Automated Web Data Mining Using Semantic Analysis

Wenxiang Dou[1] and Jinglu Hu

[1] Graduate School of Information, Product and Systems, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu-shi, Fukuoka, 808-0135, Japan
william@ruri.waseda.jp,
jinglu@waseda.jp

Abstract. This paper presents an automated approach to extracting product data from commercial web pages. Our web mining method involves the following two phrases: First, it analyzes the data information located at the leaf node of DOM tree structure of the web page, generates the semantic information vector for other nodes of the DOM tree and find maximum repeat semantic vector pattern. Second, it identifies the product data region and data records, builds a product object template by using semantic tree matching technique and uses it to extract all product data from the web page. The main contribution of this study is in developing a fully automated approach to extract product data from the commercial sites without any user's assistance. Experiment results show that the proposed technique is highly effective.

Keywords: Web data extraction, product data mining, Web mining.

1 Introduction

With the information time coming, more and more companies manage their business and services on the World Wide Web and thus these web sites have an explosive growth. Huge amounts of product have been displayed in respective commercial web sites using fixed templates. It has important meaning to extract these product data for offering valuable services such as comparative shopping and meta-search, etc.

The early approaches use the information extraction technique called wrapper [1], which is a program that extracts data from web pages based on a priori knowledge of their format. The wrapper could either be generated by a human being (called programming wrapper) or learned from labeled data (called wrapper induction). Programming wrapper needs users to find patterns manually from the HTML code to build a wrapper system. This is very labor intensive and time consuming. Systems that use this approach include RAPIER[2], Wargo[5], WICCAP[12], etc. The wrapper induction uses supervised learning to learn data extraction rules from a set of manually labeled examples. Example wrapper induction systems include Stalker[3], WL[6], etc. These wrapper construction systems actually output extraction rules from training examples provided by the designer of the wrapper. But, they have two major drawbacks. Firstly, they require a previous knowledge of the data. Secondly, additional work might be required to adapt the wrapper when the source changes.

S. Zhou, S. Zhang, and G. Karypis (Eds.): ADMA 2012, LNAI 7713, pp. 539–551, 2012.

To overcome these problems, some automatic extraction techniques were developed to mine knowledge or data from web pages [7], [10]. Embley et al. [4] uses a set of heuristics and domain ontologies to automatically identify data record boundaries. But the method requires users to predefine a detailed object-relationship model. Zhai et al. [14] proposes an instance-based learning method, which performs extraction by comparing each new instance to be extracted with labeled instances. This approach also needs users to label pages when a new instance cannot be extracted. In [8], the system IEPAD is proposed to find patterns from the HTML tag string of a page, and then use the patterns to extract data items. However, the algorithm generates many spurious patterns. Users have to manually select the correct one for extraction.

In [9], [11], [13], these automatic extraction techniques require multiple similar pages from the same site to generate a template and extract data. The typical system is RoadRunner [9], which infers union-free regular expressions from multiple pages with the same template. It is a limitation for our opnion: not all web sites can find several pages with the similar structure for every product.

In this paper, we proposed a novel technique to perform automatic data extraction. There are three different features between our method and existing automatic extraction techniques: First, we just need a single page with lists of product data. But most existing methods require multiple pages. Second, our method is a fully automatic extraction technique without any human's labor. Third, existing methods execute tag matching for finding repeated pattern based on whole page. There are two obstacles: 1) The navigator and advertisement bar also contain many repeated structures. It is a problem to accurately estimate which one is a correct repeated pattern for a page with the complex structure. 2) For similar pages of the same site, they have almost the same structure. The automatic extraction methods requiring multiple pages are prone to see a whole page as a data record. So, most existing approaches have a low accuracy. However, our method, firstly, extracts maximal repeated semantic information pattern from a page. Then, it searches the product data region and finds product data in this page by matching the extracted semantic pattern. Finally, it generates the correct repeated pattern from the found product data and uses it to extract all product data from the page. The maximal repeated semantic information pattern is more appropriate to identifying data region than the repeated tag pattern on commercial web pages because the product data with the rich semantic information assure the accuracy of the pattern. So, our method can avoid above two obstacles. Experiment results show that our system has higher accuracy than most existing automatic extraction methods.

The remaining of the paper is organized as follows. Section 2 presents the overall procedure for extracting product data. Section 3 describes the algorithm for finding product data region and extracting product data. We present and analyze experimental results in Section 4. Section 6 concludes our study.

