A Lightweight Approach for Extracting Product Records from the Web

Andrea Horch, Holger Kett and Anette Weisbecker
Fraunhofer Institute for Industrial Engineering IAO, Nobelstraße 12, 70569, Stuttgart, Germany

Keywords: Web Data Extraction, Product Record Extraction, Tag Path Clustering.

Abstract: Gathering product records from the Web is very important to both shoppers and on-line retailers for the purpose of comparing products and prices. For consumers, the reason for doing this is to find the best price for a product, whereas on-line retailers want to compare their offers with those of their competitors in order to remain competitive. Due to the huge number and vast array of product offers in the Web an automated approach for collecting product data is needed. In this paper we propose a lightweight approach to automatically identify and extract product records from arbitrary e-shop websites. For this purpose we have adopted and extended the existing technique called Tag Path Clustering for clustering similar HTML tag paths and developed a novel filtering mechanism especially for extracting product records from websites.

1 INTRODUCTION

Obtaining product records from the Web is an essential step when extracting product and especially price information from e-shop websites which is important for consumers as well as for on-line retailers. Whereas the consumers compare prices to find the cheapest price for a special product on the Web, retailers compare their own prices with the prices of their competitors in order to remain competitive.

According to (Simon and Fassnacht, 2008) price reductions compared to classical advertising activities are more efficient and can be realised significantly easier and faster. (McGovern and Levesanos, 2014) show that one of the success factors for on-line retailers is knowing the prices of the competitors and being able to adjust their own prices.

E-commerce is a constantly growing market. In 2014 the turnover of Europe's e-commerce increased by 16.3% to 363.1 billion Euros. The number of online retailers in Europe is estimated to be 640,000 (Nagelvoort et al., 2014). The rising number of online retailers leads to an increasing number of on-line product offers. Handling the comparison of such an amount of products and price data is hardly manageable on a manual basis. Hence, online retailers as well as consumers need software support for automatically comparing products and price data. Such software tools need to automatically identify, extract and structure the product and price information on the e-shop websites for comparing prices and displaying the analysed data to the users. Automatically identifying and extracting product and price information from arbitrary e-shop websites is a very challenging task as different e-shops are selling a variety of products of different domains and there are various types of e-shop software using differently structured templates for displaying the product information. The first and one of the most important steps when gathering product and price information from the Web is the identification and extraction of the single product records on the e-shop website.

This paper proposes a lightweight approach, called LightExtraction, for identifying and extracting product records from arbitrary e-shop websites. The approach was built up by following the idea of Tag Path Clustering like presented in (Grigalis, 2013) and (Grigalis and Cenys, 2014) and simplifying the filtering steps by exploiting the need for common attributes for displaying product descriptions on e-shop websites like product name, product image and product description.

The paper is structured as follows: In Section 2 we present the related work. Section 3 show the results of the analysis of a set of product records of 30 different e-shop websites. Section 4 introduces our novel approach. We demonstrate the adequacy of our approach through an experiment and highlight its results in Section 5, and we conclude in Section 6.

Horch A., Kett H. and Weisbecker A..
A Lightweight Approach for Extracting Product Records from the Web.
DOI: 10.5220/0005441404200430
In *Proceedings of the 11th International Conference on Web Information Systems and Technologies* (WEBIST-2015), pages 420-430
ISBN: 978-989-758-106-9

2 RELATED WORK

There are several existing approaches in science and practice for extracting data records from the Web.

Tools like Dapper[1], Kimono[2] or import.io[3] can be used to extract data directly from a website. Before such a tool is able to extract the relevant data it needs to be configured. The configuration is made manually by a graphical user interface, e.g. an integrated browser, where the navigation steps to reach the page, which includes the data of interest, have to be simulated in the graphical interface and the data for the extraction has to be marked.

Another interesting tool for web data extraction is Crawlbot[4]. Crawlbot offers a web service as well as an API for crawling product price data, historical weather data or news articles from the Web. For the automated extraction Crawlbot needs the URL (Uniform Resource Locator) of the product or article to be scraped. Crawlbot analyses the website of the given URL, structures it into its attributes (e.g. product name, product price) and returns the attributes of the product or article in a well structured format. The problem with using Crawlbot to get the structured product descriptions of a whole e-shop is that the URL of each product detail page has to be defined as input since Crawlbot cannot handle pages including more than one product record. Thus, for obtaining the URLs of the product detail pages another automated mechanism is needed.

