信息检索实验报告

课程名称	信息检索大实验				
学生姓名	陈曦	学号	2020302181081	指导老师	李晨亮
专业	网络安全	班级	2020 级 3 班	实验时间	2022.5.20

一、实验描述：自定义搜索引擎

使用 Lucene 建立索引，使用包括 37,600 新闻文章的 TDT3 数据集，编写一个命令行搜索引擎。输入关键词可以返回 10 个与关键词最相关的新闻文章，并输出文档相关性的得分。

推荐关键词搜索：

Hurricane

Mitch george

Bill clinton israel

“Newt gingrich” down

Nba strike closed-door

二、实验原理

1. 全文检索

有两种数据，结构化数据和非结构化数据。结构数据指具有固定格式或有限长度的数据，如数据库，元数据等；而非结构化数据指不定长或无固定格式的数

据，如 word 文档等，又叫全文数据。本次实验全文检索我们使用无固定格式的数据。

搜索也有两种，搜索结构化数据和搜索非结构化数据。搜索结构化数据如对数据库的搜索，对元数据的搜索；搜索非结构化数据如利用 windows 搜索文件内容，Linux 下的 grep 命令。本次实验需要搜索非结构化数据。

对于非结构化的搜索有两种方法。一是顺序扫描法，但是对于大量的文件，这种方法就很慢了，不过顺序扫描法对于结构化数据搜索很快；另一种是全文检索，把非结构化数据转化得有一定结构，即将非结构化数据中的一部分信息提取出来，重新组织，使其变得有一定结构，然后对有一定结构的数据进行搜索，从而达到快速搜索非结构化数据的目的。

这部分从非结构化数据中提取出来，然后重新组织的信息，我们称之索引 。例如字典，字典的拼音表和部首检字表就相当于字典的索引，由于对每一个字的解释都是非结构化的，如果字典没有音节表和部首检字表，在茫茫辞海中找一个字只能顺序扫描，即一页一页进行查找。然而字的某些信息可以提取出来进行结构化处理，比如读音，就比较结构化，分声母和韵母，分别只有几种可以一一列举，于是将读音拿出来按一定的顺序排列，每一项读音都指向此字的详细解释的页数。我们搜索时按结构化的拼音搜到读音，然后按其指向的页数，便可找到我们的非结构化数据——即对字的解释说明。

2. 全文检索的流程

全文检索的流程分为两大部分：索引流程和搜索流程。

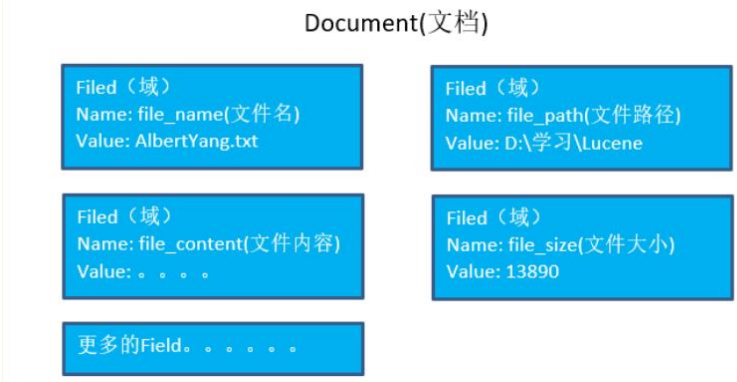
创建索引流程：确定原始内容即要搜索的内容->采集原始内容数据->创建文

档->分析文档(分词)->创建索引。

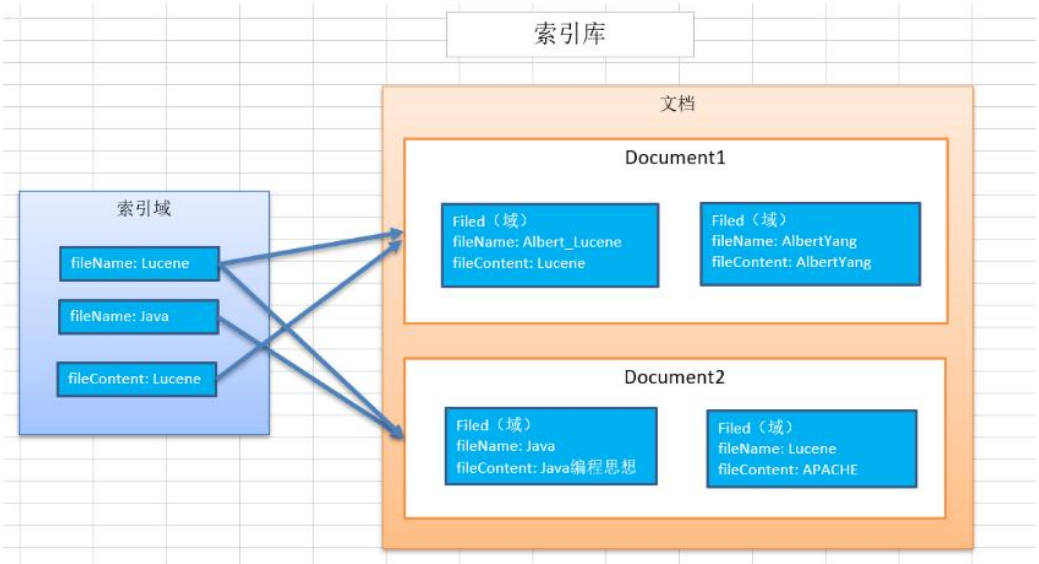
确定原始内容

采集原始内容数据：如果数据是文件系统中的某个文件，就通过 I/O 读取文件的内容。

创建文档对象：文档中包含一个一个的域。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 域（file_name 文件名称、file_path 文件路径、file_size 文件大小、file_content 文件内容），如下图：



