# Towards Automatic Incorporation of Search engines into a Large-Scale Metasearch Engine

Zonghuan Wu, Vijay Raghavan
Hua Qian, Vuyyuru Rama K
*CACS, Univ. of Louisiana at Lafayette*
*{zwu,raghavan}@cacs.louisiana.edu*

Weiyi Meng, Hai He
*Dept. of Computer Science*
*SUNY at Binghamton*
*meng@binghamton.edu*

Clement Yu
*Dept. of Computer Science*
*Univ. of Illinois at Chicago*
*yu@cs.uic.edu*

## Abstract

*A metasearch engine supports unified access to multiple component search engines. To build a very large-scale metasearch engine that can access up to hundreds of thousands of component search engines, one major challenge is to incorporate large numbers of autonomous search engines in a highly effective manner. To solve this problem, we propose automatic search engine discovery, automatic search engine connection, and automatic search engine result extraction techniques. Experiments indicate that these techniques are highly effective and efficient.*

Keywords: Metasearch Engine; Web Data Extraction

## 1. Introduction

In the context of a metasearch engine, the process of incorporating search engines consists the process of discovering search engine interfaces, connecting to them and extracting result documents from search engine returned webpages.

A significant problem in building a very large-scale metasearch engine that supports unified access to hundreds of thousands of search engines [10] [13] is the impracticality of manually incorporating these search engines. Even if this were possible, maintenance would be a nightmare. Changes to search engines take place from time to time, often leaving a search engine unusable for metasearch unless corresponding changes are made in the metasearch engine. Manual maintenance therefore is hardly practical. We believe that the entire process of search engine incorporation should be automated, to enable construction and maintenance of very large-scale metasearch engines.

The three major components that are essential to achieve automation are:

**1. Automatic search engine discovery.** Discover (identify) search engines from millions of websites on the Web.
**2. Automatic search engine connection.** Automatically connect to each discovered search engine so that user queries submitted to the metasearch engine are forwarded to search engines and search results from search engines are returned to the metasearch engine.
**3. Automatic search result extraction.** Automatically analyze each result page returned from a search engine for a query, extract useful information, such as the number of retrieved documents for the queries, URLs of result documents and so on from the page.

The state of the art large-scale metasearch engines, like profusion, can manage metasearch over around 1,000 search engines but not more. In this paper, through experiments on the initial implementation of the proposed three-component search engine incorporation framework, we demonstrate the potential capability for a metasearch engine to handle much more search engines, even in terms of hundreds of thousands.

The rest of the paper is organized as follows. Brief background information is provided in Section 2. Related works are reviewed in Section 3. Crawler-based search engine discovery, search engine connection and automatic search engine result extraction are discussed respectively in sections 4.1, 4.2 and 4.3. Experimental statistics are presented and analyzed in Section 5. Finally, in Section 6, conclusions of the paper are highlighted with a brief discussion of future work.

## 2. Background Information

### 2.1. Web Search Engines

In this work, both the traditional crawler-based "Surface Web" search engines and "Deep Web" databases that have Web search interfaces are regarded as Web search engines. Please refer to [2] to for detailed discussion of Surface Web and Deep Web. Also, in [10] and [13], we described in detail issues that arise due to the largeness of the number of search engines that we are aiming to metasearch.

We call a webpage from which users can type in queries a *search engine interface*, or a *search engine page*.

### 2.2. Search Engine Form

On the *search engine interface*, there is at least one *HTML form*, allowing users to submit queries. To identify such forms is of crucial importance in discovery. Please refer to [12] to learn more about HTML forms.

### 2.3. Search Engine Result Page

After a query is sent to a search engine, a *result page* is returned. Usually, retrieved documents are listed on the page, with their descriptions and URLs. Some other important information about the search (such as the number of retrieved documents for a query) may be present. A metasearch engine needs to extract result document URLs and other information from result pages returned from all metasearched search engines to formulate its own result pages to return to metasearch engine users.

## 3. Related Work

### 3.1. Search Engine Discovery

Most metasearch engines assume that component search engines are discovered manually. Search engine directory services such as *Completeplanet* and *InvisibleWeb* that focus on organizing Deep Web databases claim to have developed techniques to identify/discover search engines automatically or semi-automatically [2] [6]. However, technical details are proprietary and not publicly available.

### 3.2. Automatic Search Engine Connection

For a metasearch engine with a large number of component search engines, connection automation is an essential requirement since manual analysis is time-consuming and unfeasible, not to mention the difficulty in tracking occasional search engine interface changes. Metasearch engines and search engine Directory Services such as *Profusion*, *CompletePlanet* and *InvisibleWeb* all claim that they have technologies to connect to search engines automatically or semi-automatically. Unfortunately, not much detailed information is available in open literature.

