**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 | **15352427** | 姓名 | **张子豪** |

# 实验题目

**文本数据集的简单处理**

# 实验内容

1. 算法原理
   1. 读取数据

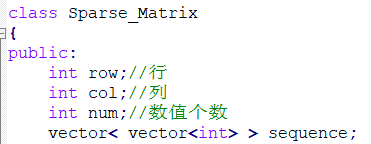
利用C++的ifstream，将所有原始数据读入vector<string>类型的变量存储起来，然后将本次实验不需要的部分，即序号和情感值剪掉，这里我采用的方法是寻找我们需要的词汇表在初始文本中的起点位置，然后利用substr函数来获取我们需要的子字符串；然后利用istringstream将获取到的子字符串的单词逐个读取出来，存入vector< vector<string> >类型变量，每个vector<string>代表一个训练文本，这个vector里的每个string则是文本中的单词。

* 1. 统计数据

遍历所有文本记录不重复词语，利用map来记录每个词语出现的次数（同一文本出现多次算一次），然后根据两个不同的公式算出两组IDF向量，最后遍历每个文本并对比之前记录下的不重复词语，完成矩阵的写入。

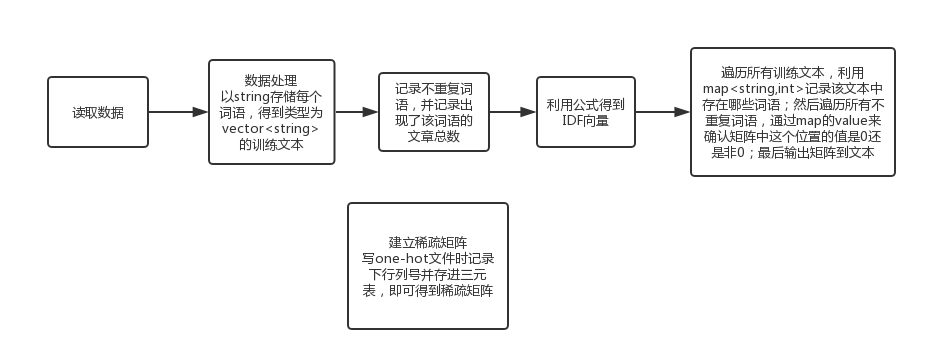
* 1. 三元顺序表

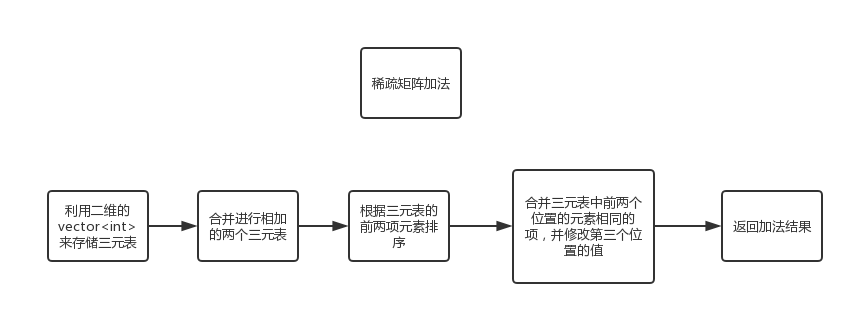
我用类封装了一个三元表，利用vector< vector<int> >来存储三元表，如下



三元表加法实现则是重载了+号，加法结果的行、列与进行相加的两个矩阵A、B的行、列均相等；将A、B的三元表都push\_back进同一个vector< vector<int> >，然后自定义一个cmp利用sort对这个vector进行排序：比较三元表第一个元素，相等则比较第二个元素，均相等则按照默认顺序；排序完成后遍历一遍合并第一、二个元素相等的项，这样就完成了+号的重载。另外我还重载了+=和<<。

