Detecting and Partitioning of Data Objects in Complex Web Pages

Shiren Ye and Tat-Seng Chua
School of Computing, National University of Singapore, Singapore 117543
{ yesr |chuats }@comp.nus.edu.sg

Abstract

This paper presents an automated approach to detect and partition useful data objects from complex Web pages. First, we derive the common structure of the pages by comparing similar pages from the same Web site. Second, we identify data region covering the descriptions of data objects by removing the irrelevant contents from the Web page. Third, we partition the nodes belonging to different data objects in the data region and construct the well-formatted and self-explainable XML output files, one for each data object. From the resulting output files, it is then easy to extract data to fill a database or form a template for presentation to the users. The experiments indicate that our technique is effective.

1. Introduction

More and more companies manage their business and publish their products and services on the Web. Collecting and organizing these dynamic information could produce the data for various value-added applications. Scenario of such applications include: (a) collating and comparing the prices and features for different products from the Web sites of different companies; and (b) listing all products and services offered by one or a group of companies. To facilitate such applications, we need tools to extract attribute information of each product (called **data object**) within a region (called **data region**) in the product Web page. Typically, a product Web page contains one data region, and each region lists information of one or more data objects.

The characteristics of such web pages from commercial sites are:

1. There are many diverse sites with pages in different formats and styles.
2. There are many irrelevant components intertwine with the descriptions of data objects in Web pages. In order to attract the attention of the users, the designers of Web sites often include many exciting items as shown in Fig. 1 to entice users to browse their pages and stay at their sites for as long as possible. Those items include ads bar, product category, search and filtering panel, navigator bar, copyright statement, feedback form etc.
3. In many Web pages, there are normally more than one data object entwined together in a data region. Furthermore, the raw source of the Web page for depicting the object might be non-contiguous. So it is difficult to discover the attributes for each object if they are enwound by the description of others.
4. The order, number and format of attributes involved in depicting the same data object may vary greatly in different pages from different sites. The available systems might fail when they encounter pages from new sites.

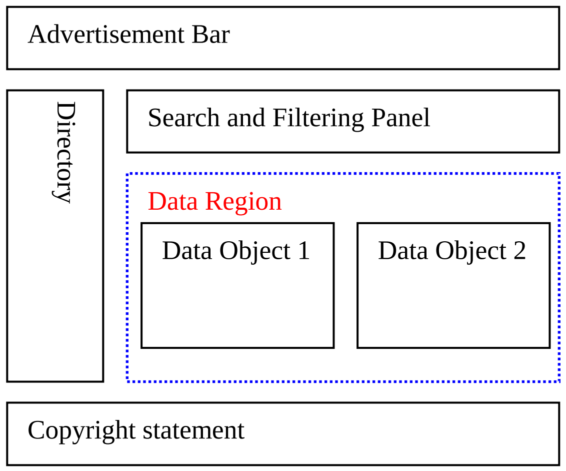

Fig. 1: The layout of a typical product description page

In this paper, we call such Web page as shown in Fig. 1 a complex Web page as it contains multiple types of data components. In real applications, what the users want from complex Web pages is the description of individual data object. It can be derived from the partitioning of data region. The users can then properly process these separate data objects.

Based on the above observations, hand-coded rules or many reported wrapper systems cannot achieve good performance on these complex pages coming from real application sites. Since such pages are generated by the same dynamic program or carefully maintained template, they share similar structure. It is therefore possible to unveil such common structure by analyzing pages form the same site, and use this knowledge as the basis to remove irrelevant contents while retaining relevant object information in data region.

This paper aims to extract and present the details of data objects contained in the Web pages using XML format. The output file, containing the corresponding description of the raw sources without irrelevant components, is well-structured and understandable. It is of high quality for human and software agents to process. The description of an object just appears in one file, and all XML output files have similar structure. So it is easy to transform them into tables or templates.

To achieve our aim of extracting and presenting data objects from complex Web pages, the first crucial problem is to detect the data region that contains the desired objects. Secondly, we need to mine the structure of the data region to extract the structure and description of individual data object.

