Nutch: A Flexible and Scalable Open-Source Web Search Engine

Rohit Khare
CommerceNet Labs
510 Logue Avenue
Mountain View, CA 94043
+1 650 714 5529
rohit@commerce.net

Doug Cutting
Nutch Organization
PO Box 5633
Petaluma, CA 94955
+1 707 696 8996
cutting@nutch.org

Kragen Sitaker
CommerceNet Labs
510 Logue Avenue
Mountain View, CA 94043
+1 415 505 1494
kragen@commerce.net

Adam Rifkin
CommerceNet Labs
510 Logue Avenue
Mountain View, CA 94043
+1 650 906 4652
adam@commerce.net

ABSTRACT

Nutch is an open-source Web search engine that can be used at global, local, and even personal scale. Its initial design goal was to enable a transparent alternative for global Web search in the public interest — one of its signature features is the ability to "explain" its result rankings. Recent work has emphasized how it can also be used for intranets; by local communities with richer data models, such as the Creative Commons metadata-enabled search for licensed content; on a personal scale to index a user's files, email, and web-surfing history; and we also report on several other research projects built on Nutch. In this paper, we present how the architecture of the Nutch system enables it to be more flexible and scalable than other comparable systems today.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**] Systems and Software—*World Wide Web*; H.3.3 [**Information Storage and Retrieval**] Information Search and Retrieval—*Search Process*; K.6.3 [**Management of Computing and Information Systems**] Software Management—*Software development*

General Terms

Design, Documentation, Experimentation

Keywords

Open Source Software, Web Search, Software Architecture

1. INTRODUCTION

Nutch is a complete open-source Web search engine package that aims to index the World Wide Web as effectively as commercial search services [9]. As a research platform it is also promising at smaller scales, since its flexible architecture enables communities to customize it; and can even scale down to a personal computer.

Its founding goal was to increase the transparency of the Web search process as searching becomes an everyday task. The nonprofit Nutch Organization supports the open-source development effort as it addresses significant technical challenges of operating at the scale of the entire public Web. Nutch server installations have already indexed 100M-page collections while providing state-of-the-art search result quality.

At the same time, smaller organizations have also adopted Nutch for intranet and campus networks. At this scale, all of its components can run on a single server.

Copyright is held by the author/owner(s).

WWW 2005, May 10–14, 2005, Chiba, Japan.

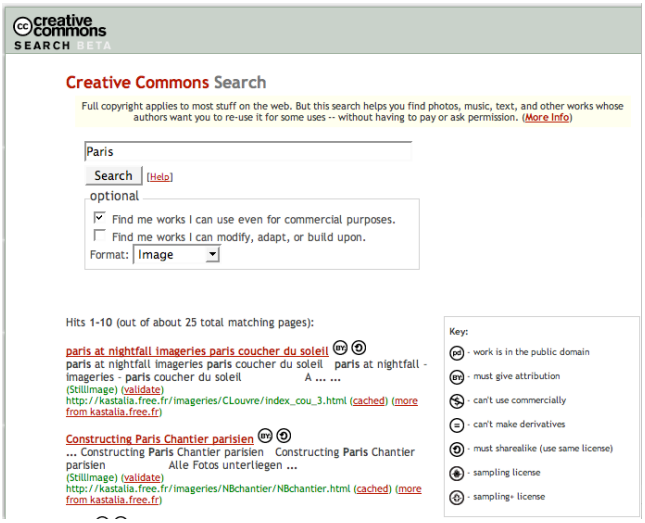

Figure 1: A Nutch-powered search engine for Creative Commons

More significantly, Nutch makes it easy to customize the search process for particular kinds of content, such as a recent Creative Commons search engine that can query using intellectual-property licensing constraints (see Figure 1).

This trend inspired our own experiment to apply Nutch at personal scale at CommerceNet. We hypothesized that the same architecture used to run public search engines on dedicated hardware ten years ago might be ready to run as a background task on a laptop. We also evaluated how Nutch could adapt to the distinct hypertext structure of a user's personal archives.

We also suggest that there are intriguing possibilities for blending these scales. In particular, we extended Nutch to index an intranet or extranet as well as all of the content it links to. This sort of 'neighborhood' search helps visitors explore an organization's collective memory.

Furthermore, many other academic and industrial research projects are building upon Nutch to explore aspects of Web search, from integration with document summarizers to teaching graduate courses in data mining.

This paper motivates Nutch in the context of other hypertext search research; describes its internal architecture and behavior; evaluates how that architecture supports or conflicts with the requirements of global-, local-, and personal-scale hypertext search problems; reports on how it has been adopted by other researchers and users; and reflects upon future directions.

Table 1: Comparing open-source text indexing packages.

	Glimpse	Namazu	MG4J	Lucene
Key Feature	Suffix arrays	Japanese support	Compact storage	Native Unicode
License	Nonprofit use	GPL	LGPL	Apache
Active	No	No	Yes	Yes

Table 2: Comparing open-source hypertext indexing packages.

	WebGlimpse	ht://Dig	SWISH-E	Nutch
Key Feature	Suffix arrays	Simplicity	Metadata	Ranking, Excerpts
License	Nonprofit use	GPL	GPL/LGPL	Apache
Active	No	No	Yes	Yes
Crawling	Local file-system only	Intranet only	Intranet only	All scales
Caching	No	No	No	Yes
Clustering	No	No	No	Yes
Link Rank	No	No	No	Yes

2. BACKGROUND

The Nutch project grew out of the first author's experience developing Lucene [13], a Java text indexing library that became part of the Apache Jakarta open source project. Lucene was his fourth major information retrieval system, building on over fifteen years of work for Excite's search engine [14]; Apple's first indexing service, V-Twin [48] (now succeeded by SearchKit [3]); and work at Xerox PARC.

A decade ago, the first public Web search engine, Webcrawler [44], was built on the NeXTStep IndexingKit. Following that example, Nutch has also turned Lucene into a Web search engine by adding crawling, parsing, graph analysis, and a user interface.

2.1 Desiderata

When AltaVista launched in 1995, whole-Web search was just blue-sky research speculating that it might be computationally feasible to index the whole Web; today it is a global industry worth more than a hundred billion dollars [6]. In less than ten years, Web search has evolved from an intellectual exercise in computer science to an everyday practice in the developed world – more than three-quarters of American internet users use a search engine each month. Today, Web search is redefining the frontiers of extremely-large-scale distributed systems.

