Information Sciences 328 (2016) 528–545

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Specification and discovery of web patterns: a graph grammar approach

Amin Roudaki[a], Jun Kong[a,*], Kang Zhang[b]

[a] North Dakota State University, 1340 Administration Ave, Fargo, ND 58102, USA
[b] The University of Texas at Dallas, 800 W Campbell Rd, Richardson, TX 75080, USA

ARTICLE INFO

Article history:
Received 16 September 2014
Revised 26 June 2015
Accepted 31 August 2015
Available online 8 September 2015

Keywords:
Web patterns
Spatial graph grammar
Graph grammar induction

ABSTRACT

Finding useful information from the Web becomes increasingly difficult as the volume of Web data rapidly grows. To facilitate effective Web browsing, Web designers usually display the same type of information with a consistent layout (referred to as a Web pattern). Discovering Web patterns can benefit many applications, such as extracting structured data. This paper presents a generic framework for discovering Web patterns and recognizing their instances (i.e., structured data) based on graph grammars. In our framework, a Web pattern is visually yet formally specified as a graph grammar, which is automatically induced through a grammar induction engine. The grammar induction engine is featured by converting the problem of (2-dimensional) graph grammar induction to (1-dimensional) string induction. Based on the induced pattern, matching instances are recognized from Web pages through a graph parsing process. We have evaluated the framework on twenty-one e-commerce Web sites. The evaluation results are promising with a high F1-score.

1. Introduction

Finding useful information from the Web becomes increasingly difficult as the volume of Web data rapidly grows. HTML, being the dominant language for the Web, describes the appearance of a Web page, but does not explicitly reveal the underlying information organization. Researchers have attempted to analyze HTML DOM structures and extract useful information. However, HTML has been used diversely by different Web designers. For example, tables can be used with different purposes, e.g., presenting tabular data or dividing the space into grids. The diversity of HTML usages makes it challenging to analyze Web data using HTML source codes. The complexity of DOM structures further complicates the problem of the Web data analysis. For example, the DOM structure of the Google homepage alone includes approximately 140 HTML tags.

Recently, layout-based analysis [13,26,46,61] received more attention for analyzing and extracting Web data. Based on design guidelines in Human Computer Interaction (HCI), Web designers usually organize Web information in a consistent layout among different pages [45,26,56]. As a result, Web pages that include similar information are presented in a consistent layout even though their HTML codes may be different. Therefore, layout-based analysis addresses the diversity issue to a certain degree. Existing layout-based approaches are, however, typically optimized for a specific domain, such as news extraction [13].

This paper presents a generic framework for discovering Web patterns and recognizing their instances based on graph grammars. In our framework, Web patterns reflect the commonly accepted practices in Web design and are used to display the same type of information. In other words, a Web pattern is a two-dimensional structure that depicts a typical layout. Graph grammars,

* Corresponding author. Tel.: +7012318179.
E-mail addresses: amin.roudaki@ndsu.edu (A. Roudaki), jun.kong@ndsu.edu (J. Kong), kzhang@utdallas.edu (K. Zhang).

http://dx.doi.org/10.1016/j.ins.2015.08.052
0020-0255/© 2015 Elsevier Inc. All rights reserved.

which visually yet formally model structures and concepts in a 2D fashion [42], provide a natural means to define Web patterns. More specifically, a graph grammar is made of a set of grammatical rules (called *productions*). Each production defines local spatial relations among relevant information objects while the complete graph grammar hierarchically glues those local spatial relations together. By formalizing a Web pattern as a graph grammar, recognizing the instants of a Web pattern is implemented as a graph parsing process in a bottom-up fashion.

Since it is time consuming to manually summarize a Web pattern as a graph grammar, our framework provides an automatic grammar induction engine to support the scalability and applicability. Briefly, the grammar induction engine automatically extracts the most common structure (i.e., a Web pattern) from sample Web pages and accordingly represents the recognized structure hierarchically as a graph grammar. The lack of domain knowledge in grammar induction may make an induced grammar hard to understand, partly due to program-generated names. In our framework, grammar induction is complemented with a sample-based grammar editor. The grammar editor provides a graphical user interface that allows the user to create or revise a graph grammar by directly manipulating information objects in the screenshot of a Web page.

In summary, this paper presents a grammar-based approach to automatically discover a Web pattern and recognize instances of the discovered pattern. Our approach is featured by a graph generation process that creates a spatial graph from a Web page. The graph generation reduces the complexity of the original Web page while keeping important spatial information. A Web pattern is automatically discovered by searching for the most important repetitive structure from spatial graphs through a grammar induction algorithm. Distinct from existing approaches, our algorithm converts the problem of (2-dimensional) graph grammar induction to (1-dimensional) string induction by taking advantage of spatial information.

This work extends our previous conference publication [43] by introducing a novel grammar induction algorithm. To our knowledge, the proposed framework is the first to use a grammar induction method to identify Web patterns and recognize their instances from the Web. We have evaluated our method on 21 Web sites by validating the instances of a Web pattern. The results are promising, and the performance of our approach measured in F1-Score is better than that of the benchmark approach MDR [32] (i.e., 97.16% of F1-Score of our approach versus 69.19% of MRD).

2. Related work

Extracting useful information from Web documents has been widely studied in recent years [31,39,44,48]. Based on the theoretical foundation, we can classify data extraction techniques into wrapper-based approaches and statistical model based ones. The Wrapper-based approaches, such as [17,19,38], induce a set of rules to define the knowledge of data extraction. Statistical models (such as Support Vector Machines [30] or Hidden Markov Models [21,47]) discover repetitive structures based on statistical analysis. Distinct from wrappers or statistical models, our approach applies the state-of-the-art grammar technology (i.e. the Spatial Graph Grammar [25]) as the theoretical foundation. Based on the Spatial Graph Grammar, the knowledge of data extraction is visually defined through grammatical rules, which are decoupled from the data extraction process.

Based on DOM structures, several approaches [22,28,37] use the machine learning technique to automatically derive a wrapper based on a set of manually labeled training data. In a training set, information pieces of interest are manually tagged with different labels, such as *title* or *price*. Based on the labels, those approaches [22,28,37] consider an HTML Web page as a sequence of characters and search for common delimiter strings between labeled information pieces. Those common delimiter strings are formalized as extraction rules in a wrapper for data extraction. Though the above approaches apply different technologies to derive a wrapper, they all require a set of training data, which are manually labeled by human experts. In order to minimize the effort of preparing a training set, Dalvi et al. recently proposed a generic framework for supervised wrapper induction based on automatically obtained noisy training data [17]. Without requiring a training set, Amin and Jamil [4] extracted from a symbol list of an HTML page a commonly occurring pattern of the highest length and super-maximal repeats. The pattern is converted to a regular expression, which is subsequently used to extract the record level data items. Different from the above approach, the regular expression in our approach implies spatial relations among relevant objects.

Some approaches [6,16] automatically derive a template from sample Web pages and use the extracted template to discover structured records. In general, Web pages that are generated from the same template have the same DOM structure but different actual contents. The multi-string alignment [16], statistical techniques [6] or tree matching and alignment [36] was used to extract structured records from a set of Web pages that follow the same template. TTAG [14] traversed the DOM structures of sample pages from the root to leaf nodes in a top down fashion and used the dynamic programming to compare and extract repetitive patterns. Reis et al. [41] proposed a *tree edit distance* method to derive a template underlying sample pages and used the derived template for data extraction. Recently, Sleiman and Corchuelo proposed an efficient simple multi-string alignment algorithm for recognizing a template and its variable contents [49]. The above approaches [6,14,16,41,49] do not require manually labeled data, which greatly reduces the manual effort in the data extraction process. However, they require that Web pages being analyzed must follow the same template.

MDR [32] generated an HTML tag tree based on table and form related tags, e.g., *table*, *form*, *tr*, *td*, and etc. Then, MDR used the string comparison technique to divide a Web page into different regions, in which data records are identified by calculating the similarity between tag strings. Zhai et al. [58] proposed DEPTA that improved MDR and supported partial tree alignment. Therefore, DEPTA accommodates variations among different Web pages and improves the applicability. Miao et al. identified the key limitation of MDR as the pairwise comparison of consecutive segments [35]. In order to address this issue, they conducted data extraction by comparing a pair of tag path occurrence patterns [35]. On the basis of MDR, Song et al. proposed the MiBAT algorithm (Mining data records based on Anchor Trees), which automatically extracts data records including user-generated content

[51]. Raeymaekers and Bruynooghe combined the tree-based approach with the string-based approach for supporting sub node extraction [40]. Zhai et al. [59] rendered a Web page and allowed users to select information objects in the screen shot to define a data pattern. Laber et al. [29] proposed a heuristic algorithm for extracting news articles based on DOM tree structure. This approach used some statistical analysis to analyze the DOM tree elements and identify the relevant information objects. Alvarez et al. [3] identified the data region of interest in a Web page and then used the clustering method, which grouped similar subtrees in a DOM tree, to extract structured records. Bing et al. [9] proposed the Record Segmentation Tree (RST) and designed efficient search pruning strategies to identify records. The Skoga framework [10] conducted a global analysis on the DOM structure to detect data records based on a DOM structure-knowledge-driven detection model, which consists of background knowledge and statistical knowledge. All of the above discussed methods use HTML DOM structure as the main source for data extraction. Instead, our approach implements data extraction based on the layout. The visual analysis can address the issues of complexity and diverse usages of HTML DOM structures to a certain degree. By actually rendering a Web page, our approach supports dynamic information objects which are generated at run time.

Recently, visual perception techniques have been applied to extract structured data. These approaches [13,46,61] basically calculate the visual similarity among different Web pages to group semantically related information. ViPER [46] used the visual information to separate and weight different data regions. Zheng et al. [61] introduced a *template independent* system to identify news articles based on visual consistency. This approach summarizes a set of visual features to present news stories and automatically generates a template-independent wrapper based on those visual features. Chen et al. [13] proposed a system for extracting news stories based on visual perception. First, it identifies the areas that contain news stories based on the content functionality, space continuity, and formatting continuity. After detecting the news areas, news stories are extracted based on the position, format and semantics. The above two approaches [13,61] are limited to extract news stories, and are not applicable to other domains. Instead, our approach is applicable to different domains. Furthermore, the usage of HTML structures in our approach makes it efficient to recognize the boundary between different blocks. ViDE [33] utilized visual features to extract data records, but this approach relied on a heuristic page segmentation algorithm [11], which was dependent on the HTML encodings. You et al. [55] integrated a vision-based segmentation algorithm (i.e., VIPS [11]) with a DOM-based analysis to extract text blocks, which were annotated based on a tree-structured conditional random fields model. This approach is applied to conference information extraction. Anderson and Hong [5] defined the visual block model according to the layout of a page and calculated the similarity to extract data records. The SILA approach [18] introduced a Positional Document Object Model (i.e., PDOM) that defined the spatial arrangement among digital objects. Similar to the spatial graph, PDOM is independent of the internal HTML encodings and simplifies the original Web page. Different from SILA, our approach uses a graph grammar to extract structured records rather than calculating the visual similarity. Kong et al. [26] combined the image processing technique with graph grammars to extract useful information. However, this approach needs a training set for recognizing atomic information objects.

