Data Regions Extraction for Semi-Structured Web Pages Using Bottom-up Approach
การสกัดพื้นที่ข้อมูลจากหน้าเว็บกึ่งโครงสร้าง โดยใช้วิธีการแบบล่างขึ้นบน

Wachirawut Thamviset (วชิราวุธ ธรรมวิเศษ)[1]* Dr.Sartra Wongthanavasu (ดร.ศาสตรา วงศ์ธนวสุ)**

ABSTRACT

In this paper, we propose an unsupervised information extraction system called Bottom-up Wrapper (BUW) for automatic extracting the data regions from the semi-structured web pages such as search result pages, product catalog pages, etc. Although, data records in a semi-structured web page are generated from backend databases and encoded into the HTML with fixed templates from server-side scripts, but these data records are represented without the structural information. Moreover, the complexity of the website is increasing, that make it difficult to automatically identify the correct data region and extract the relevant data records. While, many existing techniques use a top-down approach that starts to identify the data regions before the data records and data items. In another way, we figured out the stated problem in a bottom-up way that starts to analyze the repetitive patterns of data items, which can be used for identifying the relevant data records and data regions. This technique is completely unsupervised and maintenance-free wrapper. For performance evaluation purpose, it is empirically tested on the real world websites. Consequently, it provides the outstanding result that the proposed technique is robust and in many cases outperforms existing wrappers such as RSP and SDE (based on DEPTA).

บทคัดย่อ

บทความนี้ผู้วิจัยได้นำเสนอระบบการสกัดข้อมูลแบบไร้ผู้สอน ที่เรียกว่า Bottom-up Wrapper (BUW) สำหรับใช้สกัดพื้นที่ข้อมูลแบบอัตโนมัติจากหน้าเว็บแบบกึ่งโครงสร้าง เช่น หน้าแสดงผลการค้นหา, หน้าแสดงรายการสินค้า ฯลฯ ซึ่งแม้ว่าระเบียนข้อมูลในหน้าเว็บแบบกึ่งโครงสร้างจะถูกนำมาจากระบบฐานข้อมูล และ ถูกเข้ารหัสเป็นภาษา HTML ตามรูปแบบที่ถูกกำหนดไว้โดยโปรแกรมในฝั่งผู้ให้บริการ แต่ระเบียน

[1] *Correspondent author: twachi@kku.ac.th*

* *Student, Doctor of Philosophy Program in Computer Science (English Program), Faculty of Science, Khon Kaen University*

** *Associate Professor, Department of Computer Science, Faculty of Science, Khon Kaen University*

ข้อมูลเหล่านี้ถูกนำเสนอโดยปราศจากสารสนเทศเชิงโครงสร้าง ยิ่งไปกว่านั้นข้อมูลในหน้าเว็บมีความซับซ้อนของเพิ่มขึ้นเป็นอย่างมากทำให้เป็นการยากที่จะจำแนกพื้นที่ของข้อมูลหลักแบบอัตโนมัติได้ ซึ่งเทคนิคการสกัดข้อมูลที่อยู่แล้วส่วนใหญ่จะใช้กระบวนการแบบบนลงล่าง ที่เริ่มจากการจำแนกหาพื้นที่ของข้อมูลหลัก (main data region) ก่อนจะหาระเบียนข้อมูล (data record) และ ชิ้นส่วนข้อมูลย่อย (data item) ที่อยู่ในระเบียนข้อมูล แต่ในทางตรงข้ามผู้วิจัยได้นำเสนอเทคนิคที่ใช้กระบวนการแบบล่างขึ้นบน โดยเริ่มจากการวิเคราะห์หารูปแบบที่ซ้ำซ้อนของชิ้นส่วนย่อยสุดในระเบียนข้อมูลก่อน แล้วจึงใช้รูปแบบที่พบมาใช้ในการจำแนกตำแหน่งของระเบียนข้อมูล และ หาตำแหน่งของพื้นที่ข้อมูลหลัก ตามลำดับ ซึ่งเทคนิคที่นำเสนอนี้เป็นวิธีการแบบไร้ผู้สอนและสามารถสะกัดข้อมูลจากหน้าเว็บที่ไม่เคยพบมาก่อน สำหรับการทดสอบประสิทธิภาพนั้นได้ทดสอบกับเว็บไซต์จริงโดยมีการเปรียบเทียบประสิทธิภาพกับเทคนิค RSP และ SDE (ซึ่งพัฒนาจากเทคนิค DEPTA) และผลที่ได้พบว่าเทคนิคที่นำเสนอมีความเสถียรและในหลายกรณีได้ผลการสกัดข้อมูลที่มีประสิทธิภาพสูงกว่าเทคนิค RSP และ SDE

Key Words : Information extraction, Bottom-up approach, Semi-structured web pages
คำสำคัญ : การสกัดสารสนเทศ กระบวนการแบบล่างขึ้นบน หน้าเว็บแบบกึ่งโครงสร้าง

Introduction

There is a huge amount of useful information on the World Wide Web. According to the survey of the et al. [1], they detected more than 300,000 database websites, and these database websites contain the massive and valuable information; for example, www.Amazon.com contains several millions of books and other products, or www.Imdb.com contains more than seven millions of data records of entertainment entities as well. Generally, these web sites have provided the query interfaces for giving users instant search for their data that the search result pages are dynamically generated for each user's submitted query. However, the result pages are semi-structured documents, which are only designed for human readability. Therefore, the data records in the result pages are represented without the structural information that it is not appropriate for information processing.

Normally, there are two types of search result pages. The result page that describes a single object is called as *detail-page,* and the page that contains a list of data records is called as *list-page*. In this paper, we focus on the list-page. Figure 1 shows the example of the list-page from Amazon.com that it contains a list of data records in the main data region. Normally, a web page is written in an HTML, which can be transformed into the DOM tree structure; for example, Figure 2(a) shows the DOM tree of the web page of Figure 1. From a layout point of view, this group of data records is formatted with the same template. Thence, it is possible to identify the data records in the list-page by analyzing these repetitive patterns. Technically, it is practical to write a specific program, called *wrapper* for extracting information, and producing structured data records from the web pages of a particular website; for example, the extracted data records that are shown in Figure 2(b). However, each website has an own design template; as the result, if a large number of websites are needed to be extracted, the many specific wrappers must be written for extracting all of them.

