# 武漢大學 实验报告

专业： 网络空间安全
姓名： 胡彦
学号： 2021302021097
日期： 2024 年 5 月 24 日

课程名称： 信息检索　实验名称： 基于 Lucene 的信息检索

## 一、 实验内容

### 1.1 实验概述

从头开始编写一个自定义命令行搜索引擎用来索引 37600 篇 + 的 TDT3 新闻文章，并自己实现文档相关性评分功能。

### 1.2 实验环境

#### 1.2.1 软件环境

- java 21.0.1 2023-10-17 LTS
- Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)
- Java HotSpot(TM) 64-Bit Server VM (build 21.0.1+12-LTS-29, mixed mode, sharing)
- IntelliJ IDEA 2024.1.1 (Ultimate Edition)

#### 1.2.2 依赖库

- Lucene 9.10.0
- commons-io 2.16.1
- crimson 1.1.3
- hamcrest-core 1.3

### 1.3 参考资料

1. Lucene 9.10.0 core API
2. Java 教程 | 菜鸟教程
3. 【Java】String 的分割
4. ElasticSearch：相关性评分原理及应用
5. lucene 打分机制
6. TFIDF 改进版：BM25 算法介绍及 Lucene 的实现

# 二、 实验步骤

## 2.1 文件结构

```
.
├── Lucene.txt
└── lucene
    ├── out
    │   └── artifacts
    ├── pom.xml
    ├── src
    │   ├── main
    │   │   ├── java
    │   │   │   ├── Command
    │   │   │   │   └── Program.java
    │   │   │   ├── LuceneDependency
    │   │   │   │   ├── DocRanking.java
    │   │   │   │   ├── QueryParser.java
    │   │   │   │   └── SGMLParser.java
    │   │   │   └── LuceneMain
    │   │   │       ├── LuceneBuild.java
    │   │   │       └── LuceneQuery.java
    │   │   └── resources
    │   │       └── META-INF
    │   │           └── MANIFEST.MF
    │   └── test
    │       └── java
    ├── target
    │   ├── classes
    │   │   ├── Command
    │   │   │   └── Program.class
    │   │   ├── LuceneDependency
    │   │   │   ├── DocRanking.class
    │   │   │   ├── QueryParser.class
    │   │   │   └── SGMLParser.class
    │   │   ├── LuceneMain
    │   │   │   ├── LuceneBuild.class
    │   │   │   └── LuceneQuery.class
    │   │   └── META-INF
    │   │       └── MANIFEST.MF
    │   └── generated-sources
    │       └── annotations
    └── temp
        ├── 0.zip
        └── index
            ├── _0.cfe
            ├── _0.cfs
            ├── _0.si
            ├── _1.cfe
            ├── _1.cfs
            ├── _1.si
            ├── _2.cfe
            ├── _2.cfs
            ├── _2.si
            ├── _3.cfe
            ├── _3.cfs
            ├── _3.si
            ├── _4.cfe
            ├── _4.cfs
            ├── _4.si
            ├── _5.cfe
            ├── _5.cfs
            ├── _5.si
            ├── _6.cfe
            ├── _6.cfs
            ├── _6.si
            ├── segments_1
            └── write.lock
```

图 1

## 2.2 初步实现

这里是基于 ppt 给出的相关资料中，csdn 教程中的测试程序所编写：3.2.3 代码实现。

### 2.2.1 Lucene 原理

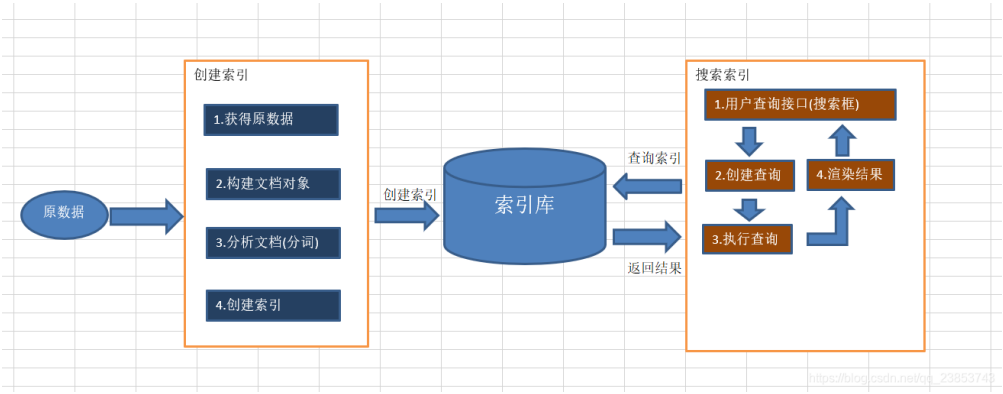


图 2

全文检索的流程分为两大部分：索引流程、搜索流程。索引流程：确定原始内容即要搜索的内容-> 采集原始内容数据-> 创建文档-> 分析文档 (分词)-> 创建索引搜索流程：用户通过搜索界面-> 创建查询-> 执行搜索-> 从索引库搜索-> 渲染搜索结果

