基于 Hadoop 的推荐系统实践与可视化分析

范文骁
1120170346
北京理工大学徐特立学院
ripplesaround@sina.com

杜科松
1120172151
北京理工大学计算机学院
1120172151@bit.edu.cn

郭鑫
1120172199
北京理工大学计算机学院
1120172199@bit.edu.cn

韩世杰
1120172151
北京理工大学徐特立学院
1120172345@bit.edu.cn

1. 课程设计题目及背景意义

当今的时代，可谓是一个处处被推荐系统所影响的时代。想上网购物，推荐系统会帮你“挑选”满意的商品；想了解咨询，推荐系统回味你准备你感兴趣的新闻；想学习充电，推荐系统会为你提供最合适你的课程..... 纵观人类历史，推荐系统还从未像现在这样影响着人们的生活。故我们选择利用大数据与软件技术课程学到的知识，来去实现一个基于 Hadoop 的推荐系统，以期通过实践探究对人们影响如此之大的系统背后的奥秘。

2. 小组成员与分工

- 范文骁：组长，负责编写推荐算法（itemCF 部分，userCF）、链接各模块功能以及代码、论文整合；
- 杜科松：组员，负责编写数据预处理部分；
- 郭鑫：组员，负责编写推荐算法（userCF 部分）；
- 韩世杰：组员，负责编写可视化部分与 mysql 写入部分。

除上述分工外，每位同学也承担了自己实现部分的论文写作部分。

序号	模块	功能点	详细内容
1	数据预处理模块	接收原始数据	将购买记录读入
2		预处理并将结果保存	生成用户与物品的共现矩阵（包含用户属性）
3	推荐算法模块	读取预处理数据	
4		userCF 算法	输出推荐结果
5		itemCF 算法	输出推荐结果
6	mysql 存储模块	保存推荐结果	
7	可视化模块	分析数据模型特点	分析推荐物品的分布、超参数对模型的影响

表 1. 模块概述

3. 设计内容与系统架构

3.1. 系统架构

系统的逻辑架构如图1所示。

3.2. 模块概述

在我们的系统中，共分为 4 大模块，如表1所示。

4. 开发工具

在开发工具上，我们选用了 IntelliJ IDEA，Eclipse，PyCharm 等工具来实现我们的功能。

5. 具体实现

在部分，我们将介绍具体的模块实现方法。模块间的数据流动如图2所示。

5.1. 数据预处理模块

数据预处理的目的是将输入的数据转换成用户和商品的共现矩阵，框架如图3所示。

其中，

- Entry：接收的参数是购买记录文件路径和生成的用户商品矩阵的保存路径，控制整个数据预处理过程，如图4所示；
- DataClean：处理购买记录，生成用户购买商品列表，保存有中间文件/test/test2，如图5所示；
- GoodsList：处理购买记录，生成商品列表，保存有中间文件/test/test1，如图6所示；
- UserGoodsMatrix：通过中间文件/test/test1和/test/test2，生成用户商品矩阵，输出到保存路径，如图7所示。

5.2. 推荐算法模块

推荐算法模块的架构如图2中所示，其中 config 为读入的文件，是设置超参数用的。在该推荐系统中，推荐系统的推荐物品的数量（top_K），用户属性（年龄，性别）的权重均为超参数。

在该系统中，共用到了两种推荐算法，分别是基于用户的协同过滤算法（userCF）和基于物品的协同过滤算法（itemCF）。

5.2.1 基于用户的协同过滤算法

UserCF 算法的核心思想很简单，通俗的讲：要给 A 用户推荐物品，先从计算一下其他用户与 A 用户的用户相似度，比如说这里面 B，C 用户和 A 用户的用户相似度最高（意思就是说，B,C 与 A 的兴趣最相似），那么再从 B,C 用户中推荐一些 B，C 用户喜欢的，并且 A 用户没有购买的物品给 A 用户，从而实现个性化推荐，这就是所谓的基于用户的协同过滤算法。步骤如下：

- 找到和目标用户兴趣相似的用户集合——计算两个用户的兴趣相似度；
- 根据兴趣相似度选出 top_K 个相似用户；
- 利用 top_K 用户计算系统对物品的推荐度，生成最终推荐结果。