The list-page is usually represented in the complex hierarchical structure. The data region in the list-page is the region that contains a list of data record, and these data records are formatted using the same template. However, the data records can be represented in various styles and formatted with different HTML tags such as tables (*<table>*, *<tr>*, *<td>*), list items (*<ol>*, *<ul>*, *<li>*), block (*<p>*, *<div>*,*<dd>*,*<dt>*), etc. In addition, the list-page usually contains other extraneous regions such as a company logo, header pane, footer pane, navigational pane, advertisement boxes and so on. The task of automatic identifying data region is a complicated problem, because the list-pages can be displayed in various layout styles. For examples, Figure 3 shows the four layout styles of list-pages: (a) a simple list-page that data records are in one table with one record per row, (b) a list-page that data records are in a list of tables with one table for one record, (c) a list-page that data records are in table(s) with three records per row, and (d) a list page that contains multiple groups of data records with different templates.

From the surveys [2, 3], many existing techniques preview use a top-down approach that starts to identify the data regions before the data records and data items. In another way, our previous work RSP (Repetitive Subject Pattern) [4, 5] figured out the stated problem on bottom-up way. The RSP algorithm is based on the assumptions that each data record in the list-page must have unique subject item, which is the leaf node of the DOM tree. Instead of mining data regions, the algorithm uses the repetitive pattern of the subject item to identify the data records directly by traversing the tree in bottom-up style. However, the quality of the generated wrapper depends on the performance of the subject finder algorithm that it may return the wrong subject when the input document is out of its conditions. We refer the interested reader to [5] for a detail of RSP algorithm.

This paper proposes the novel technique called BUW (Bottom-up Wrapper) to perform automatic data extraction from a list page by adapting a bottom-up approach from RSP algorithm [5]. Instead of only using repetitive pattern of the subject items, the proposed algorithm uses the repetitive pattern of all data items for identifying data records, and using the pattern of data records for identifying the relevant data regions. As the result, the proposed method is a completely unsupervised system and does not require the subject finder algorithm. Given only one input page, it can automatically extract the data records from all data regions, and identify the relevant data region as well. The rest of the paper is organized as follows: Section 2 presents the related work, and Section 3 presents the proposed algorithm (BUW); we provide the experimental results with various data sets in Section 4 and conclude the paper in Section 5 with some future directions of our work.

Related Work

From the survey in [2], the information extraction techniques have been categorized into four categories: manual, supervised, unsupervised and semi-supervised. Firstly, the *manual* approaches such as WebOQL [6] and XWRAP [7] that users need to program a wrapper or write the extraction rules for extracting data from each website by hand. Secondly, the supervised approaches such as NoDoSE[8] and Amilcare[9] use machine learning techniques to generate a wrapper that users need to create labeled training data of some sample pages for each website. Thirdly, the *semi-supervised* approach such as Grubber [10] and OLERA [11] can generate wrappers from unlabeled pages, but they need some interaction or decision from the users to train the extraction rules. The last category is the *unsupervised* approaches [5, 12-16] that can automatically generate wrappers for an unseen web site. Because the *manual, supervised and semi-supervised* systems require the considerable inputs from human, the applicability and the capability of these systems are comparatively better than the *unsupervised* system. However, the extensibility of the *unsupervised* system is better than other categories. As well as, the proposed method is developed to be the unsupervised.

Many existing unsupervised information extraction systems for semi-structured web pages [12-17] had been designed using top-down approaches for their data extraction tasks. The common top-down approach algorithms are: (1) identifying data regions or data sections, (2) extracting the data records from the identified data regions, and (3) aligning data items of the extracted data records. As the result, the information extraction tasks have to analyze more and more for identifying the relevant data regions and ignoring the irrelevant regions. For examples, the well-known top-down systems, MDR [14] and DEPTA [15] identify the candidate regions, and extract all of the data records from all of candidate regions by comparing the similarity of each sub tree, and finding the repetitive patterns within the document by using the tree comparing algorithm. However, the normal tree comparing algorithm also runs in a time complexity of $O(n_1n_2)$ where n_1 and n_2 are the number of nodes in each tree. Furthermore, the size and complexity of the modern web pages are increasing, that can slow down the tree-comparing algorithm.

Proposed Method

This section presents the proposed method: BUW (Bottom-up Wrapper) that the input is a list-page, which contains a list of data records; for example, the list-pages in Figure 1 and Figure 3. Technically, the contents, structures and styles of an

HTML document can be represented by the Document Object Model (DOM), which is defined by W3C. Formally, each HTML element in a document can be defined as: *Node = {tag, parent, root, value, attribute, children}*, where *tag* is a name of the HTML tag, *parent* is the parent node, *root* is the root node, *value* is a text of this node, *attribute* is a set of attributes, and *children* is a list of child nodes.

Therefore, DOM presents a document as a hierarchy of Node objects, which is called DOM tree. An HTML document considered a tree with *<html>* element acting as the root node and elements with no nested tags corresponding as the leaf nodes. Normally, the list-page can contain single or multiple data regions. Each data region contains groups of data records called data section, each data section contains a list of data records, and each data record contains a number of data items. Figure 2(b) represents the structure of the page as DOM tree that data regions, data section, data records and data items are assembled in the certain levels of the trees. We defined the data items as the leaf nodes of the trees; the data records are sub trees that contain the group of associated data items. The data section is the sub tree that contains a group of associated data records. The data region is sub tree that contains data sections. In case of a single data section document, the data section and data region can be the same node. Normally, if the data records of the same group (section or region) were generated by the same template, then they should have similar structure.

Because every data record contains a group of data items that data records of the same group must have a similar pattern of data items. As the result, instead of starting by mining data region, the first step of the propose system is finding all data item nodes. Next, the bottom-up approach will be used for identifying the data records and data regions respectively. Figure 4 shows the main algorithm of the proposed system, which consists of five steps: (1) finding data items, (2) finding candidate data records, (3) identifying data records, (4) finding candidate data regions and (5) identifying the relevant data region.

Step 1: Finding data items

Figure 5 shows the algorithm for finding data items. There are three types of data item: (1) single value data items, (2) multiple value data items, and (3) optional value data items. Figure 6 shows the typical data records, which consist of a group of data items and some static texts. We mentioned that only the single value data items and static texts that exactly have the repetitive pattern. Therefore, this step aims to identify the list of single value data items and static texts. As the result, Figure 7 shows the result of finding data items in the DOM tree of the web page in Figure 6. We can see that all leaf nodes are grouped by their tree paths. Next, the groups, which have less than three members, will be removed.