2 Our Approach: Mining Produce Data on the Web Sites

2.1 Product Data Representation Structure

There are three typical representation structures for product data from the view of the page as shown in Fig. 1. However, on the DOM tree, these representations are written in HTML by using following two structures: single data region or multiple data

regions structure as shown in Fig. 2. In fact, the data region is one of the sub-trees of DOM tree and all product data are the child sub-trees of the data region. In here, we have to explain two definitions about product data region.

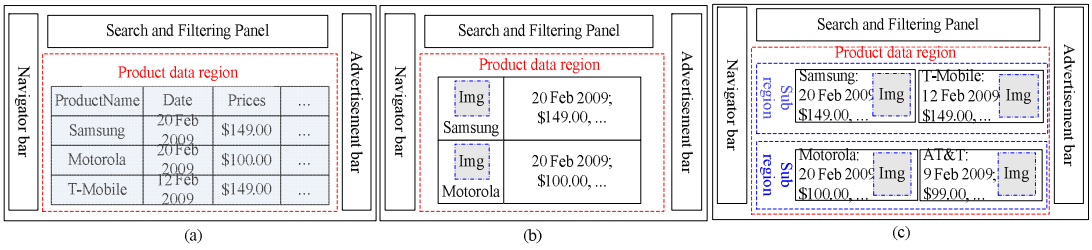

Fig. 1. Typical representation of product data on the commercial web pages. (a) Tabular structure. (b) List structure. (c) Block structure.

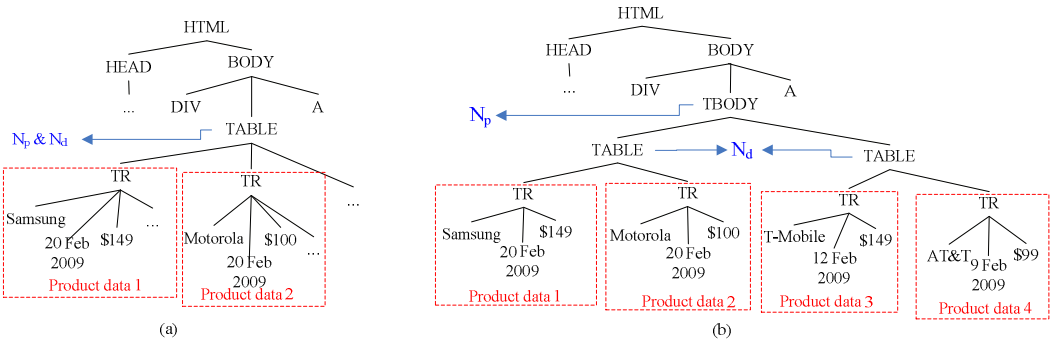

Fig. 2. HTML structure of product data on DOM tree. (a) Single data region structure. (b) Multiple data regions structure.

Definition 1: Product data region is the minimum sub-tree which contains all product data sub-trees in the DOM tree of a pag and N_p is the root node of the sub-tree. In Fig. 2a, the N_p node is the TABLE tag. But the TBODY tag is the N_p node rather than the TABLE tag in Fig. 2b because the TABLE tag does not contain all product data.

Definition 2: Direct product region is the sub-tree that contains product data sub-trees directly in the DOM tree of a pag and N_d is the root of the sub-tree. In Fig. 2a and Fig. 2b, all the TABLE tags are the N_d node. The TBODY tag is not the N_d node because it does not contain product data sub-trees directly.

So, the single data region structure means it just has a N_p node and the N_p node is also a N_d node as shown in Fig. 2a. The multiple data regions structure means it has a N_p node and several N_d nodes, and the N_p node is not a N_d node as shown in Fig. 2b.

From our experiments, we found that the pages containing the tabular structure Fig. 2a must be the single data region structure. Most pages containing the list structure Fig. 2b are the single data region structure. On the contrary, most pages containing the block structure Fig. 2c are the multiple data regions structure. These will be shown in Section 4. Some pages may contain combinations of the above structures, such as a block structure Fig. 2c embedded in an advertisement bar of a page with a list

structure Fig. 2b. It is difficult to mine all product data accurately from this page. Fortunately, most pages use only one type of representation.

Therefore, our method will, firstly, find the N_p node and then recognize the N_d node under the N_p node. Finally, we generate the product object template by aligning the product data sub-trees under the N_d node and use the model to extract all product data from the page. The detailed steps will be described in next phase.