Briefly, the contents of this paper are organized as follows. Section 2 introduces related work and Section 3 presents the overall procedure for detecting and partition data objects from complex pages. Sections 4 and 5 respectively present the algorithms for detecting data region and partitioning data objects. The results of our experiments and conclusions are respectively presented in Sections 6 and 7.

2. Related work

Extracting and integrating information from the Web has become more and more important recently because it helps to tackle an urgent business problem of collating, comparing and analyzing business information from multiple sites. Related works to our study include Information Extraction (IE), Data Integration and Web page cleaning. Many researchers have focused on such problems. We briefly discuss some of them below.

As a series of widely used IE techniques, wrapper methods [2] use a program that extracts data from Web pages and store them in a database. The wrapper could be generated either by human or learned from labeled data, both of which are labor intensive and time consuming. Moreover, the generalization of wrapper is limited when it uses the HTML tags to represent the rule. It also lacks the ability to extend to diverse sites with different representation styles.

Some researches try to mine the data in the Web pages automatically [1][3][4][5] using unsupervised learning approaches such as clustering and grammar induction, or heuristic rules. Most of these techniques require well-formatted tables, and at least two data objects (records) appearing in a page. So their function is largely similar to table detection and data extraction from tables. But our study tries to handle more complicated cases, namely, the representation of data objects represented in table, (nested) listing and text fragment surrounded by HTML tags, and any number of data objects exist in the pages.

Web page cleaning is another crucial task in Web mining. It is used to identify the redundant, useless or irrelevant components (e.g., ads bar and category) in which users are not interested. Current research used priori knowledge or supervised learning to detect frequent templates [6], coherent content blocks [7] and site style tree [8] tied to this task. Here we emphasize the development of an automated approach to detect the important portion (data region) rather than to eliminate these unimportant components in the complex pages.

3. Our Approaches

The key function involved in our technique is how to evaluate the similarity among the components in the pages and through which, partition out the descriptions of data objects. Obviously, string matching and the “bags of words” techniques cannot be used to identify irrelevant contents and partitioning objects. Although the task seems complex, we rely on the following observations to accomplish our goals.

a. In the same Web site, we observe that pages about the objects in the same category, such as products, services and member listings, always have similar representation structure and similar irrelevant contents. This is because in most such cases, they are produced by the same dynamic programs or templates that are carefully maintained by the developers. Thus we could make use of the stable representation structure of the pages that we are investigating to identify useful contents.
b. If we ignore the title of the pages, we expect the data region to appear in contiguous area, both visually on the screen and in raw source of Web pages. However, although the description for each data object is visual contiguous, the raw source of such objects might not be contiguous.
c. If we compare similar pages from the same site using HTML elements as the unit of measure, we found that most of the content differences appear in the data region, because that are used to describe different data objects with different attribute values. Thus by choosing the right content features, we should be able to make the structure corresponds to data regions from different pages to be different, while identifying irrelevant components as similar. This is because the frequency of different elements appearing in irrelevant components is quite low as compared to that in the data regions.

Suppose that the users used the spiders to download all the pages from the special site, or collected all the pages from the service providers. We denote this set of Web pages as P_{SET}. The steps involved in extracting data objects (or product descriptions) are as follows.

a. We first identify a page P_i that is most similar to the user’s query on products or services.
b. We extract the DOM structures [13] of all the pages and use that as the logical structures to compare the

similarities of Web pages. We simply extract the text fragments at the node of the DOM structure.

c. We use the tree-based kernel [9, 10, 11] to evaluate the distance between the Web pages, since the tree-based kernel could reflect the similarities in both the structure and content. We select any page p_j from P_{SET} and calculate the kernel $K(p_i, p_j)$. Because the pages similar to p_i are created by the same dynamic program or template, they should have similar structure and share many common tags among the nodes. Thus, it is not difficult to setup the threshold τ for similar page selection. The set of the pages similar to p_i is:

$$P_{SIM} = \{ p_j \in P_{SET} \mid K(p_i, p_j) > \tau \} \tag{1}$$

d. We calculate the novelty value to be described in next section for each subtree in the parse trees of similar pages. The novelty of a subtree is defined as the weighted sum of all the nodes in the corresponding tree and could reflect the degree of repetition among P_{SIM}. The data region is the subtree that has the maximal novelty (or lowest repeatability) since they have distinctive descriptions about different objects.

e. We detect the structure representation (see Fig. 2) within the data region. If there is more than one object involved, we need to partition them into different data objects and output as different XML files as described in Section 5.