The Hybrid methods consider both the DOM structure and visual perception. ViNTs [57] automatically recognized different content shapes based on the visual position of information objects. Afterward, a wrapper is generated based on an HTML structure, which represents each shape. This approach still extracts information based on HTML DOM structures, though the wrapper is derived from the visual analysis. Instead, our approach specifies extraction rules from both the layout and the DOM structure. ViNTs is limited to search results, while our approach is general to different domains.

Our work is also related to the researches that divide a Web page into different blocks. Recently, Sleiman and Corchuelo provided a comprehensive review on different approaches for region extraction [48]. Kovacevic et al. [27] used visual information to build up an M-Tree, and defined heuristic rules to recognize common page areas, such as header, left and right menu, footer, etc. Some approaches analyze the importance of different blocks within a single web page, such as a learning based approach [50] or random walk model based approaches [53,54]. In these approaches, visual features, such as width, height, or position, are used to recognize and analyze information blocks. The combination of natural language processing and DOM analysis [24] or statistical analysis [23] was proposed to recognize the main content in a Web page. The TWWF approach [60] used the DOM structure to divide a Web page. Ahmadi and Kong [1,2] used a hybrid approach for the block recognition. Xiao et al. [52] used a vision-based analysis to divide a Web page into small blocks that are then merged to form leaf blocks based on the screen size of a mobile device. Those leaf blocks are organized as a SubPage-tree. The above technologies can be potentially integrated with our approach to improve the performance by eliminating noises.

In summary, our approach is the first attempt to apply spatial properties to induce graph grammars. By abstracting a concrete Web page as a spatial graph, the pattern discovery can be considered as finding repetitive structures in a graph. Since a Web pattern is in general formed hierarchically, it is natural to define a Web pattern through a graph grammar that provides a solid theoretic foundation for two-dimensional reasoning. Graph grammar is particularly powerful in capturing the variation among instances of a Web pattern by applying a production recursively. The grammar induction makes our approach applicable to different domains. Once a Web pattern (i.e., a graph grammar) is identified automatically, it can be applied to different Web pages of the same pattern to recover structured information. Our approach decouples the specification of a Web pattern from data extraction. Consequently, our framework can efficiently recognize and validate evolving patterns by updating the graph grammar without changing source codes. Decoupling pattern specification from data extraction makes our approach robust to handle pattern evolution and variations among different Web pages. We summarized existing approaches mainly in two categories. The first category analyzes the HTML DOM documents to identify repetitive structures. One of the most popular methods is MDR [32]. The second category uses visual clues by rendering web pages, such as ViNTs [57] that analyzes the shape of records displayed on a page. Our approach uses both DOM structure and visual perception, and is applicable to different domains. Table 1 compares our approach with closely related approaches.

Table 1
The summary of major techniques.

Method	Description
Our approach	DOM structure + visual perception based on graph grammar
ViNTs [57] (applicable for search engines)	DOM structure + visual perception
[35]	DOM structure (tag path occurrence based on MDR)
[10]	Knowledge based DOM structure analysis
DEPTA [58]	DOM structure (improved MDR with partial tree alignment)
ViPER [46]	Visual perception (edit distance to calculate similarity)
FiVaTech [36]	DOM structure (tree matching, tree alignment and mining)
[3]	DOM structure using the clustering technique

Table 2
The summary of Web patterns.

	Web site name	Pattern 1	Pattern 2		Web site	Pattern 1	Pattern 2
1	http://shopping.yahoo.com/	×		22	http://www.epicurious.com	×	
2	http://scistore.cambridgesoft.com/	×		23	http://www.cooking.com	×	×
3	http://shop.lycos.com	×		24	http://www.asiatravel.com		×
4	http://www.barnesandnoble.com	×	×	25	http://www.godaddy.com		×
5	http://www.borders.com	×		26	http://www.radioshack.com	×	
6	http://www.circuitcity.com	×	×	27	http://www.dell.com	×	×
7	http://www.compusa.com	×	×	28	http://www.macys.com		×
8	http://www.drugstore.com	×		29	http://www.buyflowersonline.com		×
9	http://www.ebay.com	×	×	30	http://www.buy.com	×	×
10	http://www.etoys.com		×	31	http://www.bestbuy.com	×	
11	http://www.kidsfootlocker.com		×	32	http://www.deals2buy.com	×	
12	http://www.kodak.com		×	33	http://www.6pm.com		×
13	http://www.newegg.com	×	×	34	http://www.bigdeal.com		×
14	http://www.nothingbutsoftware.com	×	×	35	http://www.shopzilla.com	×	
15	http://www.overstock.com		×	36	http://www.haggle.com		×
16	http://www.powells.com	×		37	http://www.shop.com		×
17	http://www.softwareoutlet.com		×	38	http://www.alibaba.com	×	
18	http://www.ubid.com		×	39	http://www.pricegrabber.com	×	
19	http://www.amazon.com		×	40	http://www.shopnbc.com		×
20	http://www.shopping.hp.com	×		41	http://www.shopstyle.com		×
21	http://www.qualityinks.com/		×	42	http://www.target.com		×

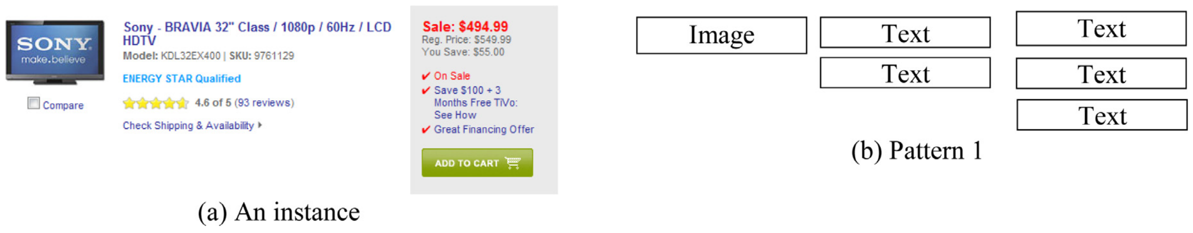


Fig. 1. Pattern 1.

3. Web patterns

Our approach assumes a consistent layout across Web pages, referred to as a Web pattern. To justify our assumption, we investigated 42 E-commerce Web sites, which included well-structured product information, and manually summarized two patterns as shown in Table 2.

Pattern 1. As shown in Fig. 1, Pattern 1 presents product information from left to right. A product picture is displayed on the left side, and other descriptive textual information is presented on the right side. The descriptive information may include several lines of texts, which are displayed from top-down and left to right.

Pattern 2. As shown in Fig. 2, Pattern 2 presents information vertically. Either a product title or a product image is presented on the top, followed by a list of textual lines.

In summary, each pattern prescribes the layout at a high level and records the essential spatial properties. Automatically (through a grammar induction as discussed in Section 7) or manually designed graph grammars can form a library that indicates commonly used Web patterns. A graph grammar in the library can be applied to a set of Web pages that followed the defined

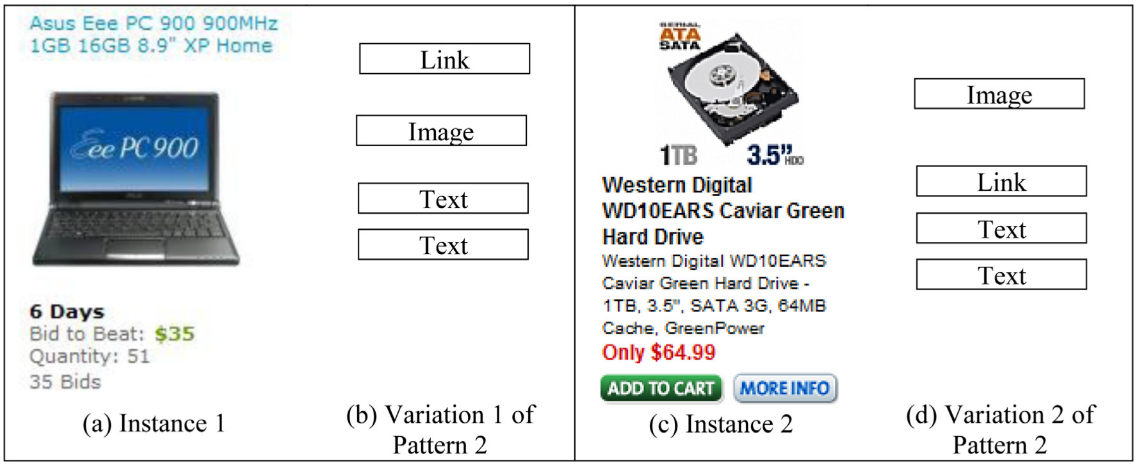

Asus Eee PC 900 900MHz 1GB 16GB 8.9" XP Home
6 Days
Bid to Beat: $35
Quantity: 51
35 Bids
(a) Instance 1
Link
Image
Text
Text
(b) Variation 1 of Pattern 2
1TB 3.5"
Western Digital WD10EARS Caviar Green Hard Drive
Western Digital WD10EARS Caviar Green Hard Drive - 1TB, 3.5", SATA 3G, 64MB Cache, GreenPower
Only $64.99
ADD TO CART
MORE INFO
(c) Instance 2
Image
Link
Text
Text
(d) Variation 2 of Pattern 2

Fig. 2. Pattern 2.

