A comparison between public-domain search engines

Marina Buzzi, IIT-CNR, Pisa, Italy

Pasquale Lazzareschi, IIT-CNR, Pisa, Italy

1 Abstract

The enormous amount of information available today on the Internet requires the use of search tools such as search engines, meta-search engines and directories for rapid retrieval of useful and appropriate information.

Indexing a website's content by search engine allows its information to be located quickly and improves the site's usability. In the case of a large number of pages distributed over different systems (e.g. an organization with several autonomous branches/departments) a local search engine rapidly provides a comprehensive overview of all information and services offered.

Local indexing generally has fewer requirements than global indexing (i.e. resources, performance, code optimization), thus public-domain SW can be used effectively.

In this paper, we compare four open-source search engines available in the Unix environment in order to evaluate their features and effectiveness, and to understand any problems that may arise in an operative environment.

Specifically, the comparison includes:

- The SW features (installation, configuration options, scalability);
- User interfaces;
- The overall performance when indexing a sample page set;
- Effectiveness of searches;
- State of development and maintenance;
- Documentation and support.

2 Introduction

Generally speaking, search engines perform three main functions: crawling the Web (or the Intranet), indexing the collected pages and searching for keywords specified by users within the indexes, as shown in Figure 1.

Due to the Web's enormous size, its multimedial nature and rapid mutations, today search engine engineers face numerous challenges in each component of its architecture, which have great impact on performance, user interaction and quality of search engine results. However, these strict requirements are relaxed if a search engine is utilized for indexing an Intranet instead of the entire Web, and thus public-domain SW can be used effectively.

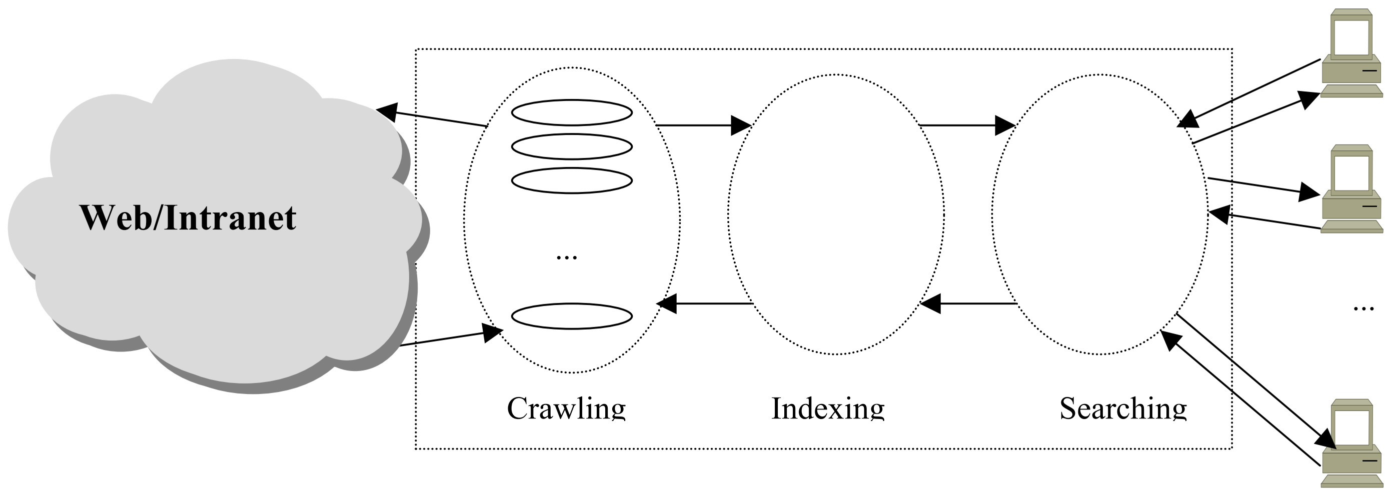

Fig. 1–Logical scheme of search engine architecture

Choosing to use a local search engine offers several advantages:

1. Completeness of the information and services indexed due to the possibility of crawling the entire set of the organization's data;
2. Transparency, since the open source code of public domain search engines allows us to understand how ranking algorithms function, and permit us to tune ranking parameters;

3. Flexibility. It is possible to customize and incorporate user interfaces in website and web applications;
4. Cleanness of results since Sponsored Links are not present.

The drawback is that crawling only the Intranet (and not the whole web, as Google does) the ranking algorithms are unable to consider global factors such as the impact of page popularity (i.e. the number of external incoming links), which positively affect the precision of results.

In this paper we compare four public domain search engines, available for the Linux environment: Nutch, DataparkSearch, mnoGoSearch and ht//Dig, and evaluate their features by applying the indexing to a set of websites belonging to our organization, i.e. the Italian National Research Council. Our analysis is limited to the Linux environment although some of these SW are also available for Windows either free or under license.