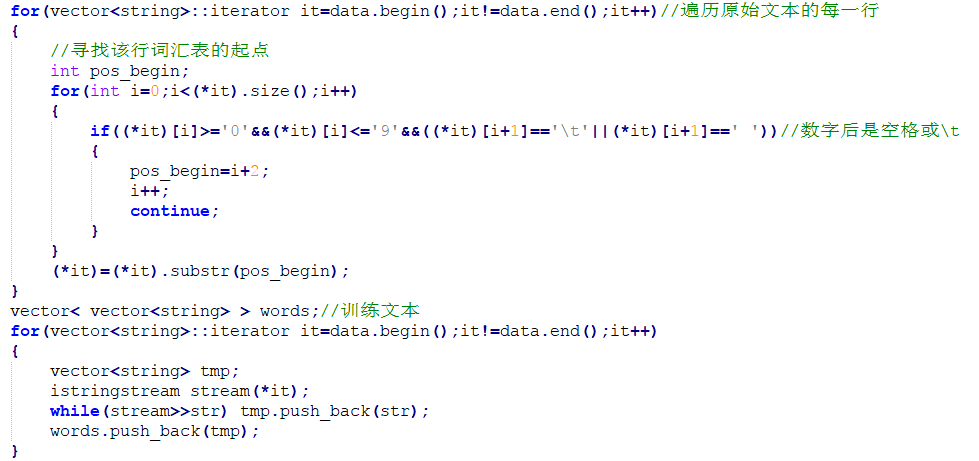
1. 伪代码



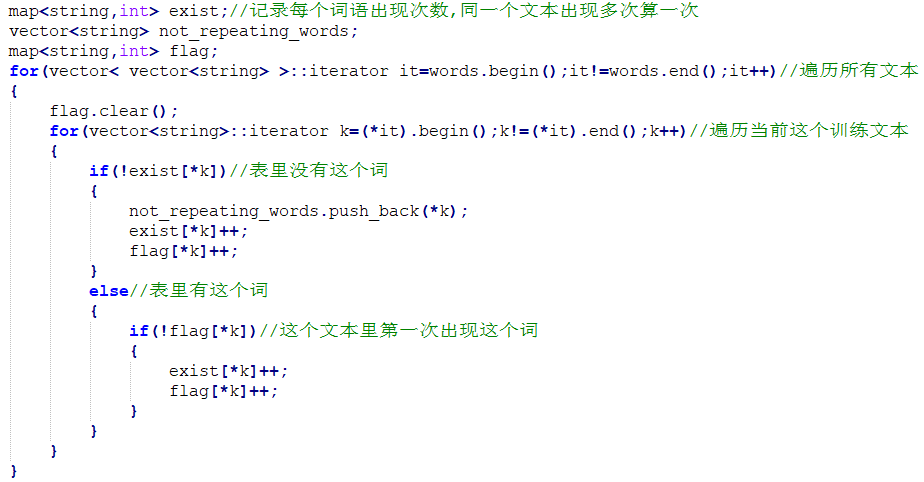