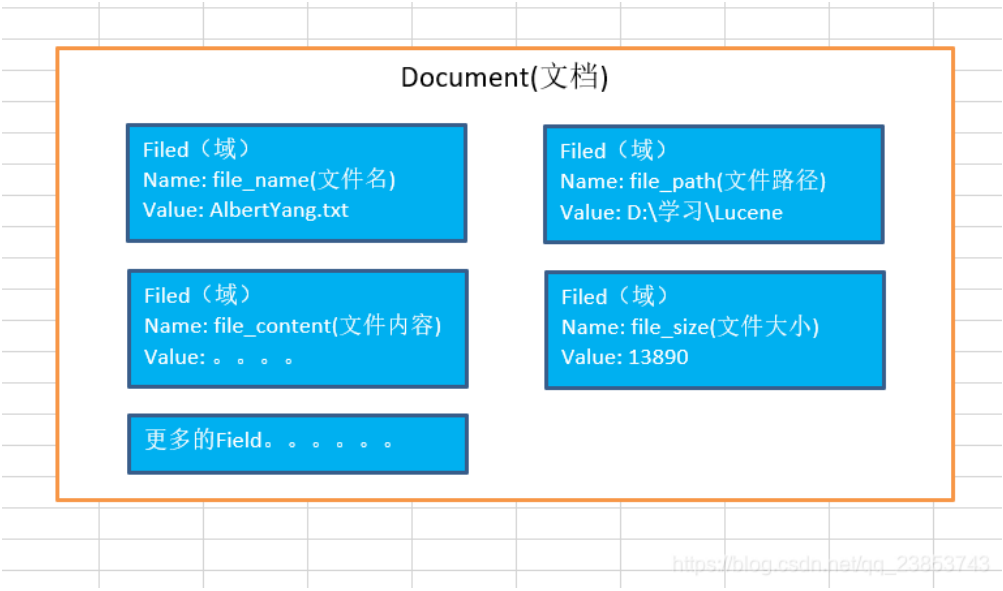


图 3

获取原始数据的目的是为了创建索引，在创建索引前需要将原始数据创建成文档（Document），文档中包括一个一个的域（Field），域中存储原始数据的内容。这里可以把 Document 理解为数据库表中的一条记录，可以把域理解为数据库中的字段。可以将磁盘上的一个文件当成一个 document，Document

中包括一些 Field（file_name 文件名称、file_path 文件路径、file_size 文件大小、file_content 文件内容）

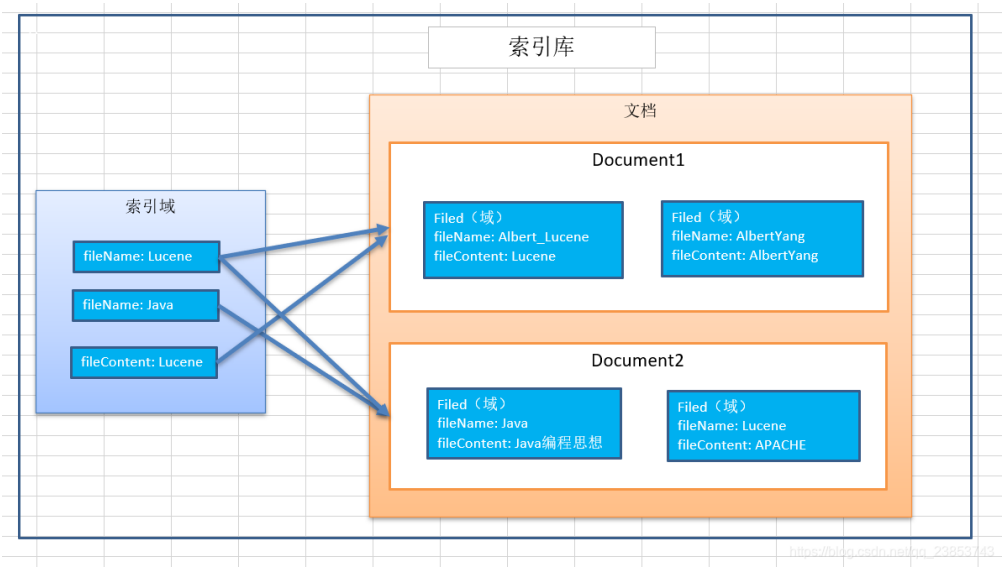


图 4

文档：对非结构化的数据统一格式为 document 文档格式，一个文档有多个 field 域，不同的文档其 field 的个数可以不同，建议相同类型的文档包括相同的 field。本例子一个 Document 对应一个磁盘上的文件。索引域：用于搜索程序从索引域中搜索一个一个词，根据词找到对应的文档。将 Document 中的 Field 的内容进行分词，将分好的词创建索引，索引 =Field 域名: 词。

### 2.2.2 创建索引

1. 创建一个 indexwriter 对象：指定索引库的存放位置 Directory 对象并指定一个分析器对文档内容进行分析。
2. 通过 IO 读取磁盘的文件信息。
3. 创建 Document 对象，将文件信息添加到 document 对象中。
4. 使用 indexwriter 对象将 document 对象写入索引库，此过程进行索引创建。并将索引和 document 对象写入索引库。
5. 关闭 IndexWriter 对象。

### 2.2.3 搜索索引

1. 创建一个 indexsearcher 对象：指定索引库的存放位置 Directory 对象并指定一个 indexreader 对象读取文档信息。
2. 创建一个 TermQuery 对象，指定查询的域和查询的关键词。
3. 执行查询。

