



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	m2	专业(方向)	软件工程(移动工程信息)
学号	15352446	姓名	钟展辉

一、实验题目

文本数据集的简单处理

二、实验内容

1. 算法原理

①用 `getline(fin,s)` 读取整行文本, 循环读取至文本读取结束。

②每行文本, 以 `find()` 函数找出第二个 `'/t'` 位置, 再用 `substr()` 函数截取出需要分析处理的后半截字符串。

③在 `getword(string s, int row)` 函数中对第 `row` 行的字符串进行处理, 以 `stringstream` 分割字符串, 将每个单词存储在二维数组中 `word_matrix` 中。

④在 `one_hot(int row)` 函数中遍历二维数组 `word_matrix`, 新单词 `push` 进字符串类型向量 `word` 中, 每遍历到其中一个单词都判断其是否已经出现过在 `word` 向量中, 有则将二维矩阵 `matrix[row][index]` 相应位置设为 1. (`matrix` 二维数组初始化为 0)。

⑤遍历二维矩阵 `matrix`, 输出到文本 `one_hot.txt` 中。

`tf`、`idf` 思路大体相似, 而稀疏矩阵则是遍历 `matrix` 矩阵, 统计行数列数和总数值个数。实际上很简单, 不详细讲。

而稀疏矩阵三元向量表, 我另起了一个代码文件写在里面。

主要方法是将矩阵中的每一个数值点都设置为一个 `Node` 节点, 按位置顺序从小到大连接起来构成链表, 得到两条链表后, 新建一个空链表, 在 `while` 循环中, 对比两条链表当前节点大小, 较小的连接到新建链表中, 并把其当前节点设为下一个节点; 若两节点位置相同则取其中一个节点, 修改数值+1, 然后两个链表的当前节点都设置为其下一个节点。如此操作直至其中一个链表的当前节点为 `NULL`, 最后将剩下的链表连接到新建链表中, 则此新建链表为相加后所得链表, 再按照稀疏矩阵三元表的形式输出。

2. 伪代码

①

```
string word_matrix[2000][20];  
vector<string> word;
```



```
int matrix[2000][5000];
void getword(string s,int row)//row 指总的字符串行数
{
    分割字符串 s，将单词存储进 word_matrix 中。
}
void one_hot(int row)
{
    遍历 word_matrix 二维数组，填充 word 向量，并设置好最终矩阵 matrix[][];
}
void TF(int row)
{
    基本与 one_hot 类似，但当发现某行字符串有重复单词时 matrix 矩阵相应位置++;
    并且计算每行字符串的单词数量 sum，相应行的 matrix[row][]除以 sum;
}
void IDF(int row)
{
    TF(row);
    统计 matrix 矩阵每一列的非零数值数目 sum，按照公式得到  $\log[(row/sum)+1]$ ，将其与
    matrix 矩阵相应列相乘，即得到 tfidf 矩阵。
}
int main()
{
    文件流....
    int cnt=0;
    while(getline(fin,s))
    {
        getword(s,cnt);
        cnt++;
    }
    one_hot(cnt);
    文件输出流...
    TF(cnt);
    文件输出流...
    IDF(cnt);
    文件输出流...
    triple_table(略)//稀疏矩阵三元表
}
```



```
struct Node
{
    int row,col,value;
};

int cmp(Node* n1,Node n2)
{比较两节点的大小，即先后顺序}

Node * createlist(int index)
{
    根据 index，读取不同文本，建立基于文本内容的链表，返回其表头。
}

int main()
{
    如原理所说的操作。
}
```

3. 关键代码截图（带注释）

读取文件：

```
103  memset(matrix,0,sizeof(matrix));
104  ifstream fin;
105  fin.open("semeval.txt");//流变量连接文件，在同一个目录则不用指定路径只需要文件名。
106  string s;
107  int row=0;
108  while(fin)
109  {  getline(fin,s) ;
110     fin.get();
111     int secondTab=s.rfind('\t');//rfind逆向寻找第一个出现该字符的位置
112     string ss=s.substr(secondTab+1);//substr(a,b)截取从第a位置开始长度为b的字符串
113     getword(ss,row);
114     row++;
115  }
116  fin.close();
```

字符串分割：

```
21  void getword(string s,int row)
22  {
23      int cnt=0;
24      stringstream ss;
25      ss.clear();
26      ss.str(s);
27      while(ss)
28      {
29          ss>>word_matrix[row][cnt++];
30      }
31      num[row]=cnt;
32  }
```



one_hot:

```
33 void one_hot(int row)
34 {
35     for(int i=0;i<row;i++)
36     {
37         for(int j=0;j<num[i];j++)
38         {
39             vector<string>::iterator index=find(word.begin(),word.end(),word_matrix[i][j]);
40             int pos=distance(word.begin(),index);
41             //用find函数判断word中是否已有该单词，配合distance函数能找出其位置；
42             if(index!=word.end())//出现过
43             {
44                 matrix[i][pos]=1;
45             }
46             else//是新单词
47             {
48                 word.push_back(word_matrix[i][j]);
49                 num_of_word++;//总不重复单词数+1
50                 matrix[i][num_of_word-1]=1;
51             }
52         }
53     }
54 }
```

矩阵相加:

```
59 Node* p=headA->next;//文本1链表
60 Node* q=headB->next;//文本2链表
61 Node* head=new Node;//新建链表头结点
62 Node* tem=head;
63 while(p!=NULL&&q!=NULL)
64 {
65     if(cmp(p,q)==-1)//节点p更小
66     {
67         tem->next=p;
68         tem=tem->next;
69         p=p->next;
70     }
71     else if(cmp(p,q)==1)//节点q更小
72     {
73         tem->next=q;
74         tem=tem->next;
75         q=q->next;
76     }
77     else//节点位置相同
78     {
79         p->value=p->value+q->value;
80         tem->next=p;
81         tem=tem->next;
82         p=p->next;
83         q=q->next;
84         num--;
85     }
86 }
87 if(p!=NULL) tem->next=p;
88 else tem->next=q;
```



