

# 计算机网络·实验报告 2.1

学号: 1511504      姓名: 胥倩雯

专业: 软件工程      实验时间: 2017.10.23

## 一. 实验项目

编程实现学术论文检索系统

## 二. 实验环境

Java + eclipse + string tool suite

## 三. 实验要求

1. 编程实现学术论文检索系统
2. 计分排序模型采用向量空间模型
3. 权重计算机制采用 lnc. ltn

## 四. 实验分析

我们知道 对于查询和文档常常采用不同的权重计算机制, 根据要求我们采用 lnc. ltn。即对于文档, 要求对数 tf, 无 idf 因子, 余弦长度归一化。词项频率为  $1 + \log(\text{tf}_{td})$ , 文档频率为 1, 归一化  $1 / \sqrt{w_1^2 + w_2^2 + w_3^2 + \dots + w_m^2}$ 。而对于查询, 需要对数 tf, idf, 无归一化, 即词项频率为  $1 + \log(\text{tf}_{td})$ , 文档频率为  $\log(N/\text{df}_t)$ , 无归一化即为 1。由此我们知道我们需要做的准备工作是计算出  $\text{tf}_{td}$ ,  $\text{df}_t$  和  $\sqrt{w_1^2 + w_2^2 + w_3^2 + \dots + w_m^2}$  (每个文档不同)。

所以, 我们的倒排索引表中要记录以上的信息。


每一行, 我们记录: 单词  $\rightarrow$  出现这个单词的文档数  $\rightarrow$  第一个文档 id 号  $\rightarrow$  第一个文档的  $1 + \log(\text{tf}_{td}) \rightarrow$  第二个文档 id 号  $\rightarrow$  第二个文档的  $1 + \log(\text{tf}_{td}) \rightarrow \dots$

我的代码中实现的结果显示的文档积分都较小, 最大的也就零点几, 我的理解是由于我的基数较小 (即 N 较小导致文档频率的值较小, N 只有 11), 现实情况中 N 可能是  $\text{df}_t$  的几万倍甚至几亿倍。

## 五. 设计思路及相关代码

**PdfboxTest. java 文件:**

1. 引入 pdfbox-app-1.6.0.jar 文件, 用于运用在 PdfboxTest. java 文件中使学术论文常见的 pdf 格式转换成 txt 格式。












 pdfbox-app-1.6.0.jar - J:\java\jar\_file

2. PdfboxTest. java 文件中的 getText(String file) 函数的参数是文件名, 这个函数用于将具体的文件由 pdf 格式转换为 txt 格式, 并且以原来 pdf 名称来命名新产生的 txt 文件。












3. 遍历 pdf 文件文件夹，将其中所有的 pdf 文件一个个转换成 txt 文件。

效果如下：

原文件：

 A Block QR Algorithm and the Singul...	2017/10/20 星期...	Chrome HTML D...	137 KB
 Advances in Articial Intelligence.pdf	2017/10/20 星期...	Chrome HTML D...	107 KB
 improvedapproximation03.pdf	2017/10/20 星期...	Chrome HTML D...	184 KB
 Lecture Notes in Artificial Intelligence...	2017/10/20 星期...	Chrome HTML D...	117 KB
 phym1.pdf	2017/10/20 星期...	Chrome HTML D...	114 KB
 Relations in GUHA style data mining....	2017/10/20 星期...	Chrome HTML D...	142 KB
 syllabus.pdf	2017/10/20 星期...	Chrome HTML D...	67 KB
 Text Categorization Using Automatic...	2017/10/20 星期...	Chrome HTML D...	230 KB
 tinyos.pdf	2017/10/20 星期...	Chrome HTML D...	186 KB
 United States Patent office.pdf	2017/10/20 星期...	Chrome HTML D...	216 KB
 VERY DEEP CONVOLUTIONAL.pdf	2017/10/20 星期...	Chrome HTML D...	196 KB

产生的 txt 文件：

 A Block QR Algorithm and the Singul...	2017/10/20 星期...	TXT 文件	16 KB
 Advances in Articial Intelligence.txt	2017/10/20 星期...	TXT 文件	3 KB
 improvedapproximation03.txt	2017/10/20 星期...	TXT 文件	32 KB
 Lecture Notes in Artificial Intelligence...	2017/10/20 星期...	TXT 文件	20 KB
 phym1.txt	2017/10/20 星期...	TXT 文件	50 KB
 Relations in GUHA style data mining.txt	2017/10/20 星期...	TXT 文件	15 KB
 syllabus.txt	2017/10/20 星期...	TXT 文件	7 KB
 Text Categorization Using Automatic...	2017/10/20 星期...	TXT 文件	28 KB
 tinyos.txt	2017/10/20 星期...	TXT 文件	90 KB
 United States Patent office.txt	2017/10/20 星期...	TXT 文件	8 KB
 VERY DEEP CONVOLUTIONAL.txt	2017/10/20 星期...	TXT 文件	55 KB

form. java 文件：

1. 定义一个 myMergeSort 类，用于归并排序调用。
2. 定义一个 node 类，用于记录每个单词所在文档 id，在这个文档中的出现频次，以及下一个文档相关记录的连接。

```
class node{
    int DOCID;
    double tf=1;
    node next=null;
    node(int i){
        DOCID = i;
    }
}
```

