PaperTest检测报告简明打印版

对比结果（相似度）：
总相似度：27% *（相似字数占总字数的百分比）*
红色相似度：15% *（句子相似度70%-100%的字数占总字数的百分比）*
橙色相似度：12% *（句子相似度40%-70%的字数占总字数的百分比）*

验真伪二维码

编 号：09b86412-f6b8-40ca-915c-2b9b6a549e8e
标 题：676907067383428
作 者：6769070673834281
长 度：28387字符（不计空格）
段 落：506个
句 子：834句
词 语：14277个
时 间：2014-6-1 13:41:24
比对库：学术期刊、学位论文、互联网资源

相似资源列表：
（1）3%即752字来源于学位论文
（2）3%即934字来源于期刊论文
（3）4%即1291字来源于百度文库
（4）17%即4825字来源于其他网络

全文简明报告：

1 绪论

1.1 研究背景

{85%:外观设计是指对产品的形状、图案、色彩或者其结合所做出的富有美感并适于工业上应用的新设计。}伴随着全球经济一体化的主流趋势，以及我国对经济，科技的文化创新的关注度逐渐增强，在这些软实力方面我们取得了长足的进步。{78%:知识产权尤其是能直接或者间接创造经济效益的注视产权，如专利，日益成为国家发展的战略性资源和国际竞争力的核心要素。}{79%:同时，各行业的产品在设计上，尤其是外观设计上越来越趋于多元化和个性化，在性能差距逐渐缩小的背景下，外观设计更是逐渐成为一个企业产品的核心竞争力。}{59%:外观设计专利在市场竞争中发挥着非常重要的作用。}鼓励专利数量与质量的快速发展，以及增强专利的信息化管理，这是在国家与政府角度可以为增强企业竞争力作所处的工作。外观设计专利信息对企业的发展所起到的作用不言而喻，随着企业对保护知识产权意识的逐步增强，对外观设计专利信息的使用将变得非常广泛。

{44%:由于国家的相关鼓励政策以及国内外企业、单位和个人对知识产权尤其是能直接产生利益的专利的重视，我国外观设计专利申请数量在快速增长。}根据国家专利局2012专利统计年报，三种专利的年申请数量成逐年递增的趋势（表1-1）。而根据另一统计，截止2012年12月，我国国内外申请外观设计专利达两百万件，且申请量正以每年近20万件的速度增长（如表1-2）。这一状况对外观设计专利信息的管理与使用提出了非常高的要求。{77%:然而，目前我国外观设计专利信息的管理与利用状况不太乐观。}{85%:一方面，外观设计专利文献检索采用关键字检索模式，其检索范围大，且外观专利图像相似程度靠人工识别进行，工作量大、效率低，}同

时人工识别的主观性导致相似度判断缺乏一种客观评判尺度。{90%:另一方面，外观设计专利服务形式单一，缺少外观设计专利行业数据库、行业外观设计专利态势分析和外观设计专利战略分析等等。}

{97%:根据国家专利局规定，依据专利法，发明专利申请的审批程序包括受理、初审、公布、实审以及授权五个阶段。}{100%:实用新型或者外观设计专利申请在审批中不进行早期公布和实质审查，只有受理、初审和授权三个阶段。}{40%:外观设计专利所采用的这种审查制度，由于专利数量的庞大，文本检索方式的落后，没有有效的利用外观专利图像这一外观专利最基本的特征，经常导致检索过程不严谨，}产生外观设计侵权纠纷。外观设计专利的申请以及当存在侵权纠纷的处理过程中，需要查询相关专利信息。一国家专利局检索系统为例，外观专利检索主要是基于外观专利信息（如专利名称、摘要、申请人、发明人等文字信息）进行检索，将结果呈现出来在通过人工方式查看图片进行外观专利相似度判定。显然这一检索过程存在不严谨的问题。一方面，由于录入信息时信息的缺失等，通过摘要等信息可能不会将所有的符合描述的外观专利检索出来。另一方面，人工对图片进行相似程度的判定，没有相似度的客观的标准。因此，对检索过程进行图像处理中的算法，取代人工判断图片的相似度，具有很大的可行性。{44%:由于图像包含很很多特征信息，因此，使用基于内容的图像检索算法，对图像进行特征提取，创建一个图像特征库，需要检索一副图片的时候，提取图片特征，}对特征库中所有特征进行相似度计算。{48%:特征提取过程将颜色、形状、纹理等特征分别计算相似度，对相似度进行合理的加权平均，进行特征融合，可以更好的结合图像的多种特征。}在此基础上，就能实现尽可能地取代文本检索以及人工检索，提高检索的相似度以及客观性。

表 1-1 国内外三种专利受理年度状况

发明实用新型外观设计合计

合计1985-2007132790614655831211692400 5181

2008289838225586312904828328

2009314573310771351342976686

2010391177409836421273122 2286

2011526412585467521468163 3347

2012652777740290657582205 0649

国内1985-2007711389145493111248643291184

2008194579223945298620717144

2009229096308861339654877611

2010293066407238409124110 9428

2011415829581303507538150 4670

2012535313734437642401191 2151

国外1985-2007616517106528682871 3997

2008952591641142841111184

200985477191011688990 75

2010981112598121491128 58

2011110583416413930128677

2012117464585315181138498

然而，对于基于图像检索的外观专利检索系统，近年来外观专利呈现的极速增长态势，加上图像处理过程重复性以及大数据量原因，传统意义上的检索方式存在稳定性、时效性上的问题，以及对硬件要求非常高。{46%:因此，分布式计算架构成为解决这一问题的方案之一，Hadoop是一个成熟的开源分布式框架。}将图像检索过程搭建在Hadoop上，可以有效解决硬件资源瓶颈的问题，使用足够数量的、硬件资源落后的平台作为Hadoop计算节点，就能实现大型处理器能承担的任务。

表 1-2 国内外三种专利有效状况

发明实用新型外观设计合计

合计2008337215469729388252119519 6

2009438036565804516183152002 3

2010564760857968793354221608 2

2011696939112059692237127399 06

2012875385150104411321323508561

{87%:国内2008127596463342332859923797}

2009180042558791454277119311 0

2010257893849454718056182540 3

2011351288110995884176923030 15

2012473187148683910449973005023

国外2008209619638755393271399

2009257994701361906326913

2010306867851475298390679

2011345651106388060243689 1

20124021981420587135503538

1.2 研究现状

{64%:目前，国内提供的外观设计专利信息检索服务多数还处于起步阶段。}外观专利信息的检索过程一般是通过提供的关键字对外观专利数据库中的字段进行模糊匹配进行查询。这一过程对数据库中的外观专利存储信息有很高的要求。同时也给检索工作带来很大的不便利性。一方面，如果关键字提交过于简单，检索结果将会非常庞大，另一方面，如果关键字提交太多，将会导致有些专利不包含在最终结果中。外观专利终究是外观，而不是文字，因此这一检索方式影响了检索服务的可用性。国内已经建立了很多的外观专利检索系统，如国家专利局检索系统。该系统专利信息全面，但是检索过程根据关键字对专利提交的信息进行检索，没有有效利用到外观专利图像特征这一重要信息。而其他一下专利检索系统因为收费，信息全面性等原因，也存在很多不实用性。而国外存在的一下专利检索系统也只是提供文本检索，没有相应的图像检索功能。

{75%:基于内容的图像检索，即CBIR(Content-based image retrieval)，目前被广泛应用于各个研究领域，如信号处理，}计算机图像处理，模式识别等。上世纪90年代，T.Kato提出了基于内容的图像检索这一概念。{41%:他实现了一个通过提取图像色彩与形状而构建的图像特征数据库，并实现了一些检索算法。}典型的CBIR系统，用户输入一张图片，系统会在建立的特征数据库中对该张图片特征进行相似度计算，得到图片相似度。{57%:传统的图像检索是基于文本的，即通过图片的文件名、描述信息和其他信息等来实现检索功能。}到目前位置CBIR的相关研究已发展近20年，{82%:传统的搜索引擎公司如Google、百度、都已提供功能类似的基于内容的图像检索服务。}

