Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp147-152)

Adaptable Wrapper Generation for Web Page Format Change

YUE-SHAN CHANG
Department of Computer Science and Information Engineering
National Taipei University
151, University Road, Sanhsia, Taipei county,
237 TAIWAN, R.O.C.

Abstract **-** In this paper, we propose an adaptive wrapper generator that can generate adaptable wrapper for adapting networked information sources (NIS) format changes. When NIS's format changed, the adaptable wrapper can start recovery phase to discover the extraction rule of the new format of target NIS. The wrapper can automatically adapt the changes of content tag and accurately extract information. The wrapper is also examined in 6 websites in 3 kind of NIS, and the result shows that the average precision is over 98%. It can conclude that the generated adaptable wrapper can adapt the NIS's format changes and accurately extract information.

Keywords: - Wrapper, Wrapper Generation, World Wide Web, Information retrieval, Adaptable.

1 Introduction

Mediator/Wrapper architecture can be used to integrate multiple, distributed, heterogeneous, and autonomous database or networked information sources(for short NISs)[11]. Such architecture usually offer end users a unified interface for accessing a variety of NISs or databases[10]. Besides it is with the functionalities of mediating between end users and NIS and translating query string from and result to end users. The mediator acts as a coordiator and integrator while the wrapper as a information retriever for querying and retrieving a specific NIS.

Wrapper is with two functions- one is to translate the format of query string sent from end user to the format acceptable by NIS, and send to target NIS; the other is to filter the response from NIS and fetch desired result and send it back to end user[4]. Wrapper generation techniques are classified into three categories. It is manually written wrapper[6], reconfigurable wrapper[3], and automatically generated wrapper [4, 7, 8] respectively.

In general the information on the Web are encapsulated and presented using HTML's tag.Wrapper retrieves desired inforamtion in HTML file by parsing and filtering specific tags. Fig. 1 shows a simple result that from Yahoo search engine and its simplified tag composition.

Kushmerick[1][2] investigated that over 44% web pages changed it's page layout or tag during six months. It is obviously wrapper, in the such case, will fail to retrieve desired information if the wrapper can not adapt to such change. Programmer need on the go to take care whether wrapper work properly or not, and maintain the failed wrappers.

1. Java 程式設計與寫作
設計物理教學輔助 java 程式。
Virtual Physics Laboratory Fu-Kwun Hwang Registered used will be able to get files related to any java applets at this site. Just look for the button near the end of the first java
分類: 網際網路>應用技術>Java
www.phy.ntnu.edu.tw/class/comp96/java/index.html - 2005/05/10 - 36k - 庫存頁面 - 更多此站結果

YAHOO	
Prefix	**Note**
<LI><DIV>	**Start**
<A></A><A><IMG></A></DIV>	**Title**
<DIV><B></B></DIV>	**Description**
<EM></EM>	**URL**
<EM></EM><A></A><A></A>	**End**

simplified tag composition

Implementing a manually written wrapper might take more two days [4]. It obviously wastes man power and time for maintaining system. Furthermore it is unknown when the target page change it's layout or tag. In this paper we will propose a wrapper generator for generating an adaptable wrapper that can adapt web page change. When NIS's format changed, the adaptable wrapper can start recovery phase to discover the extraction rule of the new format of target NIS. The wrapper can automatically adapt the changes of content tag and accurately extract information. The research is based on our previous work [4] and extent the architecture of generated wrapper for adapting the change of page format.

The paper is organized as follows. Section 2 describes simply the category of wrapper's error and related works. In Section 3, we examine the design and implementation of adaptive wrapper generator. Section 4 evaluates the performance of the system.

Section 5 presents conclusions.

2. Backgrounds and Related Works

2.1 Wrapper failure

In general wrapper is a major component face target NIS for retrieving desired information. There are some cases will cause failure of wrapper during access target NIS. [5] shows that possible cases that are raised during the interactions between wrapper and target NIS. The cases are as follows:

(1) Query language change: most web pages are accessed by send a query string using HTTP protocol. The query language change means that the parameter of the query string sent to target NIS changed. For example the query string sent to yahoo's search engine for the term "java" is as follow **"http://tw.search.yahoo.com/search?fr=fp-tab-web-t&ei=UTF-8&p=java**", the parameter "**fr=fp-tab-web-t &ei=UTF-8&p=java**" can be changed, so that wrapper can fail to access the result of search engine.