这里需要补充的是，在计算相似度的时候，我们将用户购买历史记录写成 0，1 向量的形式，然后计算余弦相似度，同时也要计算用户属性的相似度，然后将历史记录的余弦相似度和用户属性的相似度进行线性求和即可，即

$$\begin{aligned} total\_cos\_sim =& cos\_sim\_history \\ & - coefficient\_age * cos\_sim\_age \\ & + coefficient\_gender * cos\_sim\_gender \end{aligned}$$

在框架中，userCF 的实现被分类为 3 个 job，每个 job 实现的功能如下：

- userCF_job1：输入共现矩阵，在 reduce 过程中将需要计算的表达式存成字符串输入中间文件，代码框架如图8所示；
- userCF_job2：从中间文件获得用户相似度的计算表达式并计算用户相似度进而排名，将当前用户与和他相似的用户的信息存入中间文件，代码框架如图9所示；
- userCF_job3：根据相似用户的购买记录的加权平均值生成最终的推荐列表，代码框架如图10所示。

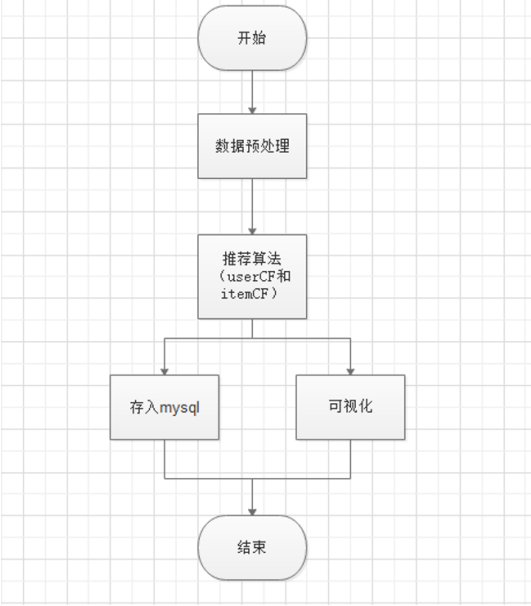

图 1. 系统逻辑架构

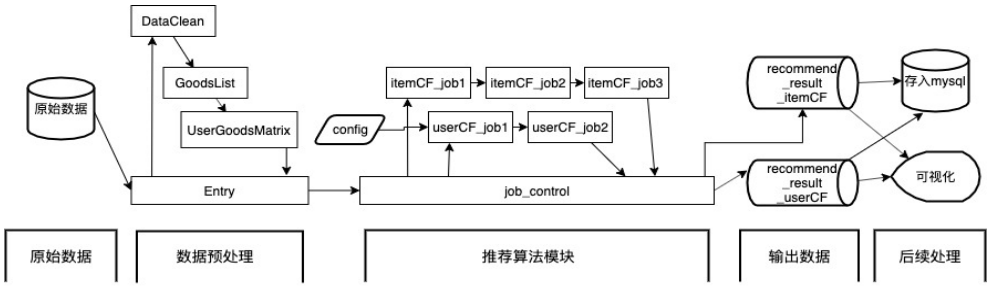

图 2. 推荐系统数据流动。其中箭头表示数据流动方向，图中没有标出中间的临时存储文件。

5.2.2 基于物品的协同过滤算法

基于物品的协同过滤算法在实现方法上与 userCF 相似，都需要先计算相似度然后排序生成最终推荐列表。由于在数据中没有记录物品的属性，所以相似度的计算直接采用购买记录的余弦相似度。

在框架中，itemCF 的实现被分类 2 个 job，每个 job 实现的功能如下：

- itemCF_job1：输入共现矩阵，在 combine 的过程中将需要计算的表达式存成字符串，在 reduce 中计算用户相似度并排名，代码框架如图11所示；
- itemCF_job2：根据相似物品的购买记录和当前用户的购买记录的加权平均值生成最终的推荐列表，代码框架如图12所示；

