Fourth International Conference on Semantics, Knowledge and Grid

Extracting Multi-Records from Web Pages

Tian Xia [1,2]

[1]Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), MOE, Beijing 100872, China

[2]School of Information Resource Management, Renmin University of China. Beijing 100872, China

xiat@ruc.edu.cn

Abstract— **Extracting multi-records from web pages is useful, it allows us to integrate information from multiple sources to provide value-added services. Existing techniques still have some limitations because of their several restrictions and accuracy. This paper proposes a new method to perform multi-records extraction task automatically. Firstly, the HTML tag tree is build based on an embedded browser interface to solve the AJAX problem. Secondly, data regions are found out by data chunk comparison, and simple tree matching method is proposed to compute the chunk similarity. Finally, the main data region is determined and the multi-records are extracted out. Experimental results show that our method dramatically outperforms other existing methods, and it can extract multi-records from pages very accurately.**

I. Introduction

Structural data objects are a very common type of information on the web. A list of these objects in a Web page often describes a list of similar items from underlying databases, e.g. a list of products or visitors' comments. In this paper, we call them multi-records. It is useful to extracting records from multiple sources in order to provide value-added services. Compared with general data extraction from Web pages, multi-records have some specific characteristics and can be exploited for automatic extraction.

Several approaches have been presented for extracting data records such as manual approach and wrapper approach[1, 2]. Most extraction methods are based on the HTML source code, in the Web 2.0 era, AJAX (Asynchronous JavaScript and XML[3]) has gained a prominent position, which is used to achieve high interactivity with asynchronous server communication. In an AJAX application, much of the page content isn't contained in the HTML source code, but is dynamically inserted by JavaScript during page load. Furthermore, CSS is used more and more widely, even plain HTML tables are often not any longer realized as table structure, but rendered as table by style sheets. These changes bring new challenges to the data records extraction.

In this paper, we propose a new method to extract multi-records from web pages without training process, and use embedded browser interface to conquer the lazy loading problem of AJAX pages. Then, we compare adjacent data chunks and improve the simple tree matching algorithm to measure the chunk similarity. After the main data region is determined by the nodes number and the region length, multi-records can be extracted out according to the domain characteristics easily.

II. Building the HTML Tag Tree

Any HTML document can be converted into an HTML DOM Tree by using its hierarchical tag structure. For AJAX Web pages, however, the content displayed in the Web browser is inconsistent with its original source code, and most import data regions appeared in the browser are dynamically inserted by JavaScript executed at the client side.

To solve this problem, our approach is based on an embedded browser interface capable of executing JavaScript and the supporting technologies required by AJAX. The embedded browser interface has two implementations: IE-based on Watij[4] and Mozilla-based on XULRunner[5].

Once the dynamic inner DOM object are fetched after page load, we then convert it into our own tree structure for convenience, the main attributes in each node include name, value, type, depth and the pointers of the parent and children. In particular, CSS has been used widely today, and the nodes with the same tag name can have various display results if they have the different class attribute. When building the new tag tree, we take the original tag name plus the value of class attribute as the new node name. For example, the node name of the HTML fragment <DIV class="T"/> is "DIV-T".

III. Mining Data Regions

Each record must be represented by a set of tag nodes, we call them data chunk. A valid data chunk should contain the corresponded data record surrounded by HTML tags, and a sequence of valid data chunks forms a data region.

Definition: A data chunk of length r consists of r sub-trees in the HTML tag tree with the following properties:

- The sub-trees all have the same parent node;
- The root nodes of the sub-trees are adjacent.
- The length of a data chunk C is defined as |C|, which is the number of the sub-trees that C contains.

Unlike the definition of special generalized node proposed by Bin Liu[6], data chunk is composed by several adjacent sub-trees, and it contains all relevant information for the final record extraction.

According to the paper [6], the data region is defined as follows:

Definition: A data region is a collection of two or more data chunks with the following properties:

- The data chunks all have the same parent node;
- The data chunks all have the same length;
- The data chunks are all adjacent;
- The similarity between adjacent data chunks is greater than a fixed threshold T.
- The length of region R is defined as $|R|$ which is the number of the data chunks that R contains.