索引库结构如下所示：

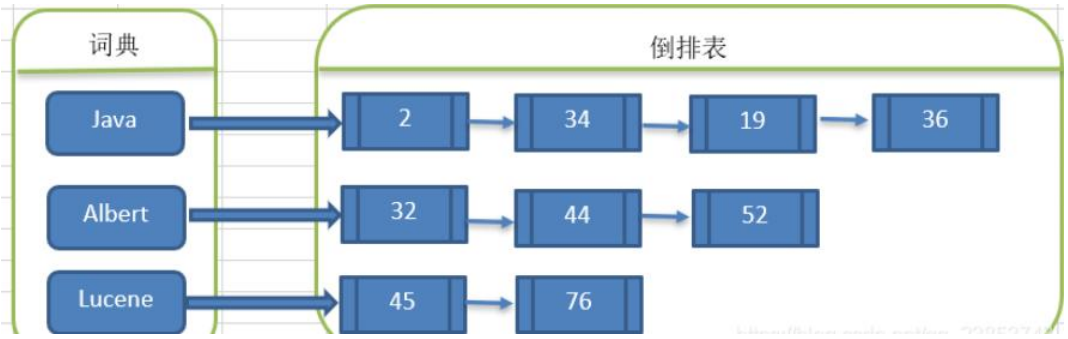


分析文档（分词）：将原始内容创建为包含域（Field）的文档（document），需要再对域中的内容进行分析，分析的过程是经过对原始文档提取单词、将字母转为小写、去除标点符号、去除停用词等过程生成最终的语汇单元，可以将语汇单元理解为一个一个的单词。

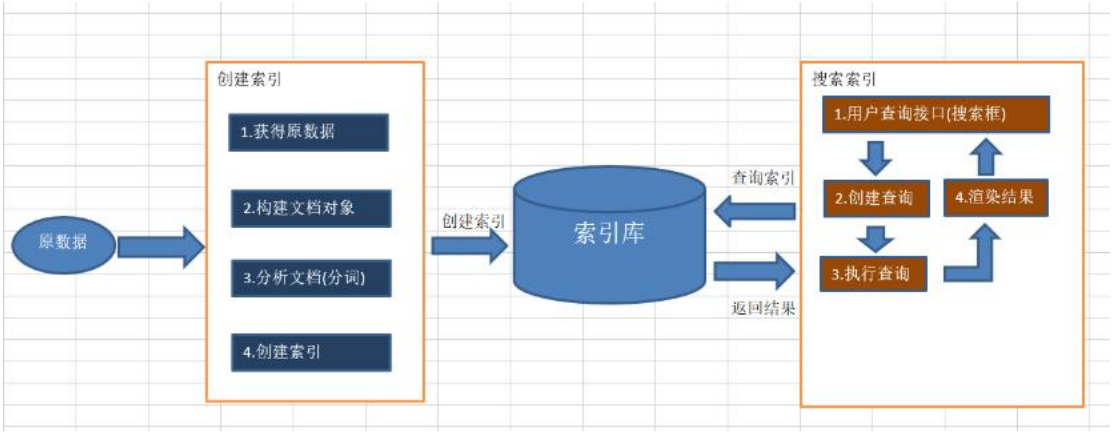
创建文档：对所有文档分析得出的语汇单元进行索引，索引的目的是为了搜索，最终要实现只搜索被索引的语汇单元从而找到 Document（文档）。

索引结构为倒排索引结构。前文提到顺序扫描方法当数据量比较大的时候，搜索效率很低；而对应的倒排索引结构是根据内容（词语）找文档，倒排索引结构也叫反向索引结构，包括索引和文档两部分，索引即词汇表，它是在索引中匹配搜索关键字，由于索引内容量有限并且采用固定优化算法搜索速度很快，找到了索引中的词汇，词汇与文档关联，从而最终找到了文档。