### 3.3. Search Engine Result Extraction

Early manual approaches of *wrapping* webpages (i.e. extracting important information from webpages) [1] [5] [11] have many recognized shortcomings, mainly due to the difficulty in *wrapper construction* and *maintenance*. Recently, however, many semi-automatic or automatic tools have been proposed for wrapper building. A few well-known examples include the ARANEUS project, developed at University of Basilicata and University "Roma Tre" in Italy [9], XWrap/XWrap Elite Project, developed at Georgia Institute of Technology [8]. Progress has been made lately in the RoadRunner project at University of Basilicata and University "Roma Tre" in designing a highly automatic data extraction approach by comparing HTML structures of two (or more) given sample pages that share a similar representation pattern. The layout schema is then generated [3]. A survey of recent wrapper developing techniques can be found in [7].

Search engine result extraction is a special case of Web data extraction. Again, no detailed information is disclosed regarding the data extraction technology employed by current commercial metasearch engines.

## 4. Proposed Techniques

### 4.1. Automatic Search Engine Discovery

We propose a two-step process (crawling and filtering) to discover search engines.

Step 1. Crawling. A special Web crawler is created to fetch webpages. Each webpage is regarded as a potential search engine interface page.

Step 2. Filtering. A set of recognition rules is then applied to determine if the page has a search engine interface. The following are the main filtering rules in the current implementation:

(1) The HTML source file of a search engine interface page should contain at least one HTML form.

(2) The form must also have a *text input control* for query input.

(3) At least one of a set of keywords such as "search," "query" or "find" appears either in the form tag or in the text immediately preceding or following the "<form>" tag.

### 4.2. Automatic Search Engine Connection

Automatic search engine connection involves four steps.

1. Parse HTML source code of a candidate webpage into a tree structure of HTML tags. For the sake of illustration, Figure 1 is a tree structure presentation for the following simple HTML page:

```
<html>
   <head>
      <title>example</title>
   </head>
   <body>
```

```
            <form>…</form>
         </body>
      </html>
```

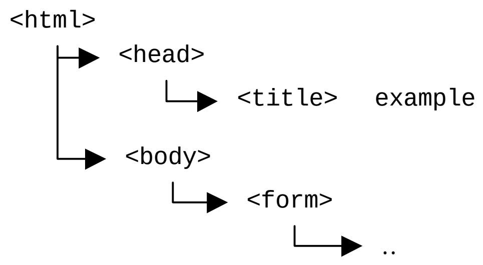

**Figure 1**

2. Extract form parameters and attributes from the FORM sub-tree and save them into an XML formatted file, which we call the *search engine description file* of the search engine.
3. Read the form information from the *search engine description file* and re-construct a test query string.
4. Send the test query to check connection correctness. If some http error code is returned, showing connection failure, further manual analysis may be needed to handle the exception.

### 4.3. Search Engine Result Extraction

The two pieces of information extracted from the returned page are: (1) The URLs and/or snippets of retrieved webpages. (2) The total number of retrieved documents, as described in Section 2.

There are two steps in automatic result extraction.

Step 1. A so-called "*impossible query*" (a query consisting of a non-existent term) is sent. All URLs on the result page are useless in terms of document retrieval. They are recorded and easily excluded from result pages for other queries. The layout pattern of the "Result Not Found" page is also recorded for future reference.

Step 2. A number of program-generated queries are sent. The result pages are compared against each other and all the common URLs are marked as useless. Two tasks are yet to be undertaken:

**1. Find the URLs of returned result documents**:

The patterns of result document URLs on the same result page are very similar. We use a unique feature, called "*Tag Prefix,*" to represent the layout pattern.

The *Tag Prefix* of a URL is a sequence of HTML tags that appear before a URL and typically on the same line as the URL.

For example, a section of HTML code may look like this:

```
<table> <tr> <td> <b> <a href=http://url1.html>url1
 Caption</a> </b> </td> </tr> … </table>
```

The tag prefix of the URL http://url1.html is "<tr><td><b>" since the tag "<tr>" implies change of a line. Other tags indicating such a change include "<p>", "<br>", "<table>", "<hr>", "<LI>", and so on.

**2 Find the number of matched documents**

This information usually appears either at the beginning or at the bottom of a result page on a text line, which may be set apart by some specific features, such as the presence of numeric numbers, or special keywords (e.g. *found, returned, matches, results,* etc.), or the "*of*" pattern (e.g. 1-20 *of* 200), or the query terms. We call this line "*document hits line*". It needs to be automatically extracted.

## 5. Experimental Results

Following experiments are conducted to evaluate the performance of these major components.

