Searching a Database Based Web Site

Filipe Silva[1] and Gabriel David[1,2]

[1] Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias
4200-465 Porto Portugal,
{fsilva,gtd}@fe.up.pt
http://www.fe.up.pt/
[2] Inesc-Porto
Rua Dr. Roberto Frias
4200-465 Porto Portugal

Abstract. Currently, information systems are usually supported by databases (DB) and accessed through a Web interface. Pages in such Web sites are not drawn from HTML files but are generated on the fly upon request. Indexing and searching such dynamic pages raises several extra difficulties not solved by most search engines, which were designed for static contents. In this paper we describe the development of a search engine that overcomes most of the problems for a specific Web site, how the limitations put to indexing dynamic Web pages were circumvented, and an evaluation of the results obtained. The solution involves using a locally developed crawler, the Oracle Text full text indexer, and meta-information automatically drawn from the DB or manually added to improve the relevance factor calculation. It has the advantage of uniformly covering the dynamic pages and the static Web pages of the site.

Keywords. information retrieval, Web search engine, indexing database-based Web sites

1 Introduction

The revolution brought by the Web to the information world is not as much due to the huge volume of information at everyone's disposal as to the dramatic increase on the information access efficiency that made possible previously unfeasible tasks [1].

Among the key tools in this context are the Web search engines. They evolved from an idea as simple as the model of the Web: the Web is a network of Web servers, each one responsible for a collection of cross-linked HTML pages (see Fig. 1). The search engine includes a crawler that, starting at a given URL, retrieves the corre-

sponding page and recursively follows its links to other pages. Then it indexes the full text of the page, along with its URL. Afterwards, it becomes able to answer requests on information related to specified words by returning the URLs of the pages containing them. Thus, the search engine can be seen as made of three components: the crawler, the indexer, and the query processor. Many developments of this basic idea have been produced in order to improve the relevance factor of each page and the precision and recall metrics of the search [2], for instance, counting the number of occurrences of each word in the page, assigning more weight to occurrences in headings or in META tags, combining with other methods of classification, etc.

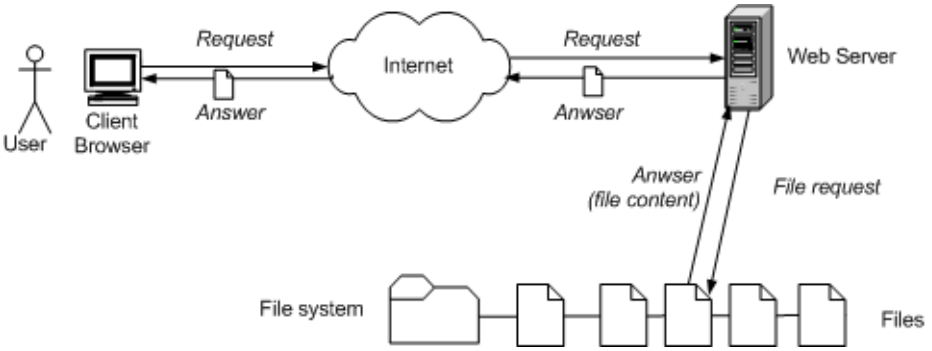

Fig. 1. Answering a static page request

However, the Web contains more than collections of HTML pages. Information systems (IS) contain large amounts of information which is typically more structured than the texts in HTML files and is organized in database systems with their own linking mechanisms. Current IS are accessible via interfaces that adopted the Web paradigm and technology. The end-user just requires a Web browser to navigate through the pages of the corresponding sites. The main difference is that these pages no longer correspond to a HTML file stored in a file system but are instead generated upon a user request by specific programs that gather the information needed from the database and deliver the HTML page directly to the user (see Fig. 2). These pages are called dynamic pages[1] in data intensive sites [3] as opposed to the static pages frozen in HTML files.

There are a number of advantages of this approach. The quality of the information presented directly benefits from the ability of DB systems to organize, maintain consistency, update and control the access to information. Generating the page at request-time enables a more flexible presentation of information, through the specification of several search criteria, and even to personalize it, if the user is known to the system.

[1] This is distinguished from pages that incorporate Dynamic HTML or JavaScript to get animation effects but are nevertheless defined beforehand in a static file.