This paper is divided into four sections. Section 3 contains a brief description of the three main components of a search engine (crawling, indexing and searching); Section 4 introduces features of the SW analyzed: Nutch, mnoGoSearch, DataparkSearch and ht//Dig; and in Section 5 the evaluation in a Linux environment of the four search engines is described and discussed. Lastly, the paper closes with conclusions and remarks on experiments performed and experience gained.

3 Search engine components

3.1 Crawling

Web crawler design presents many different challenges: architecture, strategies, performance and more. One of the most important research topics concerns improving the selection of “interesting” (for the user) web pages, according to importance criteria. Another relevant point is content freshness, i.e. maintaining freshness and consistency of temporary stored copies. For this, the crawler periodically repeats its activity, going over stored contents (re-crawling process). Crawlers are SW components which visit portions of web trees, according to certain strategies, and collect retrieved objects in local repositories. Usually a crawler starts from a set of “interesting” URLs, collects new

URLs from pages visited and continues to explore until resources are available [2]. Search engines use crawlers to collect local copies of web pages [6].

A very important problem is keeping the local collection “fresh”, which means a high probability that a stored copy is equal to the original object. At regular intervals the crawler repeats its inspection of web pages in order to refresh modified contents as well as to discover new pages. Many strategies for optimizing re-crawling have been studied but the variety of different contexts and the highly dynamic nature of the WWW make it difficult to model the web effectively. Web pages have a life cycle: they are born, change and can also disappear; they change with very different update rates, which can vary over time, thus becoming difficult to model effectively. In addition, the freshness of stored copies is influenced by many factors such as type of retrieval, updating method, visit frequency and object-replacing policy [4].

Another problem is that of selecting more “interesting” objects, for the users. A search engine is aware of hot topics because it collects user queries. The crawling process prioritizes URLs according to importance criteria such as similarity (to a driving query), back-link count, PageRank or their combinations/variations [5], [2]. Najork et al. showed that breadth-first search collects high-quality pages first and suggested a variant of PageRank [12]. However, currently search strategies are unable to exactly select the “best” paths because their knowledge is only partial. Due to the enormous amount of information available on the Internet, total-crawling is at the moment not possible, and thus prune strategies must be applied. Focused crawling [3], [8] and intelligent crawling [1], for instance, are emerging techniques for discovering web pages relevant to a specific topic or set of topics.

3.2 Indexing

Data collected by crawlers are stored and indexed. The indexer module extracts all the words from each page and records the URLs where each word occurred (e.g. generates the vocabulary). This generates very large structures (e.g. inverted indexes) that provide all the URLs of pages where a given word appears. In addition, the indexer module may also create other kinds of indexes aimed at optimizing the query phase (e.g. for immediate access to relevant pages).

The page repository contains pages collected by crawlers in their activities (i.e. retrieved from the Web/the Intranet). Search engines maintain a cache of the pages visited (whole or partial content) in order to provide an excerpt of the result pages. These activities (i.e. storage and retrieval on very large scale) require special treatment such as data compression and fast indexing. The indexing can be a separate step or be integrated into the crawler phase.

3.3 Searching

The query engine module manages search requests from users. The main problem is how to reliably filter a sufficient number of irrelevant results when a user typically specifies only one or two keywords, and the set of results is typically very large. A specific module is dedicated to ranking, i.e. sorting the results so that elements in the first positions have a high probability of being what the user is seeking. Ranking is crucial for retrieving appropriate results. In fact, the traditional techniques applied in IR (similarity-based algorithms) are not suitable for the Web due to its size, structure (hyperlink) and dynamic nature. Specific algorithms such as Page Rank and its numerous variations, which may significantly improve retrieval precision in Web searches, have been proposed in recent years.

The graphical interface for user queries is also very important for user interaction. In the last few years there has been an increasing awareness of the importance of making UIs accessible to anyone, in any condition, regardless of any disability. Any person should be able to perceive, understand, navigate, and interact with the Web [15] and if possible contribute to its development. Unfortunately at present only a very small part of the enormous amount of information available on the Internet is accessible, and web interaction may require considerable effort for the visually disabled, who interact by means of assistive technologies (e. g. screen reader and voice synthesizer). However, accessibility is necessary but alone is insufficient to guarantee easy and satisfactory interaction for anyone; thus usability criteria should be applied from the earliest stages of user interface design.

As previously mentioned in the introduction, public domain search engines offer the considerable advantage of making the source code available and thus making it possible to customize the UIs to fulfill accessibility and usability criteria [15].

4 Public domain search engine

In this section we introduce the four public domain search engines we tested. A summary of their features is then included in Table 1.