(2) Location of NIS change or backend server crash: in the case the wrapper can't connect properly to backend server. Programmer, of course, need to modify wrapper's code or stop the wrapper.

(3) Content of response change: the kind of change might include adding new or modifying old advertisement in the result page, modifying the user interface, or changing the presentation format of desired information. These cases will cause that wrapper retrieve desired information failure.

The three cases mentioned above we will only concern the third one because the first and second case need assistance of programmer or system administrator no matter modifying the code or stop the running of wrapper.

2.2 Related works

Schema-Guided Wrapper [9] proposed an approach for maintaining the wrapper failure. The system consists of *wrapper generator*, *wrapper engine*, and *wrapper maintainer*. In the system the *wrapper generator* offer a GUI that user can input a HTML document and XML schema for generating extraction rule of the document. The *wrapper engine* retrieves information according to the extraction rule. The *wrapper engine* will notify *wrapper maintainer* to recover extraction rule when the process of retrieving information failure.

[12] proposes a wrapper re-induction approach which uses the extracted data at a given time t to regenerate the wrapper at a later moment $t+s$, where s is small. Nevertheless their approach can only support very simple wrappers that are not expressive enough to deal with most modern web sites.

[13] proposes a approach for maintaining wrappers that is based on collecting some query results during wrapper operation. Then, when the source changes, they are used to generate a set of labeled examples that are then provided as input to a wrapper induction algorithm able to regenerate the wrapper.

3. Adaptable Wrapper

This section will depict the architecture of adaptable wrapper and its components. Then will explain the operation of detection and recovery module and adaptable wrapper generation.

3.1 Adaptable Wrapper Architecture

Fig. 2 shows the architecture of our proposed adaptable wrapper. The wrapper consists of three modules described as follows:

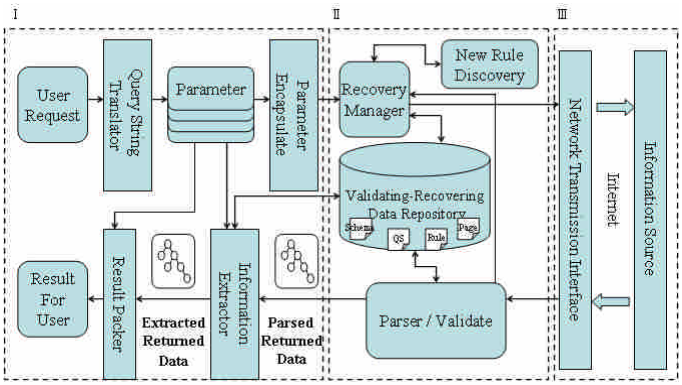

Fig. 2 Architecture of adaptable wrapper

(I) Input/Out Encapsulation module (IOEM): the module is responsible for translating user query into query string of backend NIS and retrieving desired information form NIS response. All retrieved results are packed as XML format for future processing.

(II) Detection and Recovery Module (DRM): the module is mainly parsing and validating the change of web source. When the change detected, the module will raise the recovery phase and re-deduction the extraction rule for extractor.

(III)NIS module (NM)： the module is responsible for connecting to backend NIS.

This work is to extend our previous work [4]. Here will only describe the component of DRM and its operation. The IOEM and NM please refer [4]. As mentioned above, the DRM will raise recovery phase while web page change and retrieve new extraction rule for extractor. It consists of four components that are depicted as follows:

(1) Recovery Manager (RM): RM is a major

component of DRM and is used to control the recovery phase. When the RM receives a recovery event from Parser/Validator, it initial recovery operations that include new extraction rule discovery and update the rule for extractor.

(2) New Rule Discovery component: This component is to deduce new extraction rule from new document using our new rule discovery algorithm shown in section 3.2.2.

