Computer Science and Information Systems 11(1):111–131 DOI:10.2298/CSIS130416020G

Using XPaths of Inbound Links to Cluster Template-Generated Web Pages

Tomas Grigalis[1] and Antanas Čenys[1]

[1] Department of Information Systems,
Vilnius Gediminas Technical University,
Sauletekio av. 11,
LT–10223 Vilnius, Lithuania
{tomas.grigalis, antanas.cenys}@vgtu.lt

Abstract. Template-generated Web pages contain most of structured data on the Web. Clustering these pages according to their template structure is an important problem in wrapper-based structured data extraction systems. These systems extract structured data using wrappers that must be matched to only particular template pages. Selecting single type of template from all crawled Web pages is a time consuming task. Although there are methods to cluster Web pages according to their structural similarity, however, in most cases they are too computationally expensive to be applicable at Web-Scale. We propose a novel highly scalable approach to structurally cluster Web pages by employing XPath addresses of inbound inner-site links. We demonstrate the effectiveness of our method by clustering more than one million Web pages from many real world Websites in a few minutes and achieving >90% accuracy.

Keywords: Web data extraction, structural clustering, template-generated pages, wrapper induction.

1. Introduction

Many data intensive Web sites are backed by databases. Such databases contain data that is retrieved upon a Web page request and displayed on it. For example, news portals provide articles together with their title, date of publication, authors and constantly growing number of user generated comments. E-commerce shopping sites are full of products, their pictures, description, price, etc. All this information in stored in a database and is freshly retrieved only when a Web page visitor opens the page in his browser. In a same Web site each Web page with a news article or a product is structurally similar. While browsing news stories or products in a same Web site only required information related to the unique data record in a database is retrieved. As a result, a Web site visitor sees structurally identical Web pages filled with different information. Such Web pages are also called template-generated Web pages. Template-generated Web site has a limited number of templates which are used to automatically generate new instances of Web pages. Upon a Web page request Web server queries a database, retrieves particular data record, and embeds that information into a template, which in turn is returned to Web browser. So each time browsing the same template a structurally similar Web page is returned.

The structural similarity of template-generated Web pages is an important feature for information extraction. It is possible to write simple data extraction rules, also known as wrappers, for each kind of template, and later reuse these rules on all other Web pages of the same template. Wrapper systems for data extraction are commercially popular and are the subject of extensive research over the last two decades [1]. Much effort now is directed to extracting data at Web-Scale [2–5]. Since the Web has enormous amount of data embedded into billions of Web pages only highly efficient and unsupervised approaches seem to be feasible to extract structured Web data at Web scale.

An equally important, but less recognized outcome of the new problem definition of Web scale data extraction is the need to automatically organize pages of the same site into clusters, such that each cluster contains structurally similar template-generated Web pages [1]. Then using unsupervised wrapper generating technique a high quality wrapper can be inferred for each cluster of Web pages. Here again, only unsupervised and efficient method can be used to automatically identify and cluster Web pages. Alternatively, if any of these two steps, i.e. wrapper generation or structural Web page clustering require some human effort, then we cannot claim capability to extract data at Web-Scale. Even though, as noted in [1], there exist structured data extracting techniques [6–8], that explicitly do not require such Web page clustering, the latter can definitely help in organizing and synthesizing extracted data [9]. Furthermore, unsupervised structural clustering of Web pages can substantially improve the accuracy and recall of data extraction techniques.

As we see, unsupervised clustering of Web pages is equally important problem as wrapper induction and is studied in [1, 10–12]. However, most state-of-the-art structural Web page clustering techniques are purely content based and in most cases have at least quadratic running time complexity. The high dependency on Web page content analysis creates a fundamental issue: these techniques do not scale to large Websites [1]. Database generated Websites can have millions of Webpages and there could be thousands of those Websites that we are interested in clustering in a reasonable amount of time. Even though state-of-the-art system in XML clustering, the Xproj [13], has a linear complexity, it still requires an estimated time of more than 20 hours to cluster a site with million pages [1]. There is a need of less content-dependent techniques to cluster large amount of Web pages.

In this paper we present a novel scalable method to cluster template-generated Web pages according to their structural similarity. The scalability of duplicate content detection algorithms motivated us to come up with a system capable of structurally clustering template-generated Web pages without the need of constant pair-wise comparison of their content. Although there are highly efficient approaches [1, 14] to this problem, they usually rely on URL pattern recognition and are prone to unusual and unexpected formats of URLs. Contrary to them, we employ exact locations (XPaths) of inbound inner-site links to cluster same template-generated Web pages. We call our method *UXClust* (derived from *URL XPath Clustering*). The proposed UXClust method exploits the observation that each unique XPath location containing a link usually points to same template-generated Web pages.

The main contributions of our work are:

1. We study the problem of structurally clustering template-generated Web pages and propose a novel idea to exploit XPath addresses of inbound inner-site links to radically speed up clustering time.

2. We conduct extensive experimentations with more than 1 000 000 Web pages from fourteen real world Web sites and demonstrate that proposed UXClust approach is able to cluster them in less than 4 minutes achieving higher than 90% precision and recall.
3. We compare our UXClust approach to two state-of-the-art baseline algorithms and achieve order of magnitude speed improvement while still maintaining better precision and recall.
4. Since proposed method has very low computational complexity it is applicable in Web-Scale structural data extraction and Web page clustering systems.

1.1. Running example

We now take an example to demonstrate the overall working of our system. In Fig. 1 a set of two screenshots of visually rendered Web pages from argos.co.uk are shown. Although these two pages contain information about totally different products, i.e. a sunglass and a digital camera, they both share almost identical visual appearance. The regularity of template style and its optimization for humans make each unique location of inner site link to point to the same template Web page. If it was otherwise and there were no regularities, then humans would find it difficult to navigate such Web site.

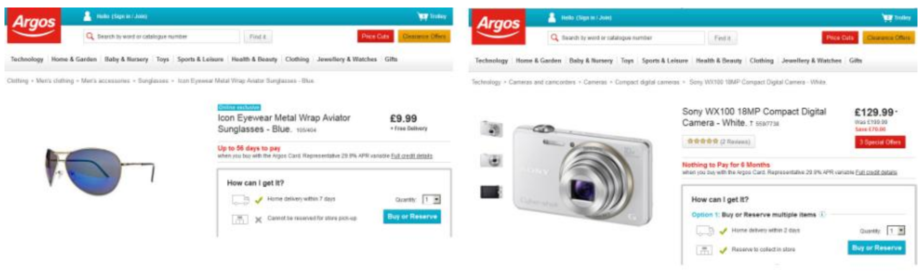

Argos
Technology | Home & Garden | Baby & Nursery | Toys | Sports & Leisure | Health & Beauty | Clothing | Jewellery & Watches | Gifts
Clothing › Men's clothing › Men's accessories › Sunglasses › Icon Eyewear Metal Wrap Aviator Sunglasses - Blue
Icon Eyewear Metal Wrap Aviator Sunglasses - Blue.
£9.99
Up to 56 days to pay
How can I get it?
Buy or Reserve
Argos
Technology | Home & Garden | Baby & Nursery | Toys | Sports & Leisure | Health & Beauty | Clothing | Jewellery & Watches | Gifts
Technology › Cameras and camcorders › Cameras › Compact digital cameras › Sony WX100 18MP Compact Digital Camera - White
Sony WX100 18MP Compact Digital Camera - White.
£129.99
Nothing to Pay for 6 Months
How can I get it?
Buy or Reserve

Fig. 1. An screenshot example of two same template-generate Web pages taken from argos.co.uk

The goal of our system is to group all Web site pages into clusters, such that each cluster contains only structurally similar pages that share the same template. To achieve this goal we exploit the available information about each links' exact location in a Web page template. As we know, HTML source code of Web pages can be represented as a tree structure where each tree node can be accessed by XPath query language expression. This way each Web page link has its unique address (XPath expression) in a Web page tree structure.