4. 返回查询结果。遍历查询结果并输出。

5. 关闭 IndexReader 对象。

### 2.3 SGML 格式的解析器

通过对数据进行观察可知：每个 tdt3 的数据都有 Doc、DocNo、DocType、TxtType、Text 构成。其中 Doc 为包括后续所有结构在内的总结构。所以我们只需要通过 <XXX> 和 </XXX> 的标识符就可以将中间的文档信息提取出来：

```
public class SGMLParser {
    private String docNo;
    private String docType;
    private String txtType;
    private String docContent;

    public String getDocNo() {
        return docNo;
    }
    public String getDocType() {
        return docType;
    }
    public String getTxtType() {
        return txtType;
    }
    public String getDocContent() {
        return docContent;
    }

    public void setDocNo(String docNo) {
        this.docNo = docNo;
    }
    public void setDocType(String docType) {
        this.docType = docType;
    }
    public void setTxtType(String txtType) {
        this.txtType = txtType;
    }
    public void setDocContent(String docContent) {
        this.docContent = docContent;
    }

    public SGMLParser(String fileContent) {
        int docNo1 = fileContent.indexOf("<DOCNO>");
        int docNo2 = fileContent.indexOf("</DOCNO>");
        int docType1 = fileContent.indexOf("<DOCTYPE>");
        int docType2 = fileContent.indexOf("</DOCTYPE>");
        int txtType1 = fileContent.indexOf("<TXTTYPE>");
        int txtType2 = fileContent.indexOf("</TXTTYPE>");
        int Text1 = fileContent.indexOf("<TEXT>");
```

```
        int Text2 = fileContent.indexOf("</TEXT>");

        this.docNo = fileContent.substring(docNo1 + 7, docNo2).trim();
        this.docType = fileContent.substring(docType1 + 9, docType2).trim();
        this.txtType = fileContent.substring(txtType1 + 9, txtType2).trim();
        this.docContent = fileContent.substring(Text1 + 6, Text2).trim();
    }
}
```

## 2.4 索引库建立

### 2.4.1 文件的读取

在该代码逻辑的基础上，因为 TDT3 的数据到根文件夹之间只存在一层文件夹嵌套，所以只需要在读取文件路径时多套一层循环即可读取 TDT3 的所有文件。

```
File total_path = new File("D:\\UniversityStudy\\Grade3Term2\\信息检索\\tdt3");
        File[] list_paths = total_path.listFiles();
        if (list_paths != null) {
            for (File list_path : list_paths) {
                File[] paths = list_path.listFiles();
                if(paths != null) {
                    for (File file : paths) {
                        if(file.isFile()){...}
```

### 2.4.2 对 SGML 格式文档的各部分信息分开读取

我们通过调用 SGMLParser 对文档进行解析，提取出文档的各部分信息并将他们保存在 Document 对象中特别地，对于文档内容 docContent，因为我们需要存储文档内容、需要存储分词信息、在后续打分算法中也需要其项向量 TermVector，所以这里我们需要自己构建 FieldType 作为构建 docContent 域的 config。

```
String file_content = FileUtils.readFileToString(file, "UTF-8");
SGMLParser sgmlParser = new SGMLParser(file_content);
Field fileDocNoField = new StoredField("docNo", sgmlParser.getDocNo());
Field fileDocTypeField = new StringField("docType", sgmlParser.getDocType(),
    Field.Store.YES);
Field fileTxtTypeField = new StringField("txtType", sgmlParser.getTxtType(),
    Field.Store.YES);


FieldType ft = new FieldType();
ft.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);
ft.setStored(true);
ft.setStoreTermVectors(true);
ft.setTokenized(true);
ft.setStoreTermVectorPositions(true);
```

```
ft.setStoreTermVectorOffsets(true);
Field fileTextField = new Field("docContent", sgmlParser.getDocContent(), ft);
```

## 2.5 索引的搜索

### 2.5.1 查询的构造

在这里，我们需要读取 QueryParser 通过解析命令所构建的 hits 值，以及 keyWords 字符串。如果该字符串中间有空格，说明该字符串原本是通过双引号输入的短语，或通过连字符输入的单词，于是构造 PhraseQuery 作为短语查询，其中 setslop(0) 可以保证短语或连字符中的单词可以相邻；对于单个的单词，直接构造 TermQuery。最后构造 BooleanQuery 用来查询所有 Query 的并集。

```
        // Find directory stored index of documents
        Directory directory = FSDirectory.open(Paths.get(".\\temp\\index"));
        // Struct the Reader and Searcher
        IndexReader indexReader = DirectoryReader.open(directory);
        IndexSearcher indexSearcher = new IndexSearcher(indexReader);
        DocRanking docRanking = new DocRanking();

        // BooleanQueryBuilder for multi-keyword query
        BooleanQuery.Builder booleanQueryBuilder = new BooleanQuery.Builder();
        // QueryParser can parse the query into hits and keywords
        QueryParser queryParser = new QueryParser(input);
        int maxHit = queryParser.getHits();
        List<String> keyWords = queryParser.getKeyWords();
        for (String keyWord : keyWords) {
            // If keyword has " ", it means that this is a Doubly quoted phrase query, and
    we use PhraseQuery.
            if(keyWord.contains(" ")) {
                String[] words = keyWord.split(" ");
                PhraseQuery.Builder phraseQueryBuilder = new PhraseQuery.Builder();
                // setSlop(0) to make words adjoining
                phraseQueryBuilder.setSlop(0);
                for (String word : words) {
                    phraseQueryBuilder.add(new Term("docContent", word));
                    docRanking.addTerms(word);
                }
                PhraseQuery phraseQuery = phraseQueryBuilder.build();
                // the Document only can be selected when every keyword can be found in it
                booleanQueryBuilder.add(phraseQuery, BooleanClause.Occur.SHOULD);
            }
            else {
                Query query = new TermQuery(new Term("docContent", keyWord));
                booleanQueryBuilder.add(query, BooleanClause.Occur.SHOULD);
                docRanking.addTerms(keyWord);
            }
        }

        BooleanQuery booleanQuery = booleanQueryBuilder.build();
```