Step 2: Finding candidate data records

In this step, the nodes that should be data records will be identified. This technique is based on the concept of the RSP (Repetitive Subject Pattern) [5] that each data item must belong to only one data record. We can find a data record of each data item by count a number of the data item nodes that have the same path in each level. Figure 8 shows counting of data items that have a tree path t_1 (*html/body/div/div/span(0)*). In this case, the nodes <*div*> whose one data item and its parent have three data items can be determined as a candidate data record. The algorithm of finding candidate data records is shown in Figure 8.

Step 3: Identifying data records

The set of candidate data records from Step 2 may include some irrelevant members. Then, this step aims to identify only the relevant data records, and the algorithm is shown in Figure 10. We have known that data record is a group of associated data items. Therefore, the data record node must contain at least two data item nodes otherwise it is not data record. After, all data record groups are collected; the small groups that have members less than three must be eliminated. Moreover, data records of the same parent node must belong to the same data record group. Therefore, if there are multiple groups that belong to the same parent node, the small group must be eliminated too.

Step 4: Finding candidate data regions

The algorithm of finding candidate data regions is shown in Figure 11. In this step, the identified data record groups of Step 3 will be classified into two types: (1) data section and (2) data region. We defined that the data section is a first level group of records; therefore, the nodes, which are the parent of two or more data records, can be determined as data sections. Otherwise, the first node, which is the parent of all records in the group, can be determined as data region. With this concept, each data record group must belong to only one data region, but it may have many data sections. However, if one data section contains all data records, then data region and data section must be the same node. For example, in Figure 3(a), the data region and the data section are the same node.

Step 5: Identifying the relevant data region

From step 4, the set of candidate data regions usually contains some extraneous regions such as navigational panels or advertisements. In this step, all data regions must be scored, and a relevant data region is the region, which has a maximum score. This algorithm based on the hypothesis that the relevant data region is the largest region. Our method for computing the score of each candidate data region consists of the following steps:

1. Let ***rg*** be a candidate data region.
2. Let ***rnum(rg)*** be a number of data records in ***rg***.
3. Let ***tp(rec)*** be a set of tree paths, which belong to data record ***rec***.
4. Let ***tnum(rg)*** be a number of tree paths, which belong to all data records in ***rg***:
$tnum(rg) = |\ t : t \in tp(rec), \forall\ rec \in rg|$
5. ***score(rg)*** = $rnum(rg) * tnum(rg)$

After the score is computed, the small candidate regions that have the value of *rnum* less than μ will be removed. (In the experiment, we had adjusted the value of μ as 3 for the best result.) Then, we assign the data region with the highest score as the relevant data region. However, all validated data regions are also returned by the system. Therefore, the users can select data records from other data regions, which match their needs.

Evaluation and Results

For the experimental evaluations, the datasets that used to test our system are the list-pages that were recently collected from the 115 well-known websites in years 2013 and 2014 such as Google, Yahoo, Bing, YouTube, Amazon, Ebay, Imdb, IeeeXplore, SpringerLink, ScienceDirect, and so on. These datasets were taken from two sources: 100 websites were taken from RSP dataset [5], and 15 websites were collected by our web loader component. There are ten web pages for each web site. In summary, the dataset consists of 1150 web pages. These websites were selected by distribution in several domains (e.g., general search engines, news sites, shopping sites, etc.), and three languages (English, Chinese and Thai). Moreover, all datasets and live demos of our system also are publicly available at http://buw.devtrainer.net/. Furthermore, the web pages in the dataset were classified by their structures into these following groups:

Group (A): This group consists of a set of list-pages that their data records are in one table, which has one record per row. See also Figure 3(a).

Group (B): This group consists of a set of list-pages that their data records are in a list of tables or blocks (*<div>*, *<li>*) that one table or block contains one record. See also Figure 3(b).

Group (C): This group consists of a set of list-pages that their data records are in table(s), which has more than each row contains multiple records. See also Figure 3(c) and Figure 3(d).

Efficiency of identifying the relevant data region

To further measure the reliability of the proposed method, we compare the accuracy of identifying the relevant data region task of the proposed method (BUW) and RSP [5]. However, RSP returns only one data region, but BUW returns a sorted list of data regions. The accuracy is an average of the percent of the correct data region, which are extracted from each input page. Table 1 shows the efficiency of the identifying the relevant data region task. In Table 1, the testing sites were grouped by their page structures. Experimental results show that the identifying the relevant data region task of BUW could achieve very high accuracy and performs better than RSP [5] in all groups.

Efficiency of extracting the relevant data records

The purpose of this test is to evaluate the efficiency of extracting data records task. We compared the proposed system (BUW) with RSP [5] and SDE [18]. Note that, SDE is open-source software that was implemented based on Partial Tree Alignment (DEPTA) [15], which is the well-known top-down wrapper. The measurement of the efficiency of data records extraction task is accuracy. The accuracy is an average of the percent of the correct data records, which are extracted from each input page. However, the results of RSP, BUW, and SDE are dissimilar. RSP returns only data records from only one relevant data region, but BUW and SDE return groups of data records from multiple data regions. Therefore, for calculating the accuracy of BUW and SDE, only the extracted data records of the main data region were selected. Table 2 and Table 3 present the experimental result of extracting data records by these three systems. However, the RSP can perform on two modes: RSP_{auto} is an automatic mode that all subject items were generated by the system, and RSP_{semi} is a semi-automatic mode that some subject items were adjusted for the best results. As the results of Table 2, BUW has more than 90% accuracy for all groups, and BUW outperforms SDE in all terms. From 115 web sites, Figure 12 shows that there are 83 websites that BUW has 100% accuracy, and there are only 17 websites that the BUW's accuracies are below that 90%.

In Table 2 and Table 3, we can see that RSP_{semi} performs very high accuracy in all cases because RSP_{semi} is semi-supervised wrapper and it can be adjusted for the best result. However, BUW, RSP_{auto} and SDE are fully unsupervised wrappers, which cannot be adjusted when they return the irrelevant results. Finally, for comparing with RSP_{auto} and SDE, the experimental results show that BUW outperforms RSP_{auto} and SDE.

Conclusions and future work

In this paper, we presented an information extraction (IE) system called BUW (Bottom-up Wrapper) that uses a bottom-up approach for automatically extracting data records from a list-page. BUW is based on assumptions that the data records in the same group must have a set of items, which are generated by the same template, and the repetitive pattern of these items can be used for identifying and clustering data records.

