Information Extraction from Semi-Structured WEB Page Based on DOM Tree and Its Application in Scientific Literature Statistical Analysis System

Li WeiDong, Dong Yibing, Wang RuiJiang
School of Information Technology
Hebei University of Economics and Business
Shijiazhuang, China
E-mail: lwdbox@126.com

Tian HongXia
School of Computer Applications
Shijiazhuang Information Engineering Vocational College
Shijiazhuang, China
E-mail: yuehun06@yahoo.cn

Abstract—**To extract information automatically from semi-structured web pages, this paper puts forward a method named IESS for discovering the record model based on DOM and Maximal Similar Sub Tree, to identify records automatically and correctly when there are some differences in expression models of records that belong to the same type. To test the performance of the method, a scientific literature statistical analysis system is designed. The practice shows that users can quickly understand the distribution of papers in their retrieving field and grasp the importance with the help of the system.**

Keywords- ***DOM; Automatic information extraction; Scientific Literature; Statistical Analysis***

I. INTRODUCTION

With the development of Internet, people can easily acquire any information they want. The structures of information on the web are quite different from each other, forming a large heterogeneous database environment. These data are extremely complex, have no fixed template or layout. Nevertheless, these data all come from the underlying database, generating the pages conforming to certain regularities. These pages have certain levels and structures, so they are usually called semi-structured data.

To automatically identify and extract information from the semi-structured web sources is an important research topic. In recent years, a large number of researches have addressed the problem, and many important research results have been put forward [1]. These results can be summarized in the following categories:

- Based on a specialized model description language, as well as the corresponding human-machine interface, the model of information identification and extraction from HTML web pages are given by the users [2]. However, this kind of algorithm has inherent dependence on the data sources, so it will become invalidated when these sources or applications have changed.
- Based on a certain regulation description language, as well as the artificial marked training samples. Through a variety of inductive learning methods, the model of information identification and extraction from HTML web pages can be automatically acquired [3, 5]. But such algorithm usually requires the provision of multiple similar input pages and various assumptions about the template of the pages.
- Making use of the implicit regularities in the template, through the method of self-induction, accessing the knowledge of the model of information identification and extraction from web pages [4]. Among the three methods, the first two ones both need large number of users' participation (offering extraction regulations and marked training samples), resulting that the operability and maintainability of the methods are not good enough.

Among the three methods, the first two both need users' participation, so their operability and maintainability are not good enough. While in the third one, the model of web pages can be acquired without any users' participation, so it has a wide practical application prospect and has become the mainstream of research on methods of information extraction from semi-structured web pages.

In this paper, a new information extraction algorithm is presented. It uses DOM tree of the pages to locate the target data region and then acquire the data records.

II. PROBLEM DESCRIPTION

In today's web, information always comes from the underlying database, generating pages according to a certain model. Each page contains many data records. A semi-structured page includes many noises. Only through information extraction, turning the sources into local data, can people make further analysis.

Semi-structured web pages can be represented as DOM trees, then data tend to be assembled in certain levels of the trees, and the target data region will be separated from noises.

After extracting the data records that meet certain conditions from the underlying database, on the basis of the template, conforming to certain format, adding the structure information of HTML to both the front and the back of the attributes. And then putting the results into one region, the structure of HTML code will constitute a DOM tree. In HTML, the type of these attributes' value is always TEXT. For a program, the template becomes the noise, so it is necessary to get rid of the interference from these tags around each attribute node.

Taking the paper searching page of Engineering Village's website as an example.

The 26th record contains only 304 characters, but its corresponding HTML code consists of 1421 characters. There is too much information about the format. Figure 2 shows the DOM tree for the HTML code. In the tree, the six branches of the subtrees belonging to *TD* separately represent different attribute of a paper: *Title, Author, source, Database, Abstract,* and *Detailed*.

Using mathematical language, the algorithm can be described as follows: For a data record R, there exists a template T making $T(R)$ belong to the page P. And R is mapped into one or a set of consecutive sibling subtrees in the DOM tree of the page.

During the process of mapping, in the data record R, a field may be divided into n parts, $C_1(K_1), C_2(K_2), \ldots\ldots, C_n(K_n)$, then the node becomes a root and K which has at least n branches becomes one of the subtrees belonging to R.

