“雷同报告检测系统”项目总结报告

摘要

雷同报告检测系统（以下简称本系统），是一个面向教育领域的大规模文档比较及雷同判定系统，能够高效的在指定的文档集以及文档与互联网资源之间发现可能雷同的文档。本系统通过计算文档相似度，从两个方面判断文档是否可能雷同：一是文档之间的比较，针对每个文档提取特征指纹，利用 Lucene 建立索引，计算相似度，找出同学之间相互抄袭的雷同文档；二是利用搜索引擎提供的 Search API，通过互联网检索，找出同学抄袭网络的雷同文档。从而帮助教师有效地发现可能雷同的报告及论文，减少教师人工判定的工作量，提高教学质量。本系统主要使用 Java 开发，移植性好，便于教师轻松的部署；同时也将利用 Web Service，为不具备部署条件的教师提供在线服务，实用性高；此外，本系统并不要求强大的计算性能，在普通服务器上就可以快速稳定的运行，成本低，利于推广。

关键词 雷同检测；特征指纹；Lucene；Web Service

目录

1. 项目简介

随着互联网的普及，学生作业的提交和批改方式都发生了很大的变化。一方面作业的电子化能够节省学生时间，方便教师批改；另一方面网络资源方便易得，电子作业易于复制的特性也给学生的抄袭带来了极大的方便。因此，如何通过技术手段自动有效的判断雷同，就成为目前教学工作的难点。

文本的雷同和抄袭问题已经引起了学术界的高度重视，2009 年举办的第一届抄袭检测国际竞赛(The 1st International Competition on Plagiarism Detection)[1] ，目的就是为了评测各种雷同检测算法在处理大规模数据时的性能。

斯坦福大学的 Moss(Measure Of Software Similarity)[2] 系统是一个简单易用的编程语言雷同检测系统。它主要用于计算 C、C++、Java、Pascal、Ada、ML、Lisp 等一些形式化语言所编写的程序之间的相似性。哈工大计算机学院已经成功的运用 Moss 在一些程序设计类课程[3]中检测学生程序之间的雷同。然而 Moss 系统只适用于有一定文法的形式语言，对于没有特定结构的自然语言不适用，也就是不适用于报告和大作业的雷同检测。

香港理工大学 Si 等人建立的 CHECK 系统[4]，采用统计关键词的方法来度量文本之间的相似性，其时间复杂性为 $O(n^2)$，对于报告数量较多的情况，其效率较低，同时也较难用于寻找互联网上的抄袭情况。小木虫网站上有关于论文是否有抄袭嫌疑的在线检测。该网站检测抄袭的算法保密，不是针对学生电子作业群体，而学生也不可能主动将自己抄袭的报告提交给该系统，老师将收到的作业全部提交则既费时也不现实，而且它还不能判断学生作业之间的雷同情况。

1993 年，Arizona 大学的 Manber 提出了一个 sif 工具[5]，用于在大规模文件系统中查找内容相似的文件。sif 工具并未明确的提出要进行文本复制检测，但是，sif 工具提出了“近似指纹（approximate fingerprints）”的概念，就是基于字符串匹配的方法来度量文件之间的相似性。这个思想被很多后来的文本复制检测机制 COPS（copy protection system）与相应的算法所采用。1995 年，Stanford 大学的 Brin 和 Garcia-Molina 等人在“数字图书馆”工程中首次提出了文本复制检测机制系统与相应的算法。Garcia-Molina 和 Shivakumar 等人又提出了 SCAM[6]（Stanford copy analysis method）原型改进 COPS 系统，用于发现知识产权冲突，COPS 系统对检测大面积文本比较有效，而且运算速度快，但是不能检测出对句子局部复制的现象，而且由于英文句号还可以作为分隔符，这种方法不容易判断句子的结束。SCAM 系统借鉴了信息检索技术中的向量空间模型，使用基于词频统计的方法来度量文本的相似性。但是 SCAM 的文本块定义过小，会出现将没有雷同现象的文本误判为雷同的现象。悉尼大学的 Wise 开发了 YAP1, YAP2, YAP3 系列工具，YAP1 和 YAP2 是用于程序复制检测的工具，YAP3[7]利用程序复制检测的方法，既检测程序复制，也检测文本复制。但主要还是用于程序检测，作为文本检测使用较少，特别是中文文本的检测尚未涉及。

张刚等人提出了一个大规模网页去重算法[8]，用于在大规模文件系统中对内容相似的文件进行去重。该算法提出了文本特征码的概念，就是基于字符串匹配的方法来度量文件之间的相似性。本文采用类似的算法，不但提高了报告间雷同比较的效率，还适用于在互联网上查找抄袭的文本。

基于上述背景，结合多年的教学实践和科研经验，我们开发了高效的雷同报告检测系统。该系统的处理对象是电子作业，其内容是自然语言文本而不是形式语言代码，支持 TXT、PDF、HTML、RTF 和 DOC 等多种文档格式。该系统可以减轻老师的工作量、减少甚至消除人工判定的误差，从而督促学生独立思考，在一定程度上保证作业的质量，为充分发挥学生的个性化和创新性奠定基础。

2. 系统总体设计

2.1 总体模块设计

本系统核心功能主要是对自然语言文本进行相似度计算，从两个方面来判断文件是否属于抄袭文件：一是在指定文档集内找出两两相似的文档；二是从网络方面，通过将搜索到的网页和本地文档比较，找出和文档相似的网页。在设计雷同报告检测系统时，将系统划分为三个模块，系统结构如图 1 所示。