BUW is a fully unsupervised information extractor. It can automatically extract data records from the unseen web page. The experimental results show that BUW has very a high accuracy and outperforms SDE [18] and unsupervised mode of RSP [5]. The strength of our wrapper lies in the testing of Group(C), which the list of data records is shown in multiple columns (grid). For example, Figure 13 shows the example of the data records boundary detection result of BUW and SDE respectively. In this case, the structure of the first record is different from other records. But, the combine of second and third records is similar as the combine of fourth and fifth records. As the result, BUW returned the correct result, but SDE returned the incorrect result; it ignored the first record and identified the combine of second and third records as one record.

However, in this work, we focused only an algorithm for extracting data records and identifying the relevant data regions; the data items of the extracted data records are not extracted, aligned and labeled. Nevertheless, the existing alignment techniques of DEPTA [15] and RSP [5] can be applied for processing the extracted data records. However, this also requires some user's effort in the labeling process. In this respect, we have extended our work for automatic gathering meta-data in data items alignment process, and hope to report our findings soon.

Acknowledgements

This work was supported by the Thailand Research Fund (TRF) under the grant no: PHD/0280/2551 due to the Royal Golden Jubilee Ph.D. Program.

References

1. He B, Patel M, Zhang Z, Chang KC-C. Accessing the deep web. Commun ACM. 2007 May;50(5):94-101.
2. Chang CH, Kayed M, Girgis MR, Shaalan KF. A survey of web information extraction systems. IEEE Transactions on Knowledge and Data Engineering. 2006 Oct;18(10):1411-28.
3. Sleiman HA, Corchuelo R. A survey on region extractors from web documents. knowledge and data engineering, IEEE Transactions on. 2013;25(9):1960-81.
4. Thamviset W, Wongthanavasu S. Structured web information extraction using repetitive subject pattern. 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON2012). Thailand; 2012. p. 1-4.
5. Thamviset W, Wongthanavasu S. Information extraction for deep web using repetitive subject pattern. World Wide Web. 2014;17(5):1109-39.
6. Arocena GO, Mendelzon AO. WebOQL: restructuring documents, databases, and webs. Proceedings of the fourteenth international conference on data engineering [Internet]. Washington, DC, USA: IEEE Computer Society; 1998. p. 24-33. Available from: http://dl.acm.org/citation.cfm?id=645483.656208
7. Liu L, Pu C, Han W. XWRAP: an XML-enabled wrapper construction system for web information sources. Data Engineering, 2000 Proceedings 16th International Conference on. 2000. p. 611-21.
8. Adelberg B. NoDoSE - A tool for semi-automatically extracting structured and semistructured data from text documents. Proceedings of the 1998 ACM SIGMOD international conference on management of data [Internet]. New York, NY, USA: ACM; 1998. p. 283-94. Available from: http://doi.acm.org/10.1145/276304.276330

9. Ciravegna F, Dingli A, Wilks Y, Petrelli D. Adaptive information extraction for document annotation in amilcare. Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval [Internet]. New York, NY, USA: ACM; 2002. p. 451-451. Available from: http://doi.acm.org/10.1145/564376.564492
10. Yang S, Wang G, Han Y. Grubber: Allowing End-Users to Develop XML-Based Wrappers for web data sources. proceedings of the joint international conferences on advances in data and web management [Internet]. Berlin, Heidelberg: Springer-Verlag; 2009. p. 647-52. Available from: http://dx.doi.org/10.1007/978-3-642-00672-2_65
11. Chang C-H, Kuo S-C. OLERA: Semisupervised web-data extraction with visual support. IEEE Intelligent Systems. 2004 Nov;19(6):56-64.
12. Chang C-H, Lui S-C. IEPAD: information extraction based on pattern discovery. Proceedings of the 10th international conference on world wide web [Internet]. New York, USA: ACM; 2001. p. 681-8. Available from: http://doi.acm.org/10.1145/371920.372182
13. Arasu A, Garcia-Molina H. Extracting structured data from web pages. Proceedings of the 2003 ACM SIGMOD international conference on management of data [Internet]. New York, NY, USA: ACM; 2003. p. 337-48. Available from: http://doi.acm.org/10.1145/872757.872799
14. Liu B, Grossman R, Zhai Y. Mining data records in web pages. Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining [Internet]. New York, NY, USA: ACM; 2003. p. 601-6. Available from: http://doi.acm.org/10.1145/956750.956826
15. Zhai Y, Liu B. Structured Data extraction from the web based on partial tree alignment. IEEE Transactions on Knowledge and Data Engineering. 2006 Dec;18(12):1614-28.
16. Zhao H, Meng W, Wu Z, Raghavan V, Yu C. Fully automatic wrapper generation for search engines. Proceedings of the 14th international conference on world wide web [Internet]. New York, NY, USA: ACM; 2005. p. 66-75. Available from: http://doi.acm.org/10.1145/1060745.1060760
17. Liu W, Meng X, Meng W. ViDE: A vision-based approach for deep web data extraction. IEEE Transactions on Knowledge and Data Engineering. 2010 Mar;22(3):447-60.
18. Dewanto S. Structured Data Extractor - An implementation of data extraction based on partial tree alignment (DEPTA) [Internet]. 2012 [cited 2013 Nov 21]. Available from: http://seagatesoft.blogspot.com/