### 2.5.2 基于打分算法获得文档

构造 Query 后，我们需要查询极大值数量的文档，以此获得所有符合条件的文档便于后续打分算法的排序。而我们通过调用 DocRanking 对象，传入 hits 和所有的 doc 之后，获得了对应的打分最高的 hits 个 doc，我们按照要求的格式，输出每一个文章对应的 score rank、[score]、DocNo、Text 摘要。摘要的实现只需对 Text 的长度进行判断，若长于设定值则输出长度等于该设定值的子字符串。关于 DocRanking 对象的实现会在后续内容中说明。

```
        // get the Documents with the Top-N highest scores
        TopDocs topDocs = indexSearcher.search(booleanQuery, 10000);
        ScoreDoc[] allDocs = topDocs.scoreDocs;
        // when maxHit larger than the length of scoreDocs, the real hits is equal to the
    length
        System.out.println("Total hits:" + Math.min(allDocs.length, maxHit));
        docRanking.setReader(indexReader);
        List<ScoreDoc> scoreDocs = docRanking.getMyScoreDocs(allDocs, maxHit);
        // rank for the documents
        int rank = 0;
        for (ScoreDoc scoreDoc : scoreDocs) {
            rank++;
            int doc = scoreDoc.doc;
            // get the score
            float score = scoreDoc.score;
            System.out.print(rank + " [" + score + "] ");
            Document document = indexSearcher.doc(doc);
            String fileDocNo = document.get("docNo");
            System.out.println(fileDocNo);
            String fileContent = document.get("docContent");
            // if the Document is too long, an abstract which length is 500 will be
    printed instead of itself
            if (fileContent.length() > 500) {
                System.out.println(fileContent.substring(0, 500) + "...");
            }
            else {
                System.out.println(fileContent);
            }
            // print a line to split two documents

    System.out.println("----------------------------------------------------------------------");
```

## 2.6 查询语句的解析器

### 2.6.1 查询语句格式判断

首先对整个 Query 以空格进行分割，检查第一个词是否为”search”，如果不是则抛出异常。

然后在后续单词中寻找开头格式为"–" 的单词，如果格式符合"–hits=" 则将 hits 赋值，不符合则抛出异常。

### 2.6.2 对单词和双引号短语的处理

对于单词，在过滤后直接添加到 keyWords 中；对于双引号短语，在经过分割后，双引号短语也被分割开，但短语的首个单词的首个字符为双引号，末尾单词的最后一个字符也为双引号，以此我们可以重新将短语组合起来并删除双引号，过滤后加入 keyWords。

```
    public QueryParser(String query) throws Exception{
        List<String> terms = new ArrayList<>(Arrays.asList(query.split(" ")));
        // get the first word as the command
        String term0 = terms.get(0);
        // default hits is 10
        hits = 10;
        // if command is valid
        if (term0.equals("search")) {
            for(int i = 1; i < terms.size(); i++) {
                String term = terms.get(i);
                // if it's a doubly quoted phrase query, build the phrase.
                if(term.startsWith("\"")) {
                    StringBuilder phraseBuilder = new StringBuilder(term);
                    while(!term.endsWith("\"") && i < terms.size() - 1) {
                        i++;
                        term = terms.get(i);
                        phraseBuilder.append(" ").append(term);
                    }
                    // before filter the punctuations and numbers, delete the double
    quotation mark

    keyWords.add(sanitizeQuery(phraseBuilder.toString().replaceAll("\"","")));
                }
                // evaluation of hits
                else if (term.startsWith("--")) {
                    if(term.startsWith("--hits==")) {
                        hits = Integer.parseInt(term.substring(7));
                    }
                    else{
                        throw new Exception("Not a valid hits");
                    }

                }
                else{
                    keyWords.add(sanitizeQuery(term));
                }
            }
        }
        else{
            throw new Exception("Not a valid query");
```

```
        }
    }
```

### 2.6.3 过滤器

根据要求, 我们需要过滤所有的数字以及标点符号，同时对大小写进行处理。在这里以空格替换来达到过滤；因为查询语句默认不区分大小写，这里为了方便全部转换为小写：

```
    // function to filter the punctuation and numbers
    private String sanitizeQuery(String query){
        return query.replaceAll("\\d+", "").replaceAll("\\p{Punct}", " ").toLowerCase();
    }
```

## 2.7 简易的命令行包装

命令行的实现就非常简单了，定义一个 scanner 来获取用户输入，如果用户输入”exit”（不区分大小写）就关闭 scanner 退出程序，否则就将用户的输入传输到 luceneQuery.ProcQuery() 查询函数中：

```
public class Program {
    public static void main(String[] args) throws Exception {
        // scanner to get user's input
        Scanner scanner = new Scanner(System.in);
        String input;
        LuceneQuery luceneQuery = new LuceneQuery();

        while(true) {
            // DOS prompt
            System.out.print("# ");
            // get input
            input = scanner.nextLine();
            // exit
            if(input.equalsIgnoreCase("exit")){
                break;
            }
            // if query is valid
            try {
                luceneQuery.ProcQuery(input);
            }
            catch (Exception e) {
                System.out.println(e);
            }
        }

        scanner.close();
        System.out.println("Program exited.");
    }
}
```