结构如下图所示：



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。整个检索流程如下图：

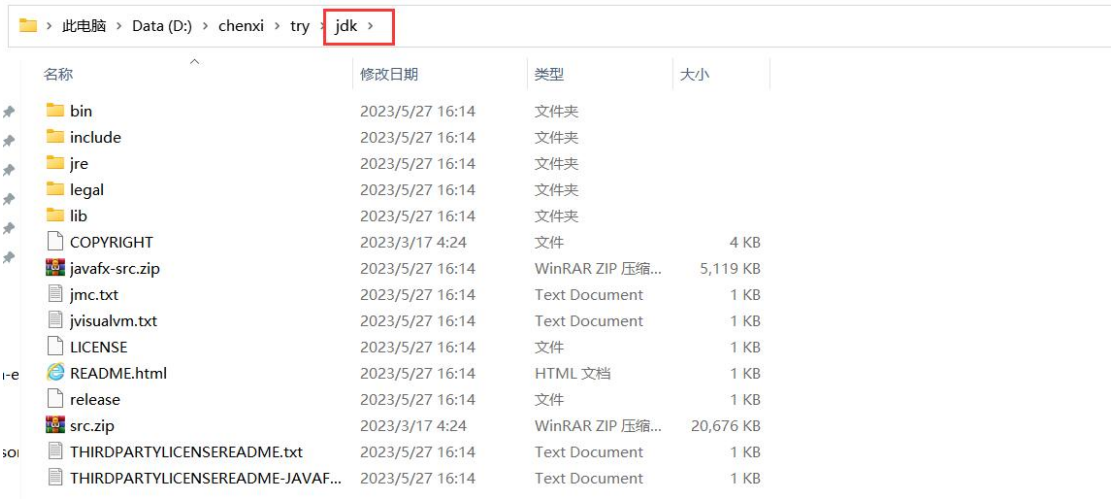


三、实验步骤

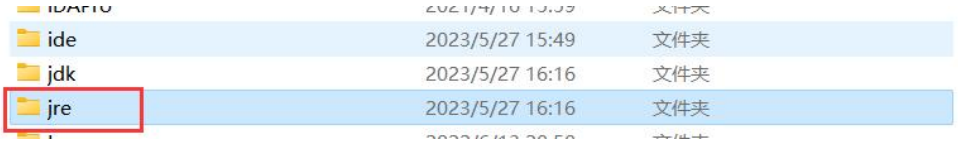
1. 配置环境

使用编辑器 IntelliJ IDEA。

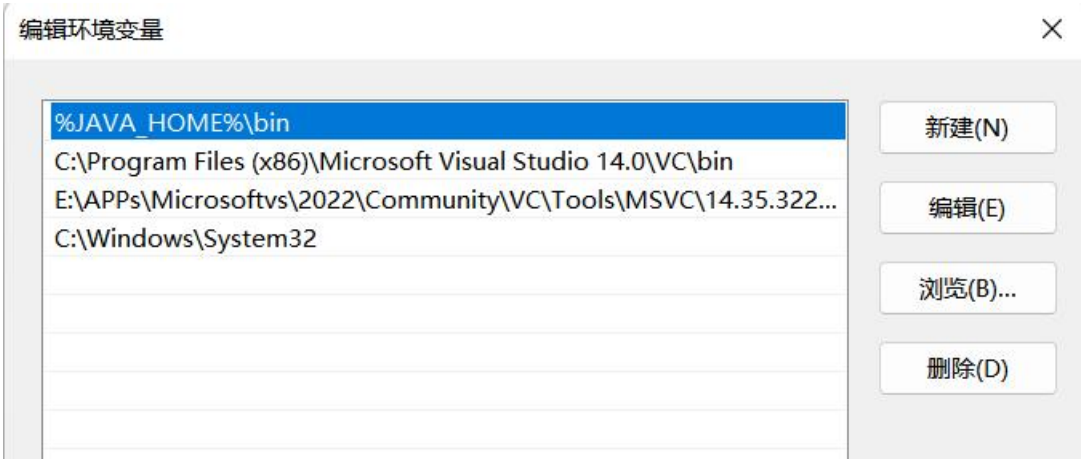
配置 JDK。先将 JDK 下载到本地。



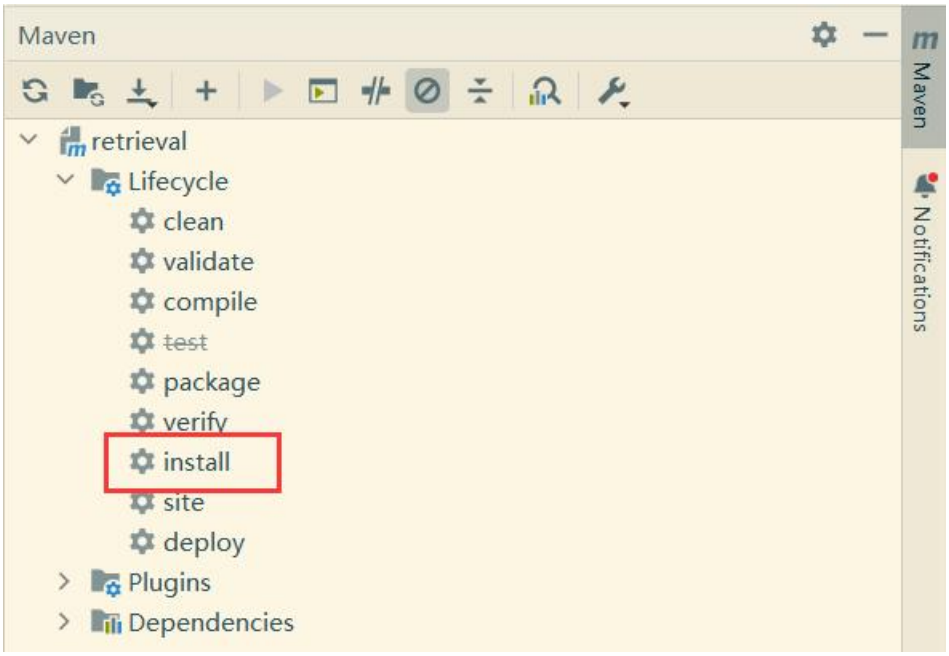
在安装 JDK 的过程中安装 JRE。



配置环境变量（系统变量）。

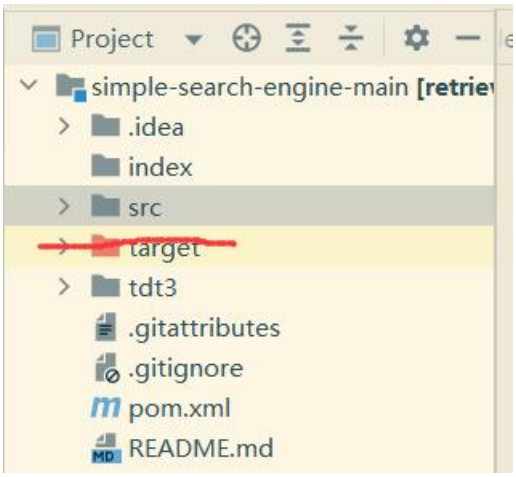

编辑环境变量
%JAVA_HOME%\bin
C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin
E:\APPs\Microsoftvs\2022\Community\VC\Tools\MSVC\14.35.322...
C:\Windows\System32
新建(N)
编辑(E)
浏览(B)...
删除(D)

安装 maven 环境。

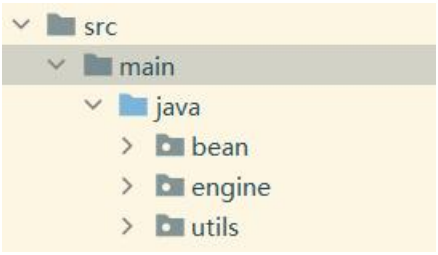

Maven
retrieval
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
Dependencies
Maven
Notifications

【代码实现】

创建 JAVA 项目，并将 TDT3 数据集解压在项目文件夹下。并在项目文件夹下建立索引文件夹，用于存放索引。



源码文件夹 src 下有 JAVA 文件夹，JAVA 文件夹其中包括 bean，engine，utils。



在 bean 文件夹中定义了 Document 类。

DocumentContent.java：

封装文档的各个属性信息。DocumentContent 类包含了文档编号、文档类型、文本类型和文本内容等属性，以及相应的 getter 和 setter 方法。通过解析文件内容字符串，可以方便地获取文档的各个属性信息，并将其封装到 DocumentContent 对象中。

```
1.  package bean;
2.
3.  public class DocumentContent {
4.      private String docNO;
```

```
5.      private String docType;
6.      private String txtType;
7.      private String text;
8.      public String getDocNO() {
9.          return docNO;
10.     }
11.     public String getDocType() {
12.         return docType;
13.     }
14.     public String getTxtType() {
15.         return txtType;
16.     }
17.     public String getText() {
18.         return text;
19.     }
20.     public void setDocNO(String docNO) {
21.         this.docNO = docNO;
22.     }
23.     public void setDocType(String docType) {
24.         this.docType = docType;
25.     }
26.     public void setTxtType(String txtType) {
27.         this.txtType = txtType;
28.     }
29.     public void setText(String text) {
30.         this.text = text;
31.     }
32.     public DocumentContent(String fileContent) {
33.         int docNOBegin = fileContent.indexOf("<DOCNO>");
34.         int docNOEnd = fileContent.indexOf("</DOCNO>", do
   cNOBegin);
35.         int docTypeBegin = fileContent.indexOf("<DOCTYPE>
   ", docNOEnd);
36.         int docTypeEnd = fileContent.indexOf("</DOCTYPE>",
    docTypeBegin);
37.         int txtTypeBegin = fileContent.indexOf("<TXTTYPE>
   ", docTypeEnd);
38.         int txtTypeEnd = fileContent.indexOf("</TXTTYPE>",
    txtTypeBegin);
39.         int textBegin = fileContent.indexOf("<TEXT>", txt
   TypeEnd);
40.         int textEnd = fileContent.indexOf("</TEXT>", text
   Begin);
```

```
41.         if (docNOBegin == -1 || docNOEnd == -1 || docType
   Begin == -1 || docTypeEnd == -1 ||
42.                 txtTypeBegin == -1 || txtTypeEnd == -1 ||
    textBegin == -1 || textEnd == -1) {
43.             int t = 5 / 0;
44.         }
45.         this.docNO = fileContent.substring(docNOBegin + 7,
    docNOEnd).trim();
46.         this.docType = fileContent.substring(docTypeBegin
    + 9, docTypeEnd).trim();
47.         this.txtType = fileContent.substring(txtTypeBegin
    + 9, txtTypeEnd).trim();
48.         this.text = fileContent.substring(textBegin + 6,
   textEnd).trim();
49.     }
50.     @Override
51.     public String toString() {
52.         return "DOCNO:\n\t" + getDocNO() + "\nDOCTYPE:\n\
   t" + getDocType() + "\nTXTTYPE:\n\t" + getTxtType() + "\nT
   EXT:\n\t" + getText() + '\n';
53.     }
54. }
```