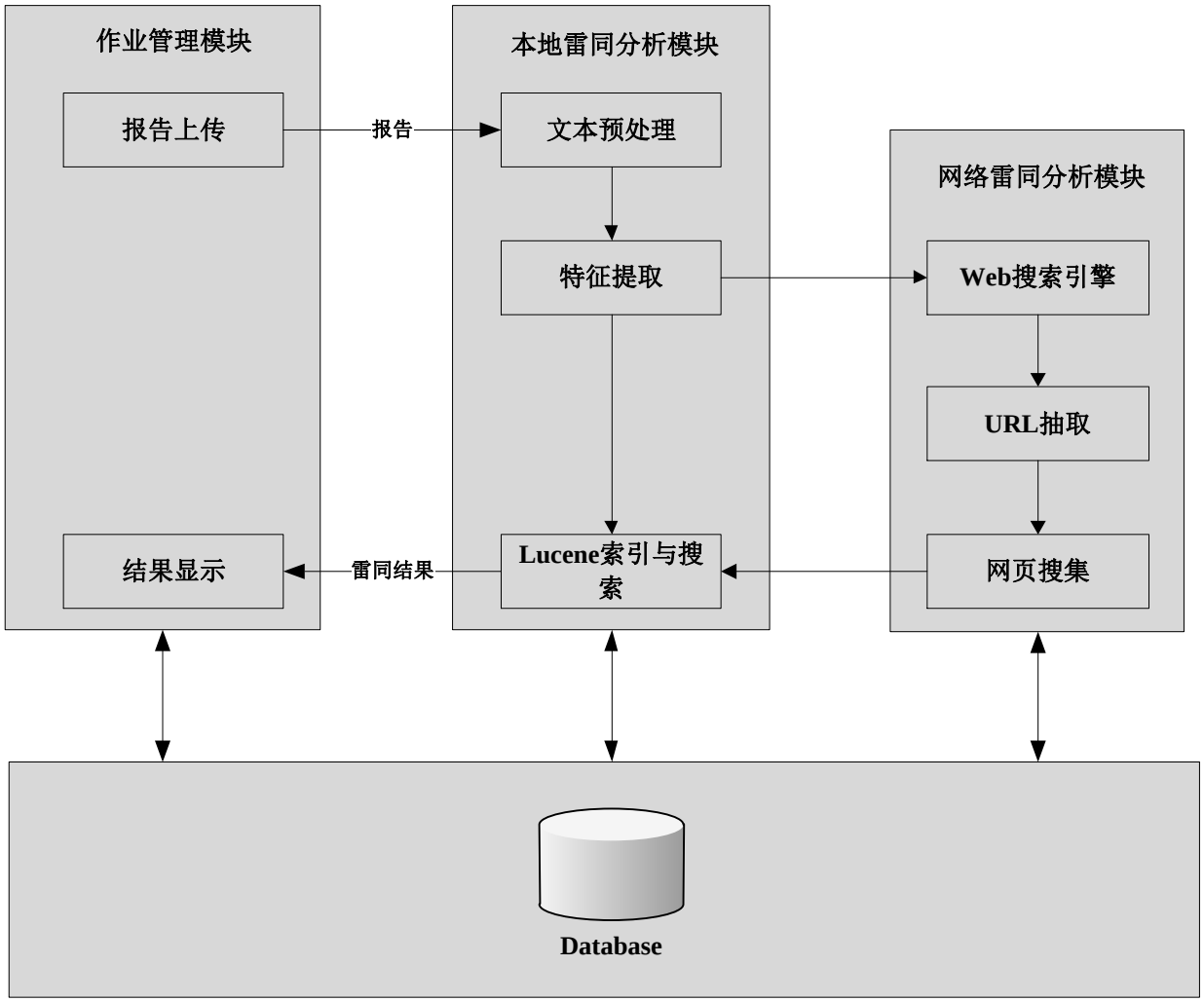

图 1 系统总体结构图

(1) 本地雷同分析模块：主要包括文本预处理，将不同的格式统一处理成纯文本的形式，对处理后的文档进行倒排索引，随机提取特征码，针对提取的特征码在倒排索引中进行搜索，进行相似度的计算，再根据搜索到的相似度进行排序，将结果存储至数据库。

(2) 网络雷同分析模块：主要是基于搜索引擎（Microsoft Bing）提供的 Search API，调用它的 Web Service。具体过程是对文本进行特征提取，将提取出的特征码投入到 Web 搜索引擎中，然后从搜索出的结果中提取 URL，根据 URL 搜集网页至本地，再调用本地检索模块进行雷同判定。

(3) 作业管理模块：报告上传部分提供接口供用户将待分析的报告文档传到服务器上；结果显示部分主要是从数据库中取出疑似雷同的文档对，通过二次比较，计算出两个文档间详细的雷同信息，并友好地在网页上显示出来。

2.2 数据库设计

根据需求分析，考虑到系统的具体的功能，一共是四个表，每个表都有各自的主键，关系表的主键和实体的主键设置的是级联关系，整个数据库的设计如图 2 所示。

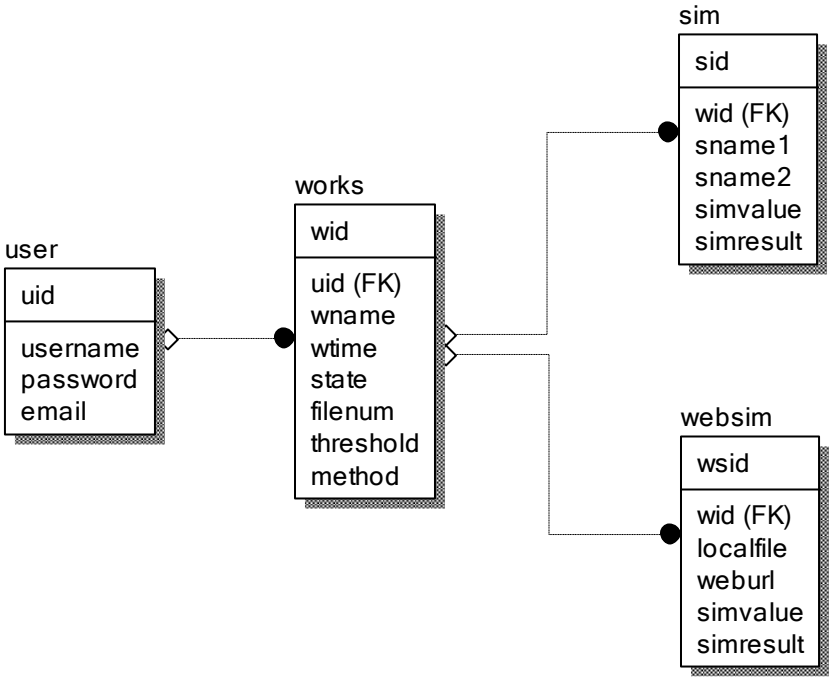

图 2 数据库 E-R 图

2.3 关键技术分析

(1) 雷同的界定

雷同很难用一个量化的指标来确定，但是目前比较通用的判断方法是，如果文本去掉标题、摘要和注释等内容后有 10%的内容相似，即认定为雷同。但是在教学领域，同学们提交的报告一般都是同一个主题；而且限于知识面，学生报告的内容受教科书及教师讲课内容的影响较大，所以本系统将相似度阈值暂定为 30%，日后教师可根据系统的实际使用情况以及应用背景的不同，自行设定合适的阈值。本系统只针对字面上的雷同进行判定，无法识别仅对文档思想进行抄袭的情况。系统的判断结果仅仅作为一种建议，教师可以根据系统给出的结果最终进行雷同确认。

(2) 文本预处理

在文本预处理的时候首先进行的是文本格式转换，将 HTML、PDF、MSWord、XML、RTF 等格式的文件都转化为纯文本文件，为下一步工作做准备。

对于提取的文本文件，需要进行汉语分词处理。词是汉语的基本单位，然而由于汉语的书写是不对词进行分割的，因此在使用计算机进行中文处理时，首先需要进行分词工作，这也是中文自然语言处理中的基本、核心问题。

(3) Lucene 全文索引和检索

文件系统内的雷同比较主要是采用了 Lucene 的全文索引和检索技术，包括建立倒排索引和对倒排索引进行搜索。

Lucene 的检索原理是搜索索引，是一种典型的用空间换时间的做法。检索的时候，需要先对检索内容建立索引，检索词只在索引上进行搜索。所建立的索引是倒排索引，也就是针对关键词，记录该关键词出现的位置、次数、关键词所对应的文件名等信息，这样能快速得到检索结果。相对于检索速度得到的回报，所付出的空间代价是微不足道的，因为文本文件所占用的磁盘空间少，而索引文件大概只占内容大小的 30%。