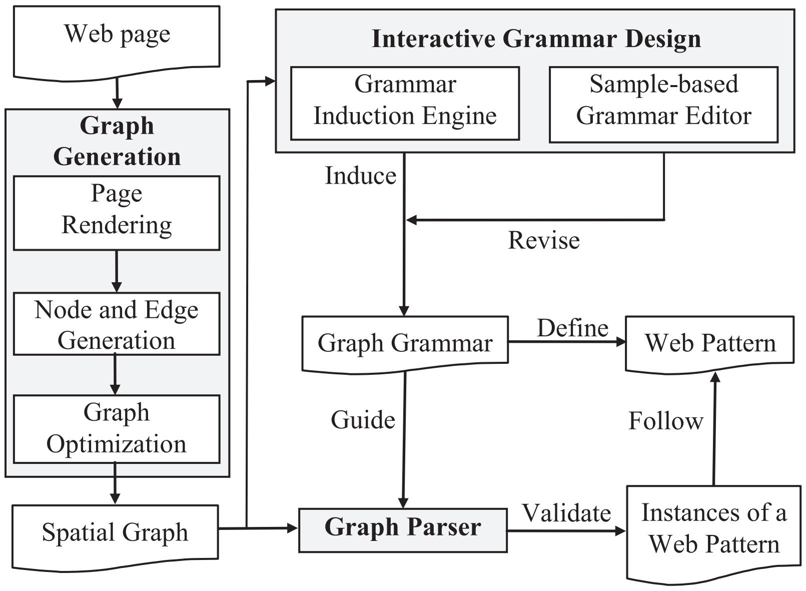


Fig. 3. Framework overview.

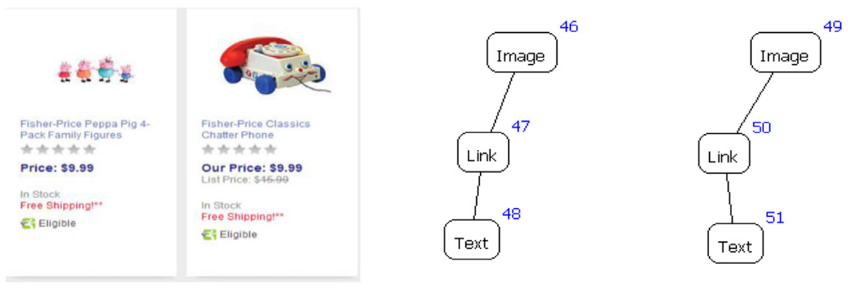
pattern. Web patterns can find many potential applications, such as extracting structured data, automatic consistency inspection, or saving energy on OLED displays [12].

4. Approach overview

We present an approach, called ***Visual Engineering for Web Patterns***, for specifying and validating Web patterns. This approach consists of three components as shown in Fig. 3. The graph generation component abstracts a Web page as a spatial graph that simplifies the original Web page and highlights important spatial relations between information objects. The interactive grammar design component visually specifies a Web pattern as a graph grammar. More specifically, a grammar induction engine automatically summarizes a Web pattern from sample Web pages as a graph grammar, while the sample based grammar editor can visually define or refine a grammar. The third component, i.e., the graph parser, parses the spatial graph of a given Web page to identify instances of a defined Web pattern according to the graph grammar.

4.1. Graph generation

Graph generation is an important step in our approach since it simplifies original Web pages by removing (1) style and layout elements that do not include any real content, (2) advertisements and (3) menus in the border areas. The simplification effectively reduces the complexity of HTML pages and potentially eliminates variations among different Web pages. The graph generation



(a) A portion of a Web page (b) Its corresponding spatial graph

Fig. 4. Graph generation.

process proceeds in the following steps: Web page rendering, node and edge generation, and graph optimization. The page rendering step determines the actual layout of a Web page (i.e., the position and size of each element) and generates dynamic elements. More specifically, the visual layout of a Web page is determined by three variables, i.e. (1) the actual HTML source code that specifies the DOM structure of the page, (2) data items such as text and picture and (3) style sheets and client side scripts which are executed by a browser at run time. In order to access dynamic contents generated through client-side scripts, we first render a Web page. Then, we traverse each HTML element in the DOM structure and record the size and position of each element. Our approach also fixes markup errors during the rendering step. After obtaining the dynamic/static HTML elements and their spatial properties, the second step generates a spatial graph in which a node represents an information object and an edge indicates a relation between the pair of connecting nodes. For example, Fig. 4 shows the screenshot of a Web page and its corresponding spatial graph. The last step optimizes the spatial graph by removing noises (such as advertisements and menus).

4.2. Grammar design and pattern validation

Based on spatial graphs, a Web pattern is visually specified through a graph grammar that includes a set of productions. The left graph in a production defines a composite information object, which is made of several atomic/composite information objects that satisfy certain spatial relations in the right graph. A graph grammar hierarchically integrates local spatial relations together to construct a Web pattern. Based on the graph grammar, a graph parser analyzes the spatial properties among information objects in the bottom-up fashion and recognizes in the spatial graph all substuctures that are consistent with the graph grammar (i.e., recognizing instances of a Web pattern). A graph grammar based specification brings the following benefits:

- A Web pattern defines essential spatial relations among objects at a high level. It is inevitable that there exist variations among instances of a Web pattern. For example, a pattern displaying a news story may place a title on the top, followed by several paragraphs. Different news stories may, however, have different numbers of paragraphs. A graph grammar is powerful to handle such variations by applying a production recursively.
- It is a one-time effort to design a graph grammar. Once a graph grammar is defined, it can be applied to different Web pages to validate instances of a defined Web pattern. Our approach decouples the pattern specification from pattern validation. Consequently, our framework is robust to handle pattern evolution and variations among different Web pages by updating the graph grammar without changing source codes.

Instead of designing a graph grammar from scratch, we provide an interactive process for users to design a graph grammar visually and intuitively. Grammar induction [7,8] is an automatic process of generating a grammar from given example graphs, saving the effort of manually designing the grammar. Our framework includes a novel grammar induction engine that converts a graph to a set of strings based on spatial properties and accordingly searches for repetative strings instead of 2D structures.

5. Graph generation

This section presents the graph generation process that converts a concrete Web page to a spatial graph.

5.1. Node generation

A spatial graph includes three types of nodes, i.e. *image*, *text* and *link*. The contents enclosed in the <*img*> or <*a*> tags are recognized as an *image* node or a *link* one, respectively. However, it is challenging to identify a *text* node since one complete sentence may be separated by several HTML tags. For example, formatting and styling tags, such as <*b*>, <*br*>, <*font*>, <*span*>, can divide a sentence into several pieces. During the graph generation process, all of those formatting and styling tags are removed and adjacent contents are consolidated as one single *text* node. The consolidation reduces the number of nodes in the spatial graph and thus speeds up the parsing process. In addition to the above three nodes, we also introduce *container* nodes, which are derived from structural HTML elements (such as <*Table*> or <*Div*>) and indicate regions within a Web page. A container node does not store real content. Instead, it is used to generate edges in the next step.

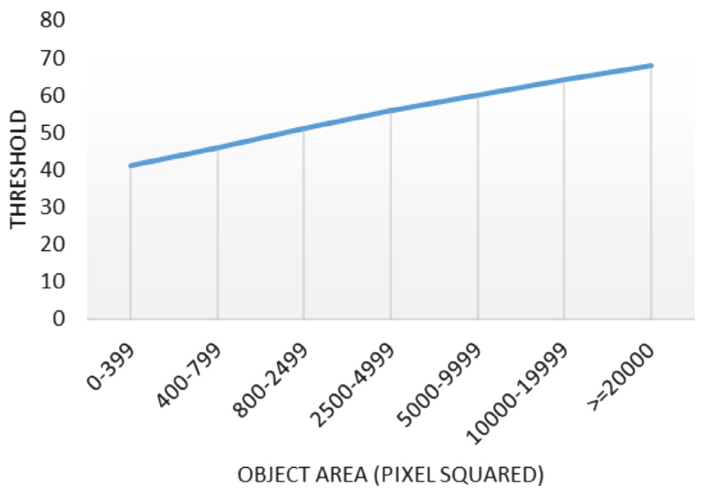


Fig. 5. Hash function determining relationships between an object size and threshold value.

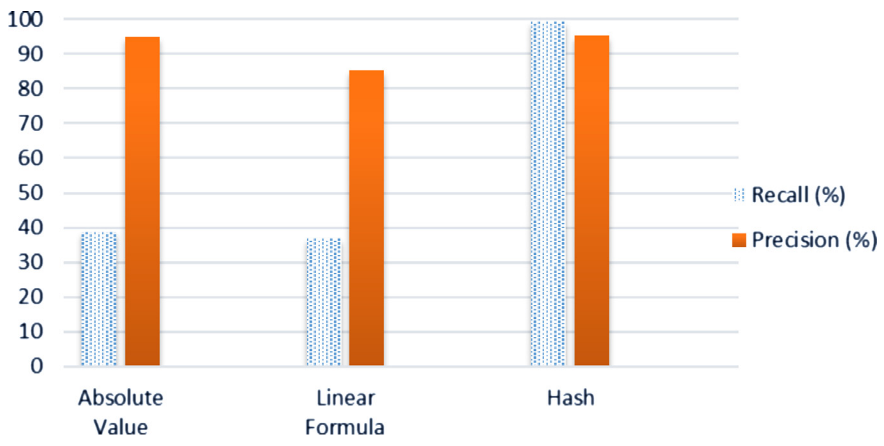


Fig. 6. Recall and precision of distance calculation approaches.

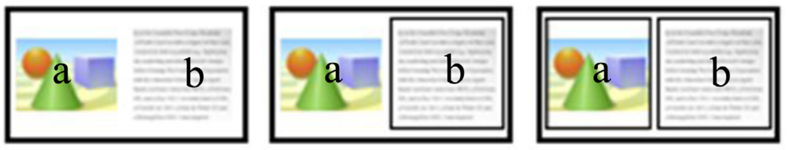
5.2. Edge generation

In a two dimensional space, an information object can have arbitrary spatial relations with adjacent nodes. Exhaustive graph parsing is time consuming. According to Mayer [34], using multimedia and putting pictures and explaining text together facilitate better learning for viewers. The HCI principle also dictates that related contents are placed in proximity [45]. Our approach uses distance to calculate whether two adjacent nodes are related or not. In a spatial graph, two closely related nodes are connected with an edge. We only parse the objects that are connected and thus reduce the search space to speed up the parsing process.

Each information object occupies a 2D space, so the distance cannot be directly calculated between two central points due to the object size. In our approach, we extend the size of object *a* to a certain degree. If at least two corners of object *b* fall in the extended rectangle of object *a*, objects *a* and *b* are closely related.

It is challenging to determine the threshold of extending an object. We have exploited three ways to determine the best value, i.e., an absolute value, a linear formula based on the size, and a hash function between the size of an object and its threshold (See Fig. 5). Fig. 6 presents the evaluation result of comparing the above three approaches on four Web sites (i.e., ubid.com, target.com, shopnbc.com and powells.com), and the hash function produces the best result.