4.1 Nutch

Nutch [13] is a search engine developed under the Apache Lucene project. This project is dedicated to develop open-source search software. Specifically, Lucene provides a library of Java-based indexing and search technology while Nutch provides web search application software built on Lucene.

Nutch is written in Java so requirements for its installation include:

- Java 1.4.x;
- Apache's Tomcat 4.x (for the searching interface via web);
- An appropriate free disk space (up to a GB);
- A high-speed connection.

Although the search engine is able to crawl the entire web we only describe the configuration for Intranet crawling: i.e. to efficiently crawl a limited group of web servers, up to around one million pages.

Configuring Nutch for Intranet crawling is an easy three-step process described in the available tutorial:

- Create a directory with a flat file of root urls;
- Edit the file conf/crawl-urlfilter.txt and replace MY.DOMAIN.NAME with the name of the domain you wish to crawl;
- Run the Crawl with the appropriate options (directory to put the crawl in, number of threads that will fetch in parallel, link depth from the root page that should be crawled, maximum number of pages that will be retrieved at each level up to the depth).

but due to the lack of complete documentation (no manual is available), advanced configuration, set-up of external parser for .doc and .pdf documents and information about how to activate re-crawling has been retrieved with difficulty.

4.2 MnoGoSearch

MnoGoSearch [11] is an SQL based search engine. The Unix source files are distributed free under the terms of the GNU General Public License while the Windows version is distributed as shareware under license (30- day trial available).

It was first released in 1998 under the name UDMSearch; in October 2000 it was acquired by Lavtech.Com Corp. and the name was changed to mnoGoSearch.

MnoGoSearch is written in C and may use several SQL databases such as MySQL, PostgreSQL, Oracle, etc. In our experiment we utilized mnoGoSearch version 3.2.38 with MySQL 4.1.

MnoGoSearch has a built-in parser for html and plain text documents; external parsers or converters are available for other mime types (for example mnoGosearch can be configured to call an external program such as catdoc to convert a MSWord document to plain text, which can be indexed by the built-in parser).

Documentation is available online and support can be obtained by subscribing to “mailing lists”, browsing the “web boards”, looking through the “bug track system” or purchasing a mnoGoSearch support.

The installation steps are described in the documentation but require some knowledge of Unix/Linux program development. The source code must be downloaded, unpacked, configured, compiled and installed. A database userID with create privilege is needed to generate the new MySQL database and create a new MySQL userID with privileges on the new database. System administrator is required to install the search interface cgi-bin program.

MnoGoSearch use configuration files in plain text format. The distribution provides samples of these files, that must be re-named and customized.

4.3 DataparkSearch

DataparkSearch Engine [7] is an SQL-based search engine available only for Unix-like platforms, released under the GNU General Public License.

DataparkSearch is derived from mnoGoSearch so the core of the search engine is common to both the SW. However, they present major differences in storage modes and ranking functions, which reflect on the precision of performance and results. Specifically, DataparkSearch updated mnoGoSearch v. 3.2.15. As detailed in the following, our experiments revealed differences both in the number of objects indexed and in the query results. DataparkSearch is designed to organize search within a website, group of websites, intranet or local system.

Support is provided by documentation (a manual is available on-line), via mailing list and forum. Furthermore, a System for the registration of bugs is available.

Installation requirements are very similar to mnoGoSearch as well as basic configuration (see Tables 1 and 2).

4.4 ht//Dig

Ht//Dig [9] is a search engine written in C++ developed at San Diego State University as a tool for searching information published on the web servers of the campus network. It was designed to satisfy search needs for a single company, or campus, but it can be easily utilized for indexing several web servers. ht//Dig relies on the Berkeley database and is not suitable for indexing the entire web since their structures and algorithms are not optimized for storing, indexing and retrieving the massive load of data available today on the Internet. It is available only for Unix-like platforms, and is released under the GNU General Public License.

It is possible to tailor the search results to user needs by means of providing HTML templates and the searches can be performed using various configurable algorithms. A system administrator is required to set up the web server for search.

The documentation includes online manuals, FAQ and considerable other information. However, development activity is low: the last stable version 3.1.6 was released in February 2002 and the last beta v. 3.2.0b6 in June 2004.

5 The Comparison

In our experiment we indexed the domain .cnr.it. The Italian National Research Council is a governmental organization dedicated to the promotion, coordination and regulation of scientific research and technological progress in Italy. It is composed of 14 Departments, 108 Research Institutes and 18 Research Areas located throughout Italy. Each Institute has its own website and sometimes projects or important research activity also have their own websites.

Specifically we set up the configuration files of the selected search engines to:

1. Start crawling from 139 urls (those of the CNR institutes, research areas and administration departments);
2. Limit the crawler depth to 10 levels from the start urls (seeds);
3. Limit the object's size to 2 megabytes;
4. Use external parser or converter for Msword, and pdf document formats;
5. Limit the crawler to the domain .cnr.it.

