测试文档

测试目的

* 通过测试，发现算法模型的输出是否合理
* 检查系统对算法的处理能力
* 为模型的合理性提供评估依据
* 根据运行的效率，提出模型优化

测试背景

目前算法模型已经建立完毕，将进入试运行和验收阶段。现在对算法的整个测试流程做一个总结。

测试环境

**硬件环境：**

window10

16G内存

256G硬盘

**软件环境：**

python+numpy+sklearn

模型测试

单元测试

我们测试了全部函数方法，测试覆盖率为86%。

模型测试

我们对不同模型进行分别集成测试，包括融合模型。

**1、模型一测试**

输入:待测商品id

输出：搭配子集1

使用数据集：用户历史购买表

算法运行时间：21.75s

**2、模型二测试**

输入：待测商品id

输出：搭配子集2

使用数据集：全部商品集

①未采用聚类：

数据：计算出全部商品的tf-idf 共88G

算法运行时间：24min

②采用聚类后：

将所有数据分成10个类，计算tf-idf时，只需要计算原本的1/10

算法运行时间：14min

对比二者，可以获得，采用聚类缩小数据的范围，程序运行的时间降低了几乎一半，说明聚类是有效的。

**3、模型三测试**

输入：待测商品id

输出：搭配子集

使用数据集：搭配套餐表

算法运行时间：

内存/CPU占比时间：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 平均 |
| 算法运行时间/s | 1018.88 | 1031.87 | 1025.5 | 1025.41 |
| CPU占用时间/s | 1013.73 | 1029.70 | 1024.59 | 1022.67 |

**4、融合测试**

输入：待测商品id

输出：最终的商品搭配集

使用数据集：模型一、二、三的搭配子集、类间表

算法运行时间：58.52s

内存/CPU占比时间：54.375s

测试总结

在单元测试中，我们测试了全部的函数方法，根据测试所有的函数均实现其功能，没有bug.在模型测试中，分别测试了三个模型以及最后的融合测试，三个模型均可以正确的输出待测商品的搭配子集。同时，我们对算法的运行时间进行评估和算法改善。针对模型2，我们通过提前聚类，缩小数据集的方法，提高了算法的效率。针对模型三，通过提前计算所有的相似度，保存在内存中而不是每次重新计算相同搭配商品与待测商品的相似度。提高了算法的运行速度。融合模型的测试，表明三个模型可以进行融合，得到最终的搭配集。