Java 实验报告

徐云凯 1713667

目录

1. 问题描述

实现邮件检索系统，被检索数据为安然公司 150 位用户 50 万封电子邮件，语言为英文。搜索引擎采用 Apache 的开源全文检索引擎 Lucene。检索时按照收件人、发件人、标题、内容等进行邮件检索。并实现交互系统。

2. 实验环境介绍

开发使用的 IDE 为 IDEA，Java 版本为 11.0.2。Lucene 版本 8.2.0。Log4j 版本 2.12.1。web 界面使用 jsp 动态网页，运行在 Tomcat 服务器中，Tomcat 版本为 7.0.92。

3. 实现思路

邮件索引系统中的索引建立为一次性操作，一次索引即可反复使用。所以将该项目分为索引建立与查询两个部分并分别实现。其中索引建立部分使用 Console 界面实现。查询部分使用 web 界面实现，运行时只需要在服务器端运行索引程序，或者直接将已经生成的索引放置在指定文件夹下即可。

4. 具体实现

4.1. 建立索引

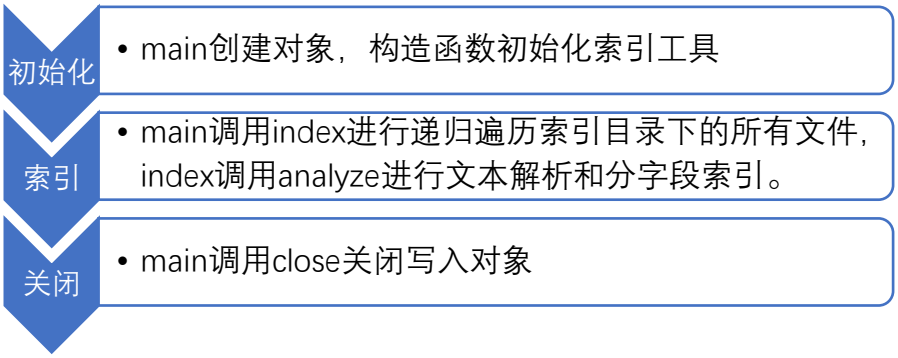
该子项目全部代码运行于 LuceneIndex 类中，main 函数入口点负责初始化类，构造函数完成构建索引器、分析器与索引写入对象等操作后，由主函数调用 LuceneIndex 类的 index()方法开始运行索引，index 方法会递归调用自身完成对待索引项目的遍历索引，期间有完善的错误处理与日志机制保证在较大数据集上运行时不会中途退出。最后由主函数调用 close()方法关闭索引写入对象结束索引过程。

4.1.0. LuceneIndex 类的成员变量与成员方法

logger 对象用于进行日志写入与信息输出。
writer 对象为该程序中所有文件索引的共用索引写入对象。
numIndexTotal 用于统计索引的文件总数。

main 函数为入口点，负责启动与关闭索引，并计时。
LuceneIndex 用于初始化索引器，分析器等对象。
index 方法用于递归调用以递归遍历所有目录下的可读取文件进行索引。
analyze 方法为文件解析器，解析文件结构分割进入 field 进行索引。
close 方法用于关闭 writer。

```java
public class LuceneIndex {
    private static Logger logger = LogManager.getLogger(LuceneIndex.class);
    private IndexWriter writer;
    private int numIndexedTotal = 0;

    /*** ...*/
    public static void main(String[] args) {...}

    /*** ...*/
    public LuceneIndex(String indexDir) throws IOException {...}

    /*** ...*/
    public void index(File data){...}

    /*** ...*/
    private Document analyze(File f) throws IOException {...}

    /*** ...*/
    public void close() throws IOException {...}
}
```



4.1.1. main()函数，程序入口点

在索引开始与结束时计时，并在最后利用 logger 输出耗时与索引文件总量。
在所有可能出现异常的位置使用 try-catch 语句环绕并处理对应异常，利用 logger 进行输出。

```java
public static void main(String[] args) {

    String indexDir = "D:\\NKU\\CS\\Java\\HomeWork\\LuceneProject\\index";
    String dataDir = "D:\\NKU\\CS\\Java\\HomeWork\\maildir";
    File data = new File(dataDir);

    LuceneIndex indexer = null;
    //索引开始时间
    long start = System.currentTimeMillis();
    try {
        indexer = new LuceneIndex(indexDir);
        indexer.index(data);
        try {
            indexer.close();
        } catch (IOException e) {
            logger.error( s: "fail to close IndexWriter");
        }
    }
    catch (IOException e) {
        System.out.println("找不到指定的索引文件夹");
    }
    //索引结束时间
    long end = System.currentTimeMillis();
    logger.info( s: "index " + (indexer != null ? indexer.numIndexedTotal : 0)
            + " files, using" + (end - start)/1000 + " secs");
}
```

