**中山大学数据科学与计算机学院**

**移动信息工程专业-人工智能**

**本科生实验报告**

**（2017-2018学年秋季学期）**

课程名称：**Artificial Intelligence**

|  |  |  |  |
| --- | --- | --- | --- |
| 教学班级 | **15M1** | 专业（方向） | **移动互联网** |
| 学号 |  | 姓名 | **Jw** |

# 实验题目

**文本数据集的简单处理**

# 实验内容

1. 算法原理

**One hot：**One-hot 编码就是独热编码，就是用N位寄存器来对N个状态进行编码，每个状态都由它独立的寄存器位表示，并且在任意的时候，其中只有一位有效。可以这样理解，对于每一个特征，如果它有m个可能值，那么经过独热编码后，就变成了m个二元特征。并且，这些特征互斥，每次只有一个激活。因此，数据会变成稀疏的。这样做的好处是：

1.解决分类器不好处理属性数据的问题。

2.在一定程度上也起到了扩充特征的作用。

在本次实验中，用一个向量表示一篇文章，向量的长度为词汇表的大小，1表示存在的单词，0表示不存在，所以词汇就相当于前面说的状态。将结果保存在onehot.txt中。

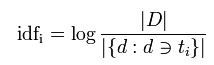
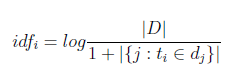
**TF( Term Frequency):** TF即词频，在一份给定的文章里，词频就是某一个词在该文章中出现的次数。这个数字通常需要归一化，以防止它偏向长的文章。（同一个词语在长文件里可能会比短文件有更高的词频，而不管该词语重要与否。）对于在某一特定文章里的词语 t i来说，它的重要性可表示为：



以上式子中 n i ,j是该词在文章d j中的出现次数，而分母则是在文章d j中所有字词的出现次数之和。

在本次实验中，TF矩阵就是将one-hot矩阵中的每一行的各个值都表示为该值/该行有值的个数。将结果保存在TF.txt中。

**TF-IDF(** term frequency–inverse document frequency**)**:这是在TF的基础上加入IDF的字词重要性评估方法。字词的重要性随着它在文章中出现的次数成正比增加，但同时会随着它在语语料库中出现的频率成反比下降。IDF是逆向文件频率，可以由总文章数目|D|除以包含该词语之文章的数目， 再将得到的商取对数，得到

