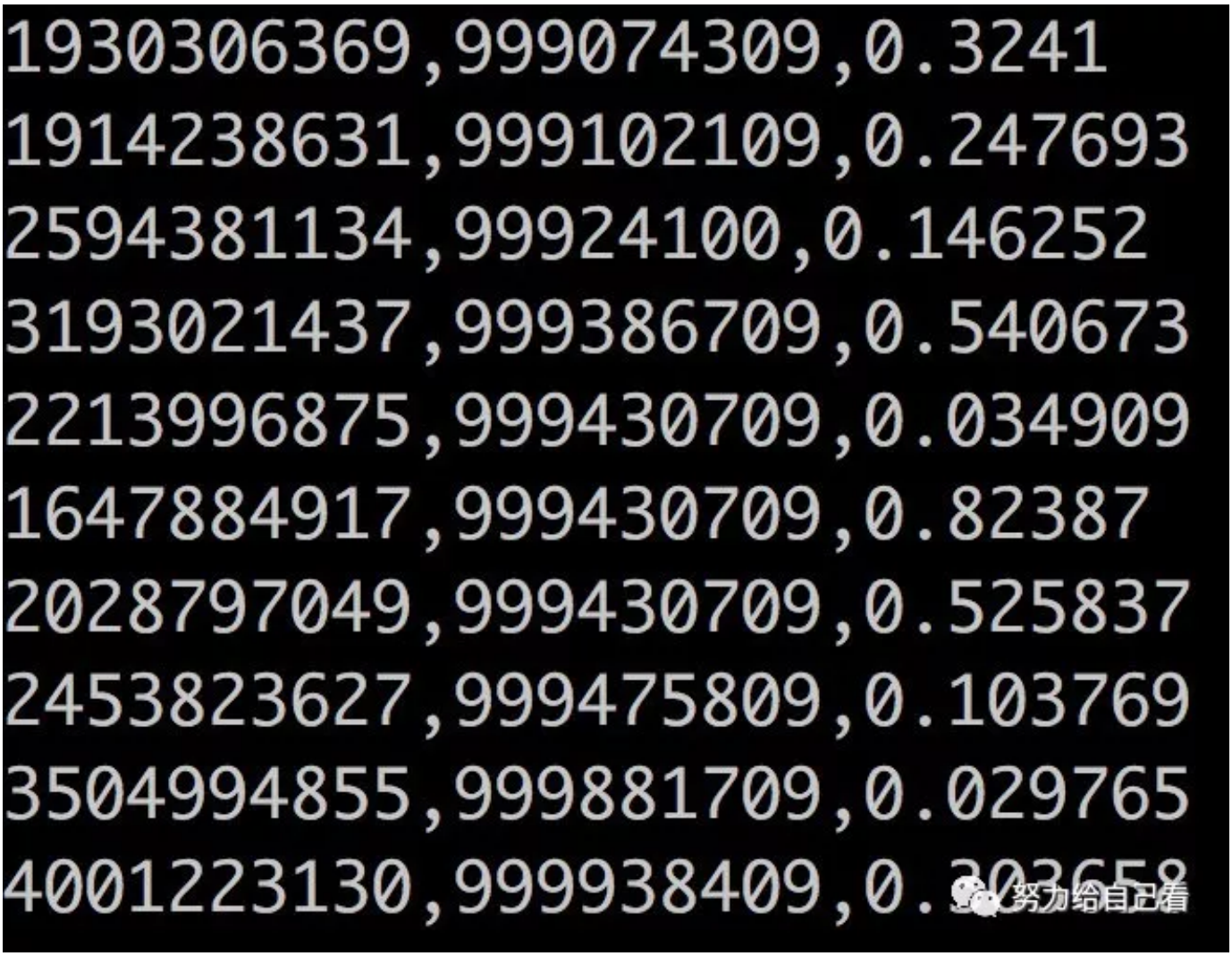

image-20190108223153096

我们计算的时候用下边这个相似度计算公式，这个公式其实本质上和cos相似度计算公式一样。

$$W_{i,j} = \frac{\left(\sum_{u \in U(i,j)} r_{ui} * r_{uj}\right) * \left(|U(i,j)| - 1\right)}{\sqrt{\sum_{u \in U(i,j)} r_{ui}^2 * \sum_{u \in U(i,j)} r_{uj}^2} * \left(|U(i,j)| - 1\right)}$$