4.1.2. LuceneIndex 构造函数

创建一个磁盘索引器 FSDirectory，一个标准分析器并使用 CREATE 模式新建索引。这里不使用内存索引器的原因的内存索引器虽然更加高效，但是在 Lucene 8.2.0 版本中被弃用，且即便不使用内存索引器，索引也可以在半小时左右完成。

```
public LuceneIndex(String indexDir) throws IOException {
    Directory directory = FSDirectory.open(Paths.get(indexDir));
    Analyzer analyzer = new StandardAnalyzer();
    IndexWriterConfig iwConfig = new IndexWriterConfig(analyzer);
    iwConfig.setOpenMode(IndexWriterConfig.OpenMode.CREATE);
    writer = new IndexWriter(directory, iwConfig);
}
```

4.1.3. index 方法

递归遍历完成索引。

```
public void index(File data){

    File[] subFolders = data.listFiles();

    for(File file : subFolders){
        if(file.isDirectory()){
            index(file);
        }
        else{
            int numIndexedRound = writer.numRamDocs();
            logger.debug( s: "index file:" + file.toString());
            try {
                Document doc = analyze(file);
                try {
                    writer.addDocument(doc);
                }
                catch (IOException e) {
                    logger.error( s: "IOException in addDocument() for file:" + file.toString());
                }
            }
            catch (IOException e) {
                logger.error( s: "IOException in read file:" + file.toString());
            }
            finally{
                if(numIndexedRound > writer.numRamDocs()){
                    logger.info( s: "this round " + numIndexedRound + " files have been indexed");
                    numIndexedTotal += numIndexedRound;
                    numIndexedRound = writer.numRamDocs();
                }
            }
        }
    }
}
```

对目录进行递归遍历

调用analyze解析文本

向磁盘写入索引

writer更新统计量时计数并输出

4.1.4. analyze 方法

该方法会尝试解析文本中的标题，发件时间，发件人，收件人，正文，文件名与文件路径等信息并添加到字段中进行索引。

```
private Document analyze(File f) throws IOException {
    byte[] file = new byte[(int) f.length()];
    FileInputStream inputStream = new FileInputStream(f);
    inputStream.read(file);
    inputStream.close();
    String mail = new String(file);
```

利用 FileInputStream 进行文件读取，因为是英文，不考虑文件编码问题。

```
    // 标题
    int subject_begin = mail.indexOf("Subject:") + 8;
    int subject_end = mail.indexOf( str: "\r\n", subject_begin);
    String subject_string = mail.substring(subject_begin, subject_end);

    // 发件时间
    int data_begin = mail.indexOf("Date::") + 8;
    int data_end = mail.indexOf( str: "\r\n", data_begin);
    String data_string = mail.substring(data_begin, data_end);

    // 发信人
    int from_begin = mail.indexOf("From:") + 5;
    int from_end = mail.indexOf( str: "\r\n", from_begin);
    String from_string = mail.substring(from_begin, from_end);

    // 收件人
    int rec_begin = mail.indexOf("To:") + 5;
    int rec_end = mail.indexOf( str: "\r\n", rec_begin);
    String rec_string = mail.substring(rec_begin, rec_end);

    // 正文
    int body_begin = mail.indexOf("\r\n\r\n") + 4;
    String body_string = mail.substring(body_begin);
```

使用 subString 方法进行截取，通过识别关键字进行截取。
最后使用 doc.add()将字符串添加到对应的字段中。

4.1.5. close 方法

```
public void close() throws IOException {
    writer.close();
}
```