Through a large number of observations, it can be found that there are two properties of the representation of the data records in HTML as DOM trees:

Property 1. Each record in the DOM tree is disposed in one or a set of consecutive sibling subtrees, each subtree only belongs to a single record.

Property 2. Each attribute in several records have the same path from the root in the DOM tree.

Data extraction algorithm is to find the template T that maps data record R to a web page, then put the data contained in HTML code into the attributes of the data records, and finally store the data into database to do further processing.

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. IESS ALGORITHM

IESS algorithm mainly consists of three steps:

- Locating the data region of the page, ignoring the noise in other regions.
- Dividing the data region into records.
- Extracting the attributes of the data records.

A. Locating the data region of the page

As shown in Figure 2, finding the data region is equivalent to finding the common parent node of the sibling subtrees forming the data records. According to the properties of the previous section, our method for finding the data region in a page consists of the following steps:

1) Consider N as a set composed by all the nodes in the DOM tree. To each node $N_i \in N$, we will assign a score named B_i, initially $B_i = 0$.

2) Compute T, the set of all the text nodes in N, namely, $T \in N$.

3) Divide T into subsets $P_1, P_2, P_3 \cdots P_k$, in a way such that all the text nodes with the same path from the root in the DOM tree are contained in the same P_i.

4) For each pair of text nodes belonging to $P_{i,1\le i\le k}$, compute N_j as their deepest common ancestor in the DOM tree, and add 1 to B_j (the score of N_j).

5) Let N_{root} be the node having the highest score $B_{\max}$. Choose the DOM subtree having $B_{\max}$ as root of the desired data region.

Then a DOM subtree model of a data region is newly created, having N_{root} as its root.

B. Dividing the data region into records

By Property 1, we can see that record may be composed of several consecutive sibling subtrees, so there must be ways to group the subtrees so that the subtrees with the maximal similarity are clustered in one group. By Property 2, we can see that the same attribute in different records have the same path from the root in the DOM tree.

To compute the similarity between two sequences of consecutive sibling subtrees named T_i and T_j in the DOM tree of a page, we perform the following steps:

1) We represent T_i and T_j as strings (we will term them TS_i and TS_j). We traverse each subtree in depth first order and, for each node, we generate a character in the string. A different character will be assigned to each tag having a different path from the root in the DOM tree. The number of the tags is named len(TS_i). For instance, TS_0 is <TBODY><TR><TD><MedBlackText><Text><TD>SpLink<Text><TD><SmBlackText><Text><TD><SpLink><Text><TD><SmBlackText><B><Text><SmBlackText><I><Text><TD>< SmBlackText ><B><Text>< SmBlackText ><Text><LgBlueLink><Text><TD><LgBlueLink><Text>.

2) We compute the similarity between T_i and T_j.

$$Simi(T_i, T_j) = 1 - \frac{|len(T_i) - len(T_j)|}{len(T_i) + len(T_j)} \quad (1)$$

3) If T_i and T_j belong to the same group, we calculate (len(TS_i) + len(TS_j)), and then we repeat the steps above.

Algorithm to divide the data region into records consists of the following stages:

- We represent L_{ij} as leaf strings that stand for the alignment of the path from N_{root} to a Text node belong to a subtree which is a direct child of N_{root}, where, *i* orders the subtree, and *j* for the leaf. For example, as shown in Figure 1, there exists L_{11} = <body><Center><Table><TBody><TR><TD><DIV><FORM><Table><TBody><TR><TD><A.MedBlackText><B>.

- By property 1, we can see that T_1 belong to the first record of the data region. By property 2, we can infer that, if T_i is the first subtree of another data record, it must have the same structure as T_1. Find the matching nodes from T_1 and T_i, and consider the number of the nodes as the similarity measure between T_1 and T_i named S_i, initially $S_i = 0$. Compare L_{1j} with $L_{ij}, i > 1$, if $L_{1j} == L_{ij}, i > 1$, then $S_i = S_i + 1$. Repeat this step until the end.
- Sort the set of subtrees according to the value of S_i. When some of them share the same value, chose the ascending order according to i.
- In the ordered set of T_i, if T_i is the first in the set, clustering $(i-1)$ subtrees as one group and so on (e.g. T_4 ordered first, then 3 subtrees constitute the first group). If the rest of the subtrees can not compose one group, they may be noises that should be ignored. Finally, all subtrees are divided into n groups.
- Repeat the process of computing $Simi(T_i, T_j)$, $1 \le i \le n, 1 \le j \le n, i \ne j$ with different grouping methods, and chose the method with the smallest standard deviation as the best record division method. Then, this group represents a complete data record.