3. 接着的操作和布尔检索时构造倒排索引表类似：对每个 txt 文件进行分词，排序，合并成倒排索引表。其中 tf 的计算：

```

for(j=0;i+j<=k-1;){
    if(doc[i].word.equals(doc[i+j].word)&&doc[i].docID==doc[i+j].docID){
        j++;
    }else{
        break;
    }
}
tmp1.tf=j;

```

首先计算它的频次。最后存储在文件时记录的 tf:

```
n.tf=1+Math.log10(n.tf);
```

4. 由于之前定义了一个数组 sum 用于记录每个文档的  $\sqrt{w_1^2 + w_2^2 + w_3^2 + \dots + w_m^2}$ ，用于归一化。所以对每个 node，进行操作如下：

```
sum[n.DOCID]+=n.tf*n.tf;
```

最后，把每个 sum 求平方根：

```

for(int i=0;i<array.length;i++){
    sum[i]=Math.sqrt(sum[i]);
}

```

并保存在 sum.txt 里。

效果如下：

倒排索引表：

```

1319 categorize 1 7 1.6989700043360187
1320 categorized 1 7 1.6020599913279625
1321 categorizing 1 7 1.0
1322 category 3 4 1.3010299956639813 7 1.6020599913279625 10 1.3010299956639813
1323 cation 2 8 1.4771212547196624 9 1.0
1324 cational 1 5 1.0
1325 cations 2 8 1.0 9 1.0
1326 causal 1 8 1.0
1327 cause 1 8 1.0
1328 caused 1 8 1.0

```

11 个文档的 Sum:

```

32.693240952590685
15.854548756871
43.88295570921775
39.85352441079645
58.218031239776366
34.14649302038415
25.366054033376475
44.96107071021392
75.658906599065
26.65421248213814
56.22752250637197

```

[Search.java 文件:](#)

1. 首先定义一个函数 Quicksort() 用于快速排序

2. 接着，读取输入的一行，用

```
String srch[]=str.split("\\W+"); 分词
```

3. 统计查询的单词以及每个单词的单词数，用 search\_word 结构的数组 w[] 存储

```
class search_word{
    String word;
    int num=1;
}
```

4. 从 sum.txt 文件中读取每个文档的分母
5. 从 output.txt 文件中读取倒排索引表的信息，并且存储到 treemap 结构的 wordmap 中：

```
TreeMap<String,invertednode> wordmap=new TreeMap<String,invertednode>();
try{
    BufferedReader br = new BufferedReader(new FileReader("doc\\output.txt"));
```

6. 我们定义了两个数组，其中 queryweight[] 记录每个查询单词的 weight，docweight[i][j] 记录查询语句里面的单词 i，文档号为 j 的 weight，并在下面对他们进行运算。至于其它的值没有必要获取，因为如果这个单词并没有在查询中出现，那么它的 queryweight 为 0。

```
double queryweight[]=new double[s];
double docweight[][]=new double[s][filenum]; //s代表查询单词，列代表文档号
```

7. 这里我们得到了每个查询单词的 weight

```
queryweight[i]=Math.log10((float)filenum/(float)inv.num)*(float)(1+Math.log10(w[i].num));
```

此处  $1+\log(\text{tf}_{t,d})$  里的  $\text{tf}_{t,d}$  是查询的单词在查询语句中的词频，所以查找 search\_str 结构的数组 w[] 里的某一项的 num 属性，然后文档频率为  $\log(N/\text{df}_t)$ ，其中 N 即 filenum，而  $\text{df}_t$  则是倒排索引表这个单词对应那一行的第二个值，无归一化为 1，要求 queryweight 把它们相乘即可。

8. 接着，我们计算文档中的单词 weight，由于 output.txt 中记录的 tf 即为已经对数 tf 后的值，所以我们直接用就好，文档频率为 1，要归一化，所以要除 sum[i]。

```
docweight[i][n.DOCID]=(double)n.tf/sum[n.DOCID];
System.out.println("第"+i+"个单词的docweight在"+n.DOCID+"文档"+docweight[i][n.DOCID]);
```