### 5.1. The Experiment on Search Engine Discovery

We carried out the experiment as follows:

1. The RDF dump from http://dmoz.org, is downloaded. It is said to be the largest human-edited directories, containing multi-million Webpages. A total of *519* webpages are collected, each having at least one form, as a result of random selection for the purpose of keeping the generality of the experiment.
2. Manual check reveals that *307* pages contain at least one search engine form. (Report A).
3. The discovery program reports 286 search pages from the same collection of webpages (Report B).
4. There are *286* URLs appearing in both reports, i.e., all of them are correctly identified. A total of *21* URLs are listed only in Report A, meaning that our search engine discovery component missed *21* search engines. There is no misclassification. The discovery correctness is 93% (286/307).

In almost all the *21* cases, it is the failure to locate "search", "find" or other keywords within the search engine forms that give rise to the problem. In one case, however, the form is written in Flash instead of regular HTML.

These experimental results indicate that our Search Engine Discovery Logic is simple yet highly effective. By crawling the Web, 9 out of 10 search engine forms can be automatically discovered by our discovery component. Thus, this process makes it feasible to locate hundreds of thousands of surface web and deep web search engines on the surface Web and in the deep Web.

### 5.2. The Experiment on Search Engine Connection

In this experiment,

1. The search engine connection program is used on the previously discovered *286* search engine pages and reports a total of *326* search engine forms identified (note that one page may contain more than one search engine form).
2. A sample query is sent to each search engine, both through search engine connection program and through Web browser (i.e.. manually).
3. The result pages retrieved by program and through the browser are compared. If they are the same, connection is successful.

Result shows that *242* Search engine forms are successfully connected. There are *18* search engines not working properly. Additionally, *9* search engine forms using Google's processing agent only allows access via a browser. Any effort to connect using a program is effectively denied. As a result, connection success is over 80% (242/(326-9-18)).

Among the *57* cases of unsuccessful connection, most forms either adopt Javascript or are poorly coded grammatically in HTML, both of which prevent the program from correct parsing. In a few cases, there are site redirections that the program fails to track.

This experiment shows that the automate connection process is quite effective in general. But the loose HTML grammar regulations and a few Web technologies, such as JavaScript, bring complications to the process.

### 5.3. The Experiment on Search Engine Result Extraction

Small-scale experiments have been done and showing encouraging results. We are still in the process of refining result extraction algorithms. Systematical experiment will be carried out in the near future.

## 6. Conclusions and Future Work

In this paper, we have proposed a framework consisting of three-component techniques to automate incorporating search engines into metasearch engines, which is essential to the automation of the process of building, maintaining, and running a large-scale metasearch engine. Initial experiments indicate that the proposed search engine discovery and search engine connection techniques are very effective and feasible.

The future work includes fine-tuning, extending and integrating the search engine discovery, connection and result extraction techniques.

## 7. References

[1] Atzeni , P. and Mecca, G., "Cut and Paste," In PODS'97.

[2] Bergman M., "The Deep Web: Surfacing the Hidden Value," www.completeplanet.com/Tutorials/DeepWeb/index.asp, BrightPlanet, 2000.

[3] Crescenzi, V., Mecca, G., And Merialdo, P., "RoadRunner: Towards automatic data extraction from large Web sites," In Proceedings of the 26th International Conference on Very Large Data Bases (Rome, Italy, 2001), pp. 109-118.

[4] Robert B. Doorenbos, Oren Etzioni and Daniel S. Weld, "A scalable comparison-shopping agent for the World-Wide Web," The first International Conference on Autonomous Agents Feb 5-8, 1997.

[5] Huck, G., Frankhauser, P., Aberer, K., and Neuhold E. J., "Jedi: Extracting and synthesizing information from the web," In CoopIS'98.

[6] FAQ from www.InvisibleWeb.com

[7] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva and Juliana S. Teixeira, "A Brief Survey of Web Data Extraction Tools," SIGMOD Record, pp. 84-93, June 2002.

[8] Liu, L., Pu, C., and Han, W., "XWRAP: An XML-enabled wrapper construction system for Web information sources," In Proceedings of the 15th International Conference on Data Engineering (San Diego, CA, 20000), pp. 611-621

[9] Mecca, G., Atzeni, P., Masci, A., Merialdo, P., and Sindoni, G., "The Araneus Web-Base Management System," In Proceedings of the ACM SIGMOD International Conference on Management of Data (Seattle, WA, 1998), pp. 544-546

[10] Meng, W., Wu, Z., Yu, C., Li, Z., "A Highly-Scalable and Effective Method for Metasearch," ACM Transcactions on Information Systems, Vol. 19, No. 3, pp. 310-335, July 2001.

[11] Sahuguet A. and Azavant, F., "Web ecology: Recycling HTML pages as XML documents using W4F," In WebDB'99.

[12] http://www.w3.org/TR/REC-html40/interact/forms.html, HTML 4.01 Specification, W3C Recommendation 24 December 1999

[13] Wu, Z., Meng, W., Yu, C., and Li, Z., "Towards a Highly-Scalable and Effective Metasearch Engine," Proc. Of Tenth World Wide Web Conference (www10), Hongkong, May 2001.