5.3. mysql 存储模块与可视化模块

在 mySQL 数据库中，我们定义数据的最终存储形式为两张关系表与两张信息表，关系表分别负责记录通过 itemCF 和 userCF 方式为每个用户 id 推荐的 top_K 项商品的 id，两张信息表分别负责存储客户与商品对应的存储信息。在 sql 中对其进行初始化的代码如 sqltxt 所示，最终建成的四张表格的具体数据形式如13所示。

在数据库初始化完成后，我们编写了 python 函数分别对最终得到的推荐结果矩阵 recommend_result_userCF.txt 和 recommend_result_itemCF.txt 进行读取，利用 pymysql 库

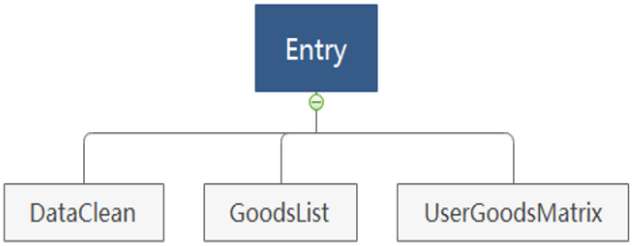

图 3. 数据预处理模块框架

```
package DataClean;

import java.io.BufferedReader;

public class Entry {
  public static void main(String[] args) throws Exception{
}
```

图 4. Entry 的具体实现

配置连接了本地的 SQL 数据库，分别对两个 result 表进行了数据存入。然后，我们利用 matplotlib 库及 SQL 的相关查询功能将最终结果可视化，使得两种方法最终推荐出的商品分布情况以及对单个用户的推荐情况可以被形象直观地展示。

同时，除了对推荐结果进行可视化之外，可视化模块可以实现对不同的超参数得到的推荐结果进行的对比分析，实现方法与之前可视化结果的方法大致无异，在此不在赘述，关于不同的超参数得到的推荐结果，将在下一章测试结果中展示。

5.4. 系统控制功能

此部分实现的功能为对用户的输入的操作码执行对应的操作，操作除了调用之前的各个模块外，还添加了一些对 hdfs 文件的基本处理，这些基本处理的实现方法在本学期之前的实验中都已经详述，这里不再赘述，整体代码结构如图15所示。

6. 系统演示与测试结果

7. 实验心得

```java
package DataClean;

import java.io.IOException;

public class DataClean{
    public static void main(String[] args) throws Exception {

    public static class Map extends Mapper<Object, Text, Text, Text>{
    public static class Reduce extends Reducer<Text, Text, NullWritable, Text>{

    }
```

图 5. DataClean 的具体实现

```java
package DataClean;

import java.io.BufferedWriter;

public class GoodsList{
    static Text v= new Text();
    static StringBuffer sb =new StringBuffer();

    public static void main(String[] args) throws Exception {

    public static class Map extends Mapper<Object, Text, Text, Text>{
    public static class Reduce extends Reducer<Text, Text, NullWritable, Text>{
}
```

图 6. GoodsList 的具体实现

```java
package DataClean;

import java.io.BufferedReader;

public class UserGoodsMatrix {
    static String v= new String();
    static String[] goodslist;
    public static void main(String[] args) throws Exception {

    public static class Map extends Mapper<Object, Text, Text, Text>{

    public static class Reduce extends Reducer<Text, Text, NullWritable, Text>{
}
```

图 7. UserGoodsMatrix 的具体实现

```java
public class userCF_job1 {

    public static class TokenListMapper extends Mapper<LongWritable, Text, Text, Text> {...}
    public static class ListCombiner extends Reducer<Text, Text, Text, Text> {...}
    public static class ListReducer_cal_sum extends Reducer<Text, Text, Text, Text> {...}
}
```

图 8. userCF_job1 的具体实现

```
public class userCF_job2 {
    public static  class TokensListMapper extends Mapper<LongWritable, Text,Text,Text> {...}
    public static  class usercfjob2Listcombine extends Reducer<Text,Text,Text,Text> {...}
    public static  class usercfjob2ListReducer extends Reducer<Text,Text,Text,Text> {...}
}
```

图 9. userCF_job2 的具体实现

```
public class userCF_job3 {
    public static class InputListMapper extends Mapper<Object, Text, Text, Text> {...}

    public static class finalListReducer extends Reducer<Text, Text, Text, Text> {...}
}
```