We take the XPath (e.g. */html/body/div[2]/ul/li[4]*) and the URL (e.g. *www.argos.co.uk/static/Browse/ID72)* of every Web page link and save this data as a tuple: (*/html/body/div[2]/ul/li[4]*, *www.argos.co.uk/static/Browse/ID72*). Links extracted from the same template Web pages have many identical XPath expressions. This way, we can cluster links according to their XPath addresses.

Then we determine to which template each unique XPath containing a link points to. We calculate structural similarity of Web pages by comparing their tree structure. Computationally intensive task of finding clusters of structural similar Web pages is done only once per unique XPath location in HTML tree. If we later encounter a link from the same location in a Web page, we know that it points to the already determined cluster.

1.2. Paper outline

The paper is organized as follows: related work is found in Section 2. Section 3 is dedicated to presenting proposed approach to structurally cluster Web pages. First of all, structural features of a modern Web page are presented. Then Web pages similarity measures are introduced. Finally, proposed method is described in detail. In Section 4 we describe the experimental evaluation process and compare proposed approach to two baseline algorithms. Conclusions are presented in Section 5.

2. Related Work

The idea that linkage among Web pages can reveal something about their similarity is not new. For example, Small [15] claims that the relationship between two documents can be calculated by analyzing how often they both are cited in the same source. Dean et al. [16] and Spertus [17] brings the same idea to the context of Web pages. Both works present methods to detect topically related pages without actually downloading and analyzing their content. Instead, proposed algorithms analyze Web pages for co-linkage and links placement. For example, densely placed links [17] or links having many same parent nodes [16] in HTML tree are considered to be linking to topically similar Web pages.

Crescenzi et al. [10] further demonstrate that the analysis of co-linkage and link placement holds true and for structural similarity of Web pages. They propose an algorithm that crawls given Web site and incrementally builds site model for it. Their work is closely related to focused-crawling [18, 19] research field where focused-crawling systems are usually built to crawl topically related Web pages, however, Crescenzi et al. look for structurally similarity. The structural similarity is also determined by comparing the placements of link collections. In other words, two pages are considered to be structurally similar if they both have groups of links placed in the same locations. One of the main observations they present and validate is that links sharing the same layout and presentation properties usually point to pages that are structurally similar. Same idea is also employed by Lin et al. [20] to hierarchically cluster Web pages.

In this paper we further extend these ideas to exploit cross linkage to infer Web page similarity. Our main observation is that different links from the exactly same position in HTML tree usually point to structurally similar Web pages. Contrary to the approach of Crescenzi et al. [10] where a group of densely placed links in the same page infer the similarity of linked Web pages, we state that links found in the exactly same XPath location and coming from different Web pages point to structurally similar Web pages. In other words, Crescenzi et al. [10] finds structurally similar Web pages by extracting their links from the same group of links in one page, while our approach requires to extract one link from each new page from exactly same XPath location.

To simplify and speed up structural comparison of Web pages Buttler [21] proposed to adapt the shingles based content similarity detection technique [22] to compare structural content of Web pages. In this adapted technique Web pages are viewed as a set of sequence of branches, paths from the HTML tree root node to a leaf node. In such way HTML tree can be encoded as a list of these tokens. The proposed path similarity measure finds the similarity of paths between two different Web pages.

Another early research field, related to Web document similarity detection, concentrates on XML (Extensible Markup Language) document and their schema comparison [1]. One widely adopted method, called tree edit distance, measures the minimum number of node insertions, deletions, and updates required to convert one XML document tree into another. This can be converted into a similarity metric by normalizing the number of edit operations with the number of nodes in the tree representing the larger document [21]. However, the computational complexity of techniques involving tree edit distance calculation [23–26] is at least quadratic. To overcome this limitation and reduce required running time Augsten et al. [27] propose approximating tree edit distance calculation technique called pg-grams. This technique, a reminiscent of Web page shingling method [22], splits XML tree into smaller trees called pq-grams and uses them to compare documents.

Joshi et al. [12] proposed an alternative scheme for representing the structural information of Web documents based on the paths contained in the corresponding HTML tree model. Since proposed model includes partial information about parents, children and siblings, it allows to derive meaningful and at the same time computationally simple structural similarity measure. Later Chakrabarti et al. [11] proved that in most cases it is not necessary to compare all XPaths of two documents to determine their structural similarity. Instead, they observed that each template style in a Web site has unique and important sections. For example, product page template can have product title displayed on same particular location in instance of that template. This way some HTML tree locations are more important than other and can aid in discerning one template from another. Authors name these important locations in HTML tree as “key” paths. The Webpages on the Website are then clustered using these “key” paths.

All above proposed Web page structural similarity measures are purely based on Web page content analysis and comparison. However, most efficient and scalable Web page clustering techniques depend on meta-data about pages, such as those based on Web page URL analysis and pattern search [1, 14]. Our own proposed method belongs to this category. However, instead of comparing URLs we compare XPath addresses of inbound-links in originating Web document.

3. Problem Definition and Proposed Approach

In this part of the paper we introduce structural features of Web pages, their similarity measures and propose a novel method to efficiently cluster Web pages according to their structural similarity.

3.1. Structural Features of Web Pages

Each Web page can be represented as a tree data structure which is a fundamental data structure in XML documents. See Fig. 2 for an example. Thus a very useful approach for extracting data, such as links, from Hypertext Markup Language (HTML) documents (Web pages) is to employ Extensible Markup Language (XML) technologies to translate HTML to valid XML code. In this approach, HTML Web pages are first normalized into Extensible HTML (XHMTL) and then then processed by XML applications [28].

```
<html>
  <head>
<title>Google</title>
<style>...</style>
  </head>
  <body>
    <div>...</div>
<center>...</center>
    <div>...</div>
  </body>
</html>
```

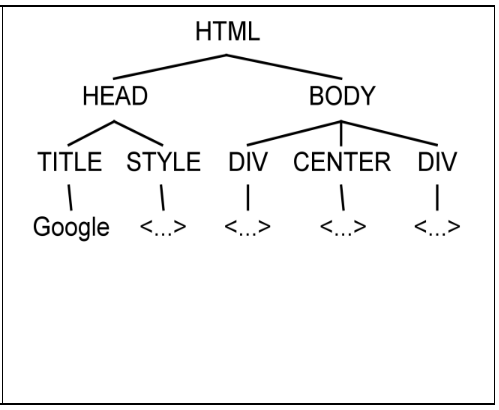


Fig. 2. HTML source code represented as tree structure

The modeled HTML tree and XPath language for selecting and extracting data is proved to be very robust technique useful in practical data extracting applications [6, 29, 30]. In the scope of this paper and for the clustering template-generated Web pages, we use XPath expressions to select inner-site links. During the link extraction process we preserve the original path from root node in the HTML tree to the particular link. This tree path is called a location path (XPath) and is later used in clustering stage of our proposed method.