4. Detecting Data Region among the Pages

4.1 Features of data region

We first generate the HTML parse tree from the raw source of the Web page based on the HTML elements embedded in the hierarchical structure, where each node in this tree is an elements. Data region is then a subtree that contains the description of the desired objects. For example, the technical features about the vending notebooks, a table listing of staff, and stock information etc. The characteristic of the data region is that it is typically the largest contiguous region in the Web page having distinct tokens that are used to depict distinctive objects. So we define the concept of repeatability for nodes and subtrees in the parse trees in order to identify data regions.

Here we use a converse measure approach, the repetition degree, to estimate the novelty. This repeatability measure is similar in concept of the kernel method. We try to get the most similar node which lies in similar context. Here, similar node means that they share content tokens and attributes of HTML elements; and similar context is simplified as having similar parents.

4.2 Formalization

Definition 1: The content similarity of Nodes n_1 and n_2 is defined as the weighted ratio of their shared tokens and attribute tags in HTML elements. Supposed that the tokens and attributes for *n_1, n_2* are *t_1, t_2*, and *a_1, a_2* respectively, note that the number of tokens and attributes are denoted by |.|; and the number of shared attributes is $s(a_1, a_2)$. So the similarity of n_1 and n_2 is

$$Sim(n_1, n_2) = (\log(\sqrt{|t_1||t_2|}) + 1)s(t_1, t_2) + w_1 \cdot s(a_1, a_2) \Big/ \sqrt{|a_1||a_2|} \tag{2}$$

Here, w_1 is the weight of the shared attributes and is set to 1 in our study $s(a_1, a_2) \big/ \sqrt{|a_1||a_2|}$ represents the ratio of shared attributes or the proportion of common HTML tags in nodes n_1, n_2. $(\log(\sqrt{|t_1||t_2|}) + 1)$ gives the coefficient of shared tokens $s(n_1, n_2)$ which has larger value when the number of similar tokens is higher. $s(n_1, n_2)$ is the similarity of tokens defined as:

$$s(t_1, t_2) = \begin{cases} 1 \; if \;\; t_1 = t_2 \;\; is\ idential\ \ in\ string - matching \\ .5 \quad\; if \;\; t_1, t_2 \;\; belong\ \ to\ the\ \ same\ \ type \\ 0 \qquad\qquad otherwise \end{cases} \tag{3}$$

Here t_1 and t_2 belonging to the same type means that are of same semantic type such as number, currency, e-mail and so on.

Definition 2: Supposed that nodes n_1 and n_2 (not roots) are in parse trees T_1 and T_2 respectively, the Repeatability of n_1 with respect to T_2 is

$$R(n_1, T_2) = \max_{\forall n \in nodes\ (T_2)} (sim(n_1, n) + w_2 \cdot sim(parent\ (n_1), parent\ (n))) \tag{4}$$

where *parent*(.) denotes the parent node of n; and w_2 reflects the influence from the context of nodes n_1 and n_2, which is set to 0.5 in our system.

Two sub-trees have high repeatability if they are similar in both contents and context (or structure).

Definition 3: If ST_1 is a subtree in parse tree T_1, the Repeatability of ST_1 with respect to T_2 is

$$R(ST_1, T_2) = \frac{R(rt(ST_1), T_2) + w_3 \cdot \sum (R(ST_x, T_2) \cdot |ST_x|)}{1 + w_3 \cdot \sum |St_x|} = \frac{R(rt(ST_1), T_2) + w_3 \cdot \sum R(ST_x, T_2) \cdot |ST_x|}{1 + w_3 \cdot (|ST_1| - 1)} \tag{5}$$

where *rt*(T) denotes the function of tree T; and ST_x is the child node of $rt(ST_1)$. $R(ST_x, T_2)$ is calculated recursively based on Formula (5). w_3 (set to 1 here) gives the contribution of the repeatability from the children of the root. $R(ST_1, T_2)$ is the average Repeatability of the nodes involved.