在/utils/FilesUtils.java 中，定义了 FileUtils 工具类。

FilesUtils.java：

提供文件操作的各种常用方法。包含了 3 个静态方法：getFileSize()、

getProjectPath() 和 getFileContent()，以获得文件大小，文件路径和文件内容。

FileUtils 工具类可以方便地支持文件操作和文件处理需求。

```
1.  package utils;
2.
3.  import java.io.*;
4.
5.  public class FileUtils {
6.      public static long getFileSize(File f) {
7.          if (f.exists() && f.isDirectory()) {
8.              File[] files = f.listFiles();
9.              long size = 0;
10.             for (File file1 : files) {
```

```
                    if (file1.isFile()) {
                        size += file1.length();
                    } else {
                        size += getFileSize(file1);
                    }
                }
                return size;
            } else if (f.exists()) {
                long size = 0;
                size += f.length();
                return size;
            } else {
                return -1;  // 查询文件或文件夹有误
            }
        }

        public static String getProjectPath() {
            File f = new File("");
            String projectPath = "";
            try {
                projectPath = f.getCanonicalPath();
            } catch (Exception e) {
                System.out.println(e);
            }
            return projectPath;
        }

        public static String getFileContent(File file) throws IOException {
            String path = file.getPath();
            FileInputStream fis = new FileInputStream(path);
            InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
            BufferedReader br = new BufferedReader(isr);
            String fileContent = "";
            String line = null;
            while ((line = br.readLine()) != null) {
                fileContent += line;
            }
            return fileContent;
        }
    }
```

在 LuceneUtils.java 中定义了工具包。

LuceneUtils.java：

定义了一个 LuceneUtils 工具类，提供了一些与 Lucene 搜索引擎相关的方法，例如获取词项在文档中的出现频率、获取词项的文档频率、计算词项在文档中的得分等。这些方法都是静态方法，可以直接通过类名调用，而不需要创建 LuceneUtils 对象。

```
1.  package utils;
2.
3.  import org.apache.lucene.index.*;
4.  import org.apache.lucene.search.IndexSearcher;
5.  import org.apache.lucene.store.Directory;
6.  import org.apache.lucene.store.FSDirectory;
7.  import org.apache.lucene.util.BytesRef;
8.
9.  import java.io.File;
10. import java.io.IOException;
11. import java.nio.file.Paths;
12.
13. public class LuceneUtils {
14.
15.     private static LuceneUtils singleton;
16.     private static IndexReader reader;
17.     private static IndexSearcher searcher;
18.     private static long totalFileSize;
19.
20.     private LuceneUtils() {}
21.
22.     public static LuceneUtils getInstance() {
23.         if (singleton == null) {
24.             synchronized (LuceneUtils.class) {
25.                 if (singleton == null) {
26.                     singleton = new LuceneUtils();
27.                     try {
28.                         Directory directory = FSDirectory.
    open(Paths.get(FileUtils.getProjectPath() + "\\index"));
29.                         reader = DirectoryReader.open(dir
    ectory);
```

```
                            searcher = new IndexSearcher(reader);
                            totalFileSize = FileUtils.getFileSize(new File(FileUtils.getProjectPath() + "\\tdt3"));
                        } catch (Exception e) {
                            System.out.println(e);
                        }
                    }
                }
            }
            return singleton;
        }

        public static IndexSearcher getSearcher() {
            return searcher;
        }

        public static int getTermFreq(int docId, Term term) throws IOException {
            BytesRef termBytes = term.bytes();
            Terms terms = reader.getTermVector(docId, "text");
            if (terms == null) {
                return 0;
            }
            PostingsEnum postingsEnum = MultiTerms.getTermPostingsEnum(reader, "text", termBytes);
            TermsEnum termsEnum = terms.iterator();
            BytesRef termBytes1;
            while((termBytes1 = termsEnum.next()) != null){
                if (term.text().matches(termBytes1.utf8ToString())) {
                    PostingsEnum docPosEnum = termsEnum.postings(postingsEnum);
                    docPosEnum.nextDoc();
                    return docPosEnum.freq();
                }
            }
            return 0;
        }

        public static int getDocFreq(Term term) throws IOException {
```

```
65.           Terms terms = MultiTerms.getTerms(reader, "text")
    ;
66.           TermsEnum termsEnum = terms.iterator();
67.           BytesRef br;
68.           while ((br = termsEnum.next()) != null) {
69.               if (term.text().matches(br.utf8ToString())) {
70.                   return termsEnum.docFreq();
71.               }
72.           }
73.           return 0;
74.       }
75.
76.       public static int getDocNum() {
77.           return reader.maxDoc();
78.       }
79.
80.       public static float termScore(int docId, String field,
     String term, int df) throws IOException {
81.           Term term1 = new Term(field, term);
82.           float tf = getTermFreq(docId, term1);
83.           float idf = (float) Math.log((getDocNum() - df +
    0.5) / (df + 0.5));
84.           float K = (float) (0.5 + 1.5 * Integer.parseInt(s
    earcher.doc(docId).get("fileSize")) * getDocNum() / totalF
    ileSize);
85.           float R = 3 * tf / (tf + K);
86.           return idf * R;
87.       }
88.
89.       public static void main(String[] args) throws IOExcep
    tion {
90.           LuceneUtils luceneUtils = LuceneUtils.getInstance
    ();
91.           Term term = new Term("text", "hurricane");
92.           int df = getDocFreq(term);
93.           System.out.println(df);
94.           System.out.println(getDocNum());
95.       }
96.   }
```