Another important feature of modern Web pages is the widespread use of Cascading Style Sheets (CSS) language [31] that allows authors and users to attach style (e.g., fonts and spacing) to structured documents. By separating the presentation style of Web pages from their content, CSS simplifies Web authoring and site maintenance. Furthermore, CSS absolute positioning property enables Web designers to put each individual page element in specified locations regardless of the original position in HTML source code. Recall, that we exploit the XPath positions of inbound links to cluster Web pages. Naturally, some questions arise: what if links are positioned with CSS properties, would our proposed UXClust approach still be valid? Our observations indicate, that even most technologically sophisticated contemporary Web sites that fully utilize CSS, such as amazon.com, cnn.com, bbc.co.uk, etc., still have big source code HTML pages and thus big HTML trees. Even if links are positioned by CSS they still have unique XPath address in HTML tree. Unique XPath locations are all we need to enable the working of our proposed UXClust approach.

3.2. Structural Similarity of Web Pages

Structural similarity of Web pages describes how similar is their HTML source code. Since we are only interested in structural similarity of template-generated Web pages the actual text that does not belong to HTML markup code is not taken into account when calculating similarity. In other words, the textual content that is visible to ordinary Web site browser is the object of more importance to a related research field, i.e. duplicate or near duplicate Web pages detection [22]. So in this Section we are going to describe a few similarity measures that are used to compare structural similarity of Web pages and not their textual content. These methods and measures often rely on HTML tree and XPaths.

For notation purposes we assume P_1 and P_2 to be two HTML Web pages, T_i to be a ordered tree structure of corresponding HTML code, and N_i to be a set of HTML tree nodes of which L_i are leaf nodes, and $i=\{1,2\}$. Leaf nodes of HTML tree are those nodes that themselves do not have any children nodes.

As noted in the previous Section of this paper, each Web page can be viewed as an ordered tree structure. So the most straightforward technique to calculate similarity of two Web pages is to calculate the cost of transforming one tree structure into another. For this purpose basic operations like inserting, deleting, replacing or moving individual tree nodes or entire sub-trees in the tree structure are associated with certain costs to perform these operations. In case of structural trees created from HTML markup code, the problem is usually a bit more simple, since the root node is known, the sibling nodes are ordered and as the sub-trees (especially when documents are same template-generated) are hardly ever changing their distance to the root node [32]. This can be used as a similarity metric by normalizing the number of edit operations with the number of nodes in the tree representing the larger Web page [21]. If, as noted before, P_1 and P_2 are two Web pages to be compared, *editDistance*() is a function that calculates basic Web page tree operations (like inserting, deleting, replacing) required to transform P_1 to P_2 and *max*() is a function returning the biggest number, then tree edit distance (TED) can be formalized into the following formula at (1):

$$\mathrm{TED}(P_1,P_2)=(\ \mathrm{editDistance}(P_1,P_2)\ )\ /\ (\ \max(|N_1|,|N_2|)\ ) \qquad \mathbf{(1)}$$

However, basic tree edit distance algorithms [23–26] have a big drawback – they do not scale well, because they have at least a linear dependence on the size of each HTML tree and quadratic dependence on the combined size of the two trees. A faster method to compare Web pages is to use the *pq-gram distance* [27], which approximately match ordered labeled trees. The pq-grams of a tree are all its sub trees of a particular shape. Intuitively, two trees are close to each other if they have many pq-grams in common. For a pair of trees the pq-gram distance can be computed in $O(n \log n)$ time and $O(n)$ space, where n is the number of tree nodes [33]. For $p > 0$ and $q > 0$, the pq-gram distance, $\mathrm{pqGDist}(T_1,T_2)$, between two trees T_1 and T_2 having corresponding sets of pq-grams $P^{p;q}(T_1)$ and $P^{p;q}(T_2)$,is defined as follows [27]:

$$\mathrm{pqGDist}(T_1,T_2) = 1 - 2\ (\ |P^{p;q}(T_1) \cap P^{p;q}(T_2)|\ /\ |P^{p;q}(T_1) \cup P^{p;q}(T_2)|\ ) \qquad \mathbf{(2)}$$

Another way to measure structural similarity between two Web pages is to compare tag paths of each leaf node [12]. Leaf nodes are those nodes in HTML tree that do not have any children. Tag path of such node is concatenated string of tags name leading from root node to the particular leaf node. Such concatenated string can be acquired by calculating absolute XPaths of particular leaf node. So in a such way each Web page P_i can be represented as a bag of XPath strings, formally by set $xp(P_i)$ of strings. A common paths distance (CPD) measure can be calculated via intersection of two XPath sets $xp(P_1)$ and $xp(P_2)$ generated from two Web pages P_1 and P_2. See the following formula [32]:

$$CPD(P_1,P_2) = 1 - |\, xp(P_1) \cap xp(P_2) \,| \,/\, \max(\, |xp(P_1)| \,,\, |xp(P_2)| \,) \qquad (3)$$

3.3. Proposed Approach

In this Section of the paper we in detail discuss our proposed approach to cluster structurally similar Web pages. See Fig. 3 where a general architecture of proposed system is presented. The overall working process consists of four main steps: Web site crawling, link extraction, URL and XPath tuple generation, first step approximate clustering, and final clusters refinement stage. In the following paragraphs we describe each of these steps with more details.

Web site crawling step is used to collect Web pages from given Web sites. The process begins with providing to the system a seed URL from which it begins crawling process. We utilize breadth-first crawling strategy, where the algorithm recursively follows collected hyperlinks. The priority to follow and download is given to first seen URLs, in other words, first-in-first-out method is used. The main result of this Web site crawling step is downloaded documents that are now ready to be processed.