2.2 Outline of Our Approach

The important step for structured data extraction is to find the correct repeated pattern. We proposed a novel and effective technique (PDM) to find the boundary of product data. It is based on the following two observations:

1. Product data have the diverse contents such as image, price, title, description, etc. So, product data region has richer semantic information than any other regions which always have the monotone content on the page. For example, the navigator bar consists of links and the advertisement bar basically is dominated by images.
2. Among the product data, their contents are similar. For example as shown in Fig. 1, every phone product are decribed by the similar contents. The product data region contains many such content model about the product. Therefore, the repeated rich content informationis almost unique in most commercial web pages.

Based on the above observations, we devise the following workflow to identify product data and extract them from a commercial web page:

1. We use the nekoHTML parser to preprocess the page and remove some useless nodes and information for our work from the page such as javascript language, etc. Then, we build the DOM tree of the page.
2. The semantic analyzer developed by us identifies the semantic type of the information located at every leaf node of the DOM tree. Next, The semantic information vector is built for every node from down to up on the DOM tree.
3. We align the semantic information vector of nodes in every level of the DOM tree from the down to up and get the maximum repeated semantic vector pattern (MRSV). Then, the pattern is used to find the product data region and N_d node.
4. In final, we build the product object template of the page by matching the product data sub-trees under the N_d node and use this model to extract all product data.

The difference between our approaches with existing methods is that we use the novel semantic analysis and align technique to identify the structured data. It is effective to be applied in the product data extraction because the MRSV always reflects the correct product data region in most commercial sites. Especially, it also can achieve high accuracy when it handles complicated pages from the commercial sites.

3 Product Data Identification

3.1 Semantic Analysis about the Information of the Page

We develop a semantic analyzer to identify the semantic type on the page. In our work, all pair-tags (like <tr></tr>) on DOM tree are nodes and other tags and contents

under these pair-tags are the semantic information. We classify the semantic information into seven semantic types: title (*TIT*), description (*DES*), price (*PRI*), number (*NUM*), image (*IMA*), single tag (*S_T*) and special (*SPE*).

The semantic analyzer consists of several if-then rules. These rules are used to match with the semantic information and the result deduced by the best matching rule will be as semantic type of the information. The following show the rules:

Definition of Expressions

X: is variable. $(X)^1$ denotes *X* has only one appearance, $(X)^?$ denotes *X* has only one or no appearance, and $(X)^+$ denotes *X* is larger than or equal one appearance. ¬*X* denotes it is not *X*.

W: denotes a word.
C_W: denotes the first letter capitalized word.
T_{ag}: denotes a single tag such as
, <p>, etc.
N_{um}: denotes the number.
Σ: is a alphabet {a, b, c, …}.
Σv: is a character set {$, ¥, £, …}. These characters are popular value expressions.
I_{mg}: The information could be an image. So, I_{mg} denotes an image tag <img …>.
Num(X): the function *Num(X)* denotes the number of the variable *X*.
X·Y: denotes the union between two variables *X* and *Y*.

Semantic Rule Pool

R_{TIT} identifies the title type:

$$If\ (W)^+\ \&\ Num(C_W) >= 1/2 * Num(W)\ then\ TIT. \quad (1)$$

R_{DES} identifies the description type:

$$If\ (W)^+\ \&\ Num(C_W) < 1/2 * Num(W)\ then\ DES. \quad (2)$$

R_{IMG} identifies the image type:

$$If\ (T_{ag})^1\ \&\ I_{mg}\ then\ IMG. \quad (3)$$

R_{S_T} identifies the single tag type:

$$If\ (T_{ag})^1\ \&\ \neg I_{mg}\ then\ S\_T. \quad (4)$$

R_{NUM} identifies the number type including quantity, size, etc:

$$If\ ((N_{um})^+\ \&\ (W)^?)\ |\ (N_{um} \cdot (x \in \Sigma))^1\ then\ NUM. \quad (5)$$

R_{PRI} identifies the price type:

$$If\ ((x \in \Sigma v) \cdot N_{um})^+\ \&\ (W)^?\ then\ PRI. \quad (6)$$

The information which cannot match any rules will be as special semantic type.

3.2 Building Semantic Information Tree

After obtaining the semantic types of leaf nodes, we start to generate semantic information vectors (SIV) for all non-leaf nodes of the DOM tree. The SIV consists of seven items and every item expresses a semantic type as shown in the follows:

$$SIV = [S\_T, IMG, DES, TIT, NUM, PRI, SPE]$$

The SIV of the node is generated by adding up the value of the SIV of its all child nodes as shown in Fig. 3. For example, the leaf node
 and <img> in the lowest level of the tree are tagged the semantic type as the *S_T* and *IMG* respectively by the semantic analyzer. Then, the SIV of their parent node <td></td> is[1,1,0,0,0,0,0] by adding up [1,0,0,0,0,0,0] and [0,1,0,0,0,0,0]. Final, the SIV [4,6,0,6,0,2,0] of the root node <<body></body> can be generated by iterative computing from down to up. From the DOM tree with semantic information vectors, we can see the distribution of the information of the web page clearly. In the next phases, we will present how to use the semantic information to find product data region and identify product data.