The experiments were carried out on an Intel Xeon 3.4 Ghz machine, with 2GB RAM, and OS Debian Linux v. 3.1, Apache 2.0.54, Tomcat 4.1.31, Java 1.4.2 and MySQL 4.1.11. All the system software were in the default configuration.

5.1 Search engine features

Table 1 summarizes basic search engine features (Development activity, License, Installation, Configuration, Multiplatform, Scalability, Documentation and Support) as publicly documented while Table 2 describes our experience in installing, configuring and running the SW, highlighting any problem encountered in our specific experiments.

Despite some initial problems we encountered on Nutch (for retrieving information about how enabling plug-ins and activating re-crawling) and with ht//Dig (since its last beta version does not run in our experiment) we were finally able to activate the search engines with homogeneous configurations (as far as possible), in order to render the evaluation significant. MnoGoSearch did not reveal any problems, while DataparkSearch sometimes halted the crawling process without any error message but once re-started, it continued crawling from the last level processed.

	Nutch	***mnoGoSearch***	***DataparkSearch***	***ht//Dig***
Development activity	High. Last version 0.7.2 released on 31 March 2006	High. Last version 3.2.38 released on 15 March 2006	High. Last version 4.38 released on 13 March 2006	Low. Last stable version 3.1.6 (1 February 2002). Last beta 3.2.0b6 (16 June 2004)
Licence	Apache license	For Unix/Linux GNU General Public License. For Windows ad hoc license (shareware)	GNU General Public License	GNU General Public License
Installation	Requires: Java 1.4.x Apache Tomcat 4.x System administrator can be required for Tomcat	On Unix/Linux must be compiled from source. Require: C compiler, a SQL database (MySQL, PostgreSQL, Oracle, ect). System administrator can be required to create the database and set up the web server for search	Must be compiled from source. Require: C compiler, a SQL database (MySQL, PostgreSQL, Oracle, ect) System administrator can be required to create the database and set up the web server for search	Must be compiled from source. Require C++ compiler System administrator can be required to set up the web server for search
Configuration	Plain text and xml files to be customized manually. The distribution includes samples	Plain text files. The distribution includes templates that must be renamed and customized manually	Plain text files. The distribution includes templates that must be renamed and customized manually	Plain text file. The distribution provides a template that must be customized manually
Multiplatform	Should run on all systems where Java runs	Unix/Linux as source code. Windows executable	Only Unix/Linux	Only Unix/Linux
Scalability	From Intranet to the entire Web	Millions of pages	Up to a million pages	Up to a million pages
Documentation	Online documentation include: FAQ, tutorial and wiki. The tutorial describes only the first installation and run. It is not easy to learn how to update (re-crawl). Complete manual lacking	Online documentation include: complete manual, FAQ and mailing list	Online documentation include: complete manual, wiki and mailing list	Online manuals, FAQ and a lot of other information
Support	Mailing lists	mailing lists, web boards, bug track system or purchase a mnoGoSearch support	Mailing lists and online bug system	Mailing lists and online bug reporting system

Table 1 – SW features: Development activity, Licence, Installation, Configuration, Multi-platform, Scalability, Documentation and Support

	Nutch	***mnoGoSearch***	***DataparkSearch***	***ht//Dig***
Installation	Easy. Requires only download and unpack of distribution file. For the web interface it is necessary to copy the Nutch web application in the Tomcat web application directory	Described in the manual. Easy for those with experience in development of Unix application. System administrator required to create the database and set up the web server for search	Described in the manual. Easy for those with experience in development of Unix application. System administrator required to create the database and set up the web server for search	Described in the manual. Version 3.2.0b6 is easy for those with experience in development of Unix application. Version 3.1.6 required hacking the configure file. System administrator required to set up the web server for search
Configuring the crawler	Manual edit of configuration files. The tutorial, available on the website, covers only basic configuration for crawling an intranet. Difficult to find documentation about setup plug-ins for mime types different from text/html and text/plain	Manual edit of configuration files. The distribution includes configuration template files that can be easily re-named and customized manually	Manual edit of configuration files. The distribution include configuration template files that can be easily re-named and customized manually	Manual edit of configuration files. The distribution includes configuration sample files that can be easily customized manually
Configuring the search interface (web server)	System administrator is required for Tomcat. More effort is required for integration in other web server (Apache)	System administrator is required for web server setup	System administrator is required for web server setup	System administrator is required for web server setup
Running the crawler	Easy for Intranet crawling. Not easy to find documentation for re-crawling	Easy. Can be started multi- tread or multi-process	Easy. Can be started multi- tread or multi-process	Easy
Problems	Lack of a complete manual	None in our experiments	Sometimes the crawler (indexer) stops without apparent cause. Using storage mode cache (the fast one) we have to change the system limit of open files to 10000. Anyway it can be re-started without losing the previous work	Version 3.2.0b6 does not run in our experiment (url not found error); thus we use the previous stable version

Table 2 – Our experience in Installation, Configuring the crawler, Configuring the web, Running the crawler, Problems

5.2 Performance

Concerning the overall performance indexing a set of pages, we crawled (and re-crawled) the websites belonging to the domain .cnr.it by using the selected search engines in configurations that were as homogeneous as possible as shown in Table 3. ht//Dig, in fact, does not support multi-thread and does not store copies of retrieved pages as well as DataparkSearch.