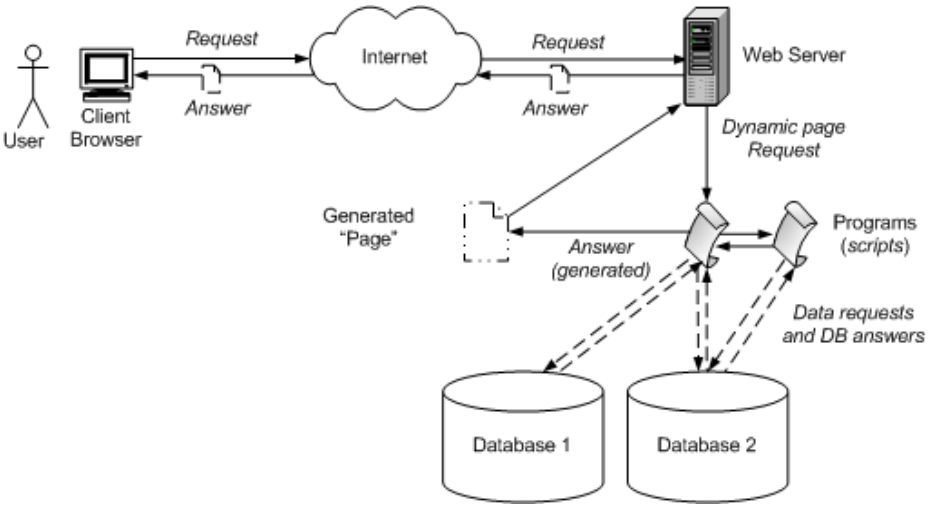

Fig. 2. Answering a dynamic page request

The user is probably not aware of these technical issues and sees the site as a set of linked pages irrespective of their source, real or "virtual" files. So, the user expects a search facility that answers a set of relevant pages to a query involving one or more terms. This goal of uniformly dealing with static and dynamic sites is especially relevant in mixed sites combining structured data with non-structured text.

This is the case of the information system of the Faculty of Engineering of the University of Porto (SiFEUP, [4,5]), an integrated system that combines academic records of various kinds with class summaries, bibliographic data with teaching materials, etc. SiFEUP is the case study considered in this paper.

The following sections present in more detail the problems raised to the search engines by dynamic Web sites, the solution that has been developed, and an assessment of the results achieved.

2 Why Search Engines Fail in Dynamic Web Sites

Before the discussion of the above mentioned difficulties, there is a comparison with the usual way of querying an IS with a Web interface. Depending on the sophistication of the site, the interface to the DB may be as obvious as a series of forms where the user types the search criteria receiving as an answer a page with a table containing the records which have been retrieved by the DB query processor. In some systems it is even possible to write down an SQL query.

The main problems with the forms-based approach are:

– the forms to be used must be anticipated by developers, trying to match the typical user requests but leaving out more specific needs;

- in a DB, the information is stored in different tables and columns and retrieval must specify which one is to be used, but users don't like to choose fields before the search [6];
- the search requires exact-match and the SQL to simulate a generic search (including stemming and the stripping of capitalization and accents) becomes rather cluttered and complex [7];
- all the records satisfying the search criteria are retrieved with equal importance and ordered by column contents, but the relevance of the items in the answer are different and acknowledging this is crucial when large numbers of records are retrieved [6].

Not everything is bad with the forms-based query with respect to the full-text indexing:

- DB indexing is usually done in real-time, while in the other case it is deferred and thus require periodic update;
- the DB needs less computing resources (CPU and storage) than the search engine;
- the output sorting method can be specified, while in the search engine is usually fixed to be the decreasing relevance order.

In more worked interfaces, the tabular nature of the system is much more distant from the user eyes, and the page includes significant pieces of text and data collected from several DB tables, creating something closer to a document than to a DB extract.

A generic search facility is expected to retrieve these documents and not lists of records from the DB. The combination of the Web and DB technologies produce highly mutable pages which may be different depending on the user and on the moment of the request. So, the very notion of document must be revised. In this paper, if to similar requests the system answers pages with different contents because the user is different, they are considered different documents. If the difference is only due to changes in the DB between the requests, these are considered different versions of the same document.