## 2.8 文档相关性打分算法与排序

### 2.8.1 BM25 算法

BM25 是一种用来评价搜索词和文档之间相关性的算法，它是一种基于概率检索模型提出的算法，简单描述：我们有一个 query 和一批文档 Ds，现在要计算 query 和每篇文档 D 之间的相关性分数，我们的做法是，先对 query 进行切分，得到单词 $q_i$，然后单词的分数由三部分组成：

- 单词 $q_i$ 和 D 之间的相关性
- 单词 $q_i$ 和 query 之间的相关性
- 每个单词的权重

最后对每个单词的分数我们做一个求和，就得到了 query 和文档之间的分数。

单词权重：单词的权重最简单的就是 idf 值，即 $log[\frac{N}{df_t}]$，也就是有多少文档包含某个单词信息进行变换。如果在这里使用 IDF 的话，那么整个 BM25 就可以看作是一个某种意义下的 TF-IDF，只不过 TF 的部分是一个复杂的基于文档和查询关键字、有两个部分的词频函数，还有一个就是用上面得到的 ct 值。

单词和文档的相关性：tf-idf 中，这个信息直接就用“词频”，如果出现的次数比较多，一般就认为更相关。但是 BM25 洞察到：词频和相关性之间的关系是非线性的，具体来说，每一个词对于文档相关性的分数不会超过一个特定的阈值，当词出现的次数达到一个阈值后，其影响不再线性增长，而这个阈值会跟文档本身有关。在具体操作上，我们对于词频做了”标准化处理“，具体公式如下：

$$\frac{(k_1+1)tf_{td}}{k_1[(1-b)+b\times(L_d/L_{ave})]+tf_{td}}$$

其中，$tf_{td}$ 是词项 t 在文档 d 中的权重，Ld 和 Lave 分别是文档 d 的长度及整个文档集中文档的平均长度。$k_1$ 是一个取正值的调优参数，用于对文档中的词项频率进行缩放控制。如果 $k_1$ 取 0，则相当于不考虑词频，如果 $k_1$ 取较大的值，那么对应于使用原始词项频率。b 是另外一个调节参数（0 b 1），决定文档长度的缩放程度：b = 1 表示基于文档长度对词项权重进行完全的缩放，b = 0 表示归一化时不考虑文档长度因素。

单词和查询的相关性：当查询很长时需要考虑，对于查询词项也可以采用类似的权重计算方法。$\frac{(k_3+1)tf_{tq}}{k_3+tf_{tq}}$。但在这里我们忽略这一项。

最后的公式就是对每个单词以上三者的乘积进行求和。

$$RSV_d=\sum_{t\in q}log[\frac{N}{df_t}]\cdot\frac{(k_1+1)tf_{td}}{k_1[(1-b)+b\times(L_d/L_{ave})]+tf_{td}}\cdot\frac{(k_3+1)tf_{tq}}{k_3+tf_{tq}}$$

### 2.8.2 获取 Term 和 reader

我们需要 indexReader 来对任意一篇文档的 TermVector 等信息进行查看和提取，同时需要 term 来计算词频等关键数值：

```
// to get terms from search command
public void addTerms(String term) {
    this.terms.add(term);
}
```

```
// get the reader
public void setReader(IndexReader reader) {
    this.reader = reader;
}
```

### 2.8.3 获取词频

我们需要获取 term 在指定文档中的词频，我们通过 TermVector 反向提取出该文档的 terms，然后对于枚举的所有 term，将转为 string 格式、utf-8 编码的字符串与关键词相同的 term 的词频返回；对于所有文档的词频，只需要将每个文档的词频加起来即可。

```
// function to get the frequency of the doc
private int getTermFreq(int doc, String term0) throws IOException {
    TermVectors termVectors = reader.termVectors();
    Terms terms = termVectors.get(doc, "docContent");
    if(terms != null) {
        TermsEnum termsEnum = terms.iterator();
        BytesRef term = null;
        while((term = termsEnum.next()) != null) {
            if(term.utf8ToString().matches(term0)) {
                return (int)termsEnum.totalTermFreq();
            }
        }
    }
    return 0;
}

// function to get the frequency in all docs
private int getDocFreq(String term0) throws IOException {
    Term term = new Term("docContent", term0);
    return reader.docFreq(term);
}
```

### 2.8.4 获取文档长度

通过提取指定文档的”docContent” 域，将域内容的字符串提取用空格分割，得到文档的单词数；平均文档长度则通过计算总文档长度除以文档个数得到：

```
private int getDocNum(){
    return reader.numDocs();
}

// function to get the size of the file
private int getContentSize(int doc) throws IOException {
    Document document = reader.document(doc);
    Field field = (Field) document.getField("docContent");
    return field.stringValue().split(" ").length;
```

```
    }

    // function to get the average size of files
    private void setAvgContentSize() throws IOException {
        int numDocs = getDocNum();
        int totalContentSize = 0;
        for (int i = 0; i < numDocs; i++) {
            totalContentSize += getContentSize(i);
        }
        this.avgContentSize = (float) totalContentSize / numDocs;
    }
```

### 2.8.5 计算文档的 BM25 分数

根据上述的理论，因为查询语句不长，我们直接忽略该项，计算方法为：

$$Score(q,d) = \sum_{t \in q} log[\frac{numDocs - docFreq + 0.5}{docFreq + 0.5}] \cdot \frac{termFreq(1+k_1)}{[k_1(1-b+b \times contentSize/avgContentSize)]}$$