Lucene 的检索过程就是建立一个排好序的关键字列表，用来存储关键字和记录的关系，并使用关键字和记录编号的映射以及关键字在记录中出现的次数、频率、位置的映射来检索，这样就避免了在所有文章中搜索关键字的麻烦，大大提高了检索效率。同时，Lucene 使用一套非常优秀的评分机制来对检索出的文档进行相似性排序。这套评分机制综合了检索词的词频、逆文档频率、激励因子（权重）等因素来衡量文档间的相似性，以使结果更为合理。

(4) Web Service 技术

在网页的雷同比较中主要使用的是搜索引擎的 Web 服务，Web 服务（Web Service）是一种基于 XML 的软件技术，它提供了一个标准的方式，用于程序之间的通信和操作。Web 服务定义了一部分协议规范来实现松散耦合下的对象访问，主要技术标准如下：HTTP（Hyper Text Transfer Protocal 超文本传输协议）、XML 语言（eXtensible Markup Language 可扩展标记语言）、SOAP（Simple Object Access Protocol 简单对象访问协议）、WSDL（Web Service Description Language Web 服务描述语言）。

3. 系统详细设计与实现

3.1 作业管理模块的详细设计与实现

作业管理模块分为用户管理、作业管理、雷同结果显示三个子模块。

用户管理的主要功能是用户注册、登录、个人基本信息的查看以及修改，其中个人基本信息包括用户名、密码和电子邮箱。当用户登录成功以后，便可以添加一个新作业。用户可以设置雷同检测的相似度阈值，可以选择只进行本地雷同检测、只进行网络雷同检测、或本地和网络雷同检测均进行。添加完一个作业以后，用户可以查看并修改作业的设置，使用文件上传模块上传电子文档。当用户点击了“开始进行雷同检测”后，会在数据库中插入一个检测任务。后台程序则不停的查询数据库并取出任务一个个执行。对于已经完成的雷同检测任务，用户可以查看具体的雷同结果。

作业管理模块的程序流程图如图 3 所示。

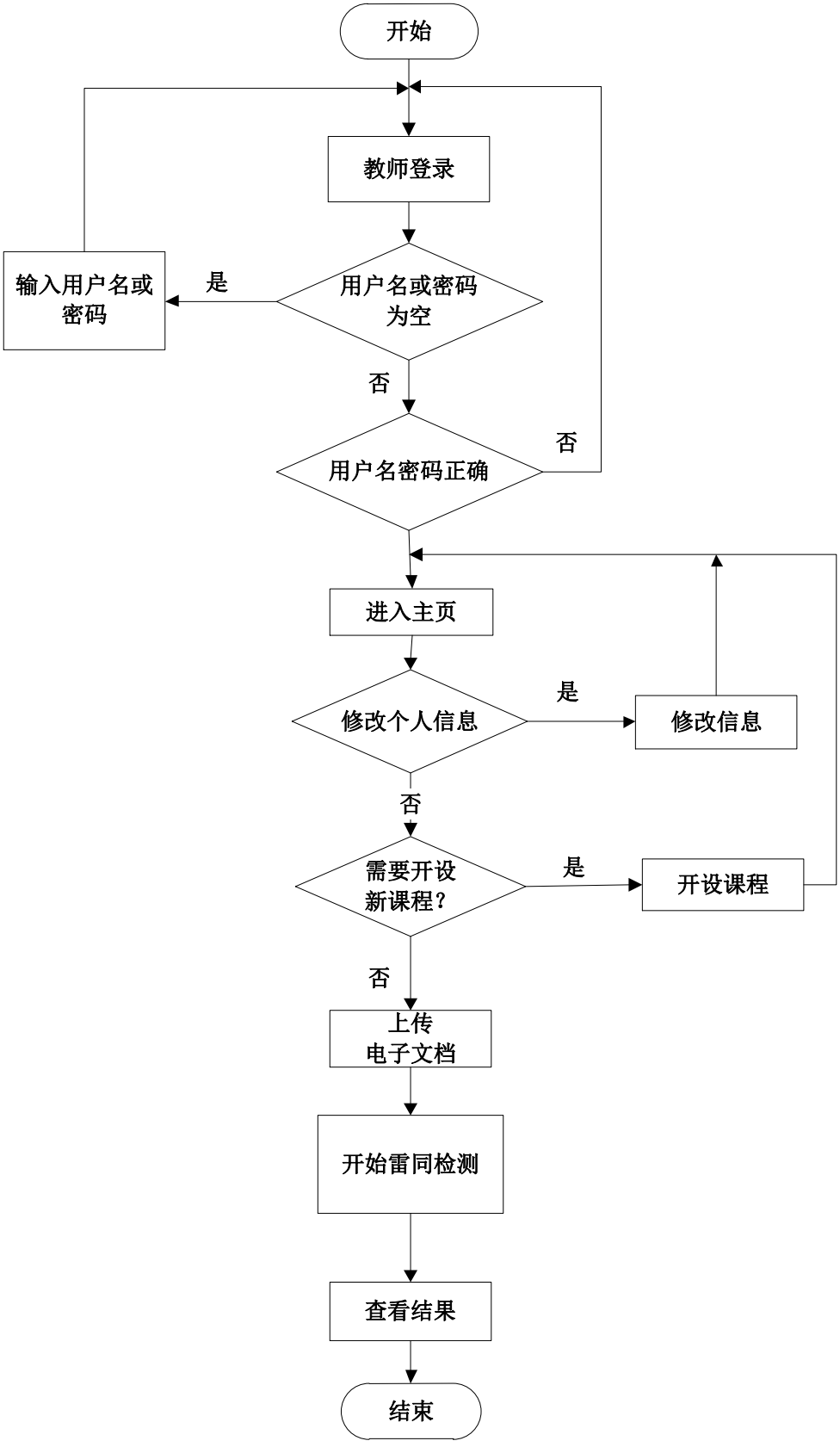

图 3 作业管理模块程序流程图

3.2 本地雷同分析模块的详细设计与实现

文件系统内的雷同检测主要包括文本的预处理，文本特征码的抽取，Lucene 索引与检索，相似度计算以及排序。文件系统内的雷同检测主要是基于 Lucene 工具包进行相似度计算，得出雷同结果。Lucene 工具包是一个用 Java 写的全文索引引擎工具包，通过实现其接口可以在应用程序中方便的实现全文索引[9]。本模块的系统结构和源代码组织结构图如图 4 所示：