Repeatability of subtree and other parsing trees is used to identify and remove redundant components in complex product pges, as such irrelevant components tend to be similar. In order to avoid the cases where some data objects sharing many similar descriptions that will increase the repeatability of the data region, we will consider a group of N (i.e. N = 3-5) pages (called window) when evaluating the repeatability of the subtrees. We randomly select N pages from P_{SIM} and compute the average repeatability of their corresponding parse trees as the final evaluation measure as.

Definition 4
$$R(n_1) = \underset{T\in Tree(P_{SIM})}{\overset{N}{avg}} R(n_1, T) \qquad (6)$$

The largest subtree ST_1 covering the novel data region should have the smallest repeatability. So we need to find the larger subtree that has smaller repeatability. This subtree always contains all nodes about the data objects with novel vocabulary. We could use $(log(|ST_1|)+1)/R(ST_1)$ to achieve this objective. The subtree ST_1 with the largest $(log(|ST_1|)+1)/R(ST_1)$ will be the data region.

Definition 5: Data region for parse tree T_1 $DR(T_1)$ is
$$DT(T_1) = \arg\max_{ST_i\subseteq T_1} \log(|ST_i|)+1 \big/ R(ST_i) \qquad (7)$$

Note that the use of log(.) function is used to remove the influence of very large subtree with many nodes.

Here we briefly discuss the rationale of the above formulae. Supposed that ST_1 is the right data region and there is a subtree ST_2 rooted at the next level of ST_1 that has the largest $(log(|ST_2|)+1)/R(ST_2)$. ST_2 could then be wrongly considered as the spurious data region. However, there should be at least one sibling subtree of ST_2, has small repeatability also. Thus all of them would contribute to low repeatability to their parent ST_1, and this would cause ST_1 to have the largest $(log(|ST_1|)+1)/R(ST_1)$ than others. On the contrary, if there is no such sibling subtree of ST_2 that has low repeatability, it means that other nodes in $\{ST_1\text{-}ST_2\}$ do not contain the description about the data objects. Hence ST_2 being the data region is reasonable.

4.3 Algorithm

Based on the above discussion, we design an algorithm to detect data region as follows. The computation cost is $N*|T|^2$, where N is the length of window, $|T|$ is the number of nodes in the parse tree.

```
GetDataRegion(Current_Parsing_Tree $T_0$,
      Parsing_Tree_Set {$T$}, window_size $N$)
{
1.   Randomly select $N$ trees $T_1, \dots, T_N$ from {$T$};
2.   For each (node $n_i$ in nodes($T_0$))
3.   {
4.        Calculate $R(n_i, T_k), k=1, \dots N$;
5.        $R(n_i) = \sum_{k=1}^{N} R(n_i, T_k) \big/ N$
6.        Recursively calculate the repeatability for
          subtrees rooted at $n_i$;
7.   }
8.   Return the subtree having the largest
     $(log(|ST_1|)+1)/R(ST_1)$.
```

5. Partitioning Objects in Data Regions

5.1 The Data Object Representation Structure

There are five typical representation structures for objects as shown in Fig. 2. Of course, some pages may contain a combination of the above structures, such as a horizontal tabular structure (2b) embedded in a listing structure (2c) for a person's CV. But the fundamental organization in the data region usually uses only one type of representation. This is true especially when the contents in the data region are dynamically generated by the dynamic programs in most commercial sites.

	CPU	HD	Screen
Satellite 1000	PIII 800M	20G	14.1
Satellite 3000	PIII 1.5G	40G	14.1

(a) Horizontal tabular structure

	Satellite 1000	Satellite 3000
CPU	PIII 800M	PIII 1.5G
HD	20G	40G
Screen	14.1	14.1

(b) Vertical tabular structure

Toshiba 1000
- CPU PIII 800M
- HD 20G
- Memory 256M
- Screen 14.1

Toshiba 3000 (optional)
- CPU PIII 1.5G
- HD 40G
- Memory 512M
- Screen 14.1

(c) (Nested) List

Toshiba 1000	CPU PIII 800M; HD 20G; Memory 256M; Screen 14.1

(d) Fragment structure

Toshiba 1000, 256 MB PC2100 DDR RAM up to 2048MB….

