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

（2016 学年秋季学期）

课程名称：Artificial Intelligence

教学班级	14M2	专业（方向）	移动互联网
学号	14353205	姓名	刘万里

一、 实验题目

文本数据集的读写和简单处理

1、将数据集的数据表示成 one-hot 矩阵，TF 矩阵，TF-IDF 矩阵，并分别保存为“onehot”，“TF”，“TFIDF”三个文件 。

2、将数据集的 one-hot 矩阵表示成三元组矩阵，保存为“smatrix”文件。

二、 实验内容

1. 算法原理

因为需要去重且不排序，C++中并没有这样的数据结构来存储，因此按照数据集中的字符串出现的顺序，用一个 map<string,int>去存储，int 值存储这个字符串第一次出现的下标，再维护一个 map<int,int>的 RepeatTimes,first 为出现的下标，second 为在数据集中出现的次数，再维护一个 map<int,int>的 AppearLines，first 存储下标,second 存储的是对应下标的字符串出现过的行数，读入文件过程中维护整型变量数据的行数 Lines,不同的字符串的个数 Index,这样就可以得到一个二维矩阵。并且每次都用一个结构体 Three_Var 存储一个字符串对应的行数 i，列数 j，以及出现过的行数 v，并放入 Three_var 类型的 vector 容器中。

OneHot: 遍历的是一个矩阵，如果行数和列数和当前 vector 中的 Three_Var 中的 i,j 分别相等，则写入 1，否则写入 0；

TF:对 OneHot 进行归一化即可。因此需要在读取文件的过程中用一个名为 NumsOfLines 的 vector 记录每一行的单词的个数。

TF-IDF:按照公式计算即可，因为我们在读取文件时维护了每个下标对应的字符串出现过的行数 AppearLines，所以只需计算即可。

Smatrix1:通过 Lines 和 Index 遍历矩阵，如果行数和列数和当前 vector 中的 Three_Var 中的 i,j 分别相等，则得到一个三元组，写入 smatrix1.

遍历完每一行进行换行即可得到对应的矩阵。

2. 伪代码

```
/*----------读入数据----------*/

While（这一行有数据）
{
    如果是一句话的字符串（不是表情，数字之类的）
```

```
            存储下来，并依次在 FirstIndex 保存下标
            维护好字符串的下标索引 Index
        维护好 RepeatTimes，记录每个字符串重复的次数
        记录好这行的字符串数据的个数（即维护好 temp）
    }
    For(遍历 RepeatTimes)
    {
        维护 AppearLines;
        得到三元组 Three_var，并放入 vector 中；
    }
    /*-------------读取完一行----------------------------*/

    行数 Lines++;
    维护 NumsOfLines 来记录每一行出现的字符串个数；
    /*--------------------------读取完成----------------------------*/

    /*-----------------------处理数据并得到结果输出到文档-------------------*/
    pos = 0;//三元组 vector 的开始下标
    For(i = 0；遍历 Lines)
    {
        For(j=0;遍历出现过的字符串 Index)
        {
            If(当前行数 i ，下标分别与当前 vector 三元组的行数，下标相等)//匹配
            {
                得到结果并输出到对应文档；
            }
            Else   //不匹配
            {
                三个文档写入 0；
            }
        }
        /*-------------遍历了一行--------------*/
        3 个文档读入换行符；
    }
```

3. 关键代码截图（带注释）

读取文件：

读取一行数据并进行相应的数据结构的维护

```
while(ss>>dict)  //读入每行的数据集内容的词汇
{
    it1=FirstIndex.find(dict); //first元素的map查找函数
    if(it1==FirstIndex.end())//迭代器到了end()，说明这个词还没出现过
    {
        FirstIndex.insert(pair<string,int>(dict,index));
        index++;  //维护map中的下标
    }
    it2=RepeatTimes.find(FirstIndex[dict]);//计算该词汇在数据集中重复出现的次数
    if(it2==RepeatTimes.end())//这个下标第一次出现的话，放进这个repaeatTimes<int,int>中
        RepeatTimes.insert(pair<int,int>(FirstIndex[dict],1));
        //第一个整数记录它在map中的下标，第二个整数记录它重复的次数
    else
        it2->second++;//否则，维护好这个RepeatTimes即可。
    temp++;//这行的词汇个数（重复不算1个）
}
//读取完这行词汇了
```