1. 关键代码截图（带注释）

得到训练文本



得到所有不重复词语并统计出现次数



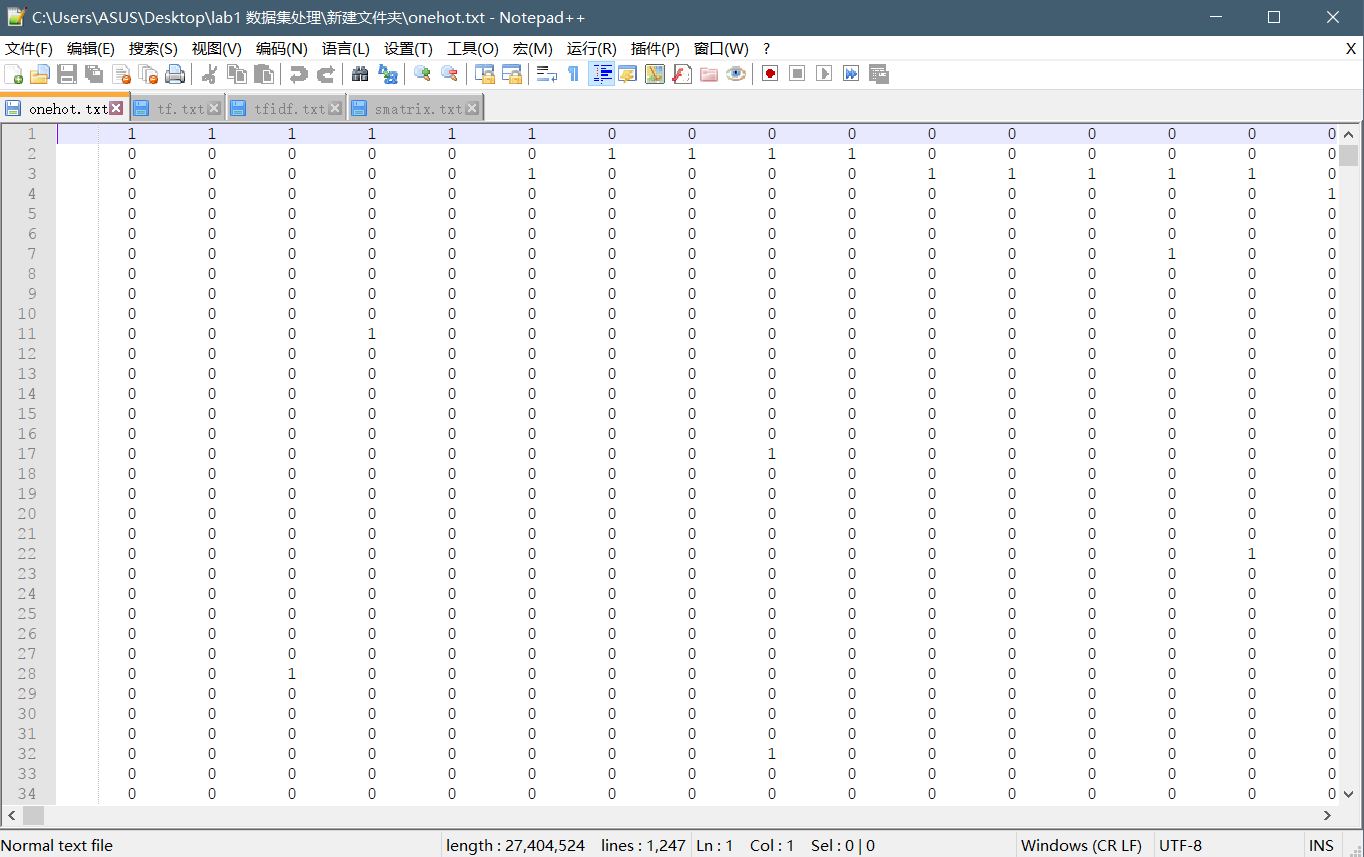
三元表加法