 或者 

然后

/mathrm{tf{}idf_{i,j}} = /mathrm{tf_{i,j}} /cdot  /mathrm{idf_{i}} 

将结果保存在TFIDF.txt中。

**稀疏矩阵三元顺序表:** 前面说的one-hot矩阵是稀疏的，它的非0元素很少，而且分布

没有规律，如果用二维数组来存储，会造成大量的空间浪费，所以我们对稀疏矩阵一般 采取压缩存储的方法，即只存储其非零元素，其中一种常用方法就是使用三元顺序表。

三元组是指形如(x,y,z)的集合，其中x和y表示该元素在第x行和第y列，z为该元素

的值。假设以顺序存储结构来表示三元组表，就称为顺序三元组表。一个例子如下：



本次实验中需要将one-hot矩阵表示成三元顺序表并保存在smatrix.txt中

1. 伪代码

代码采用python来写

**1.文本数据处理**

**Begin**

**file = 打开相应的数据文件**

**lines = file.readlines() /\*读取文本中每一行数据，每一行是一个list\*/**

**for line in lines： /\*遍历每一行\*/**

**for word in line: /\*遍历每一行的每个单词**

**IF word in Hash：**

**Hash[word] = 这个word在全部文章中是第几个出现的**

**If word 是第一次出现在当前文章：**

**Num[word]+1**

**Num[] = 相应的IDF值**

**/\*获取每个矩阵，c1是one-hot，c2 是 one-hot三元顺序表，c3是TF，**

**C4是TFIDF**

**\*/**

**[c1,c2,c3,c4] = getContent(lines,Hash,num)**

**输出c1,c2,c3,c4到相应文本中去**

**End**

**getContent(lines,Hash,num):**

**初始化c1,c2,c3,c4为空**

**c2[0] ,c2[1], c2[2] = 0 /\*这三个是三元组的行数、列数、数值个数**

**for line in lines: /\*遍历每一行\*/**

**for word in Hash：/\*遍历词汇表\*/**

**if word 在当前行中出现：**

**更新c2中的行数、列数的最大值**

**增加c2中的数值个数**

**增加一组三元组到c2**

**c1增加1到当前行的末尾**

**c3增加当前行word出现的次数到当前行的末尾**

**c4增加前行word出现的次数\*IDF到当前行的末尾**

**else:**

**c1、c2、c3 增加一个0到当前行末尾**

**将c3和c4中的当前行的每个值 / 当前行的非0元素的个数**

**return [c1,c2,c3,c4]**

**2.AplusB.