9. 最后，为了查询对于这个查询语句 j 号文档的积分，只要将 queryweight[j] 与 docweight[j][i] (j++) 逐个相乘求和即可。

```
for(int j=0;j<s;j++){
    score[i]+=queryweight[j]*docweight[j][i];
    System.out.println("第"+i+"个文档积分加: "+q
}
```

网页版 (sts 版) 原理与 java 版相同，只是把 Search.java 的处理放在了 HomeController 中。并且在显示结果的时候，我用了

```

<c:forEach items="${ts}" var="arr">
<c:if test="${arr.getscore() != 0}">
    <tr><td><h3 style="color:#FFD1A4">${arr.gettitle()}</h3><a href="J:/doc/pdf/${arr.ge
</c:if>

```

<c:if></c:if>语句判断积分是否为 0，如果为零，就不显示这个文档。

详细代码请看代码文件夹中的 java 文件夹和 sts 文件夹。

## 六. 相关配置

### Java:

若电脑上装有 eclipse (Java - version1.7)，直接引入“1511504 胥倩雯 实验报告 2\java\tfidf”文件夹即可。

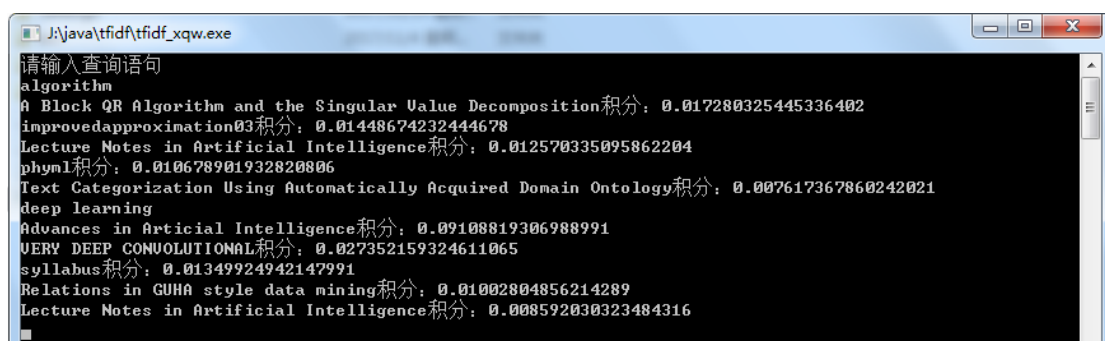
其中，tfidf\doc\pdf 文件夹里是原 pdf 文件，tfidf\doc\input 文件夹里是将 pdf 文件转换成 txt 之后的一系列文件，然后 output.txt 是对 input 文件夹里的文件进行分析后产生的倒排索引表，title.txt 是文件的名称，而 sum 是要用来归一化所需要的分母。

input	2017/11/4 星期...	文件夹	
pdf	2017/11/4 星期...	文件夹	
output.txt	2017/10/25 星期...	TXT 文件	225 KB
sum.txt	2017/10/25 星期...	TXT 文件	1 KB
title.txt	2017/10/25 星期...	TXT 文件	1 KB

而\tfidf\src 文件夹中存放的是 Java 源码，其中 PdfboxTest 用来将 pdf 转换为 txt，而 form 是用来构建倒排索引表，Search 是用来查询，由于我的查询之前的准备工作已经做好（已转换成 txt 并生成索引表），故可直接运行 Search 来进行查询。

form.java	2017/10/20 星期...	JAVA 文件	9 KB
PdfboxTest.java	2017/10/25 星期...	JAVA 文件	4 KB
Search.java	2017/11/4 星期...	JAVA 文件	7 KB

### 可执行文件 (exe)：



直接输入想要查询的语句（如：deep learning, algorithm, NP hard 等）

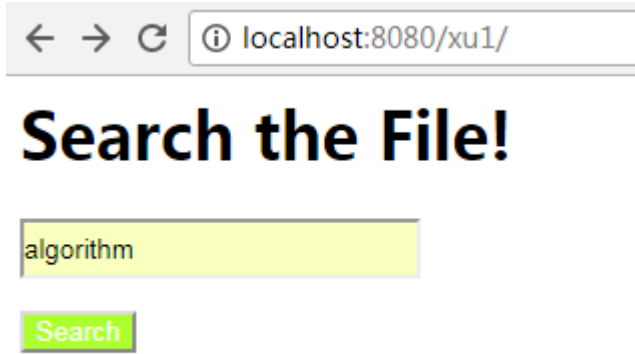
tfidf\_xqw 存放在“1511504 胥倩雯 实验报告 2\java\tfidf”文件夹中。