> *Nutch provides a transparent alternative to commercial web search engines. Only open source search results can be fully trusted to be without bias. (Or at least their bias is public.) All existing major search engines have proprietary ranking formulas, and will not explain why a given page ranks as it does. Additionally, some search engines determine which sites to index based on payments, rather than on the merits of the sites themselves. Nutch, on the other hand, has nothing to hide and no motive to bias its results or its crawler in any way other than to try to give each user the best results possible.* — *www.nutch.org*

Transparency is essential to the operation of a free society, and it is as essential for societal infrastructure software as anything else. Ken Thompson's Turing Award lecture [52] is a classic warning that it can be unsafe to rely on any software without verifying it all the way down to the bare metal. Can we prove *or* disprove claims of, say, right-wing bias in automated news mining [30].

The Nutch team is committed to a transparent development process. However, they have chosen to encourage developers with a liberal license similar to that of the Apache Software Foundation, rather than legally mandating source sharing.

> *Nutch aims to enable anyone to easily and cost-effectively deploy a world-class web search engine. This is a substantial challenge. To succeed, Nutch software must be able to:*
> - *fetch several billion pages per month*
> - *maintain an index of these pages*
> - *search that index up to 1000 times per second*
> - *provide very high quality search results*
> - *operate at minimal cost* — *www.nutch.org*

Nutch has also set clear technical goals for itself, related to *feasibility* (ability to it operate at this scale at all); and *economy* (ability to operate efficiently enough to be practical).

All the same, the task itself is becoming easier as the community matures and computers grow steadily more powerful. Apparently, even a lone college graduate can create a profitable 600M-page search engine from scratch in C++ today (Gigablast, [27]). The nonprofit Internet Archive even passed the 10B-page mark [43]!

2.2 Related Work

Many search engines have source code available for at least non-commercial use, spanning the scale from simple text indexers to full-fledged web search engines; they also compete with proprietary software and appliances for the so-called "Enterprise Search Platform" market [22].

Open-Source Text Indexing. Table 1 surveys source-available full-text search systems in use today: Glimpse [34], Namazu [38], the Managing Gigabytes/MG4Java system [7], and Lucene [13].

Lucene is an unusually flexible text search engine: it indexes incrementally, with small indexes (~30% of original); runs in very little RAM (independent of corpus size); supports arbitrary boolean queries across multiple user-defined document fields; returns results ordered by relevance; and supports user-defined lexical analysis and stemming algorithms. It is well-regarded enough that it has even been re-implemented in several more languages: C++ (CLucene), C# (NLucene, Lucene.net), Python (Lupy), Perl (Plucene, Lucene::QueryParser), and Ruby.

Open-Source Hypertext Indexing. Searching the Web requires a text-search engine, but it also requires an efficient and robust web crawling system, capable of saturating available network bandwidth, refreshing pages intelligently, obeying the Robot Exclusion Standard, choosing high-quality new pages to explore, using disk space efficiently, and continuing to function even in the face of unexpectedly large documents, slow Web servers, broken links, thousands of URLs for the same document, and corrupted documents. It must also let users invoke a text search and view the results. Finally, the hypertext nature of the Web offers additional opportunities for result improvement over a simple text-search engine, including link-text indexing and link structure analysis. It's a major engineering effort, even starting with a text indexer.

A number of Web search systems are in wide use today, including WebGlimpse (though it has significant license constraints) [33], ht://Dig [36], Swish-E [46], and Nutch [12]. Nutch offers high-quality excerpting, link-structure analysis, link-text indexing, relevance-ordered search results, content caching, and built-in support for parallelization on a cluster in order to index crawls two orders of magnitude larger than any of these other systems.

3. ARCHITECTURE

Nutch has a highly modular architecture that uses plug-in APIs for media-type parsing, HTML analysis, data retrieval protocols, and queries [40]. The core has four major components:

Searcher: Given a query, it must quickly find a small relevant subset of a corpus of documents, then present them. Finding a large relevant subset is normally done with an inverted index of the corpus; ranking within that set to produce the most relevant documents, which then must be summarized for display.

Indexer: Creates the inverted index from which the searcher extracts results. It uses Lucene storing indexes.

Database: Stores the document contents for indexing and later summarization by the searcher, along with information such as the link structure of the document space and the time each document was last fetched.

Fetcher: Requests web pages, parses them, and extracts links from them. Nutch's robot has been written entirely from scratch.

Figure 2 outlines the relationships between elements that refer on each other, placing them in the same box, and those they depend on in a lower layer. For example, `protocol` does not depend on `net`, because `protocol` is only an interface point for plugins that actually provide much of Nutch's functionality.

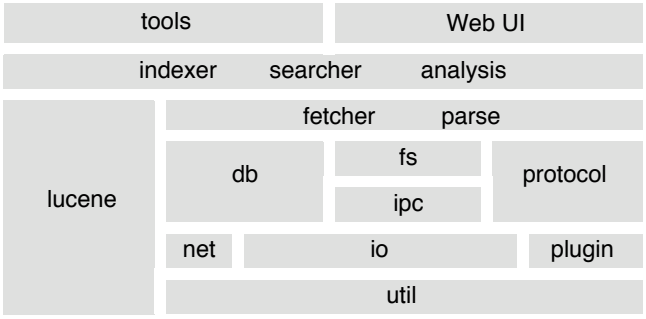

Figure 2: Layer diagram of Nutch package dependencies.

3.1 Crawling

An intranet or niche search engine might only take a single machine a few hours to crawl, while a whole-web crawl might take many machines several weeks or longer. A single crawling cycle consists of generating a fetchlist from the webdb, fetching those pages, parsing those for links, then updating the webdb.

In the terminology of [4], Nutch's crawler supports both a *crawl-and-stop* and *crawl-and-stop-with-threshold* (which requires feedback from scoring and specifying a floor).

It also uses a uniform refresh policy; all pages are refetched at the same interval (30 days, by default) regardless of how frequently they change There is no feedback loop yet, though the design of `Page.java` can set individual recrawl-deadlines on every page).

The fetching process must also respect bandwidth and other limitations of the target website. However, any polite solution requires coordination before fetching; Nutch uses the most straightforward localization of references possible: namely, making all fetches from a particular host run on one machine.

3.2 Indexing Text

Lucene meets the scalability requirements for text indexing in Nutch. Nutch also takes advantage of Lucene's multi-field case-folding keyword and phrase search in URLs, anchor text, and document text. The typical definition of Web search does not require some index types Lucene does not address, such as regular-expression matching (using, say, suffix trees) or document-similarity clustering (using, say, term-vectors).

3.3 Indexing Hypertext

Lucene provides an inverted-file full-text index, which suffices for indexing text but not the additional tasks required by a web search engine. In addition to this, Nutch implements a link database to provide efficient access to the Web's link graph, and a page database that stores crawled pages for indexing, summarizing, and serving to users, as well as supporting other functions such as crawling and link analysis. In the terminology of one web search engine survey [4], Nutch combines the text and utility databases into its page database.