图 10. userCF_job3 的具体实现

```
public class itemCF_job1 {
    public static class TokenizerMapper extends Mapper<Object, Text, Text, Text> {...}

    public static class Combine_toVector extends Reducer<Text, Text, Text, Text> {...}

    public static class myReducer extends Reducer<Text, Text, Text, Text> {...}
}
```

图 11. itemCF_job1 的具体实现

```
public class itemCF_job2 {
    public static class InputMapper extends Mapper<Object, Text, Text, Text> {...}

    public static class finalReducer extends Reducer<Text, Text, Text, Text> {...}
}
```

图 12. itemCF_job2 的具体实现

```
mysql> desc item_CF_result;
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| user_id    | int(10) | YES  |     | NULL    |       |
| recommend1 | int(10) | YES  |     | NULL    |       |
| recommend2 | int(10) | YES  |     | NULL    |       |
| recommend3 | int(10) | YES  |     | NULL    |       |
+------------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc user_CF_result;
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| user_id    | int(10) | YES  |     | NULL    |       |
| recommend1 | int(10) | YES  |     | NULL    |       |
| recommend2 | int(10) | YES  |     | NULL    |       |
| recommend3 | int(10) | YES  |     | NULL    |       |
+------------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc customer;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| user_id   | int(10)     | NO   | PRI | NULL    |       |
| user_name | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc item;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| item_id   | int(10)     | YES  |     | NULL    |       |
| item_name | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

图 13. SQL 建表结果

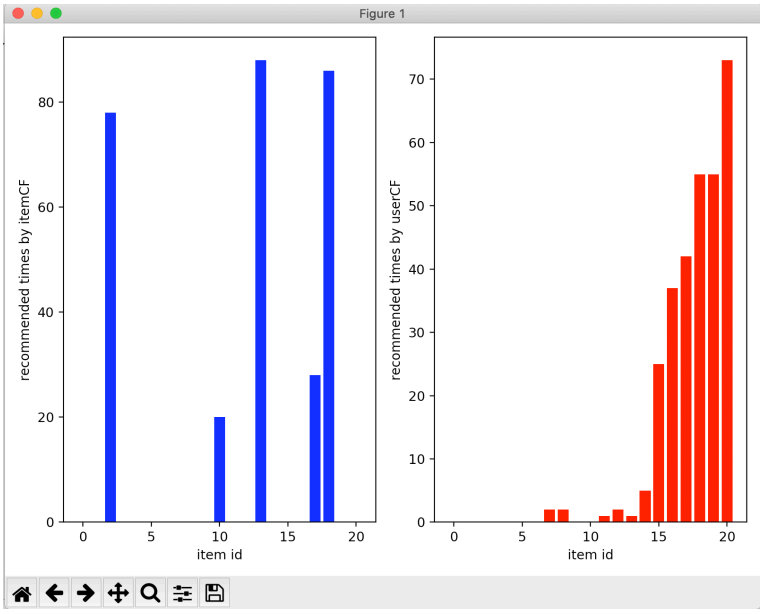

图 14. 两种方法推荐结果分布情况，其中左侧的分布是基于 itemCF 推荐的结果，右侧的分布是基于 userCF 推荐的结果。

```java
public class rec_sys_main {
    public static void main(String[] args) throws Exception {
        System.out.println("————————————————————————————");
        System.out.println("欢迎使用本推荐系统");
        String input_op = "init";
        Scanner input= new Scanner(System.in);
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(URI.create("recinput"), conf);
        int rec_or_not = 0;
        while (true){
            System.out.println("————————————");
            System.out.println("\t可选操作");
            System.out.println("put\t\t\t\t: 上传数据集");
            System.out.println("data_preprocess\t: 执行数据预处理");
            System.out.println("config\t\t\t: 修改超参数");
            System.out.println("recommend_alg\t: 运行推荐算法");
            System.out.println("result\t\t\t: 查看推荐结果");
            System.out.println("clean\t\t\t: 初始化");
            System.out.println("请输入您的操作: ");
```

图 15. 系统控制功能