3.3 Maximum Repeated Semantic Vector

As we mentioned in the above, the biggest difference between our work and existing methods is that we identify the data region by finding MRSV (maximum repeated semantic vector) pattern of the page rather than repeated tag pattern. For some complex commercial pages, many repeated tag patterns could be found and it is difficult to find a correct one from them automatically. But, due to many products with rich information contained by most pages, the MRSV always is the correct pattern of product data. For obtaining the MRSV of a page, we need to generate the MRSV of every non-leaf node of the DOM tree from down to up and the MRSV of the root generated in the last will be the MRSV of this page. In the following, we explain how to generate the MRSV of a node. There are two steps:

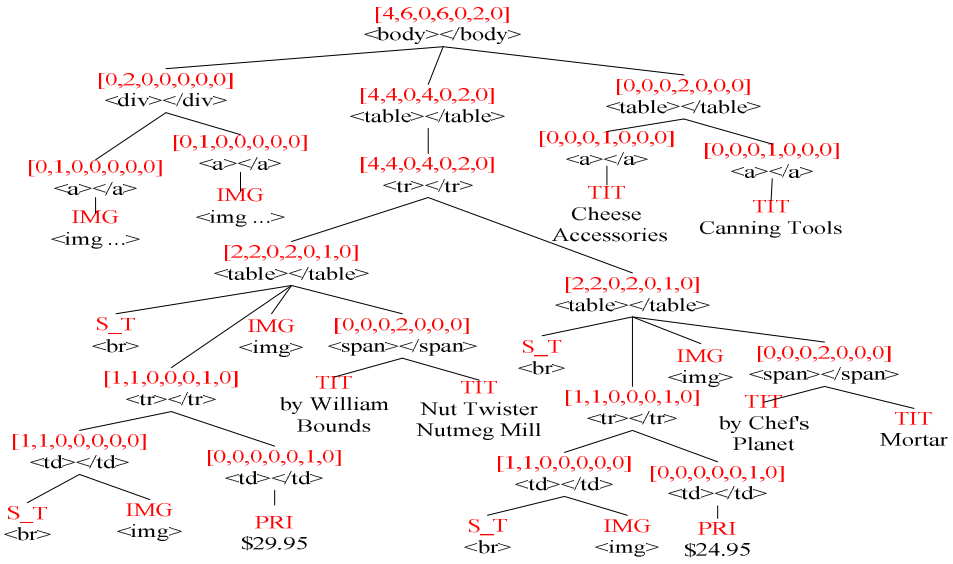

Fig. 3. The DOM tree with semantic information vectors

1. Findind all RSV patterns contained by the node.

Under a node, there could be several RSV patterns. There are two sources for them: 1) the first source is the similar semantic vectors of several child nodes. It can be detected by using vector similarity computation. We designed a new vector similarity formula which can reflect the similarity of semantic types and the information quantity between two SIVs (semantic information vectors). Let us see the definition 3.

Definition 3: Suppose v and w are two semanic information vectors, the similarity of v and w is

$$SIM(v,w) = \frac{\sum_{i=1}^{m} (min(x_i, y_i) / max(x_i, y_i))}{max(n_v, n_w)}, x_i > 0 \text{ or } y_i > 0 \tag{7}$$

Here, x_i ,y_i denotes the value of the ith item of vector v and w respectively, and n_v, n_w denotes the number of the items with values larger than zero in vector v and w respectively. We use this formula to detect whether there are similar SIVs among child nodes of the node. If the similarity of two SIVs is larger than a set threshold, then they are similar. After finding similar SIVs, we get their average as the RSV. 2) The second source is the MRSV patterns generated by child nodes of the node.

2. Selecting the maximum one from found RSV patterns.

After finding all RSV patterns contained by the node, we compare them and select the maximum one as the MRSV of the node. So, we need to transform every RSV into a measure value. Definition 4 shows the transforming formula.