(3) Validating-Recovering Data Repository (stand for VRDR): This component stores all recovery necessary data including query string (QS), extraction rule, web page that fetched at wrapper initialized, and schema of target NIS.

(4) Parser/Validator: Web page retrieved from NIS will be parsed to check and verify whether validation or not. If the web page changed, the component will notify the recovery manager to enforce recovery phase; otherwise it will filter most redundant part of web page and send it to IOE module.

3.2 Detection and Recovery Module

Fig. 3 shows the procedure of DRM when the wrapper invalid. First RM save QS into VRDR and then send to NIS. When the returned page changed, the Parser/Validator will notify the RM to enter recovery phase.

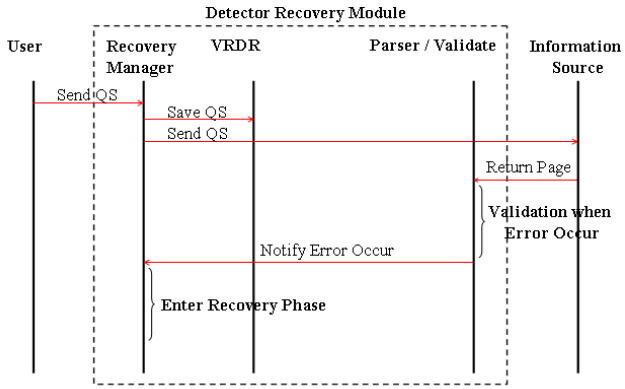

Fig. 3 The procedure of DRM when wrapper invalid

3.2.1 Maximum Similarity Space

In this subsection we define *Maximum Similarity Space (MSS)* for discover the extraction rule of desired information. We use famous search engine-Yahoo as an example. In the example, in general, we assume users just are interested in that information about Title, Description, and URL, as shown in Fig. 4(a).

These repeated information are involved in a sequence of tags, for example * <a> *. In general same information unit have same tag sequence. Wrapper extracts each information unit according to associated tag sequence. Each piece of information unit is named *Similarity Space (SS)* unit, for example *Title Similarity Space (TSS)*. A record in the result page consists of some desired and undesired information. A *MSS* is a repeated record that consists of all desired and undesired information defined as follow:

$$MSS = SS_1 + SS_2 + \ldots + SS_n$$

where SS_i is a SS of desired or undesired information unit. Desired information has a field in the schema that stored in VRDR. An example of MSS in tag are as Fig.4(b)

Fig. 4(a) Web page similarity space

<**div**><**a**><**b**>**TITLE** </**b**>**TITLE** </**a**>
<a>*Undesired part*<img>
Undesired part<a>*Undesired part*
<**div**><**b**>**DESCRIPTION** </**b**> **DESCRIPTION** </**div**>
Undesired part <**b**>**URL**</**b**>**URL**
Undesired part <a>*Undesired part*

Fig. 4(b) An example of MSS in tag

3.2.2 New Rule Discovery Algorithm

When RM is notified the wrapper invalid, it will start new rule discovery algorithm (NRD algorithm) for inferring new extraction rule that can be used for information extractor. Fig. 5 shows the algorithm. The case of wrapper invalidation means that the tag sequence shown in Fig. 4(b) is changed. The NRD algorithm will infer new rule as follow.

(1) First wrapper fetches pre-stored query string and its result pattern from VRDR. The result pattern is used to compare with new fetched web page for finding that the location and order of desired information (Title, Description and URL) in the new page. Obviously the algorithm can also adapt the changed of order of desired information. Here we assume that at least one record in the new page exist in the pre-stored result pattern.

(2) Then wrapper infers all of extraction rules according to the location of desired information by finding that all embrace tags. All HTML tag can be classified into three kind as follow:
   (a) with ending tag: Tag need a ending tag such as <A>, <B>, <Table>….
   (b) without ending tag: Tag need not a ending tag such as
, <HR>, <IMG>….
   (c) optional ending tag: ending tag is optional, such as <P>, <TR>, <TD>….