Therefore a generic search mechanism is justified. However, typical search engines, which were designed for static contents, do not index dynamic Web pages or just do that for those that are specifically linked from a static page and do not follow on the links the dynamic page may contain. Indexing and searching dynamic pages raises several extra difficulties not solved by most search engines [8,9,1]:

- the number of URLs of dynamic pages in a site may be infinite due, for example, to the use of a session number as an entry parameter in the URL, which changes on each request, despite the actual contents remaining the same;
- there is the possibility of falling into an infinite loop of page generation not easy to detect;
- many dynamic pages are not directly accessible from a link in another page but only as a result of the submission of a form filled in with appropriate values, but allowing the crawler to submit a form is questionable because it is not easy to know which values to choose (the number of combinations may be infinite) and sometimes to

submit a form causes a change in the DB, something a crawler is not supposed to do;
- the inherent dynamics of these pages leads to indexes built by the search engine that become outdated at a fast pace;
- in sites with access control, the crawlers are assigned a general public status, reducing the interest of the engine to the qualified user;
- it is hard to decide on index refresh policies due to the absence or variability of last change date.

In conclusion, a significant number of the pages that a user may be interested in are not indexed by most search engines. They constitute part of what is called the Invisible or Hidden Web [10,11,12,13,14,15], along with the pages that have no access path to them.

3 Contribution to Reducing the Invisible Web

Among the techniques to overcome some of the barriers listed in the last section one is the analysis of the Web forms that the crawler goes into. It looks for input fields with associated lists of values which may be repeatedly used in automatic submissions of that form, in order to collect the maximum number of result pages [16,17,18,19,20].

Going much further requires the use of meta-information about the DB. This is the approach followed in this paper because the motivating case study problem is a local organizational Web IS. Therefore, there is knowledge about the DB schema and access to the DB contents, which can be used by the search engine. This situation is common to other intranet IS. The solution described in this section proposes an architecture where the inside information needed is clearly identified and localized.

The proposed search engine (see Fig. 3) includes the usual components (the crawler, the indexer, and the query processor) plus a DB information processor, which concentrates the DB specific knowledge required to improve the recall.

The crawler is a complex component accomplishing several tasks. The *Page Finder* is in charge of following the links and does that with no other concern besides not going outside the target URL domain. As opposite to many crawlers, it accepts URL parameters and insists on following every link to any depth, leading both to dynamic and static pages. The target domain and exceptional URLs that must not be followed are stored in the *Configuration Data*. The starting points are stored in the *Starting URL* list. The links found in the processing of a document are queued in the *Documents URL* list, for a later visit.

