**高级算法课程**

**实验报告**

**实验一：用minHash实现集合的相似性连接**

姓名：朱宸慷

学号：2021110908

班级：2103103

评分表：（由老师填写）

|  |  |
| --- | --- |
| 最终得分： | |
| 对实验题目的理解是否透彻： | |
| 实验步骤是否完整、可信 ： | |
| 代码质量 ： | |
| 实验报告是否规范 ： | |
| 趣味性、难度加分 ： | |
| 特 色： | 1 |
| 2 |
| 3 |

**一、实验题目概述**

分别实现两重循环算法和minHash方法求解如下计算问题，利用教师发布在教学QQ群中的三个公开实验数据集(AOL, DILICIOUS, LINUX)开展对比实验和扩展性实验，观察Hash函数个数对算法性能的影响，并根据实验结果讨论经验参数设置办法。

**二、对实验步骤的详细阐述**

整体步骤

1. 实现Naïve算法(两重循环计算交集大小)
2. 实现minHash算法求解问题，比较两种算法的计算结果是否一致，并解释观察到的现象
3. 设置不用的Hash函数个数，比较返回结果的差别和运行效率的差别，总结得出Hash函数个数的最佳经验设置公式
4. 更换数据集，验证上述公式的一致性
5. 抽取数据集中不同比例的数据子集开展实验，讨论方法的扩展性
6. 撰写实验报告

collector思想

1. 从文件中读取形如<setId,setItem>的数据集，构建形如<setId,[setItem]->set>的哈希表，实现一个可复用的数据收集器。
2. 根据入参确定路径以及读取方式，包括在：算法开始前读取数据，比较时读取minhash结果，比较时读取naïve结果
3. 不能直接存取类成员，需要通过get\_keys(),get\_allvalues()方法获取成员

minHash算法

1. 从哈希表中随机抽样n\_sample项，提取这些key对应的value的集合，将这些集合合并后去重，称为allValue数组。
2. 随机打乱allValue数组，构成一个哈希函数。此处由于allValue足够大，可以相信其在每次实验中都是互异的，为了降低复杂性不再检查是否产生重复。
3. 从前遍历allValue数组，记录元素的index，基于集合的O(logn)查找来判断该元素在哈希表的value中是否存在。若存在，则将index记录为该set在该哈希函数下的minHash值
4. 对本次抽样的所有集合都计算一轮本哈希函数下minHash值并保存
5. 完成所有哈希函数的计算后计算相似性并输出

Naïve算法

1. 从哈希表中随机抽样n\_sample项，此处步骤由collector完成，保证与minHash算法使用相同的数据
2. 分别计算每两个集合之间的相似性并输出

**三、实验数据**

**1. 实验设置**

**实验环境**：

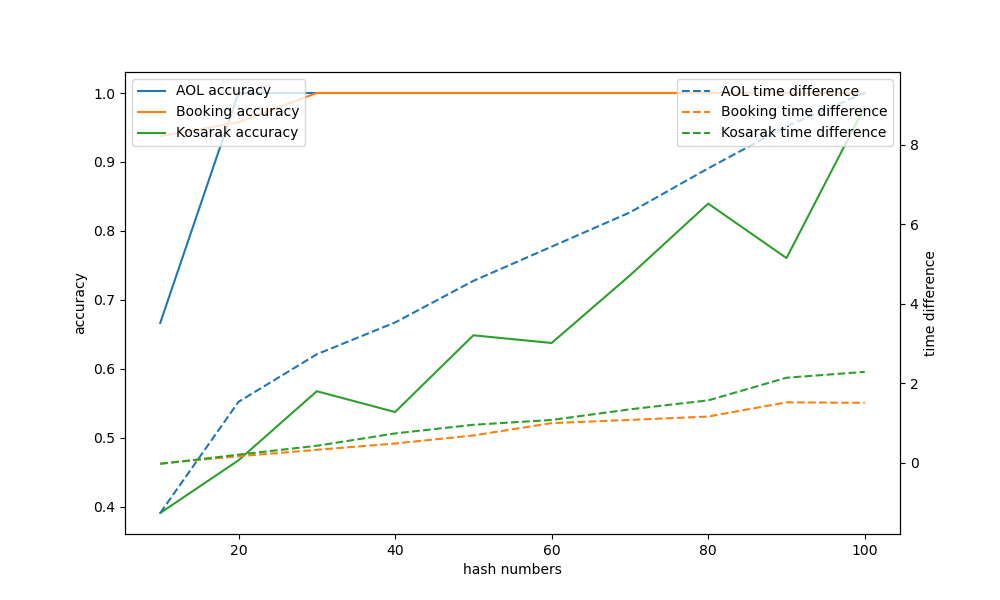
Python 3.11

Windows 11

**数据**：



**2. 实验结果**

****

**四、对实验结果的理解和分析**

1. 随着哈希函数数量的增加，三个数据集的准确率都在提升
2. 算法在AOL和Booking数据集的表现良好，在Kosarak上则一般。根据在过程中生成的文件可以推断以下结论
   1. 前二者相似的集合较少，容易达到较高的正确率

图形用户界面, 应用程序

描述已自动生成 图形用户界面, 文本, 应用程序

描述已自动生成

* 1. Kosarak数据集即使在100个哈希函数的情况下，仍然会产生四百余个相似的集合，在比较准确率的基数上就不占优势

手机屏幕的截图

描述已自动生成

1. Naïve算法在AOL数据集上表现良好，在另外两个数据集表现一般。在哈希函数大幅增加的情况naive算法下与minHash算法的时间差基本保持稳定
2. 综合来看，数据集Kosarak比较有参考价值。因此可以得出结论，哈希函数在80左右较好，此时准确率超过80%且到达极值点。此时更改哈希函数数量会导致准确率下降。如果将哈希函数的数量提升到100左右，那么一方面运行时间会大幅增加，并且此时对于准确率的提高也有限，同时相对于naïve算法的时间优势也在降低。

**五、实验过程中最值得说起的几个方面**

1. 编写代码时算法的时间复杂度经过了多次优化。其中包括收集数据后O(logn)建立集合，基于集合使用O(logn)查找来代替O(n)的遍历等集合操作来代替数组遍历操作。
2. Collector模块的设计理念是复用代码，返回哈希表。但是在这一思想的前提下，过程中生成的中间文件的结果三元组(setId,setId,[simItems]->set)不能很好的满足构建哈希表的要求，于是舍弃能精确计算准确率的[simItems]->set部分，从而使Collector模块代码可以重入。
3. 本实验为了保证执行效率通过调用Numba库将部分函数编译为了机器码，跳过了python解释器的翻译步骤，从而加快执行效率。
4. 目的同上，本实验还有一个部分完成的Go语言实现。