其中：

Wi,j 表示标号为i和j的两个item的相似度

U(i,j) 表示同时对i和j两个有评分的用户的集合

rui 表示用户u对item i的评分

λ 为平滑参数

实际上我们在用的时候可以把分子分母相乘的后半部分当做一个常数舍去，对结果没有没有影响。那我们在计算的时候就可以只看前半部分了，通过分析我们就会发现，对于每个用户来说分母都是相同的，是所有用户对i的打分的平方和然后乘以所有用户对j的打分的平方和，而分子就是自己对i和j的乘积，我们分别把分母拆开，就可以得出其实就是自己对i的打分除以所有用户对i的打分的平方和（相当于归一化）然后乘以自己对i的打分除以所有用户对i的打分的平方和。由此我们代码实现的时候就很简单了。

我们举一个简单的例子来说明这个公式怎么应用

	item1	item2
A	2	5
B	1	3
C	4	2

我们要计算item1和item2的相似度，现在我们已经知道了所有同时对两个物品打分的用户A、B、C那么两个物品的相似度计算过程，首先把打分进行归一化，先求得所有用户对item1的打分的平方和 $2^2+1^2+4^2=20$ 然后求得所有用户对item2的打分的平方和 $5^2+3^2+2^2=38$ 然后对所有打分进行归一化后再分别相乘求和，最后的相似度为

$$相似度=\frac{2}{20}*\frac{5}{38}+\frac{1}{20}*\frac{3}{38}+\frac{4}{20}*\frac{2}{38}$$

那么我们现在就有了一个思路，首先把所有打分进行归一化计算，然后找出所有对i和j打分的集合，然后计算出i和j的相似度。

下面就是按照这个思路的代码实现，代码为python写的MapReduce任务。

- 归一化并两两取对过程

map1.py

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import math

item_score_dic = {}
user_item_score_list = []
for line in sys.stdin:
    ss = line.strip().split(',')
    if len(ss) != 3:
        continue
    user = ss[0].strip()
    item = ss[1].strip()
    score = float(ss[2].strip())
    user_item_score_list.append((user,item,score))
    score = pow(score,2)
    if item_score_dic.has_key(item):
        item_score_dic[item] += score
    else:
        item_score_dic[item] = score

for uis in user_item_score_list:
    user, item, score = uis
    if item_score_dic.has_key(item):
        score_sqr = math.sqrt(item_score_dic[item])
        print ('\t'.join([user,item,score/score_sqr]))
```

reduce1.py

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
import sys

current_user = None
item_score_list = []

for line in sys.stdin:
    ss = line.strip().split('\t')
    if len(ss) != 3:
        continue
    user = ss[0].strip()
```

```
        item = ss[1].strip()
        score = float(ss[2].strip())
        if not current_user:
            current_user = user
        if current_user != user:
            for i in range(0, len(item_score_list) - 1):
                for j in range(i+1, len(item_score_list)):
                    item_a, score_a = item_score_list[i]
                    item_b, score_b = item_score_list[j]
                    print('\t'.join([item_a, item_b, score_a * score_b]))
                    print('\t'.join([item_b, item_a, score_a * score_b]))
            item_score_list = []
            current_user = user

        item_score_list.append((item, score))

    for i in range(0, len(item_score_list) - 1):
        for j in range(i + 1, len(item_score_list)):
            item_a, score_a = item_score_list[i]
            item_b, score_b = item_score_list[j]
            print('\t'.join([item_a, item_b, score_a * score_b]))
            print('\t'.join([item_b, item_a, score_a * score_b]))
```

- item1和item2相似对求和阶段

map2.py

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
import sys

for line in sys.stdin:
    ss = line.strip().split('\t')
    if len(ss) != 3:
        continue
    item_a = ss[0].strip()
    item_b = ss[1].strip()
    score = ss[2].strip()
    print('%s#%s\t%s' % item_a, item_b, score)
```

reduce2.py

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
import sys

current_items = None
sum = 0.0

for line in sys.stdin:
    ss = line.strip().split('\t')
    if len(ss) != 2:
        continue
```

```
        item_item = ss[0].strip()
        score = float(ss[1].strip())
        if not current_items:
            current_items = item_item
        if current_items != item_item:
            item_a, item_b = current_items.split('#')
            print('\t'.join(item_a, item_b, sum))
            sum = 0.0
            current_items = item_item

        sum += score

    item_a, item_b = current_items.split('#')
    print('\t'.join(item_a, item_b, sum))
```

以上就是算法的代码实现过程，这个算法有一个缺点就是，当数据量非常大的时候，物品两两取对的数量会非常大，很容易内存不够，所以在实际的应用中我们应该随机取一定量的数据进行计算，而不是把所有的数据都加到计算里边来。

如果觉得好看，可以点一下 **好看** ，如果觉得对你有一点点帮助，可以**赞赏**作者一点，还可以推荐和分享给你的朋友

「努力给自己看」