由此得到打分函数：

```
    // function to get BM25 score of one term
    private float getBM25(int doc, String term) throws IOException {
        int termFreq = getTermFreq(doc, term);
        int docFreq = getDocFreq(term);
        float contentSize = getContentSize(doc);
        float k1 = (float) 1.2;
        float b = (float) 0.75;
        float K = (float)(k1 * (1 - b + b * (double) contentSize / avgContentSize));
        float tf = termFreq * (1 + k1) / (K + docFreq);
        float idf = (float)Math.log((float)(getDocNum() - docFreq + 0.5) / (docFreq + 0
    .5));
        return tf * idf;
    }

    // function to get score of the doc with all terms
    private float getScore(int doc) throws IOException {
        float Bm25 = 0;
        for (String term : terms) {
            Bm25 += getBM25(doc, term);
        }
        return Bm25;
    }
```

### 2.8.6 根据打分进行排序并返回 Top-N

这里我们传入在 LuceneQuery 中得到的所有符合条件的 doc，对每个 doc 计算他们的分数并保存在 score 变量中，然后加入到 myScoreDocs 列表中，方便后续进行排序；调用 Collections 进行排序，

依据为 score 大小。

```
// function to get sorted Docs with the top-N highest BM25 scores
private void setMyScoreDocs(ScoreDoc[] allDocs) throws IOException {
    for (ScoreDoc oneDoc : allDocs) {
        oneDoc.score = getScore(oneDoc.doc);
        this.myScoreDocs.add(oneDoc);
    }

    Collections.sort(this.myScoreDocs, (o1, o2) -> {
        if(o1.score < o2.score){
            return 1;
        }
        else if(o1.score > o2.score){
            return -1;
        }
        return 0;
    });
}
```

然后就是可调用的 get 函数了，传入所有的 doc 以及 hits，计算出平均长度后得到打分并排序的 List<ScoreDoc> 结构，根据 hits 和该结构的 size 判断返回大小：

```
// function to calculate avgFileSize and get the docs returned
public List<ScoreDoc> getMyScoreDocs(ScoreDoc[] allDocs, int hits) throws IOException {
    setAvgContentSize();
    setMyScoreDocs(allDocs);
    if(myScoreDocs.size() < hits){
        return this.myScoreDocs;
    }
    else{
        return this.myScoreDocs.subList(0, hits);
    }
}
```

# 三、 实验结果

## 3.1 Q1- hurricane

```
# search hurricane
Total hits:10
1 [0.25766698] NYT19981108.0095
Jose Antonio Amaya Garcia has lived long enough to survive three hurricanes,
but the devastation wrought on Central America by the gargantuan maelstrom
known as Mitch took even him by surprise. ``It's a punishment from
God,'' Amaya, an elderly carpenter, said late last week as he searched
under an avalanche for what was left of his house. He is tiny and
frail in his soiled shirt and pants, the last clothing he owns. ``I
am 73, and I've never seen a disaster like this.'' As the scope of
th...
---------------------------------------------------------------------
2 [0.25766698] NYT19981108.0132
Jose Antonio Amaya Garcia has lived long enough to survive three hurricanes,
but the devastation wrought on Central America by the gargantuan maelstrom
known as Mitch took even him by surprise. ``It's a punishment from
God,'' Amaya, an elderly carpenter, said late last week as he searched
under an avalanche for what was left of his house. He is tiny and
frail in his soiled shirt and pants, the last clothing he owns. ``I
am 73, and I've never seen a disaster like this.'' As the scope of
th...
---------------------------------------------------------------------
3 [0.2449264] NYT19981003.0052
In the days before Hurricane Georges struck the Gulf Coast last week,
storm forecasters faced one of their most trying responsibilities:
predicting the inherently uncertain behavior of a hurricane in time
to move people out of harm's way. The basic problem is that even though
experts have improved their ability to forecast the track of a hurricane
by 30 or 40 percent in the last decade or so, the predictions are
still off, on average, by about 200 miles 72 hours before the storm's
expecte...
---------------------------------------------------------------------
4 [0.2449264] NYT19981005.0056
In the days before Hurricane Georges struck the Gulf Coast last week,
storm forecasters faced one of their most trying responsibilities:
predicting the inherently uncertain behavior of a hurricane in time
to move people out of harm's way. The basic problem is that even though
experts have improved their ability to forecast the track of a hurricane
by 30 or 40 percent in the last decade or so, the predictions are
still off, on average, by about 200 miles 72 hours before the storm's
expecte...
---------------------------------------------------------------------
5 [0.2163661] VOA19981201.0500.0794
The hurricane season has officially ended, although one storm is still
pushing its way through the Atlantic Ocean. Hurricane Nicole is no
threat to anyone right now, but she is probably the last of what weather
forecasters are calling the deadliest Atlantic hurricane season in
more than 200 years. As VOA's Challus McDonough reports, the forecasters
are analyzing the data from this year's storm to help them prepare
for the future. U.S. hurricane researchers say they collected huge
amounts ...
---------------------------------------------------------------------
6 [0.17221531] NYT19981113.0421
All that remains of the cruise ship are two life rafts, seven life
```