Definition 4: Suppose w is a RSV and the information value of w is as follows

$$V(w) = \sum_{i=1}^{m} y_i \cdot n_w \cdot r_w \tag{8}$$

Here, y_i denotes the value of the ith item of vector w, n_w denotes the number of the items with values larger than zero in w and r_w denotes the repeated number of vector w. Definition 4 means more semantic types, information quantity and repeated number the RSV has, larger value it can obtain. Therefore, the RSV with the maximum value will be the MRSV of the node. Similarly, the MRSV of the N_p node (root node of the product data region sub-tree) with rich semantic information has the highest probability to become the MRSV of the whole page.

For example, Fig. 4 shows the detailed searching process of the MRSV with the similarity threshold 0.7. The three nodes , <table></table> and in the second level of the tree obtain their MRSVs respectively. In the table of the MRSV, n denotes the repeated number and v is the information value of the MRSV. When the algorithm is implemented to the root <body></body>, it firstly detects whether there are similar SIVs among its three child nodes. From the Fig. 4, we can see the SIV [0,2,0,2,0,0,0] of the node is similar with the SIV

[0,1,0,2,0,0,0] of another node because the similarity is equal to 0.75. So, there is a RSV pattern among child nodes of the root and we extract it by averaging the two SIVs. So, the RSV is [0,1.5,0,2,0,0,0]. Second, the algorithm detects whether the child nodes of the root have their MRSVs. In this example, all three child nodes have their MRSVs. In final, there are four RSV patterns for the root. Through comparing their information values, the RSV [2,2,0,2,0,1,0] with the maximum value 56 is the MRSV of the root. Therefore, it is also the MRSV of the whole page.

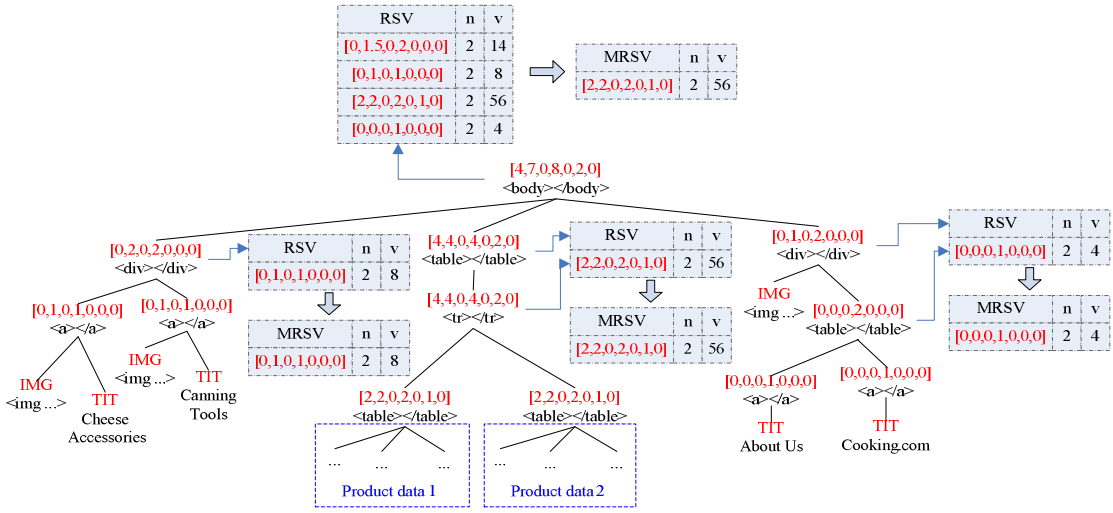

Fig. 4. The searching process of the MRSV pattern with the similarity threshold 0.7

3.4 Identifying the Boundary of the Product Data

In this phase, we will identify the product data using the MRSV. It is easy to identify the product data region (the sub-tree whose root node is the N_p node) because the MRSVs of all nodes are same with the MRSV of the root on the pass from the root to the N_p node if the MRSV of the root is correct pattern of the product data region. But for identifying product data (sub-trees under the N_d node), there are two situations:

1. The MRSV of the root reflects the semantic information of product data directly. This situation means the N_d node is the lowest level node containing the MRSV of the root node. So, it is simple to identify product data in this situation because the sub-trees of product data are under the N_d node directly. Therefore, we firstly find the N_d node, and then we align the SIV of every child node of the N_d node with the MRSV of the root. If the SIV of some child node is similar with the MRSV, then all sub-trees under the child node is the product data information. While, The N_d node can be found by aligning the MRSV because it is the lowest level node whose MRSV is same with the MRSV of the root. The page with a single data region structure belongs to this situation because the N_p node and the N_d node point to the same node.