Nutch also has to add support for HTML extraction to Lucene. When a page is fetched, any embedded links are considered for addition to the fetchlist and that link's anchor text is also stored. What is eventually indexed by Lucene is only the text of a Web page, though: while a high-fidelity copy is stored in the page database, font, heading, and other structural information does not propagate to the Lucene indexing layer.

3.4 Removing Duplicates

The `nutch dedup` command eliminates duplicate documents from a set of Lucene indices for Nutch segments, so it inherently requires access to all the segments at once. It's a batch-mode process that has to be run before running searches to prevent the search from returning duplicate documents. It uses temporary files containing the 5-tuple `(MD5 hash, float score, int indexID, int docID, int urlLen)` for each page.

```
to eliminate URL duplicates from a segmentsDir:
    open a temporary file
    for each segment:
      for each document in its index:
        append a tuple for the document to the
            temporary file, with hash=MD5(URL)
    close the temporary file
    sort the temporary file by hash
    for each group of tuples with the same hash:
      for each tuple but the first:
        delete the specified document
            from the index
```

Listing 1: Pseudocode for removing pages with the same URL.

Presumably the documents with the same URLs were fetched at different times, so Nutch tries to sort the records so that the ones for the newest fetches come first. A second pass, using `hash=MD5(content)` and slightly different sorting rules, eliminates multiple URLs for the same document from the index.

3.5 Link Analysis

Nutch includes a link analysis algorithm similar to PageRank [42] It even uses .15 as the random-jump probability (called `DECAY_VALUE` here). It is performed by the `DistributedAnalysisTool`; even the single-machine `LinkAnalysisTool` merely calls into it. It uses an iterative method, rather than, say, solving for a matrix eigenvalue directly [29]. Nutch has already demonstrated the ability to harness multiple servers to compute link ranking for 100M-page subsets of the World Wide Web.

Distributed link analysis is a bulk synchronous parallel process. At the beginning of each phase, the list of URLs whose scores must be updated is divided up into many chunks; in the middle, many processes produce score-edit files by finding all the links into pages in their particular chunk. At the end, an updating phase reads the score-edit files one at a time, merging their results into new scores for the pages in the web database.

Distributed analysis doesn't use Nutch's homegrown IPC service; like fetching, work is coordinated through the appearance of files in a shared directory. There are better techniques for distribution (MapReduce, [15]) and accelerating link analysis [26].

3.6 Searching

Nutch's search user interface runs as a Java Server Page (JSP) that parses the user's textual query and invokes the search method of a `NutchBean`. If Nutch is running on a single server, this translates the user's query into a Lucene query and gets a list of hits from Lucene, which the JSP then renders into HTML. If Nutch is instead distributed across several servers, the `NutchBean`'s search method instead remotely invokes the search methods of other `NutchBeans` on other machines, which can be configured either to perform the search locally as described above or farm pieces of the work out to yet other servers.

Distributed searching is built on top of a custom SIMD cluster parallelism toolkit in the package `net.nutch.ipc`, which provides a blocking 'call' method to perform the same operation in parallel across several servers, then gather the results together afterwards. In the terminology of [49] Nutch uses *partition-by-document*, where all the postings for a certain document are stored on the same node. Consequently every query must be broadcast to all nodes. In Nutch, the `net.nutch.searcher.DistributedSearch.Client` class provides this functionality; it implements the same `net.nutch.searcher.Searcher` interface that Nutch uses to invoke searches on locally-stored segments, represented by `net.nutch.searcher.FetchedSegments` objects.

3.7 Summarizing

Summaries on a results page are designed to *avoid* clickthroughs. By providing as much relevant information as possible in a small amount of text, they help users improve precision.

Nutch's summarizer works by retokenizing a text string containing the entire original document, extracting a minimal set of excerpts containing five words of context on each side of each hit in the document, then deciding which excerpts to include in the final summary. It orders the excerpts with the best ones first, preferring longer excerpts over shorter ones, and excerpts with more hits above excerpts with fewer, and then it truncates the total summary to a maximum of twenty words.

Currently, Nutch considers equal-length excerpts containing the same number of hits to be duplicates and only keeps the first one for the summary, resulting in a predictable, if less useful, display.

Future content-type-specific summarizer modules could provide specialized presentations for email messages; but that would require changing the interface to the summarizer.

To distribute the workload, summarizing hits for the results page requires extracting excerpts from the original documents. Since those documents are stored at various nodes in the Nutch cluster, they must either be copied across the network and summarized on the machine conducting the search, or summarized on the machines that store them and the summaries copied across the network; Nutch chose the latter strategy.

4. GLOBAL SCALE

The original motivation for the Nutch project was to provide a transparent alternative to the growing power of a handful of private search services over most users' view of the Web. However, as Nutch has been adopted with greater enthusiasm by smaller organizations, the Nutch Organization has de-emphasized operating a multi-billion-page index in the public interest.

4.1 Experience from Yahoo! Research Labs

When the Nutch project was publicly announced in June 2003, Overture Inc. sponsored a public trial of a 100M-page index that was online at Yahoo! Research Labs until November 2004 [41]. It was based on a very early version of the code and had not been re-crawled or maintained since October 2003; a recent search for "http" suggested that only 60M pages remained online and link-analysis scores had only been applied to portions of the index.

The major contribution of this experiment was to establish that a free Java-based package was capable of scheduling, fetching, and indexing a very large crawl. It is instructive to consider its strengths, weaknesses, and omissions. To its credit, Nutch did complete a significant crawl using the Open Directory Project as a seed list, and it manages a cache of the content it fetched. This experiment did not address concurrent query rates; while it established that a Nutch query can be sent to several independent back-end query servers and the results can be merged and excerpts generated within a second, there was no performance analysis of a separate web server farm that would be necessary to process thousands of concurrent queries against the same index. The goals that were set for testing at global scale were:

- *Scale-up fetching:* multiple, simultaneous page fetches (>100 pages/sec/node, or 10M/day).
- *Scale-out indexing:* parallel, distributed webdb update that can process 100M entries (>100 pages/sec/node).
- *Scale-out querying:* each search node would have to process 1-40 queries/sec against 2M-20M pages.

Ironically, one of the weakest points of this Nutch experiment is its signature feature, the ability to "explain" its rankings. A search for the word "nutch" yielded a seemingly bizarre top-ranked hit: CBS Sportsline. However, the excerpt indicated the term is used six times on the page body. Since it is a dynamically generated news site, we compared it to the cached source file that was indexed originally instead, but the term still didn't occur there.

It turns out that this was an interaction between the Nutch crawler, the Mircosoft Web server that Sportsline was using, and how pages were cached for later display. Since some of its "browser-detection" code in an HTML page echoed the User-Agent string, the term "nutch" occurred multiple times – but the <SCRIPT> elements were also being stripped from the cached pages.