关闭写入对象。

4.1.6. 日志与运行效率

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{yy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
    </Console>
    <File name="IndexInfo" fileName="log/LuceneIndex.log" immediateFlush="false" append="true">
      <PatternLayout pattern="%d{yy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
    </File>
    <File name="IndexWarning" fileName="log/LuceneIndexWarning.log" immediateFlush="false" append="true">
      <PatternLayout pattern="%d{yy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
    </File>
  </Appenders>
  <Loggers>
    <Root level="all">
      <AppenderRef ref="Console" level="info"/>
      <AppenderRef ref="IndexInfo" level="debug"/>
      <AppenderRef ref="IndexWarning" level="warn"/>
    </Root>
  </Loggers>
</Configuration>
```

Log4j 被配置为将日志输出为三部分，info 级别以上的被输出到控制台。Debug 级别以上的输出到 IndexInfo.log 文件中，Warn 级别以上的单独输出到 IndexWarning.log 中。

经过实际索引运行，索引全部 517k 封邮件耗时约半小时，期间 cpu 占用低于 5%，磁盘占用低于 1%。该程序的性能瓶颈为磁盘存取的状态切换开销。通过多线程无法获得优化，但是可以通过减少磁盘写索引次数获得一定程度的优化。

4.2.在线查询

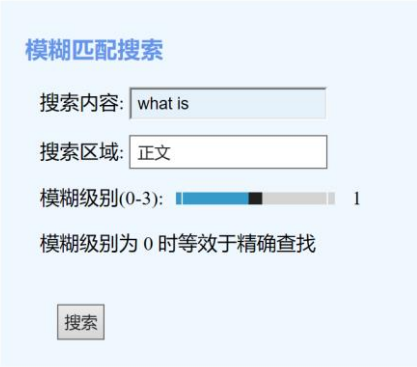
查询网页由两部分组成，index.jsp 和 search.jsp。
index.jsp 负责处理前端显示并与用户交互，index.jsp 通过 get 方法把查询参数传输给 search.jsp，search.jsp 负责处理并完成查询。

3.2.1. index.jsp

该页面构成主要基于 html5，并通过 css 与 javascript 美化了页面。界面支持 4 中检索方式，精确匹配、通配符匹配、前缀匹配与模糊匹配搜索。前三种在传参时只发送搜索内容和字段信息并附带搜索类型。最后一种额外附加一个模糊级别信息。

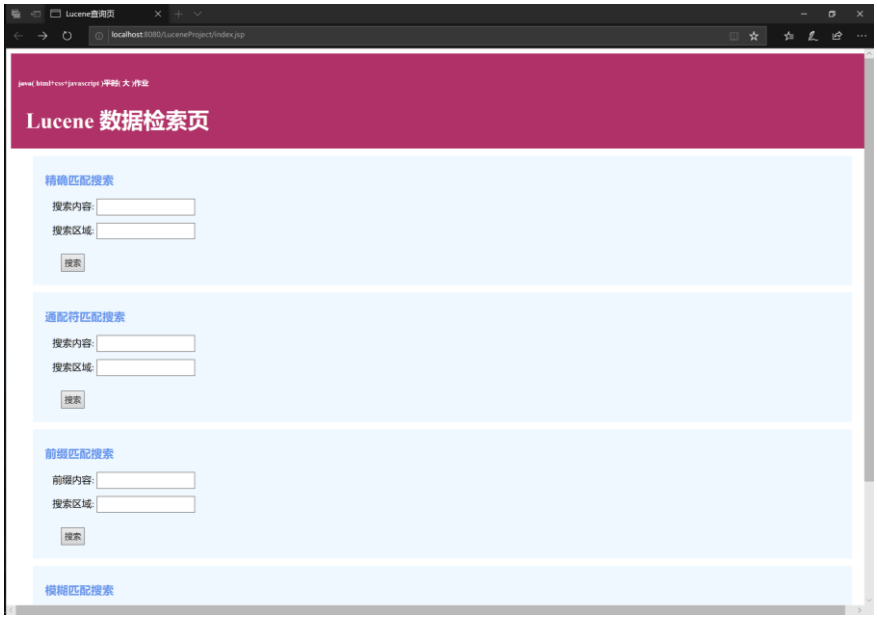
界面中，搜索字段部分提供了中文的下拉列表方便输入。

另外，在模糊输入部分提供了更加用户友好的界面，使用一个可以滑动的滑块提供模糊级别的输入，并且利用 html5 特性进行实时显示。




