Information Discovery, Extraction and Integration for the Hidden Web

Jiying Wang

Department of Computer Science
University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong
cswangjy@cs.ust.hk

Abstract

In this paper, we report our initial investigations on the problems of automatically extracting data objects from a given hidden-web source (i.e., the web site with an HTML search form) and automatically assigning semantics to the extracted data. We also propose some future work to address the problem of information discovery and integration for hidden-web sources.

1. Introduction

Currently, there exist a number of standard tools, such as search engines and hierarchical indices that can help people in finding information. However, a large number of the web pages returned by filling in search forms are not indexable to most search engines since they are dynamically generated by querying a back-end (relational or object-relational) database. Unlike text documents, the dynamically generated web pages usually contain complex data objects with nested structures. Referred to as the Deep Web [1] or Hidden Web [5], the set of such pages is estimated to be around 500 times the size of the "surface web". Besides the larger volume than the surface web, the deep web also has a higher quality and a faster growth rate [1].

Consider, for example, a user who wants to search for some information, such as configuration and price of a notebook computer, before he/she shops on the Web. Since such information only exists in the back-end databases of notebook vendors, the user first needs to discover the URLs of several notebook vendors, go to the homepages of the vendors, send queries through HTML forms, extract the relevant information from the result web pages, and finally compare or integrate the results from multiple sources. Therefore, there arises the need for new information services that can help users locate information in the hidden web, i.e., to discover the promising information sources, disseminate the queries, extract the corresponding results from web pages and integrate the retrieved information.

To minimize user effort in such an information retrieval process and enable the tools to scale with the growth of the web, we explore the problem of automatically interacting with information sources in the hidden web. This problem has four aspects:

- **Information discovery:** How to automatically locate the web sites containing structured data of interest to the user?
- **Information extraction:** How to induce wrappers to extract relevant data objects from discovered web sources?
- **Information understanding:** Having extracted data objects with complex structures, how to automatically or semi-automatically annotate or label the fields of the extracted data?
- **Information integration:** How to integrate the various data objects from multiple sources with or without knowing their schemas?

In this paper, we concentrate our investigation of these four aspects on some specific web sites, referred to as ***webbases***, which provide a complex HTML search form for users to query the back-end databases. Webbases usually employ templates (script programs) to generate query result pages carrying multiple instances of data objects, and those data objects usually follow a particular alignment and format.

For webbases, some of these four aspects have been studied for years while some of them have not even been touched. *Information discovery* tools, such as crawlers and spiders for general search engines, nowadays mainly focus on searching for web pages that are indexable by words or terms. The data objects contained in webbases have been ignored until the concept of the "hidden" web arose. Recently some companies provide directories for categorizing webbases, but they mainly depend on expertise judgment. How to automatically locate a

"suitable" webbase for users to query is still an open question.

For some given web pages containing regular structured data objects, *information extraction* tools are designed to induce wrappers from the pages. Previous approaches on wrapper construction either are hand-tailored for each source or need good examples for them to learn. Currently, work on the fully automatic wrapper induction problem is still in a nascent stage ([2], [3] and [4]).

After extracting data objects from webbases, although understanding the semantic meaning of the data attributes is very critical for latter data manipulation such as querying and reporting, the *information understanding* problem still waits for a good solution.

Among these four aspects, *information integration* is the one that has been examined the longest and there exist many good algorithms (see [5] and [9] for surveys). However, most of the existing approaches make the assumption that the schemas of multiple sources are known in advance, which is not a realistic assumption for the case of webbases. The possibility of integrating web data with partial schemas or no schema information at all is interesting and worth investigation.

In this paper, we report in section 2 our initial investigations on two of these aspects: the problem of automatically extracting data objects from a given webbase and the problem of automatically assigning semantics to the extracted data ([10], [11] and [12]). Furthermore, we propose some future work in section 3 to address the remaining two questions: information discovery and integration.

2. Data Extraction and Label Assignment

Given a web site with an HTML search form, we design a system ***DeLa*** (*D*ata *E*xtraction and *L*abel *A*ssignment). *DeLa* is built to automatically induce a regular expression wrapper from the result pages of a given webbase, and also to automatically extract data objects using the induced wrapper and assign semantic labels to the data attributes. In this section, we first present the employed data model and then introduce the system architecture for *DeLa*. We use a simple example to illustrate how the four components of the system work.