由于外观专利数据量大这一特点，分布式计算成为解决性能瓶颈问题一个很好的方案。Hadoop是分布式系统的一个很好的实现。可以轻松的构建一个分布式计算与存储等功能的系统，完成外观专利大数据量处理这一功能。

1.3 研究内容

对于基于内容的图像检索技术过程，第一步是把所有的图片进行特征提取，构建一个特征库，第二步，将要检索的图片进行特征提取，第三步，将提取到的特征与特征库中的特征进行相似度计算，根据相似度的阈值，返回给用户。

进行的图片检索过程，由于特征库中大量特征的存在，会是一个大数据量计算过程，因此处理器的处理能力在外观专利这一大数据量的情境下将成为一个瓶颈。

本文构建的检索系统的核心是基于图像的外观专利检索，输入特定的图片，使用普通模式对图片进行专利库检索，另一种方式则是搭建在Hadoop上的图像检索过程。通过对比，两种方式的优缺点。即使搭建在Hadoop平台上，在对全部专利库进行大数据量计算过程也不适合B/S方式与用户交互。

1.4 论文结构

{41%:本文第2章介绍了该系统设计与实现过程中使用的技术，第3章分析系统需求，设计系统架构，第4章介绍了系统的实现以及系统成果展示，}第5章设计实验对比普通模式检索和基于Hadoop图像检索的性能，并进行分析，最后一部分，即第6章介绍结论。

2 开发技术

本系统使用的主要开发语言是Java，Web展示使用的是JSP，专利信息的存储使用的是MySQL数据库，图像存储在本地文件系统上，只在数据库中存放图像文件的路径信息，分布式图像检索使用的特征库将MySQL中的特征信息导出为文本格式，并且存储在HDFS上，{49%:Web服务器使用搭建在Linux上的Apache-Tomcat，而Hadoop环境使用四台虚拟机，安装的操作系统为CentOS。}图像检索主要使用的是颜色、形状、纹理相似度加权的融合，而专利批量获取使用一个简单的爬虫对国家专利局的特征库进行获取，存储到本地。下

面将分别介绍相关内容。

2.1 运行与开发环境

{92%:Java语言是跨平台的面向对象的程序设计语言，由Sun Microsystems公司在1995年5月推出，}{82%:Java语言包括 Java程序设计语言和Java平台，即JavaEE，JavaME，JavaSE。}{83%:Java 技术具有通用性、高效性、平台移植性和安全性等特点，目前Java语言广泛应用于个人PC、移动端、互联网、科学研究等方面，}是目前全球开发者最多的开发语言。{65%:尤其是目前在全球云计算和移动互联网的产业市场广阔的环境下，Java语言一定还会有更加广阔的前景。}{62%:本系统使用的Hadoop框架就是使用Java语言进行编写的。}

{65%:JSP全称为Java Server Pages，即Java服务器页面，}{100%:是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。}JSP是将Html插入到Java代码，构成JSP文件，用户访问时由Web服务器对编写好的JSP文件进行解析，将解析的最后最终Html页面反馈给用户。

{96%:MySQL是一个关系型数据库管理系统，由瑞典公司MySQL AB开发，目前被Oracle公司收购，在Web方面是一个应用非常广泛的关系型数据库。}{41%:其中为Java提供的JDBC类库，为使用者提供了很好的数据库连接API。}MySQL支持自定义引擎。

Linux是一个开源的类Unix操作系统，内核开源，在内核基础上存在很多发行版，如CentOS，Ubuntu，Fedora等。{71%:其实严格来说，Linux这个词本身只表示Linux内核，人们习惯的Linux指的是使用Linux内核，}并且使用符合GNU规范的软件所组成的不同发行版的操作系统。

2.2 基于内容的图像检索

{75%:基于内容的图像检索即CBIR(Content-Base Image Retrieval)，其系统建立几检索过程为，在建立图像特征库时，}{52%:系统对输入的图像进行分析并根据使用的算法提取特征并存入特征库，当用户输入一副图像进行检索时，系统对将要检索的图像进行同样的特征提取，}{70%:遍历特征库中的所有图像，进行特征相似度计算。}然后将符合阈值的图像数据返回。{100%:本质上来讲，CBIR是一种近似匹配技术，融合了图像处理、计算机视觉、图像理解和数据库等多个领域的技术成果，}{43%:其中的特征提取、索引的建立、特征相似度计算使用计算机处理完成，避免了人工参与描述的主观性。}{62%:基于内容的图像检索的核心是利用图像的可见特征对图像进行检索。}{56%:从广义上讲，图像的特征包括基于文本的特征和视觉特征两类，其中文本特征包括图像的描述、注释、名称等文本信息，图像的视觉特征则包含颜色、纹理、形状，}或者其他统计特征。{93%:此外，视觉特征又可分为通用的视觉特征和领域相关的视觉特征。}{91%:前者用于描述所有图像共有的特征，如颜色、形状、纹理等，与图像的表示具体内容无关。}{62%:后者与具体的应用紧密有关，例如人的面部特征提取或指纹特征提取等模式识别领域。}

2.3 Hadoop简介

{80%:Hadoop是有Apache公司开发的，一个分布式系统开源框架。}{95%:Hadoop主要是由HDFS和MapReduce两部分组成。}{100%:HDFS是Google File System（GFS）的开源实现。}{100%:MapReduce是Google MapReduce的开源实现。}{60%:其子项目还包括Google BigTable开源实现的Hbase，以及Hive，Zookepper等。}{100%:主要研究基础为Google的论文：Google File System（大规模分散文件系统），MapReduce（大规模分散FrameWork），}{100%:BigTable（大规模分散数据库），Chubby（分散锁服务）。}

Hadoop对数据的处理过程如图2-1所示，数据Data经过分割成线数据块，每个数据块通过Haddop系统中的Map任务处理后得到结果每个Map的结果经过Reduce的最终处理得到结果Result。

图2-1 Hadoop数据处理过程

Hadoop的集群主要由以下节点组成，{72%:如图2-2所示：NameNode中记录了文件是如何被拆分成数据块以及这些数据块都存储到了那

些DateNode节点。}{96%:NameNode保存了文件系统运行的状态信息。}{40%:DataNode中存储的是被拆分的数据块，一般DataNode与TaskNode设置为相同的节点。}{100%:Secondary NameNode帮助NameNode收集文件系统运行的状态信息。}{98%:JobTracker当有任务提交到Hadoop集群的时候负责Job的运行，负责调度多个TaskTracker。}{93%:TaskTracker负责某一个Map或者Reduce任务。}

图2-2 Hadoop的集群

3 系统分析与设计

本章介绍了系统的功能分析与设计，3.1节介绍了系统的功能以及功能分析，需要如何呈现给用户。3.2- 3.6节介绍系统结构、数据库、图像特征提取算法、检索方式等的设计。

3.1 系统基本功能分析

{76%:外观专利图像检索系统主要的功能分为，}用户登录、文本检索、图像检索、基于Hadoop图像检索、专利上传以及从国家专利局网站获取专利信息写入本地数据库。系统主要的功能用例图如图3-1所示。

图3-1 系统功能用例图

3.1.1 用户登录注册

提供如下功能:

支持用户基本注册功能，注册信息非空验证等功能。

支持基本用户登录功能，登录用户为非管理员则不能上传专利。

3.1.2 检索功能

提供传统的文本检索方式，用户输入专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等信息系统可以通过用户提交的这些信息进行专利检索模糊匹配功能。经过分析，系统文本检索行为定义如下:

当用户输入所有条件为空的时候，不返回任何结果，提示检索条件不能都为空。并跳转到检索界面。

当用户输入条件非法时候，提示输入非法，如在日期检索条件中输入非日期字符串（合法的格式为1991-1-1），提示输入格式出错，并跳转到检索界面。该过程在服务端进行处理。检索结果以序号，专利号，概述，相似度为表头的表格形式呈现并且进行分页处理。

图片检索功能为用户上传一张图片，系统对图片进行处理之后返回给用户检索结果。{45%:根据分析，规定图片类型为jpg格式，当上传图片为非jpg格式的时候，提示用户上传jpg格式的图片，当上传的图片为jpg格式时候，}向用户返回图片检索结果。检索结果以序号，专利号，概述，相似度为表头的表格形式呈现并且进行分页处理。

外观与功能上与普通图片检索一样，过程为用户上传一副图片，系统对图片进行处理之后返回给用户检索结果。{45%:根据分析，规定图片类型为jpg格式，当上传图片为非jpg格式的时候，提示用户上传jpg格式的图片，当上传的图片为jpg格式时候，}向用户返回图片检索结果。检索结果以序号，专利号，概述，相似度为表头的表格形式呈现并且进行分页处理。

3.1.3 专利上传

专利上传功能是在管理员登录的情况下，进行专利信息的录入上传功能，并且要求用户添加专利图像。

界面要求用户输入专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等信息。{56%:然后通过文件控件选取将要上传的图片，上传的图片分别要求有主视图，后视图，左视图，右视图，俯视图以及仰视图这六视图。}

{46%:上传过程要求有必填字段，其余字段可以不填写，并会对填写的字段进行格式判断，输入的日期格式是否正确，如输入的日期应该为1991-1-1的形式，}输入的编号是否为正确的位数与格式，如公开号应该为CN302270844S 的形式。图像不必要全部上传，六视图中可以上传一部分，也可以上传所有。

3.1.4 专利库构建

通过后台对国家专利局网站进行访问，获取专利局数据库中的某个二级分类下的特定数量的专利信息，并添加到数据库中，将图片下载到本地，专利数据库的构建主要是通过这种方式进行的。该过程由于国家专利局服务器响应时间等原因，用户启动新的下载任务后转到后台执行。构建过程中碰到网页格式不符合要求的，比如一项外观专利二级分类存在多个等，这里规定二级分类只能为一个，遇到这种不符合要求的专利信息，{64%:系统不会录入到数据库中。}

3.1.5 其他功能

根据洛迦诺分类，给用户显示所有外观专利主分类，并通过主分类链接，查看主分类下的二级分类。目前定义主分类为国际外观设计分类表（第八版）的分类内容，其中主分类为，如表3-1所示。系统在网页上提供帮助信息以及文档下载信息。用户可以通过帮助信息进行检索，并且下载一些文档。

表3-1 洛迦诺分类法外观专利产品大类

{98%:01 食品02 服装和服饰用品03 未列入的旅行用品、箱子、阳伞和个人用品04 刷子}

{87%:05 纺织品、人造或天然材料 06 家具 07 其他 未列入的家用物品 08 工具和金属器具}

{100%:09 用于商品运输或装卸的包装和容器 10 钟、表和其他计量仪器、检测和信号仪器 11 装饰品 12 运输或提升工具}

{99%:13 发电、配电和输电的设备 14 录音、通讯或信息再现设备 15 其他 未列入的机械 16 照相、电影摄影和光学仪器}

{100%:17 乐器 18 印刷和办公机械 19 文具用品、办公设备、美术用品及教学材料 20 销售和广告设备、标志}

{100%:21 游戏器具、玩具、帐篷和体育用品 22 武器、烟火用具、用于狩猎、捕鱼及捕杀有害动物的器具 23 液体分配设备，}{100%:卫生、供暖、通风和空调设备，固体燃料 24 医疗和实验室设备}

{88%:25 建筑构件和施工元件 26 照明设备27 烟草和吸烟用具28 药品、化妆品、梳妆用品和器具}

{99%:29 防火灾、防事故救援装置和设备 30 动物的管理与驯养设备 31 其他未列入的食品或饮料制作机械和设备 99 其他杂项}

3.2 系统架构设计

专利检索系统架构如图3-2所示，用户通过Web服务器访问特定的服务，文本检索功能直接访问数据库，通过接口对数据库中的存放的

专利信息进行模糊检索，图片检索功能则通过图像检索接口提供的两个方法，SearchImageWithHadoop()和SearchImageWithoutHadoop(),读取数据库中专利信息以及存放在文件系统中的图片信息进行图像检索过程。

图3-2 专利检索系统架构

Windows安装VmwareWorkstation，并将安装MySQL数据库，外观专利信息以及系统登录用户信息等都存放在Windows下的MySQL数据库中。{52%:Hadoop集群式搭建在VMwareWorkstation四台CentOS虚拟机。}四台虚拟机配置与角色如图3-2中所示。表3-2表示了角色和配置信息。用户通过浏览器浏览Web页面提交请求，Web服务器处理这些请求并返回相同格式的结果。{42%:主要的的检索请求包括文本检索，图像检索，基于Hadoop的图像检索。}

文本检索直接由服务器通过JDBC连接数据库，对数据库中的专利信息表进行检索条件模糊查询。然后将固定格式的检索结果通过Session返回给用户。返回的结果格式为专利编号、描述、相似度（此处相似度全部为1）的形式。可以通过专利编号的链接打开一个专利的详细页面，查看一个专利的详细信息。

普通图像检索过程，将用户提交的图像保存到本地临时路径，通过将图像路径以及保存到本地的专利库文件路径传入检索引擎进行图像特征值匹配。匹配结果按照专利编号-相似度的形式进行展示，对于同一个专利的不同视图取最高相似度进行记录，最终将去重后的结果反馈给客户端浏览器进行显示。

表3-2 系统运行环境搭建信息

HostName担任角色IP地址

Windows-HostDatabase192.168.137.1

centos.51WebService/NameNode/JobTracker192.168.137.51

centos.52DataNode/TastNode192.168.137.52

centos.53DataNode/TastNode192.168.137.53

centos.54DataNode/TastNode192.168.137.54

{43%:对于基于Hadoop的图像检索过程，将用户提交的图像缓存到本地路径，对图像进行特征提取，调用MapReduce程序，}将图像信息与特征库信息传入MapReduce程序，MapReduce程序中对图像进行特征提取，将待检索图像的特征值作为全局变量存入Configuration中，每个Map作业访问全局的待检索图像特征值，与特征库中已有的特征值进行匹配将结果返回。系统的整体框架如图3-2所示。

3.3 数据库与数据访问

本节介绍了数据库结构设计、代码访问数据库时数据模型以及数据访问对象的设计。包括数据库关系设计，数据表字段设计以及其含义，数据表外键、约束等建立。{51%:各个数据模型的含义、模型字段的意义，数据访问对象与数据模型类之间的关系等。}

3.3.1 数据库设计

本节介绍数据库中建立的表，以及表个字段的含义，主要介绍外观专利的数据库设计。设计的数据库表与表之间的引用如图3-3所示。

根据前面的定义，专利数据库的建立主要通过后台对国家专利局网站进行访问，获取专利局数据库中的某个二级分类下的特定数量的专利信息，并添加到数据库中，将图片下载到本地的方式进行。构建过程中碰到网页格式不符合要求的，比如一项外观专利二级分类存在多个等。同时考虑到数据库大数据量等问题，使用简单的分表技术，将专利图像数据库分为六张表，每张表分别存储专利六视图中的一种视图。如果某专利没有对应的视图，将不需要做任何存储处理。