读取完一行之后维护好 AppearLines 这个 map，并且维护变量 Lines,vector 容器 NumsOfLines

```
for(it2=RepeatTimes.begin();it2!=RepeatTimes.end();it2++)
{
    it3=AppearLines.find(it2->first);//每个词汇对应的下标重复的次数
    //计算在所有数据集中出现的次数
    if(it3==AppearLines.end()) //如果是空的，说明第一次出现
        AppearLines.insert(pair<int,int>(it2->first,1));
    else
        it3->second++; //it3的second存储的是对应下标的字符串出现的行数
    Three_Var item{Lines,it2->first,it2->second};
    //利用机构体记录这个单词出现的行数，下标，重复次数
    smat.push_back(item);
    ThreeVarNums++;//结构体vector中放入的个数
}
Lines++;  //第几行计数
NumsOfLines.push_back(temp);//记录好每行的词汇个数
```

现在进行对数据的处理和计算，输出：
遍历整个矩阵，先遍历每一行：

```
for(int i=0;i<Lines;i++) //行数
```

进行每一行的处理：

进行当前遍历到的这行的这个字符串，查看与三元变量 Three_var 的行和列是否对应，对应说明存在，不对应说明不存在这个字符串，OneHot 即为 0，其余同理。三元组输出时输出的不是出现过的行数，是 1 即可。

```
/*- - - - - - - - - 每一行的处理- - - - - - - - - */
for(int j=0;j<index;j++) //字符串下标
{
    if(smat[pos].i==i&&smat[pos].j==j)  //匹配到一个字符串出现
    {
        OneHot<<1<<" ";       //OneHot矩阵为1
        double rate=(double)smat[pos].v/NumsOfLines[i];  //归一化频率
        TF<<rate<<" ";
        TF_IDF<<rate*log((double)Lines/(1+AppearLines[j]))<<" "; //TF-IDF矩阵
        SMATRIX1<<i<<" "<<j<<" "<<1<<endl;  //输出到三元组
        if(smat[pos].i>=Lines/2)
            smat[pos].i-=Lines/2;
        pos++;//找下一个字符串了
    }
    else
    {
        OneHot<<0<<" ";       //不出现，这3个矩阵都是0，三元组则例外
        TF<<0<<" ";
        TF_IDF<<0<<" ";
    }
    /*-----  - - - - - - - - --处理完一行了-----  - ---*/
}
```

处理完一行就换行：

```
    OneHot<<endl;       //每一句新闻后都换行咯
    TF<<endl;
    TF_IDF<<endl;
```