在/utils/MyCollector.java 中，计算每个文档的得分。

MyCollector.java：

实现搜索结果的收集和排序。通过使用 MyCollector 类，可以方便地对搜索结果进行排序，并且可以根据需要选择返回前 10 个搜索结果。

```
1.  package utils;
2.
3.  import org.apache.lucene.index.Term;
4.  import org.apache.lucene.search.ScoreDoc;
5.  import org.apache.lucene.search.ScoreMode;
6.  import org.apache.lucene.search.SimpleCollector;
7.
8.  import java.io.IOException;
9.  import java.util.*;
10.
11. public class MyCollector extends SimpleCollector {
12.
13.     private List<ScoreDoc> scoreDocs = new ArrayList<Scor
    eDoc>();
14.     private List<String> terms;
15.     private List<Integer> df = new ArrayList<Integer>();
16.     private Map<Integer, Float> docs = new HashMap<Intege
    r, Float>();
17.     private String field;
18.     private LuceneUtils luceneUtils = LuceneUtils.getInst
    ance();
19.
20.     public MyCollector(QueryHandler queryHandler) {
21.         this.terms = queryHandler.getTerms();
22.         this.field = queryHandler.getField();
23.         for (String term : terms) {
24.             try {
25.                 this.df.add(LuceneUtils.getDocFreq(new Te
    rm(this.field, term)));
26.             } catch (IOException e) {
27.                 e.printStackTrace();
28.             }
29.         }
30.     }
31.
32.     @Override
33.     public void collect(int doc) throws IOException {
34.         if (docs.containsKey(doc)) {
35.             return;
36.         }
```

```
37.             float score = 0;
38.             for (int i = 0; i < this.terms.size(); i++) {
39.                 score += LuceneUtils.termScore(doc, this.fiel
   d, this.terms.get(i), this.df.get(i));
40.             }
41.             this.docs.put(doc, score);
42.         }
43.
44.         public List<ScoreDoc> getSortedScoreDocs() {
45.             setScoreDocs();
46.             return this.scoreDocs;
47.         }
48.
49.         public List<ScoreDoc> getSortedScoreDocs(int n) {
50.             setScoreDocs();
51.             System.out.println("TotalHits: " + this.scoreDocs.
   size());
52.             if (n <= this.scoreDocs.size()) {
53.                 return this.scoreDocs.subList(0, n);
54.             } else {
55.                 return this.scoreDocs;
56.             }
57.         }
58.
59.         private void setScoreDocs() {
60.             for (Integer i : docs.keySet()) {
61.                 this.scoreDocs.add(new ScoreDoc(i, docs.get(i)
   ));
62.             }
63.             Collections.sort(scoreDocs, new Comparator<ScoreD
   oc>() {
64.                 @Override
65.                 public int compare(ScoreDoc o1, ScoreDoc o2)
   {
66.                     if (o1.score < o2.score) {
67.                         return 1;
68.                     } else {
69.                         return -1;
70.                     }
71.                 }
72.             });
73.         }
74.
75.         @Override
```

```
76.     public ScoreMode scoreMode() {
77.         return ScoreMode.COMPLETE_NO_SCORES;
78.     }
79. }
```

在/utils/QueryHandler.java 中，分析文档，生成查询对象。

QueryHandler.java：

解析查询字符串并生成查询对象。通过使用 QueryHandler 类，可以方便地处理各种查询字符串，并生成对应的查询对象。

```
1.  package utils;
2.
3.  import org.apache.lucene.index.Term;
4.  import org.apache.lucene.queryparser.classic.ParseException;
5.  import org.apache.lucene.search.*;
6.
7.  import java.io.IOException;
8.  import java.util.ArrayList;
9.  import java.util.List;
10. import java.util.regex.Matcher;
11. import java.util.regex.Pattern;
12.
13. import static org.apache.lucene.search.BooleanClause.Occur.SHOULD;
14.
15. public class QueryHandler {
16.
17.     private List<List<String>> mustPhrases = new ArrayList<>();
18.     private List<List<String>> shouldPhrases = new ArrayList<>();
19.     private List<String> terms = new ArrayList<>();
20.     private String field;
21.
22.     public QueryHandler(String field, String query) throws IOException {
23.         this.field = field;
24.         Pattern p = Pattern.compile("\"(.*?)\"");
25.         Matcher m = p.matcher(query);
```

```
26.           while (m.find()) {
27.               List<String> phraseString = new ArrayList<>()
   ;
28.               String phrase = m.group().replace("\"", "").t
   rim().toLowerCase();
29.               String[] phrases = phrase.split("\\s+");
30.               for (String phrase1 : phrases) {
31.                   String[] phrase2 = phrase1.split("-");
32.                   for (String phrase3 : phrase2) {
33.                       phraseString.add(phrase3);
34.                   }
35.               }
36.               this.mustPhrases.add(phraseString);
37.           }
38.           query = m.replaceAll(" ").trim().toLowerCase();
39.           String[] terms = query.split("\\s+");
40.           for (String term : terms) {
41.               String[] t = term.split("-");
42.               if (t.length > 1) {
43.                   List<String> phraseString = new ArrayList
   <>();
44.                   for (String t1 : t) {
45.                       phraseString.add(t1);
46.                   }
47.                   this.shouldPhrases.add(phraseString);
48.               } else {
49.                   this.terms.add(term);
50.               }
51.           }
52.       }
53.
54.       public String getField() {
55.           return this.field;
56.       }
57.
58.       @Override
59.       public String toString() {
60.           String mp = "";
61.           for (List<String> phrase : this.mustPhrases) {
62.               for (String t : phrase) {
63.                   mp += t + " ";
64.               }
65.               mp += ",";
66.           }
```

```
67.           mp += "\n";
68.
69.           String sp = "";
70.           for (List<String> phrase : this.shouldPhrases) {
71.               for (String t : phrase) {
72.                   sp += t + " ";
73.               }
74.               sp += ",";
75.           }
76.           sp += "\n";
77.
78.
79.           String t = "";
80.           for (String term : this.terms) {
81.               t += term + ",";
82.           }
83.           return "Field: " + this.field + "\nMustPhrases: "
    + mp + "ShouldPhrases: " + sp + "Terms: " + t;
84.       }
85.
86.       public Query getBooleanQuery() {
87.           BooleanQuery.Builder booleanBuilder = new Boolean
  Query.Builder();
88.
89.           for (List<String> phrases : this.mustPhrases) {
90.               PhraseQuery.Builder phraseBuilder = new Phras
  eQuery.Builder();
91.               for (String term : phrases) {
92.                   phraseBuilder.add(new Term(this.field, te
  rm));
93.               }
94.               phraseBuilder.setSlop(0);
95.               booleanBuilder.add(phraseBuilder.build(), Boo
  leanClause.Occur.MUST);
96.           }
97.
98.           for (List<String> phrases : this.shouldPhrases) {
99.               PhraseQuery.Builder phraseBuilder = new Phras
  eQuery.Builder();
100.              for (String term : phrases) {
101.                  phraseBuilder.add(new Term("text", term))
  ;
102.              }
103.              phraseBuilder.setSlop(0);
```

```
104.             booleanBuilder.add(phraseBuilder.build(), SHO
    ULD);
105.         }
106.
107.         for (String term : this.terms) {
108.             booleanBuilder.add(new TermQuery(new Term("te
    xt", term)), SHOULD);
109.         }
110.
111.         return booleanBuilder.build();
112.     }
113.
114.     public List<String> getTerms() {
115.         List<String> result = new ArrayList<String>();
116.         for (String term : this.terms) {
117.             result.add(term);
118.         }
119.         for (List<String> phrases : this.mustPhrases) {
120.             for (String phrase : phrases) {
121.                 result.add(phrase);
122.             }
123.         }
124.         for (List<String> phrases : this.shouldPhrases) {
125.             for (String phrase : phrases) {
126.                 result.add(phrase);
127.             }
128.         }
129.         return result;
130.     }
131. }
```