2.1 Data model

In this paper, we employ the *nested type* as an abstraction to model the data objects contained in result pages returned by webbases. For example, the top figure in Figure 1 shows a web page of an online bookstore with a search form in its left side for users to query its back-end database of books. If we type a word, such as "Harry Potter", in the textbox with label "Title", we get a result page, shown in the bottom figure of Figure 1, containing four book objects with "Harry Potter" appearing in their title. In this page, each book has a book title, zero or one author and one or more edition information, e.g., the first book has no author and the third book with the title "A Guide to the Harry Potter Novels" has two editions, a "hardcover" in "Apr, 2002" and a "paperback" in "Apr, 2002". Therefore, we can model the book object contained in this page as the following nested type: ***Book*** < Title, (***Author*** < Name >)?, {***BookEdition*** < Format, Publish Date, Publisher >} >, where the symbols < > represent an unordered list tuple, the symbols { } represent a set and the symbol ? represents an optional attribute.

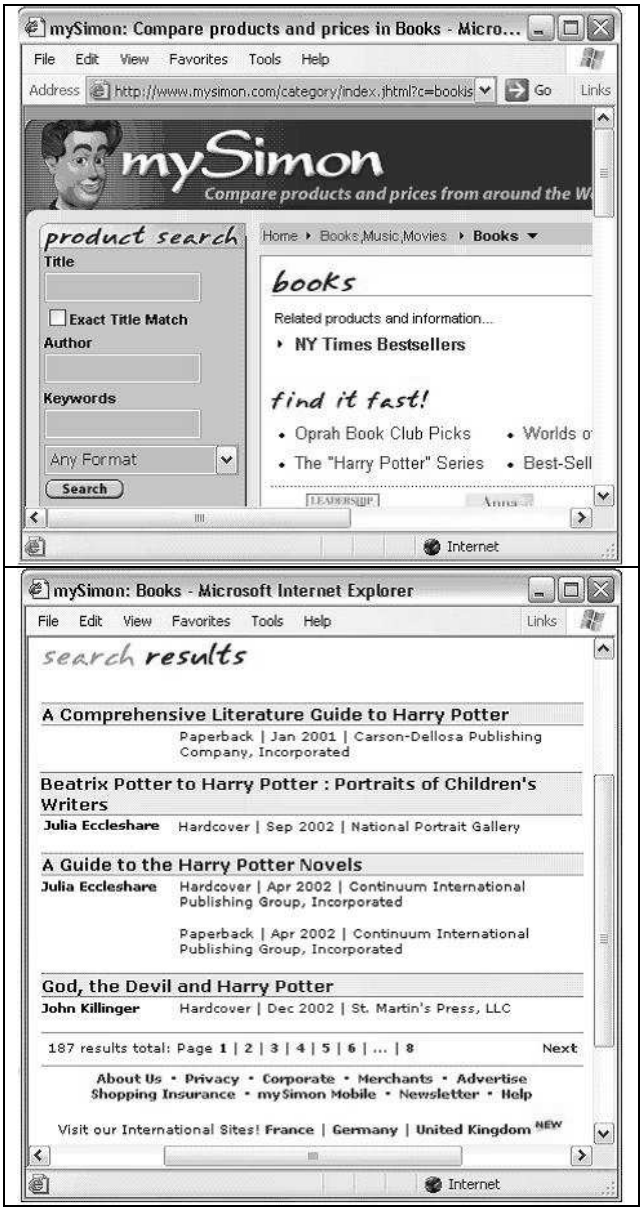

Figure 1. An example web site with an HTML search form and an example result page.

We consider the data objects contained in web pages as string instances of the implied nested type of its back-

end database, where these instances are encoded in HTML tags. Thus, a *regular expression* can be employed to model the HTML-encoded version of the nested type. In Figure 2 we show the HTML code for the first book and the third book contained in the web page in Figure 1 and the corresponding regular expression wrapper to extract book instances from the page. After we extract data objects from a web page, we choose to re-arrange the extracted data from the web page in a table manner such that each row of the table represents a data instance and each column represents a data attribute. For example, Figure 3 shows the table containing book instances extracted from the result page in Figure 1. Note that it is not necessary to represent the data in only one table where some attributes are duplicated; we can easily build more tables or relations for multiple-value attributes according to the regular expression wrapper. Here, we choose to unify the data representation of each web site as a single table, because it is easier for the later data integration.