关闭文件流，完成文件写入操作。

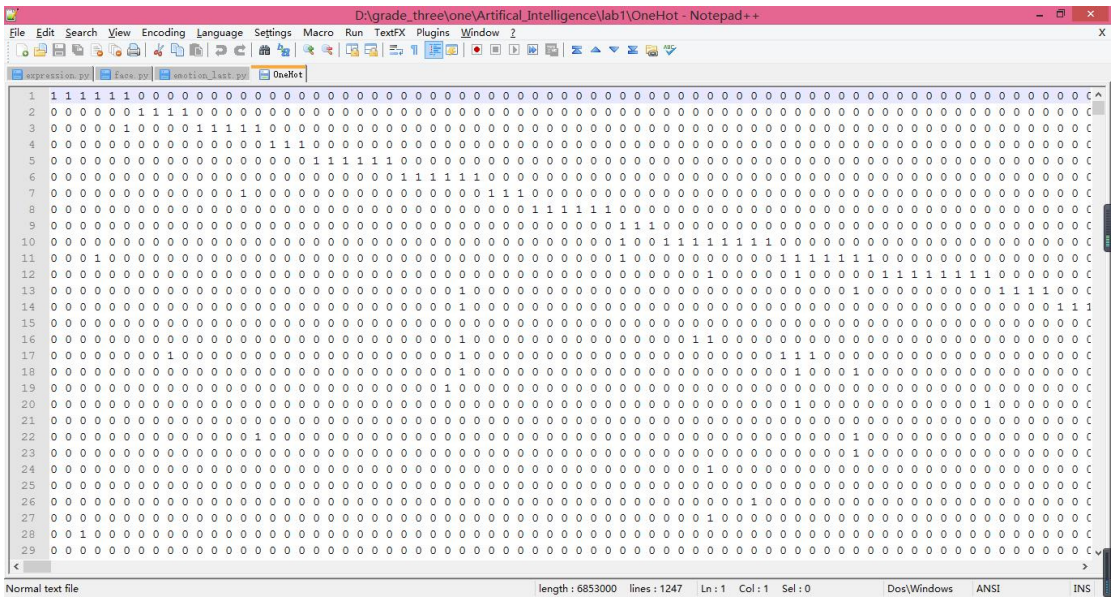
4. 创新点&优化

无

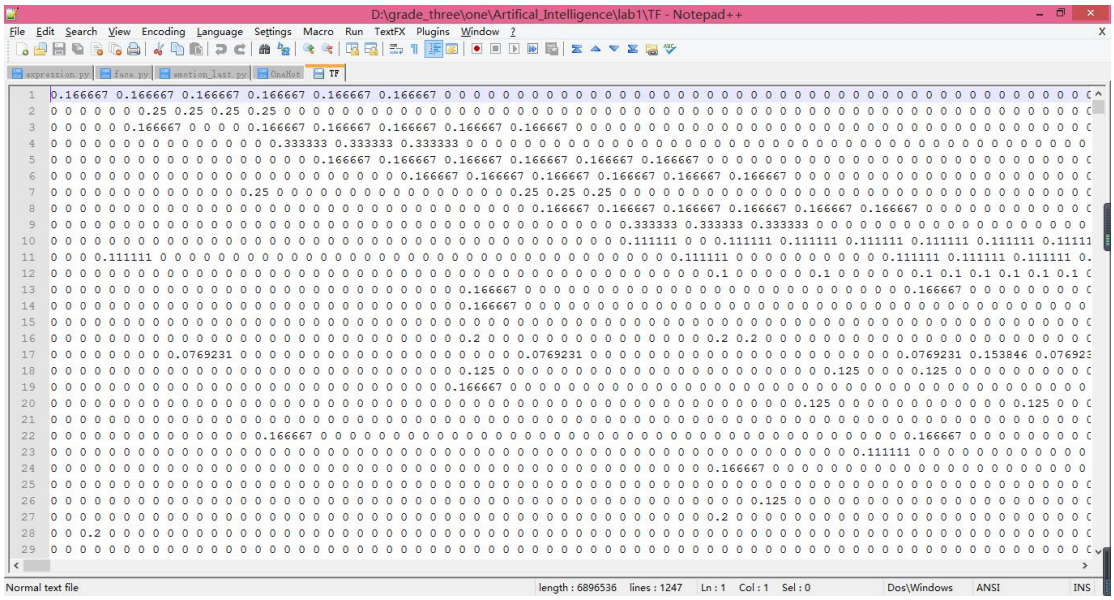
三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

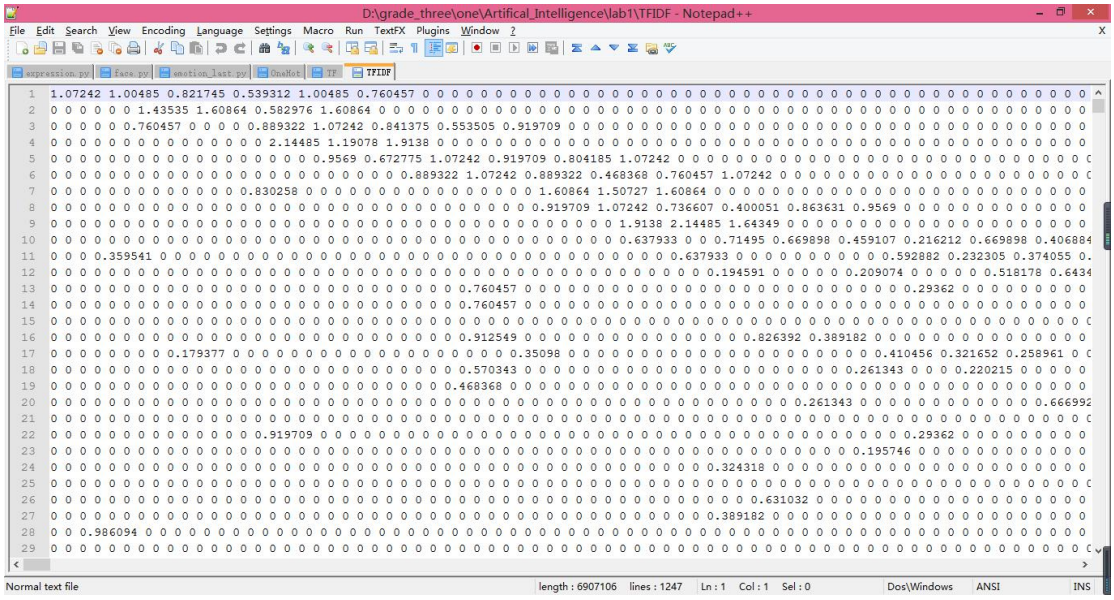
OneHot:



TF:



TF-IDF:



Smatrix1:

```
1246
2749
8189
0 0 1
0 1 1
0 2 1
0 3 1
0 4 1
0 5 1
1 6 1
1 7 1
1 8 1
1 9 1
2 5 1
2 10 1
2 11 1
2 12 1
2 13 1
2 14 1
3 15 1
3 16 1
3 17 1
4 18 1
4 19 1
4 20 1
4 21 1
4 22 1
4 23 1
5 24 1
5 25 1
```