图3-3 数据库表关系图

专利一级分类表，如表3-2所示。CFName表示分类的编号，由于一级分类与二级分类变化不多，所以不提供接口对该两张表进行修改操作。

表3-2 一级分类表

{87%:字段名描述类型是否为空是否为主键}

CFName分类编号varchar(16)否是

CFDescribe分类描述varchar(16)否否

专利二级分类表，各个字段及其定义如表3-3所示。

表3-3 二级分类表

{87%:字段名描述类型是否为空是否为主键}

CSName分类编号varchar(16)否是

CSDescribe分类描述varchar(16)否否

专利信息表，各个字段及其定义如表3-4及其内容所示。

表3-4 专利信息表

{87%:字段名描述类型是否为空是否为主键}

PId专利号int(14)否是

{85%:PName专利名称varchar(20)否否}

{64%:PDescribe摘要varchar(50)是否}

ApplyId申请号varchar(50)否否

{83%:ApplyDate申请日期timestamp否否}

{87%:Applicant申请人varchar(50)否否}

Applicants申请组织varchar(50)否否

{86%:Inventor发明人varchar(50)否否}

OpenId公开号varchar(50)否否

OpenDate公开日期timestamp否否

CertificationDate办证日期timestamp否否

CFId一级分类号varchar(16)否否

CSId二级分类号varchar(16)否否

专利图像表，如表3-5所示。由于考虑到每个专利都有且只有最多六张（设计系统时候做出的约定）专利图像，因此将专利图像表做了分表处理。{60%:主视图，后视图，俯视图，仰视图，左视图，右视图都单独使用一张表。结构一样。}{52%:这样方便插入，可以在大数据量的情况下提高插入速度。}

表3-5 专利图像表

{87%:字段名描述类型是否为空是否为主键}

PId专利编号varchar(16)否是

ViewPath图像存放路径varchar(16)否否

{60%:FingerPrint特征值text是否}

3.3.2 数据模型设计

图3-4及其内容所示数据模型与数据访问对象类关系图，专利具体信息的获取、插入，专利特定信息的获取，专利图像信息的获取等操作都通过数据访问对象定义的方法进行操作。图像检索对象模型，如图中PatentInfo 类图，从特征库中提取专利信息创建检索对象，三个字段Pid，FingerPrint，Distance分别表示专利的专利编号，专利特征值，专利与用户提交的检索图片的特征值距离。Pid与FingerPrint都是通过数据库中外观专利图像信息表进行获取的，而表示中间未处理相似度的特征值距离，{55%:则是通过对图像检索对象进行特征匹配时候得到的。}系统定义图像检索对象对于文本检索过程不存在相似度定义，因为对于文本难以定义标准表示检索的文本信息与外观专利的相似程度。因此对于文本检索过程定义所有的特征值距离（中间相似度）为1。而对于图像检索过程，作为中间处理对象的图像检索对象，相似度存储的是两个特征值的欧式距离等为标准化的特征值，如某两个图像的颜色相似度进行计算得到的是42.14515，此结果显然不适合直接呈现给用户。此对象不是直接作为结果呈现给用户的。

图3-4 数据模型数据访问对象关系类图

图像检索结果模型如图PatentInfos类图，检索结果与检索对象字段类型一样，但是含义不同。原本打算复用同一个类，但是这样会造成开发上的误解。因此重新定义一个结构相同的类，表示图像检索结果。该结果将作为返回值传给浏览器，用户通过浏览器看到检索结果。三个字段Pid，Discribe，Similarity分别表示专利的专利编号，检索描述，相似度。检索描述字段是为了检索过程中出现的特殊检索结果准备的，相似度则是更直观的体现检索图像与特征库中的图像的相似度，这是由于对于不同的特征，{45%:如颜色特

征与形状特征，其Distance的取值范围是不一样的，因此通过设置一个阈值，计算百分比，能更好的表现检索图像与特征库中的图像的相似度。}

外观专利图像所有信息类图，如图PatentView类图，上传专利的时作为存放专利图像路径信息的对象。包含的字段为Pid 以及各个视图的路径信息。。

计算单个图像的特征值的时候对象信息，如图中PatentViewItem类图，字段含义分别为View 为专利图像类型（左视图，右视图等），Pid是专利编号，FileName为专利存放文件名。

SearchInfo为查找专利时的查找条件，如图中SearchInfo类图。{76%:主要提供属性的Getter()，Setter()方法。}作用是用户提交文本检索时，将用户提交的检索条件进行存储，并对格式进行标准化判断。

PatentInfoA11存放了专利的所有信息，如图中PatentInfoA11类图，该类在专利插入，获取专利所有信息，显示专利详细内容的时候使用。用户提交一个专利信息插入请求，根据用户输入的专利信息创建一个PatentInfoA11对象，存储专利的信息以及专利的图像路径信息，然后进行插入。当用户查看某个专利的具体信息时，则会获取用户提交的某个专利的专利编号，获取PatentInfoA11对象，Web服务器根据PatentInfoA11对象将特定的网页信息反馈给用户，并根据PatentInfoA11的存放的图像信息，转换成图像的网络地址。

3.3.3 数据访问对象层设计

主要提供的类为 LoginDAO，PatentDAO以及ClassificationDAO，其中LiginDAO为用户注册登录接口，PatentDAO为专利信息增删改查接口，ClassificationDAO为分类查询接口。类图与方法如图3-5所示。

图 3-5数据访问对象类图

3.4 图像特征与相似度计算

{43%:颜色、形状、纹理或三者的结合是外观专利的主要保护对象，因此也可以作为外观专利图像识别的主要图像特征。}下面介绍本系统中主要使用的三种特征提取算法的选取过程与选取思路。

3.4.1 颜色特征

主要的颜色特征提取方式包括，{89%:颜色直方图(ColorHistogram) 、颜色相关图(ColorCorrelogram) 、颜色矩(ColorMoment) 、颜色一致性矢量(Color Coherence Vectors,} CCV)等。

{89%:颜色直方图是最常用也是最简单的颜色特征提取方式，描述了图像颜色的统计分布特性，其核心思想是使用颜色空间，如RGB，CIE，HSI，}{72%:HSV等空间对图像的颜色进行量化，然后统计每一个量化在整幅图像的颜色集中所占的比重。}颜色直方图在某些颜色区块整体位移的图像中，很难区分两张图片的具体区别，因此有了颜色相关图这一特征提取方法，颜色相关图其主要思想是改进颜色直方图没有有效的表示颜色之间的相对关系这一特点，用颜色对相对于距离的分布来描述图像特征信息，{97%:该特征反映了像素对的空间相关性，以及局部像素分布和总体像素分布的相关性，特点是容易计算，特征范围小，效果好。}

{94%:假设图像的记号为I(x,y)（x、y为空间坐标），颜色集合包含C1,C2,C3...Cn。}设置两种颜色之间的距离d。{69%:可以得到直方图: Bin的个数为n的平方（颜色的组合数目），对于其中Bin表示为公式2-1（其中，||*||表示像素值为Ci,}Cj的两个像素的空间距离）。

(3-1)

{97%:当设置不同的距离d1，d2，d3...dm(共D个)，Bin的维数为(n*n*D)。}{94%:再进一步，如果只考虑相同颜色之间空间关系，就称为颜色自相关图(color auto-correlogram)，Bin的维数为(n*D)。}{48%:本系统中颜色特征提取使用的就是颜色自相关图。}

3.4.2 形状特征

{69%:形状是图像最本质也是最难以刻画的最主要的特征，刻画的难度在于对图像中感兴趣目标的分割。}{90%:对形状特征的提取主要方式是寻找几何不变量。}{94%:目前用于图像检索的形状特征描述有基于边缘和基于区域的形状两种主要的方法。}{79%:基于边缘形状特征利用图像的边缘信息基于区域形状特征则是利用区域内的灰度分布信息。}专利图像不仅是对外部轮廓的表示，还应该包括专利图像的内部轮廓信息，传统的基于轮廓的形状特征描述方式，只简单利用到了图像外层的轮廓信息，这种描述方式会损失专利图像的内部轮廓特征，导致信息的不准确性，严重影响系统对形状的识别率。{53%:本系统的检索使用的主要为边界直方图(edge histogram)。}