图 5

### 3.2 Q2- mitch george

```
# search mitch george
Total hits:10
1 [0.14621946] NYT19981127.0213
``Curious George'' is a best-selling children's book series. Furious
George is a struggling punk rock band. You wouldn't necessarily confuse
the two. George in the books is a monkey. George in the band is a
person. But then again, maybe you might. The ``Curious George'' books
have bright yellow covers with red lettering and pictures of George
the monkey. The Furious George albums are bright yellow with red lettering
and also likenesses of George the monkey _ in leather and sunglasses.
To ...
------------------------------------------------------------------------
2 [0.13285832] CNN19981029.1130.0584
Hurricane Mitch is losing strength as it hovers off the coast of Honduras.
At last report, the storm was near the coastal city of limon. Forecasters
expect the storm will remain a threat to the northwestern caribbean
for two more days. Martin Savidge takes a look at how one city is
coping with Mitch. In cancun, it was another nervous night. It will
be another day of wondering and waiting on what hurricane Mitch will
do. It follows what had been a previously anxious day for tourists.
When ...
------------------------------------------------------------------------
3 [0.1160383] APW19981027.0241
Honduras braced for potential catastrophe Tuesday as Hurricane Mitch
roared through the northwest Caribbean, churning up high waves and
intense rain that sent coastal residents scurrying for safer ground.
President Carlos Flores Facusse declared a state of maximum alert
and the Honduran military sent planes to pluck residents from their
homes on islands near the coast. At 0900 GMT Tuesday, Mitch was 95
miles (152 kilometers) north of Honduras, near the Swan Islands. With
winds near 180 mp...
------------------------------------------------------------------------
4 [0.11563389] PRI19981102.2000.0373
Mitch is now a mere tropical storm but as a Hurricane it was devastating.
At one point during its destructive sweep through the Carribean last
week, Mitch became the fourth most powerful Atlantic storm of the
century. It is the deadliest hurricane to hit central America since
Fifi killed 3000 people in Honduras in 1974. Mitch's death toll is
now put at more than 6,000 mostly due to floods and landslides. In
a moment we'll speak with a reporter in Managua about the relief operations
there....
------------------------------------------------------------------------
5 [0.099651136] CNN19981027.0130.0045
This is "CNN Headline News." I'm David Goodnow in Atlanta. Thanks
for joining us. The national hurricane center is predicting hurricane
Mitch could cause catastrophic damage, and people in central America
are bracing for the worst. The category 5 hurricane still is at least
three days from landfall with maximum sustained winds near 180 miles
an hour. Forecasters predict Mitch will bring at least 10 to 15 inches
of rain to parts of central America. Ginger Blackstone has more. Along
the coa...
------------------------------------------------------------------------
6 [0.09934183] APW19981105.1220
Better information from Honduras' ravaged countryside enabled officials
```

图 6

### 3.3 Q3- bill clinton israel

```
# search bill clinton israel
Total hits:10
1 [0.12230524] NYT19981115.0095
In an attempt to rebuild religious identity among young Jews, who
are marrying non-Jews and abandoning the faith in large numbers, Jewish
organizations plan to start a program that will pay for any Jew in
the world between age 15 and 26 to travel to Israel for 10 days. The
program, Birthright Israel, is expected to cost $300 million over
five years and to be financed by the Israeli government, a group of
major Jewish donors from North America and the Council of Jewish Federations.
Israeli...
------------------------------------------------------------------------
2 [0.11297733] PRI19981223.2000.2008
The latest flashpoint in the Middle East is along the border between
Israel and Lebanon. Today, guerrillas in south Lebanon fired dozens
of rockets into northern Israel injuring at least 13 people. There
might have been more casualties, but Israelis were ready for the attack.
Two days ago, an Israeli air strike killed a woman and her six children
in eastern Lebanon. It was the latest tragedy in a 13-year long war
between Iranian-backed Hezbollah guerrillas and Israel. The BBC's
Lise Duset...
------------------------------------------------------------------------
3 [0.10906364] NYT19981210.0357
An Egyptian gas pipeline is snaking its way east across the Sinai
desert, laying the groundwork for a regional power network that could
bind Israel closer to its Arab neighbors and radically change its
energy and security strategy. By directly fueling electric power plants,
as its builders intend, the Sinai pipeline would end Israel's dependence
on sources a continent or more away for vital energy imports. Israeli
officials agree that the Egyptian connection would be the easiest
and cheap...
------------------------------------------------------------------------
4 [0.1032838] NYT19981114.0164
Lisa Smart's presence on the operating table at Beth Israel Medical
Center on a Friday afternoon last November began in an utterly unremarkable
way. Mrs. Smart, a 30-year-old financial analyst, had found the doctor
performing the procedure, Robert Klinger, the way many busy, young
and healthy people do _ by breezing through her HMO's book of approved
doctors and choosing one who was convenient and recommended by her
primary care physician. Her only real medical problem was as common
as it...
------------------------------------------------------------------------
5 [0.089212835] APW19981218.0864
Israel will take all necessary measures against Iraq should it ``dare''
to attack the Jewish state, the defense minister said Friday. The
minister, Yitzhak Mordechai, was more direct in his warning than Israeli
Prime Minister Benjamin Netanyahu who on Thursday, the first day of
U.S. airstrikes against Iraq, would only say that Israel reserved
the right to defend itself. Iraqi leader Saddam Hussein has not threatened
to attack Israel in retaliation for the U.S. bombings, and Mordechai
told...
------------------------------------------------------------------------
6 [0.08101578] PRI19981126.2000.2027
The case of Jonathan Pollard has long been a thorn in the side of
```