New Rule Discovery Algorithm
Begin
  if (Error == True)
    fetch data from VRDR;
    fetch new homepage;
    find similarity space of Title, Description, URL;
    compare they sequence;
    return result;
    parse tag;
    find new rule;
    fetch new rule and put into the VRDR;
  else
    return;
  endif
endbegin

Fig. 5 New Rule Discovery Algorithm

If the algorithm matches forward a starting tag that belong to category (a), for example <A>, it will find backward an ending tag </A> and determine the location of ending tag whether crossing the location of starting tag of another information unit or not. If the starting tag is category (b), the tag is directly added to extraction rule. If the starting tag belongs to category (c), the algorithm would find, as category (a), its ending tag. Repeat the algorithm for all pre stored result patterns and look for all of SS of desired information unit and MSS.

(3) Store all new information including extraction rule, query string, and result pattern into VRDR for future use.

3.3 Adaptable Wrapper Generation

3.3.1 Adaptable Wrapper

This subsection will describe the execution flow of adaptable wrapper, as shown in Fig. 6.

First adaptable wrapper translate user query string and encapsulate associated parameter into NIS acceptable query string by a query template that in the IOEM. The translated query string is sent to DRM. DRM first save the query string into VRDR and pass to backend NIS by the NM. Then the wrapper waits for the response from backend NIS and gets the result. The Parser/Validator will verify the return page and filter out most redundant part. If the wrapper is valid, then extracts all of desired information unit and packs the result as XML format for future use and return to users. If the wrapper is invalid, the enter recovery phase to infer new extraction rule, as depicted in section 3.2.

Adaptable Wrapper execution flow
Begin
  translate users query string;
  save query string;
  send the transformed query to information source;
  wait for the return and get the result;
  if (some error occurred)
    enter recovery phase;
  else
    packing result;
    return result;
  endif
endbegin

Fig. 6 Adaptable Wrapper Algorithm

3.3.2 Adaptable Wrapper Generation

The adaptable wrapper can be generated using wrapper generator. The wrapper generation is shown in Fig. 7. The generation flow is similar with our previous work [4] except for adding new components, such as DRM, into component repository. It is shown that can reduce significantly the development and maintenance time.

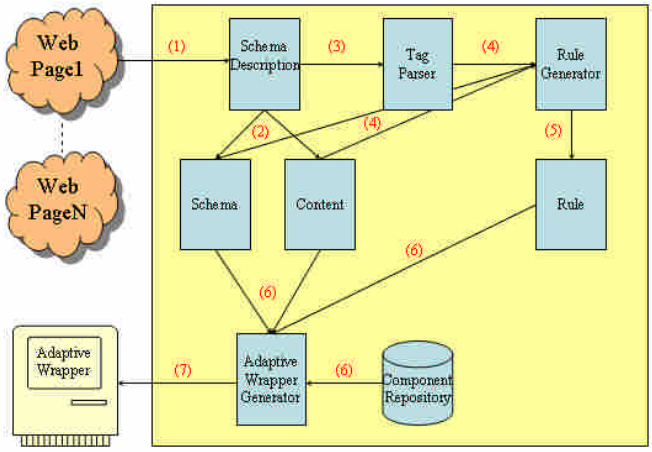

Fig. 7 Adaptable Wrapper Generation Flow

The first step of wrapper generation is to select the target NIS and input its schema of desired information, as step 1 and 2 of Fig. 7. In our example the desired information is Title, Description, and URL respectively. Schema and content in step 2 is stored in a temporary file. Step 3, 4 and 5 will parse the web page, generate extraction rule of the page like as NRD

algorithm, and store the rule into a temporary file. Finally, adaptable wrapper generator generates a wrapper using previous created data and component repository as step 6 and 7.

Fig. 8 shows a Simple UI for Generating Adaptable Wrapper. In this prototype, user input the schema and copy desired content of web page into the UI. And finally press the generator button to generate an adaptable wrapper.