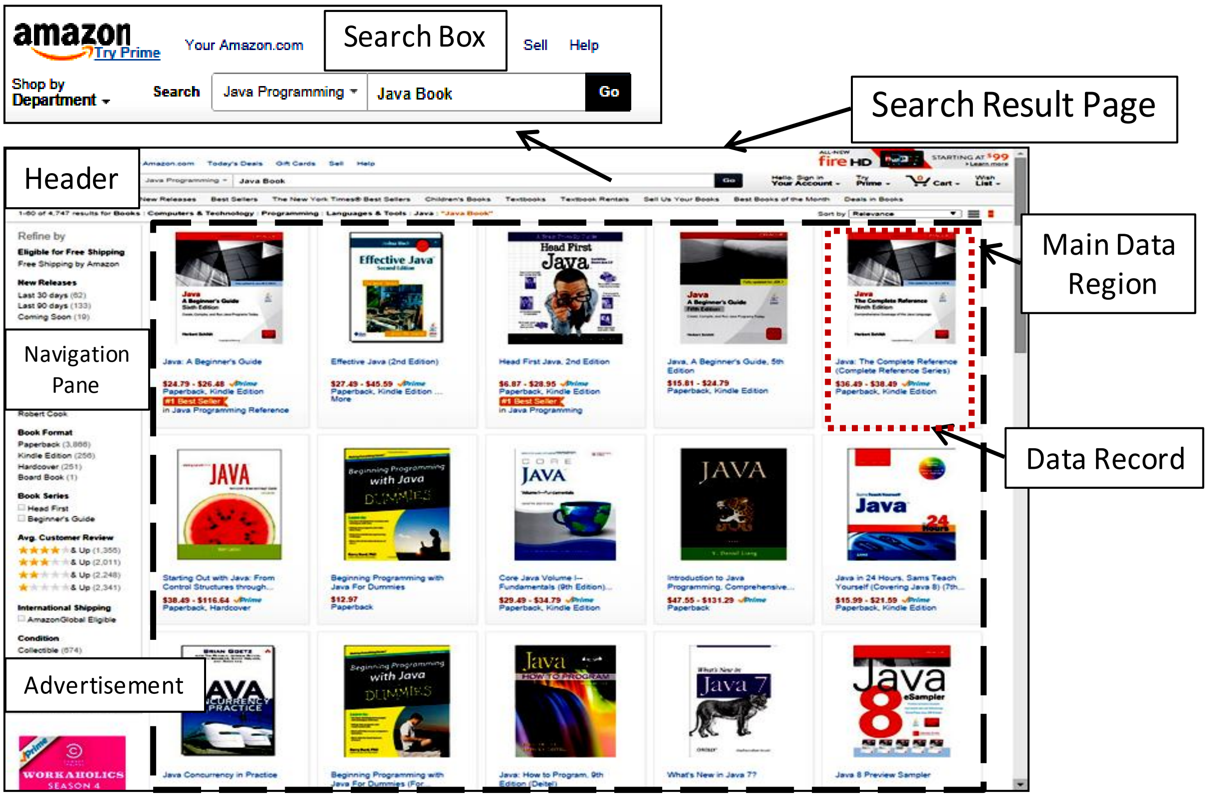
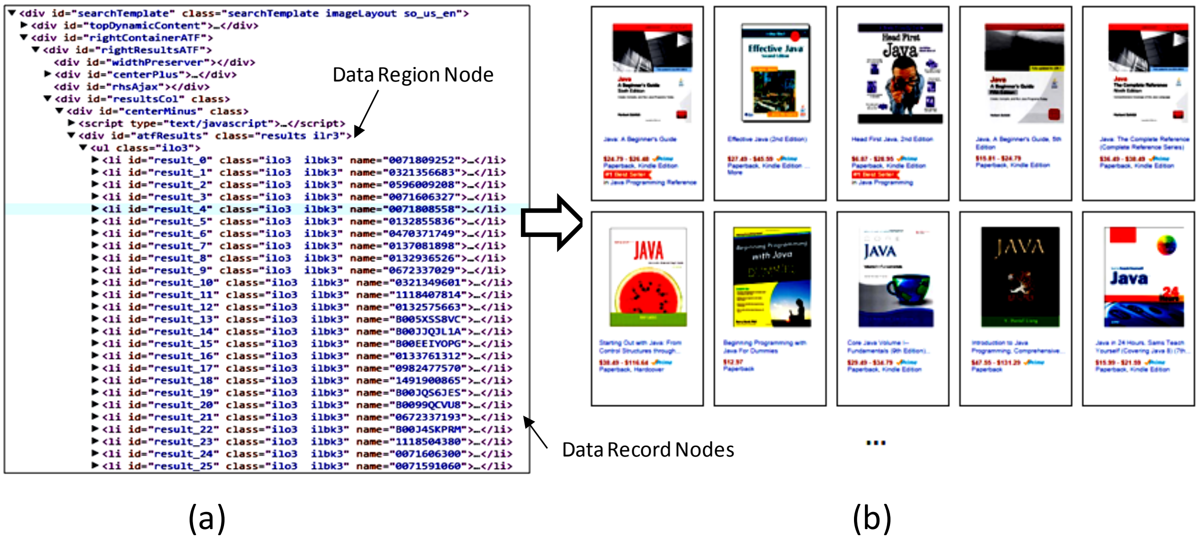


Figure 1 Example List-Page from Amazon.com



```
▼<div id="searchTemplate" class="searchTemplate imageLayout so_us_en">
  ▶<div id="topDynamicContent">…</div>
  ▼<div id="rightContainerATF">
    ▼<div id="rightResultsATF">
       <div id="widthPreserver"></div>
      ▶<div id="centerPlus">…</div>
       <div id="rhsAjax"></div>
      ▼<div id="resultsCol" class>
        ▼<div id="centerMinus" class>
          ▶<script type="text/javascript">…</script>
          ▼<div id="atfResults" class="results ilr3">
            ▼<ul class="ilo3">
              ▶<li id="result_0" class="ilo3  ilbk3" name="0071809252">…</li>
              ▶<li id="result_1" class="ilo3  ilbk3" name="0321356683">…</li>
              ▶<li id="result_2" class="ilo3  ilbk3" name="0596009208">…</li>
              ▶<li id="result_3" class="ilo3  ilbk3" name="0071606327">…</li>
              ▶<li id="result_4" class="ilo3  ilbk3" name="0071808558">…</li>
              ▶<li id="result_5" class="ilo3  ilbk3" name="0132855836">…</li>
              ▶<li id="result_6" class="ilo3  ilbk3" name="0470371749">…</li>
              ▶<li id="result_7" class="ilo3  ilbk3" name="0137081898">…</li>
              ▶<li id="result_8" class="ilo3  ilbk3" name="0132936526">…</li>
              ▶<li id="result_9" class="ilo3  ilbk3" name="0672337029">…</li>
              ▶<li id="result_10" class="ilo3  ilbk3" name="0321349601">…</li>
              ▶<li id="result_11" class="ilo3  ilbk3" name="1118407814">…</li>
              ▶<li id="result_12" class="ilo3  ilbk3" name="0132575663">…</li>
              ▶<li id="result_13" class="ilo3  ilbk3" name="B005XSS8VC">…</li>
              ▶<li id="result_14" class="ilo3  ilbk3" name="B00JJQJL1A">…</li>
              ▶<li id="result_15" class="ilo3  ilbk3" name="B00EEIYOPG">…</li>
              ▶<li id="result_16" class="ilo3  ilbk3" name="0133761312">…</li>
              ▶<li id="result_17" class="ilo3  ilbk3" name="0982477570">…</li>
              ▶<li id="result_18" class="ilo3  ilbk3" name="1491900865">…</li>
              ▶<li id="result_19" class="ilo3  ilbk3" name="B00JQS6JES">…</li>
              ▶<li id="result_20" class="ilo3  ilbk3" name="B0099QCVU8">…</li>
              ▶<li id="result_21" class="ilo3  ilbk3" name="0672337193">…</li>
              ▶<li id="result_22" class="ilo3  ilbk3" name="B00J4SKPRM">…</li>
              ▶<li id="result_23" class="ilo3  ilbk3" name="1118504380">…</li>
              ▶<li id="result_24" class="ilo3  ilbk3" name="0071606300">…</li>
              ▶<li id="result_25" class="ilo3  ilbk3" name="0071591060">…</li>
```