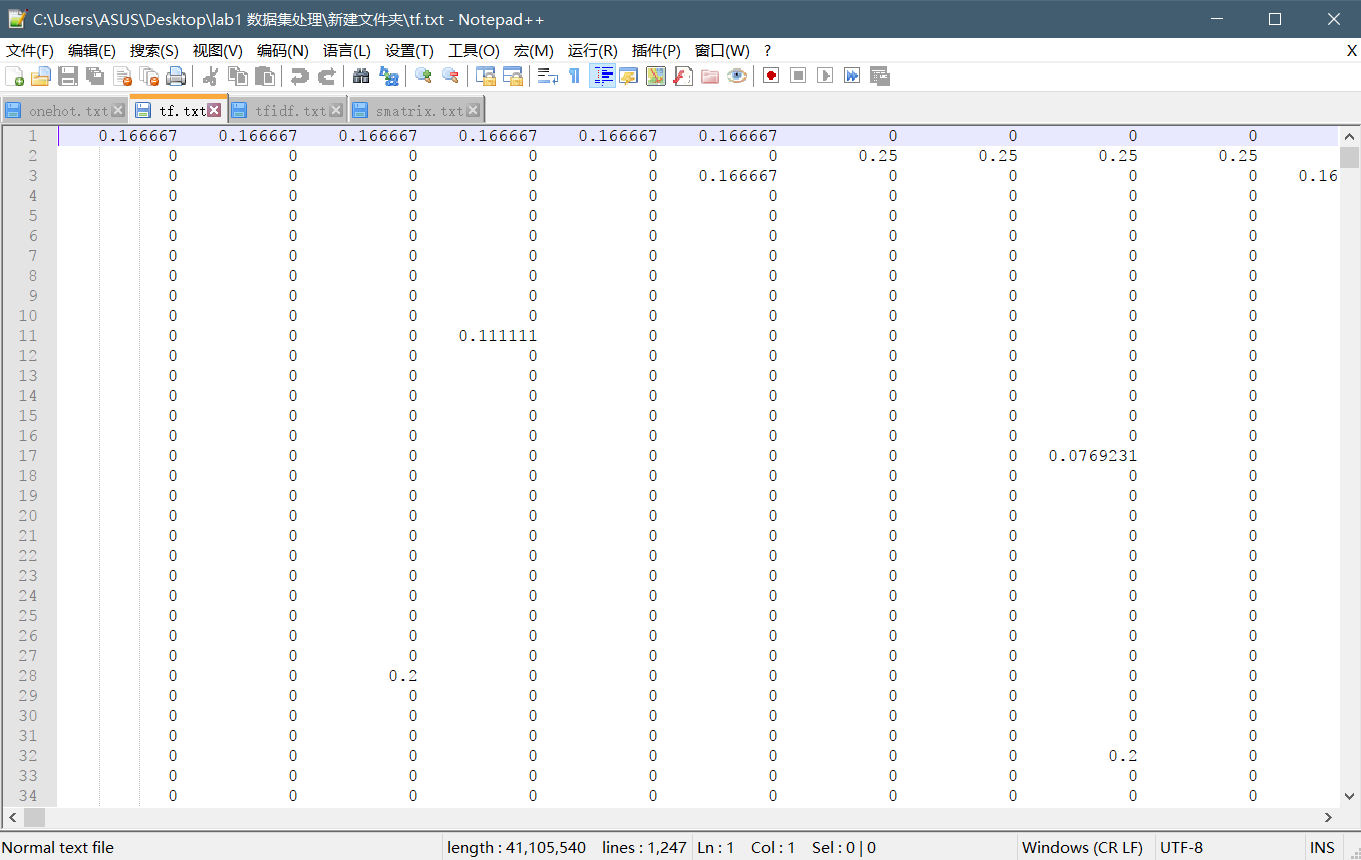
# 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