2. The MRSV of the root reflects the semantic information of the sub data region. This situation means the N_p node is the lowest level node containing the MRSV of the root rather than the N_d node. This situation often happens in the page with

multiple data regions becauses there are a N_p node and several N_d nodes in this page and the N_p node is not the N_d node. When there are several sub data regions sharing most product data averagely in the page, the SIVs of the N_d nodes of these sub data regions will be the MRSV of the whole page. Fig. 5 shows the example about three sub data regions sharing the product data of the page. The <table></table> with the SIV [12,12,0,12,0,6,0] is the N_p node and three <tr></tr> nodes with the SIV [4,4,0,4,0,2,0] are the N_d nodes. If using the above searching algorithm, the three sub data regions will be mistaken for the product data.

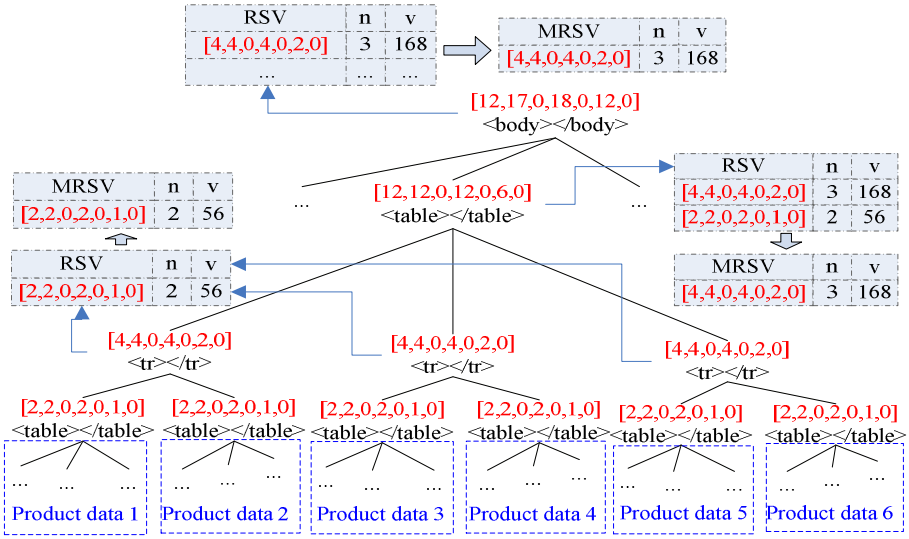

Fig. 5. Identifying the product data on the multiple regions

So, we need to check whether found product data sub-trees are correct. We compare the SIV of the root of the found sub-tree with the product of its MRSV and the repeated number of the MRSV because the SIV of the sub data region basically is the sum of the SIVs of the product data contained by it. If they are similarity, then the found sub-tree is the sub data region rather than product data. So, we continue to extract the real product data from these found sub data regions by aligning the MRSV of them with the SIVs of their child nodes.

For example, let us see Fig.5 again. The three sub data regions with the SIV [4,4,0,4,0,2,0] of the root, firstly, will be extracted and we select one among them to compare its SIV [4,4,0,4,0,2,0] with the product of its MRSV [2,2,0,2,0,1,0] and repeated number 2. Due to the product 2*[2,2,0,2,0,1,0] equal to [4,4,0,4,0,2,0], they are similarity. So, they are not product data and we continue to extract real product data under these sub data regions by comparing the SIVs of child nodes with their MRSV. We can see all <table></table> nodes with the SIV [2,2,0,2,0,1,0] are the root nodes of product data sub-trees.

Therefore, the MRSV information of nodes can help us to find product data region and identify the boundary of the product data. In the last, we generate the product object template from found product data and use it to find all product data.

4 Experiments

In this section, we evaluate our system, PDM (Product Data Mining), which implements the proposed techniques. The system is implemented in Eclipse and runs on a 1.66GHz Double Core with 512MB RAM.

The number of sites in our experiments is 154. The total number of pages is 209 and 85 pages from 30 sites are used to compare our system with RoadRunner system. We use Google engine to select diverse commercial sites randomly from the Web. The selection of pages is also based on the diversity of products and representation structure. These pages are preprocessed by using the nekoHTML parser.

For our system, the selection of the threshold of the semantic vector similarity is important. Therefore, we use four different similarity thresholds to run our system respectively and compare their results for finding the most appropriate one.

Table 1 shows the results of PDM with the four different thresholds 0.65, 0.7, 0.75 and 0.8. The pages are divided into five categories according to the representation and data region structure and are listed in the first and second column. The "L", "T", and "B" denotes three representations List, Tabular and Block structure respectively, and the "S" and "M" denotes single data region and multiple sub data regions. The fourth column marked with "*r*" shows the number of the product data records and the columns marked with "*c*" and "*f*" shows the number of correct product data records extracted by our system and the number of found product data records.