(e) Sentence

Fig. 2: Typical representation of extracted objects

If a set of objects are represented in many pages with different structures, it is difficult to compile some general wrappers to extract information from them. Typical wrapper rules such as "<B>Country</B> <I>Code</I>" induced by CCWRAP [2] from one site would likely to fail on other sites since the formats of pages in the unknown sites are likely to be different. But if we are able to separate data objects belonging to different products, it would simplify the task of extracting detailed attributes of such objects.

5.2 The Output Structure

Actually, not all nodes in the data region have low repeatability. Some nodes appearing in the similar location among the trees that have high repeatability could help to reveal the framework for depicting the objects. For example, the nodes such as "CPU" and "HD" would repeat again and again and thus have high repeatability. We call those nodes Attribute Names about the data objects, since they provide the right attribute names to tag the descriptions in the other nodes near them. If we use XML to output the result of one data object, it could be.

```
<Object>
    <Title>Satellite 1000</Title>
    <CPU> PIII 800M</CPU>
    <HD>40G</HD>
</Object>
```

Here Title is the object name. We compose the pair of attribute name and its description as entity in XML for visualization and usage convenience. If there is only one object in the data region, the title might be located in HTML tags <title> or <H1>.

This XML output is self-explainable and flexible. All XML output files from these similar pages will have the same structure. Many software agents could efficiently process it.

5.3 Algorithm for Object partition

If there is only one object involved in a data region, the transformation from data region to XML output is straight-forward. When multiple objects appear in the data region with one of the typical organization styles as shown in Fig. 2, we employ the divide and conquer strategy with heuristic knowledge to detect the number of data objects in each data region as follows.

- Tabular structure. We first use the method reported in [12] to regulate the unified cell tagged by "Colspan=n" or "Rowspan=n". We compare the coordinates of cells in each column and each row. Since the same attributes about different object will be aligned either vertically or horizontally in display, we could use the coordinate values to determine whether it is the horizontal or vertical tabular structure. If the average similarity among the nodes in each column is greater than that among the nodes in each row, it means that it is the vertical representation structure. For vertical structure, we simply compare the first column with the other columns, and verify whether the cells in the first column are about the proper attribute names or general attribute values. The steps for constructing data objects in horizontal tabular structure are similar.
- (Nested) Listing structure. In order to find out whether it contains only one object or a set of neighbouring objects, we need to investigate the repetition within the data region. If we can find similar sub trees that cover most of the data region, they will be used to generate different data objects. Otherwise, we consider the whole data region as one object and hence one XML output file will be generated.
- Fragment and Sentence structures. We assume that such region contains only one object. We parse the whole fragment as an entity in an XML output file.
- Compound structure. It is the combination of the above cases. It is usually maintained by human being. Fortunately for such intricate type, there is usually only one object involved in most cases. Our strategy is simply to parse the entire parse tree into an XML file.

The following is the pseudo codes for partitioning objects in the data region. Here t_1 is the threshold for selecting the candidates of attribute names; and t_2 is the threshold for differentiating the objects in listing structure. The maximal computation happens in the detection of direction in the tabular structure. We need to compare all cells in each column and each row, so the computation cost is $|T_0|^3$.

```
PartitionObject(Current_Data_Region_Tree $T_0$,
  threshold $t_1$, $t_2$)
{
1.  Foreach (node $n_i$ in nodes($T_0$))
2.  { if ($R(n_i)$) > $t_1$)
3.     { Tag $n_i$ as attribute_name;}
4.  }
5.  if ($T_0$ is tabular structure)
6.  {
7.     Regulate the table;
8.     Detect the tabular representation direction --
       vertical or horizontal;
9.     Combine the pair of corresponding cell and attribute
       name as an XML elements;
10.    Parse the first column or row as title;
11. }
12. else if ($T_0$ is listing) //one object in this case also
13. {
14.    For each (node $n_i$ in the second layer nodes of $T_0$)
15.    {
16.       Calculate the inner repeatability $R(n_i, T_0)$;
17.       if ($R(n_i, T_0)$ > $t_2$)
18.          {Parse the subtree rooted at $n_i$ as XML output;}
19.       else //only one object
20.           {Parse $T_0$ as XML output; break;}
21.    }
22. }
23. else  //other structure
24. {   Parse $T_0$ as XML output; }
}
```

6. Experiment and Discussion

Our experiments try to detect and partition data object from complex pages about products, such as technical data for "*satellite PC*". This is the questions frequently asked in real applications and the users expect comprehensive answers involving tabulated results from multiple sites.