In addition to the distance, the hierarchical DOM structure also provides hints to the relation between two objects. In general, closely related objects are placed in the same region, i.e., a container. According to the HTML DOM structure, we define a containment tree that specifies hierarchical relations among containers and information objects. For example, if an HTML element <*p*> (corresponding to a *text* object *t*) is enclosed in an HTML element <*table*> (corresponding to a container *c*), then the container *c* has a parent–child relation with the text object *t* in the containment tree. Since CCS style sheets or script languages could change the position of an information object that is completely different from the definition in the DOM structure, we adjust the containment tree based on the actual rendering. More specifically, we traverse every object in the containment tree: if an object



(a) The same container (b) The nested containers (c) The sibling containers

Fig. 7. Different combinations of containers.

is completely contained in its parent container, we keep the current containment relation. Otherwise, the parent of this object becomes the smallest container that completely contains it.

Based on the containment tree, if information object *a* has a close relation with information object *b*, *a* must have one of the following containment relations with *b*:

- As presented in Fig. 7 (a), *a* and *b* are in the same container; or
- As presented in Fig. 7 (b), the container of object *a* is the direct parent of the container of object *b*; or
- As presented in Fig. 7 (c), the container of object *a* is the sibling of the container of object *b*.

5.3. Optimization

The optimization step removes useless information in a spatial graph. For example, some Web designers place a small icon (such as the icon of a truck to indicate shipping) between a product image and product description. This design makes it slightly different from other instances of the same pattern, in which the product image is directly adjacent to the product description. We can eliminate this deviation by removing the small icon that does not really contribute to the real content. We have manually evaluated 50 Web sites, and observed that small elements (such as icons) are in general not important to real contents. Based on the observation, we have summarized two heuristic rules to remove small objects:

- Remove a *text* or *link* node, whose size includes less than 200 square pixels;
- Remove an image node, whose size includes less than 1400 square pixels.

A Web page is generally divided into 5 regions: top, bottom, left, right and center [27]. Menus and advertisements are in general placed in the border areas. Removing menus and advertisements can reduce the size of a spatial graph without losing useful information, since they are irrelevant to data extraction. Instead of removing individual information objects, our approach searches for a container, which is at least 200 pixels in width and is at least overlapping with 60% of a border area. If such a container is found, all of the information objects within this container are removed. Finally, any image that occurs more than twice is removed from the spatial graph. These repetitive images are typically buttons, icons or styling images used for formatting and layout without holding real contents.

6. Graph grammar based pattern definition

Based on spatial graphs, Web patterns are formally defined through graph grammars.

6.1. Spatial graph grammar formalism

Graph grammars with their well-established theoretical background can be used as a natural and powerful syntax-definition formalism [42] for visual languages, which model structures and concepts in a 2-dimensional fashion [15]. The parsing algorithm based on a graph grammar can be used to check the syntactical correctness and to interpret the language's semantics.

Different from other graph grammar formalisms, the Spatial Graph Grammar (SGG) [25] introduces spatial notions to the abstract syntax. In SGG, nodes and edges together with spatial relations construct the pre-condition of a production application. The direct representation of spatial information in the abstract syntax makes productions easy to understand since grammar designers often design rules with similar appearances as the represented graphs. Allowing designers to specify design knowledge in both structural and spatial properties simultaneously, the spatial graph grammar is ideal for specifying Web patterns. For example, the SGG production in Fig. 8a models the composition of information object *product* from both structural and spatial properties. More specifically, a *product* information object is made of three information objects, i.e. *link*, *image* and *text*. Furthermore, the link is placed above the image, which is further above the text. SGG also supports syntax-directed computations through action code. An action code is associated with a production, and is executed when the production is applied.

Applying a production to a spatial graph, usually called a *host graph*, is referred to as a graph transformation, which can be classified as an *L-application* (in a forward direction) or *R-application* (in a reverse direction). A *redex* is a sub-graph in the host graph which is isomorphic to the right graph in an R-application or to the left graph in an L-application. A production's L-application to a host graph is to find in the host graph a redex of the left graph of the production and replace the redex with the right graph. The language, defined as all possible graphs that have only terminal objects, can be derived through L-applications

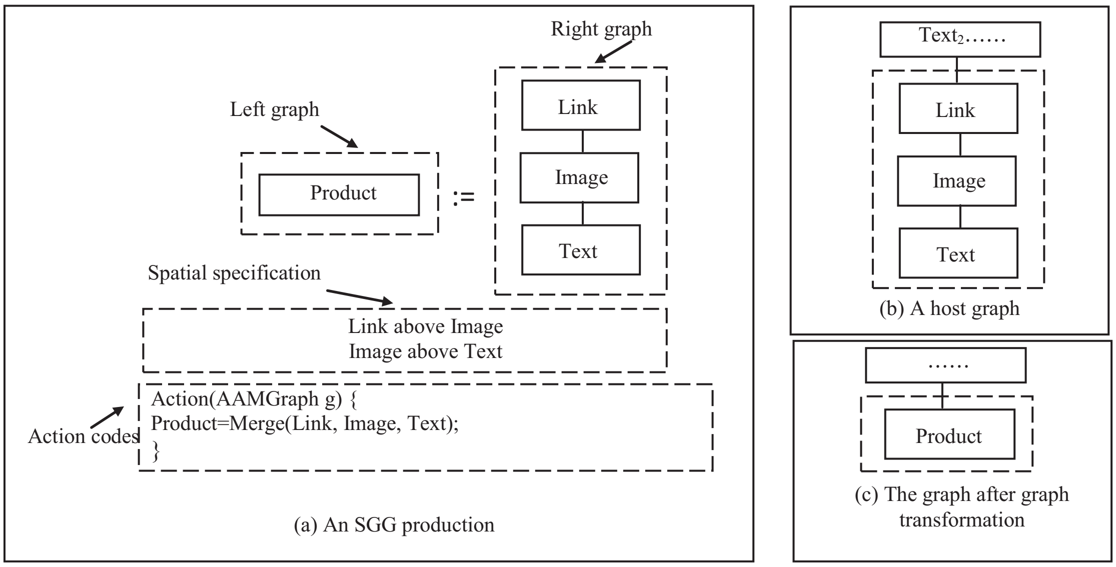


Fig. 8. The spatial graph grammar formalism.

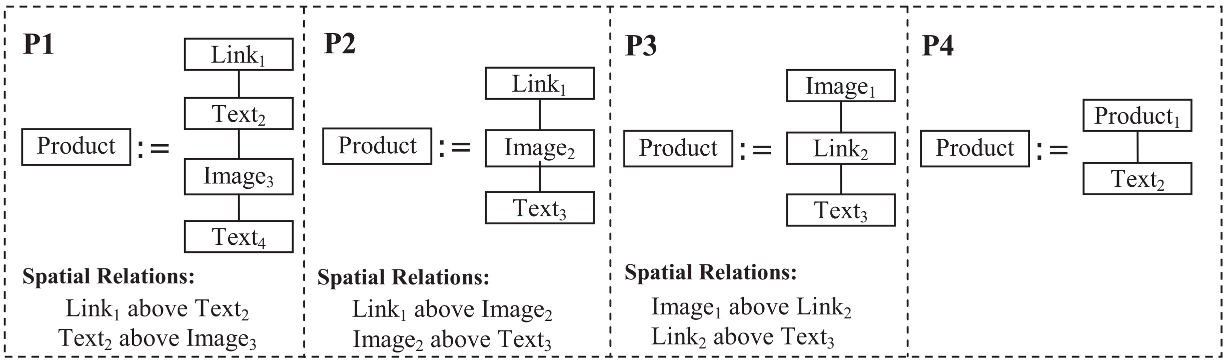


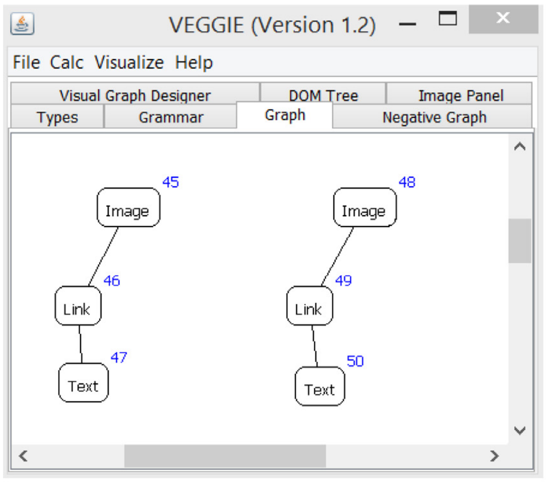
Fig. 9. The graph grammar for pattern 2.

(i.e., a generating process) from an initial graph. On the other hand, an R-application is the reverse replacement (i.e., from the right graph to the left graph) used to parse a graph. In this paper, R-applications (i.e. a parsing process) are used to recognize instances of a Web pattern. For example, Fig. 8b shows a spatial graph. The redex that matches the right graph in Fig. 8a is highlighted in the dashed rectangle. After an R-application, we can recognize a *product* object and the new host graph is updated in Fig. 8c.

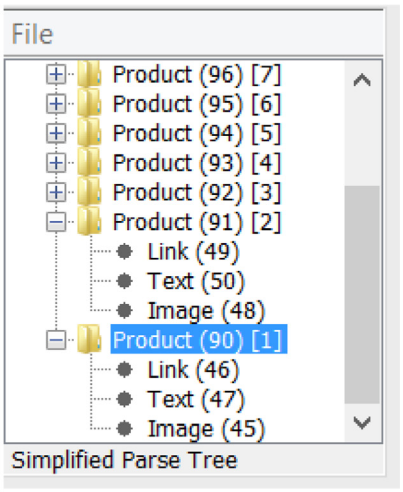
6.2. Pattern specification and validation

Fig. 9 presents the spatial graph grammar for Pattern 2 (See Fig. 2). More specifically, Productions P1 to P3 specify the variations among instances of Pattern 2. P1 specifies that a product is made up of a title (i.e. a *link* object), a brief description (i.e. a *text* object), a product picture (i.e. an *image* object) and other descriptions (i.e. a *text* object). P2/P3 specifies that a product includes a title, a product picture and other descriptions. All those objects are displayed vertically top down. In P2, a title is placed above an image and vice versa in P3. P4 is a recursive production that is used to recognize an arbitrary number of lines of product description.

Based on the defined graph grammar, the SGG parser takes a spatial graph as input and recognizes in the spatial graph all substructures that are consistent with the defined graph grammar. Each graph transformation reveals a local composition. For example, the application of Production P1 in Fig. 9 indicates that a composite object *product* consists of four information objects, i.e. *image*, *link* and two *text* objects. A sequence of graph transformations, i.e., the parsing process, assembles local compositions into a global hierarchical structure. For example, Fig. 10a presents a spatial graph, which includes instances of Pattern 2 illustrated in Fig. 2. Applying the graph grammar in Fig. 9 to the spatial graph in Fig. 10a can recognize instances of Pattern 2 as presented in Fig. 10b.