Furthermore, the "explain" link itself brought up a page that has not improved significantly since: a very cryptic equation involving tf (term frequency), idf (inverse document frequency), and "boost" factors. The most a savvy web surfer could probably garner from this is the top-level breakdown of how often the term occurs within the page vs. in the anchor text of links to that page. The link-graph analysis scores, in particular, are inconsistent in ways that suggest the collection has only been partially analyzed: some pages still have the default score for pages found via same-site links (0.5) or cross-site links (2.0). The bottom line, though, was that searching for "nutch" would only return a list of the most popular Web sites running that particular Web server package!

```
public static void walk(Node doc, URL base, Properties metadata) {
  // walk the DOM tree, scanning for license data
  Walker walker = new Walker(base);
  walker.walk(doc);
  // interpret results of walk
  String licenseUrl = null;
  if (walker.rdfLicense != null) {         // 1st choice: subject in RDF
    licenseUrl = walker.rdfLicense;
  } else if (walker.relLicense != null) {  // 2nd: anchor w/ rel=license
    licenseUrl = walker.relLicense.toString();
  } else if (walker.anchorLicense != null) {// 3rd: anchor w/ CC license
    licenseUrl = walker.anchorLicense.toString();
  }
  if (licenseUrl != null) {
    metadata.put("License-Url", licenseUrl);
  }
}
```

Listing 2: Simplified HTML parsing plug in code for extracting Creative Commons licenses.

To be sure, the two-year old Overture experiment is not the only evidence of a 100M-page service. The KNOWITALL research project [18] at the University of Washington operates an even larger crawl today for its students as well as some public services [10].

4.2 Global-scale Challenges

The essential challenge of a global-scale Web search service is ranking relevant results – especially in the face of organized resistance from so-called "link-spammers." Nutch includes much of the machinery to operate a global-scale crawling and indexing operation. It even supports a basic level of whole-Web link graph analysis along with many tunable parameters for scoring results; current work with Nutch even includes automatic tuning based on user feedback [25]. But precisely because Nutch is an open project, global Web search engines based on it such as Mozdex [37], risk escalating an "arms race between search engine and spammer" [12].

Some "search-engine optimization" techniques are already well-known; see [24] for an extensive discussion. It is believed that substantial portion of the R&D budgets of commercial search services are dedicated to this problem [6]. Consider even a simple cycle of escalation that Nutch would have to address:

1. Webmaster repeats desired terms on a page. Search service counters by eliminating consecutive occurrences of a term.
2. Webmaster includes irrelevant (but popular) passages as "invisible" HTML. Search service counters by parsing enough HTML to determine what a browser would present.
3. Webmaster intersperses that term amongst "normal" text. Search service counters by comparing statistical distribution of terms to language-specific benchmarks.
4. Webmaster detects crawler's User-Agent and sends decoy text instead. Search service counters by comparing variance of page content across browser types.
5. Webmaster sets up a "link farm" of interconnected sites to increase incoming link count. Search service counters by identifying statistically-unlikely graph structures…

However, as this cycle continues, each counter-move would become clearly visible in the Nutch source code. However, if the success of the open-source development community with security breaches, bug detection, and email spam filtering are any indication, openness works in practice (if not in theory).

5. LOCAL SCALE: INTRANETS

An intranet is not merely a small subset of the global Web. On one hand, it may be easier to crawl because it can be spam-free and can be better structured (which suggests using an interactive crawl planning tool like SPHINX [35]). Furthermore, a simple intranet installation of Nutch is much easier to configure, install, and maintain on a single server than a distributed global-scale cluster. On the other, smaller collection sizes (<1M pages) with sparse internal linking can inhibit link analysis and places relatively greater emphasis on anchor text indexing.

5.1 Experience at Oregon State University

When the Open Source Lab at Oregon State University began testing Nutch in mid-2004, the campus had already installed a Google Search Appliance in 2002 [22] (before that, they used another commercial solution from Inktomi since 1998). In August 2004, OSU replaced it with Nutch, citing its "flexibility and extensibility."[1]

In June, their staff compared a search quality comparison and found Nutch to be equal to Google's quality [8]. Google's product included a synonym feature called KeyNames that works like a sponsored link to promote a specific target page, but it was adopted spottily. Nutch, for its part, returned too many similar pages. Aside from those issues, though, they found the majority of the sample queries awarded a perfect 10 to both engines in a subjective evaluation. While they noted that Nutch was weaker on spam detection and elimination, its greater transparency influenced the final decision.

5.2 Experience at CommerceNet

We thought it would be useful to crawl not only the content on CommerceNet's own sites, but also the pages our collection linked to. This new blended-scale engine at http://-labs.commerce.net:8180/ has proven particularly useful because its baseline crawl includes our collaboration tools, a blog and a wiki that link to many other sites and articles we collectively found noteworthy.

We changed 19 lines of code in four files to add this feature; eight of these were just to add the property `db.one.hop.outside.-desired` to the `nutch-default.xml` configuration file.

This uncovered one serious problem, apart from the fact that this solution could not crawl *n* degrees beyond the "fence" specified in an URL filter regular expression. Namely, our blog contains enough comments spam that a noticeable fraction of the collection was unwanted or obscene. This is an annoyance in Web search, but has much more serious consequences for personal-scale search. Letting a fetcher blindly follow a link in an *email* spam can serve notice that an address is working (and apparently belongs to a very gullible consumer!).

5.3 Intranet-specific Challenges

When retreating from the public sphere to private uses, new security and privacy issues come into focus. An organization's intranet implies a boundary between authorized insiders and outsiders; this can also extend to cover access rights to other information services. At CommerceNet, we were not able to delegate credentials to the Nutch fetcher to retrieve content from some local password-protected servers. Furthermore, if such support were built-in, it might create the presumption that Nutch should continue to enforce a security policy on the content once cached and indexed locally.

That suggests an intranet Nutch deployment may need individual user authorization to access subsets of a collection. An efficient suggestion for extending a UNIX filesystem permission-like policy to Nutch might be to add `user` and `group` fields to Lucene's document indices with tokens for each user. Then a query filter could force the addition of restrictive clauses to every search based on users' credentials. This is similar to a query-expansion technique used in the next section for Creative Commons.

6. LOCAL SCALE: COMMUNITIES

6.1 Experience with Creative Commons

Creative Commons is a nonprofit organization that promotes the use of openly licensable content using a small set of codes for indicating attribution, royalty, and other policy constraints. It helps users find content they can reuse, like songs that can be sampled, art that can be clipped, and text that can be excerpted:

"Show me all photos of Paris that I can make derivative works from and sell afterwards"
"I'd like to find songs about Love that I can remix"
"Find me some CC-licensed music videos"
"Has anyone ever written about the Statue of Liberty?"