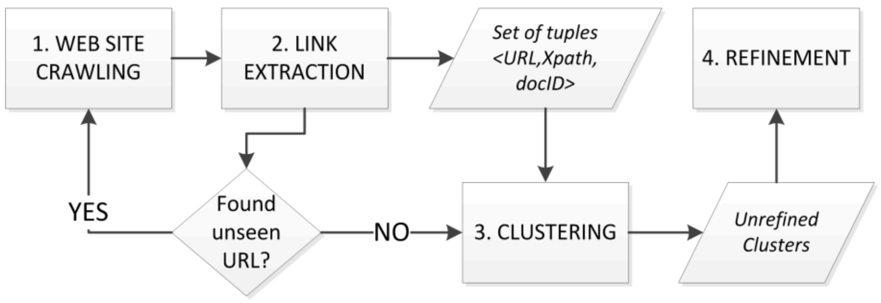
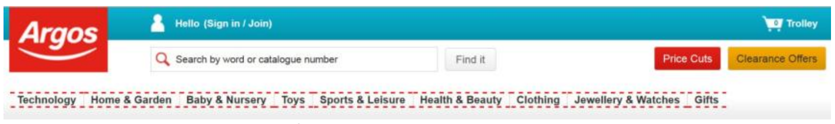


Fig. 3. A sample set of XPaths from a Web page

Each downloaded Web page is processed to create a HTML tree from the source code. This is done to simplify link extraction: we execute XPath expressions to select URLs. Of course, a more straightforward approach would be to utilize regular expressions to extract links; however, we are interested in obtaining not only URLs, but also and their location in the document tree. Each unseen URL is forwarded to Web site crawling process and saved into a tuple set consisting of extracted URL, XPath location of URL in originating document, and ID of downloaded page. For an example, see Fig. 4(a) where a snippet of rendered Web page from argos.co.uk is shown. This snippet contains a list of menu links pointing to different categories of the site, such as “Technology”, “Home & Garden“, etc. Link collection process extracts these links together with additional data describing the link: its XPath and the URL itself. Each tuple also gets and *DocID* which later lets identify downloaded page and find corresponding tuple.

It's worth noting that during the Web site crawling process we download only unseen unique links. In case a page contains several XPath locations with the same link (URL) we extract only the first one, i.e. the one located in the top post part of the Web page tree compared to the other locations. Only unique URLs are downloaded and later forwarded to the clustering process.



(a) a snippet of rendered Web page with highlighted links

LINK TEXT	URL	XPATH	DOCID
Technology	http://www.argos.co.uk/...6169.htm	/html/body/div[4]/ul/li[1]/a	1
Home & Garden	http://www.argos.co.uk/...5908.htm	/html/body/div[4]/ul/li[2]/a	2
Baby & Nursery	http://www.argos.co.uk/...5732.htm	/html/body/div[4]/ul/li[3]/a	3
Toys	http://www.argos.co.uk/...6252.htm	/html/body/div[4]/ul/li[4]/a	4
<...>			

(b) a sample list of generated tuples

Fig. 4. An example of tuple generation process

The main task of clustering in this first stage (marked number 3 in Fig. 3) is to group downloaded pages in a way that each group contains only pages whose URL originate in the same XPath location in HTML tree. In this stage we do not analyze the content of downloaded pages. Only associated tuples are used to cluster those pages. To be more precise, we simply group pages according to their inbound link XPath address. For example, if we have three XPaths [*/html/body/div[2]/a*, */html/body/div[2]/a*, */html/body/p[2]/center/a*] then we will have two resulting clusters. One would contain [*/html/body/div[2]/a*, */html/body/div[2]/a*] and the other [*/html/body/p[2]/center/a*]. So any resulting cluster contains only links with identical XPaths, i.e. links extracted from the same location.

As it is discussed before, template-generated Web pages have visual and structural regularity. Links in same template-generated Web pages appear on the same locations and have same XPaths. Furthermore, each template based and database backed Web site has only a limited number of different templates. This way, only a limited number of unique locations in HTML tree containing links exist. So grouping Web pages by XPaths of originating URL location is enough to approximately cluster them according to their structural similarity. Another important outcome is that the computational complexity of this clustering technique is very low compared to two other baseline methods (see Table 2 in Experimental Evaluation Section of this paper).

The previous clustering process clusters Web pages according to their inbound-links XPaths. Depending on the design of a Web site template there could be from tens to a few hundred different XPaths with links, and Web pages are grouped into the same amount of clusters. We call these clusters as *unrefined clusters*, because the clustering process takes into account only XPaths of inbound links and do not compare the structural similarity of Web pages. It means that Web pages belonging to separate clusters, indeed, can be structurally identical. Consider for example the list of links and their XPath locations in Fig. 4. Although all those links have different XPath address, they all point to structurally identical Web pages. These Web pages are same template-generated and display list of subcategories in any of main categories, such as Toys, Technology, Gifts, etc.

Depending on Web site template design there could be from a few hundreds to a few thousands unrefined clusters. During pilot experiments we observed that many unrefined clusters contain only a small amount of URLs. These URLs may often be found in other unrefined clusters, where thousands of unique URLs are located. These minor unrefined clusters may point to a top ten products page, news pages, policy pages, "about us" pages, etc. As it is obvious, these kinds of pages usually do not contain important structured data or that the data is redundant like in top ten products page example. So we decided to introduce *CUT-OFF threshold (*we use value of 100*)* to remove some unrefined clusters. This threshold is a limit of minimum unique URLs per cluster. If any cluster contains less than a cut-off threshold amount of links, it is removed. This way only high quality, many URL containing unrefined clusters are pushed to the next final stage.

The final refinement stage of the proposed method takes a predefined (we use value of 5) amount of sample pages from each of unrefined clusters. The content of sampled Web pages is analyzed to generate a common template fingerprint of that particular cluster. The idea to use Web page fingerprints instead of direct HTML tree comparison comes from closely related research field addressing duplicate content detection on the Web. The main idea there is to detect identical or near duplicate Web pages by analyzing their structure and content. These works [22, 34] are motivated by the fact that Web contains many Webpages on different domains with identical or near identical content. A user searching the Web is only interested in unique content on each Web page from search results list and duplicated content is no use for him. So search engines try hard to remove similar Web pages from occurring in a search results. Since Web pages come from thousands of domains and there could be literally millions of Web pages to compare each against another a scalable technique was developed to be able to cope with such big amount of comparisons. However, we do not directly utilize Web page shingling [22] and min-hashing [34] techniques to generate the fingerprints. Although they are very scalable but, on the other hand, they also are very approximating, i.e. inner-site template differences can be ignored and whole site can be seen as one template. So instead shingling and min-hashing to generate a fingerprint we employ bag of paths method [12]. For each sampled Web page from a cluster we extract all HTML tree paths that do not contain text. Then paths occurring only above threshold value of 0.3 are taken to be included into a fingerprint. So our version of Web page fingerprint is a set of selected tag paths.

Generated fingerprints for each template of clustered Web pages are used to detect similarities among clusters. As discussed above, two or more clusters may actually contain same template-generated Web pages. If such similarity among two clusters is detected and it is greater than predefined (we use value of 0.8) threshold value, the refinement process merges those two clusters and forms a new one. The process is repeated until no new clusters can be formed. Here basically we employ the union-find algorithm.

4. Experimental Evaluation

In this Section of the paper we experimentally evaluate proposed approach to structurally cluster template-generated Web pages. We also compare our proposed approach to two baseline methods: common XPaths [12] and pq-grams [27]. Since, to the best of our knowledge, there is no benchmark dataset suitable for our system, we need to create it. Most of datasets containing crawled Web pages at best contain saved Web pages and their URLs. However, our proposed approach utilizes XPath addresses of inbound inner-site links. So in addition to physically saved Web pages we also need to have those XPath addresses. And as noted before, no dataset can provide that necessary meta-data about saved Web pages. Experiments were conducted on laptop computer with Ubuntu 12.04 operating system, Intel® Core™ i7-2670QM CPU @ 2.20GHz, 8 GB RAM, 7200 RPM hard drive.