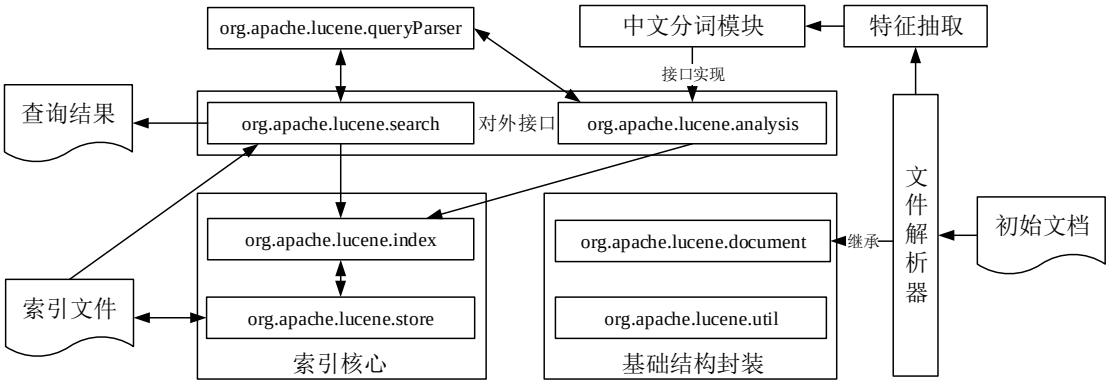

图 4 系统结构与源代码结构图

在文件系统内进行文本相似性计算的主要过程如下：

首先是对文本进行预处理。文本预处理主要是对各种形式的文本进行正文提取，为接下来的特征抽取以及建立倒排索引做准备。

接下来对原文本建立倒排索引。使用的是 Document 的“Path”以及“Content”域。其中“Path”域保存原文本的路径，“Content”域保存由之前的预处理模块获得的文本正文。文件的主体信息被存到这个域中进行倒排索引。

建立倒排索引之后，对文本进行特征抽取以及相似文本检索。特征抽取的主要过程是随机抽取一定数量和长度的特征码，然后对所有的特征码进行组合，构成特征向量。接下来，将特征向量投入索引文件中进行搜索。搜索到的文件按照 Lucene 的评分机制进行排序。搜索结果是一个封闭的文档集合，对该集合内的文档计算相似度并显示雷同结果。

文件系统内处理文本相似的流程图如图 5 所示：

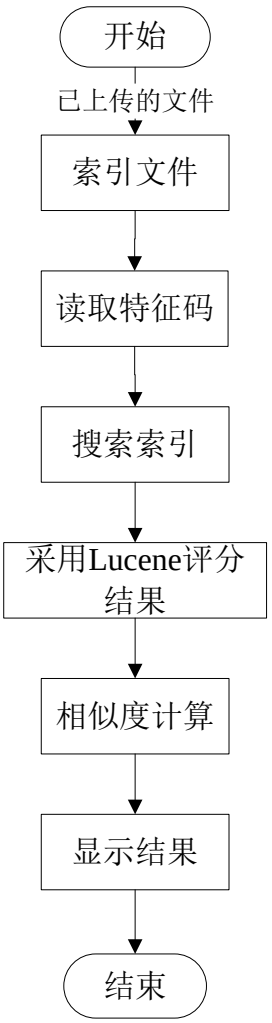

图 5 文件系统内雷同检测模块流程图

3.2.1 文本预处理

文本预处理负责对各种形式的文本进行正文提取。目前所支持的文本格式有：MS Word, PDF, HTML 以及 TXT。以下以 MS Word, PDF 以及 HTML 文本格式为例介绍一下正文提取的方法。

对 MS Word 格式文本的解析采用的是开源的 POI 工具包（版本为 3.6）。部分关键代码如下：

```
POITextExtractor extractor = ExtractorFactory.createExtractor(file);
content = extractor.getText();
```

对 PDF 格式文本的解析采用的是 PDFBox 技术，PDFBox 是一个开源的 Java PDF 库，该库允许用户访问 PDF 文件的各项信息。通过使用 PDFBox 提供的 API，可以实现从一个 PDF 文件中提取出文本信息。部分关键代码如下：

```
PDDocument document = PDDocument.load(file);
PDFTextStripper stripper = new PDFTextStripper();
StringWriter writer = new StringWriter();
stripper.writeText(document, writer);
content = writer.toString();
document.close();
```

对 HTML 格式文本的解析所采用的则是 htmlparser 工具包，部分关键代码如下：

```
Source htmlSource = new Source(new FileInputStream(file));
content = htmlSource.getRenderer().toString();
```

文本预处理的流程图如图 6 所示：

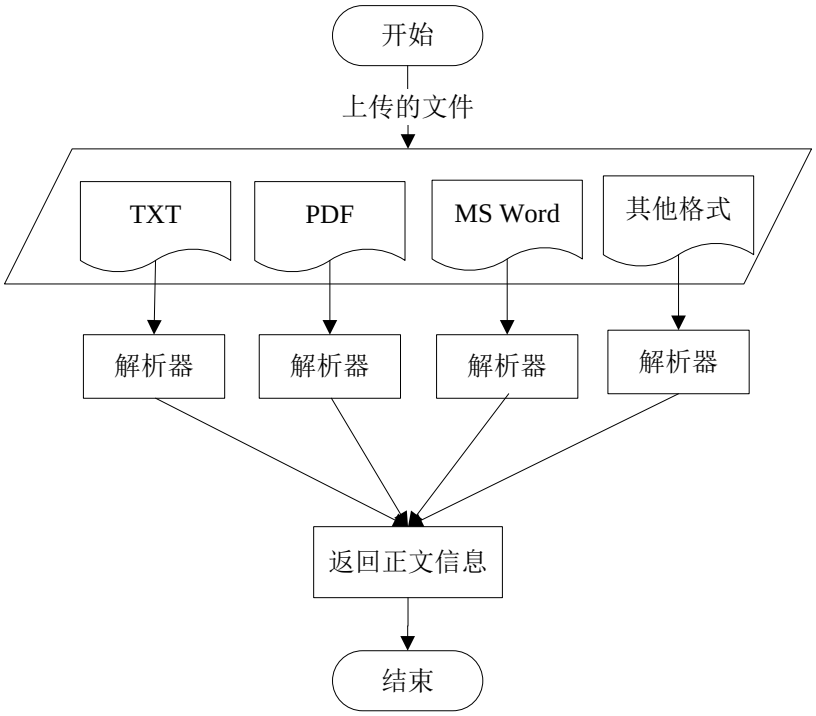

图 6 文本预处理阶段流程图

3.2.2 建立索引

建立索引的过程就是将各种不同格式的文件处理成方便查询的索引文件的过程，相似性比较的过程其实是对文档关键词以及关键词位置信息在大量文档中的搜索的过程，如果不建立索引，就需要将文档顺序读入内存，然后根据字符串匹配算法，检查这个文件是不是含有所要搜索的关键词，这将耗费比搜索引擎搜索关键词高不止一个数量级的时间。建立索引的目的是为了是随机访问存储在索引中的关键词，进而找到该关键词所关联的文档。Lucene 采用倒排索引（Inverted Index）的机制。倒排索引也称反向索引，索引维护了一个词/短语表，对于索引中的每个词/短语，都有一个链表描述了有哪些文档包含了这个词/短语，以及这些词和短语出现的频率。搜索倒排索引能快速得到搜索结果。