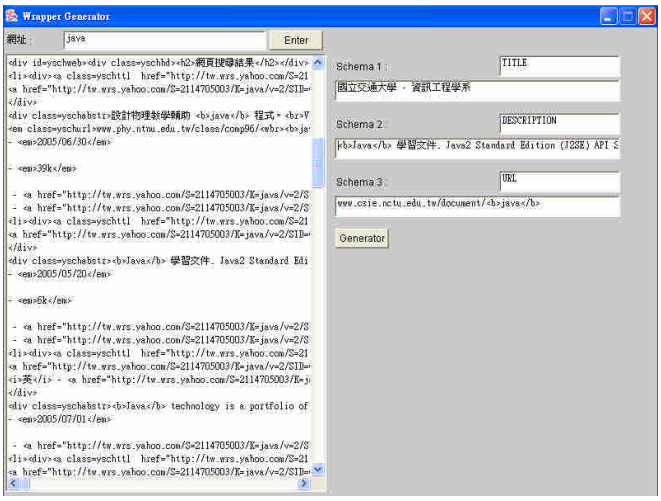

Fig. 8 A Simple UI for Generating Adaptable Wrapper

3.4 Web Page Adaptive Test

Adaptive test of an adaptable wrapper is made using two analogous search engines, Yahoo and Lycos. In this test we generate two adaptable wrappers for Yahoo and Lycos respectively, because these two search engines, according to our observation, have similar result but different format that result in the extraction rule is different.

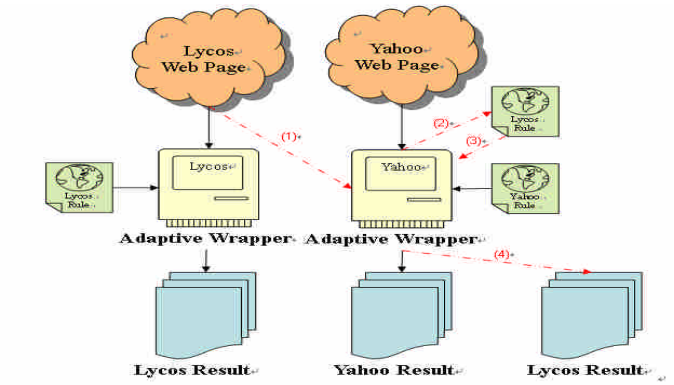

Fig.9 Adaptive test

In the test we first redirect Lycos web page to Yahoo's wrapper, as shown in Fig. 9. That obviously raises the wrapper invalid. The Yahoo's wrapper immediately initiate NRD algorithm for learning Lycos's extraction rule and save extraction rule into VRDR. The wrapper finally output Lycos's result. It shows that the wrapper is adaptive for different web page.

4. System Evaluation

In this section we evaluate the system performance including effectiveness and efficiency. The wrapper generator and generated wrapper are implemented in java language. The metrics of effectiveness include recall and precision [8]. The definition of two metrics is as follow:

$$recall = E_c/N_t$$

$$precision = E_c/E_t$$

where E_c :number of the correct data items extracted by the wrappers; N_t : number of correct data items that should be extracted in a page; E_t :number of data items extracted by the wrappers.

We develop wrappers for three kinds of web sites and six web sites. There are Yahoo, Go, Lycos, Overture, Sina, and CNN respectively. The effectiveness evaluation results are shown in Table 1. Table 1 shows that average recall is over 99% and average precision is over 98%. Although the data size of the evaluation is small, we believe that the result is sameness.

Table 1 Recall & Precision Evaluation

	N_t	E_c	E_t	Recall (%)	Precision (%)
Yahoo	100	99	100	99	99
Go	100	99	100	99	99
Lycos	100	100	100	100	100
Overture	120	120	120	100	100
Sina	100	100	100	100	100
CNN	100	100	110	100	90.9

Table 2 Efficiency Evaluation

	Parsing Time (ms)		Retrieving Time (ms)	
	Round-Trip	System	Round-Trip	System
Yahoo	1322	296	26789	1347
Go	2953	189	47197	2534
Lycos	3147	170	103168	1512
Overture	2926	249	5937	968
Sina	2812	208	37331	1312
CNN	2711	162	30589	1370

In addition, we evaluate the system efficiency. The

evaluation metrics include parsing time and retrieving time. The parsing time is taken when the web page change and the wrapper parse and discover the new extraction rule. The retrieving time is taken when the wrapper retrieve 120 records from backend web sites. Table 2 shows the evaluation. Obviously most time is consumed in the round-trip of network. In the Table 2 the Overture web site has low round-trip time because it can be retrieved 40 records/connection, the other is retrieved 10 records/connection. The average overhead of system is under 10% of all parsing time and retrieving time. It is obviously reasonable.