Figure 2 (a) The DOM tree of the page of Figure 1, (b) Extracted data records

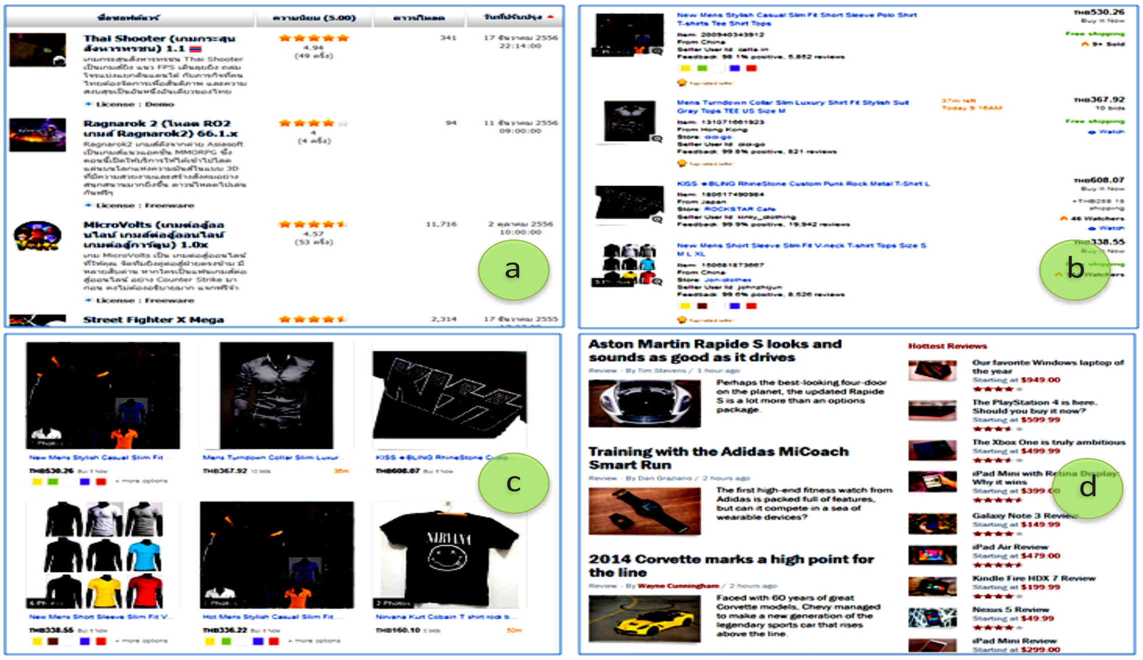


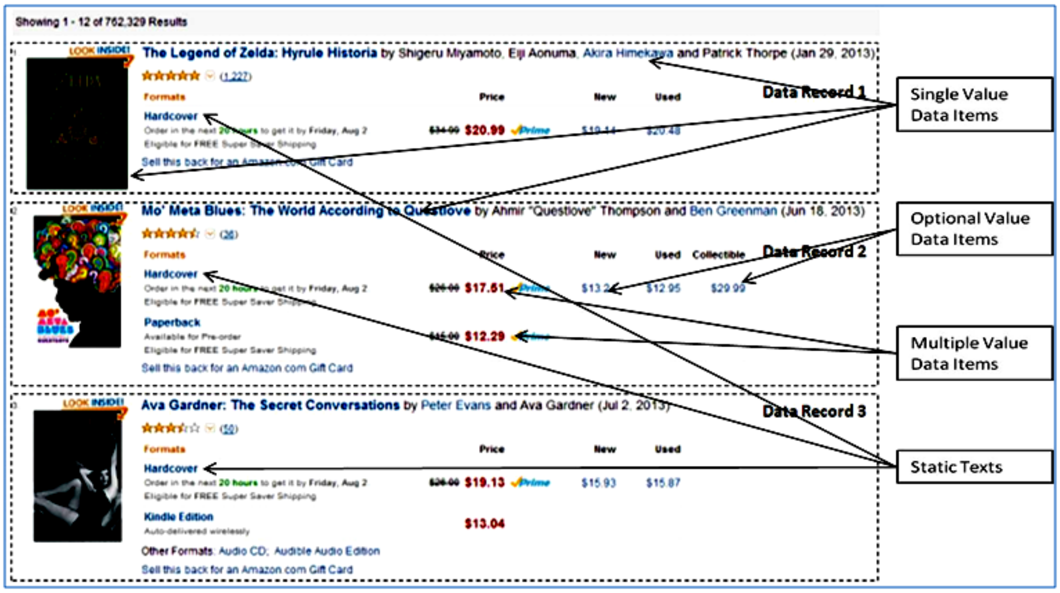
Figure 3 Layouts of the web data records: (a) one table with five columns and one record per row, (b) list of tables with one table for one record, (c) tables with three records per row, (d) two groups of data records with different templates.

```
procedure Wrap(dom){
    T  = findDataItems(dom);
    cr = findCandidateRecords(T);
    DG = identifyDataRecords(T,cr);
    RG = findCandidateRegions(DG);
    identifyMainDataRegion(RG);
}
```

Figure 4 Algorithm of the proposed system: Bottom-Up wrapper (BUW).

```
function findDataItems(dom){
  L  = getAllLeafNodes(dom);
  T  = SetOfTreePath();
  di = MapOfNodesSet();
  foreach( x in L){
     t = path(x);
     T <-- t;      // put t into T
     di[t] <-- x;  // put x into di[t]
  }
  foreach(t in T){
     if(countOf(di[t])<3){
       remove t from T;
     }
  }
  return T;
}
```

Figure 5 Algorithm of step 1: finding data items.