Over the years many scientific approaches to automatically identify and extract data records from the Web have been developed. Some of the approaches are based on machine learning, others are based on phrase analysis or Tag Path Clustering. There are also hybrid methods, which rely on several techniques.

The most popular scientific approaches for automatically detecting and extracting data records from websites are the MDR (Mining Data Records in Web Pages) algorithm described in (Liu et al., 2003) and the ViNTs (Visual information aNd Tag structure based wrapper generator) tool introduced in (Zhao et al., 2005).

The MDR algorithm compares the child nodes of each node in an HTML tree starting at the root node for discovering data regions inside a web page. The node comparison is done either by calculating the string edit distance (e.g. Levenshtein distance) or by a tree matching algorithm (e.g. Simple Tree Matching). The similarity of nodes is defined by a preset threshold. Through this procedure the algorithm searches for similar child node combinations in each node. A node containing several similar child nodes is considered as a data region including a set of data records. MDR traverses only the trees of nodes which are not covered by already identified data regions and which include at least three child nodes. The similar-structured child nodes of a data region are the data records.

The MDR algorithm is very useful when searching for the data regions inside a website, but another approach is needed for identifying the data region of a website containing the relevant data records for extraction.

ViNTS is a tool for automatically generating a wrapper for extracting search result records of an arbitrary search engine. For building a wrapper for a search engine ViNTS needs some sample result pages and an empty result page of the search engine as input. ViNTS renders the sample result pages and removes all content lines, which also appear in the empty result page, in order to remove all irrelevant content. On the sample result pages ViNTS identifies some candidate search result records as sample input for the wrapper generation step. The candidate search result records are detected by three steps. In the first step the Candidate Content Line Separators, which are HTML tags like $<p>$ or $<tr>$ separating single search result records, have to be determined. For this purpose ViNTS translates the content lines of the HTML tree into a pair of type code and position code. The type code specifies the content type of the content line like, for example, *text*, *link-text* or *link*, whereas the position code represents the left x coordinate of the rendering box of the content line. All pairs (type code, position code), which can be found at least three times inside the HTML tree, are considered as probable Candidate Content Line Separators. In (Zhao et al., 2005) this step is done by a suffix tree. In the next step the search result page is segmented into multiple content line blocks by using the Candidate Content Line Separators. Consecutive blocks are grouped by their visual similarity with regard to a preset threshold. The visual similarity is calculated by the type distance, shape distance and position distance. The type distance of blocks specifies their edit distance (e.g. Levenshtein distance) of their type codes. The shape distance measures the difference between the indention sequences of the shapes of the blocks. The position distance of two blocks defines the difference between their closest points to the left boundary of the search result page. In the last step of the candidate record detection the first line of every record becomes identified by a set of four predefined heuristic rules:

[1] http://open.dapper.net/
[2] https://www.kimonolabs.com/
[3] https://import.io/
[4] http://www.diffbot.com/products/crawlbot/

(1) the line following an <*hr*>-tag, (2) the only line in the block starting with a number, (3) the only line in the block having the smallest position code or (4) the only line in the block following a blank line. After having identified the sample candidate result records ViNTS builds the wrapper. For this purpose ViNTS determines the tag paths beginning at the root node of the result page (<*html*>-tag) for each identified first line element. The minimal sub-tree of the result page, including all search result records, is calculated based on the tag paths. The search result records are sub-trees of the result page, which are siblings and have the same or a similar tag structure. These sub-trees can be separated by a separator fulfilling the following conditions: (1) common subset of the sub-trees of all records, (2) occurs only once in a sub-tree of each record and (3) contains the rightmost sub-tree of each result record. There can be several separators for a dataset. The wrapper is built by using the smallest tag path for detecting the data region including the search result records and the separators to separate the result records within the data region.

ViNTS needs sample result pages and an empty result page as input, which can be difficult when extracting product records from e-shop websites since there is usually no empty result page which can be used.