索引机制如图 7 所示

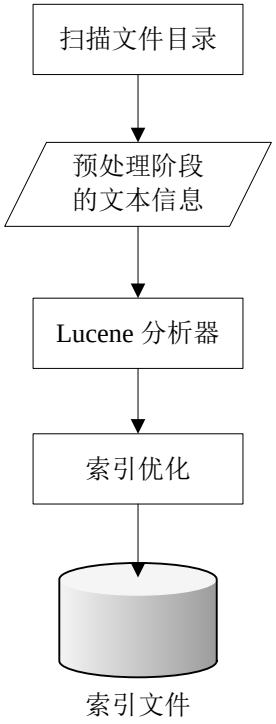

图 7 索引机制图

根据索引的机制，调用 Lucene 定义的工具包，以及 Java 本身定义的文件操作包，对文件目录内的文件建立倒排索引，存放服务器单独的文件夹内。

3.2.3 特征抽取以及相似文本检索

特征码的提取主要是采用随机化的算法和借鉴大规模网页去重算法。采用随机化算法主要是为了降低偶然性，通过多次随机的抽取特征码进行相似度的计算，提高雷同检测的正确性。借鉴大规模的网页去重算法的主要原因是一般的聚类问题是把某些特征相似的元素聚成一类，而雷同报告的检测则是要相似度达到一定的阈值之后才判断为一类，因此不采用一般的聚类方法，而采用基于特征码向量的相似度计算来判断雷同。

考虑到文档中存在句号，问号等句间分隔符，以句号为例：特征提取的算法首先以一个句号作为一个提取位置，分别在句号两边提取 L/2 长的词串作为报告的特征向量中的一个特征码。特征码的定义与提取中的参数 L 十分重要。特征码太长会给存储计算带来较大的开销，特征码太短又会降低它的区别能力。一般的可以取 L 为 10。按照信息论中多元文法(N-Gram)的定义这个特征码相当于一个 10 阶文法，如果按照 6763 个汉字计算，这个特征码重复概率大约为 $1/(6763)^{10}$，因此也就保证了特征码的在文档中的特殊性。

抽取出特征向量之后，接下来将特征向量放入 Lucene 搜索引擎中进行搜索。搜索结果按照 Lucene 的评分机制进行排序，并根据预先设定好的阈值进行雷同判定，将相似度大于阈值的文档对判定为雷同，并将雷同信息写至数据库。

将特征向量投入到 Lucene 搜索引擎中计算时，Lucene 所采用的模型为空间向量模型(VSM) 经过中文分词后的特征码，根据词项在文档(tf)和文档集(idf)中的频率(frequency)计算词项的权重，通过计算文档和查询的内积或余弦等来表示文档和查询的相关程度。每一串特征码搜索结果排序就是基于这种排分机制。Lucene 排分机制的计算公式如式 2-1 所示：

$$score(q,d) = \sum_{t\ in\ q} \text{tf(t in d)} \bullet idf(t) \bullet \text{boost(t.filed in d)} \bullet \text{lengthNorm(t.field in d)} \quad (2\text{-}1)$$

评分公式中的因子如表 4-1 所示。

表 1 评分公式中的因子

评分因子	描述
tf(t in d)	文档 d 中出现特征码 t 的频率
idf(t)	特征码 t 在倒排文档中出现的频率
boost(t.field in d)	域的加权因子，它的值在索引过程中设置
lengthNorm(t.field in d)	域的标准化值，记载某一域中所有项出现的个数，通常在索引时计算该值并将该值存储到索引中
coord(q,d)	协调因子，基于文档的特征码个数
queryNorm(q)	每个查询的标准化值，之每个特征码的权重的平方和

计算出相似度之后，要进行相似度的判断，文档和该文档本身的相似度一定是 1，也就是 100%，因此要排除对搜索文档本身的计算，是对搜索算法的一个微小的剪枝。只考虑文档和其他文档的相似度。因为文档搜索是个 $O(n^2)$的算法，一个文档和另一个文档相似，在搜索另一个文档的时候必然出现和该文档的相似，导致在显示雷同结果的时候，相似的文件对会重复显示，解决该问题的方法是，将超过阈值的文件集合，使用字典序排序，将搜索文档和结果集合配对，文件名和文件名之间采用特殊字符链接形成文件对，建立 Hash 表，将文件对压入 Hash 表。

在计算完所有文件的时候，从 Hash 表中取出文件对，根据特殊符号将文件对分成两个文件名，取得每个文件的文件路径，显示雷同的文件名的同时给每个文件名加入其文件路径的链接，方便用户复查文件是否存在大面积雷同，防止文件误判的产生。