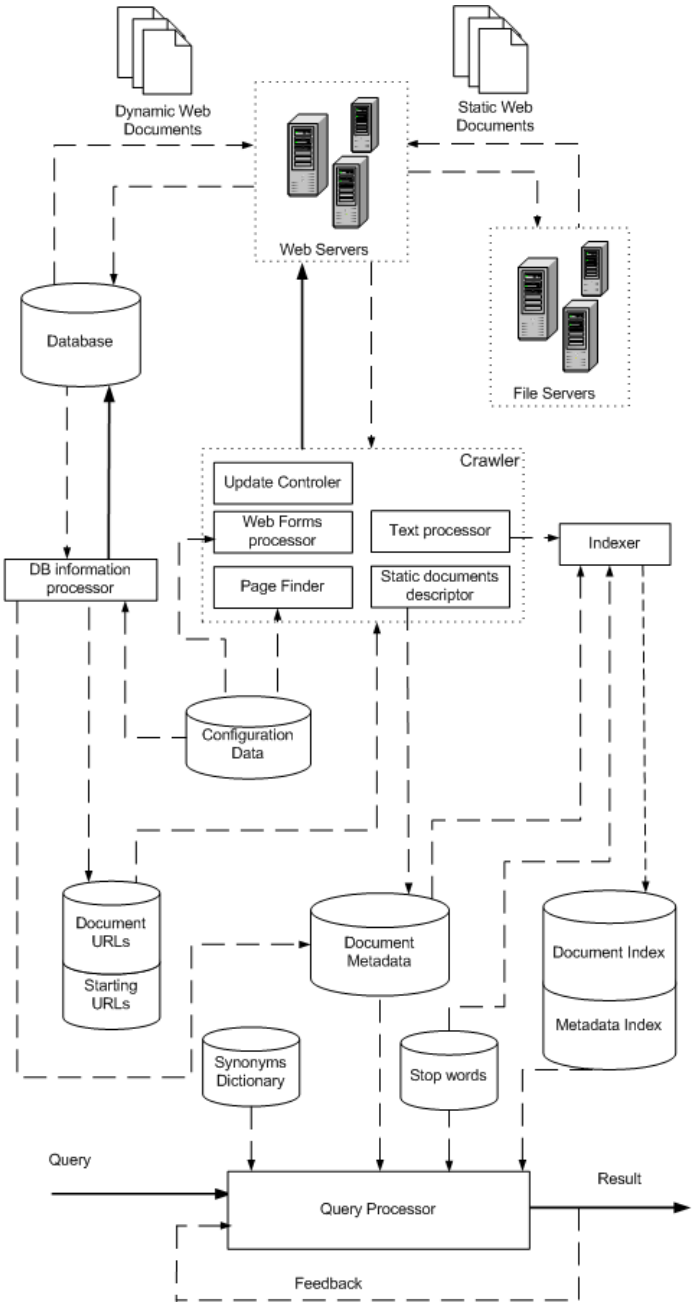

Fig. 3. System architecture

The ideas behind dealing with forms are: values in hidden fields of a form, normally used to forward previously collected data, are kept as they are and used in the automatic submission of the form; the other fields are filled in with the default values, if any, or taken to be null. Combinations of possible values present in combo-boxes and radio buttons are not currently used as, in SiFEUP, they would mean to obtain a subset of the default answer. The *Form Processor* will apply this technique to the forms classified as able to be followed by a meta-tag in the page itself. This is an example of establishing some rules for the development of new modules in the IS.

The *Update Controller* uses HTTP information (*If-changed-since* tag) to recognize a change in a static document, and the information given by the *DB Information Processor* for the dynamic pages.

The *Static Page Descriptor* gets metadata about the static pages. The metadata relative to the dynamic pages comes from the *DB Information Processor* as well. This information is stored in the *Document Metadata* table. Among other data, the size of the document is available.

The last module in the *Crawler* component is the *Text Processor.* It cleans up the text, for instance stripping HTML tags, and converts from different file formats like MS Office, PDF, etc.

The other modules have been specifically developed for this project. However, for the *Text Processor* and the *Indexer*, an Oracle Tool called *Oracle Text* has been used, and properly configured.

The *Indexer* builds two inverted file indexes whose entries are the words appearing respectively in the document and in the metadata (*Document Metadata*), except for the stop-words listed in the corresponding table.

The *Query Processor* is the third component. It receives a user query, may change it using the synonyms dictionary and the stop words, and consults the indexes to build the result, according to the relevance information of the pages.

To calculate the relevance factor, the starting point is the value given by Oracle Text. This number lies between 0 and 100 and is given by $3f(1 + \log(N/n))$ where f is frequency of the term in the document; N is the total number of documents and n is the number of documents containing the term. This way, a term showing up in many documents is less relevant than a term contained in fewer documents.

However, Oracle Text is not able to index at the same time the HTML text read and the extra metatags produced by the meta-descriptor of the dynamic pages. So, a modification of the relevance factor has been done to account for this and for the importance of the page in its environment. The first component is based on the contents of the two indexes and is weighted between 0 and 30. The second component gives more relevance to the pages based on tables which are central and possess more relationships going into them. This factor is weighted typically between 0 and 15 and is responsible for giving more relevance to the home page of a professor than to his list of publications, where his name is likely to appear a lot more frequently.

To complete the description of the system developed, the *DB Information Processor* gets metadata about the dynamic pages, including criteria to support the refreshing policy and improving the pages relevance factor. It also deals directly with DB

columns that are known, in the *Configuration Data*, to contain URLs, like the links to the teacher or the student non-official pages.

As the RDBMS used by the case study IS is Oracle, the language chosen for the development has been PL/SQL. The tests performed with the system indexed about 140 000 pages, both dynamic and static, including non HTML documents.

The sample test queries used were taken from a two-days log of the actual searches performed on the main system forms by the users. Additionally some experimentation has been made by experts, for more complex and critical queries. These included operators like AND, OR and distance in the text between certain specific terms. The answer time to queries varied from negligible to 10 seconds, with a typical value around 3 seconds.