	Download file size limit	*File size limit of stored copy*	*# threads*	*Max hops (crawling levels)*	*Mime types*	*URL filter*	*Starting URLs*
Nutch	2100 KB	Apparently no limit	10	10	text, html, msword, pdf	.cnr.it	homepages of CNR Institutes, sections, areas, and administration departments
mnoGoSearch	2100 KB	64 Kbytes	10	10	text, html, msword, pdf	.cnr.it	homepages of CNR Institutes, sections, areas, and administration departments
DataparkSearch	2100 KB	No local copy (only excerpt)	10	10	text, html, msword, pdf	.cnr.it	homepages of CNR Institutes, sections, areas, and administration departments
ht//Dig	2100 KB	No local copy (only excerpt)	1	10	text, html, msword, pdf	.cnr.it	homepages of CNR Institutes, sections, areas, and administration departments

Table 3 – Configuration parameters

The total number of indexed files varies from 122,983 (ht//Dig) to 80,098 (DataparkSearch). As shown in Table 4, the search engines, although having fairly similar configurations (stop rules, filter definitions, 10 crawling levels, etc.) presented some differences in the number of:

(a) retrieved objects
(b) errors due to object not found, connection refused, host down (note that an host down may hide thousands of pages)
(c) files with mime-types discarded.

CNR is connected to the Italian Academic and Research Network "GARR-B" which has a backbone at 10Gbps, and links to 2,5 or 1 Gbps and 622, 155, or 34 Mbps. We carried

out the experiments from the Pisa CNR Research Area, connected at the GARR network with three 2.5 Gbps links and one 155 Mbps link.
It is obvious that the results are calculated only on a specific case and thus they are not statistically significant, but since experiments are executed in the same system under similar conditions they may provide a useful indication for system administrators. Specifically the network traffic and condition (status DNSs, routers, hosts, etc.), which may vary depending on the time, is a condition beyond the control of the system administrator, which may greatly impact results. However, with regular re-crawling, the effects of temporary errors tend to disappear.

	Time to crawl the .cnr.it domain	***Number of correctly indexed objects***	***Errors: not found, connection refused, can't resolve***	***Content type different from text/html, text/plain, pdf or msword***
Nutch	8h 32m	112,918	8,513	3,848
mnoGoSearch	5h 20m	95,163	9,594	3,289
DataparkSearch	14h 22m	80,098	9,486	1,753
ht//Dig	18h 55m	122,983	14,327	3,687

Table 4 – Crawling times, indexed objects and errors

To optimize performance, DataparkSearch was set-up with cache storage mode [7] and mnoGoSearch with blob storage mode [11].

5.3 Effectiveness of user queries

We verified the efficiency of the search modules by formulating the following ten queries:

1. nanotechnologies
2. cardiology
3. grid computing
4. accessibility usability
5. research activities
6. “research activities” (Quotation marks force the proximity of the two words)
7. IIT Director

8. CNR president
9. Istituto di Tecnologie Didattiche
10. Bologna Research Area Library

	Nutch	***mnoGoSearch***	***DataparkSearch***	***ht//Dig***
nanotechnologies	201	161	99	447
cardiology	91	109	98	179
grid computing	398	287	345	1,040
accessibility usability	71	64	44	120
research activities	3,652	917	434	12,680
"research activities"	1,418	181	198	12,680
IIT Director	21	25	18	37
CNR president	818	512	43	496
Istituto di Tecnologie didattiche	314	63	6	586
Bologna Research Area Library	69	109	29	234

Table 5 – Number of results per query

Note that ht//Dig generated more results since it was configured for stemming, i.e. the search for *research activities* generates results for the strings: '(research or researched or researching or researcher or researches or researchers) and (activities or activity)'.

As response time we measured the time from query submission to the result return. To do so we used the wget command to submit queries and the unix time command to get the elapsed time. As shown in Table 7 all the response times are acceptable since they reach at maximum about 1 sec.