**

**Begin**

**A = 读入的A三元顺序表**

**/\*A[0] ,A[1],A[2]分别为行数、列数、数值个数，A[3]起是一三元组，用list表**

**示，下同**

**\*/**

**B = 读入的B三元顺序表**

**C[0] = max(A[0],B[0])**

**C[1] = max(A[1],B[1])**

**hash = [] /\*存储最后一共有哪几个二维坐标\*/**

**d = DICT()**

**/\*字典，判断(x,y)是否出现过，d[(x,y)]为该坐标代表元素的最终值\*/**

**for i in range(3,len(A)):**

**key = (A[i][0],A[i][1]) /\*以三元组前两位为关键字\*/**

**if key没出现过：**

**d[key ] = A[i][3]**

**hash.append((A[i][0],A[i][1])**

**else:**

**d[key] += A[i][3]**

**对B的处理同A，但是当d[key]=0时要删掉**

**对hash 中的二元坐标 按第一位为第一关键字，第二位为第二关键字排序**

**C[3] = len(d)**

**按照hash中的坐标顺序一个一个将三元组加入到C**

**return C**

**End**

1. 关键代码截图（带注释）

**1.文件数据处理**

**数据读入及预处理：**



**计算各个矩阵:**



**2.A plus B**



1. 创新&优化部分

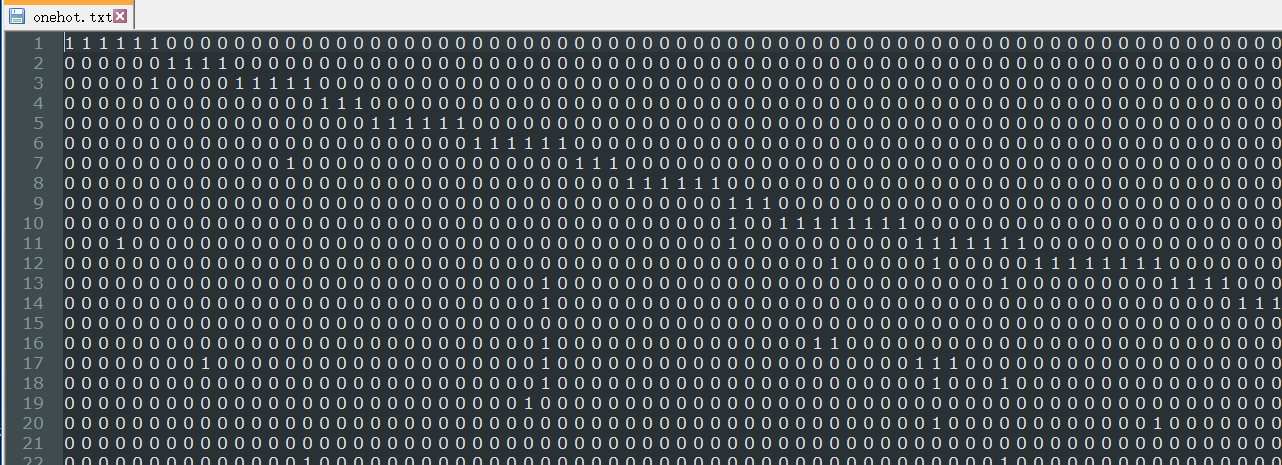
本次实验是简单的数据处理，没有什么好优化的地方。

# 实验结果及分析

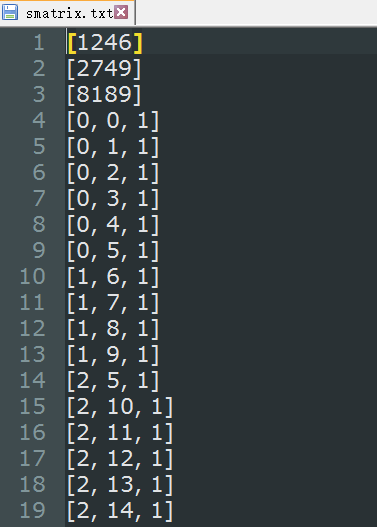
1. 实验结果展示示例（可图可表可文字，尽量可视化）

**注：所有运算涉及到小数的最后结果均保留12位小数**

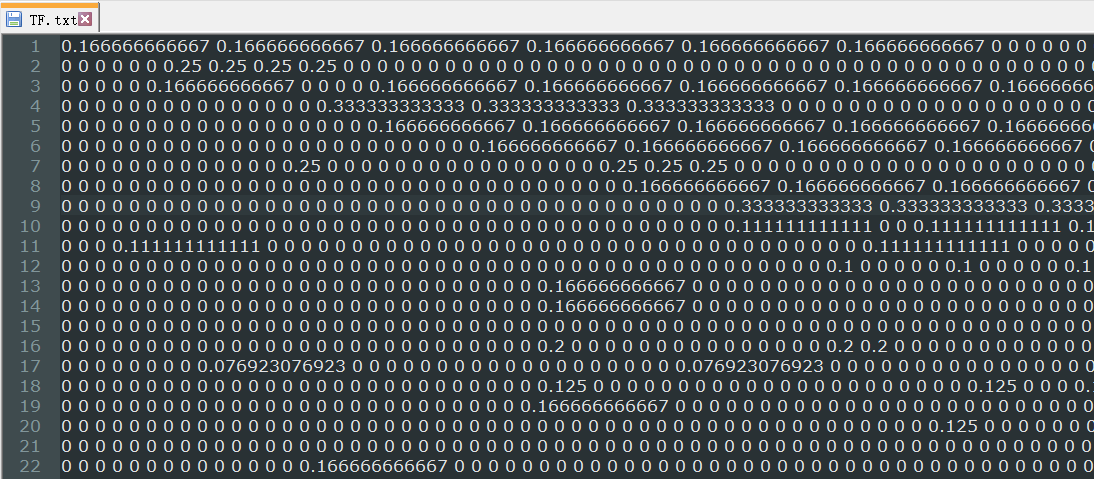
**One-hot：**由于结果数字太多，这里仅展示部分结果，完整结果请查看onehot.txt



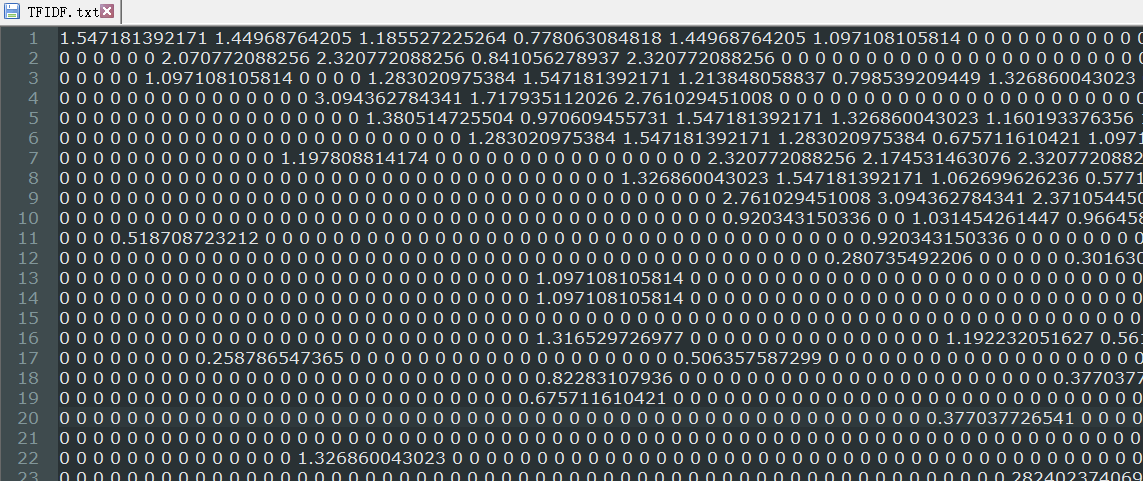
**samtrix：**由于结果数字太多，这里仅展示部分结果，完整结果请查看samtrix.txt