```
<div class="part">
    <h3>模糊匹配搜索</h3>
    <form action="search.jsp" method="GET" oninput = "showLevel.value = parseInt(level.value)">
        <input type="hidden" name="type" value="3">
        <div class = "search">
            <div class = "select">
                搜索内容: <input type="text" name="query"><br>
            </div>
            <div class="select">
                搜索区域: <input list="field" name="field"><br>
            </div>
            <div class="select">
                模糊级别(0-3): <input id="rangeFuzzyLevel" type="range" min="0" max="2" value="0" name="level">
                <output name="showLevel">0</output><br>
                <p>模糊级别为 0 时等效于精确查找</p>
            </div>
            <div class="submit">
                <input type="submit" value="搜索" />
            </div>
        </div>
    </form>
</div>
```

利用 form 的 oninput 属性实现滑块值实时显示



输入页面整体效果

3.2.2. search.jsp

首先进行参数解析，同时完成异常参数检测并进行处理。

```
<%
    // 将传入的参数替换为内部参数
    Map<String, String> fi= new HashMap<String, String>(){{
        put("全部位置","contents");
        put("日期","data");
        put("正文","body");
        put("发件人","from");
        put("收件人","to");
        put("主题","subject");
        put("文件名","fileName");
        put("文件位置","fullPath");
    }};
    int type = 0;   // 查询类型
    if(request.getParameter("type")!=null)
        type = Integer.parseInt(request.getParameter("type"));
    int fuzzyLevel = 0;  // 模糊查询所需要的额外参数
    if(type == 2)
        fuzzyLevel = Integer.parseInt(request.getParameter("level"));
    String errorInfo = "url参数错误";  // 初始化参数异常时的提醒信息
    String q = request.getParameter("query");
    String ori = request.getParameter("field");
    String f0 = null;
    if(ori!=null)
        f0 = new String(ori.getBytes("ISO8859-1"), StandardCharsets.UTF_8); //  解码传入的中文参数
    String f = fi.get(f0);
```

在将 get 方法传递的参数进行解码和检测异常之后，当参数无误则进入服务器后台开始搜索。

```
if(q != null && f!=null){
    String indexDir = "D:\\NKU\\CS\\Java\\HomeWork\\LuceneProject\\index\\";
    Directory dir = FSDirectory.open(Paths.get(indexDir));
    IndexReader reader = DirectoryReader.open(dir);
    IndexSearcher is = new IndexSearcher(reader);
    Analyzer analyzer = new StandardAnalyzer();

    Query query = null;
    switch (type){
        case 1:{
            query=new WildcardQuery(new Term(f,q));
            break;
        }
        case 2:{
            query = new PrefixQuery(new Term(f, q));
            break;
        }
        case 3:{
            query = new FuzzyQuery(new Term(f,q), fuzzyLevel);
            break;
        }
        default:{
            QueryParser parser = new QueryParser(f, analyzer);
            try {
                query = parser.parse(q);
            }
            catch (ParseException e) {
                errorInfo = "服务器端索引器创建异常";
            }
        }
    }
```

初始化各个搜索对象，并根据传入的搜索类型实例化不同的 query 对象，以实现不同的查找方式。

```
long start = System.currentTimeMillis();    // 搜索开始

TopDocs hits = null;
try {
    hits = is.search(query, 30);
}
catch (IOException e) {
    errorInfo = "服务器端搜索异常";
}

QueryScorer scorer = new QueryScorer(query);
Fragmenter fragmenter = new SimpleSpanFragmenter(scorer, 100);
SimpleHTMLFormatter simpleHTMLFormatter = new SimpleHTMLFormatter("<font color='red'>","</font>");
Highlighter highlighter = new Highlighter(simpleHTMLFormatter, scorer);
highlighter.setTextFragmenter(fragmenter);