### String MVC 版

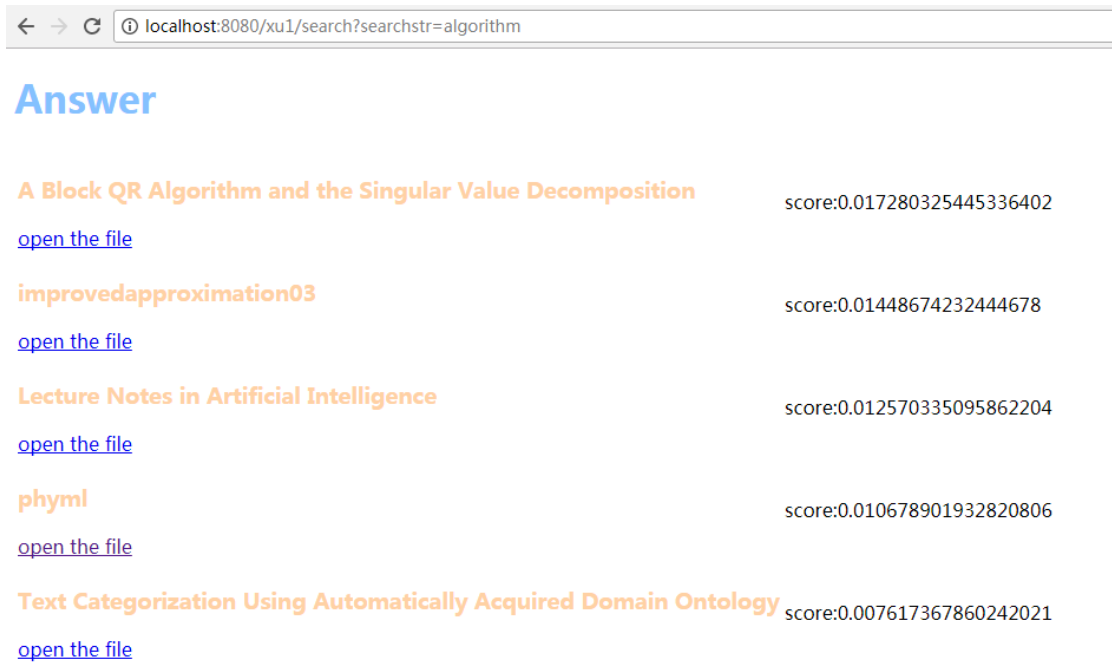
在“1511504 胥倩雯 实验报告 2\string mvc 版”文件夹中有相应的源码，把它引入 spring tool suite 中即可，由于我用的 sts 版本较低，是 3.6.0，可能与较高的版本不适配，可能要下 spring tool suite3.6.0。实验效果显示如下，点击运行后，可在 Chrome 中输入 localhost:8080/xu1/。

## 七. 实验效果截图

string tool suite 截图



搜索结果如下（显示所有积分不为 0 的结果）：



接着，我搜索了 improvedapproximation03.pdf 里的一句话

The vertex cover is a special case of the set-cover problem that requires to select a minimum number (or minimum cost) collection of subsets that cover the entire universe. The set-cover problem with hard capacities generalizes the

搜索：

# Search the File!

vertex cover is a special case

Search

显示结果为:

## Answer

improvedapproximation03

score:0.12491816332484217

[open the file](#)

Advances in Articial Intelligence

score:0.027710198540962315

[open the file](#)

tinyos

score:0.01884520270405075

[open the file](#)

phym1

score:0.015094073151481075

[open the file](#)

VERY DEEP CONVOLUTIONAL

score:0.013744717133171339

[open the file](#)

Java 截图:

algorithm

A Block QR Algorithm and the Singular Value Decomposition积分: 0.017280325445336402  
improvedapproximation03积分: 0.01448674232444678  
Lecture Notes in Artificial Intelligence积分: 0.012570335095862204  
phym1积分: 0.010678901932820806  
Text Categorization Using Automatically Acquired Domain Ontology积分: 0.007617367860242021

deep learning

Advances in Articial Intelligence积分: 0.09108819306988991  
VERY DEEP CONVOLUTIONAL积分: 0.027352159324611065  
syllabus积分: 0.01349924942147991  
Relations in GUHA style data mining积分: 0.01002804856214289  
Lecture Notes in Artificial Intelligence积分: 0.008592030323484316

此处, 我限制了只显示积分排序前五的文档名称

## 八. 总结

通过这次实验, 我知道了如何用空间向量模型根据查询语句对文档进行积分并且排序, 更知道了我们求积分可以采用不同的权重计算机制, 并且可能会得到不同的结果。我知道了该如何编程实现学术论文检索系统, 文档排序的机制是什么。我知道了在空间上比较文档和查询语言的相似度是根据向量的夹角而不是距离来计算。总之, 这次实验使我受益匪浅。