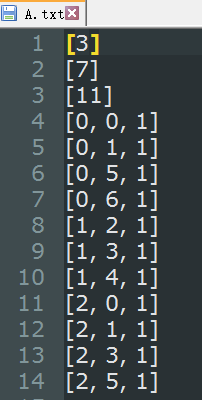
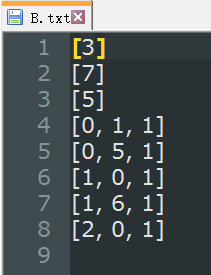
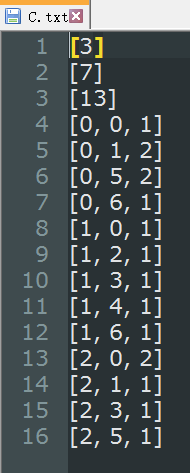
**TF：**由于结果数字太多，这里仅展示部分结果，完整结果请查看TF.txt



**TFIDF(IDF中分母+1)：**由于结果数字太多，这里仅展示部分结果，完整结果请查看TFIDF.txt



**AplusB（只支持整数，输入格式严格如下）：**

 **A:** **B: C:**

**2. 评测指标展示即分析（如果实验题目有特殊要求，否则使用准确率）**

1.对于one-hot, TF, TFIDF, 三元顺序表来说，主要看求出的矩阵的每一个元素的位置

以及值是否准确。求出的结果与其他一些同学比较后基本一致

2.对于两个三元系数矩阵相加，看结果是否正确。

# 思考题

**1. IDF 的第二个计算公式中分母多了个 1 是为什么？**

答：防止因为涉及到的词语不在语料库中而导致分母为0，进而使计算出错的情况。

**2. IDF数值有什么含义？TF-IDF数值有什么含义？**

答：1. IDF数值是逆向文件频率，是一个词语普遍重要性的度量。算法是log(总文档

数/(出现该词语的文件数+1))，由该公式可以看出，一个词语出现的次数越小，

它的IDF值越大，这说明该词语具有良好的类别区分能力。

2. TF-IDF是综合了TF和IDF的优点，IDF是可以很好地找被少数文件包含的词

语。但如果某一类文档C中包含词条t的文档数为m，而其它类包含t的文档

总数为k，显然所有包含t的文档数n=m+k，当m大的时候，n也大，按照

IDF公式得到的IDF的值会小，就说明该词条t类别区分能力不强。但是实际

上，如果一个词条在一个类的文档中频繁出现，则说明该词条能够很好代表这

个类的文本的特征，这样的词条应该给它们赋予较高的权重，并选来作为该类

文本的特征词以区别与其它类文档。这就是IDF的不足之处.这是应该增加词

频即TF值，它是一个词语在某个文件出现的频率，TF-IDF = TF\*IDF

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语

**3. 为什么要用三元顺序表表达稀疏矩阵？**

答：如果不用三元顺序表，而采用二维数组来表达稀疏矩阵，会造成大量的空间浪

费，因为这个二维矩阵中有用的元素非常少。