在 engine 文件夹下需要写两个主要程序,将项目的逻辑结构组织起来:BuindIndex 用于创建索引，Search 用于实现搜索功能。

BuildIndex.java：

导入工具包。

```
1.  package engine;
2.
```

```
3.  import bean.DocumentContent;
4.  import org.apache.lucene.analysis.Analyzer;
5.  import org.apache.lucene.analysis.standard.StandardAnalyz
    er;
6.  import org.apache.lucene.document.*;
7.  import org.apache.lucene.index.IndexOptions;
8.  import org.apache.lucene.index.IndexWriter;
9.  import org.apache.lucene.index.IndexWriterConfig;
10. import org.apache.lucene.store.Directory;
11. import org.apache.lucene.store.FSDirectory;
12. import utils.FileUtils;
13.
14. import java.io.File;
15. import java.io.IOException;
16. import java.nio.file.Paths;
```

创建一个建立索引的函数。

使用流行的搜索引擎库 Apache Lucene 创建一个简单的索引。simpleIndex 方法创建了一个索引写入器，并通过一个目录来存储索引，一个分析器来解析和分词文本，以及一个索引写入器配置对象进行配置。

```
1.  public class BuildIndex {
2.      public static void simpleIndex() throws IOException {
3.          Directory directory = FSDirectory.open(Paths.get(
    FileUtils.getProjectPath() + "\\index"));
4.          Analyzer analyzer = new StandardAnalyzer();
5.          IndexWriterConfig config = new IndexWriterConfig(
    analyzer);
6.          IndexWriter indexWriter = new IndexWriter(directo
    ry, config);
```

创建一个文件对象 tdt3，它指向项目路径下的一个名为 tdt3 的已经解压的数据集。接着，获取 tdt3 数据集中的所有文件，并将它们存储在一个数组中。遍历一个包含多个文本文件的目录，读取每个文本文件的内容，并将其封装成一个 DocumentContent 对象，以便进行索引。

```
1.          File tdt3 = new File(FileUtils.getProjectPath() + "\
    \tdt3");
2.          File[] childDirs = tdt3.listFiles();
3.          for (int i = 0; i < childDirs.length; i++) {
```

```
                File[] txtList = childDirs[i].listFiles();
                for (File file : txtList) {
                    String fileName = file.getName();
                    String filePath = file.getPath();
                    long fileSize = FileUtils.getFileSize(file);
                    DocumentContent documentContent = new DocumentContent(FileUtils.getFileContent(file));
```

创建多个 Field 对象，用于存储文档的不同属性信息。每个 Field 对象都有一个名称、一个值和一个存储选项。

```
                    Field fileNameField = new TextField("fileName", fileName, Field.Store.YES);
                    Field filePathField = new StoredField("filePath", filePath);
                    Field fileSizeField = new StoredField("fileSize", fileSize);

                    Field docNOField = new StoredField("docNO", documentContent.getDocNO());
                    Field docTypeField = new StringField("docType", documentContent.getDocType(), Field.Store.YES);
                    Field txtTypeField = new StringField("txtType", documentContent.getTxtType(), Field.Store.YES);
```

创建一个 FieldType 对象 ft，并使用它来创建一个 Field 对象 textField，用于存储文档的文本内容。

```
                FieldType ft = new FieldType();
                     ft.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);// 存储
                    ft.setStored(false);
                    ft.setStoreTermVectors(true);
                    ft.setTokenized(true);
                    ft.setStoreTermVectorPositions(true);
                    ft.setStoreTermVectorOffsets(true);
                    Field textField = new Field("text", documentContent.getText(), ft);
```

将添加的域全部装载到 document 对象中。最后关闭写入索引对象。

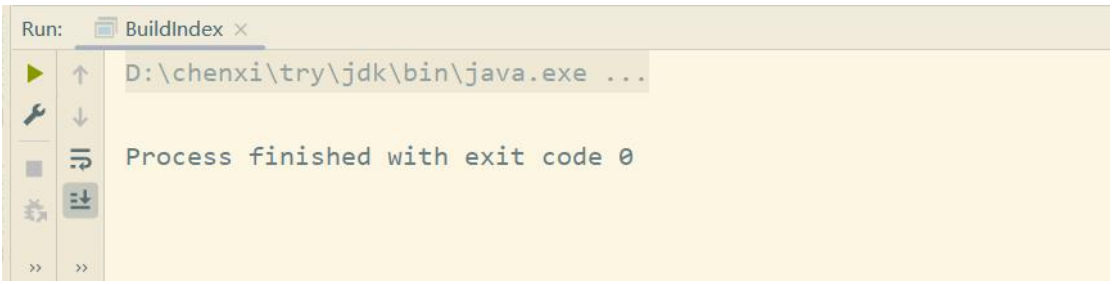
```
  Document document = new Document();
                    document.add(fileNameField);
                    document.add(filePathField);
```

```
                    document.add(fileSizeField);
                    document.add(docNOField);
                    document.add(docTypeField);
                    document.add(txtTypeField);
                    document.add(textField);
                    indexWriter.addDocument(document);
                }
            }
            indexWriter.close();
        }
```