978-0-7695-3401-5/08 $25.00 © 2008 IEEE
DOI 10.1109/SKG.2008.47

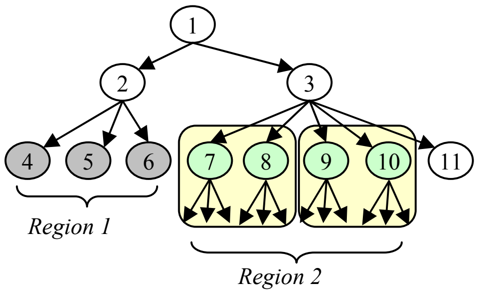

Fig. 1 An illustration of data chunks and data regions

For example, in figure 1, we find that the nodes 4, 5 and 6 are similar and form the data region labelled 1, each node of 4, 5 or 6 is a data chunk; the pairs of nodes (7, 8), (9, 10) are similar and form the data region labelled 2, data region 2 has two data chunks, the first data chunk has two sub-trees rooted by nodes 7 and 8, and the second chunk also has two sub-trees rooted by nodes 9 and 10. Furthermore, region 2 has more nodes than region 1, so it is the main region appeared in this artificial tag tree.

A. Comparing data chunks

Let $C_L(p, i)$ is the i^{th} chunk with length L under the parent node p, then $C_L(p, i)$ can be represented as follows:

$$C_L(p,i)=\{t(p,(i-1)\times L+1),...,t(p,i\times L)\}$$

Where $t(p, j)$ is the j^{th} sub-tree which belongs to the parent node p, and the data chunk i contains L sub-trees from $(i-1)\times L+1$ to $i\times L$.

Given a specified parent node p and the length r of the current data chunk, we can build all candidate data chunks staring from a special node, and put them into a queue sequentially. And then, the possible data regions can be found by comparing neighbour data chunks.

Algorithm FindCandidateRegions(*p*, *r*, *T*):

```
1   for start=1 to r:
2       queue = FindCandidateChunks(p, r, start);
3       chunk1 = queue.poll(), chunk2 = Null;
4       tempDR = {chunk1};
5       while queue.size() > 0:
6           chunk2 = queue.poll( );
7           if SIM (chunk1, chunk2) > T:
8               tempDR = tempDR ∪ {chunk2};
9           else if LEN(tempDR) > $\lambda$:
10              AddCandidateRegion(tempDR);
11              tempDRs = {chunk2};
12          chunk1 = chunk2;
```

Fig. 2 Algorithm for finding regions

In the above algorithm, T is the threshold for similar data chunks judgment. The function FindCandidateChunks finds all the data chunks of p with length r starting from the *start-th* node.

The overall algorithm (EMR) for extracting multi-records is given in figure 3. It traverses the tag tree from the root node recursively. The parameter *node* is the current node to do candidate regions identification, K is the maximal possible chunk length, and T is the threshold as mentioned above.

Algorithm EMR(*node*, *K*, *T*):

```
1   if node.depth() > 2:
2       for r=1 to K do:
3           FindCandidateRegions(node, r, T);
4   for each child ∈ node.children():
5       EMR(child, K, T);
```

Fig. 3 The overall EMR algorithm

Let N be the total number of nodes in the tag tree, the complexity of EMR is $O(N\bullet K)$ without considering chunk similarity computation.

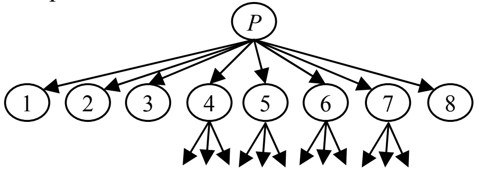

Fig. 4 Combination and comparison

We use figure 4 to illustrate the comparison process. Figure 4 has 8 nodes, which are below a parent node, p. Let K=3 in this example, when the length of the current data chunk is r=1, then p can be divided into 8 data chunks, and we compute the following data chunk comparisons:

(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8)