(a) A spatial graph



(b) Instances of Pattern 2

Fig. 10. Recognition of product information from a spatial graph.

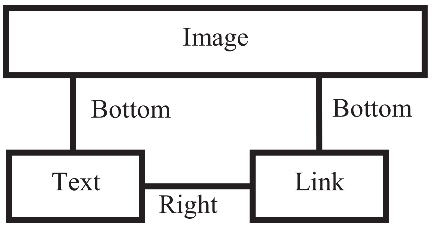


Fig. 11. A spatial graph.

7. Grammar induction

Instead of writing a graph grammar from scratch, our framework includes a novel grammar induction engine for automating the grammar design and construction. The grammar induction algorithm uses spatial information to sequence a 2D graph as a 1D string and accordingly converts the problem of searching for 2D structures as searching for 1D strings, and thus reduces the complexity significantly. More specifically, based on spatial properties, we converted a graph to a string, which is formalized through regular expressions. Repetitive structures are searched through regular expressions. Finally, the most popular regular expression is converted to a graph grammar based on the hierarchical structure within the regular expression.

7.1. A grammar induction engine

Our grammar induction algorithm has two features distinct from the previous approaches. First, to our knowledge, none of existing grammar induction engines considers spatial properties of a graph in the induction process. However, spatial information may play an important role in real applications. The layout in which a line of texts is displayed above an image can have a different meaning from the layout which places an image above a line of texts. Therefore, our approach considers the spatial information in the induction process. In our approach, two graphs are identical only if they have both the same structure (i.e., connections among nodes) and layout.

Second, traditional node replacements in the induction process cannot keep the spatial relations among terminal nodes. For example, Fig. 11 shows a spatial graph, in which each label near an edge indicates the spatial relation between the corresponding pair of nodes. If the *image* and *text* nodes are induced to a non-terminal node *P*1, the spatial relation between *P*1 and the remaining node *Link* becomes undefined due to the irregular shape of *P*1. Our approach starts the induction from a single node and gradually extends it to more complex graphs. During the extension, we search for repetitive structures and induce them as productions. The extension based approach preserves spatial relations among terminal objects during the induction process.

Searching repetitive structures in a graph is an NP-Complete problem due to cycles. Our approach sequences terminal objects based on left–right and top–bottom relations. Such a sequence allows us to search for structures in certain directions to avoid cycles. Our induction approach is presented in Fig. 12 and proceeds in four steps, i.e., generating a unique string for each node, summarizing Java regular expressions, expanding strings and graph grammar induction. The four steps are further explained with detailed examples in the following subsections.

```
1.  A unique string is generated for each node in the graph G.
2.  Summarize each string as a java regular expression
3.  Extend each node in G to the right and to the bottom. Each produced subgraph is
specified through a string.
4.  Find the top 20 common java regular expressions.
5.  for (i=1; i<MaxLevel;i++)
6.  {   extend the top 20 java regular expressions;
7.      find the top 20 java regular expressions;
8.  }
9.  Among top 20 java regular expressions, search for the regular expression that
covers the most area.
10. Convert the regular expression to a graph grammar
```

Fig. 12. Grammar induction algorithm.

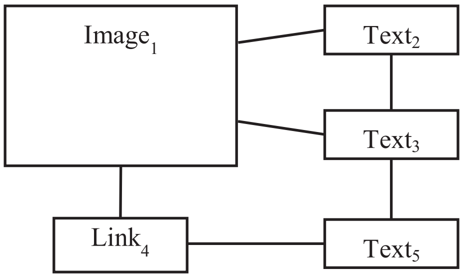


Fig. 13. A sample spatial graph.

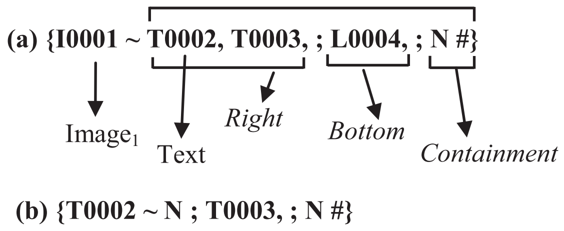

(a) {I0001 ~ T0002, T0003, ; L0004, ; N #}
Image1
Text
Right
Bottom
Containment
(b) {T0002 ~ N ; T0003, ; N #}

Fig. 14. Strings of image$_1$ and text$_2$.

7.1.1. *Step 1 - Create a string for each node*

Our approach converts a graph to a string, which consequently translates a graph search problem to a string search problem. Therefore, the first step is to convert each node in a graph to a unique string. During the conversion, we consider three types of spatial relations, i.e., *Bottom*, *Right* and *Containment*. More specifically, given node a, its corresponding string includes two parts, separated by the symbol ~. The first part presents the type and ID of node a, and the second part has three sections, separated by a semicolon. The first section specifies the nodes (including the node type and ID) that are connected and located to the right of node a, the second section contains nodes which are below, and the last section defines nodes that are contained by node a. The exclusion of spatial relations *left* and *top* avoids cycles in the following search. For example, Fig. 13 shows a spatial graph, and Fig. 14 presents the generated strings for nodes Image$_1$ and Text$_2$, respectively. In the above example, abbreviations of I, T, L represent Image, Text and Link, respectively; symbol N means that there is no node in that section; and symbol # indicates the end of a string.

7.1.2. *Step 2 - Summarize Java regular expressions*

We abstract the string of every node as a Java regular expression and summarize a list of unique regular expressions. For example, the string of node Image$_1$ in Fig. 14a is abstracted as the following regular expression.

$$\{\backslash\{I\backslash d\{4\} \sim (T\backslash d\{4\}\backslash,)\{1\}?(T\backslash d\{4\}\backslash,)\{1\}?[\hat{}\,;\,]^{*};(L\backslash d\backslash\{4\}\backslash,)\{1\}?[\hat{}\,;\,]^{*};[\hat{}\,;\,]^{*}\#\backslash\}$$

A regular expression keeps the node type, but it removes the detailed ID. Each detailed ID is represented with a general format that includes 4 digit numbers. For example, node *Image*$_1$ is abstracted as I\d{4} that indicates an image node with four digits as its ID. The above regular expression precisely defines nodes that are connected to Image$_1$ with certain spatial relations. For example, (T\d{4}\,){1} in the above regular expression indicates that there is one text node locating right to Image$_1$. *[^;]∗* is used as a wild card that allows variations when searching for an instance of this regular expression. Graphs that have the same spatial and structural properties are abstracted to the same regular expression. For example, any subgraph, which has an *Image* node in the top-left corner with at least two adjacent *Text* nodes at the right side and at least one adjacent *Link* on the bottom, is considered an instance of the above regular expression. Given the example in Fig. 13, we can summarize the following regular expressions:

$$\{\backslash\{I\backslash d\{4\}\sim(T\backslash d\{4\}\backslash,)\{1\}?(T\backslash d\{4\}\backslash,)\{1\}?[\hat{};]^*;(L\backslash d\backslash\{4\}\backslash,)\{1\}?[\hat{};]^*;[\hat{};]^*\#\backslash\}$$

$$\{\backslash\{T\backslash d\{4\}\sim[\hat{};]^*;(T\backslash d\{4\}\backslash,)\{1\}?[\hat{};]^*;[\hat{};]^*\#\backslash\}$$

$$\{\backslash\{L\backslash d\{4\}\sim(T\backslash d\{4\}\backslash,)\{1\}?[\hat{};]^*;[\hat{};]^*;[\hat{};]^*\#\backslash\}$$

In the following description, ***Set_RegularExpression*** is used to indicate all regular expressions generated in Step 2.

7.1.3. Step 3 - Expand strings

To improve the search efficiency, we divide a graph into a group of subgraphs by gradually extending the strings generated in the first step. More specifically, a string in Step 1 indicates a minimal subgraph, which only includes the node in the top-left corner (e.g., Image$_1$ in Fig. 14.a) and its adjacent nodes. The node in the left-top corner is referred to as the *origin* of the subgraph. We can extend this minimal subgraph to the right and to the bottom by replacing each adjacent node with a string that takes the adjacent node as the origin. For example, the string of node Image$_1$ in Fig. 14 a can be extended to the following string:

$$\{I0001\sim\{T0002\sim N;T0003,;N\#\},\{T0003\sim N;T0005,;N\#\},;\{L0004\sim T0005,;N;N\#\};N\#\}$$

In the above, node T0002 is extended to the subgraph of {T0002∼N;T0003,;N#}. The first round of extension increases the maximum distance between the origin (e.g., Image$_1$ in the above example) and any other node in the extended subgraph to 2. The extension continues until the node in the bottom-right corner is reached or the maximum distance reaches 10. This extension is applied to every graph string generated in Step 1. Therefore, given a graph of n nodes, the third step will generate n subgraphs. In other words, the original graph is divided into n subgraphs and the search of repetitive structures is performed within each subgraph.

7.1.4. Step 4 - Graph grammar induction

Based on the regular expressions produced in Step 2 and n subgraphs generated in Step 3, we search for the instances of each regular expression in those subgraphs, and select the top 20 regular expressions, recorded in ***Candidates_GG***, which have the most instances in n subgraphs. In the first iteration step, each selected regular expression only specifies a minimal structure, which only includes the origin and its neighbors. We take those top 20 regular expressions as the initial set of repetitive structures, and gradually expand them in the following iterations. In one iteration of expansion, except the origin, a node with type x in a regular expression is replaced with every eligible regular expression in ***Set_RegularExpression***, which has a node with type x as the origin. Since several regular expressions may have the same node type in the top-left corner, expanding one regular expression will produce multiple new expanded regular expressions. Furthermore, the expansion may replace only one node or multiple nodes simultaneously. For example, considering the example in Fig. 13, expanding the following regular expression can produce a total of 7 new expanded regular expressions (See Appendix A).

$$\{\backslash\{I\backslash d\{4\}\sim(T\backslash d\{4\}\backslash,)\{1\}?(T\backslash d\{4\}\backslash,)\{1\}?[\hat{};]^*;(L\backslash d\backslash\{4\}\backslash,)\{1\}?[\hat{};]^*;[\hat{};]^*\#\backslash\}$$

For example, the first expanded regular expression replaces node T\d{4} with the regular expression {T\d{4}∼[^;]∗;(T\d{4}\,){1}?[^;]∗;[^;]∗#\}. Each expanded regular expression is added to ***Candidates_GG***. Number 10000 indicates the first level of expansion. In the first 3 expanded regular expressions, only one node is expanded; in the remaining four expanded regular expressions, multiple nodes are expanded simultaneously. In the above example, only one regular expression starts with a *Text* Node. If there is more than one, we need to iterate each eligible expression for the expansion. After the expansion, we calculate the number of instances for each regular expression in ***Candidates_GG*** and again select the top 20 regular expressions. Furthermore, each selected regular expression must have at least 4 instances. Only those selected regular expressions are kept in ***Candidates_GG***. Then, we continue the second round of expansion and calculation. The expansion is performed for 10 iterations. After all rounds of expansion, we finally select the top 20 regular expressions. Based on the observation that important information occupies the main area in a Web page, we calculate the total area that is covered by the instances of each regular expression, and choose the regular expression that covers the largest area as the final induced graph grammar.