考虑到阈值设定太小会导致误判率的加大，阈值太大会加重漏判的比例，本系统将阈值设定为 0.5。

计算相似度的流程图如图 8 所示：

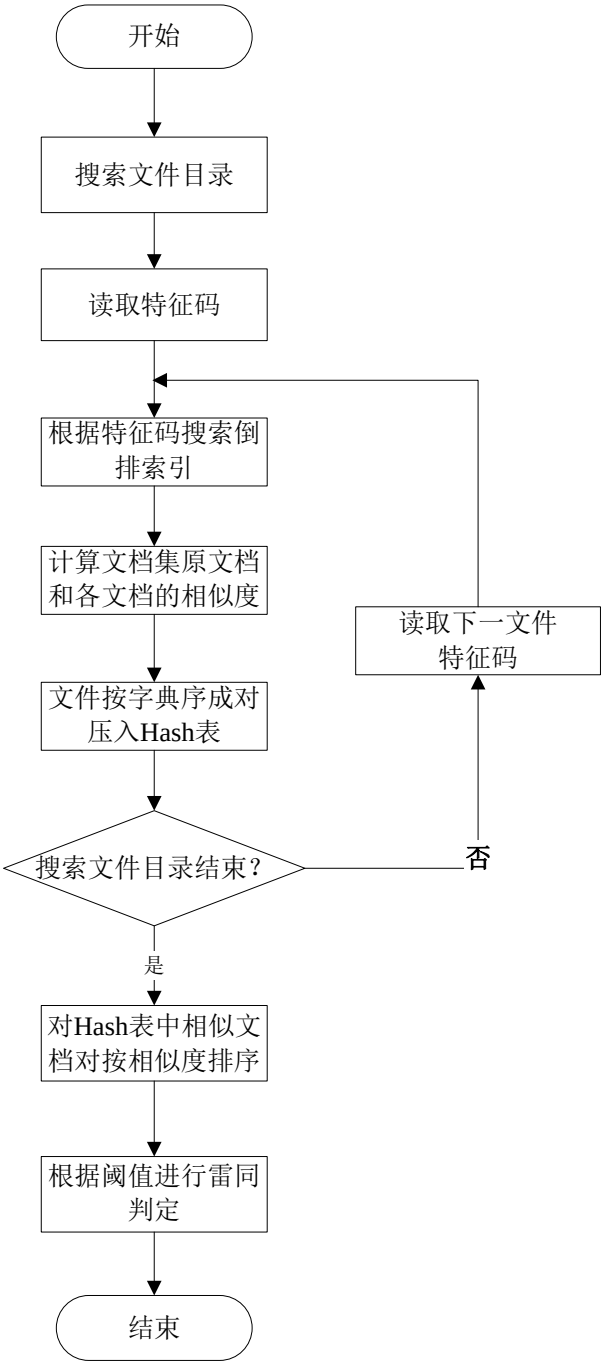

图 8 相似度计算流程图

3.3 网络雷同分析模块的详细设计与实现

网络部分的实现主要是调用 Microsoft 公司 Bing 搜索引擎所提供的 Web Service API。实现时，将从报告文件中抽取出来的特征码投入到 Bing 搜索引擎中，Bing 所提供的 Web Service 将以 XML 文件形式返回 Bing 搜索引擎的搜索结果，搜索结果中包含网页的标题，摘要，URL，日期等信息。然后通过对该 XML 文件进行解析（解析时我们采用了开源的 dom4j 工具包）抽取出每个搜索到的网页的 URL 地址。有了 URL 地址，便可以利用 java 提供的网络包将相关网页搜集至本地，保存为网页文件。接下来便可采用本地方式进行相似文本的检测。

实现的主要步骤如下：

1. 抽取文件的特征码。考虑到特征码的随机性，也就是说，抽取出来的特征码不一定具有代表性，在我们的系统中，抽取了 10 个特征码，这样，就增加了网页的命中率。
2. 访问 Web 资源。将抽取出的多个特征码作为请求参数分别提交给 Bing 搜索引擎的 Web Service，得到若干个 XML 格式返回结果，利用多线程技术抓取至本地以作后续的解析。
3. 解析 Web Service 的返回结果。Web Service 以 XML 文件形式返回它的搜索结果，因此这个过程主要进行 XML 文件的解析。我们所使用的是 dom4j 技术。
4. 网页搜集。通过上一步对 Web Service 调用结果进行解析，得到了一个 URL 地址集合。于是利用多线程技术将该集合内的网页抓取至本地，保存至临时文件夹。
5. 归入索引。对搜集到的网页进行正文提取，并归入索引，为后续的相似性检测作准备。
6. 相似检测。对课程报告所在目录中的每一个文件进行特征抽取，然后将特征码放入索引文件中进行搜索。这个过程与本地相似性检测基本一致。

网络部分雷同检测的流程图如图 9 所示：

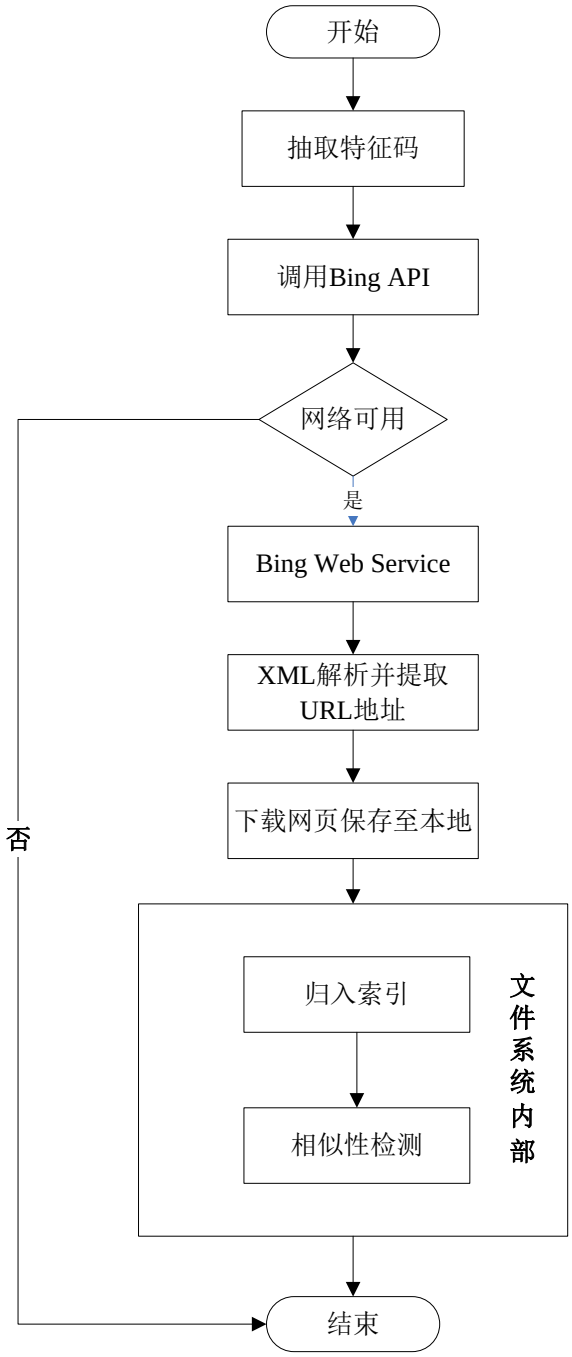

图 9 网络雷同流程图

3.4 开发环境和工具

(1) JDK 1.6

系统使用 Java 作为主要开发语言。因为 Java 移植性好，使系统可以轻松的在不同的环境部署。Java 是面向对象的语言，比面向过程的语言更适合项目的开发。Java 有很多开发好的库，例如本系统中的 Lucene，使用这些库可以大大简化系统的开发。

(2) MySQL 5.1.10

系统使用 MySQL 作为数据中心。MySQL 是目前最流行的开放源代码 SQL 数据库管理系统，具有体积小、速度快、成本低等优点。开放源代码的 MySQL 不仅可以免费使用，同时

开源社区对 MySQL 有良好的技术支持。使用 JDBC 可以让系统和 MySQL 数据库轻松实现数据交流。

(3) OpenSolaris 2009.06

OpenSolaris 是一个由 Sun Microsystems 所发起的开放源代码项目，用来建立以 Solaris 操作系统为主的开发者社区。本系统建立在安全、稳定、高性能的 OpenSolaris 操作系统上。其他用户可以根据自己的需求，将系统部署在任何支持 Java 和 MySQL 的系统上。

(4) NetBeans 6.7.1

NetBeans 是一个全功能的开放源码 Java IDE，可以帮助开发人员编写、编译、调试和部署 Java 应用，并将版本控制和 XML 编辑融入其众多功能之中。我们使用 NetBeans 进行系统的开发。

(5) GlassFish 3.0

GlassFish 是一个免费、开放源代码的 Java 应用服务器，是一款强健的商业兼容应用服务器，达到产品级质量，可免费用于开发、部署和重新分发。GlassFish 在静态文件传输方面的性能比 Tomcat 要强得多，而且可以支持更多的并发访问。我们使用 GlassFish 作为应用服务器。