3.4.3 纹理特征

{100%:纹理特征描述方法大致可以分为四类：统计法、结构法、模型法、频谱法。}{67%:Gabor滤波是一种频谱法提取图像纹理特征的算法。}{77%:Gabor变换本质属于加窗傅立叶变换，Gabor函数可以在频域不同尺度、不同方向上提取相关的特征。}{83%:Gabor 滤波器的频率和方向与人类的视觉系统非常相似，所以常用于纹理特征提取与识别。}{84%:在空间域，二维Gabor滤波器是一个高斯核函数和正弦平面波的乘积，它的公式化定义为（其中，λ：正弦函数波长，θ：Gabor核函数的方向，ψ：相位偏移，}{100%:σ：高斯函数的标准差，γ： 空间的宽高比）：}

(3-2)

(3-3)

(3-4)

(3-5)

(3-6)

3.4.4 图像相似度计算

{51%:本文的核心是通过基于Hadoop框架的分布式计算来解决图像检索大数据量的问题，由于建立的专利图像特征库已经将所有专利图像的特征进行提取并存储，}因此需要进行分布式计算的主要为图像的特征值匹配过程。本节中主要对两幅图像的相似性计算进行设计描述。假设待检索图像为P0，专利库中已经提取过特征的专利图像为Pi（i取值范围为1....I，I表示专利库图像数量）。用Dc（P0，Pi），Ds（P0，Pi），，Dt（P0，Pi）表示待检索图像与某个视图的颜色、形状，纹理相似度，其中对于形状和纹理的相似度分别为特征值的欧氏距离，颜色相似度为直方图相交距，并欧氏距离与相交距做归一化处理。最终计算得到的特征融合相似度D，表示为公式3-7（其中C1，C2，C3分别为相似度的权值，和为1）：

(3-7)

相似度计算中，以形状特征为主。{64%:图像颜色特征以颜色相关直方图描述，图像形状特征使用边界直方图描述，纹理特征采用Gabor滤波方式进行描述。}

3.5 检索系统

检索系统包括文本检索，图像检索和分布式图像检索。文本检索只要通过专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等字段进行数据库模糊检索，图像检索与分布式图像检索采用相同的图像检索算法，只是后者使用Hadoop框架进行分布式计算。

3.5.1 文本检索设计

文本检索功能流程为用户通过浏览器页面填写检索条件，如专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等信息，并判断输入的信息是否合法。当不合法时候，浏览器跳转回检索界面，等待用户输入合法的检索条件，当用户输入的信息合法之后，浏览器将检索条件提交给Web服务器，Web创建SearchInfo对象，传递该对象并调用PatentDAO的SearchPatent(SearchInfo si)方法获取检索结果，结果是一个List<PatentInfos>对象，将该列表对象返回给用户，用户可以通过某个专利编号链接，访问专利的详细信息。如图3-6所示，为文本检索过程时序图。

图 3-6文本检索返回结果页过程

3.5.2 图像检索

{54%:如图3-7所示，对于外观专利图像的普通检索过程，可以表示为：}

（1）读取数据库中的外观专利图像信息，将本地文件系统中的图像进行特征提取(特征包括形状、纹理以及颜色，分别将特征以字符串形式进行存储)，提取结果为“图像文件名@外观专利编号@颜色特征###形状特征###纹理特征”字符串的形式，并对特征进行存储，形成外观专利图像特征库，以文本形式进行保存。

(2)当用户提交一幅图像进行图像检索时，系统对缓存的待检索图像进行特征提取，特征提取的方式与外观专利图像库提取的特征相同，并且格式保持一致。

{56%:(3)将待检索图像的特征与外观专利图像库中的特征进行实例化，}创建检索对象通过检索方法对两个对象进行匹配(逐条读取特征库文本文件中的记录进行相似度计算，并按照相似度由大到小进行排序，去掉重复的专利)，{48%:得到图像检索结果，结果以专利编号、描述、相似度的形式进行呈现。}

图3-7 图像检索过程

{49%:基于内容的图像检索主要过程是是待检索图像与图像特征库中图像特征的相似度计算。}图像特征的包括形状、纹理以及颜色等多维特征融合。融合原理为对每种特征的相似度进行加权计算。

3.5.3 基于Hadoop的图像检索

对于一个检索的任务中大数据量问题，通过Hadoop框架的分布式使用，实现了对待检索图像特征值提取之后在特征库中进行分布式检索的过程。{84%:MapReduce是Hadoop框架的核心组成，是专门用于数据计算。}

{93%:对于Hadoop的Map函数和Reduce函数，处理的数据是键值对，也就是说 Map 函数接收的数据与输出的数据是两个参数的键值对的形式，}{88%:同样Reduce函数接收的数和输出的结果也是键值对。}因此系统中分布式计算的工作主要就是重写Map类和Reduce类的MapReduce函数。经过上文中对图像检索过程的分析，可以设计分布式图像检索过程图3-8所示。图像特征库经过系统的导出保存为文本文件形式，其中每条记录表示为一行字符串，其形式为“图像文件名@外观专利编号@颜色特征###形状特征###纹理特征”，专利的颜色，形状，纹理特征都已经在到处的过程中进行计算，这是由于外观专利的图像变动不大，特征计算一次可以永久使用，因此不必要每次都进行特征计算，避免重复的大数据量操作，降低效率。{47%:将外观专利图像特征库上传到Hadoop集群的HDFS文件系统。}方

便MapReduce程序的访问，以及可以通过HDFS的安全存储和分布式存储等特点，提高MapReduce读取特征库的速度。进行检索时，主函数设置Map类和Reduce类，并对输入的图像进行特征提取，特征提取算法使用的是与导出专利库时候使用的算法是一样的。

Map函数执行的主要功能为对输入的专利特征值进行图像特征匹配实例化，与主函数计算出的待检索图像特征值一起组成一个检索对，进行计算得出这两个特征值之间的的归一化相似度，以专利编号作为键，以特征值相似度作为值进行作业提交。合并每个Map输出的结果，作为输入传递给Reduce函数，Reduce对处理过的专利编号-相似度这一对键值对进行最终的处理，取相同编号中最大的相似度，输出得到检索最终结果。

{76%:图3-8 基于MapReduce的图像检索过程}

根据以上分布式计算设计以及系统架构设计，{41%:外观专利图像检索在Hadoop这一分布式框架下的MapReduce分布式计算过程可以用一下进行描述，过程如图3-8:}

(1)提交检索请求：Web接收到用户提交的图像信息以及分布式图像处理请求，将图像缓存单本地目录，并调用Hadoop命令，{71%:运行编译好的MapReduce程序Jar包。}命令提交后，{60%:Hadoop通过JobClient向JobTracker请求一个作业ID进行分布式图像检索作业，}JobTracker通过Hadoop命令提供的Jar文件信息，以及运行参数（运行参数包括待检索图像文件路径、特征库输入路径以及检索结果输出路径）检查作业的相关信息，通过初始化的配置，加载Mapper类和Reducer类，此过程中，待检索图像的特征值也同时进行计算，并将计算出的特征值放入全局配置中，{42%:方便每个Mapper的map函数都可以访问到，同时，计算每个TaskTracker节点外观专利特征库的输入情况，将特征库分割成大小相同的块，}{42%:将作业资源进行复制，创建TaskRunnable对象为作业运行做准备。}

{50%:（2）MapReduce初始化与分配：JobTracker接到JobClient请求的作业任务，将作业初始化之后放入一个作业任务队列中。}JobTracker将输入数据即特征库进行划分，划分成相同大小的块，每个块分配一个Job，JobTracker将每个Job和数据块尽量分配给相同的节点，即执行Job的TaskNode和存放数据的DataNode尽量保证是同一个。