A regular expression can consist of multiple levels. Starting from the deepest level (i.e., level 10), we extract productions in a bottom–up fashion. For example, given the following regular expression, we can summarize a graph grammar as shown

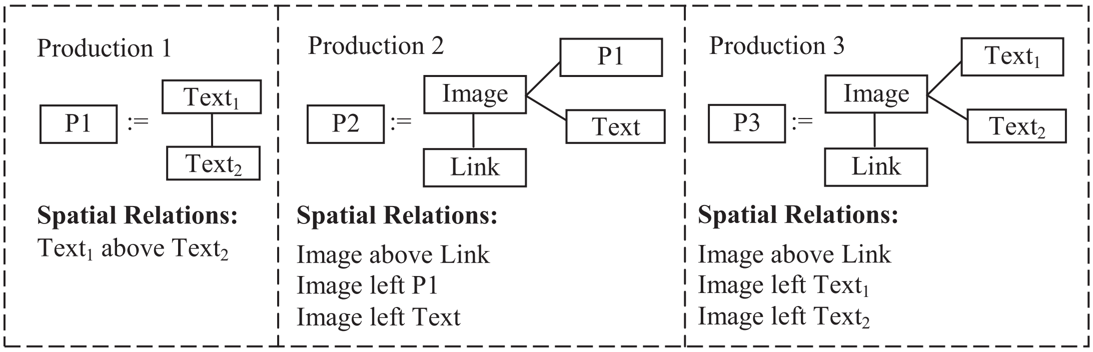
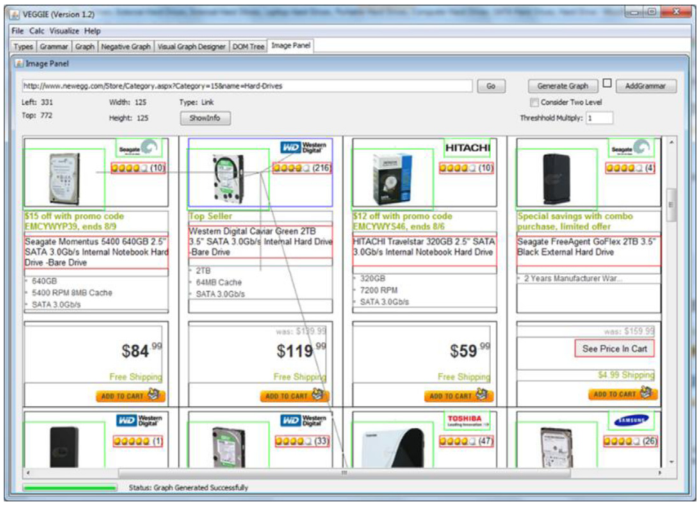


Fig. 15. An induced graph grammar.



http://www.newegg.com/Store/Category.aspx?Category=15&name=Hard-Drives

Left: 331 Width: 125 Type: Link

Top: 772 Height: 125

Consider Two Level

Threshhold Multiply: 1

Status: Graph Generated Successfully

Fig. 16. A sample-based grammar editor.

in Fig. 15.

$$\{\backslash\{I\backslash d\{4\}\sim(10000\backslash\{T\backslash d\{4\}\sim[\hat{};]*;(T\backslash d\{4\}\backslash,)\{1\}?[\hat{};]*;[\hat{};]*\#\backslash\}10000\backslash,)\{1\}?(T\backslash d\{4\}\backslash,)\{1\}?$$
$$[\hat{};]*;(L\backslash d\{4\}\backslash,)\{1\}?[\hat{};]*;[\hat{};]*\#\backslash\}$$

The first production is extracted from the regular expression with the deepest level, i.e., **10000\{T\d{4} ~[^;]*; (T\d{4}\,){1}?[^;]*;[^;]*#\}10000**. The second production (P2) is summarized based on P1. Since P1 is extended from a single text node, we also summarize P3 that includes a terminal node instead of substructure of P1.

7.2. *Interactive grammar design*

Due to the lack of domain knowledge, a grammar induction algorithm in general assigns recognized substructures with program-generated names, which make it hard to understand the induced grammar. In order to complement the automatic grammar induction, we develop a sample-based grammar editor, in which grammar designers directly manipulate the screenshot of a Web page to design or elaborate a graph grammar. The grammar editor looks like a regular Web browser (See Fig. 16). In the editor, each recognized information object is highlighted in a rectangle. When a user clicks an information object, edges that are connected with the selected object are displayed accordingly. Based on the editor, a grammar designer can construct a production by visually selecting related information objects to define the right graph. The sample-based grammar editor bridges the gap between the concrete layout of a Web page and its abstraction (i.e., a spatial graph), since the editor visualizes the spatial graph in a concrete context. The visualization facilitates grammar designers to recognize the abstract structure underlying the layout of a Web page, which benefits the designers to understand and elaborate an induced graph grammar.

8. Experiment

The method of Mining Data Records (i.e., MDR) [32] was used as a benchmark to compare with our approach since MDR is the only executable open source that we find available online. In the evaluation, we chose to extract product-related information

Table 3
Experimental results.

	Domain name	# of structured records	MDR		Our approach (Manual)		Our approach (Auto)	
			Correct	Found	Correct	Found	Correct	Found
1	shopping.yahoo.com	15	0	0	14	14	14	14
2	scistore.cambridgesoft.com	13	13	13	13	14	13	15
3	shop.lycos.com	18	0	0	18	18	18	18
4	barnesandnoble.com	48	0	0	48	48	48	48
5	borders.com	27	27	32	28	29	28	30
6	circuitcity.com	5	5	5	5	7	5	7
7	compusa.com	18	0	0	18	21	18	21
8	drugstore.com	15	15	15	13	14	13	14
9	ebay.com	20	20	20	20	20	20	20
10	etoys.com	32	0	0	32	32	32	32
11	kidsfootlocker.com	29	29	29	29	29	29	29
12	kodak.com	20	0	0	20	20	20	20
13	newegg.com	20	0	0	20	26	20	26
14	nothingbutsoftware.com	24	24	24	24	24	24	24
15	overstock.com	18	18	18	18	18	18	18
16	powells.com	50	50	50	50	51	50	51
17	softwareoutlet.com	14	0	0	14	15	14	16
18	ubid.com	8	0	0	8	9	8	9
19	amazon.com	7	0	0	7	8	7	8
20	shopping.hp.com	5	5	5	5	5	5	5
21	qualityinks.com	24	24	24	24	26	24	26
	Total	430	230	235	423	443	423	451
	Recall/Precision		53.5%/97.9%		99.5%/95.5%		99.5%/95.0%	
	F1-Score		69.19%	97.49%	97.16%			

from online ecommerce Web sites. It is worth mentioning that our approach is domain-independent and can be applied to different domains as long as those Web pages are designed using the same pattern.

8.1. Setup

Experiment web pages: Liu et al. [32] evaluated MDR on 46 Web sites. From those 46 Web sites, we eliminate the Web sites that are not accessible and not in the category of ecommerce, and finally select 21 Web sites as the test set.

Measurement: We measured the performance with the standard metrics:

- recall $= \frac{E_{\text{correct}}}{N_{\text{total}}}$;
- precision $= \frac{E_{\text{correct}}}{E_{\text{total}}}$;

where N_{total} is the total number of data records contained in a Web page; E_{correct} indicates the number of correctly extracted data records; and E_{total} denotes the total number of data records extracted from a Web page. We also calculated the F1-Score, which is the harmonic mean of *precision* and *recall* and is defined as $\frac{2\times\text{recall}\times\text{precision}}{\text{recall}+\text{precision}}$. The F1-Score has been commonly used as a metric to evaluate the overall performance in many approaches [13,29]. MDR is a general process that can recognize any repetitive structures within a Web page. Those recognized records may belong to different categories. Since those product-unrelated records may affect the recall and precision of MDR, we only calculate the number of product records recognized by MDR in order to compare two approaches fairly. In other words, structured records that are extracted by MDR but not related to products are not counted toward E_{correct} and E_{total}. In addition, we evaluate the processing time and the complexity of the spatial graph in our approach. To evaluate the quality of grammar induction, we further compare the performance between a manually designed grammar and an automatically induced one.

8.2. Evaluation

8.2.1. Precision/recall/F1-score

The results are presented in Table 3. The recall of our approach is 99.5% for both manual and automated grammar, compared to 53.5% with MDR. The high recall rate in our approach indicates that the visual analysis is powerful in recognizing repetitive structures. The precision of our approach is 95.5% for the manually designed grammar and 95.0% for the automated grammar, both close to 97.9% with MDR. The falsely recognized records in our approach are mainly caused by noises. For example, if an advertisement is placed in the central area and its layout is similar to pattern 2 this advertisement may be recognized as a product record. In order to improve the precision, it is critical to improve the graph generation process by removing potential noises. In summary, our approach has a high F1-Score of 97.49% for the manually designed grammar and 97.16% for the automatically induced grammar, compared with 69.19% with MDR.

Table 4
The complexity of spatial graphs and processing time.