When r=2, according to *FindCandidateRegions* algorithm, we compare the following chunk pairs:

(1-2, 3-4), (3-4, 5-6), (5-6, 7-8)

(1-2, 3-4), (3-4, 5-6), (5-6, 7-8)

When r=3, we compare:

(1-2-3, 4-5-6)

(2-3-4, 5-6-7)

(3-4-5, 6-7-8)

B. Computing similarity between data chunks

Bin Liu uses normalized string edit distance algorithm to measure the similarity of generalized nodes[6]. However, this approach can not identify the data region correctly in some cases. For example, in figure 5, the real region is composed of two data chunks as indicated by the dashed circles, but the tag string edit similarity between two solid circles is still very high, and so, (H-A-B, A-B-T) is wrongly selected as the final data regions in this sample.

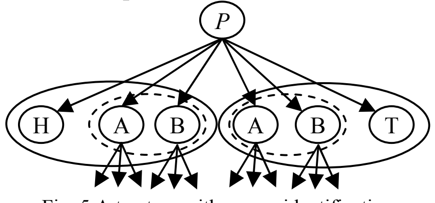

Fig. 5 A tag tree with wrong identification

Another approach to compute the similarity is based on tree matching[7, 8]. However, previous tree-based methods assume that all HTML tags have equal weights which can not reflect the characteristics of HTML features. Moreover, data chunks are composed of sub-trees, so we can not use tree matching directly.

Similar to string edit distance, the tree edit distance algorithm aims at finding the minimal operational cost for transforming a tree into another tree[9]. It consists of three operations: node deletion, node insertion, and node replacement, each of which is assigned an operational cost. In this algorithm, the tree mapping is defined as follows:

Let $T[i]$ be the i-th node of a tree T in a preorder walk of the tree. The size of a tree is defined as the number of nodes in the tree. Then, a tree mapping from T_1 with size n_1 to T_2 with size n_2 is a set M of tuples(i, j) that satisfies the following conditions:

For all $(i_1, j_1), (i_2, j_2) \in$ M,

(1) $i_1 = j_1$ *iff* $i_2 = j_2$

(2) $T_1[i_1]$ is on the left of $T_1[i_2]$ *iff* $T_2[j_1]$ is on the left of $T_2[j_2]$

(3) $T_1[i_1]$ is an ancestor of $T_1[i_2]$ *iff* $T_2[j_1]$ is an ancestor of $T_2[j_2]$

The above definition requires that each node can appear no more than once in a mapping, and the order between sibling nodes and the hierarchical relation between nodes are both preserved.

In general, tree mapping is a difficult problem and the algorithm for tree edit distance measure has high complexity. Our work is based on a restricted matching algorithm called simple tree matching (STM). STM evaluates the similarity between two trees by measuring the maximum tree mapping value through dynamic programming strategy with complexity $O(n_1 \bullet n_2)$, where n_1 and n_2 are the size of two trees respectively.

When multi-records contain long descriptive items, such as the descriptions of products, their inner structures may be quite different, and then, the real data chunks can not be identified correctly. Considering the different tag positions appeared in data chunks, we assign a weight to each tag node by the following formula:

$$w(n_i) = \frac{\alpha}{Depth(n_i) + \alpha}$$

Where n_i is the *i-th* node of the data chunk, $Depth(n_i)$ represents the distance that from current n_i to the corresponded sub-tree's root node, α is an adjustable factor, the bigger value of α means the little influence among different depth of nodes. Then, we give our modified STM algorithm as follows:

```
Algorithm ModifiedSTM( $T_A$, $T_B$ ):
1   if the roots of $T_A$ and $T_B$ contain distinct symbols:
2       return 0;
3   else:
4       $m$ = the number of sub-trees of $T_A$;
5       $n$ = the number of sub-trees of $T_B$;
6       Initialization:
              $M[i, 0] = 0$ for $i = 0,\dots,m$;
              $M[0, j] = 0$ for $j = 0,\dots,n$;
7       for $i$ = 0 to $m$:
8           for $j$ = 0 to $n$:
9               $M[i,j]$ = MAX($M[i,j-1]$, $M[i-1,j]$,
    $M[i-1,j-1]$ + ModifiedSTM($T_A[i]$, $T_B[j]$));
10      return $M[m, n]$ + AVG($w(T_A)$, $w(T_B)$)
```

Fig. 6 Modified STM algorithm

The main skeleton of ModifiedSTM is similar to STM. The only difference is the way of incrementing the value of node mapping at line 10.