From the values of recall and precision in the last two rows for each threshold, we can see, when the threshold is smaller than 0.7, the recall is the lowest because the product data are easy to be missed in the page with any structure. While, when the threshold is larger than 0.7, the system did not perform better for handling the page with multiple sub data regions. With the threshold be larger, the recall and precision become more and more low. Therefore, the semantic vector similarity threshold with the valueof 0.7 is the most appropriate for our system.

Table 1. Experimental Results

Category		No.of sites	Product Data Mining								
				0.65		0.7		0.75		0.8	
			r	*c*	*f*	*c*	*f*	*c*	*f*	*c*	*f*
L	S	52	856	792	821	852	854	852	854	852	854
	M	12	151	142	146	134	141	136	142	121	136
T	S	9	151	140	148	151	152	151	152	151	152
B	S	21	447	443	445	443	445	443	445	443	445
	M	60	1105	1043	1057	1061	1073	1044	1113	1028	1101
Total		154	2710	2560	2617	2641	2665	2626	2706	2595	2688
Recall				94.46%		97.45%		96.90%		95.76%	
Precision				97.82%		99.10%		97.04%		96.54%	

In the following experiment, Table 2 shows the comparing results between RoadRunner and our PDM system with 0.7 threshold. The column 1 gives the structure type of pages. The column 2 gives 30 commercial sites selected randomly from the above 154 sites and column 3 shows the product type. The column 3 and 4 give the number of pages and the number of product data records in these pages. The columns marked with "corr." and "found" denote the number of correct product data records extracted by the system and the number of found product data records.

The last two rows of the table give the total number of product data records in each column, the recall and the precision of each system.

see p as a r: It means almost the whole page is extracted as a data record.

see dr as a r: It means sub data regions are extracted as data records and the product data under these regions are viewed as their nest structure items.

Table 2. Comparison of Extraction Results of PDM and RoadRunner

	Source				PDM (0.7)		RoadRunner		
category	site	product	page	records	corr.	found	corr.	found	remark
L / S	Apple Store	iPod	2	28	28	28	28	32	4 wrong
L / S	Marblehead	Socks, Bags	3	20	20	20	20	20	
L / S	Israel Sport Shop	Clothes	2	5	5	5	5	5	
L / S	Wristick	Watches	3	20	20	20	20	20	
L / S	Pcuniverse	Camera Lenses	3	30	20	20	-	-	see *p* as a *r*
L / S	Network Camera Store	Network Camera	3	30	30	30	30	30	
L / S	Nobles Camera	Batteries	2	25	25	25	-	-	see *p* as a *r*
L / S	AbelCineTech	Cables	3	29	29	29	29	29	
L / S	Ofease	Canon, Scanner,…	3	42	42	42	42	42	
L / S	Hammicks BMA	Business Book	3	60	60	60	60	63	3 wrong
L / S	Biblion	Fiction Book	3	139	139	139	139	141	2 wrong
L / S	IEEE computer Society	Tutorials	3	30	30	30	30	30	
L / S	Amazon	Movies & TV	3	75	75	75	-	-	see *p* as a *r*
T / S	The Phone Store	Mobile phone	3	40	40	40	40	40	
T / S	AISC Store	books	2	28	28	28	28	28	
T / S	Net32	Alloys	3	60	60	60	-	-	see *p* as a *r*
T / S	Whitakers	Rods, Jackets,…	3	20	20	20	20	20	
T / S	Network Webcams	IP Camera	3	71	71	71	71	73	2 wrong
T / S	Pcmag	Cell Phones	3	30	30	30	30	30	
B / S	Newegg	Books	3	39	39	39	-	-	see *p* as a *r*
B / S	Oldnavy.gap	Man's Shirts	2	70	70	70	-	-	see *p* as a *r*
B / S	Fashion163	Swiss Watch	3	57	57	57	57	61	4 wrong
B / S	Tracsat.co.uk	Sky	3	13	6	10	-	-	see *p* as a *r*
B / M	My Jewelry Box	Diamond Rings	3	72	72	72	39	51	see *dr* as a *r*
B / M	Amazon	DVDs	2	28	28	28	28	30	2 wrong
B / M	ShopSunGlassesOnline	Sun Glasses	3	45	45	45	-	-	see *p* as a *r*
B / M	Motorola	Batteries & Doors	3	45	45	45	-	-	see *p* as a *r*
B / M	Forever Jewelers	Pendants	3	60	60	60	60	60	
B / M	Hartgem	Signet Rings	3	24	24	24	24	24	
B / M	Onhop.ca	Components	3	63	63	63	63	63	
B / M	PerfumeSpace	Men's Perfume	3	48	48	48	48	48	
B / S	Embrace Jewelry	Earrings, Rings,…	3	14	0	6	14	18	4 wrong
	Total		85	1360	1339	1349	935	967	
	Recall/Precision				98.5% / 99.3%		68.9% / 96.7%		

n wrong: It means *n* incorrect data records are found.