	URL	Size			Processing time (Milliseconds)	
		HTML tags	Nodes	Reduction percent	Graph generation	Pattern validation
1	shopping.yahoo.com	1065	504	47.3%	1063	373
2	scistore.cambridgesoft.com	1213	488	40.2%	1301	368
3	shop.lycos.com	522	233	44.6%	210	139
4	barnesandnoble.com	2328	851	36.6%	2702	1899
5	borders.com	812	342	42.1%	384	297
6	circuitcity.com	1664	557	33.5%	1094	198
7	compusa.com	1509	473	31.3%	752	228
8	drugstore.com	690	279	40.4%	248	507
9	ebay.com	1647	875	53.1%	2714	2112
10	etoys.com	677	234	34.6%	176	140
11	kidsfootlocker.com	608	276	45.4%	174	68
12	kodak.com	679	301	44.3%	238	85
13	newegg.com	1724	861	49.9%	1200	6703
14	nothingbutsoftware.com	1048	148	14.1%	305	33
15	overstock.com	1416	889	62.8%	1841	2038
16	powells.com	2531	951	37.6%	1488	14860
17	softwareoutlet.com	1117	247	22.1%	260	137
18	ubid.com	360	111	30.8%	100	22
19	amazon.com	1208	552	45.7%	530	511
20	shopping.hp.com	1046	392	37.5%	257	239
21	qualityinks.com	513	237	46.2%	129	144
	Average	1160.8	466.7	40%	817.4	1481

8.2.2. The complexity of spatial graphs

The graph generation process simplifies the original Web page by consolidating information pieces and removing noises. Table 4 compares the size of a Web page to that of its corresponding spatial graph. On average, the size of a spatial graph is only 40% of the corresponding Web page.

8.2.3. Processing time

We evaluated the MDR and our approach on a desktop with a Core 2 Duo CPU 2.26 GHz and 4 GB RAM, running Windows 7 Professional. Table 4 presents the processing time for the graph generation and pattern validation. The average graph generation time for an average Web page with 1160 HTML tags is less than 1 s, while the pattern validation is less than 1.5 s on average. The processing time does not include the time used for downloading and rendering the page since it depends on the network speed. We were not able to record the MDR processing time since we can only access to the executable version of MDR. MDR has a user interface, in which a user has to click multiple buttons in a series, and it will then generate a file with the data in the end. Because of the user interaction, it is hard to precisely compare MDR in terms of the processing time with our approach in which we are measuring the processing time in the code.

8.3. Discussion

Graph Generation is one of the most important components in our approach. As shown in Table 4, the generated spatial graphs reduce the complexity of the original Web page by 60%. The precision can be further improved by introducing computer vision techniques to the graph generation process, such as analyzing the background and foreground colors or styles. In the evaluation, some product records are not recognized by our approach. For example, in Fig. 17, text objects 1 and 2 are far away and the distance exceeds the defined threshold. Therefore, text objects 1 and 2 are not connected. Consequently, they are not recognized correctly by our approach. This problem could be solved by analyzing the background color of information objects. The same group of information objects often uses the same background color. If two objects have the same background color within the same container and no other object between them, those two objects are likely to be closely related.

The complex usage of *div*/*table* may also reduce the recall. Some Web page designers use tables or divisions for the layout purpose. Therefore, information object of the same product may be organized in different containers, making those objects not closely related in the graph generation process. If we also define some containment relations based on the background color of information objects, they may be connected in the spatial graph. We also notice it is not sufficient to solely depend on the background color. For example, though text objects 1 and 3 have the same background color, they may not be related if there is a separation line between them. Therefore, integrating computer vision techniques (such as color and line detection) with the distance and DOM structure analysis may improve the quality of a spatial graph.

The evaluation results show that our approach is powerful in recognizing structured records. However, it may falsely recognize some product-unrelated records, such as advertisements that are presented similarly to product records. Though the graph generation process has removed advertisements placed in the border areas, the advertisements placed in the central area are

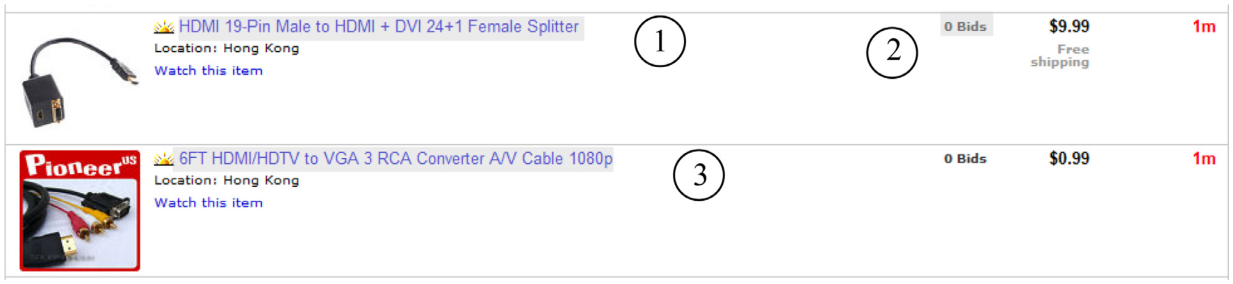


Fig. 17. Long distance between text elements.

Table 5
The precision and recall rates of various approaches.

Approach	Our approach	ViNTs [57]	[35]	[10]	DEPTA [58]	ViPER [46]	FiVaTech [36]	[2]
Precision	95.5%	98.7%	96.2%	98.9%	98.18%	98.6%	97.4%	97.39%
Recall	99.5%	98.7%	93.1%	97.6%	99.68%	98.0%	97.0%	98.55%
F1-score	97.5%	98.7%	94.6%	98.2%	98.9%	98.3%	97.2%	97.9%

hard to remove. In the future, we will consider adapting content analysis techniques [20,27] to our approach. Based on the content analysis, we can distinguish main contents from other sections (e.g. advertisements or menu) in a Web page and limit the data extraction in the main area.

As mobile devices become increasing popular, Web browsing has been extended from desktops to mobile devices. Due to the hardware difference, such as screen size and computing power, the layout of mobile Web pages is different from that of desktop Web pages, which makes it challenging to design a generic cross-device and cross-domain solution. In our approach, grammar induction automatically discovers a pattern underlying sample Web pages (such as pages from the same Web site). Then, data extraction is conducted based on the derived graph grammar. Since each pattern is uniquely defined through a graph grammar and grammar definition process is automatic, our approach is device- and domain-independent.

Similar to our approach, a majority of existing approaches use MDR as a benchmark to compare with (See Table 5) since MDR provides the open source that is easy to use. Unfortunately, without access to the source or executable code of other related approaches, it is infeasible to make a direct comparison. In order to compare our approach with related work, we have extracted the precision, recall and F1-score of related approaches and presented them in Table 5. The comparison indicates a good performance of our approach, in particular the recall.

9. Conclusion

Based on graph grammars, this paper presents a novel solution for specifying Web patterns and validating the instances of a Web pattern (i.e., data extraction) from different websites without the need of training and adjustment. Our approach utilizes both the visual features and the HTML DOM structure to abstract a Web page as a spatial graph. The spatial graph significantly simplifies the complexity of the original Web page by consolidating information pieces together and removing noises, and thus improves the efficiency of the pattern specification and validation. On top of spatial graphs, a Web pattern is defined as a graph grammar and its instances are recognized through a graph parsing process. To minimize the effort of designing a graph grammar, our approach is featured with a grammar induction engine which automatically summarizes a Web pattern from sample pages. Distinct from existing grammar induction algorithms, our approach takes advantage of spatial information and converts the 2D graph induction to the 1D string search.

We have implemented a prototype and tested it on 21 Web sites in the domain of E-commerce. The evaluation results are promising. Our approach has a high F1-Score (97.49% for a manually designed grammar and 97.16% for an automated one), compared to 69.19% of the MDR approach. The evaluation indicates that our approach produces a good performance in both precision and recall. The main advantage of our approach lies in its ability to convert complex HTML DOM structures to simple spatial graphs (i.e., reducing the complexity of the original page by 60% on average) and to automatically extract a Web pattern as a graph grammar from sample pages.

As the future work, we will add and identify more spatial relations between information objects, and optimize the graph generation algorithm. These optimizations increase the quality of generated spatial graphs, which can affect both the precision and recall. We also plan to use computer vision techniques to recognize boundaries between different regions and to use content analysis techniques to analyze actual contents.

Appendix A

The original expression:

$$\{\backslash\{I\backslash d\{4\}\sim(T\backslash d\{4\}\backslash,)\{1\}?(T\backslash d\{4\}\backslash,)\{1\}?[\hat{};]^{*};(L\backslash d\backslash\{4\}\backslash,)\{1\}?[\hat{};]^{*};[\hat{};]^{*}\#\backslash\}$$

Expanded regular expressions:

```
1. {\{I\d{4}~(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]*#\}10000\,){1
   }?(T\d{4}\,){1}?[^;]*;(L\d{4}\,){1}?[^;]*;[^;]*#\}
2. {\{I\d{4}~(T\d{4}\,){1}?(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]
   *#\}10000\,){1}?[^;]*;(L\d{4}\,){1}?[^;]*;[^;]*#\}
3. {\{I\d{4}~(T\d{4}\,){1}?(T\d{4}\,){1}?[^;]*;(10000{\{L\d{4}~(T\d{4}\,){
   1}?[^;]*;[^;]*;[^;]*#\}10000\,){1}?[^;]*;[^;]*#\}
4. {\{I\d{4}~(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]*#\}10000\,){1
   }?(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]*#\}10000\,){1}?[^;]*;
   (L\d{4}\,){1}?[^;]*;[^;]*#\}
5. {\{I\d{4}~(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]*#\}10000\,){1
   }?(T\d{4}\,){1}?[^;]*;(10000{\{L\d{4}~(T\d{4}\,){1}?[^;]*;[^;]*;[^;]*#\
   }10000\,){1}?[^;]*;[^;]*#\}
6. {\{I\d{4}~(T\d{4}\,){1}?(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]
   *#\}10000\,){1}?[^;]*;(10000{\{L\d{4}~(T\d{4}\,){1}?[^;]*;[^;]*;[^;]*#\
   }10000\,){1}?[^;]*;[^;]*#\}
7. {\{I\d{4}~(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]*#\}10000\,){1
   }?(10000\{\{T\d{4}~[^;]*;(T\d{4}\,){1}?[^;]*;[^;]*#\}10000\,){1}?[^;]*;
   (10000{\{L\d{4}~(T\d{4}\,){1}?[^;]*;[^;]*;[^;]*#\}10000\,){1}?[^;]*;[^;
   ]*#\}
```

References

[1] H. Ahmadi, J. Kong, Efficient web browsing on small screens, in: Proceedings of the Working Conference on Advanced Visual interfaces, 2008, pp. 23–30.

[2] H. Ahmadi, J. Kong, User-Centric Adaptation of Web Information for Small Screens, J. Visual Lang. Comput. vol. 23 (1) (2012) 13–28.

[3] M. Alvarez, A. Pan, J. Raposo, F. Bellas, F. Cacheda, Finding and extracting data records from Web pages, J. Signal Process. Syst. 59 (2010) 123–137.

[4] M.S. Amin, H. Jamil, An efficient Web-based wrapper and annotator for tabular data, Int. J. Softw. Eng. Knowl. Eng. vol. 20 (2) (2010) 215–231.