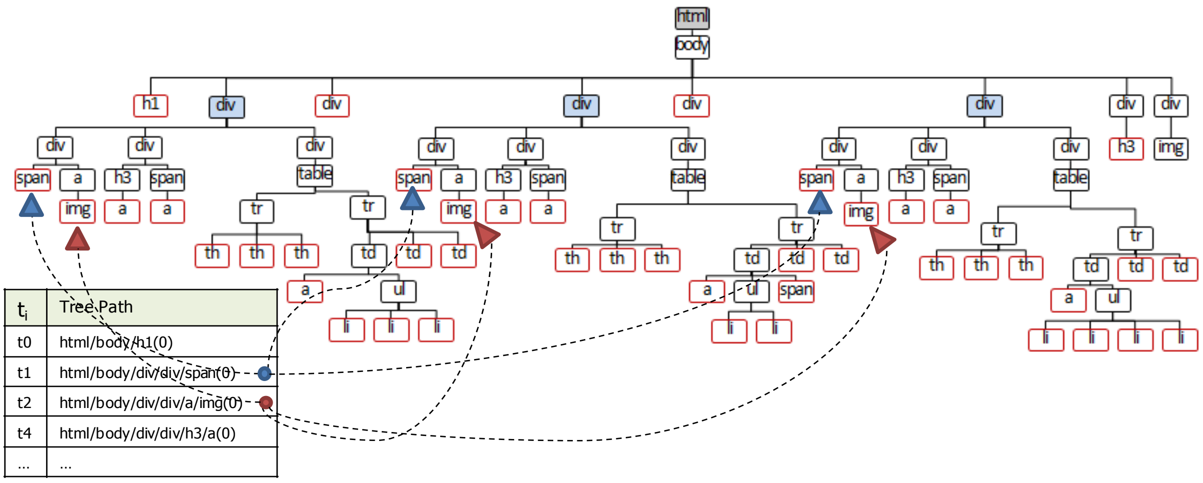
Showing 1 - 12 of 762,329 Results

The Legend of Zelda: Hyrule Historia by Shigeru Miyamoto, Eiji Aonuma, Akira Himekawa and Patrick Thorpe (Jan 29, 2013)
(1,227)
Formats | Price | New | Used
Hardcover | ~~$34.99~~ $20.99 Prime | $19.14 | $20.48
Order in the next 20 hours to get it by Friday, Aug 2
Eligible for FREE Super Saver Shipping
Sell this back for an Amazon.com Gift Card

Mo' Meta Blues: The World According to Questlove by Ahmir "Questlove" Thompson and Ben Greenman (Jun 18, 2013)
(26)
Formats | Price | New | Used | Collectible
Hardcover | ~~$26.00~~ $17.51 Prime | $13.24 | $12.95 | $29.99
Order in the next 20 hours to get it by Friday, Aug 2
Eligible for FREE Super Saver Shipping
Paperback | ~~$15.00~~ $12.29 Prime
Available for Pre-order
Eligible for FREE Super Saver Shipping
Sell this back for an Amazon.com Gift Card

Ava Gardner: The Secret Conversations by Peter Evans and Ava Gardner (Jul 2, 2013)
(50)
Formats | Price | New | Used
Hardcover | ~~$26.00~~ $19.13 Prime | $15.93 | $15.87
Order in the next 20 hours to get it by Friday, Aug 2
Eligible for FREE Super Saver Shipping
Kindle Edition | $13.04
Auto-delivered wirelessly
Other Formats: Audio CD; Audible Audio Edition
Sell this back for an Amazon.com Gift Card

Figure 6 Type of Data Items.



t_i	Tree Path
t0	html/body/h1(0)
t1	html/body/div/div/span(0)
t2	html/body/div/div/a/img(0)
t4	html/body/div/div/h3/a(0)
…	…

Figure 7 Finding data items in the DOM tree of the web page in Figure 6.

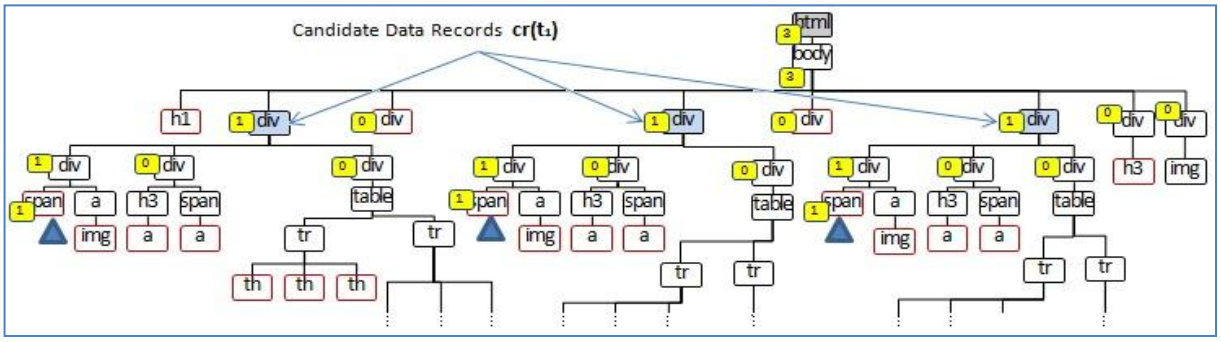


Figure 8 Counting of data item nodes $di(t_1)$ in the DOM tree.

```
// T is a set of leaf nodes.
function findCandidateRecords(T){
  clear values in tcount;
  foreach(t in T){
    foreach( x in di[t]){
      bottomUpCount(t,x);
    }
  }
  cr = MapOfSet();
  foreach(t in T){
    cr[t] = Set();
    foreach(a in getNodesOfPath(t)){
      if( tcount[t,a]==1
          AND tcount[t,a.parent]>1)
        cr[t] <-- a; // put into cr[t]
    }
  }
  return cr;
}

procedure bottomUpCount(t,a){
   tcount[t,a]++;
   if(a.parent != ROOT )
     bottomUpCount(t,a.parent);
}
```

Figure 9 Algorithm of step 2: finding candidate data records.

```
function identifyDataRecords(T,cr){
  CR=Set(); // Set of Candidate Records
  foreach(t in T){
    foreach( rec in cr[t]){
      CR <-- rec;  //put into CR
      tp[rec] <-- t;
    }
  }
  // DR is Set of Data Records
  DR=Set();
  foreach(rec in CR){
      tx[rec] = getAllLeafNodes(rec);
      cp = countOf(tp[rec]);
      cx = countOf(tx[rec]);
      if(cp>1 && cx>1)
        DR <-- rec;  //put into DR
  }
  // DG is Map of Data Record Groups
  DG=MapofSet();
  foreach(rec in DR){
     t = path(rec);
     dtp = tp[DG[t]];
     if( dtp==NULL
       OR IntersectOf(tp(rec),dtp)>1){
        DG[t] <-- rec;
     }
  }
  return DG;
}
```

Figure 10 Algorithm of step 3: Identifying data records.

```
function findCandidateRegions(DG){
  RG = SetOfNode();
  foreach(dg in DG ){
    bottomUpCount(dg);
    n = size(dg);
    foreach(dr in dg) {
     # Data region
       rg is the first parent node,
       which has a node count = n.
     # Data section
       sg is the first parent node,
       which has a node count > 1.

       RG <-- rg;
       SG[RG] <-- sg;
    }
  }
  return RG;
}
```

Figure 11 Algorithm of step 4: Finding candidate data regions.