(Walther et al., 2010) present an approach for extracting structured product specifications from producer websites. For the retrieval of the product specification the algorithm locates the product detail page on the producer's website and extracts and structures the product attributes of the product specification. For searching the producer's page with the product specification (Walther et al., 2010) process keyword-based Web search by using the popular search engines Google, Bing and Yahoo. After the Web search step (Walther et al., 2010) rank the results by using a method called "Borda ranking" described in (Liu, 2006) followed by the analysis of the page URI, the page title and the page content based on domain specific terms for finding the producer site within all candidates which were found by the Web searches. For extracting the product data in form of key-value pairs (Walther et al., 2010) execute three different wrapper induction algorithms on the product detail page. Each of the three algorithms cluster the HTML nodes, which contain text to a node cluster as a first step. The first algorithm is chosen if there are example key phrases provided as input. The algorithm filters the clusters created in the first step of the nodes, which contain the example phrases. The XPath description of the nodes is used for wrapper generation. If no key phrases are provided as input the second algorithm is used, which exploits domain knowledge from already stored product data as key phrases to find the relevant nodes in the cluster for generating the wrapper. If there are neither example key words nor domain knowledge provided as input the third algorithm generates the wrapper from training sets, which are product pages of related products. In the last step the key-value pairs are extracted by text node splitting based on identifying separators like a colon in the text nodes.

The problem when using the approach of (Walther et al., 2010) for automated product record extraction is that example product data for arbitrary product domains is required, which has to be given by the users in the form of key phrases, or which must be provided from the system as knowledge. For obtaining good results, the key phrases provided by the users or system must fit the phrases of the product detail page of the producers, otherwise the approach will not work. Additionally, numerous steps and different algorithms are needed for the data extraction task.

The approach proposed in (Anderson and Hong, 2013) for extracting product records from the Web is based on Visual Block Model (VBM) a product of the HTML tag tree and the Cascading Style Sheet (CSS) of a web page. The VBM is created by the rendering process of a layout engine like WebKit[5]. (Anderson and Hong, 2013) filter the basic blocks of the page, which are blocks containing other visual blocks. In the next step the similarity of the basic blocks is defined by calculating the visual similarity, the width similarity and the block content similarity. Blocks are visually similar since all of their visual properties are the same. Width similarity of blocks is given if their width properties are within a 5 pixels threshold of each other. The block content similarity exists if the blocks include similar child blocks, which is calculated by using Jacard index described in (Real and Vargas, 1996) and a preset similarity threshold. For the product record extraction (Anderson and Hong, 2013) select a seed candidate block, which is a single basic block. The seed block is identified by selecting a visual block in the centre of the page and tracing the visual blocks around that block by moving clockwise in the form of a Ulam Spiral[6] until reaching a basic block which is taken as seed block. The seed block is within one or more container blocks where one is assumed to be a data record block. Thus, all of them are taken as candidate blocks. Clusters for all of the candidate blocks are created based on the calculation of block content similarity to all blocks in the VBM. The cluster including the maximum number of

[5]http://www.webkit.org/

[6]http://mathworld.wolfram.com/PrimeSpiral.html

container blocks is taken as the cluster containing the product records.

The approach of (Anderson and Hong, 2013) depends on the selection of a correct seed block. For the selection of the seed block the algorithm starts in the page centre and moves clockwise in the form of a Ulam Spiral to identify a basic block including product information. The clockwise direction was chosen so as to not reach the edge of the page or include noisy features like a left menu. The algorithm follows the assumption that the page menu appears on the left side of the page. But this assumption is not correct for pages of the Arab world like bestarabic.com[7] where the page menu is located on the right side of the page. Thus, the approach can fail for pages with a non-standard page structure.

Another approach called ClustVX described in (Grigalis, 2013) and (Grigalis and Cenys, 2014) is based on clustering XPaths and CSS elements of the HTML elements in the DOM tree of a web page. The proposed extraction process takes the web page rendered by a web browser and starts pre-processing. The pre-processing comprises the embedding of visual features into the element attributes, the transforming of the HTML code into valid XHTML and the removing of text formatting elements (e.g. $<i>$ or $<b>$). After the pre-processing an XPath string enriched with visual information (e.g. font, font-size, font-colour), called Xstring, is generated for each element of the page tree. The elements are clustered by Xstring similarity, that means elements having the same Xstring belong to the same cluster. For identifying the data region of the elements in the cluster the longest common XPath prefix of all elements in the cluster is calculated. In order to segment data records the approach identifies if each data record of a data region has its own parent node or if all data records of the region are under the same parent node, which is done by comparing the XPath strings of the elements beginning after the longest common XPath prefix. If each element has its own parent node the data records are the children of the longest common tag path node. In the case that all data records have the same parent node, the approach uses a technique called "HTML tree hopping". The HTML tree hopping technique searches the first data item in the first data record of the data region, then searches the first item of the second data record. The separator of the data records can be found in the HTML tree above the first item of the second data record and can be used to separate all data records of the data region. For determining the importance of the data regions in order to detect the data region including the relevant page data (Grigalis and Cenys, 2014) calculate the visual weight of each data region. The visual weight of a data region is the product of the average area of one data record and the square of the number of data items. The data records of the most important data region are extracted by collecting the elements of the identified tag paths from the HTML tree.