[5] N. Anderson, J. Hong, Visually extracting data records from the deep Web, in: Proc. WWW'13, 2013, pp. 1233–1238.

[6] A. Arasu, H. Garcia-Molina, Extracting structured data from Web pages, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, 2003, pp. 337–348.

[7] K. Ates, J. Kukluk, L. Holder, D. Cook, K. Zhang, Graph grammar induction on structural data for visual programming, in: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, 2006, pp. 232–242.

[8] K. Ates, K. Zhang, Constructing VEGGIE: machine learning for context-sensitive graph grammars, in: Proceedings of the 19th IEEE International Conference on Tools with Artificial intelligence, vol. 02, 2007, pp. 456–463.

[9] L.D. Bing, W. Lam, Y. Gu, Towards a unified solution data record region detection and segmentation, in: Proc. CIKM'11, 2011, pp. 1265–1274.

[10] L.D. Bing, W. Lam, T.L. Wong, Robust detection of semi-structured Web records using a DOM structure-knowledge-driven model, ACM Trans. Web 7 (4) (2013) Article ID 21.

[11] D. Cai, S. Yu, J. Wen, M.W., Extracting content structure for web pages based on visual representation, in: Proc. Asia Pacific Web, 2003, pp. 406–417.

[12] T.C. Chang, S.S. Xu, Object-image-based quality-on-demand energy saving schemes for OLED displays, Electron. Lett. vol. 50 (22) (2014) 1595–1597.

[13] J. Chen, K. Xiao, Perception-oriented online news extraction, in: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries, 2008, pp. 363–366.

[14] S. Chuang, J.Y. Hsu, Tree-structured template generation for Web pages, in: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, 2004, pp. 327–333.

[15] P.T. Cox, T. Smedley, Building environments for visual programming of robots by demonstration, J. Visual Lang. Comput. vol. 11 (5) (2000) 549–571.

[16] V. Crescenzi, G. Mecca, P. Merialdo, RoadRunner: towards automatic data extraction from large Web sites, in: Proceedings of the 27th International Conference on Very Large Data Bases, 2001, pp. 109–118.

[17] N. Dalvi, R. Kumar, M. Soliman, Automatic wrappers for large scale Web extraction, in: Proceedings of the VLDB Endowment, 4, 2011, pp. 219–230.

[18] O. Ermelinda, M. Ruffolo, SILA: A spatial instance learning approach for deep web pages, in: Proc. CIKM'11, 2011, pp. 2329–2332.

[19] D. Freitag, N. Kushmerick, Boosted wrapper induction, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, 2000, pp. 577–583.

[20] E. Fersini, E. Messina, F. Archetti, Enhancing web page classification through image-block importance analysis, Inf. Process. Manage. 44 (4) (2008) 1431–1447 (Jul. 2008).

[21] J. Geng, J. Yang, Automatic extraction and integration of bibliographic information on the Web, IDEAS'04 (2004) 193–204.

[22] C. Hsu, M. Dung, Generating finite-state transducers for semi-structured data extraction from the Web, Inf. Syst. 23 (9) (1998) 521–538 (Dec. 1998).

[23] F. Hu, T. Ruan, Z.Q. Shao, J. Ding, Automatic Web information extraction based on rules, Proc. WISE 2011 (2011) 265–272.

[24] P.M. Joshi, S. Liu, Web document text and images extraction using DOM analysis and natural language processing, in: Proceedings of the 9th ACM Symposium on Document Engineering, 2009, pp. 218–221.

[25] J. Kong, K. Zhang, X. Zeng, Spatial graph grammars for graphical user interfaces, ACM Trans. Comput. Human Interact. 13 (2) (2006) 268–307 (Jun. 2006).

[26] J. Kong, O. Barkol, R. Bergman, A. Pnueli, S. Schein, C.Y. Zhao, K. Zhang, Web interface interpretation using graph grammars, IEEE Trans. SMC – Part C 42 (4) (2012) 590–602.

[27] M. Kovacevic, M. Diligenti, M. Gori, V. Milutinovic, Recognition of common areas in a Web page using visual information: a possible application in a page classification, in: Proceedings of the 2002 IEEE International Conference on Data Mining, 2002, p. 250.

[28] N. Kushmerick, D. Weld, R. Doorenbos, Wrapper induction for information extraction, in: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997, pp. 729–737.

[29] E.S. Laber, C.P. de Souza, I.V. Jabour, E.C. de Amorim, E.T. Cardoso, R.P. Rentería, L.C. Tinoco, C.D. Valentim, A fast and simple method for extracting relevant content from news webpages, in: Proceeding of the 18th ACM Conference on Information and Knowledge Management, 2009, pp. 1685–1688.
[30] P. Ladyzynski, P. Grzegorzewski, Retrieving informative content from Web pages with conditional learning of support vector machines and semantic analysis, in: Proc. ICAISC 2012, Part II, 2012, pp. 128–135.
[31] A.H. Laender, B.A. Ribeiro-Neto, A.S. da Silva, J.S. Teixeira, A brief survey of web data extraction tools, SIGMOD Rec. 31 (2) (2002) 84–93 (Jun. 2002).
[32] B. Liu, R. Grossman, Y. Zhai, Mining data records in Web pages, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 601–606.
[33] W. Liu, X. Meng, W. Meng, Vide: a vision-based approach for deep web data extraction, IEEE Trans. Knowl. Data Eng. 22 (3) (2010) 447–460.
[34] R.E. Mayer, Multimedia Learning, Cambridge University Press, New York, 2005.
[35] G.X. Miao, J. Tatemura, W.P. Hsiung, A. Sawires, L. Moser, Extracting data records from the Web using tag path clustering, in: Proc. WWW'2009, 2009, pp. 981–990.
[36] K. Mohammed, C.H. Chang, FiVaTech: page-level web data extraction from template pages, IEEE Trans. Knowl. Data Eng. 22 (2) (2010) 249–263.
[37] I. Muslea, S. Minton, C. Knoblock, A hierarchical approach to wrapper induction, in: Proceedings of the Third Annual Conference on Autonomous Agents, 1999, pp. 190–197.
[38] I. Muslea, S. Minton, C.A. Knoblock, Hierarchical wrapper induction for semistructured information sources, Auton. Agents Multi-Agent Syst. 4 (1–2) (2001) 93–114 (Mar. 2001).
[39] A. Penev, R.K. Wong, Grouping hyperlinks for improved voice/mobile accessibility, in: Proceedings of the 2008 international cross-disciplinary conference on Web accessibility, 2008, pp. 50–53.
[40] S. Raeymatkers, M. Bruynooghe, Sub node extraction with tree based wrappers, in: Proceedings of the European Conference on Artificial Intelligence, 2008, pp. 137–141.
[41] D.C. Reis, P.B. Golgher, A.S. Silva, A.F. Laender, Automatic web news extraction using tree edit distance, in: Proceedings of the 13th International Conference on World Wide Web, 2004, pp. 502–511.
[42] G. Rozenberg (Ed.), Handbook on Graph Grammars and Computing by Graph Transformation: Foundations, vol. 1, World Scientific, 1997.
[43] A. Roudaki, J. Kong, Graph grammar based Web data extraction, in: Proceedings of International Conference on Software Engineering and Knowledge Engineering, 2011, pp. 373–378.
[44] S. Sarawagi, Automation in information extraction and data integration (tutorial), in: Proceedings of VLDB 2002, 2002 http://dblp.uni-trier.de/rec/bibtex/conf/vldb/Sarawagi02.
[45] B. Shneiderman, C. Plaisant, Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley Longman Publishing Co., Inc., 2009.
[46] K. Simon, G. Lausen, ViPER: augmenting automatic information extraction with visual perceptions, in: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 381–388.
[47] M. Skounakis, M. Craven, S. Ray, Hierarchical hidden Markov models for information extraction, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, 2003, pp. 427–433.
[48] H.A. Sleiman, R. Corchuelo, A survey on region extractor from web documents, IEEE Trans. Knowl. Data Eng. 25 (9) (2012) 1960–1981.
[49] H.A. Sleiman, R. Corchuelo, TEX: an efficient and effective unsupervised Web information extractor, Knowl. Based Syst. 39 (2013) 109–123.
[50] R. Song, H. Liu, J. Wen, W. Ma, Learning block importance models for web pages, in: Proceedings of the 13th International Conference on World Wide Web, 2004, pp. 203–211.
[51] X.Y. Song, J. Liu, Y.B. Cao, C.Y. Lin, H.W. Hon, Automatic extraction of Web data records containing user-generated content, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, 2010, pp. 39–48.
[52] X.Y. Xiao, Q. Luo, D. Hong, H.B. Fu, X. Xie, W.Y. Ma, Browsing on small displays by transforming Web pages into hierarchically structured subpages, ACM Trans. Web vol. 3 (1) (2009) Article ID 4.
[53] X. Yin, W.S. Lee, Using link analysis to improve layout on mobile devices, in: Proceedings of the 13th International Conference on World Wide Web, 2004, pp. 338–344.
[54] X. Yin, W.S. Lee, Understanding the function of web elements for mobile content delivery using random walk models, in: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, 2005, pp. 1150–1151.
[55] Y. You, G. Xu, J. Cao, Y.C. Zhang, G. Huang, Leveraging visual features and hierarchical dependencies for conference information extraction, in: Proc. APWeb, LNCS 7808, 2013, pp. 404–416.
[56] Z. Zhang, B. He, K.C.-C. Chang, Understanding Web query interfaces: best-effort parsing with hidden syntax, in: Proc. 2004 ACM SIGMOD International Conference on Management of Data, 2004, pp. 107–118.
[57] H. Zhao, W. Meng, Z. Wu, V. Raghavan, C. Yu, Fully automatic wrapper generation for search engines, in: Proceedings of the 14th International Conference on World Wide Web, 2005, pp. 66–75.
[58] Y. Zhai, B. Liu, Web data extraction based on partial tree alignment, in: Proceedings of the 14th International Conference on World Wide Web, 2005, pp. 76–85.
[59] Y. Zhai, B. Liu, Extracting Web data using instance-based learning, World Wide Web 10 (2) (2007) 113–132 (Jun. 2007).
[60] Q. Zhang, Y. Shi, X. Huang, L. Wu, Template-independent wrapper for web forums, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, pp. 794–795.
[61] S. Zheng, R. Song, J. Wen, Template-independent news extraction based on visual consistency, in: Proceedings of the 22nd National Conference on Artificial Intelligence, vol. 2, 2007, pp. 1507–1512.