Query keywords	***Nutch***	***mnoGoSearch***	***DataparkSearch***	***ht//Dig***
nanotechnologies	0.166	0.056	0.051	0,047
cardiology	0.040	0.067	0.047	0.030
grid computing	0.041	0.059	0.049	0.192
accessibility usability	0.074	0.059	0.029	0.033
research activities	0.056	0.065	0.041	1.066
"research activities"	0.035	0.065	0.070	1.057
IIT Director	0.081	0.066	0.059	0.026
CNR president	0.054	0.099	0.077	0.226
Istituto di Tecnologie didattiche	0.153	0.148	0.189	0.202
Bologna Research Area Library	0.157	0.075	0.091	0.247

Table 6 – Times of queries in sec.

These times were measured for a single query sent to an idle server. However we were interested in understanding how quality of service for the user may vary when the search engine is serving an increasing number of parallel queries

5.4 Quality of service

Increasing the rate of parallel requests, the server may reach saturation so it becomes unable to send a response in a reasonable time and the client may go in time-out.

Figure 2 and 3 show the medium and maximum response times when the number of queries increases. On the x axis are represented the number of queries sent to the server and on the y axis the response time in seconds. We started the test by sending a set of 10 contemporary queries and then increasing the request rate sending 20, 30, 40, 50,… 200 requests. Since the queries are generated on a single system they are not truly parallel; however, after the initial period the server reaches the steady state and receives a number of simultaneous requests to serve. However, in order to verify whether a single client might be a performance bottleneck we generated the same load, sending simultaneous requests to the server from two clients, obtaining analogous results.

In our experiment, Nutch and ht//Dig were able to sustain a load of less than 50 queries (with a reasonable response time) while DataparkSearch and mnoGoSearch gave a better performance, reaching the same response time at 180 requests. It is obvious that the saturation point depends mainly on the specific implementation but also from the resources of the machine where the SW runs and the tuning parameters of server software (Apache, Tomcat, Mysql, etc.), so with a more powerful system or a carefully tuned server the saturation point may translate ahead, or on a less powerful machine it may be reached sooner. Nutch, among the software considered, is the only one designed for a distributed architecture as well, and in such an environment the results could be different. Last, note that ht//Dig generated more results since it was configured for stemming (the number of results is larger) thus performance can be affected.