Given two data chunks $C_1=\{t_{11}, t_{12}, \dots, t_{1n}\}$, and $C_2=\{t_{21}, t_{22}, \dots, t_{2n}\}$, $|C_1| = |C_2|$, the chunk mapping can be calculated as follows:

$$CM(C_1, C_2) = \sum_{i=1}^{n} ModifiedSTM(t_{1i}, t_{2i})$$

More specifically, the similarity between two data chunks is obtained by the following equation:

$$SIM(C_1, C_2) = \frac{CM(C_1, C_2)}{AVG(W_{all}(C_1), W_{all}(C_2))}$$

Here, $W_{all}(C_1)$ denotes the sum of node weights for all nodes in the data chunk C_1.

C. Determining the main data region

The EMR algorithm extracts more than one candidate data regions. In most cases, we only need to find the most important region out, and we call it the main data region. To determine the main data region, we sort the candidate regions by their importance, and then take the top one as the main data region. The rules to judge the regions' importance are listed below:

(1) The number of nodes that a data region contains. This is the most important factor for regions sorting. For instance, the length of a region which contains navigation links may be greater than the length of the final data region, but its nodes is usually less than the main region.

(2) The length of the data region. If two data regions have the same node's number, we choose the region with greater length as the important one. So, both region (a, a, a, a) and region (a-a, a-a) have the same coverage, but the first region is more important than the second.

IV. Extracting Multi-Records

After we obtain the final main data region, extracting multi-records from this region is relatively easy because the final records are limited to a very small tag set. Furthermore, for most data extraction task, the objective is very clear in advance, and the specific characteristics of the domain of interest can be taken into account to extract Web data in a precise manner. In some cases, a data chunk may contain two or more actual data records, this repetition problem can be solved by semantic items identification.

In our experiments, we extract the visitors' comments of hot news. The data items include IP address, date time and the comment detail. IP address and date times are extracted by regular expression pattern matching. The rest of the data chunk should contain the comment detail, and it is extracted by adjacent text node merging.

V. Experimental Results

In paper[6], the author has evaluated the MDR algorithm and compared it with several state-of-the art systems, so, we only compare our method with MDR. Since MDR can not process AJAX web pages, such as the comment page of sina.com, we first build the HTML tag tree by the method of section 2, and then execute MDR algorithm. The main data region is also identified by section *determining the main data region* for MDR algorithm.

The test web pages are selected from 4 portals: www.sina.com.cn, www.sohu.com, www.qq.com, and cn.yahoo.com, 10 pages are selected for each site, and the extraction purpose is to find out the visitors' comments from portals. The EMR algorithm and the MDR algorithm are applied to theses pages with 11 different similarity threshold value, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85 and 0.9. For EMR, the experimental settings are given below:

(1) K=6 (The maximal possible chunk's length)

(2) λ=2 (The minimal candidate region's length)

(3) α=5 (The adjustable factor for computing node weight)

The experimental results are illustrated in figure 7. Since the entire test pages are comment pages, and each page can generate a main data region, so the recall is 100%, and we only use the precision measure here.

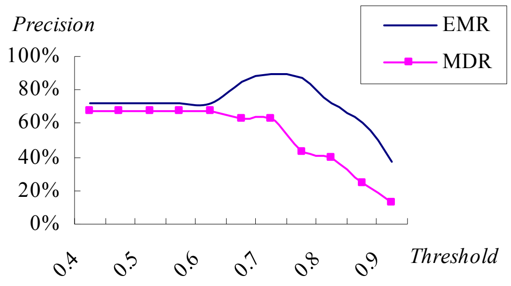

Fig. 7 Experimental results

As shown in figure 7, we can see that, the EMR outperforms the STM for all threshold values. Particularly, raising the threshold does not increase the precision for MDR algorithm. It means that we must use a low threshold value to achieve high precision, such as 0.3 in paper [6]. EMR algorithm uses a new similarity measure, it achieves better results, and the best result in experiments is 0.9 when threshold is 0.7.