long end = System.currentTimeMillis();  // 搜索结束
```

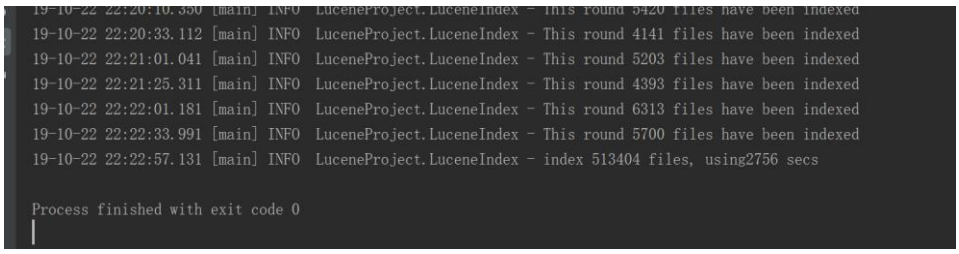
使用 search 进行搜索，将结果利用 highlighter 类提供的方法进行优化处理后输出。

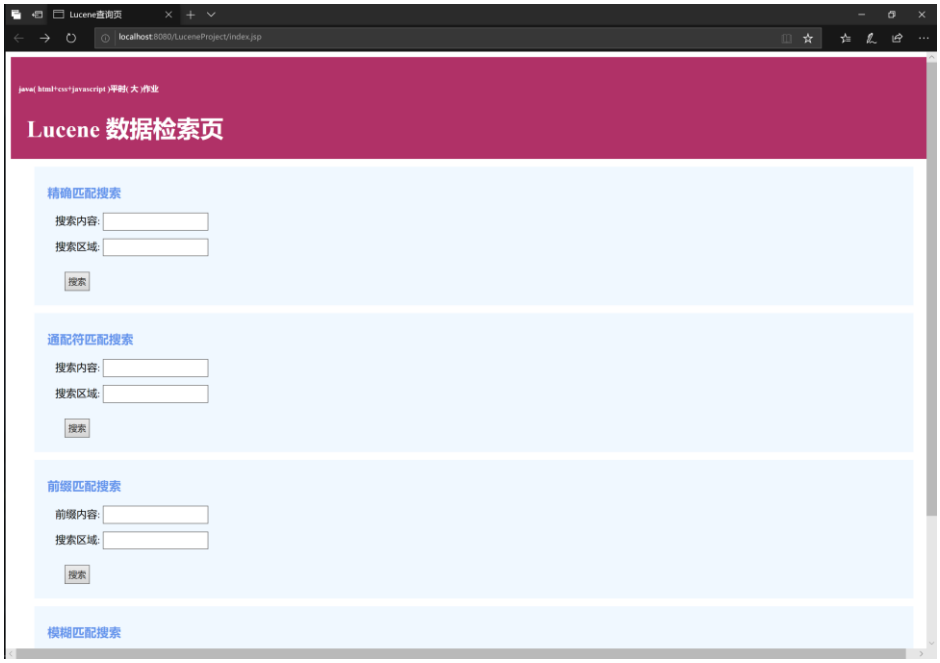
```
%>a
<h4><%= ("在" + f0 + "中查找关键词 “" + q + "” 查找到总计 " + hits.totalHits.value + " 个结果，耗时" + (end - start) + "毫秒" ) %></h4>
<%
        for (ScoreDoc scoreDoc : hits.scoreDocs) {
            Document doc = null;
            try {
                doc = is.doc(scoreDoc.doc);
            }
            catch (IOException e) {
                errorInfo = "服务器端搜索结果解析异常";
            }
            String content = doc.get(f);
            String highlighter_string = "";
            if(content!=null){
                try {
                    TokenStream tokenStream = analyzer.tokenStream(f, content);
                    try {
                        highlighter_string = highlighter.getBestFragment(tokenStream, content);
                    }
                    catch (InvalidTokenOffsetsException e) {
                        errorInfo = "服务器端搜索结果编码解析异常";
                    }
                } catch (IOException e) {
                    errorInfo = "服务器端TokenStream初始化异常";
                }
%>
<div class="result">
    <p class="path">文件位置：<ins>.\<%=doc.get("fullPath").substring(doc.get("fullPath").indexOf("maildir\\") + 8)%></ins> </p>
    <p class="content">检索到的文本：  <%=highlighter_string%></p>
</div>
<%
            }
        }
        try {
            reader.close();
        }
        catch (IOException e) {
            errorInfo = "服务器端读取器关闭失败";
        }
    }
    else{
```

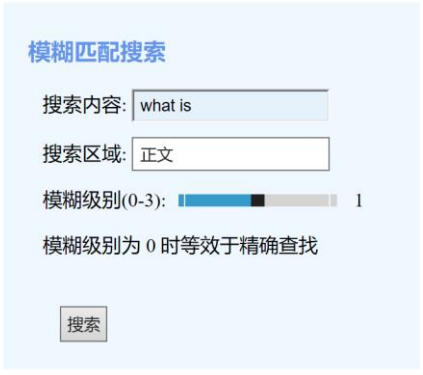
```
    else{
%>
<p> <%=errorInfo%> </p>
<%
    }
%>