HTML code of the embedded data:

```
<TR><TD><B> A Comprehensive   ... </B></TD></TR>
<TR>
   <TD> </TD>
   <TD> Paperback | Jan 2001 | Carson-Dellosa  ...  </TD>
</TR>
...
<TR><TD><B> A Guide to the ...  </B></TD></TR>
<TR>
   <TD><B> Julia Eccleshare </B></TD>
   <TD> Hardcover | Apr 2002 | Continuum ... <BR>
        Paperback | Apr 2002 | Continuum ... <BR> </TD>
</TR>
...
```

Corresponding regular expression wrapper:

```
<TR><TD><B> text <B></TD></TR>
<TR>
   <TD> (<B> text </B>)? </TD>
   <TD> (text <BR>)* </TD>
</TR>
```

Figure 2. HTML code for the example data and the corresponding wrapper.

..\book_html\example.htm

Title	Author	Format	Date	
A Comprehensive Literat...		Paperback	Jan 2001	Carson-D...
Beatrix Potter to Harry Pot...	Julia Eccleshare	Hardcover	Sep 2002	National ...
A Guide to the Harry Pott...	Julia Eccleshare	Hardcover	Apr 2002	Continuu...
A Guide to the Harry Pott...	Julia Eccleshare	Paperback	Apr 2002	Continuu...
God, the Devil and Harry ...	John Killinger	Hardcover	Dec 2002	St. Martin'...

Figure 3. Table representation of the extracted data.

2.2 System architecture

The system consists of four components as shown in Figure 4.

- **Form crawler.** Given a web site with an HTML search form, the form crawler collects the labels of each element contained in the form and sends queries though the form elements to obtain the result pages containing data objects. We adopt the hidden web crawler, *HiWe* [8] for this task. *HiWe* was built on the observation that "most forms are usually associated with some descriptive text to help the user understand the semantics of the element." It is equipped with a database that stores some values of the task-specific concepts and assigns those values as queries to the form elements if the form labels and the database labels match. Similarly, our system, *DeLa,* further utilizes the descriptive labels of the form elements by matching them to the attributes of the data extracted from the query-result pages. Readers can refer to [8] for more details about the form crawler.

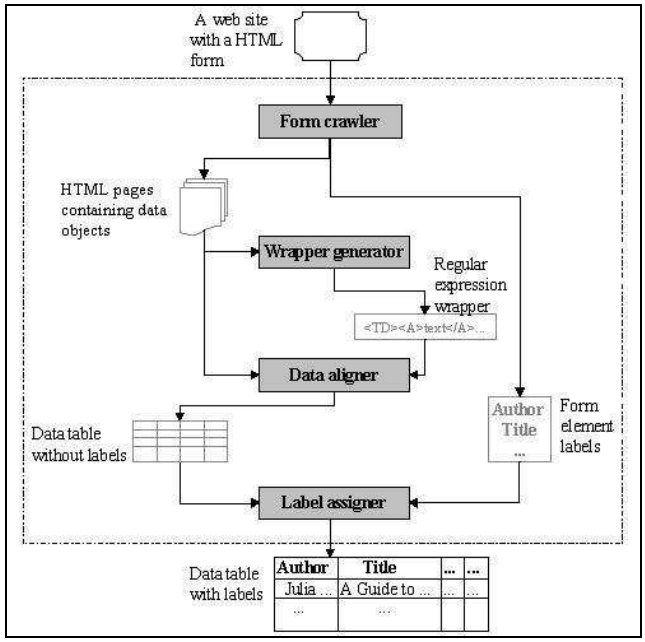

Figure 4. The *DeLa* Architecture.

- **Wrapper generator.** The pages collected by the form crawler are output to the wrapper generator to induce the regular expression wrapper based on the pages' HTML-tag structures. Since pre-defined templates generate the web pages, the HTML tag-structure enclosing data objects may appear repeatedly if the page contains more than one instance of a data object. Therefore, the wrapper generator first considers the web page as a token sequence composed of HTML tags and a special token "text" representing any text string enclosed by pairs of HTML-tags, then extracts repeated HTML tag substrings from the token sequence and induces a regular expression wrapper from the repeated substrings according to some hierarchical relationships among them. The wrapper generator is inspired by previous work on *IEPAD* [2].

We present in [12] a more technical comparison between our work and theirs.