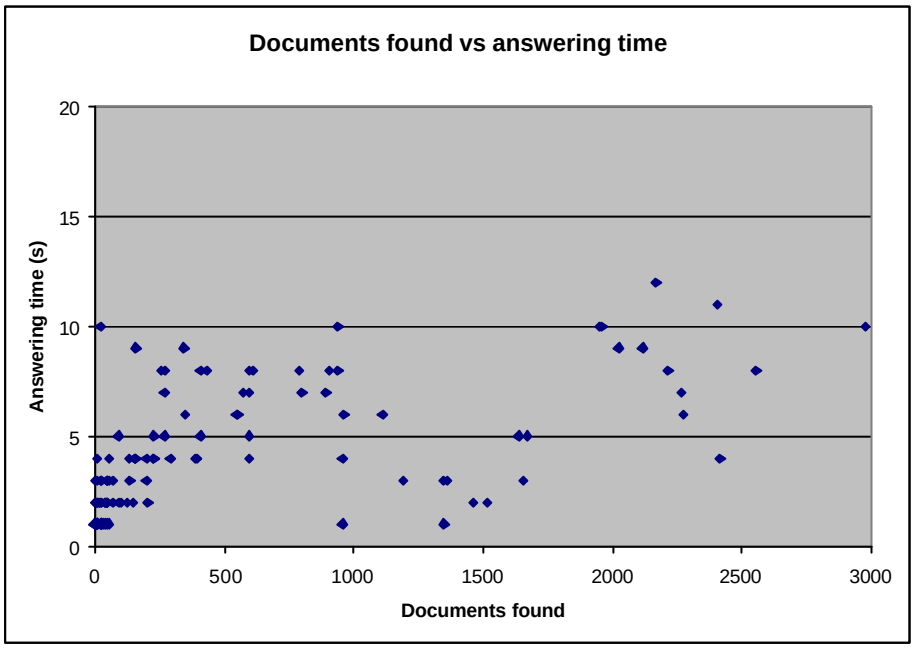

Fig. 4. Search results

The analysis of the graphic in Fig. 4 shows that some queries produce large quantities of documents. They correspond to specifying a common single term query. This is not a bad result from the viewpoint of finding the appropriate documents, because the more relevant are shown first. However, it implies a longer computation time while the user is waiting, spent mainly on sorting the result. Fortunately, most queries lie on the left side of the graphic meaning shorter and, most of the time, faster answers to 3-4 term query.

4 Conclusion

Current IS are subject of an apparent contradiction. On one hand they are built on top of databases storing highly structured information. On the other hand, the presentation of this information is done via Web interfaces made of HTML pages, usually not structured. The bridge between the DB and the user is made by dynamic Web sites, in the organizational intranet which is, sometimes, accessible by the whole Internet. Using complementary information from the RDBMS relationships in the pages generated from that data, a better recall figure is obtained.

The main conclusion is that adding to the IS the generic textual search composed with metadata from the DB improves the access to information, with respect to the traditional method based on menus, hyperlinks and Web forms, especially for the occasional users. It has been noticed that, even for regular users, it increases the visibility of certain contents in the periphery of the system.

The criteria followed in modifying the relevance factor, though always subjective, proved to bring to the first rows the kinds of pages the user expects. For example, asking for a person's name retrieves, most of the times, the corresponding official personal page. This happens because those pages are central in the system. Sometimes, the first row is the bibliographic page of that person, as it contains multiple occurrences of the name as the author of the different publications. Finding the desirable weight factors requires a careful tradeoff.

The results of the interaction with an Information Retrieval system are of a different nature from those obtained from a database. When querying a DB, the answer given to the user is a record set corresponding, in a deterministic and objective way, to the query, irrespective of the DBMS used. In an interaction with an Information Retrieval system, the answers are less determined by the query, though some level of relationship always exists. Determinism in these searches is derived more from the nature of the document preprocessing techniques like index construction and clustering, and query interpretation, than from the structure of the information. Different systems typically produce results not fully coincident.