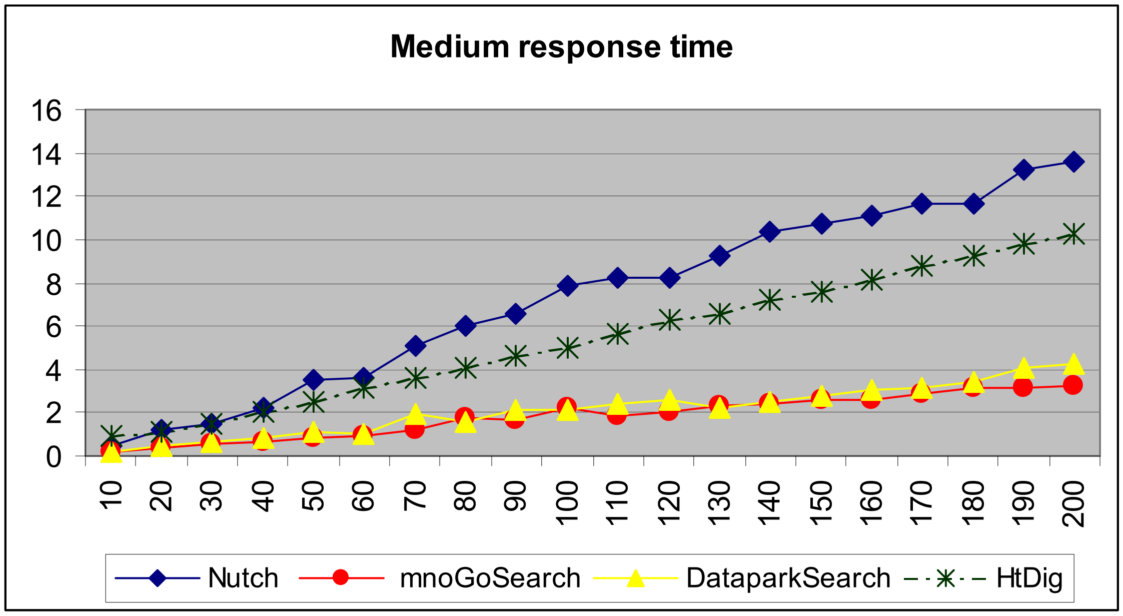

Fig. 2 - Medium response time

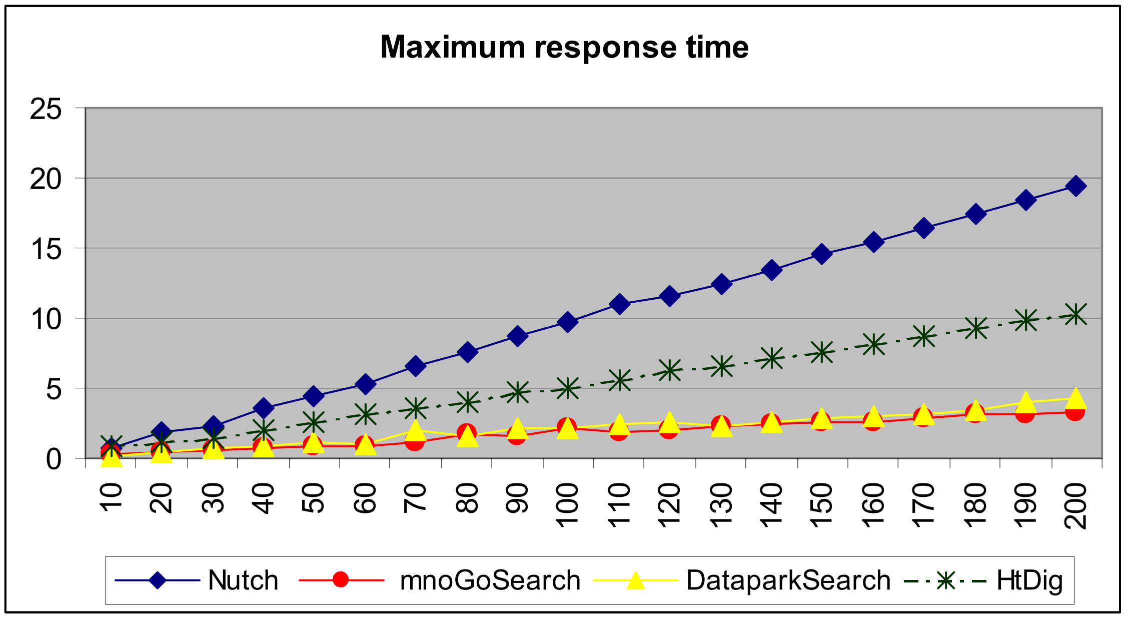

Fig. 3 – Maximum response time

5.5 Precision of query results

Concerning the precision of the query responses, since all results were ordered by ranking and the verification process was manual, we only analyzed the first page of results (10 items). Furthermore, we highlighted if the first result was pertinent or at least suitable. Let us define the precision of results as the number of pertinent (or suitable) results divided by the number of items analyzed. Thus by definition the precision is ≤ 1. For example, we analyzed 10 results; if all elements are pertinent we have precision 10/10 =1, if only 5 results are pertinent we obtain precision 5/10 = 0.5.

Queries	***Nutch***	***mnoGoSearch***	***DataparkSearch***	***ht//Dig***
nanotechnologies	1st pertinent p =1	1st pertinent p =1	1st pertinent p =1	1st pertinent p =1
cardiology	1st pertinent p =1	1st pertinent p =1	1st pertinent p =1	1st pertinent p =1
grid computing	1st pertinent p =1	1st suitable p =1	1st pertinent p =1	1st pertinent p =1
accessibility usability	1st pertinent p =1	1st pertinent p =1	1st pertinent p =1	1st pertinent p =1
research activities	1st pertinent p =1	1st suitable p =1	1st suitable p =1	1st pertinent p =1
"research activities"	1st pertinent p =1	1st suitable p =1	1st pertinent p =1	1st pertinent p =1
IIT Director	1st pertinent p =0.5	1st pertinent p =0.8	1st suitable p =0.5	1st pertinent p =1
CNR president	1st suitable p =0.5	1st pertinent p =0.9	-- p =0	1st suitable p =0.4
Istituto di Tecnologi didattiche	1st pertinent p =1	1st pertinent p =0.6	1st suitable p =0.5	1st pertinent p =0.9
Bologna Research Area Library	1st pertinent p =0.5	1st pertinent p =1	1st pertinent p =0.4	1st pertinent p =0.7

Table 7 – Results precision of first ten results ordered by ranking

In the query set analyzed, DataparkSearch does not reach an acceptable degree of precision while mnoGosearch performed slightly better than ht//Dig and Nutch (as showed in Table 7).

5.6 Elements of accessibility and usability

We analyzed the UI accessibility (for the simple search and the result page) by using the W3C Markup Validation Service [14], a free service that checks HTML and XHTML pages for conformance to W3C Recommendations [15] and other standards.

As shown in Table 8 only the UI of ht//Dig for simple search passed the validation; however the errors in the HTML code may be easily corrected. This kind of freedom is very important since most commercial search engines do not permit modification of their interfaces.

	Simple search	***Result page***
Nutch	Failed validation, 12 errors	Failed validation, 8 errors
mnoGoSearch	Failed validation, 2 errors	Failed validation, 44 errors
DataparkSearch	Failed validation, 7 errors	Failed validation, 54 errors
ht//Dig	Tentatively passed validation (HTML 4.0 Transitional)	Failed validation

Table 8 -User interfaces: result of the W3C validation service;

Concerning usability, mnoGoSearch and DataparkSearch offer very basic UIs for a simple search. ht//Dig has a little more complex interface that includes three pop-up windows at the top for selecting search options (Match, Format, and Sort by). Nutch UI includes irrelevant images that should be removed.