</body>
</html>
```

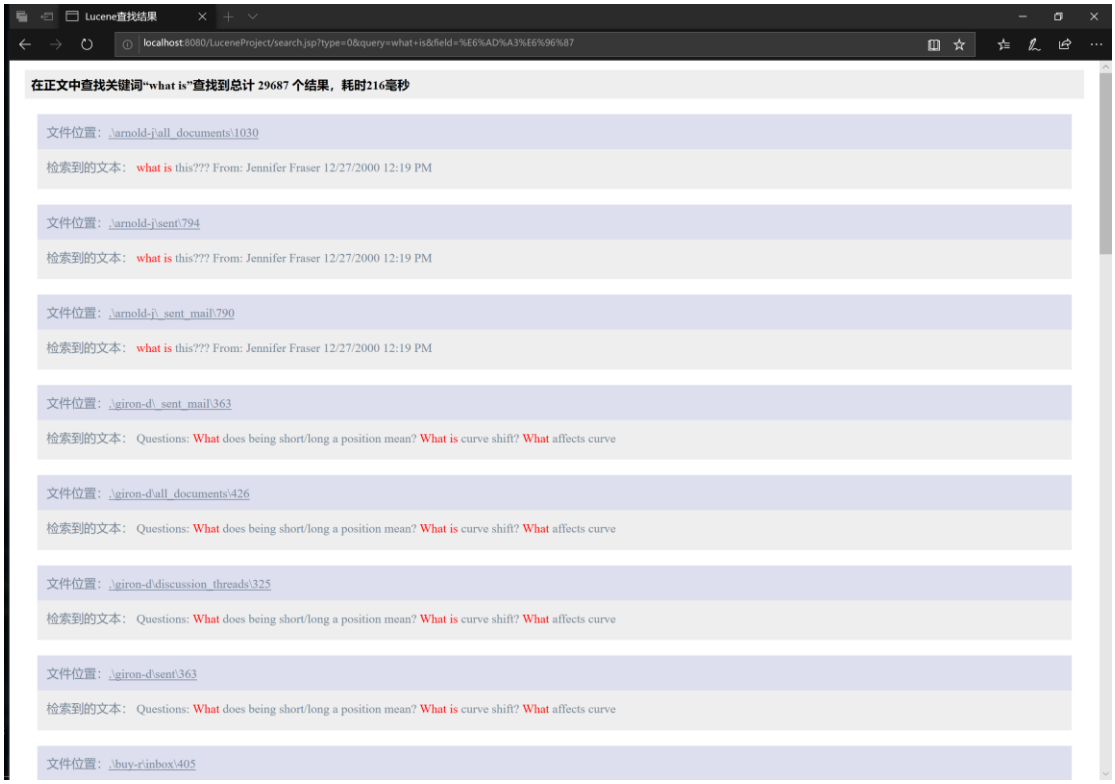
当检测出参数有异常时，则输出对应的错误信息，跳过服务器端的后台搜索部分。

5. 结果展示








Lucene查找结果
localhost:8080/LuceneProject/search.jsp?type=0&query=what+is&field=%E6%AD%A3%E6%96%87
在正文中查找关键词"what is"查找到总计 29687 个结果，耗时216毫秒
文件位置：.\arnold-j\all_documents\1030
检索到的文本： what is this??? From: Jennifer Fraser 12/27/2000 12:19 PM
文件位置：.\arnold-j\sent\794
检索到的文本： what is this??? From: Jennifer Fraser 12/27/2000 12:19 PM
文件位置：.\arnold-j_sent_mail\790
检索到的文本： what is this??? From: Jennifer Fraser 12/27/2000 12:19 PM
文件位置：.\giron-d_sent_mail\363
检索到的文本： Questions: What does being short/long a position mean? What is curve shift? What affects curve
文件位置：.\giron-d\all_documents\426
检索到的文本： Questions: What does being short/long a position mean? What is curve shift? What affects curve
文件位置：.\giron-d\discussion_threads\325
检索到的文本： Questions: What does being short/long a position mean? What is curve shift? What affects curve
文件位置：.\giron-d\sent\363
检索到的文本： Questions: What does being short/long a position mean? What is curve shift? What affects curve
文件位置：.\buy-r\inbox\405