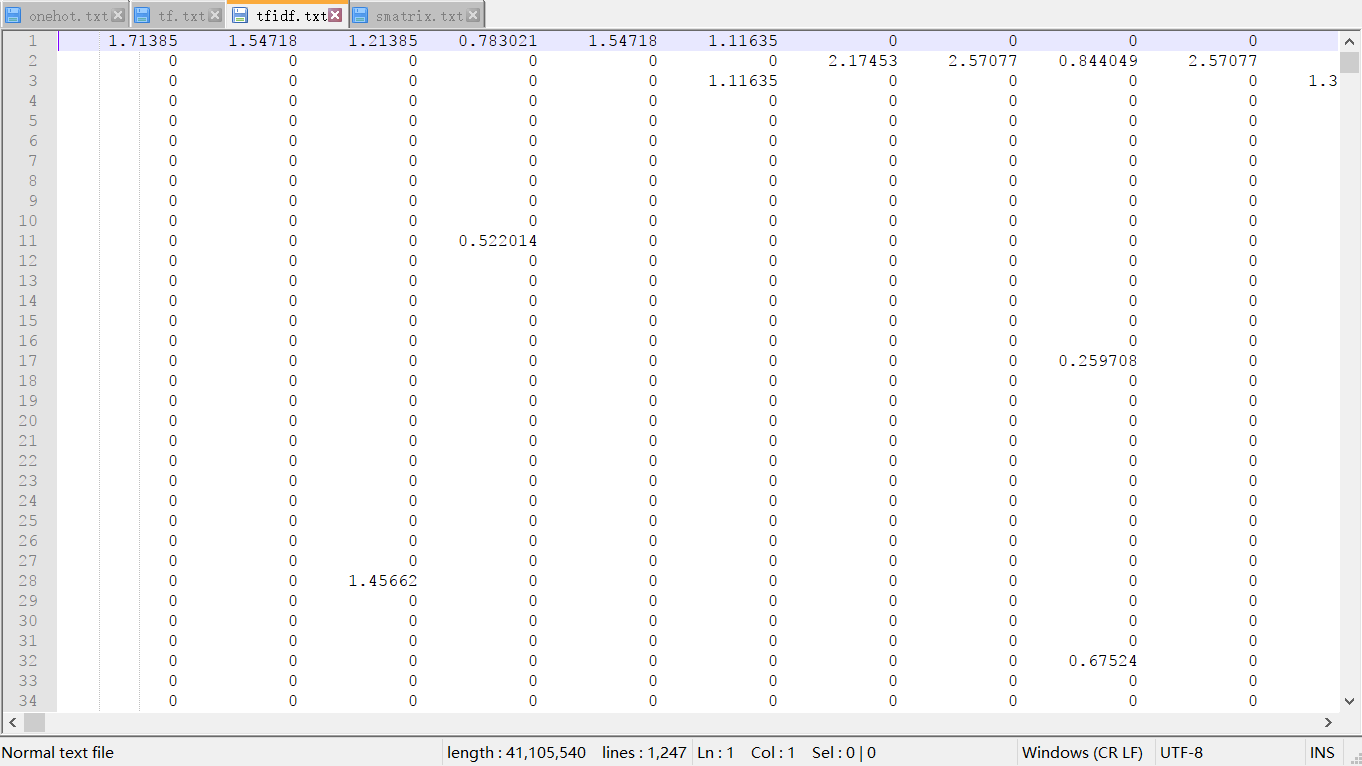
Onehot矩阵：



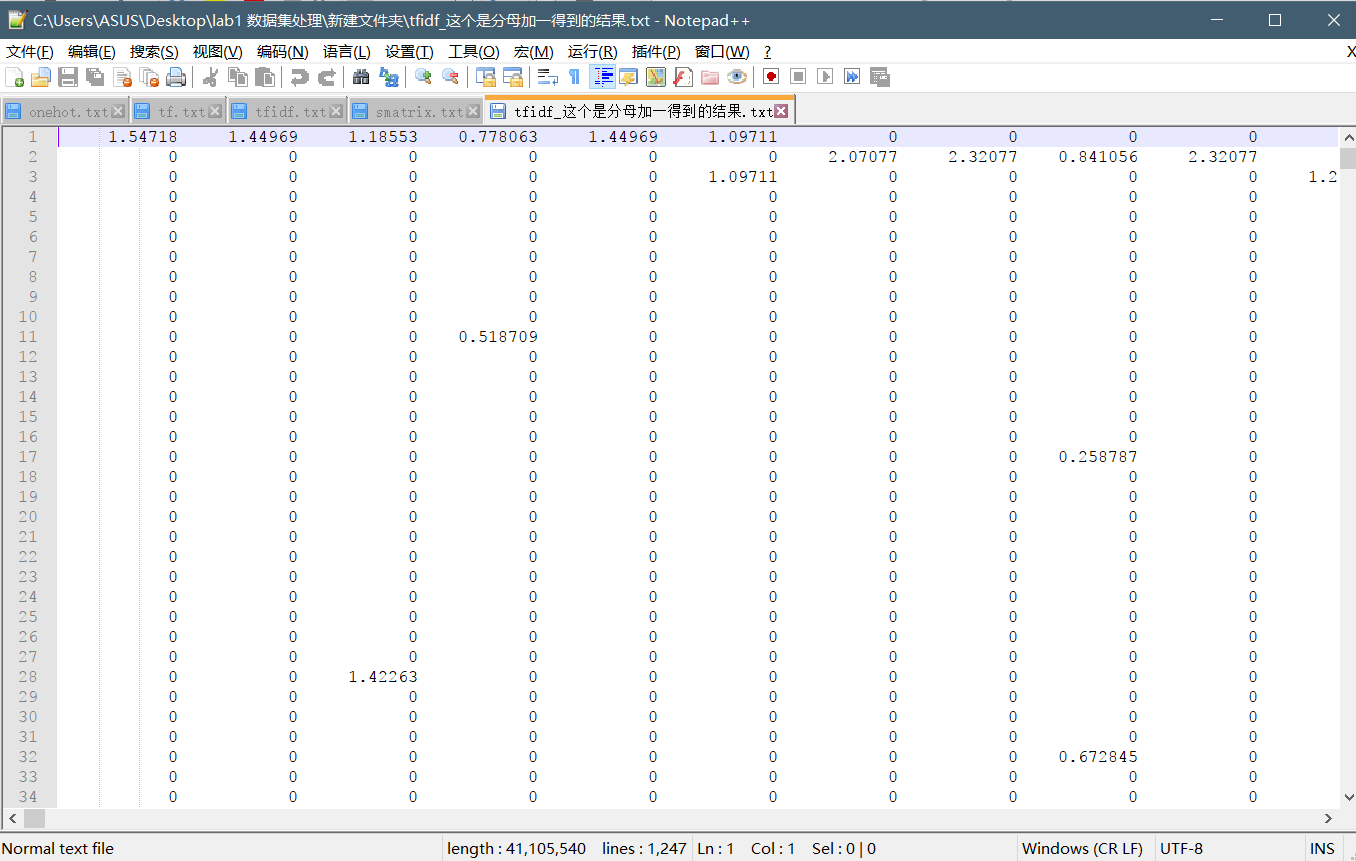
Tf矩阵：



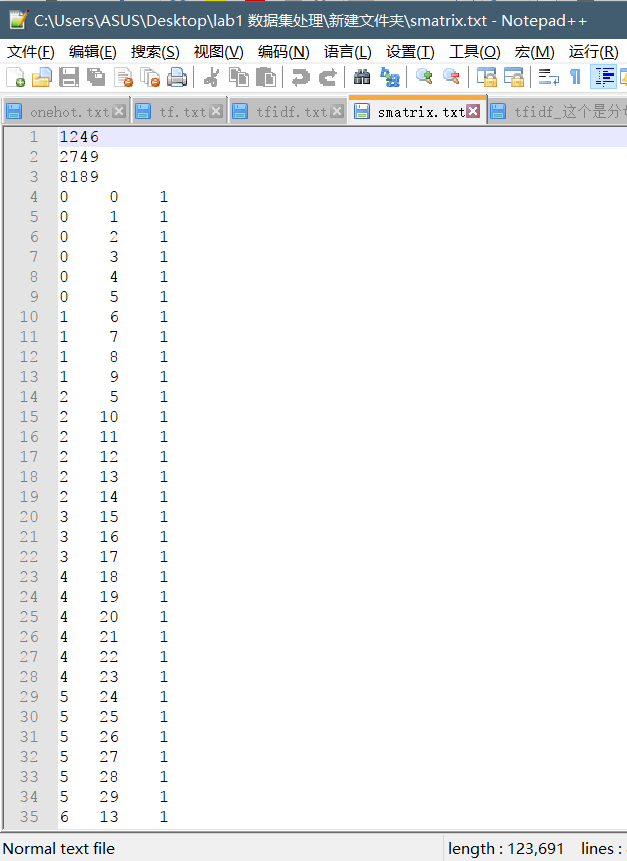
Tf-idf矩阵（分母不加1）：



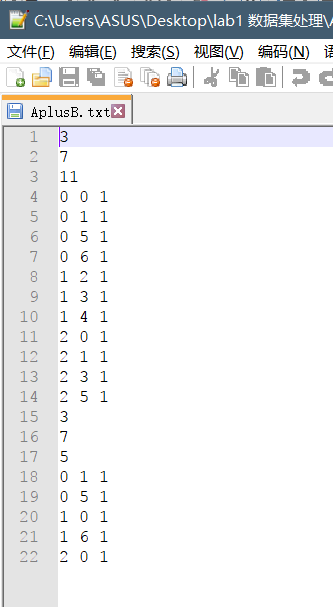
Tf-idf矩阵（分母加1得到的结果）：



Smatrix：



接下来展示稀疏矩阵加法。



进行相加的两个三元表



结果展示，符合预期目标

**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）**

这次实验只是简单的处理数据，因此只与身边的同学对比了一下结果数据的前几项，均相同。

# 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：为了避免分母为0（即所有文档都不包含该词）。

**2. IDF数值有什么含义？TF-IDF数值有什么含义？**

答：IDF数值代表了一种权重，与该词语出现在这D篇文本中的篇数文件频率成反比，即在不同文本中出现次数越多，IDF越低。

TF是指某一给定的词语在该文本中出现的频率。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：稀疏矩阵中零元素非常多，对于阶数较高的大型稀疏矩阵，按照常规分配方法来存储的话会浪费相当多的内存，因此可以仅存放非零元素，即利用三元表存下行、列、值，来达到节约存储空间的目的。