Regarding UIs of the result pages, mnoGoSearch and DataparkSearch have numbered the results. This feature is useful for orienting blind individuals. Nutch UI has only a push button for proceeding to the next page of results and this feature is annoying for a person who wishes to go directly to another page of results. The other search engines provide direct links to the first 10 pages of results. Last, ht//Dig shows only the first 100 results.

A more detailed discussion should be addressed regarding accessibility for disabled persons. For instance, all analyzed search engine result pages are unstructured, navigation via Tab keys is strictly sequential, access keys are lacking, etc.; thus the interfaces should be improved in order to simplify interaction for the blind as described in [10].

6 Conclusion

In this paper we have analyzed four open source search engines available for Unix-like environments. Specifically we installed, configured and tested the four SW on a Linux system, and indexed the websites of the Italian National Research Council (domain .cnr.it).

Results showed that all the analyzed SW may be used for indexing Intranet websites with a limited number of objects (up to a million) but they present differences both in performance and query precision.

Concerning performance, in our experiment we observed that mnoGoSearch and DataparkSearch achieved the best results, being able to sustain a load of 180 parallel queries, while Nutch and ht//Dig reached saturation very early (50 parallel requests).

However ht//Dig generated more results per query (since it was configured for stemming), thus performance may be affected.

Regarding precision in query results, mnoGoSearch obtained a high degree of precision in the results of the selected queries, followed by ht//Dig and Nutch. In our experiment the degree of precision of DataparkSearch was not acceptable. However DataparkSearch permits modifying some parameters for tuning and customizing the calculation of the ranking function, so further tests should be carried out.

Concerning crawling time, mnoGoSearch was the fastest, followed by Nutch. Anyway, it indexed far fewer files than Nutch so the set of all results per query was less.

Regarding robustness of crawling and indexing pages, mnoGoSearch and Nutch did not present problems, as did DataparkSearch and ht//Dig. Furthermore, ht//Dig appeared to be abandoned: the last (beta) version was released in 2004 whereas the last version of the other search engines selected was published in 2006.

Lastly, each of the search engines analyzed provides web UIs for user queries which may be easily modified and customized by the system administrator, to be integrated in the home page of the organization's web site.

In conclusion, at this time mnoGoSearch appears to be the most suitable tool for indexing an Intranet. This product is sufficiently consolidated since it was developed starting in 1998. However in the near future, Nutch, which is quite new (first released as Apache Lucene sub-project only last year) should also be the most promising SW for indexing local Intranet, if performance optimization is achieved. Actually Nutch was designed to run in a distributed architecture. We only analyzed the SW on a single machine; further studies should be carried out to evaluate its performance and behavior in a distributed environment.

7 References

[1] C. Aggarwal, F. Al-Garawi, P. Yu, "Intelligent crawling on the World Wide Web with arbitrary predicates", WWW 2001, pp. 96-105.

[2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, "Searching the Web", ACM Transactions on Internet Technology, Vol. 1, Num. 1, August 2001, pp. 2-43.

[3] S. Chakrabarti, K. Punera, M. Subramanyam, "Accelerated focused crawling through online relevance feedback", WWW 2002, pp. 148-159.

[4] J. Cho, H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental Crawler", VLDB 2000, pp 200-209.

[5] J. Cho, H. Garcia-Molina, L. Page, "Efficient Crawling Through URL Ordering", WWW7/Computer Networks 30 (1-7): 161-172 (1998).

[6] C. Chung, C. Clarke, "Topic-oriented collaborative crawling", CIKM 2002, pp. 34-42.

[7] DataPark Search Engine http://www.dataparksearch.org/

[8] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, M. Gori, "Focused Crawling Using Context Graphs", VLDB 2000, pp. 527-534.

[9] ht//Dig Search Engine. http://www.htdig.org/

[10] Leporini, B., Andronico P., Buzzi M. (2004). Designing Search Engine User Interfaces for the visually impaired. ACM International Cross-Disciplinary Workshop on Web Accessibility 2004, NY, USA, pp. 57-66.

[11] MnoGoSearch Engine. http://www.mnogosearch.org/

[12] M. Najork, J. Wiener, "Breadth-first crawling yields high-quality pages", WWW 2001, pp. 114-118.

[13] Nutch Search Engine. http://lucene.apache.org/nutch/

[14] W3C QA Markup Validation Service v0.7.2. http://validator.w3.org/

[15] W3C. Web Accessibility Initiative http://www.w3.org/WAI/