数据挖掘第二次作业

李晨昊 2017011466

2020 年 2 月 29 日

目录

1 数据属性类型练习

- 教师的职称：Ordinal，职称是一组离散的值，且可以比较大小（职称有高低之分）
- 手机号码：Nominal，手机号码是一组离散的值，且无法比较大小（其大小关系无意义），也不能运算
- 体重：Ratio，体重可以进行各种运算
- 出生日期：Ordinal，出生日期是一组离散的值，且可以比较大小
- 出生地：Nominal，出生地是一组离散的值，且无法比较大小，也不能运算
- 年龄：Interval，年龄的乘除运算无意义（或者说实际中很少用到）

2 计算统计信息

1.
   - 均值：28.01875
   - 中位数：28.1
   - 众数：26.5
2.
   - 最小值：7.8
   - 第一四分位数：26.5
   - 中位数：28.1
   - 第三四分位数：33.575
   - 最大值：43.0

   盒图见1。

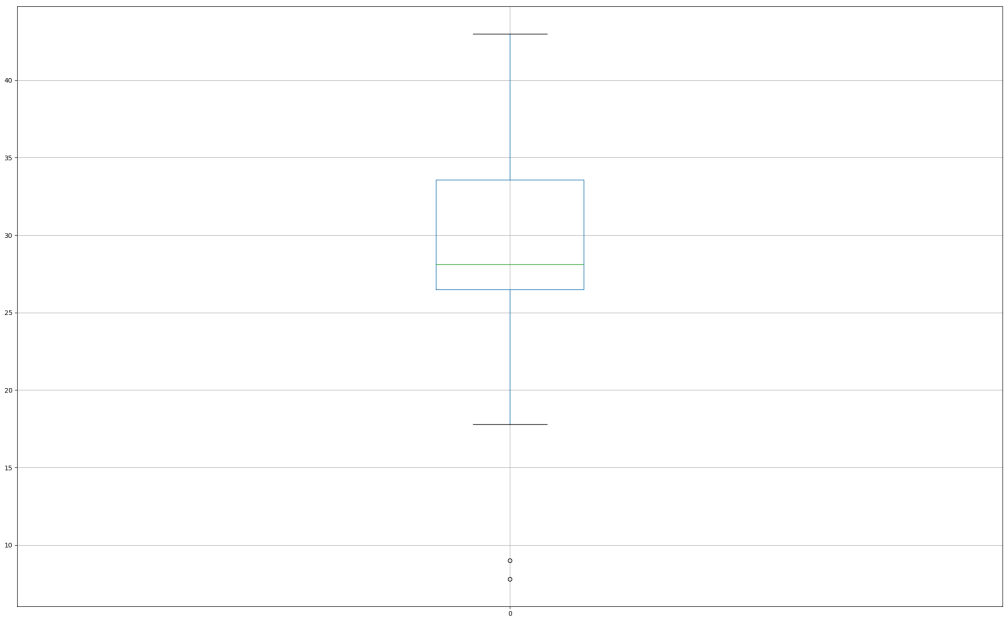

图 1: 盒图

3 文本数据的表示

我使用 Rust 完成本实验，代码和数据都附在了提交的 doc 文件夹中，在该文件夹中执行`cargo run --release`即可运行程序。

3.1 实现思路

3.1.1 词典构造

首先读入每个文件的全部内容，存储在一个Vec<String>中，这样后续使用的字符串来自其中的引用，可以减少复制的开销。

接着为每个文件建立词典，方法是先删除所有除了字母，数字和空格的内容，即删除标点符号，然后按照空格分词，然后把这些字符串插入散列表中，同时维护计数。

与此同时，构建全局的词典和共现矩阵，方法都是平凡的。共现矩阵定义为HashMap<&str, HashMap<&str, u32>>，其中可以考虑每个词和自身的共现，也可以不考虑，经测试结果基本没有区别，所以我选择了考虑，这样编写简单一些。

在建立了全局的词典之后修改每个文件的词典，依据每个词的计数值计算出 tf-idf 值。

3.1.2 距离计算

文件相似度通过每个文件的词典来计算。将每个文件的词典的列表按照距离来排序，选择前几个即可。排序通过 Rust 提供的sort_by_cached_key，这样可以避免重复计算。Euclidean 距离的计算通过枚举给定文件的词，将其 tf-idf 值与目标文件对应词的 tf-idf 值（若这个词不存在则取 0）相减，平方，再求和；再加上目标文件中不在给定文件中的词的 tf-idf 值的平方和。结果不需要开根号，因为只是为了比较大小。

Cosine 距离的计算通过枚举给定文件的词，将其 tf-idf 值与目标文件对应词的 tf-idf 值（若这个词不存在则取 0）相乘，再求和，得到向量点乘的值；再除以两个文件中各自所有词 tf-idf 值的平方和的积的平方根，得到 cos(向量夹角)，这个值与向量间的相似程度成正相关，因此对它取负，再排序，这样前几个元素就是相似程度最高的。

词语相关度通过共现矩阵来计算。计算方式与上面类似，不再重复。

3.2 结果与分析

运行程序得到如下输出：

```
doc similarity with doc 52
by euclidean distance: [31, 256, 258, 194, 212]
by cosine distance: [31, 258, 256, 194, 97]
word similarity with word ”crime”
by euclidean distance: [”trial”, ”college”, ”Sebastian”, ”Randy”,
   ”revealed”]
by cosine distance: [”trial”, ”college”, ”lawyer”, ”say”, ”
   Sebastian”]
```

其中分析了与名称为 52 的文章相似的文章和与词 crime 相关的词。这两个值都是随意取的，可以通过修改代码来替换成别的值。

文章部分，文章 52 看起来像是节目预告或者新闻之类的文章，其中依据时间记录了一系列事实，许多事实与犯罪相关。文章 31 是完全类似的，所以两种方法都把它排在第一位。还有几个前几名的文章也是类似的，还有几个文章只是记录犯罪内容的，相关度较低一些，也排在前几名。

词语部分，找到的 trial 显然与 crime 关系很大，还有 lawyer 的关系也比较大。其他词的关系看起来没那么大，可能是人名或者地名，猜测也许在几篇记录犯罪的文章中都出现了它们。