As this is a new area, it is hard to find a publicly available test corpus to test our technique. Current available corpora on faculty category, product category, and search snapshot are relatively well-structured with simple contents. Moreover, they assume fixed answer template. In this research, we expect to extract detailed object level information (e.g. personal detail) rather than just category level information (faculty category) with unknown template structure. For these reasons, we need to build our own set of test corpus, comprising more abundant contents in product information. We therefore collected pages regarding notebooks and computers from different retailers and review sites as listed in Table 1. From these sites, we selected 100 pages (with at least 10 pages from each site) that contain detailed descriptions of the target objects. About 90% of pages present product information in the form of tabular or list structures, while the rest present products in the form of text fragments or sentences.

Table 1: Sites for downloading product pages

http://www.pcworld.com/reviews/chart_test_report
http://sg.hardwarezone.com/priceguide/cat.php
http://www. amazon.com
http://www.gateway.com/home/products
http://www.csd.toshiba.com/cgi-bin/tais/pc/pc_home.jsp?comm=ST
http://list.auctions.shopping.yahoo.com/23336-category.html?alocale=0us
http://www.nextag.com/Notebooks~300359z0zBwzmainz5-htm

For each Web page, we built the parse trees using the HTML DOM [13]. The kernel method for retrieving similar pages from the same site could achieve quite high performance. This is because there are high similarities between the pages produced by the same program, and they are very dissimilar to other pages that are hand-coded or derived from different programs. Because of the limitation of paper length, here we focus only on evaluating the two most critical steps in our framework – the quality of the extracted data region; and the accuracy of partitioning the objects within the data region.

6.1 Test of Data Region Extraction

Since the data region is a subtree of the parse tree in the corresponding page, we could evaluate the quality of the detected data region in two aspects: (a) whether it covers all nodes depicting the data objects; and (b) whether it contains irrelevant components. The intuitive approach is to compute the overlapping degree between the test data region and the ideal data region using recall and precision. However, because the cost of missing the nodes about the objects is much higher than that of covering irrelevant nodes, hence recall is more important than precision. Fortunately, missing of data object nodes nearly never happens in our testing, although there is about an average of 1-6% of nodes that are not relevant to the data objects directly. We found that this is caused by the definition of data region. For example, if there are three nodes a, b, and c in the second layer of the data region, two subtrees A, B rooted at a and b are about two data objects, but a small subtree C rooted at c contains irrelevant content (such as a link to the promotor page). We had to include subtree C in the proposed data region if we want the subtree to cover A and B. This problem only lies in the pages using listing structure, which may be manually maintained. They do not have high quality structure since the nodes about the data objects are not encapsulated and mixed with irrelevant nodes. But these irrelevant nodes are likely to be ignored in the following object partitioning step.

We also found that the length and number of nodes in the data region are much smaller than that in the original Web page (reduce by an average of more than 80%) when most irrelevant components are excluded. Here, many large subtrees in the parse tree, such as scripts, intrinsic data blocks, and visual elements (search panel, advertisement bar), are eliminated since they have high repeatability.

6.2 Test on the Partitioning of Data Objects

In the step of partitioning objects, we divide the data regions into XML output, each corresponds to a data object. Table 2 gives the statistics of the test corpus in terms of the distribution of data representations and the number of unique data objects or output files. The performance of partitioning objects for each unique data object is tabulated in Table 3, which shows an average F1 of 94 %. Our error analysis reveals the following sources of errors. One typical error occurs when analyzing the structure of raw table (vertical and horizontal tabular) entries involving unknown attributes, which are being used to construct spurious data object. Another source of errors is that the system mistakenly partition one object into many objects, where the content features of grouped attributes are very similar to each other. The other major source of errors is the nested list structure, as the quirky similarity between the sub-trees might mislead segmentations.