References

1. LAWRENCE, Steve ; GILES, C. Lee - Accessibility of Information on the Web. Nature. 400, no. July 1999 (1999) 107-109.
2. JONES, Karen Spark ; WILLETT, Peter - Overal Introduction. In Karen Spark Jones e Peter Willett, eds.- Readings in Information Retrieval. Morgan Kaufmann, 1997, 1-7.
3. FRATERNALI, P. - Tools and approaches for developing data-intensive {Web} applications: a survey. ACM Computing Surveys. 31, no. 3 (1999) 227-263. http://www.ucsd.edu/cse132B/WSMT.pdf. 20-12-2002.
4. DAVID, Gabriel ; RIBEIRO, Lígia Maria - Getting Management Support from an University Information System. Proceedings of the European Cooperation in Higher Education Information Systems, EUNIS99, Espoo, Finland, 1999.

http://sifeup.fe.up.pt/sifeup/WEB_BIB$PESQUISA.download?p_file=F6934/Eunis99sent.doc. 19-12-2002.
5. DAVID, Gabriel ; RIBEIRO, Lígia Maria - Impact of the Information System on the Pedagogical Process. 7th International Conference of European University Information Systems, EUNIS 2001, Berlin, Germany, 2001. http://sifeup.fe.up.pt/sifeup/WEB_BIB$PESQUISA.download?p_file=F14804/eunis2001_b_final.doc. 19-12-2002.
6. RAPPOPORT, Avi - Search Engines: The Hunt is on. 2000. http://www.networkcomputing.com/1120/1120f1.html. 02-07-2002.
7. ANDERSSON, Eve, GREENSPUN, Philip ; GRUMET, Andrew - Internet Application Workbook. 2001. http://philip.greenspun.com/internet-application-workbook/. 21-12-2002
8. BERGMAN, Michael K. - The Deep Web: Surfacing Hidden Value. The Journal of Electronic Publishing. 7, no. 1 (2001). http://www.press.umich.edu/jep/07-01/bergman.html. 20-12-2002, http://www.brightplanet.com/deepcontent/tutorials/DeepWeb/. 20-12-1002.
9. LAWRENCE, Steve ; GILES, C. Lee - Searching the World Wide Web. Science. 280, no. 5360 (1998) 98-100. http://www.neci.nec.com/~lawrence/science98.html. 9-12-2002.
10. BARKER, Joe - Invisible Web: What it is, Why it exists, How to find it, and Its inherent ambiguity. 2002. http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/InvisibleWeb.html. 20-12-2002.
11. BOTLUK, Diana - Exposing the Invisible Web. LLRX. no. October 1999 (1999). http://www.llrx.com/columns/exposing.htm. 21-12-2002.
12. HARTMAN, Karen ; ACKERMANN, Ernest. The Invisible Web. A presentation at Computers in Libraries 2000, 2000. http://www.webliminal.com/essentialweb/invisible.html. 20-12-2002
13. LACKIE, Robert J. - Those Dark Hiding Places: The Invisible Web Revealed. 2001. http://library.rider.edu/scholarly/rlackie/Invisible/Inv_Web.html. 20-12-2002.
14. SHERMAN, Chris ; PRICE, Gary - The Invisible Web: Finding Hidden Internet Resources Search Engines Can't See. Independent Publishers Group, 2001.
15. CLYDE, Anne - The Invisible Web. Teacher Librarian. 29, no. 4 (2002). http://www.teacherlibrarian.com/pages/infotech29_4.html. 20-12-2002.
16. BENEDIKT, Michael, FREIRE, Juliana ; GODEFROID, Patrice. VeriWeb: Automatically Testing Dynamic Web Sites. Proceedings of the 2002 WWW conference, Honolulu, Hawaii, USA, 2002. http://www2002.org/CDROM/alternate/654/ 22-12-2002.
17. LIDDLE, Stephen W., EMBLEY, David W., SCOTT, Del T. ; YAU, Sai Ho. Extracting Data Behind Web Forms. Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002a. http://www.deg.byu.edu/papers/vldb02.pdf. 20-12-2002.
18. LIDDLE, Stephen W., YAU, Sai Ho ; EMBLEY, David W. On the Automatic Extraction of Data from the Hidden Web. Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS-2001), Yokohama, Japan, 2001. http://www.deg.byu.edu/papers/daswis01.pdf. 22-12-2002.
19. RAGHAVAN, Sriram ; GARCIA-MOLINA, Hector - Crawling the Hidden Web. Technical Report, Computer Science Department, Stanford University, 2000. 2000-36. http://dbpubs.stanford.edu/pub/2000-36. 22-12-2002.
20. YAU, Sai Ho - Automating the Extraction of Data Behind Web Forms. Masters Thesis, 2001. http://www.deg.byu.edu/papers/TonyYauThesis.doc. 22-12-2002.