- **Data aligner.** Given the induced wrapper and the web pages, the data aligner first extracts data objects from the pages by matching the wrapper with the token sequence of each page. It then filters out the HTML tags and rearranges the data instances into a table similar to the table defined in a relational DBMS, where rows represent data instances and columns represent attributes. Note that the extracted data objects may have optional or multi-valued attributes, e.g., in Figure 1 the first book has no author listed and the third book has two editions. Furthermore, sometimes several attributes of the data object are encoded together into one text string that is not separated by HTML tags, e.g., the format, publish date and publisher information of the books in Figure 1. Therefore, the data aligner needs to distribute multiple values of one data attribute into several rows and separate the attributes encoded in one string to several columns, if possible. Figure 3 shows the table representation of the four books contained in Figure 1, where the third book with two editions is re-arranged into two rows (the third and the fourth) and the last three attributes are separated.
- **Label assigner.** The label assigner is responsible for assigning labels to the data table by matching the form labels obtained by the form crawler to the columns of the table. The basic idea is that the query word submitted through the form elements will probably reappear in the corresponding fields of the data objects, since the web sites usually try their best to provide the most relevant data back to the users. For example in Figure 1, the web page is generated to answer the query "Harry Potter" submitted through the form element labelled by "Title". Therefore, the first column of the data table in Figure 3, with "Harry Potter" appearing in all five rows, can be marked as "Title", the label of the form element. Similarly, the second and the third columns are marked as "Author" and "Format", which also comes from the form element labels in Figure 1. Note that the mappings between form elements and data attributes are usually not exactly one-to-one. The label assigner sometimes needs to employ other information, such as the data format, as clues to understand the semantics of the data objects. For example, in Figure 3, the fourth column is marked as "Date", since the data it contains are in a date format.

To test the performance of *DeLa* on wrapper induction and label assignment, we employ 3 categories of hidden webbases (web sites with a complex HTML search form) and manually collect some result pages from those web sites. Next, we use *DeLa* to generate a wrapper for each web site, extract data objects from the result pages and restoring the retrieved data with discovered attribute labels into a table. The experimental results in [10] indicate that *DeLa* performs very well in automatically inducing wrappers (over 90% precision) and assigning meaningful labels to the retrieved data (over 80% correctness). As far as we know, ROADRUNNER [3] and IEPAD [2] are the only works that try to solve the problem of fully-automatic wrapper induction. However, both of them have limitations. ROADRUNNER assumes no disjunctive attributes and IEPAD assumes no nested attributes of data objects to be extracted. In contrast, we believe such assumptions may not be realistic in the web, and our work can handle not only plain-structured data but also nested-structured data possibly with disjunctive attributes. Moreover, our work also demonstrates the feasibility of heuristic-based, semantic-label assignment and the effectiveness of the employed heuristics, which we believe sets the stage for more fully automatic data annotation of web sites.

3. Webbase Discovery and Web Information Integration

As mentioned before, our final goal is to develop a series of information services from source discovery to data integration, which can automatically interact with information sources in the hidden web to help users locate high quality and relevant information. In our previous work, we concentrated on the problem of automatic wrapper induction for web pages containing regular-structured data objects. We also explored the possibility of heuristic-based automatic annotation of data objects extracted from webbases. The remaining problems in building an information integration system for webbases will be our future focus, i.e., the problem of webbase discovery and integration. In fact, these two problems are strongly interdependent, i.e., the results of the discovery phase are the sources for integration and how to integrate information sources (or what we refer to as integration models) directly decides the factors or features of webbases to be examined at the discovery phase.

Currently search engines are the most widely-used tools to help users locate or discover desired information. Adopting most of their techniques from Information Retrieval (IR), search engines index web documents by keywords and consider the web pages to be relevant once they contain the query keywords. There are mainly three ways employed by search engines to discover new web pages on the web (web pages that have not been visited yet): random IP attempt, user registration and following the hyperlinks of visited pages. These ways are effective in locating static web pages, i.e. the web pages that can be accessed by a specific URL address. However, web pages in the hidden web cannot be identified by these traditional ways, because they are generated on the fly to answer user queries by retrieving data from back-end server databases.