The following summarizes the experimental results in Table 2.

1. Our system PDM gives perfect results for every site except for the last one. For three pages of the site, all product data are not found because descriptions of product data in these pages just have an image and a title and surrounding other regions have much richer semantic information than the product data region. From the last two rows, we can see that PDM has a 98.5% recall and99.3% precision. While, RoadRunner just has a recall of 68.9%.
2. When the page has a complex structure, RoadRunner system often happens the "*see p as a r*" error. While, our system will still perform better in these pages.
3. Many repeated patterns will be generated if aligning tag structure by using several pages and the incorrect data records are extracted easily. Therefore, our system has a higher precision than RoadRunner.

The experiment results show that our system has the good effectiveness and high accuracy for mining product data in commercial pages

5 Conclusions

In this paper, we propose a novel and effective approach to extract product data from web sites. Our work is related to the structured data extraction from web pages. Although the problem has been studied by some researchers, existing techniques have respective limitations and most automatic extraction methods have the low accuracy. Our method is a fully automated extraction technique without any human's assistance. We use the novel semantic analysis and align technique to identify the structured data. It is very effective to be applied in the product data extraction. Meanwhile, it also can achieve high accuracy when it handles complicated pages from the commercial sites. Experiments results using a large number of commercial pages showed the effectiveness of the proposed technique.

References

1. Kushmerick, N., Weld, D., Doorenbos, R.: Wrapper induction for information extraction. In: Proc. of the 15th IJCAI (1997)
2. Califf, M.E., Mooney, R.J.: Relational learning of pattern-match rules for information extraction. In: Pro. of the AAAI 1999/IAAI 1999 Conf., pp. 328–334 (1999)
3. Muslea, I., Minton, S., Knoblock, C.: A hierarchical approach to wrapper induction. In: Proc. of the 3th Annual AA Conf., pp. 190–197 (1999)
4. Embley, D.W., Campbell, D.M., et al.: Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents. In: Proc. CIKM, pp. 52–59 (1998)
5. Raposo, J., Pan, A., Alvarez, M., Hidalgo, J., Vina, A.: The Wargo System: Semi-Automatic Wrapper Generation in Presence of Complex Data Access Modes. In: Proc. of 13th Int'l Workshop Database and Expert Systems Applications, pp. 313–320 (2002)
6. Cohen, W.W., Hurst, M., Jensen, L.S.: A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. In: Proc. 11th Int'l Conf. World Wide Web, pp. 232–241 (2002)

7. Embley, D., Jiang, Y., Ng, Y.-K.: Record-boundary discovery in Web documents. In: Proc. of ACM SIGMOD 1999, pp. 467–478 (1999)
8. Chang, C., Lui, S.: IEPAD: Information Extraction Based on Pattern Discovery. In: Proc. of the 2001 Intl. World Wide Web Conf., pp. 681–688 (2001)
9. Crescenzi, V., Mecca, G., Merialdo, P.: ROAD RUNNER: Towards Automatic Data Extraction from Large Web Sites. In: Proc. of.the 2001 Intl. Conf. on Very Large Data Bases, pp. 108–118 (2001)
10. Bar-Yossef, Z., Rajagopalan, S.: Template Detection via Data Mining and its Applications. In: Proc. WWW, pp. 580–591 (2002)
11. Arasu, A., Garcia-Molina, H.: Extracting Structured Data from Web Pages. SIGMOD (2003)
12. Zhao, L., Wee, N.K.: WICCAP: From Semi-Structured Data to Structured Data. In: Proc. of 11th IEEE Int'l Conf. and Workshop Eng. of Computer-Based Systems (ECBS 2004), p. 86 (2004)
13. Ye, S., Chua, T.S.: Learning Object Models from Semistructured Web Documents. IEEE Transaction on Knowledge and Data Engineering 18(3), 334–349 (2006)
14. Zhai, Y., Liu, B.: Extracting Web Data Using Instance-Based Learning. In: Proc. Sixth Int'l Conf. Web Information Systems Eng. (2005)