It recently announced a nearly 1M-page Nutch-based search engine that crawls the public Web looking for CC-licensed content, indexing license properties and text, and making it all searchable with a visual icon language integrated into its search results (see Figure 1).

```
public static String FIELD = "cc";
public Document filter(Document doc, Parse parse, FetcherOutput fo) {
    String licenseUrl = parse.getData().get("License-Url");
    if (licenseUrl != null) {
      /* Add the features represented by a license URL.  Urls are of
       * the form "http://creativecommons.org/licenses/xx-xx/xx/xx",
       * where "xx" names a license feature. */
      URL url = new URL(urlString);
      // tokenize the path of the url, breaking at slashes and dashes
      StringTokenizer names = new StringTokenizer(url.getPath(), "/-");
      if (names.hasMoreTokens())
          names.nextToken(); // throw away "licenses"
      // add a feature per component after "licenses"
      while (names.hasMoreTokens()) {
          doc.add(Field.Keyword(FIELD, names.nextToken()));
      }
    }
    return doc;
}
```

Listing 3: Simplified IndexingFilter code for the Creative Commons plug-in.

6.1.1 How the Creative Commons Plug-in Works

This demonstrates how easily Nutch can be extended to accommodate metadata that would be hard to with any other search engine. There are only about 500 lines of CC-specific code.

The Creative Commons plugin includes three main pieces: the indexing filter, the HTML parsing filter, and the query filter. The HTML parsing filter, along with any other registered HTML parsing filters, has the opportunity to add metadata or other data to the parse data that gets serialized and ultimately handled by the indexing phase. Similarly, the indexing filter adds fields to a Lucene Document after examining Nutch parse and fetcher output, and the query filter adds clauses to a Lucene `BooleanQuery` after examining a Nutch `Query`. In fact, plug-ins do nearly all of the construction of parse data and Lucene documents and queries; for example, there's a `BasicIndexingFilter` plugin that populates the "url", "content", and "anchor" fields of the Lucene document index, and a `BasicQueryFilter` plugin that searches them (see Listing 2).

The "Walker" here is a visitor class that walks a DOM tree, searching for certain kinds of links used for Creative Commons licenses. The "filter" method is the external interface for HTML parsing plugins (see Listing 3).

6.2 Community-specific Challenges

Unique user communities have idiosyncratic requirements. Semantic tagging systems such as SemTag [16] could be used in conjunction with shared community knowledge to offer richer customization of Nutch components such as analysis.

Metadata Communities have unique document metadata, as the Creative Commons experience demonstrated. Unlike PICS, though, since true parameterization points towards typed fields and relational databases; the sweet spot for Nutch is metadata that fits within Lucene's model of the world, namely keyword/full-text search of fields.

Semantics Ultimately, what ties together communities is common semantics, which in turn can be used to offer more comprehensive and custom search. Paul Ford pointed out [20] that, "Lucene's relationship to the Semantic Web may seem unclear -- after all, the Semantic Web is about resource discovery by analyzing triples, not full-text search. However, along with URIs, literal values make up a good portion of RDF, and Lucene offers an easily embeddable means to provide for search within those literal values. Most notably, Lucene is integrated into Kowari, where it allows for combinations of graph-based querying and old-fashioned keyword lookup."

Internationalization Nutch supports automatic identification of and search on document language, its user interface is available in 13 languages, and it supports search in most Western languages reasonably well. However, it still needs significant work to improve its search quality in many languages. Special code is needed to perform morphological analysis of CJKV (Chinese, Japanese, Korean, and Vietnamese) languages and agglutinative languages such as German, and language-sensitive stemming is

extremely important for highly inflected languages such as Finnish.

7. PERSONAL SCALE

The World Wide Web can also be a lens for viewing an individual's personal information management needs. Many desktop applications available on all major PC operating systems that can search files on a hard drive (e.g. [17]). We believe that files-in-folders is only one approximation of a much richer Personal Web that connects files, email, attachments, contacts, and, yes, all the public Web pages read before in a socially-connected graph. As PCs grow more powerful, it may become reasonable to accommodate an efficient implementation of the entire machinery of a Web search engine as a background task.[2]

7.1 Experience on a Local Filesystem

Our first experiment was to expose a local filesystem using Apache[51], and seed Nutch with a localhost URL. A hard drive controlled by the user, however, is quite different from the hostile, messy public Web, and several Nutch self-protection mechanisms interacted poorly with it:

- It didn't crawl all the files in large directories;
  The `db.max.outlinks.per.page` default is 100.
- It didn't index complete documents;
  The `http.content.limit` default is 64K.
- It indexed entire mailboxes as a single document;
  A replacement mail-specific parser would make two passes, to extract the link structure and then to "fetch" each message.
- It took days to complete a crawl;
  Out of politeness, the minimum delay between fetches from the same webserver is 1 second.
- It duplicated every file;
  Since Web sites are unreliable, Nutch quite sensibly caches copies of all fetched content.
- It can't forget files easily;
  Private information that inadvertently gets indexed may remain visible even after the original is deleted.

These problems are all easily addressed — many were solved by using a file: protocol plug-in rather than using HTTP — but this litany of problems highlights how different the personal-scale is from the global-scale problem. There are also benefits to using global-scale techniques in this domain, though. A minor example is that since Lucene indexes words in an URL field along with document text, a single interface can search for both matching filenames and file contents, which are often two separate modes on other search tools (e.g. grep vs. locate [32]). A more promising one is link analysis: while explicit hyperlinks between local files are too sparse to apply Web techniques effectively, there is a much richer social connectivity graph between conversations held in email and through attachments that appears promising.

Keeping copies of everything is unnecessary when the original files are at hand – there's even an (as yet unimplemented) preference-setting in the code which clearly suggests interest (`file.content.ignored` in `nutch-default.xml`).

Installation is another challenge. Nutch itself is only 500K, hardly bigger than Google Desktop Search at 400K [45]. Running it, though, requires a user interface, which needs servlet container,

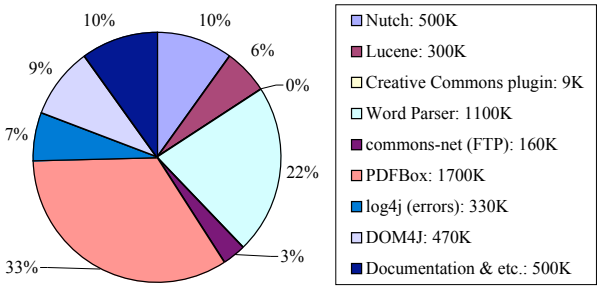

Figure 3: Component sizes in a Nutch installation.

such as Tomcat (another 1.8MB); and a slew of other libraries that inflate the Nutch web archive file to more than 5MB (Figure 3).

Miniaturizing a personal-scale Nutch could be done by replacing components with services provided by the desktop OS. By way of comparison, Google Desktop Search appears to use COM to call existing Windows services for text extraction and network access.