Unlike collecting static web pages, it is more difficult to design spiders or crawlers to automatically collect dynamically-generated web pages since the spiders need

to deal with HTML search forms, which are designed for human use. To the best of our knowledge, work on the problem of webbase discovery is still at a nascent stage. Past works on constructing wrappers for webbases or integrating information from webbases usually assume that the target web sources are already known. Thus, they ignore the need to help users find the desired sources they want to extract data from or they want to integrate. Currently some commercial companies, such as CompletePlanet.com and Invisibleweb.com, provide some categories of webbases according to their topics. To build such categories, web sites were manually examined, filtered by their quality or appropriateness and finally classified. However, creating and updating such categories is quite time consuming and also possibly labour intensive.

The next step after discovering appropriate web sources and constructing wrappers for them is to be able to integrate data objects extracted from multiple sources, so as to provide a uniform interface for users to review or query the integrated data. Previous research on information integration ([5] and [9]) assumed either that each web site cooperatively provides the schema information or that the user can specify the relational schema for each web site. However, it is not guaranteed either that cooperation from web sites is always provided to the integration system or that every user is a database expert and able to specify the schemas.

Optimistically, we would like to integrate data from autonomous web sources with little or no human effort. This goal is hard to achieve especially when the contents of many valuable webbases are only accessible through search interfaces. Recently, the problem of automatic interaction with webbases to obtain statistical summaries of the contents has drawn more and more attention. [6] and [7] proposed automatic approaches to summarize and classify hidden-web databases containing text contents. They send topically focused queries to the target webbases and determine the topic coverage of the databases by exploiting the number of matches each query generated. As the pioneering work on automatic content summaries for hidden-web databases, [6] and [7] are fairly simple in the management of automatic interaction between "uncooperative" sources as they only target at text-webbases whose search interface is usually one textbox for keyword query submission.

In order to build a large scale information integration system for the hidden web, the cost of manually locating information sources and manually obtaining source descriptions will not be acceptable. Therefore the following problems are worth studying: automatically locating webbases relevant to a given area; automatically determining the topic or domain of a given webbase; automatically determining the appropriateness or quality of information from a given webbase; automatically obtaining descriptions about a given webbase's content, such as schema or semantics of the contained data. Without good solutions to these problems, the cost of discovering sources and obtaining source descriptions will become the "bottleneck" in building large-scale information integration systems. Moreover, an efficient method to reconcile inconsistency between data objects from various sources is also critical in building an integration system for web sources that are autonomous and varied.

References

[1] BrightPlanet Corp. "The deep web: surfacing hidden value."

[2] C.H. Chang, and S.C. Lui. "IEPAD: information extraction based on pattern discovery," *Proc. 10th World Wide Web Conf.* 681-688, 2001.

[3] V. Crescenzi, G. Mecca and P. Merialdo. "ROADRUNNER: towards automatic data extraction from large web sites," *Proc. 27th Intl. Conf. on Very Large Data Bases*, 109-118, 2001.

[4] D. Embley, Y. Jiang and Y.K. Ng. "Record-boundary discovery in web documents," *Proc. ACM SIGMOD Conf.*, 467-478, 1999.

[5] D. Florescu, A.Y. Levy, and A.O. Mendelzon. "Database techniques for the world-wide web: a survey," *SIGMOD Record* **27**(3), 59-74, 1998.

[6] P. G. Ipeirotis and L. Gravano. "Distributed search over hidden web: hierarchical database sampling and selection," *Proc. 28th VLDB Conf.*, 2002.

[7] P. G. Ipeirotis and L. Gravano. "Probe, count and classify: categorizing hidden-web databases," *Proc. ACM SIGMOD Conf.*, 2001.

[8] S. Raghavan, and H. Garcia-Molina. "Crawling the hidden web," *Proc. 27th Intl. Conf. on Very Large Data Base*, 129-138, 2001.

[9] S. Raghavan, and H. Garcia-Molina. "Integrating diverse information management systems: a brief survey," In *IEEE Data Engineering Bulletin* **24**(4), 44-52, 2001.

[10] J. Wang and F. Lochovsky, "Data extraction and label assignment for Web databases," *Proc. 12th World Wide Web Conf.*, to appear, 2003.

[11] J. Wang and F. Lochovsky. "Data-rich section extraction from HTML pages," *Proc. 3rd Intl. Conf. on Web Information System Engineering*, 313-322, 2002.

[12] J. Wang and F. Lochovsky. "Wrapper Induction based on Nested Pattern Discovery," *Technical Report HKUST-CS-27-02*, Dept. of Computer Science, Hong Kong U. of Science & Technology, 2002 (submitted for publication).