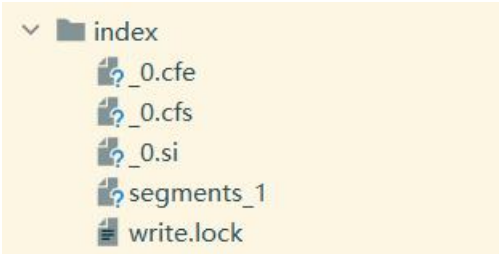
执行上面的创建索引函数。

```
    public static void main(String []args)throws Exception
    {
            simpleIndex();
        }
    }
```

运行代码可以成功。



Index 文件夹下查看到索引已经生成。



Search.java：

导入工具包。

```
package engine;

import bean.DocumentContent;
```

```
import org.apache.lucene.document.Document;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import utils.FileUtils;
import utils.LuceneUtils;
import utils.MyCollector;
import utils.QueryHandler;

import java.io.File;
import java.io.IOException;
import java.util.List;
import java.util.Scanner;
```

获取用户输入的搜索关键字，并将其存储在字符串变量 keywords 中，以便后续进行搜索操作。通过使用 Scanner 类可以方便地从标准输入中读取用户输入，而不需要手动处理输入流。

```
public class Search {
    public static void main(String[] args) throws IOException {
        Scanner in = new Scanner(System.in);
        System.out.println("Please input your keyword:");
        String keywords = in.nextLine();
        in.close();
```

使用 Lucene 搜索引擎来执行查询操作，并将查询结果按相关性排序后返回前 10 条搜索结果。在实际应用中，可以根据需求设置不同的查询条件和查询参数，以便更好地支持搜索和查询操作。

```
        LuceneUtils.getInstance();
        QueryHandler queryHandler = new QueryHandler("text", keywords);
        Query q = queryHandler.getBooleanQuery();
        MyCollector collector = new MyCollector(queryHandler);
        LuceneUtils.getSearcher().search(q, collector);
        List<ScoreDoc> scoreDocs = collector.getSortedScoreDocs(10);
```

遍历搜索结果，并输出每个搜索结果的摘要信息。通过使用 Document 和 DocumentContent 对象可以方便地获取文档的各个属性信息，并生成摘要信息以便展示在搜索结果列表中。

```
 for (int i = 0; i < scoreDocs.size(); i++) {
            Document document = LuceneUtils.getSearcher().doc(scoreDocs.get(i).doc);
            String filePath = document.get("filePath");
            File file = new File(filePath);
            DocumentContent documentContent = new DocumentContent(FileUtils.getFileContent(file));
            String abstractText;
            if (documentContent.getText().length() < 400) {
                abstractText = documentContent.getText();
            } else {
                abstractText = documentContent.getText().substring(0, 400) + "....";
            }
            System.out.println((i + 1) + "\t[" + scoreDocs.get(i).score + "]\t" + documentContent.getDocNO() + "\n" + abstractText + "\n");
        }

    }

}
```

可以成功运行出来搜索程序。

四、实验结果

1. 搜索 hurricane

```
D:\chenxi\try\jdk\bin\java.exe ...
Please input your keyword:
hurricane
TotalHits: 33
1   [11.475757] CNN19981008.1130.0447
Hurricane georges is now the third costliest hurricane on record. Actuaries estimate georges caused $2.5 billion in insured losses throughout the

2   [11.187919] NYT19981005.0056
In the days before Hurricane Georges struck the Gulf Coast last week, storm forecasters faced one of their most trying responsibilities: predictir

3   [11.187919] NYT19981003.0052
In the days before Hurricane Georges struck the Gulf Coast last week, storm forecasters faced one of their most trying responsibilities: predictir

4   [8.333079]  APW19981004.0574
At a shelter in Puerto Rico last week, Hillary Rodham Clinton spoke of a dlrs 39 million dollar U.S. grant to rebuild hurricane-ravaged homes. But

5   [8.22878]   CNN19981001.0130.0919
And the remnants of hurricane georges are lingering over the south bringing heavy rain in some areas. Thousands of people still are without power.

6   [8.143787]  NYT19981009.0279
NATIONAL FORECAST Dull, damp and rather dismal weather will prevail across the Northeast on Saturday as low pressure drifts east along the souther

7   [8.067708]  CNN19981008.1130.0948
And insurers are getting the Bill from hurricane georges. Early estimates say the storm is the third-costliest ever, causing $2.5 billion in insur

8   [8.058244]  PRI19981007.2000.0327
The insurance industry says Hurricane Georges cost at least $2.5 million in insured damage to the United States and its island territories. That w

9   [7.921946]  CNN19981008.1130.0000
Hurricane georges hits the history books. We'll tell you why. And how to avoid a common mistake made by airline travelers. "Headline News." I'm Ca

10  [7.858507]  VOA19981005.1800.3091
This appears to be a tough time for American Property & Casualty Insurance Companies. A survey of industry analysts says most will report lower pr


Process finished with exit code 0
```

2. 搜索 mitch george

```
D:\chenxi\try\jdk\bin\java.exe ...
Please input your keyword:
mitch george
TotalHits: 99
1   [8.072929]  VOA19981002.1800.3173
Pianist and composer George Winston is recognized world-wide as the pioneer of the new-age music movement. His Wyndham Hill record label set the t

2   [7.5628133] NYT19981008.0480
What is wrong with Coach George? At the very time the New York Yankees needed a football-style, two-a-day drill on the fine art of throwing the ba

3   [7.0024977] VOA19981002.1800.3484
Boxing veteran George Foreman has found a way to celebrate his 50th birthday. He's going back into the ring and he'll be fighting Larry Holmes, wh

4   [6.7749186] NYT19981006.0214
Anyone who subscribes to the sentimental fallacy that great artists are nicer people than the rest of us (the reasoning goes that because they sup

5   [5.7928734] MNB19981002.2100.3508
A few program notes -- this evening on MSNBC, "weekend magazine" with stone Phillips. And tonight on CNBC, our broadcast "in profile" takes on and

6   [5.6435194] NBC19981002.1830.1603
He started out in Texas as a singing cowboy. And when he died today at the age of 91, Gene autry was worth hundreds of millions of dollars. He had

7   [5.610644]  APW19981007.0849
Being the Masters and British Open champion, Mark O'Meara appeared the ideal player to partner golf-loving Prince Andrew in the pro-am before the

8   [5.130806]  CNN19981009.1130.1045
Bid farewell to the American eagle. Bid hello to Georgia peaches and if beloved oak tree will adorn newly minted quarters next year. They're just

9   [5.1067295] NBC19981005.1830.0525
While Capitol Hill debated impeachment, President Clinton was focused on the economy here at home and overseas. The president met tonight with top

10  [4.67633]   NYT19981004.0200
The New York Times said in an editorial on Monday, Oct. 5: In the final days of a dismal session, Trent Lott appears to have buried the best chanc


Process finished with exit code 0
```