C. *Extracting the attributes of the data records*

Record fields on the page always have remarkable signs that can be used to distinguish different fields, as shown in Figure 1: source, database, abstract, detailed, and such as job recruitment online: job location, telephone, contact, and etc.

By leaf string generation method, align the leaf string of the first data record, and then traverse the path from root to leaf. Once meet with the TEXT tag, get its value (with special attention to deal with punctuations and spaces). Then compare the acquired value with that at the same location in the other records, if they are equal, this is just the segmentation point of the record, and the same characters can be used as field names. Remove the HTML code between segmentation points, and then filter the results as the field values with field analysis application.

For instance, the leaf strings contain Source at the same position, set Source as a field in the database, and the elements between Source and Database as its value:

……<TR><TD> **Source :**<B>……**Database:**……
……<TR><TD> **Source :**<B>……**Database:**……
……<TR><TD> **Source :**<B>……**Database:**……
……<TR><TD> **Source :**<B>……**Database:**……

This is the end of the entire algorithm.

IV. APPLICATION S

Every time we use a literature retrieval system such as SCI, EI or ISTP to search scientific papers, the system always returns us a result with massive data including a lot of useless information. Only through long-term analysis can we acquire the effective information. To solve this problem, we have designed a scientific literature statistical analysis system, where IESS algorithm is used.

A. *System design*

The key of the system is to turn the web page into structured data, and store these data into the database. On the basis of the database, we can carry out further statistical analysis. The structure of the system is shown in Figure 3.

The whole process is considered as a project. Project management module is in charge of managing information about the objective, time, query, and key words. Literature retrieval module is the entrance where WebBrowser has been used to create a multiwindow browser. When we have acquired the satisfying result, we can download the pages with web page download module. After downloading, information processing module will take over the next task, where IESS algorithm is implemented. Database is the kernel of the system, where all relevant information and data are stored. It is the foundation of the statistical analysis.

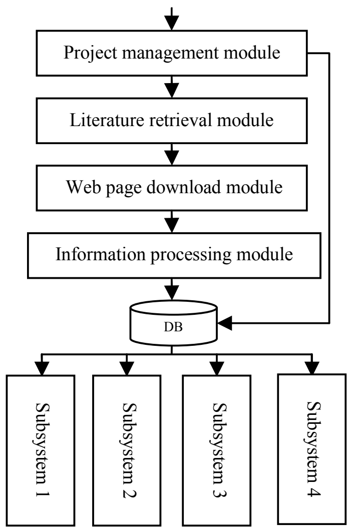

Project management module
Literature retrieval module
Web page download module
Information processing module
DB
Subsystem 1
Subsystem 2
Subsystem 3
Subsystem 4

Figure.3 Structure of the system.

B. *Application of IESS algorithm*

We divide the web pages returned from retrieving into two layers. After downloading, we have acquired multiple pages in the first layer, but the specific information of each paper is saved in the second layer. So in the information processing module, there are two central tasks: downloading the page in the second layer and resolving the semi-structured web pages into structured fields, then storing these fields into database.

There are some basic information of the paper such as title, source, and author in the first layer pages. Under this basic information, there are two buttons respectively named Abstract and Detailed. The hyperlink under Detailed button provides accessing to the second layer pages. In the same way, by calling GetHttpStream, we can download the page in the second layer, where information all belongs to one paper. Its structure is so clear that it is quite easy to extract information. So we will pay more attention to information extraction from the first layer pages. Here we have chosen IESS algorithm to achieve the task.

Through testing, we find that this system can greatly improve the utilization efficiency of information.

V. CONCLUSIONS

In this paper, we have presented a new method to automatically detecting structured records in a web page and extract the data fields. Our method will have a wide application prospect in many different domains such as literature analysis, e-commerce, job hunting, web spiders, etc. We have also validated our method through a designed scientific literature statistical analysis system, obtaining good effectiveness. With our method, it is easy to do a variety of statistical analysis. It will greatly improve the utilization efficiency and depth of the Internet, leading to a qualitative leap of the scientific research tracking ability.

ACKNOWLEDGMENT

No one has enough good ideas for such a paper. So thanks belong to all the people that ever helped us.