4.1. The Dataset

To create a dataset consisting of Web pages with required additional information about inner-linkage we programmed a basic Web site crawler. We utilize a breadth-first crawling approach where all neighboring Web pages are crawled first. This is contrary to a depth-first crawling which prioritizes exploring Web site as far as possible along each linking branch before backtracking. It is demonstrated [35] that traversing the Web graph in breadth-first search order is a good crawling strategy, as it tends to discover high-quality pages early in the crawl. Implemented crawler does not uses cookies and is single-threaded, thus it is suitable to download one page from one site at a time. However, during the crawling process we ran multiple instances of the crawler – each for different Web site. The crawling algorithm encompasses a URL cleaning procedure, which removes forced session id tokens, such as *phpsessid, cfid, aspsessionid*, etc.

We chose 14 Websites containing structured data in template-generated Web pages. In Table 1 data about each Website is presented. The data include Web site address, total amount of downloaded pages, total size of Web pages in gigabytes, average size of downloaded Web page in kilobytes, total amount of unique XPath locations containing inner-site links, and total amount of unique XPath locations with cut-off threshold applied, i.e. XPath locations containing more than 100 unique inner-site links. The last row contains aggregated data among all Web sites. As we can see from the table, more than one million Web pages where downloaded totaling in size of 119 gigabytes on disk. Average size of single Web page among all Web sites is 99 kilobytes. A more complicated design of a Web site results in more XPath locations where inner-site links can be found. Some sites, such as argos.co.uk and bigbox.lt, have such sophisticated design that there are round five thousands XPath locations. Each such XPath corresponds to one unrefined cluster. To simplify similarity calculation among such big amount of clusters and to reduce running time, we introduced a cut-off threshold. The cut-off threshold lets us to disregard all clusters that have less than predefined threshold limit of unique pages. In our experiments the threshold was equal to 100. In the last column of the table a number of filtered clusters are listed.

All data about each downloaded Web page is stored in a relational MySQL database. One table is dedicated to store all data: URL, XPath, DocID, and Web site ID.

Downloaded Web pages are stored in file folders. There is one folder for each Web site. The file names of stored Web pages are identical to DocIDs, which are stored in a database table. Running time for structural clustering of saved Web pages includes database queries to retrieve and save data.

Table 1. Basic data about each Web site used in experimental evaluation

Nr.	**Web site**	**Pages**	**Total size in GB**	**Average Size in KB**	**URL XPaths**	**URL XPaths (*cut-off*)**
1.	argos.co.uk	111142	7.33	69.20	4472	224
2.	azon.lt	102313	18.28	187.32	306	86
3.	bigbox.lt	120557	27.03	235.12	5055	145
4.	citylights.com	13698	0.34	26.29	553	19
5.	currys.co.uk	9993	0.67	70.42	1078	9
6.	elshop.lt	90001	2.27	26.44	246	36
7.	ikea.com	122686	10.16	86.83	1754	117
8.	ilterzogirone.it	66404	3.64	57.55	1317	109
9.	imk.lt	74295	14.46	204.14	1864	116
10.	iristorante.it	116410	5.35	48.21	1196	119
11.	kompiutera.lt	21623	1.08	52.24	368	35
12.	smartbuy.lt	15638	0.41	27.62	441	29
13.	tesco.com	88773	15.64	184.76	3618	192
14.	varle.lt	117514	12.44	110.96	875	61
	Average:	**76503**	**8.51**	**99.08**	**1653**	**92**

4.2. Ground Truth and Measurements

For each Website we identified one kind of template with most interest to us. In many cases these are template-generated pages containing product data, such as title, price, picture, description, etc. Then we manually analyze URLs of these pages to identify repeating patterns, such as keywords or URL structure. A regular expression is written for each Web site to match those URLs with high interest to us. This way we generate a ground truth (golden data) that is used to calculate precision, recall, of our proposed clustering method. The precision of a given cluster is the fraction of Web pages in its computed cluster that are also found in the corresponding ground truth cluster. The recall of a given cluster is the fraction of Web pages from the corresponding ground truth cluster that were grouped into the same computed cluster. To calculate these measures, that are taken from information retrieval research field, we first need to describe all possible outcomes of clustering process. Our main goal is to assign two or more Web pages to the same cluster if and only if they are similar. A true positive (TP) decision assigns two structurally similar Web pages to the same cluster, a true negative (TN) decision assigns two structurally dissimilar Web pages to different clusters. There are two types of errors that can occur. A false positive (FP) decision assigns two structurally dissimilar Web pages to the same cluster. A false negative (FN) decision assigns two structurally similar Web pages to different clusters. Then the precision (P), recall (R) is calculated as follows:

$$P = TP / (TP+FP); \quad R = TP / (TP+FN) \tag{4}$$

4.3. Selecting Parameters for Two Baseline Algorithms

Since pg-Grams and Common Paths (CP) methods are very computationally expensive and result in a long running time (for detailed comparison of running times please see Table 3), we decided to take a much smaller amount of Web pages from each Web site to benchmark them. First, we test the two baseline algorithms with the first Web site from the dataset, namely with *argos.co.uk*. We check if the different number of selected Web pages has any significant impact on precision and recall. As seen in Fig. 5 and Fig. 6, there is no significant difference on precision and recall with both algorithms if we select a different number of Web pages.

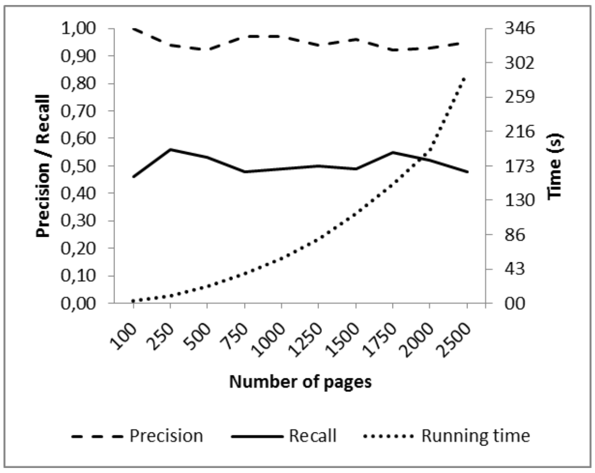


Fig. 5. The effect on recall, precision and running time by selecting different numbers of pages for Common Paths algorithm

However, there is a significant difference in terms of running time: the more pages are selected to cluster, the longer the running time. Actually, the running time increases almost exponentially with both algorithms.

Thus for pg-Grams we take only 100 Web pages from each Web site and use p=2 and q=3 values as suggested in original paper [27]. Similarly, for a faster Common Paths method we take 1000 Web pages. The amount of taken Web pages is also indicated in parentheses next to the algorithm name in Table 2 and 3. For our UXClust method we take all Web pages as seen in Table 1. This way, of course, the size of ground truth clusters differs among three benchmarked methods. The *GT-P* columns under each method in Table 2 indicate the number of ground truth pages for each Web site. For each of three tested methods we use the same 0.8 Web page similarity threshold value.