3. 搜索 bill clinton israel

```
TotalHits: 738
1   [12.418948] APW19981001.1161
JERUSALEM (AP) _ U.S. President Bill Clinton rejected a bid by Prime Minister Benjamin Netanyahu raised during their meeting this week to free Jonat

2   [12.10818]  APW19981001.0571
Prime Minister Benjamin Netanyahu said Thursday that he discussed the fate of Jonathan Pollard with U.S. President Bill Clinton this week, but that

3   [11.7862215]    APW19981002.0564
The United States is trying to coax a pledge from Israel not to expand Jewish settlements, in exchange for assurances from the Palestinians that the

4   [11.594105] APW19981001.1184
During their meeting this week, U.S. President Bill Clinton rejected a bid by Prime Minister Benjamin Netanyahu to free Israeli spy Jonathan Pollard

5   [11.38991]  APW19981001.0262
An Israeli newspaper reported Thursday that U.S. President Bill Clinton has agreed to release Jonathan Pollard, an American serving a life term for

6   [10.709862] APW19981002.1076
Israel on Friday sealed its borders with the West Bank and Gaza Strip indefinitely following warnings by the defense minister that the Islamic milit

7   [10.61038]  APW19981008.0222
Cabinet hard-liners on Thursday threatened to topple Prime Minister Benjamin Netanyahu if he makes concessions to the Palestinians at a Mideast summ

8   [10.5123825]    APW19981009.0545
Thousands of Palestinians, some shooting in the air from automatic weapons and chanting ``revenge,'' marched Friday in the funeral procession of a 2

9   [10.510975] APW19981001.0542
Palestinian leader Yasser Arafat said Thursday that this week's top-level talks in Washington have produced progress on a West Bank troop pullback.

10  [9.576195]  APW19981009.0300
Thousands of Palestinians, some brandishing automatic weapons and chanting ``revenge,'' marched Friday in the funeral procession of a 20-year-old ma
```

4. 搜索 “newt gingrich” down

```
TotalHits: 21
1   [17.84813]  NYT19981007.0464
A week after the White House and congressional Democrats disavowed his ``war'' on Speaker Newt Gingrich, James Carville, President Clinton's former

2   [15.516676] CNN19981006.0130.0334
President Clinton is warning that time is running short to avoid a federal government shutdown. He is urging Congress to pass several spending bill

3   [13.188275] CNN19981003.0130.0260
The house Judiciary committee will meet on Monday to vote on beginning a formal impeachment inquiry. It plans to ask Ken Starr whether he has found

4   [12.878454] PRI19981008.2000.0071
House Speaker Newt Gingrich announced the tally this afternoon as the House voted to launch the third presidential impeachment inquiry in U.S. hist

5   [12.201279] CNN19981001.0130.0128
More documents from the Ken Starr report could be released to the public Friday. Sources familiar with the material say it contains no bombshells f

6   [10.210049] CNN19981008.2130.0041
It is original the third time in history lawmakers have take thn step. Today, House of Representatives voted to begin a far-reaching impeaching inq

7   [8.862707]  CNN19981003.1130.1494
a california publishing firm says it's making public some private phone conversations of people talking sex, drugs and much more. garrick utley rep

8   [8.761782]  CNN19981003.0130.1487
In the era of cell phones, hidden cameras and the world wide web, some say peeping toms have moved out of the bushes and into the mainstream. Garri

9   [7.940074]  NYT19981008.0412
Republican congressional leaders presented the Clinton administration Thursday with a list of lending conditions that they want imposed on the Inte

10  [7.8114033] NYT19981003.0097
Not only have Dick Gephardt and Bill Clinton never been close, but sometimes they have acted as if they belonged to separate political parties. So


Process finished with exit code 0
```

5. 搜索 nba strike closed-door

```
D:\chenxi\try\jdk\bin\java.exe ...
Please input your keyword:
nba strike closed-door
TotalHits: 139
1   [13.039276] NYT19981006.0396
Patrick Ewing did not want to sound like a striking longshoreman demanding health benefits. ``I'm not going to try and put it in dollars and cents.

2   [12.878238] APW19981001.1151
While Michael Jordan and Magic Johnson have engaged the public in ways other than basketball, Larry Bird has done it in just one way: with his game

3   [12.542613] CNN19981006.1130.1337
The NBA lockout forced the league to cancel the entire preseason, Monday. The NBA is scared the player lockout will eat into the regular season.

4   [11.201659] VOA19981005.1800.1274
The National Basketball Association has wiped out the remainder of its exhibition schedule because of stalled labor negotiations with players. NBA

5   [9.918757]  VOA19981006.1700.3100
The head of Kenya's national teachers union says he's willing to end the two day old strike if the government is willing to negotiate a settlement

6   [9.876673]  CNN19981008.1600.1350
NBA news, the players and the league resume collective bargains negotiations for the first time after two months today. No progress made today. The

7   [9.787881]  CNN19981005.2130.1325
the nba is another step closer to losing regular season games because of labor disputes for the first time in history. the league announced today t

8   [9.666967]  CNN19981005.1600.1233
NBA training camps were supposed to open tomorrow, but because of the lockout, they have been postponed indefinitely. And now, the league has cance

9   [9.122181]  APW19981003.0172
ATHENS, Greece (AP) - Doctors at public hospitals in Athens returned to work Saturday after their union suspended a lengthy strike which had seriou

10  [9.122181]  APW19981003.0167
Athens, Greece (AP) - Doctors at public hospitals in Athens returned to work Saturday after their union suspended a lengthy strike which had seriou
```