VI. Related Work

The existing methods about web data extraction are mainly based on analyzing HTML structural trees, and the main direction is automatic wrapper induction and extraction. Recently, Wei Liu et al. perform the extraction task using only the visual information of the response pages[10], this work is based on VIPS[11], VIPS is proposed by Deng Cai in 2003, it is suitable for block identification of traditional Web page, but some rules (e.g. font color) can not applied to CSS based pages.

Bing Liu et al. have developed an effective MDR algorithm for mining data records from web pages[6]. The algorithm has two steps. In the first step, it identifies the data region of the Web page; in the second one, it extracts the records themselves. This method uses the normalized string edit distance of tag nodes, and can not process AJAX pages. We have given the comparison with our method as shown in the experimental section. Yanhong Zhai and Bing Liu also propose a DEPTA algorithm for data record extraction and data items alignments and extraction[12], its main contribution is focused on arbitrary data item extraction, for records extraction, it has little significant difference to MDR.

Paper[7] proposes an enhanced tree matching algorithm that improves the tree edit distance by considering the characteristics of THML features, our modified STM algorithm is inspired by this paper.

VII. Conclusions

In this paper, we propose a new method to extract multi-records from web pages. Firstly, we use embedded web browser interface to fetch the entire page content from AJAX web pages, and then mining the candidate data regions which may contain the final multi-records based on chunk comparison. Improved STM algorithm is also proposed to compute the chunk similarity. Finally, we extract the multi-records from the main data region considering the characteristics of domain. Experimental results show that our method outperforms the existing methods significantly. Several research directions to extend and improve this work in the future include:

(1) Identifies the web pages which contain multiple records, and fetches all the paging web pages.

(2) Finds a way to delete the duplicate records.

Acknowledgment

This work has been supported by the open project of Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education.

References

[1] V. Crescenzi, G. Mecca, P. Merialdo, "RoadRunner: towards automatic data extraction from large web sites," in *Proc. VLDB'01*, 2001, p.109.

[2] A. Arasu, G.M. Hector, "Extracting structured data from web pages," in *Proc SIGMOD'03*, 2003, p.337.

[3] J.J. Garrett. (2005) Ajax: A New Approach to Web Applications. [Online]. Available: http://www.adaptivepath.com/ideas/essays/-archives/000385.php

[4] (2008) The WATIJ website. [Online]. Available: http://watij.com/

[5] (2008) The XULRunner website. [Online]. Available: http://developer.mozilla.org/en/docs/XULRunner

[6] B. Liu, R. Grossman, Y. Zhai, "Mining Data Records in Web Pages" in *Proc SIGKDD'03*, 2003, p. 601.

[7] Y. Kim, J. Park, T. Kim, J. Choi, "Web information extraction by HTML tree edit distance matching," in *Proc ICCIT'07*, 2007, p.2455.

[8] D. C. Reis, P. B. Golgher, A. S. Silva, A. F. Laender, "Automatic web news extraction using tree edit distance," in *Proc 13th WWW*, 2004, p.502.

[9] K.C. TAI, "The tree-to-tree correction problem," *Journal of ACM.,* vol. 26, pp. 422–433, Jul. 1979.

[10] W. Liu, X. Meng, W Meng, "Vision-based web data records extraction," in *Proc SIGMOD-WebDB2006*, 2006.

[11] D. Cai, S. Yu, J.R Wen, W.Y. Ma, "VIPS: a vision-based page segmentation algorithm," Microsoft Research, Tech.Rep. MSR-TR-2003-79, 2003.

[12] Y. Zhai, B. Liu, "Web data extraction based on partial tree alignment," in Proc 14th WWW, 2005, p.76.