7.2 Personal-scale Challenges

A mailbox is an archetypal example of a document with many authors. Many other artifacts that are not traditional hypertext documents also have implicit authorship links: instant messages, buddy lists, version-control change logs, blog entries, calendar items, and photos are only a few examples.

Email quotation is the equivalent of a hyperlink citation. A single email message can include text from multiple authors. Unlike the Web convention that citation is an endorsement, research into community-clustering shows that on USENET discussions, quotation should be interpreted as *dis*approval [1].

There are several limitations to analyzing email messages using tools such as pipermail or MHonarc:

- There is no link between articles to the individuals that wrote them — mailto: links aren't followed and ranked like http:.
- "Thread," "Date," and "Author" index pages only serve to confound the ranking of individuals, threads, and messages by combining *all* of them on an equal basis in one file.
- There is no immediately obvious way to strip out the purely "navigational" link, which tend to emphasize adjacent messages in time rather than by thread.
- There is no obvious way to "backpropagate" scores from the sites an email message cites. A message that cites the New York Times should be more valuable, no less. With a copy of the entire World Wide Web link graph, the hubs-and-authorities model would address this [28].
- There is little precedent for increasing the rank of a message the more often a user refers back to it, or based on the fraction of included links the user has clicked through.

Even Zoë [50], which also uses Lucene to power a new kind of blog-like user interface to email and represents social connections with links, does not take advantage of link analysis algorithms to rank messages. We wrote a script that turned 1GB of personal email into ~150K files for each message *and* correspondent. By linking threads, authors, and readers (and no navigation), we were able to experiment with Nutch's analysis of a "social network."

8. REFLECTIONS

Nutch is simultaneously more transparent, flexible, and scalable than other Web search solutions. It has already demonstrated three orders of magnitude greater capacity than other open source engines; it is more flexible than earlier open-source information retrieval systems; and it is more transparent and flexible than proprietary Web search sites.

8.1 Flexibility

Nutch's value as an experimental platform has already been borne out by its increasing adoption in niche communities, digital libraries, classrooms, and academic research projects.

Webmasters: By empowering small websites to not only add search capabilities to their own content, but also easily crawl any part of the Web they or their readers care about, Nutch has enabled a number of new niche search services. The Linux Search Engine[3] is a focused subset of ObjectsSearch,[4] a commercial company running a public search engine over 1M-pages. This company has both embraced the Nutch's mission of transparency — uniquely amongst Nutch adopters, they left the "explain" links on — but also extended the search process in innovative new ways. They offer clustering, thumbnail previews of pages, and "quickinfo," a named entity extraction service for locating people and companies. Another example of a Nutch-driven service that even offers anchor-text indexing of images is Playfuls, a gaming-specific service with a crisp mission:

> *"Playfuls.com is a specialized gaming search engine. We don't search useless and off-the-topic websites. You can be sure that all our results come from gaming websites… Our news search database is updated 12 times a day, and our web search database is updated every 48 hours."*[5]

An even smaller collection of interest is SearchMitchell (Figure 4), a new-fangled old-fashioned small-town search engine. At the other extreme, MozDex has taken on the "public television" aspect of Nutch's mission[6] and built a 100M-page collection with advertising revenue (and with the "explain" links intact). They use AMD Opteron servers for back-end searching and crawling and AMD Athlon XPs for front-end JSP servers (currently down for maintenance).

Librarians: The digital library community has its own interests in large-scale information retrieval. The California Digital Library eXtensible Text Framework (XTF [23]) and the National Science Digital Library have also built search engines based on Lucene.[7] In NSDL's case, they moved off of a commercial solution for the three classic open-source rationales: flexibility, low cost, and stability [39]. At the University of California, they modified the indexer to process 200-word chunks that make it easy to present hits in context and navigate directly into large documents.[8]

Arguably, the ultimate scaling challenge in library science is keeping up with electronic media, and the Internet Archive's effort to catalog audio, video, and Web content for future generations motivates several Nutch-based experiments. They have URL-based retrieval of 40B past page versions,[9] had a (currently-offline) search over an 11B-page archive,[10] and are experimenting with Nutch over a 250K-page UK test corpus.[11] At the UC Santa Cruz School of Engineering, researchers are modifying Nutch to convert its craws into Internet Archive-compatible ARC format.[12] The Chronica[13] project at the University of San Francisco is investigating Nutch for a date-range constrained historical search over ARCs as well.[14]

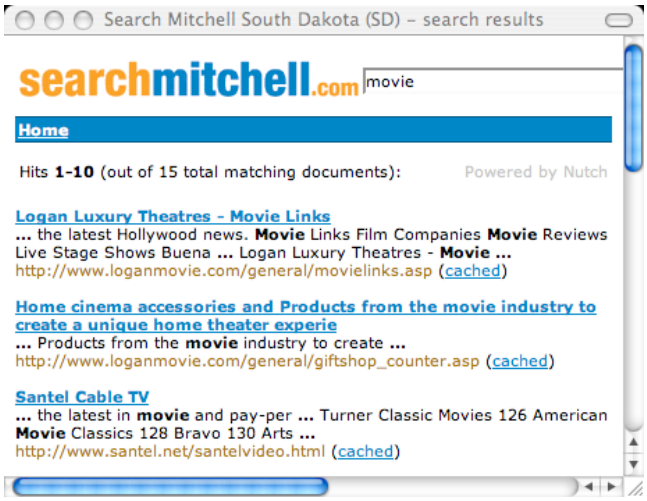

Figure 4: A Nutch-powered community search-engine for the Chamber of Commerce of Mitchell, South Dakota.

Teachers: A cleanly-architected large software system is also a pedagogical asset, particularly in a cutting-edge area with few other textbook case studies. Nutch has already appeared in course notes and homework assignments in classes on data mining at Brooklyn Polytechnic,[15] linguistics at the University of Michigan,[16] and artificial intelligence at Indiana University,[17] information retrieval at Arizona State,[18] and web services at University of Washington.[19]

Researchers: A good measure of Nutch's claim of flexibility is the ability to build new experimental systems with it. Nutch has already made a small but growing impact on the Web search engineering literature. At Cornell, a CS professor's investigation of how users' clickthrough choices can be used to feed back into relevance calculations [25] is running on their library Web site with Nutch.[20] MEAD, a multi-document summarizer has been integrated with Nutch to provide more detailed search hit excerpts [47]. Customizing link analysis to experiment with personalized graph analysis in Nutch also holds promise [2]. A German team built the geoPosition plugin[21] to Nutch to provide basic geographic distance-bounded search to a local search engine.[22]

8.2 Scalability

As discussed in §4.1, Nutch hasn't scaled beyond 100 million pages so far, for both economic and technical reasons.