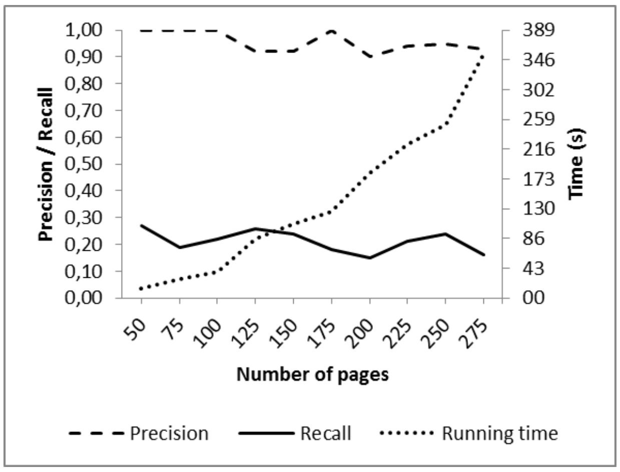


Fig. 6. The effect on recall, precision and running time by selecting different numbers of pages for pg-Grams algorithm

4.4. Results

Table 2 lists the precision (P) and recall (R) of applying our proposed (UXClust) and two baseline (pg-Grams and Common Paths) clustering algorithms on each Web site.

Table 2. The results of using the proposed and two baseline algorithms to cluster structurally similar Web pages

Nr.	**Website**	**UXClust**			**pg-Grams (100)**			**CP (1000)**		
		GT-P	*P*	*R*	*GT-P*	*P*	*R*	*GT-P*	*P*	*R*
1.	argos.co.uk	30460	0.92	0.72	40	1	0.15	332	0.94	0.49
2.	azon.lt	36643	1	1	34	1	0.32	370	1	0.87
3.	bigbox.lt	10282	0.68	0.94	13	0.88	0.54	176	0.8	0.96
4.	citylights.com	2026	1	0.64	22	0.33	0.36	190	1	0.56
5.	currys.co.uk	2009	1	0.81	56	1	0.3	627	1	0.45
6.	elshop.lt	22019	1	1	22	1	0.55	244	1	0.76
7.	ikea.com	4909	1	1	9	1	0.56	99	1	1
8.	ilterzogirone.it	2866	1	1	10	1	0.7	49	1	0.98
9.	imk.lt	10173	0.96	1	16	0.82	0.56	191	0.95	0.93
10.	iristorante.it	967	1	0.9	4	1	1	35	1	0.74
11.	kompiutera.lt	11440	0.54	1	41	0.44	1	528	0.53	1
12.	smartbuy.lt	3632	1	1	36	1	0.25	396	1	0.99
13.	tesco.com	13066	0.97	0.96	30	0.81	0.83	274	0.79	0.15
14.	varle.lt	59450	1	1	48	1	0.73	483	0.98	0.98
	Average:	**14996**	**0.93**	**0.93**	**27**	**0.88**	**0.56**	**285**	**0.93**	**0.78**

As seen in Table 2, our proposed UXClust method achieves 0.93 average precision and recall and outperforms two other baseline methods. The pg-Grams algorithm performs worst. We believe it is possible to tune up this algorithm to achieve better results with structurally sophisticated Web pages, however, it is beyond the reach of this paper. Common Paths and our UXClust approaches achieve the same precision as in both of them the same Jaccard similarity measure over the XPaths of Web pages is used to determine structurally similar templates. However the use of XPath clustering in UXClust significantly increases recall from 0.78 to 0.93.

As we further look to the results of proposed UXClust approach we see that on almost all Web sites it achieves very high recall, except for the site number 4 (citylights.com). Here the recall result is just 0.64. A closer look relieved, that this happens due to unexpected behavior of citylights.com template engine. Each time a requested URL resource does not contain corresponding entry in underlying database, the Web server returns a “sorry” message (as a small inclusion) and displays a regular Web page with list of books. The template of this page is different from the one that should be returned if book data would be found. The only way to be sure of returned template is to verify each response. However, parsing each downloaded page takes a lot of time, and our proposed method tries to avoid it. That way, the clustering recall of the citylights.com Web pages cannot be further improved using our proposed method. We consider this site to be an exception.

Table 3. Running time comparison between proposed and two baseline methods

Nr.	**Website**	**UXClust**		**pg-Grams (100)**		**CP (1000)**	
		Time (s)	*GT-P/s*	*Time (s)*	*GT-P/s*	*Time (s)*	*GT-P/s*
1.	argos.co.uk	31	982	49	0.82	61	5.44
2.	azon.lt	12	3053	66	0.52	71	5.21
3.	bigbox.lt	35	293	123	0.11	190	0.93
4.	citylights.com	01	2026	21	1.05	26	7.31
5.	currys.co.uk	01	2009	61	0.92	67	9.36
6.	elshop.lt	05	4403	16	1.38	37	6.59
7.	ikea.com	18	272	65	0.14	89	1.11
8.	ilterzogirone.it	12	238	70	0.14	38	1.29
9.	imk.lt	25	406	108	0.15	153	1.25
10.	iristorante.it	13	74	50	0.08	46	0.76
11.	kompiutera.lt	05	2288	54	0.76	100	5.28
12.	smartbuy.lt	02	1816	21	1.71	31	12.77
13.	tesco.com	56	233	165	0.18	229	1.20
14.	varle.lt	17	3497	83	0.58	132	3.66
	Total:	**235**	**893**	**950**	**0.40**	**1271**	**3.14**

The precision of UXClust drops when Web pages from different templates are clustered together. This happens on 6 out of all 14 Web sites. The main difficulty here is to determine the similarity of two Web pages. Our experiments revealed that the bag of XPaths approach (Common Paths), which we use in our algorithm, is not always working on modern Web pages. In some sites, there are very small differences between different templates. Furthermore, a template of a product page may differ a little depending on the product type. In those cases only a human viewer could tell if those two pages are similar and the similarity may be more semantic one, than a visually structural.

In Table 3 we compare the running time of our proposed and two baseline methods. Recall that each method in run with different amount of Web pages. Thus we calculate two scores: the whole running time in seconds and normalized measure of ground truth pages clustering speed per second (GT-P/s). The latter reveals how quickly ground truth cluster is formed. As we see in Table 3, our proposed method outperforms two other approaches by order of magnitude: UXClust is able to cluster 893 ground truth pages per second, while Common Paths method clusters just 3.14 ground truth pages per second and pg-Grams – just 0.40. Proposed UXClust approach is able to structurally cluster more than a million Web pages in just 235 seconds.

4.5. Thresholds impact on overall precision, recall and running time

We also investigate the trade-offs between precision, recall and running time. To do that we vary values of all four thresholds used in proposed UXClust method, i.e. the Web page similarity threshold, XPath occurrence threshold, XPath sampling threshold and CUT-OFF threshold (for a detailed explanation of each threshold please refer to 3.3 Section of this paper). We show the running time curve only with those thresholds that have considerable effect on running time.