4. 系统测试和性能分析

4.1 系统测试环境

MORS 系统内核采用 Java 语言进行开发，并基于 Open Solaris 操作系统以及 Glassfish 3.0 Web 服务器搭建配套网站，以提供更完美的服务。系统的部署过程可参考随本报告一起提交的部署说明书。MORS 具体测试环境如下：

(1) 软件测试环境

运行本系统所需要的支持软件：

- 操作系统：Open Solaris
- 所需软件：Java 运行时环境（JRE），Java 开发工具包（JDK5.0 以上）
- 服务器：Glassfish 3.0
- 数据库：MySQL 5.1.10

需要说明的是,由于 Java 语言的跨平台特性，本系统可移植到 Windows, Linux 等系统，经过测试，不会带来跨平台问题。

(2) 硬件测试环境

为运行本系统所要求的硬件最小配置：

- 奔腾 IV 处理器，主频 550MHZ
- 内存容量 1G

需要说明的是：由于 MORS 系统内核运行对内存需求较高，因此内存容量建议 1G 以上，2G 更佳。

4.2 性能分析

(1) 测试使用的文档集

我们使用了三门计算机类课程的实验报告作为系统测试用的文档集。以下将用课程 A、课程 B、课程 C 来表示。

(2) 测试方法

由于没有标准的雷同文档测试集，因此我们采取人工投票的机制对 MORS 系统自动雷同检测的结果进行人工的评判。以三个人的投票团队为例，如果超过（包含）2 个人认为一份报告与另一份报告雷同，或者与网络资源雷同，则 MORS 的自动检测结果被认为是正确的。

因为同样的原因，我们无法获得召回率，因此采用准确率作为衡量系统性能的标准。

(3) 测试结果

对三门课程分别进行性能测试的结果如下表所示：

- **本地测试**

表 2 课程 A

报告数量	报告平均字数	阈值	系统发现雷同数	核对确定雷同数	准确率
2397	627	0.5	3235	3193	98.70%

表 3 课程 B

报告数量	报告平均字数	阈值	系统发现雷同数	核对确定雷同数	准确率
994	2696	0.3	3222	3195	99.16%

表 4 课程 C

报告数量	报告平均字数	阈值	系统发现雷同数	核对确定雷同数	准确率
183	1570	0.3	53	53	100%

- **网络测试**

表 5 课程 B

报告数量	报告平均字数	阈值	系统发现雷同数	核对确定雷同数	准确率
994	2696	0.3	26	26	100%

表 6 课程 C

报告数量	报告平均字数	阈值	系统发现雷同数	核对确定雷同数	准确率
183	1570	0.3	175	170	97.14%

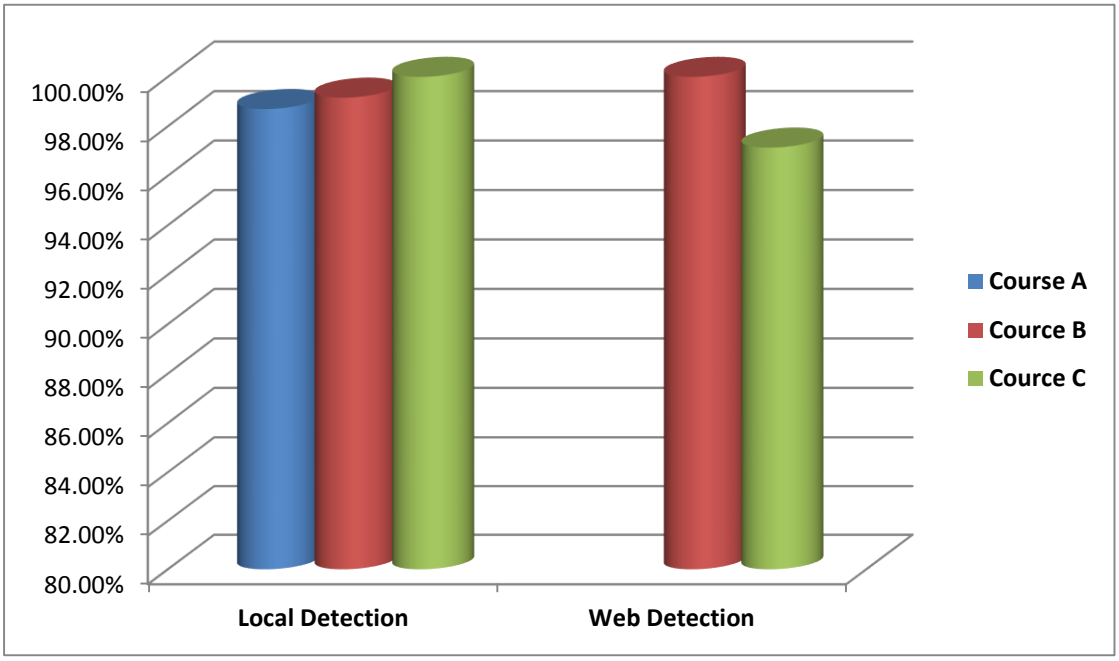

图 10 测试结果柱状图

(4) 结果分析

课程 A 的本地自动检测结果中雷同文档数目非常多，这是由于我们所采用的课程 A 的报告是给定题目的问答式的报告，自由发挥性不强。因此对于这类报告可将阈值设高一点。

对比 MORS 系统自动检测以及人工核对的结果可以看出，MORS 在本地以及网络雷同检测都表现良好，准确率均在 97%以上，误判率很低。

我们所选择的三门课程为计算机专业中比较具有代表性的学科，根据评测结果可以看出，MORS 对不同的学科均有良好的雷同检测能力。

因此，评测结果显示 MORS 具有良好的实用性。

5. 经费使用情况

项目共有经费￥15000，共使用￥14975，具体使用情况如下表：

表 7 经费使用情况

项目	经费
服务器购置	￥9000
图书及参考资料购买	￥5000
打印、复印、网费等	￥975
合计	**￥14975**

6. 收获与体会

6.1 项目组组长柳俊丞

曾经以为，创新只是绞尽脑汁想到的一个好点子。但是，现在我终于明白，创新不仅仅是一个好的设想，设想有时候只是空想，只有经过实践考验过的创新，才是真正的创新。所谓创新，新固然重要，但是创造的过程更重要。我们的在项目实施的过程中，并不是一帆风顺，也遇到许多的困难，但往往就是在我们最困难的时候，突然想到一个好的点子，问题优雅的解决了，这就是创新。我们项目中的一些创新，是我们在项目开始前，并没有想到的，很多的困难，也是项目开始之前没有预料到的，很多时候，当我们尝试了很多方法都没有解决问题时，甚至几乎陷入僵局时，突然有了新的想法，正所谓山重水复疑无路，柳暗花明又一村，创新就在这个时候产生了。要创新首先要超越自我，没有实践过的人永远不可能有真正的创新。