4. 创新点&优化（如果有）

将稀疏矩阵三元表转化为链表形式相加，最后再转化回稀疏矩阵三元表。

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）

模拟 ppt 上的样例

```
file.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1      asdasd  apple phone good sell
2      fdfdf  citizen buy phone phone
3      fsfsff  citizen feel apple phone expensive good
```

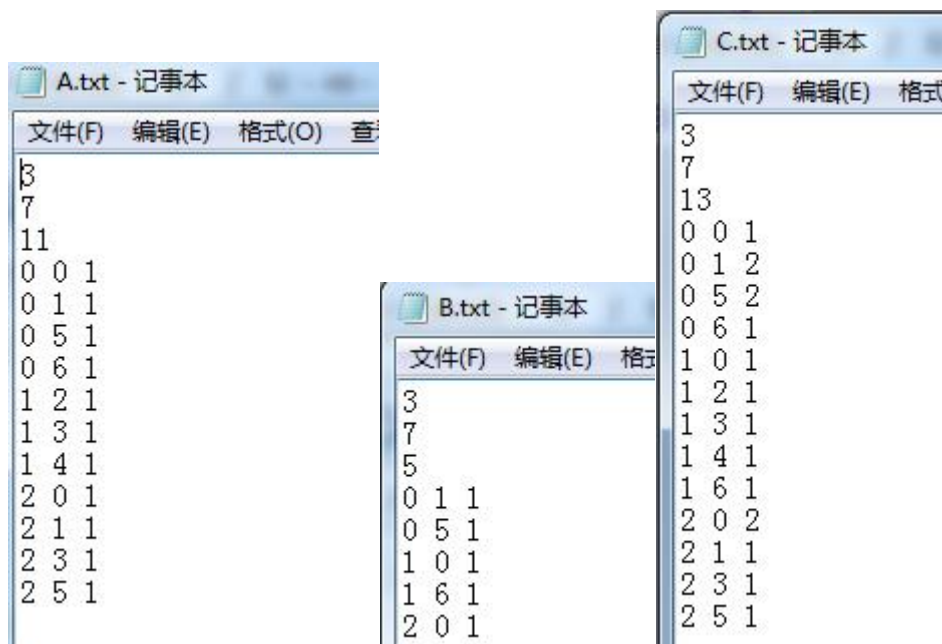
```
one_hot.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
apple phone good sell  citizen buy feel expensive
1 1 1 1 1 0 0 0 0
0 1 0 0 1 1 1 0 0
1 1 1 0 1 1 0 1 1
```

```
tf.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
apple phone good sell citizen buy feel expensive
0.25 0.25 0.25 0.25 0 0 0 0
0 0.5 0 0 0.25 0.25 0 0
0.166667 0.166667 0.166667 0 0.166667 0 0.166667 0.166667
```

```
tf_idf.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
apple phone good sell citizen buy feel expensive
0 -0.0719205 0 0.101366 0 0 0 0
0 -0.143841 0 0 0 0.101366 0 0
0 -0.047947 0 0 0 0 0.0675775 0.0675775
```

```
triple_table.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮
13
8
0 0 1
0 1 1
0 2 1
0 3 1
1 1 1
1 4 1
1 5 1
2 0 1
2 1 1
2 2 1
2 4 1
2 6 1
2 7 1
```

矩阵相加: $A+B=C$



```
A.txt - 记事本
文件(F) 编辑(E) 格式(O) 查
3
7
11
0 0 1
0 1 1
0 5 1
0 6 1
1 2 1
1 3 1
1 4 1
2 0 1
2 1 1
2 3 1
2 5 1

B.txt - 记事本
文件(F) 编辑(E) 格式(O) 查
3
7
5
0 1 1
0 5 1
1 0 1
1 6 1
2 0 1

C.txt - 记事本
文件(F) 编辑(E) 格式(O) 查
3
7
13
0 0 1
0 1 2
0 5 2
0 6 1
1 0 1
1 2 1
1 3 1
1 4 1
1 6 1
2 0 2
2 1 1
2 3 1
2 5 1
```

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）

上述截图结果与 ppt 上样例结果一致。

四、 思考题

1. IDF 的第二个计算公式中分母多了个 1 是为什么？

为防止该词语在语料库中不存在，即分母为 0

2. IDF 数值有什么含义？TF-IDF 数值有什么含义？

当某个词在语料库中各个文档出现的次数越多，它的 IDF 值越低，当它在所有文档中都出现时，其 IDF 计算结果为 0，而通常这些出现次数非常多的词或字为“的”、“我”、“吗”等，它对文章的权重计算起不到一定的作用。 $tfidf_{i,j}$ 表示词频 $tf_{i,j}$ 和倒文本词频 idf_i 的乘积，TF-IDF 值越大表示该特征词对这个文本的重要性越大。

3. 为什么要用三元顺序表表达稀疏矩阵？

稀疏矩阵，顾名思义就是偌大的一个矩阵中只有寥寥可数的几个非零数值，如果用二维数组来表示这个矩阵的过于浪费系统资源，也会增大算法复杂度，换为三元顺序表又高效又节省空间又直观。

|----- 如有优化，重复 1, 2 步的展示，分析优化后结果 -----|



PS: 可以自己设计报告模板，但是内容必须包括上述的几个部分，不需要写实验感想