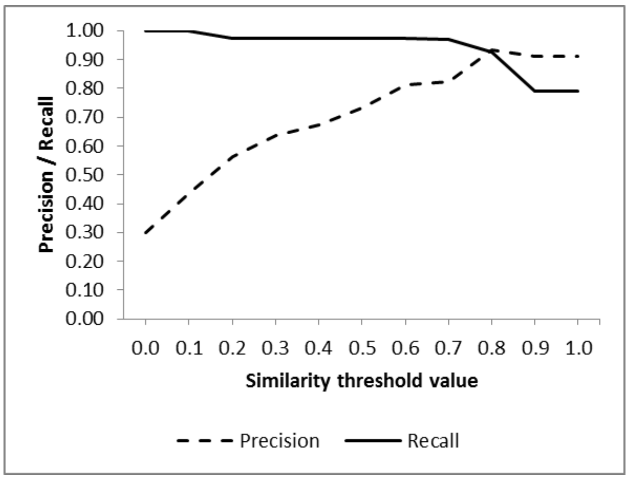


Fig. 7. Similarity threshold parameter effect on average precision and recall

In Fig. 7 we see what happens to average precision and recall on all Web sites when Web page similarity threshold increases from 0 to 1. As we see, precision starts from around 0.3 and reaches its maximum at around 0.9, while recall starts from 1 and drops to 0.8. The precision and recall intersects when similarity threshold is around 0.8. This is the exactly same value of Web page similarity threshold that is used in our proposed algorithms.

While conducting experiments we also noticed that precision and recall for each Web site may differ a few percent on each new run of our algorithms. This happens because Web page sampling algorithm uses random function to select a few samples of Web pages to calculate their fingerprint. Depending on sampled Web pages the resulting fingerprint may differ a bit. This leads to different similarity calculation results in clusters refinement stage. However, those differences on precision and recall are no more than a few percent. Move over the similarity threshold has no effect on running time.

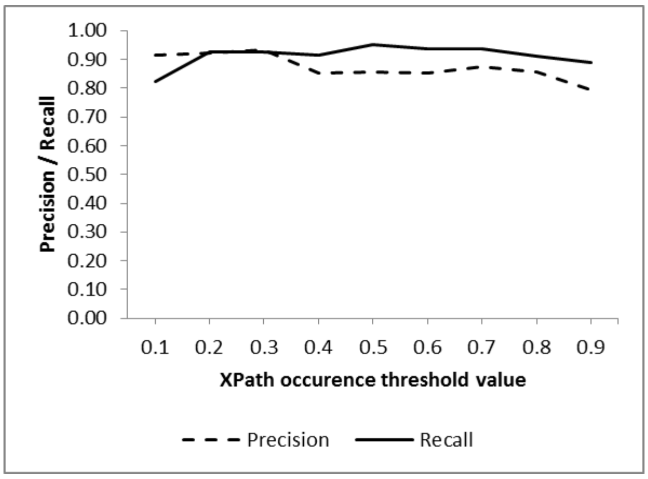


Fig. 8. XPath occurrence threshold parameter effect on average precision and recall

As shown in Fig. 8 the best results in terms of precision and recall are achieved when setting the XPath occurrence threshold between 0.2 and 0.3.Varying XPath occurrence threshold value has small effect on precision and recall. These results mean that in clusters fingerprint generation process there is a small difference between selecting all found unique XPaths from sampled Web pages or including just most common ones. This threshold also does not have any effect on running time, thus we do not include running time axis.

As seen in Fig. 9 the more Web pages are taken into sampling (higher threshold value) the longer UXClust method runs. Actually, we can see exponential growth in running time as XPath sampling threshold parameter increases. While selecting appropriate value for the threshold the results of recall and precision must be taken into considerations. Here we see the best fit at 3 where running time is still low and precision and recall are above 0.9.

CUT-OFF threshold value determines the smallest size of XPath clusters that are taken into Web page structural clustering process. Naturally, as seen in Fig. 10, the bigger the threshold, the less XPath clusters are compared to be merged, the less the running time. However, if we would select enormously high value of this threshold – depending on the Web site, there could be no clusters left to merge and thus large part of Web pages would be discarded from clustering. Our observation demonstrates that CUT-OFF threshold value of 100 lets to achieve good results.

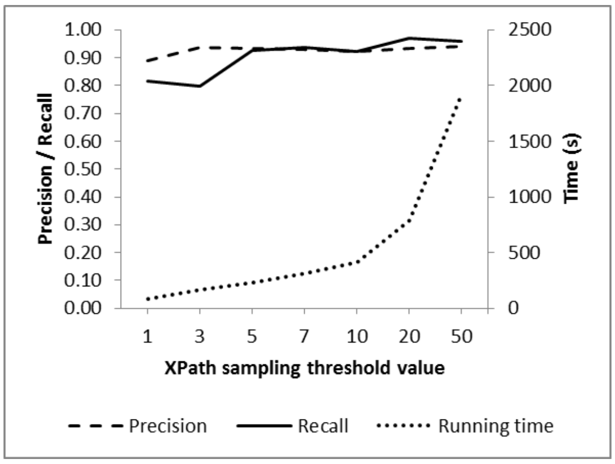


Fig. 9. XPath sampling threshold parameter effect on precision, recall and running time

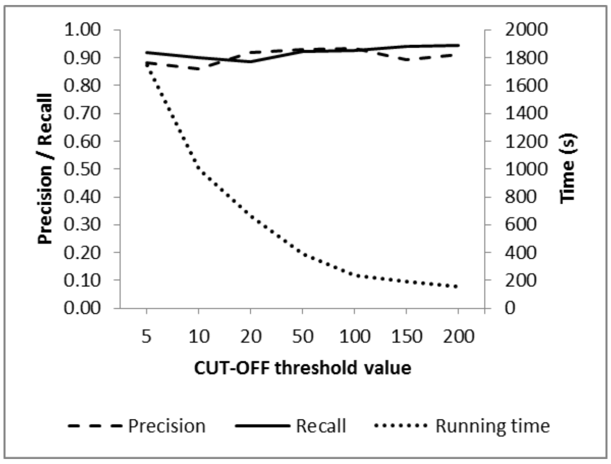


Fig. 10. CUT-OFF threshold parameter effect on precision, recall and running time

5. Conclusions

In this paper we presented a novel completely unsupervised method to structurally cluster template-generated Web pages. The method leverages XPath locations of inner-site links to drastically speed up clustering process. Using this approach more than a one million of Web pages are clustered in less than 4 minutes. In addition to speed efficiency, the proposed method achieves > 90% precision and recall.

Even though these results are very encouraging, there remain a few open challenges to be solved:

(1) Web page similarity measurement is a bottleneck of precision and recall in our algorithms. Experimental evaluation of our proposed algorithms revealed that current

Web page structural similarity measures find it difficult dealing with contemporary Web pages, which size, in fact, can reach >200 KB of HTML code. And the real differences between two same site templates can be traced to just a few lines of code. We stipulate that there is need to include semantic analysis step to better understand the purpose of each template in a site. This could help to identify key locations or text strings that discern one template from another. Only then it will be possible to further increase precision and recall of structural clustering process.