REFERENCES

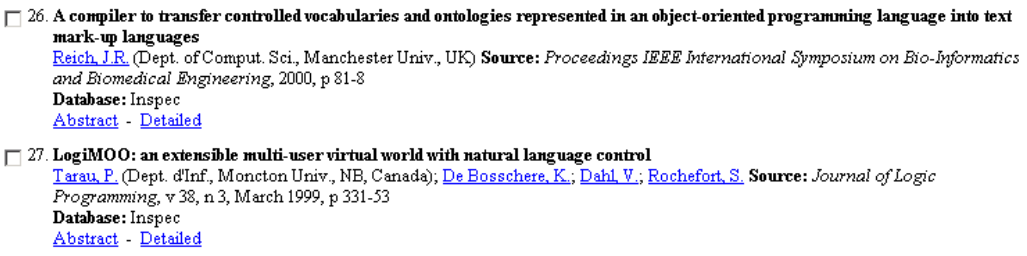
[1] Alberto H F Laender, BerthierA R ibeiro - Neto, Altigran S daSilva, Juliana S Teixeira . A brief survey of Web data extraction tools[J]. ACM SIGMOD Record, June 2002, 31 (2) : 84 - 93.

[2] A. Pan et al., Semi-automatic wrapper generation for commercial web sources, Proceedings of IFIP WG8.1 Conference on Engineering Inform, Systems in the Internet Context (EISIC), 2002: 265 - 283.

[3] A. Arasu, H. Garcia-Molina, Extracting structured data from web pages, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2003:491-509.

[4] Arvind Arasu . Hector Garcia - Molina . Extracting Structured Data from Web Pages[C]. Proceedings of the 2003 ACM SIGMOD international conference on on Management of data, June 2003: 337 - 348.

[5] ZHANG Cheng, CHEN Zi-yu, GU Ping, YANG Rui-long. Automatic Blog recognition with DOM tree. Application Research of Computers, 2008（05） : 1489-1491.


26. **A compiler to transfer controlled vocabularies and ontologies represented in an object-oriented programming language into text mark-up languages**
Reich, J.R. (Dept. of Comput. Sci., Manchester Univ., UK) **Source:** *Proceedings IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, 2000, p 81-8
Database: Inspec
Abstract - Detailed

27. **LogiMOO: an extensible multi-user virtual world with natural language control**
Tarau, P. (Dept. d'Inf., Moncton Univ., NB, Canada); De Bosschere, K.; Dahl, V.; Rochefort, S. **Source:** *Journal of Logic Programming*, v 38, n 3, March 1999, p 331-53
Database: Inspec
Abstract - Detailed

Figure.1 Part of the paper searching page of Engineering Village's website.

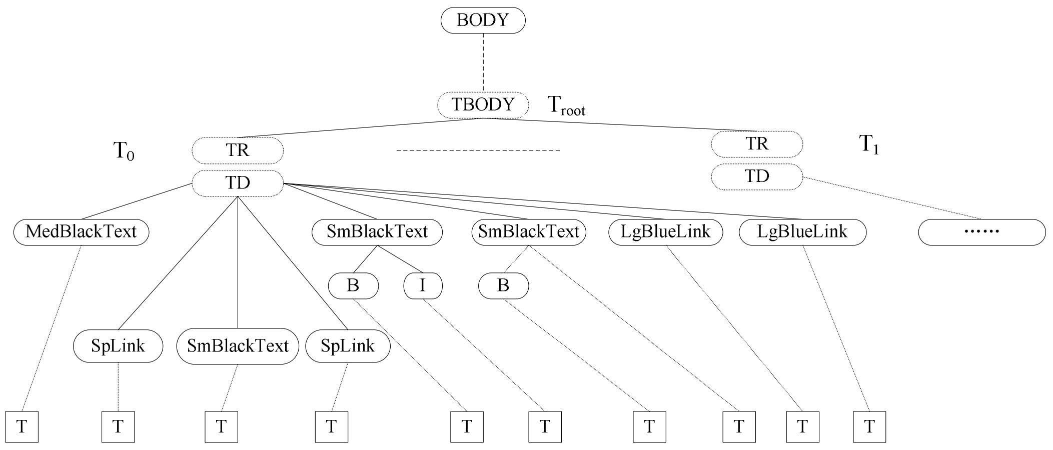


Figure.2 DOM tree for the HTML code shown in Figure. 1