{53%:（3）Map处理：每个TaskTracker接收到JobTracker分配的Job，将作业资源中的Map类进行本地化，并获取Job对应的划分的数据块，}对每个块中的每一行，作为参数执行输入给Mapper类的map函数，即继承MapRunnable的类从特征数据块中读取一个的图像特征记录，然后依次调用Mapper的map函数，map函数做的事情包括，读取全局待检索图像的特征值，创建检索对象，输入读取到的特征值，将两个特征值进行计算相似度，将结果使用专利编号-相似度的形式作为键值对输出。{72%:该输出结果将作为Reducer的输入。}{56%:每一个map过程执行完之后，发出信号通知TaskTracker任务完成。}

{79%:（4）Reduce处理：对于每一个Job，JobTracker都记录了TaskTracer和map输出的对应关系。}JobTracker创建的Reducer任务周期性访问JobTracker，请求Mapper的输出结果，当某个数据块处理完成后，Reducer的reduce函数对map结果进行处理，此过程中，对相同键的输入进行去重复操作，即检索图像对于同一个外观专利的（相同编号）的不同视图有着不同的相似度，只需要记录同一个键的最大相似度的值就可以。Reduce处理后得到的结果为一项外观专利与检索图像的相似度。Reducer将结果以键值对的形式输出到HDFS中，以文本形式进行分布式存储。

（5）检索请求的完成：所有Job完成后，JobTracker清除作业相关信息，将作业执行结果反馈给用户。

（6）后期处理：将分布式存储的文本信息获取到本地，通过读取文件将结果进行处理，经Web服务器将结果反馈给用户，通过浏览器可以查看到检索出的专利编号以及相似度等信息，访问专利编号链接，可以查看专利的详细信息。

3.6 其他功能

为保证结果的严谨性以及大数据量要求，需要对专利库进行建立，这里有两种建立专利库的方式，一是人工一条条记录的添加，通过人工的方式填写外观专利信息表单，内容包括专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类，选择本地的专利视图图片，提交插入。而是通过爬虫的方式，将国家专利局特定外观专利信息进行解

析获取，保存到本地文件系统和数据库。{88%:下面将分别介绍这两种数据库建立方式。}

3.6.1 专利上传设计

外观专利的信息主要包括内容包括专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等，以及六视图，立体图等图像信息。专利上传操作具体过程可以描述为:

（1）用户输入专利的专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等信息并选择本地图片，提交添加专利请求，将填写的信息以及选择的文件提交给Web服务器。

（2）服务器将图像进行本地存储，创建专利信息对象，将用户提交的信息进行格式化处理，并检查是否合法。返回图像存储的相对路径，并计算图像的特征值，将特征值保存到相应的外观专利图像信息对象中。

（3）通过数据访问对象将创建的外观专利信息插入数据库，返回插入的专利编号自增值，通过自增值插入对应的图像信息。

（4）将图像信息通过浏览器返回给用户，提示插入成功或插入错误等相关信息。

3.6.2 专利获取设计

通过人工方式一笔笔上传专利信息速度慢，而且无法保证上传过程中信息的准确性等问题，因此可以通过编写爬虫的方式对国建专利局网站上的外观专利信息获取，并将专利信息添加到本地数据库将图像信息保存到本地文件系统。该过程可以描述为:

（1）输入需要获取的主分类编号、二级分类编号、获取专利列表页数、每页多少条专利，向国家专利局网站的发起查询请求，获取特定分类的所有专利列表。

（2）通过获取到的专利列表的专利申请编号，请求每一条专利的详细信息页面，对页面进行解析，得到需要的Html节点的信息，以及图片链接。

（3）创建专利信息对象，将图像获取到本地文件系统，计算图像特征值，插入专利信息，遍历特定分类的查询结果列表，将所有专利信息获取到本地。

3.6.3 专利图像特征值导出

图像特征库由于专利信息几乎没有变动的特点，因此存储方式限制较低，既可以存储到数据库中，也可以导出到文本文件中，然后存储到HDFS中进行分块存储，既通过数据备份等Hadoop特性，保证数据的安全性，也方便MapReduce程序的读取方便。由于数据库设计中专利图像特征值存储使用一个字段存储存储颜色、形状、纹理的特征。因此导出使用的形式为每个外观专利图像为一行，表示格式为“图像文件名@外观专利编号@颜色特征###形状特征###纹理特征”，这样方便MapReduce对数据读取。

4 系统实现

本章节介绍了系统功能的实现，包括主要代码、代码注释、实现思路以及实现效果展示等。主要介绍的实现为数据模型及数据访问对象、图像特征提取匹配算法、文本检索以及结果展示、专利库建立（用户添加外观专利和批量获取专利信息到本地）以及图像检索的实现。

4.1 数据模型及数据访问对象

4.1.1 数据模型

数据模型实现的类分别为记录专利文字信息（如专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类）的PatentInfo类、记录外观专利所有信息（如文字、图像路径等信息）的PatentInfoA11、记录作为返回结果对象表示检索图像检索结果条目的PatentInfos类、检索结果比较大小（用来对集合排序）的PatentInfosComparator类以及其他如PatentView、PatentViewItem、SearchInfo等。

4.1.2 专利信息

该类主要功能为专利文本检索、获取专利的详细信息、专利的增删改查、专利特征值的增删、以及为其他功能获取专利特定信息提供接口如导出专利特征库所需要的获取格式化后的外观专利图像特征及其专利信息等功能。

```
public class PatentDAO {

//省略部分代码

//根据专利编号获取专利所有信息

public PatentInfoA11 getPatentInfoA11(String PId) {...}

//根据查询条件 返回查询专利编号列表结果

public List<String> Search(PatentInfoA11 info) {...}

//插入专利

public void InsertPatentInfoAndView(PatentInfoA11 patent) {...}

//插入图像及其特征值

public void InsertPatentView(PatentViewItem view) {...}

//修改图像及其特征值

public void UpdatePatentView(PatentViewItem view) {...}

//根据视图种类获取特定行数的图像信息

{59%://返回值作为输出到文本的格式化字符串}

public List<String> SearchView(String view, String limit, String row) {...}
```

4.2 特征提取与匹配

4.2.1 公共接口IImageFeature

为特征提取与匹配方式提供公共接口IImageFeature，接口主要提供的需要实现的方法有:

（1）extract(BufferedImage image)，传入BufferImage对象对图像进行特征计算提取。

（2）getDistance(IImageFeature feature)，与实现IImageFeature接口的相同类型的对象计算获取相似度，这里相似度使用欧氏距离、矩阵相交距等表示。

（3）getStringRepresentation()，获取专利特征值字符串表示。

（4）setStringRepresentation(String featureVector)，通过字符串初始化实现IImageFeature接口的类型对象的内部特征矩阵，不需要每次通过图像初始化特征矩阵，方便重复使用计算相似度。

```
public interface IImageFeature {
{100%:     public String getFeatureName();}
{100%:     public String getFieldName();}
{50%://传入BufferImage对象对图像进行特征计算提取。}
{80%:     public void extract(BufferedImage image);}
    public double[] getDoubleHistogram();
//获取相似度
{50%:     float getDistance(IImageFeature feature);}
//获取专利特征值字符串表示。
    String getStringRepresentation();
//通过字符串初始化特征矩阵
    void setStringRepresentation(String featureVector);
```

4.2.2 颜色、形状、纹理

以本系统中使用的颜色特征表示方式颜色自相关直方图为例，继承并实现IImageFeature接口的类AutoColorCorrelogram主要实现了:

（1）extract(BufferedImage image)，实现细节为:

```
public void extract(BufferedImage bi) {
final Raster r = bi.getRaster();
int[][][] hsvImage = hsvImage(r);
extract(hsvImage); }
```