图 7

### 3.4 Q4- ”newt gingrich” down

```
# search "newt gingrich" down
Total hits:10
1 [0.72120655] NYT19981107.0032
House Speaker Newt Gingrich, who orchestrated the Republican takeover
of Congress in 1994 and presided this year over what at times seemed
like the political destruction of President Clinton was himself driven
from office Friday by a party that swiftly turned on him after its
unexpected losses in Tuesday's midterm elections. Catching virtually
everyone on Capitol Hill by surprise, Gingrich announced Friday night
in two conference calls to other Republicans that he would not seek
re-electi...
------------------------------------------------------------------------
2 [0.70033866] NYT19981106.0531
House Speaker Newt Gingrich, who orchestrated the Republican takeover
of Congress in 1994 and pressed the impeachment inquiry into President
Clinton, was driven from office Friday by a party that swiftly turned
on him after its unexpected losses in Tuesday's midterm elections.
Catching virtually everyone on Capitol Hill by surprise, Gingrich
announced Friday night in two conference calls to other Republicans
that he would not seek re-election as Speaker and would leave Congress
altogether...
------------------------------------------------------------------------
3 [0.63097936] NYT19981105.0509
A struggle for control of the House is under way, with Rep. Robert
Livingston conducting a telephone campaign that could lead to him
running against Newt Gingrich as speaker. But Gingrich's counter-campaign
has given some members pause about ousting him. At the same time,
a small band of Republicans vowed on Thursday that they would not
vote to re-elect Gingrich under any circumstances, a move that, because
of the Republicans' shrunken House majority, could tie the party in
knots for mont...
------------------------------------------------------------------------
4 [0.56322724] NBC19981106.1830.0071
Good evening. Major news tonight, when the congressional election
results came in on tuesday night, a huge disappointment for the republicans,
Newt gingriching in the speaker of the house, immediately blamed the
news media. Since then many of his fellow republicans have been saying
openly Gingrich is the problem, and he must go. Some of his former
allies are openly challenging his leadership team. And tonight, Gingrich
has decided to give up the fight. NBC's Tim russert broke the story
ea...
------------------------------------------------------------------------
5 [0.51481694] NYT19981107.0119
Their wish finally came true: Democrats helped topple House Speaker
Newt Gingrich. But rather than celebrate, many of Gingrich's toughest
foes are now lamenting the loss of the politician they most liked
to demonize. ``We won't have Newt to kick around any more _ and we'll
miss him,'' said Mandy Grunwald, a Democratic media consultant. ``In
the way the Republican Party spent years using Ted Kennedy as their
punching bag _ and made millions in direct mail off him _ I don't
think we've ever...
------------------------------------------------------------------------
6 [0.42191803] NYT19981106.0464
House Speaker Newt Gingrich, who orchestrated the Republican revolution
```

图 8

### 3.5 Q5- nba strike closed-door

```
# search nba strike closed-door
Total hits:10
1 [0.6594337] NYT19981229.0365
As the prospect of the National Basketball Association season being
canceled increases, so has talk that a rival league will be formed.
Many player agents have said recently that if the season was terminated,
a new league involving some of the NBA's biggest stars could be in
place by next fall. And in a nationwide conference call on Monday,
the players association director, Billy Hunter, mentioned for the
second time in two weeks that the season's cancellation would make
room for a compet...
------------------------------------------------------------------------
2 [0.55097914] NYT19981031.0165
Four short years ago, hockey was hot. The Rangers won the Stanley
Cup and rap artists wore National Hockey League jerseys, even if they
couldn't name many players. National sports magazines and business
publications proclaimed the ascendancy of the NHL, comparing its growth
potential to that of the National Basketball Association, Major League
Baseball and the National Football League. That October, when baseball's
playoffs and World Series were canceled by a strike, hockey's position
see...
------------------------------------------------------------------------
3 [0.54045] NYT19981108.0125
On Oct. 30, children, teen-agers and adults packed the balcony overlooking
the gymnasium floor at the Indiana Institute of Technology to watch
the Fort Wayne Fury practice. Excited fans jockeyed for position,
standing on tiptoe and pushing one another aside to videotape and
photograph the action on the court. The Fury is one of the strongest
teams in the Continental Basketball Association, but the crowd had
not gathered to watch it run through its plays _ they were there to
see the Fury's...
------------------------------------------------------------------------
4 [0.46866906] NYT19981113.0416
At 1 p.m. Thursday, cool melodies and shimmering chords floated over
midtown Manhattan. These were the sweetest sounds to date from a deepening
labor dispute between National Basketball Association players and
team owners. The unlikely source of this music was the new NBA Store,
on Fifth Avenue and 52nd Street. If you are a player, the store is
enemy territory; for the owners, the store is one more golden goose.
For a tourist, the store is a cornucopia of NBA paraphernalia. For
Wayman Tis...
------------------------------------------------------------------------
5 [0.39020473] PRI19981216.2000.1185
This is THE WORLD. I'm Lisa Mullins. For two and a half months now,
NBA fans have been hearing more about salary caps and escrow tax systems
than about fast breaks and slam dunks. With no negotiations scheduled
to end the lock-out, some pro basketball fans are giving up on the
season. Even some players are considering alternative plans. Stephon
Marbury of the Minnesota Timberwolves says if the season is cancelled,
he will head for Europe, and Vlade Divac of the Charlotte Hornets
is also s...
------------------------------------------------------------------------
6 [0.35242614] APW19981001.1151
While Michael Jordan and Magic Johnson have engaged the public in
```

图 9

## 四、 小结

本次大作业我基于 Lucene 设计了命令行程序，大大加深了我对 tf-idf 的概念、以及 tf-idf 算法和 bm25 等文档相关性算法的理解；同时让我熟悉了 java 语言编程，最大的感受是 IDEA 的代码自动补全功能实在是太好用了，直接省去了重复性步骤需要手动敲的麻烦。同时也让我对信息检索产生了兴趣，希望可以在后续中应用到实际。