Table 2 Data objects distribution

Type	Representation	# of pages	# of data object
1	Vertical Tabular	30	126
2	Horizontal Tabular	30	87
3	Listing	30	40
4	Text Fragment	4	7
5	Sentence	5	5
6	Combined	0	0

Table 3: Performance of partitioning data objects

Tyoe	Correc t	Incor-rect	Miss-ing	Pre. (%)	Recall (%)	F1 (%)
1	126	5	0	96.2	100	98.1
2	87	9	0	90.6	100	95.1
3	34	7	6	82.9	100	90.6
4	6	3	1	66.6	85.7	79.9
5	5	5	0	100	100	100
Avg.	258	29	7	90.8	97.4	94.0

In general, we found that we are able to extract all data from well-structured pages with no missing cases but with some incorrect detection. This conclusion is similar to that reported in [1] on very simple cases. Also, our overall performance of constructing data object is near to that of object mining and extraction reported in Omini [14], but we could process more complicated structures such as nested listing and fragments.

7. Conclusion

Web page content mining is the key technology in realizing semantic Web and business intelligence. This paper focuses on tackling the problem of detecting and partitioning the data objects from complex Web pages, where the presentation structure is varied and the useful data is surrounded by many irrelevant components. Our tests demonstrate that our system is able to correctly partition the data objects with over 97% in precision and over 90% in recall.

The main contributions of our research are three-fold. First, we define the similarity measure between the nodes and construct the Repeatability measure to evaluate the novelty of nodes or subtrees appearing in the selected page set. These measures can be used to effectively reflect the essential features in the irrelevant components, data regions, and the description framework of data objects.

Second, we propose an automated approach to identify the data region, which is the smallest subtree of the parse tree covering the description of all desired data objects. Further processing of data region is therefore free from the interference from the irrelevant components.

Third, we investigate the algorithm for partitioning the data objects in the data region. This algorithm could recompose the non-contiguous description about the different objects and produce the independent self-explainable XML output files for the objects.

To improve the performance of our system in processing complex pages, we need to further improve and fine tune techniques for removing irrelevant information from these pages, and extracting distinct data objects from the data region. We also need to develop larger test corpus and methodologies to test our new framework. Finally, we would like to use this research as the base to develop techniques to compare and reason about product information from multiple commercial Web sites.

References

[1] B. Liu, R. Grossman, & Y. Zhai. Mining Data Records in Web Pages. KDD-2003.

[2] N. Kushmerick, Wrapper Induction: Efficiency and Expressiveness, AI 118 pp15-68, 2000.

[3] K.Lerman, C. Knoblock, & S. Minton, Automatic data extraction from lists and tables in Web sources. IJCAI-01, Workshop on Adaptive Text Extraction and Mining, 2001.

[4] D. Buttler, L.Liu, C. Pu, A fully automated extraction system for the World Wide Web. IEEE ICDCS-21, 2001.

[5] C-H. Chang, S-L Lui, IEPAD: Information extraction based on pattern discovery, WWW-10, 2001.

[6] Bar-Yossef, Z. and Rajagopalan, S. Template Detection via Data Mining and its Applications, WWW 2002, 2002.

[7] Shian-Hua Lin and Jan-Ming Ho. Discovering Informative Content Blocks from Web Documents, KDD-02, 2002.

[8] L. Yi, B. Liu. Eliminating Noisy Information in Web Pages for Data Mining, KDD-03, 2003.

[9] K.R. Müller, S. Mika, et al, An introduction to kernel-based learning algorithms. IEEE Neural Networks, 12(2):181-201, 2001.

[10] M. Collins & N. Duffy, Convolution Kernels for Natural Language, Advances in Neural Info Proc. Sys., vol(14), 2002.

[11] T. Gärtner, a Survey of Kernels for Structured Data. Newsletter of the ACM SIGKDD. 5(1), Jul 2003.

[12] K. Shimada, A. Fukumoto & T. Endo, Information Extraction from Personal Computer Specifications on the Web Using a User's Request, IEICE on Information and Systems, pp1386-1395, 2003.

[13] DOM, http://www.w3.org/TR/DOM-Level-2-HTML/ 2003.

[14] D. Buttler, L. Liu, et al. OminiSearch: A method for searching dynamic content on the Web, ACM SIGMOD 2001.