Maintaining an up-to-date copy of the entire Web is inherently an expensive proposition, costing substantial amounts of bandwidth. (Perhaps 10 terabytes per month at minimum, which is about 30 megabits per second, which costs thousands of dollars per month)

Answering queries from the general public is also inherently an expensive proposition, requiring large amounts of computer equipment to house terabytes of RAM and large amounts of electricity to power it [19]; as well as one to three orders of magnitude more bandwidth.

• Nutch's link analysis stage requires the entire link database to be available on each machine participating in the calculation and includes a significant non-parallel final collation step.

• Because Nutch partitions its posting lists across cluster nodes by document, each query must be propagated to all of the hundreds or thousands of machines serving a whole-web index. This means that hardware failures will happen every few hours, the likelihood

of a single slow response causing a query to wait several seconds for a retransmission is high, and the CPU resources required to process any individual query become significant.

• Because crawls retrieve unpredictable amounts of data, load-balancing crawls with limited disk resources is difficult.

More prosaically, much of Nutch's distribution across clusters must be done manually or by home-grown cluster-management machinery; in particular, the distribution of data files for crawling and link analysis, and the maintenance of search-servers.txt files all must be done by hand. Large deployments will require fault-tolerant automation of these functions.

Further work is being done in this area to enhance Nutch's scalability. The Nutch Distributed File System (NDFS, [11]) is in current development versions of Nutch, to enhance performance along the lines proposed by the Google File System [21]. NDFS has been used recently to run a link analysis stage over 35 million pages on twelve machines.[23]

In the last several years, much work has focused on eliminating economic scalability limitations on services such as file downloading by distributing the work of providing the service among its users. Until 2004, this has appeared technically infeasible for a full-text search engine for the whole Web [31]. However, recent and ongoing work suggests that this kind of peer-to-peer distribution may soon be possible [49].

8.3 Transparency

Somehow, software can never be too flexible or too scalable, but there are recurrent fears that it *can* be too transparent. We do not agree. Nutch is committed to an open development culture that can fuel a positive feedback loop between users, researchers, developers, and even spammers that can complement – if not check – the growing dominance of a few corporations in today's Web search marketplace.

For users, if Nutch cannot find relevant pages as well as state-of-the-art commercial services, it may not get much headway. Today, Nutch's main line of defense against spam is its link analysis — which spammers have been subverting since its advent. But we believe that if search has to rely on secrecy to beat spam, then the spammers will probably win.

For researchers, we hope to level the playing field with R&D efforts behind corporate firewalls. If Nutch can showcase academic advances more rapidly than other engines, and more importantly, make it easy to combine elements of multiple extensions in new experiments, it can help decentralize the process of innovation.

For developers, we note that open source code does not require open service. Many commercial ventures are already hosting proprietary extensions of Nutch, along with proprietary crawl data to provide value to their customers without being compelled to contribute those changes. We hope they will, but Nutch's license consciously reduces this barrier to adoption.

For citizens, we acknowledge that Nutch, if successful, still may not be an unalloyed good. The same tools that let a dissident diaspora maintain a vibrant Web community search engine also make it easier for authorities to create their own selective views of the Web to index.

> *"Search is close to a duopoly. Historically we know there are risks when that happens. It's too important an application to not be transparent."* — *Mitch Kapor [5]*

9. ACKNOWLEDGEMENTS

The authors would like to thank Mike Cafarella for working with the first author on several prior articles that this paper draws upon.

The Nutch Organization would also like to thank Overture (now Yahoo!) for its generous support from the very beginning of the project; its board members: Mitch Kapor, Tim O'Reilly, Graham Spencer, Raymie Stata, and Peter Savich; and all of the volunteers who have contributed code, documentation, and support.

This work was also supported in part by CommerceNet, LLC.

10. REFERENCES

[1] Agrawal, R., Rajagopalan, S., Srikant, R. and Xu, Y. *Mining Newsgroups Using Networks Arising From Social Behavior* in *Proc. of the 12th Int'l World Wide Web Conference*, 2003.

[2] Aktas, M. S., Nacar, M. A. and Menczer, F. *Personalizing PageRank Based on Domain Profiles*, in *Workshop on Web Mining and Web Usage Analysis*, (Seattle, WA, Aug 2004).

[3] Apple Developer Connection. *Search Kit Reference.* 2004.

[4] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A. and Raghavan, S. *Searching the Web* in *ACM TOIT*, 2001.

[5] Battelle, J. *Watch Out, Google: Nutch could rewrite the rules of search development* in *Business 2.0*, 8 August, 2003.

[6] Battelle, J. *The Search: Business and Culture in the Age of Google*, Penguin: to appear (2005).

[7] Bell, T. C., Moffat, A., Witten, I. H. and Zobel, J. *The MG Retrieval System: Compressing for Space and Speed* in *Communications of the ACM*, 1995, *38* (4). pp. 41-42.

[8] Benedict, L. *Comparison of Nutch and Google Search Engine Implementations on the Oregon State University Website.* Oregon State University, June 2004.

[9] Brin, S. and Page, L. *Anatomy of a Large-Scale Hypertextual Web Search Engine*, in *Proc. of the 7th Int'l Conference on World Wide Web*, (Brisbane, Australia, 1998), pp. 107-117.

[10] Cafarella, M. *CSE454 Lecture Notes: Inside Nutch.* University of Washington, February 3 2004.[24]

[11] Cafarella, M. *Nutch Distributed File System.* August 2004.

[12] Cafarella, M. and Cutting, D. *Building Nutch: Open Source Search* in *ACM Queue*, 2004, *2* (2).

[13] Cutting, D. *Lucene.* 2001. http://jakarta.apache.org/lucene/

[14] Cutting, D. R. and Pedersen, J. O. *Space Optimizations for Total Ranking* in *Conference Proceedings of RIAO '97*, 1997.

[15] Dean, J. and Ghemawat, S. *MapReduce: Simplified Data Processing on Large Clusters* in *OSDI'04: 6th Symp*, 2004.

[16] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R. V., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J. and Zien, J. *SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation* in *Proc. of the 12th Int'l World Wide Web Conference*, 2003.

[17] Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R. and Robbins, D. C. *Stuff I've Seen: A System for Personal Information Retrieval and Re-use* in *Proc. of SIGIR 2003.*

[18] Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S. and Yates, A. *Web-Scale Information Extraction in KnowItAll* in *Proc. of the 13th Int'l World Wide Web Conf.*, 2004. pp. 100-110.

[19] Eubank, H., Swisher, J., Burns, C., Seal, J. and Emerson, B. *Design Recommendations for High-Performance Data*

Centers, in *Low-Power Data Center Design Charette*, (San Jose, CA, 2-5 Feb 2003), Rocky Mountain Institute, p. 100.