（2）getDistance(IImageFeature feature)，使用的是矩阵相交距计算得到特征值的相似度。

```
public float getDistance(IImageFeature vd) {
if (!(vd instanceof AutoColorCorrelogram)) return -1;
return jsd(((AutoColorCorrelogram) vd).correlogram);
```

（3）getStringRepresentation()，获取专利特征值字符串表示。{46%:则将矩阵（二维数据）correlogram表示为“acc 4 5.0 3.0 3.0 3.0 0.0 0.0 0.0 0.0 3.0 1.0 0.0 0.0 12.0 11.0 10.0 9.0 0.0 0.0……” 的形式。}

```
public String getStringRepresentation() {
int maxDistance = this.distanceSet.length;
StringBuilder sb = new StringBuilder(numBins * maxDistance);
//writeen by dempk for add "acc" to StringRepresentation
{87%:sb.append("acc");}
sb.append(' ');
sb.append(maxDistance);
sb.append(' ');
for (int i = 0; i < correlogram.length; i++) {
for (int j = 0; j < correlogram[i].length; j++) {
sb.append(correlogram[i][j]);
sb.append(' ');
          }
```

{100%:return sb.toString().trim();}

（4）setStringRepresentation(String featureVector)，将“acc 4 5.0 3.0 3.0 3.0 0.0 0.0 0.0 0.0 3.0 1.0 0.0 0.0 0.0 12.0 11.0 10.0 9.0 0.0 0.0 0.0……”形式的字符串还原成二维数组，表示一个图像的特征信息。

4.2.3 特征融合封装

ContentBasedImageRetrieval类封装了上面介绍的颜色、形状、纹理特征提取类，并进行了获取图像特征值并进行字符串格式化、通过特征值字符串初始化相应的封装特征对象、相似度归一化并定义了特征融合方式等操作。下面简单介绍这些封装。

（1）private static float MAX_COLOR = 1等静态字段，定义了归一化时候的阈值，方便更直观的表示相似度，而不是直接使用欧氏距离或者矩阵相交距表示。

（2）public String getSourceFeature()，获取图像特征字符串表示。使用“###”隔开。

（3）public float getMixDistance()，获取融合特征相似度，通过分别获取颜色、形状、纹理相似度加权的形式计算得到，计算公式3-7。下面是一种特征相似度的归一化过程。

```
private float getDistance(int type) {

// 1 color distance // 2 shape // 4 texture

switch (type) {

case 1:

float sc = sColorFeature.getDistance(dColorFeature);

return sc > MAX_COLOR ? 0 : (MAX_COLOR - sc) / MAX_COLOR;

case 2://省略

case 4://省略
```

4.3 文本检索与检索结果

文本检索提供对专利号，专利名，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类等字段的模糊匹配检索方式。用户将合法的检索条件提交给Web服务器制动的Servlet时，Servlet获取条件参数，根据条件参数传递的值初始化SearchInfo对象，并通过PatentDAO提供的SearchPatent方法对数据库中的记录进行模糊匹配检索。{42%:下图为检索条件为专利名输入“椅子”，专利号，专利描述，申请号，申请日期，申请人，发明人，公开号，公开日期，颁证日，主分类，次级分类

等为空时的检索结果。}

图4-1 搜索结果列表

并可以通过专利编号链接访问专利的详细信息，例如查看专利1000002354的链接可以看到信息专利号： 1000002354，专利名：椅子（2），专利描述：左视图与右视图对称，省略左视图，申请号： 02358450.5，申请日期： 2002-07-19，申请人：蔡演国，发明人：蔡演国，公开号： CN3279274，公开日期：2003-02-26，颁证日期：1991-01-01的外观专利详细信息，以及其图像显示。

4.4 专利库构建

分为两种构建方式，人工上传和访问国家专利局官网通过爬虫获取需要的信息，保存到本地数据库和文件系统中。

4.4.1 专利上传

专利上传操作具体实现为：

（1）用户提交信息合法性验证，代码略。

（2）服务器将图像进行本地存储：

```
// 1.upload images
List<PatentViewItem> views = new ArrayList<PatentViewItem>();
// 图片上传路径
String uploadPath = request.getSession().getServletContext()
.getRealPath("/").replace("DesignPatent", "img")
.replace("DesignPetent", "img");
// 图片临时上传路径
String tempPath = request.getSession().getServletContext()
.getRealPath("/")+ "cache/";
// 文件夹不存在就自动创建:
{100%:if (!new File(uploadPath).isDirectory())}
{100%:new File(uploadPath).mkdirs();}
{100%:if (!new File(tempPath).isDirectory())}
```

{100%:new File(tempPath).mkdirs();}

（3）插入外观专利信息，即插入文本信息，代码为:

```
PatentDAO p = new PatentDAO();

String pid = p.InsertPatent(patent);
```

（4）通过数据访问对象将创建的外观专利信息插入数据库，返回插入的专利编号自增值，通过自增值插入对应的图像信息。将图像信息通过浏览器返回给用户，提示插入成功或插入错误等相关信息。

代码如下:

```
// 3.添加专利图片信息

if (!pid.equals("0")) {

for (PatentViewItem pv : views) {

pv.setPId(pid);

p.InsertPatentView(pv);



request.setAttribute("result", "插入成功专利编号为" + pid);

} else

request.setAttribute("result", "插入失败,请检查输入信息正确且是否存在重复");

RequestDispatcher rder = request.getRequestDispatcher("/PatentUpload.jsp");

{75%:rder.forward(request, response);}
```

如图4-2所示演示专利上传过程: 输入专利相关信息，选择本地文件系统图片，点击添加。插入成功后会提示“插入成功专利编号为XXXXXXXX”。

图4-2 专利上传演示

4.4.2 专利获取

```
public void PatentRepatileWork() {

//省略部分代码
```

```
//…..

int count = 0;

// 解析html 获取申请号

while (page >= pagestart) {

//申请号列表

patentList.clear();

patentList.addAll(ParserList(html));

// 2 遍历申请号列表 Get相应的html页面 解析得到一个专利的详细信息 并插入数据库

for (String s : patentList) {……}

}

}
```

其中解析专利详细信息到数据库并获取图片保存到本地文件系统的方法为ParserDetail()，其实现为（省略部分代码）：

```
private void ParserDetail(String html) {

Parser parser = Parser.{75%:createParser(html, "UTF-8");}

// 过滤基本信息 td 标签

NodeFilter tagFilter = new TagNameFilter("td");

// 过滤图片信息

NodeFilter imgFilter = new TagNameFilter("img");
```

4.5 图像检索

图像检索的实现包括图像特征导出（导出格式、导出位置以及导出范围）、普通图像检索、分布式图像检索实现（MapReduce程序实现）。下面依次介绍这些功能的具体实现。

4.5.1 图像特征导出

如前面分析设计部分的定义，导出格式应该为“图像文件名@外观专利编号@颜色特征###形状特征###纹理特征”的一项外观专利图像特征信息占据一行的字符串形式。导出使用PatentDAO中的SearchView(String view, String limit, String row) 方法，获取列表。剩余的工作就是使用流输出到文本文件中。

4.5.2 图像检索

图像检索类主要提供给Servlet调用，调用特定的方法可以执行不同的图像检索方式。图像检索类如写所描述，

```
public class SearchImage {
List<PatentInfos> SearchImageFromDatabase(String imagePath) {}
List<PatentInfos> SearchImageWithHadoop(String imageName) {}
List<PatentInfos> SearchImageWithoutHadoop(String imageName) {
ContentBasedImageRetrieval cbir = new ContentBasedImageRetrieval(7);
cbir.setSourceImage(imageName);
List<PatentInfos> infosList = new ArrayList<PatentInfos>();
//省略部分代码
//......
PatentInfosComparator com = new PatentInfosComparator();
{50%:Collections.sort(infosList, com);}
return infosList;
}
public static String RandTextName() {}
```