6.2 项目组成员唐国华

在项目开展的过程中，感触最深的一点就是对于未知领域的好奇心，当一个问题是处于未解决的时候，我们都不知道后面会遇到什么问题，只好渐渐地去探索，而在探索的过程中，往往碰到各种奇怪的情况，使刚有个起步的想法即刻夭折。经过几次以后，一开始的热情就会降低很多，而能不能够继续坚持下去很大程度上取决与一个人的好奇心。还有对于创新很重要的一点事坚韧性。在对于一个问题的坚持不断地追求下，问题才可能会被慢慢地吃透，从而被解决。在项目开展过程中，我们遇到过各式各样的问题，也有过无数的讨论和争执，如果没有对于事物的坚韧的精神，可能会持续不下去吧，整个项目最黑暗的时候也就是项目会有重大突破的时候。

项目开展的过程，也是个人不断成长的过程，无论是学术理论上的抑或是实践编程上的，还是为人处事，团队配合，自己感觉都有了不小的进步。希望今后的国家创新性实验项目能够更为推广，从而使更多的学生参与到这个有意义的活动中来。

6.3 项目组成员郭江

在项目中我负责的是文本比较及相似度计算部分，也可以说是整体集成部分。在这部分工作中，我力图降低各个独立模块之间的耦合性，增加模块内聚性，使用策略模式以增强代码的复用程度。同时，效率在这部分也显得尤为重要。进行 web 检索时的多线程机制，如何最大限度地减少 IO 操作，这些都是提高时间效率的关键。空间上，因为 Java 的垃圾自动回收机制，使得内存管理简单了很多，但是还是遇到过内存溢出的问题，因此在程序中必须尽量优化空间。在显示模块未完成之前，格式化输出也显得很重要。Mors(系统名称)最初的成形版本由于效率低，准确率不高，而且输出不规范，对用户极度不友好而成为不为人知的历史。哦，提到了准确率，这是系统的灵魂所在。在特征码的选择方式，数目，以及文本比较方式上有很多技巧，都是在一步步的实验中被发掘出来，并得到最优化，也许目前并不是最优，但是实验者不会固步自封。 在项目开展过程中，车万翔老师给了我们很多非常珍贵的指导，队友之间的交流或是争论中也使得一些原本模糊的问题慢慢地走向明朗。相信 Mors

一定不会辜负大家的努力，而终将成为一个优秀的实用系统。

7. 结论

本文从项目简介、总体设计、系统详细设计、测试与性能分析等几个方面阐述了的“报告雷同检测系统”的设计与实现过程。

7.1 项目完成情况

从整体上看，报告雷同检测系统基本达到预期目标。能够实现正确的文件系统内的雷同报告的检测，显示文件和文件之间雷同的文件对，上传我也能告的文件名和路径，并详细显示具体雷同之处；以较高的准确率实现网络雷同的检测，显示本地文件的路径以及与之雷同的网页的超链接。

7.2 特色与创新

本系统是专门面向教育领域的学生报告雷同检测系统，与论文及学术抄袭系统不同，我们针对学生报告的特点以及通过和一些有丰富教学经验的教师交流，为本系统设定了独特的实现方案和判断标准，使其更适合教师在日常教学中对学生报告雷同的检测。

本系统使用的是基于检索的算法，检索的算法复杂度为 $O(n)$,比传统的聚类算法的时间复杂度 $O(n^2)$要低的多，处理时间不会随着文档数量的增加而明显增大，适合大规模文档的处理，可以让使用者在较短的时间内就可以得到结果。

本系统加入了二次比较模块，第一次比较时使用基于检索的算法，找出可能雷同的文档对；第二次比较时使用迭代的最长公共子串算法，对第一次比较返回的可能雷同文档对进行详细比较，找出其具体雷同的位置。使用二次比较，极大的缩短了算法运行时间，提高了系统的效率；同时能够提高比较精度，简单明了的展示雷同结果，给用户更好的体验。

目前所有雷同检测程序或者系统都是基于给定的文档集合，还没有一个针对整个互联网的雷同检测系统出现。虽然有些系统声称可以针对互联网进行雷同检测，也是事先在互联网上将相关资源搜集起来，再进行比较。本系统借助于搜索引擎的支持，可以动态的在整个互联网上进行检索，发现雷同。

7.3 将来的工作

同时，本系统也有一些不足之处有待完善。首先在网络雷同结果中，考虑到对网页提取正文的结果含有很多无关的内容，比如广告链接、JavaScript 代码、CSS 代码等，所以没有实现像本地雷同那样可以具体的查看具体雷同之处，而只是提供了网页的链接；其次目前网站功能比较单一，文件雷同中，如果文件中的特殊标点符号，比如句号，逗号过少，会导致抽取特征码随机抽取的时间部分增加，从而增加了整个系统的运行时间；在网络雷同中，如果文件的文字信息也就是自然语言信息太少而含有大量的代码或英文字符，特征向量会不足以分割成有效的特征码，导致网络部分搜索结果的随机性加大，准确性降低，目前想到的解决方法是将抽取的摘要放入文件系统雷同比较前建立的倒排索引中进行反向搜索，结合互搜索的信息判断，由于时间紧迫，这些功能将会在以后的版本升级中解决完

成。

致谢

首先感谢指导教师车万翔博士。车老师渊博的知识、扎实的专业基础、严谨的治学态度和对科学的献身精神，是我们一生学习的榜样。从车老师那里，我们学到的不仅仅是知识，更重要的是思维的方法和治学的态度。项目的完成凝聚了车老师的大量心血，我们的每一点进步都倾注着他的关心和帮助。在这里，衷心的感谢车老师的教导和培养。

感谢孙志岗等老师提供测试数据，帮助测试我们的系统，并提供改进建议；感谢在项目过程中提供指导、关心和帮助的其他老师和同学。

特别感谢 Sun Microsystems 公司给我们提供这样一个难得的机会，使得我们可以尽情的施展自己的才华。感谢 Sun Microsystems 公司开发了这么多好用的开源工具和技术，使得我们的项目得以顺利完成。

参考文献

[1]. http://www.uni-weimar.de/medien/webis/research/workshopseries/pan-09/index.html

[2]. http://theory.stanford.edu/~aiken/moss/

[3]. http://cms.hit.edu.cn/course/view.php?id=44

[4]. SiA，LeongHV，Lau RWH. CHECK: A document plagiarism detection system. In: proceedings of the ACM SymPosium for Applied Computing. 1997:70~77

[5]. Udi Manber. Finding Similar Files in a Large File System. In Proceedings of the Winter 1994 USENIX Technical Conference.1994: l~10

[6]. BrinS, DavisJ, Gareia-MolinaH. Copy Detection Mechanisms for Digital Documents. In: Proceedings of the ACM SIGMOD Annual Conference. 1995

[7]. Wise MJ. YAP3: Improved detection of similarities in computer programs and other texts. In: Proceedings of the SIGCSE'96. 1996, 130~134

[8]. 张刚, 刘挺, 郑实福, 车万翔, 李生. 大规模网页快速去重算法. 新加坡 ICCC2001

[9]. http://lucene.apache.org/java/docs/index.html

[10]. http://lucene.apache.org/java/2_3_2/scoring.html