文件的读写

1. ofstream , ifstream, fstream

(相关链接 http://blog.csdn.net/kingstar158/article/details/6859379/)

文本数据集的表达形式

文本数据是指，每一个文本元素（一般称为 document 或者 text）包括几句话，以及标签。

2. one-hot: 在用以表示文本向量时，向量每一个值（1 或者 0）标志着有无这个元素。

3. Term-Frequently: 在用以表示文本向量时，向量的每一个值标志对应的词语出现的次数。

4. TF-IDF

计算过程

1) TF：Term Frequency 词频，即一个词在一个文件(例如一篇文章)中出现的次数归一化后的频率，计算式为 $\mathrm{tf}_{\mathrm{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$ ，$n_{i,j}$ 表示在文件 d_j 中词语 i 出现的次数，分母则是 d_j 所有词语的个数。之所以要归一化，是为了防止过长的文档词语太多而计算时会偏向过长文档。

2) IDF：inverse document frequency 逆向文件频率，计算式为：

$$\mathrm{idf}_{\mathrm{i}} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

，|D|表示文件总数，分母是包含词语 i 的文件总数。为了防止分母为 0，通常会选择分母加 1，即 $1 + |\{j : t_i \in d_j\}|$ 。

由 IDF 的计算方法可知，包含该词语的文件越少，计算的值越大，则词语 i 对描述一个文件的越重要，越可以当作特征。

3) TD/IDF 的计算式：$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$ ，

稀疏矩阵

稀疏矩阵指的是，矩阵中大部分元素的值是 0，只有少部分是非零元素。对于短文本，甚至于长文本数据，经过处理之后得到的表示矩阵，很大概率时一个稀疏的矩阵。当数据量非常大的时候，如果要存储整个稀疏矩阵，那么需要耗费巨大的内存，而且影响运算速度。因此，我们可以将巨大的稀疏矩阵表达成一个稀疏矩阵。在这里要求大家掌握稀疏矩阵的三元组顺序表。

	6	md	
	7	nd	
	9	td	
0	0	0	2
1	0	4	6
2	0	7	7
3	1	2	1
4	2	2	2
5	2	6	3
6	3	5	8
7	4	3	5
8	5	1	9
	i	j	v

稀疏矩阵 M 的三元组顺序表

图中 md，nd 指的是原矩阵的行列数，td 指的是原矩阵非零元素的个数。i，j 指的是该元素所在的行列，v 指的是该元素的值。

（图一）

实验任务

a) 数据读写

b) 将数据集的数据表示成 one-hot 矩阵，TF 矩阵，TFIDF 矩阵，并分别保存为”onehot”，”TF”，”TFIDF”三个文件

c) 将数据集的 one-hot 矩阵表示成三元组矩阵并储存成图一形式（文件第一行是行数，第二行是列数，第三行是非零数据个数，第四行下去是每个元素的信息）。保存为“smatrix”文件

d) （选作，加 1 分）实现稀疏矩阵加法运算。用的数据集是 semEval 文件，一共 1246 条数据，那么将原矩阵分成两个矩阵，A 矩阵包含前 623 条数据，B 矩阵包含后 623 条数据，用 A 和 B 两个矩阵完成加法运算任务，完成运算后用稀疏矩阵保存为”AplusB”文件。

注意事项

1. 共 5 个实验结果文件，和代码文件打包，命名为“学号_拼音名字”交到 FTP。
2. 编程语言可用 c++, python, matlab, java，不能使用现成库。