5. Conclusions

In this paper, we proposed an adaptable wrapper generation to generate wrapper for adapting web page format change. Programmer can reduce the wrapper development time and leave out wrapper maintenance frequently.

The work extends our previous works and adds four components for adapting web page change and retrieving desired information correctly. We also propose extraction rule discovery algorithm to infer the new extraction rule for wrapper adapting the web page change.

Finally we develop wrappers for three kinds of web sites and six web sites. There are Yahoo, Go, Lycos, Overture, Sina, and CNN respectively. The effectiveness evaluation results are shown that average recall is over 99% and average precision is over 98%. In addition, the average overhead of system is under 10% of all parsing time and retrieving time. It is obviously reasonable.

Acknowledgements:
This work was supported by the National Science Council of the Republic of China under Grant No. NSC 92-2213-E-305-003 and No. NSC 94-2752-E-009-006 -PAE. In addition, we would like to thanks Mr. Wu Jung-Chih for his assistance in this project.

Reference:

[1] N. Kushmerick, "Regression testing for wrapper maintenance", In AAAI-99 (Orlando), pages 74–79, 1999.

[2] N. Kushmerick, "Wrapper verification", World Wide Web Journal, 3(2):79–94, 2000. (Special issue on Web Data Management).

[3] Chia-Hui Chang, Jen-Jie Chiou, Harianto Siek, Jiann-Jyh Lu and Chun-Nan Hsu, "Reconfigurable Web Wrapper Agents", IEEE INTELLIGENT SYSTEMS September/October 2003, Vol. 18, No. 5, pp. 34-40.

[4] Yue-Shan Chang, Min-Huang Ho, Wen-Chen Sun, Shyan-Ming Yuan, "Supporting unified interface to wrapper generator in Integrated Information Retrieval", Computer Standards & Interfaces 24 (2002), 291-309.

[5] Zoltán Ádám MANN, "Validating Access to External Information Sources in a Mediator Environment (Technical Report)", 2001.

[6] Hongzhi Wang, Jianzhong Li, Zhenying He, "An Effective Wrapper Architecture to Heterogeneous Data Source", Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on 27-29 March 2003, Page(s):565 – 568.

[7] Alberto Pan, Juan Raposo, Manuel Álvarez, Justo Hidalgo, Ángel Viña, "Semi-Automatic Wrapper Generation for Commercial Web Sources", Engineering Information Systems in the Internet Context 2002, pp.265-283.

[8] Hongkun Zhao, Weiyi Meng, Zonghuan Wu, Vijay Raghavan, Clement Yu, "Fully automatic wrapper generation for search engines", WWW 2005,66-75.

[9] Xiaofeng Meng, Dongdong Hu, Haiyan Wang, Chen Li, "Schema-Guided Wrapper Maintenance for Web-Data Extraction", WIDM 2003, 1-8.

[10] Yue-Shan Chang, Ming-Huang Ho, Shyan-Ming Yuan, "A Unified Interface for Integrating Information Retrieval," Computer Standards and Interfaces, Vol. 23, No.4, Sept. 2001, pp.325-340.

[11] Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock, " Retrieving and Integrating Data from Multiple Information Sources," *Int'l J. Intelligent and Cooperative Information Systems*, vol. 2, no.2, pp127-158, 1993.

[12] Mohapatra R., Rajaraman., K., Sam Yuan, S. Efficient Wrapper Reinduction from Dynamic Web Sources. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence. 2004.

[13] Raposo J.; Pan, A., Alvarez, M., Hidalgo, J., "Automatically maintaining wrappers for Web sources**, "**9th International Database Engineering and Application Symposium, 2005. 25-27 July 2005 Page(s):105 – 114.