Table 1 Experimental results of identifying the relevant data region compared with the RSP

Structure Groups	#Sites	#Pages	Accuracy	
			BUW	RSP
Group A	38	380	98.03%	97.63%
Group B	69	690	98.29%	86.52%
Group C	8	80	100.00%	97.50%
All Sites	115	1150	**98.32%**	90.96%

Table 2 Experimental results of extracting data records group by page structures

	#Sites	#Pages	#Actual Records	Accuracy			
Structure Groups				BUW	RSP$_{semi}$	RSP$_{auto}$	SDE
Group A	38	380	7151	90.25%	99.36%	96.98%	69.04%
Group B	69	690	12803	96.40%	97.60%	84.81%	56.87%
Group C	8	80	2334	**99.88%**	99.81%	98.44%	54.13%
All Sites	115	1150	22288	94.43%	98.34%	89.78%	60.71%

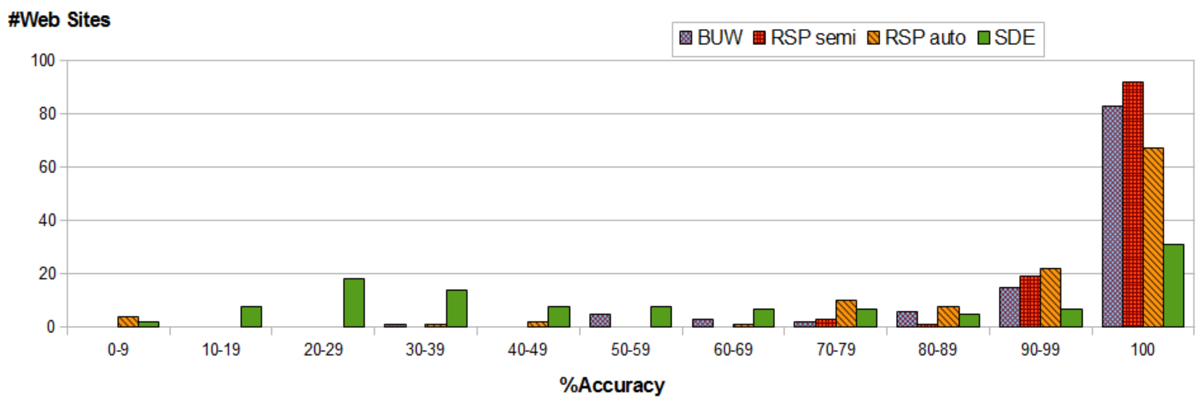


Figure 12 Evaluation results of extracting the relevant data records for all sites.

Table 3 Experimental results of extracting data records group by features

Categories	#Web Sites	#Pages	#Actual	Accuracy			
			Records	BUW	RSP$_{semi}$	RSP$_{auto}$	SDE
Languages							
English	91	910	18235	95.45%	98.86%	89.98%	58.67%
Chinese	12	120	2047	93.73%	99.10%	94.85%	65.13%
Thai	12	120	2006	90.54%	96.81%	85.95%	71.72%
Domains							
Book	12	120	1906	100.00%	99.96%	96.66%	75.82%
Shopping	17	170	5175	94.80%	98.03%	86.25%	46.69%
General	10	100	1585	84.06%	91.76%	89.76%	62.29%
News	10	100	1610	96.36%	95.37%	85.37%	50.77%
Article	6	60	1228	88.99%	100.00%	100.00%	60.95%
Software	21	210	3091	96.15%	99.92%	95.63%	66.14%
Media	15	150	3535	96.67%	99.67%	95.00%	66.19%
Research	12	120	1708	97.42%	99.02%	71.93%	63.47%
Blog	6	60	800	98.33%	98.33%	91.83%	42.76%
Jobs	6	60	1650	83.64%	100.00%	83.27%	63.57%

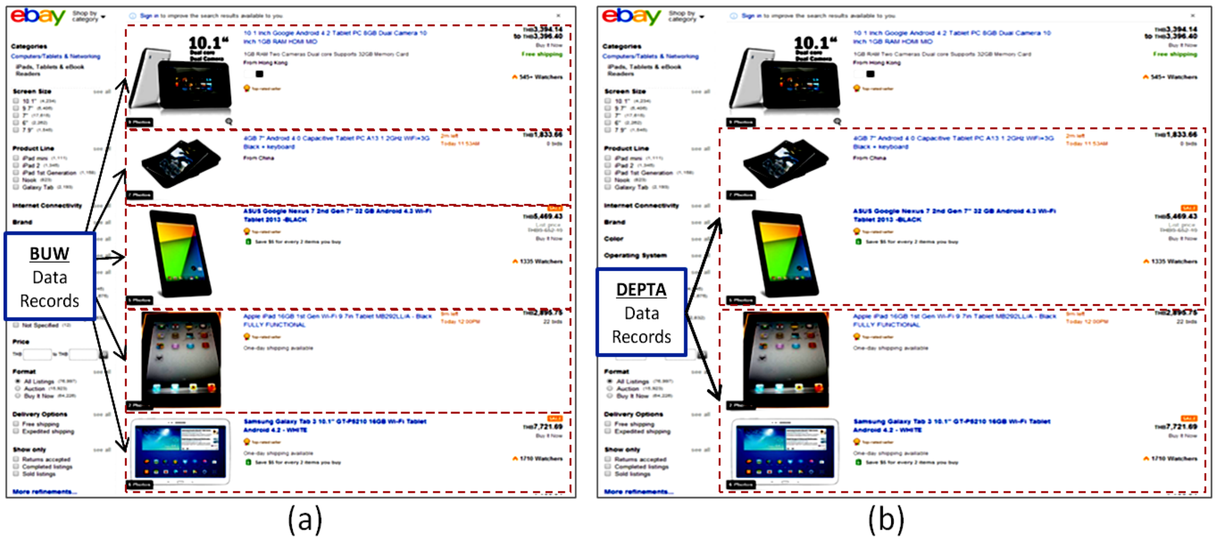


Figure 13 Data records boundary detection result of BUW and DEPTA (SDE) from the website Ebay:com.