[20] Ford, P. *WWW'04 Semantic Web Roundup.* XML.com, 2004.

[21] Ghemawat, S., Gobioff, H. and Leung, S.-T. *The Google File System* in *Proc. of the 19th ACM Symp. OS Principles*, 2003.

[22] Gincel, R. *Focusing Enterprise Search* in *Infoworld*, October 18, 2004. (42), pp. 36-42.

[23] Haye, M. *Cross-instance Search System: Search Engine Comparison.* Report for the California Digital Library by Snyder-Haye Inc, January 2004. 8pp.

[24] Henzinger, M. R., Motwani, R. and Silverstein, C. *Challenges in Web Search Engines*, in *Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence*, (2003), pp. 1573-1579.

[25] Joachims, T. *Optimizing Search Engines Using Clickthrough Data* in *ACM Knowledge Discovery and Data Mining*, 2002.

[26] Kamvar, S. D., Haveliwala, T. H., Manning, C. D. and Golub, G. H. *Extrapolation Methods for Accelerating PageRank Computations* in *12th Int'l WWW Conf.*, 2003.

[27] Kirsch, S. *A Conversation with Matt Wells* in *ACM Queue*, April, 2004. vol. *2* (2), pp. 18-24.

[28] Kleinberg, J. M. *Hubs, Authorities, and Communities* in *ACM Computing Surveys (CSUR)*, 1999, *31* (4).

[29] Langville, A. N. and Meyer, C. D. *A Survey of Eigenvector Methods of Web Information Retrieval* in *SIAM Review*, 2003

[30] Lasica, J. D. *Balancing Act: How News Portals Serve Up Political Stories* in *Online Journalism Review*, 2004.

[31] Li, J., Loo, B. T., Hellerstein, J. M., Kaashoek, M. F., Karger, D. R. and Morris, R. *On the Feasibility of Peer-to-Peer Web Indexing and Search*, in *Proc. of IPTPS'04*, 2004.

[32] Lindsat, K. *Secure Locate.* 2002. http://www.geekreview.org/slocate

[33] Manber, U., Smith, M. and Gopal, B. *WebGlimpse: Combining Browsing and Searching* in *Usenix TC*, 1997.

[34] Manber, U. and Wu, S. *Glimpse: A Tool to Search Through Entire Filesystems* in *Usenix Technical Conf.*, 1994.pp23-32.

[35] Miller, R. C. and Bharat, K. *SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers*, in *Proc. of the 7th Int'l WWW Conf*, (Brisbane, Australia, April 1998).

[36] Morgan, E. L. *ht://Dig Search.* May 2001. http://www.infomotions.com/musing/opensource-indexers/htdig/

[37] mozDex. *mozDex Open Source Engine.* 2004.

[38] Namazu. *Namazu User's Manual.* 2004. http://namazu.org/doc/manual.html

[39] National Science Digital Library. *New Search Engine at NSDL.org* in *NSDL Whiteboard Report*, July, 2003. (32). http://content.nsdl.org/wbr/Issue.php?issue=32

[40] Nutch Organization. *Nutch API.* 2004. http://www.nutch.org/docs/api/index.html

[41] Overture. *Public Nutch Index Demo*, 2003. http://research.yahoo.com/demo/nutch/

[42] Page, L., Brin, S., Motwani, R. and Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web.* Stanford University, Palo Alto, November 1999.

[43] Patterson, A. *Why Writing Your Own Search Engine Is Hard* in *ACM Queue*, 2004, *2* (2).

[44] Pinkerton, B. *Finding What People Want: Experiences with the WebCrawler* in *Proc. of the 2nd Int'l WWW Conf.*, 1994.

[45] Pogue, D. *Google Takes on Your Desktop. New York Times*, New York, October 21 2004.

[46] Rabinowitz, J. *Indexing and Retrieving Data with Perl and SWISH-E* in *Proc. of the Usenix Technical Conference*, 2004.

[47] Radev, D., Allison, T., Craig, M., Dimitrov, S., Kareem, O., Topper, M. and Winkel, A. *A Scaleable Multi-Document Centroid-Based Summarizer* in *HLT-NAACL Demos*, 2004.

[48] Rose, D. and Stevens, C. *V-Twin: A Lightweight Engine for Interactive Use* in *5th Text REtrieval Conf.*, 1997. pp. 279-90

[49] Shi, S., Yang, G., Wang, D., Yu, J., Qu, S. and Chen, M. *Making Peer-to-Peer Keyword Searching Feasible Using Multi-level Partitioning*, in *Proc. of IPTPS'04*.

[50] Szwarc, R. *Zoë.* 2003. http://zoe.nu/

[51] Thau, R. S. *Design Considerations for the Apache Server API*, in *Proceedings of the Fifth International Conference on World Wide Web*, (Paris, France, 1996).

[52] Thompson, K. *Reflections on Trusting Trust* in *Communications of the ACM*, 1984, *27* (8). pp. 761-763.

Hyperlinks

[1] http://search.oregonstate.edu/about/

[2] http://www.tbray.org/ongoing/When/200x/2003/07/30/OnSearchTOC

[3] http://www.objectssearch.com/linux/index.html

[4] http://www.objectssearch.com/search.jsp?query=pink&clustering=yes&thumbshots=yes

[5] http://static.playfuls.com/info/index.php?resource=whyplayfuls

[6] http://www.mozdex.com/faq.html

[7] http://search.comm.nsdl.org/cgi-bin/wiki.pl?LuceneIndexing

[8] http://texts-stage.cdlib.org/search — CONFIDENTIAL until January 2005

[9] http://wayback.archive.org/

[10] http://recall.archive.org/

[11] http://crawlprojects.archive.org/search.jsp?query=http

[12] http://wiki.soe.ucsc.edu/bin/view/WA/OtherResources

[13] http://chronica.cs.usfca.edu/chronica/

[14] http://cs.usfca.edu/~rstevens/archiveproject/archives/000431.html

[15] http://cis.poly.edu/suel/webshort/

[16] http://tangra.si.umich.edu/~radev/LNI-winter2004/hw/hw3.pdf

[17] http://www.cs.indiana.edu/classes/b659/final/total.ppt

[18] http://rakaposhi.eas.asu.edu/s04-cse494-mailarchive/msg00029.html

[19] http://www.cs.washington.edu/education/courses/cse454/04wi/project.html

[20] http://kodiak.cs.cornell.edu:8080/search.jsp?query=ithaca

[21] http://nutch.eventax.com/

[22] http://www.UmkreisFinder.de/ — enter a city, a term, and choose a distance.

[23] http://cvs.sourceforge.net/viewcvs.py/nutch/nutch/src/java/net/nutch/tools/DistributedAnalysisTool.java

[24] http://www.cs.washington.edu/education/courses/454/04wi/slides/nutch.ppt