(2) Since our approach employs XPath locations of URLs, there is need to extend current Web crawlers to save this kind of additional information about downloaded Web pages. We could not identify any publicly available Web crawler with such functionality.

(3) There is a need to come up with methods dealing with unexpected Web server behavior. For example, we ran into a problem when some products are somehow "turned off" or not found in a database but their URL remains valid. Then Web server quietly redirects request to Web pages of different template, such as category list. So some kind of procedures detecting those unexpected behaviors should be employed.

References

1. Blanco, L., Dalvi, N., Machanavajjhala, A.: Highly efficient algorithms for structural clustering of large websites. Proc. of the 20th international conference on World wide web. pp. 437–466. ACM Press, New York, New York, USA (2011).
2. Bohannon, P., Dalvi, N., Filmus, Y.: Automatic web-scale information extraction. Proc. of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 609–612 (2012).
3. Gulhane, P., Madaan, A., Mehta, R.: Web-scale information extraction with vertex. IEEE International Conference on Data Engineering (ICDE). pp. 1209–1220 (2011).
4. Cafarella, M.J., Halevy, A., Wang, Z.D., Wu, E.: WebTables : Exploring the Power of Tables on the Web. International Conference on Very Large Data Bases. pp. 538–549 (2008).
5. Elmeleegy, H., Madhavan, J., Halevy, A.: Harvesting relational tables from lists on the web. VLDB J. 20, 209–226 (2011).
6. Zhai, Y., Liu, B.: Structured Data Extraction from the Web Based on Partial Tree Alignment. IEEE Trans. Knowl. Data Eng. 18, 1614–1628 (2006).
7. Liu, W., Meng, X., Meng, W.: ViDE : A Vision-Based Approach for Deep Web Data Extraction. IEEE Trans. Knowl. Data Eng. 22, 447–460 (2010).
8. Grigalis, T., Radvilavičius, Čenys, A., Gordevičius, J.: Clustering visually similar web page elements for structured web data extraction. Web Eng. 435–438 (2012).
9. Nguyen, H., Fuxman, A., Paparizos, S.: Synthesizing products for online catalogs. In Proc. of the VLDB. pp. 409–418 (2011).
10. Crescenzi, V., Merialdo, P., Missier, P.: Clustering Web pages based on their structure. Data & Knowledge Engineering. 54, 279–299 (2005).
11. Chakrabarti, D., Mehta, R.: The paths more taken: matching DOM trees to search logs for accurate webpage clustering. Proc. of the 19th international conference on World Wide Web. pp. 211–220 (2010).
12. Joshi, S., Agrawal, N., Krishnapuram, R., Negi, S.: A bag of paths model for measuring structural similarity in Web documents. Proc. of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 577–582. ACM Press, New York, New York, USA (2003).

13. Aggarwal, C., Ta, N., Wang, J.: Xproj: a framework for projected structural clustering of xml documents. In Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 46–55 (2007).
14. Hernández, I., Rivero, C.R., Ruiz, D., Corchuelo, R.: A statistical approach to URL-based web page clustering. Proc. of the 21st international conference companion on World Wide Web. pp. 525–526. ACM Press, New York, New York, USA (2012).
15. Small, H.: Co-citation in the scientific literature- A new measure of the relationship between two documents.pdf. J. Am. Soc. Inf. Sci. 4, 28–31 (1973).
16. Dean, J., Henzinger, M.: Finding related pages in the World Wide Web. Comput. networks. 11, 1467–1479 (1999).
17. Spertus, E.: ParaSite: Mining structural information on the Web. Comput. Networks ISDN Syst. 587–595 (1997).
18. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific Web resource discovery. Comput. Networks. 31, 1623–1640 (1999).
19. Diligenti, M., Coetzee, F.M., Lawrence, S., Giles, C.L., Gori, M.: Focused Crawling Using Context Graphs. In Proc. of the VLDB. pp. 527–534 (2000).
20. Lin, C., Yu, Y., Han, J., Liu, B.: Hierarchical web-page clustering via in-page and cross-page link structures. Adv. Knowl. Discov. Data Min. 222–229 (2010).
21. Buttler, D.: A short survey of document structure similarity algorithms. The 5th International Conference on Internet Computing. pp. 3–9 (2004).
22. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the Web. Comput. Networks ISDN Syst. 29, 1157–1166 (1997).
23. Nierman, A., Jagadish, H.: Evaluating structural similarity in XML documents. WebDB. 2, 61–66 (2002).
24. Zhang, K., Shasha, D.: Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. SIAM J. Comput. 18, 1245–1262 (1989).
25. Tai, K.: The tree-to-tree correction problem. J. ACM. 422–433 (1979).
26. Demaine, E., Mozes, S.: An optimal decomposition algorithm for tree edit distance. Autom. Lang. Program. 146–157 (2007).
27. Augsten, N., Böhlen, M., Gamper, J.: Approximate matching of hierarchical data using pq-grams. In Proc. of the VLDB. pp. 301–312 (2005).
28. Myllymaki, J., Jackson, J.: IBM Research Report Robust Web Data Extraction with XML Path Expressions. Tech. Report, IBM. (2002).
29. Simon, K.: ViPER: augmenting automatic information extraction with visual perceptions. ACM Int. conference on Information and knowledge management. pp. 381–388 (2005).
30. Hong, J.L., Siew, E.-G., Egerton, S.: Information extraction for search engines using fast heuristic techniques. Data Knowl. Eng. 69, 169–196 (2010).
31. Bos, B., Celik, T., Hickson, I., Lie, H.L.: CSS Level 2 Revision 1. 1, 1–487 (2011).
32. Gottron, T.: Clustering template based web documents. Adv. Inf. Retr. 40–51 (2008).
33. Augsten, N., Böhlen, M., Gamper, J.: The pq-gram distance between ordered labeled trees. ACM Trans. Database Syst. V, 1–35 (2010).
34. Manku, G.S., Jain, A., Das Sarma, A.: Detecting near-duplicates for web crawling. Proc. 16th Int. Conf. World Wide Web. 141–150 (2007).
35. Najork, M., Wiener, J.: Breadth-first crawling yields high-quality pages. Proc. 10th Int. Conf. World Wide Web. 114–118 (2001).

Tomas Grigalis received the B.Sc. and M.Sc. from Vilnius University in 2004 and 2008 respectively. Currently he is a PhD student in Vilnius Gediminas Technical University, Lithuania. His research interests include automatic structured Web data extraction and integration, Deep Web.

Professor **Antanas Čenys** has published more than 70 articles in refereed scientific journals, more than 60 publications in international conference proceedings. His articles are cited more than 500 times. In 1999 has won Lithuanian National Award of Science. He also is Lithuanian representative at NATO RTO Information Systems Panel. Research interests include information systems, information and systems security, data extraction.

Received: April 16, 2013; Accepted: October 25, 2013