（1）SearchImageFromDatabase()，通过将待检索图片路径传递给方法在数据库中对图像检索。

（2）SearchImageWithHadoop()，将调用hadoop命令运行指定的jar包，对传入的待检索图片进行检索。

（3）SearchImageWithoutHadoop()，直接在本地文件系统中检索待检索图像，并进行去重排序等操作。

如图4-3，图4-4，图4-5所示为普通图像检索的演示过程。首先选择一张本地图片，然后检索，会返回检索结果（分布式检索操作过程与普通检索过程一致）。

图4-3 专利图像检索图像上传

图4-4 检索结果

图4-5 查看结果详细信息

4.5.3 Mapper实现

Mapper中待实现函数map()流程为：从输入数据块中读取键值对，并进行处理，设置新的键值对进行输出到context中，相应的信息存储到JobTracker中。{47%:输出的键值对可以通过combiner的进行处理。map()函数实现如下：}

```
{60%:public static class ImageRetrieval1Mapper extends}
Mapper<Object, Text, Text, FloatWritable> {
{100%:public void map(Object key, Text value, Context context)}
throws IOException, InterruptedException {
//获取待检索图像特征值
Configuration conf = context.getConfiguration();
String sourceImageFeature = conf.get("source_feature");
ContentBasedImageRetrieval1 cbir = new ContentBasedImageRetrieval1(7);
String inputLine = value.toString();
String[] inputs = inputLine.split("@");
String output = inputs[2];
String desFeature = inputs[3];
//初始化匹配对象
cbir.setSourceFeature(sourceImageFeature);
cbir.setDestinationFeature(desFeature);
float dis = cbir.getMixDistance();
FloatWritable distance = new FloatWritable(dis);
Text word = new Text();
word.set(output);
```

```
//输出键值对<key,value>
{75%:context.write(word, distance);}
}
```

4.5.4 Reducer实现

{45%:实现Reducer接口的reduce()函数，可以对map()输出的键值对进行最终处理然后输出到HDFS中。}其中reduce的输入键值对不是一个，而是使用迭代器的形式可访问的多个键值对。每个reduce输入的键值对拥有相同的键。reduce()实现如下:

```
{80%:public static class ImageRetrievalReducer extends}
Reducer<Text, FloatWritable, Text, FloatWritable> {
//继承Reducer接口
{94%:public void reduce(Text key, Iterable<FloatWritable> values,}
Context context) throws IOException, InterruptedException {
float sum = 0;
FloatWritable result = new FloatWritable();
for (FloatWritable val : values) {
sum = Math.max(sum, val.get());
}
if (0 == sum)
return;
{100%:result.set(sum);}
{100%:context.write(key, result);}
}
}
```

5 实验及测试

本章在搭建的环境下设计实验在不同数据量情况下，对普通方式的图像检索与分布式图像检索进行性能对比，比较两种方式性能（主要是对同一张图像检索的执行时间）变化曲线，通过对比分析两者之间的优势与不足。

5.1 实验环境搭建

在Linux搭建的Hadoop集群下，进行普通检索和分布式检索测试，搭建Hadoop集群环境中各台虚拟机承担的任务如表5-1及其内容所示。{57%:配置每台机器的Java运行环境，配置SSH以及Hadoop环境。}

表5-1系统运行环境搭建信息

HostName担任角色IP地址

centos.51WebService/NameNode/JobTracker192.168.137.51

centos.52DataNode/TastNode192.168.137.52

centos.53DataNode/TastNode192.168.137.53

centos.54DataNode/TastNode192.168.137.54

conf/hdfs-site.xml配置如下:

```
<?xml version="1.0"?
<?{66%:xml-stylesheet type="text/xsl" href="configuration.}xsl"?
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
</configuration>
```

mapred-site.xml配置（HDFS相关配置）如下:

```
<?xml version="1.0"?
<?{66%:xml-stylesheet type="text/xsl" href="configuration.}xsl"?
```

{71%:<configuration>}

<property>

{80%: <name>mapred.job.tracker</name>}

<value>http://192.168.137.51:9001</value>

</property>

</configuration>

{100%:core-site.xml 配置如下:}

<?xml version="1.0"?

<?{66%:xml-stylesheet type="text/xsl" href="configuration.}xsl"?

{71%:<configuration>}

<property>

{80%: <name>hadoop.tmp.dir</name>}

{80%: <value>/usr/hadoop/tmp</value>}

{94%: <description>A base for other temporary directories.}</description>

</property>

<property>

{86%: <name>fs.default.name</name>}

<value>hdfs://192.168.137.51:9000</value>

</property>

</configuration>

配置完成后启动Hadoop集群，执行分布式检索命令为runWithoutHadoop.sh image.jpg 30000形式，其中image.jpg表示待检索的图像名称，默认存放位置在脚本runWithHadoop中配置，30000表示使用大小为30000条数据的数据文件进行检索。

5.2 普通检索与分布式检索对比

5.2.1 实验

分别在图像特征数量为1条，1200条，12000条，30000条，60000条，300000条，对相同的图像进行检索，结果如图5-1所示。此时每个特征库的大小分别为5.4KB、6.3MB、63MB、158MB、315MB、1.6GB。可以看到普通模式检索方式执行时间是线性的，而分布式检索为非线性，这是由于数据块导致的Map任务个数、以及I/O等因素影响下的结果，去除虚拟机等根本原因导致的这些干扰因素之后可以看到分布式检索明显在大数据情况下明显优于普通模式下的检索方式。

图5-1 不同数据量下检索时间

5.2.2 实验结果分析

传统模式下图像的检索式在一台机器上进行计算，拥有的计算资源为一台机器的CPU，内存等供以计算，因此检索时间接近为线性。而Hadoop由于任务初始化、资源分配、网络延迟以及分块机制等占有时间会影响执行时间。影响分布式检索执行时间的主要因素如下：

（1）MapReduce执行需要对数据按照分块分配Map和Reduce任务，数据量越大，Map和Reduce任务数量也就也越大。由于Hadoop数据分块以64MB为单位，对于大小分别为5.4KB、6.3MB、63MB、158MB、315MB、1.6GB的特征库，占有的数据块分别为1个、1个、1个、3个、5个、25个，分别有相同数量的Map任务需要分配，因此数据大小过大的时，会导致太多的Map任务排队以及Reduce任务得不到执行，影响执行时间。一个Map分配与输入输出占用的时间在该环境下是非常大的。

{46%:（2）如果TaskTracker节点执行Map任务需要计算的数据块不在该节点上，需要网络传输该数据块，因此也会消耗一定时间。}

（3）实验环境搭建在虚拟机下，相当于将一台机器分割成多台使用，磁盘I/O，网络资源都将分割，无法模拟真实环境下多台机器进行分布式计算，会导致数据存在不合理性，也会影响执行时间。

（4）硬件资源瓶颈等其他因素影响。

6 结论

{63%:针对普通模式检索方式，本文提出了基于Hadoop的外观专利图像检索系统，主要的工作如下：}

{60%:（1）介绍了外观专利图像检索相关技术和Hadoop框架。}

（2）结合国内外研究，借鉴已有的研究成果，实现了一种融合颜色、形状、纹理的多特征融合特征提取匹配算法。

{43%:（3）设计外观专利检索系统，包括文本检索，图像检索以及基于Hadoop的图像检索，并对比了普通模式和分布式模式两者在不同数据量下的执行时间对比。}

（4）Web文本检索、图像检索等功能的实现。

Hadoop框架在大数据量的情况下存在很大的优势，未来工作的展望：

（1）优化图像检索算法，提高匹配准确率，同时优化检索范围。

（2）使用更简洁的Hadoop调用接口，提高执行效率，优化节点配置，提高HDFS文件系统I/O速度。

（3）优化特征值存储方式，目前的表示方式一条特征值长度为5.6KB左右，进行数据压缩可以减小数据块数量。