ClustVX is suitable for identifying and extracting data records from arbitrary web pages, but it includes many different process steps which are not necessary. Using the novel approach proposed in this paper the data records of a web page can be identified and extracted by a few steps. As we prove in Section 5 our novel approach achieves as good results as the approaches of (Liu et al., 2003) and (Grigalis and Cenys, 2014).

[7] http://www.bestarabic.com/mall/ar/

3 PRODUCT RECORD ANALYSIS

In order to develop an algorithm for identifying and extracting product records from arbitrary e-shop websites we have analysed the element structure of product records of 30 different e-shop websites.

The selected e-shop websites comprise a wide variety of product categories, various page structures as well as different languages, diverse character sets and different currencies as there were e-shop websites selected from the United States of America, the United Kingdom, Spain, Greece and Germany.

For each of the e-shop websites the product records of a randomly selected product overview page was analysed. The selected websites, the criteria and the collected data for the analysis of the product records as well as the result data for the assay are shown in Table 1.

In the rows of the table the following data for the selected e-shop websites can be found:

- Column 1: The URL of the e-shop websites.
- Column 2: The number of analysed product records which corresponds to the number of product records available on the selected product overview page of the e-shop website.
- Column 3: The name of the tag which represents a product record on the page.
- Column 4-6: The most frequent number of parent elements of the HTML elements which represent the product records. That means, most of the product record elements have this number of parents. The maximum number of parent elements shows the maximum number of parents one or more of the product record elements have. The minimum number of parent elements correspond

Table 1: Product Record Analysis.

URL	number of analysed product records	tag name of product record element	most frequent number of parents	min. number of parents	max. number of parents	most frequent number of children	min. number of children	max. number of children	most frequent number of img elements	min. number of img elements	max. number of img elements	most frequent number of anchor elements	min. number of anchor elements	max. number of anchor elements	average length of included text
http://de.vila.com/	10	div	11	11	11	41	25	76	6	2	20	5	3	12	609
http://raanthai.co.uk/	72	div	19	19	19	28	27	29	1	1	1	4	4	4	129
http://www.bestbuy.com/	14	div	7	7	7	158	126	158	1	1	1	15	14	15	1,056
http://merseyfuels.co.uk/	10	tr	11	11	11	13	13	13	1	1	1	2	2	2	170
http://www.zazzle.de/	60	div	11	11	11	8	8	8	1	1	1	2	2	2	45
http://www.e-shop.gr/	9	table	19	19	19	25	20	25	3	3	3	3	2	3	305
https://www.hairshop-pro.de/	24	div	10	10	10	21	21	21	1	1	1	3	3	3	102
http://www.my-hairshop.de/	10	li	10	10	10	20	20	20	1	1	1	4	4	4	167
http://www.basic-hairshop.de/	15	li	10	10	10	11	11	11	1	1	1	3	3	3	62
http://www.powells.com/	25	li	9	9	9	15	15	23	1	1	6	3	3	3	61
http://heyshop.es/	42	div	8	8	8	6	6	8	1	1	1	1	1	1	33
http://us.nextdirect.com/	24	div	9	9	9	8	8	8	1	1	1	2	2	2	35
http://www.media-dealer.de/	19	form	9	9	9	41	41	45	2	2	3	4	4	4	471
http://www.thinkgeek.com/	11	div	8	8	8	6	6	9	1	1	1	1	1	1	53
http://www.flaconi.de/	17	div	7	7	7	15	15	17	1	1	1	3	3	3	85
http://www.mrwonderfulshop.es/	16	li	10	10	10	13	13	13	2	2	2	2	2	2	71
http://www.dutyfreeshops.gr/	15	div	7	7	7	18	18	20	1	1	2	4	4	4	77
http://www.sammydress.com/	60	li	8	8	8	14	13	17	1	1	1	3	3	3	98
http://www.fragrancenet.com/	17	section	9	9	9	20	20	20	1	1	1	3	3	3	86
http://www.perfume.com/	20	div	10	10	10	10	10	10	1	1	1	2	2	2	47
http://www.sunglasshut.com/	13	div	14	14	14	47	39	47	2	2	2	8	7	8	1,740
http://surrealsunglasses.es/	18	li	10	10	10	39	39	42	2	2	2	6	6	6	65
http://www.smartbuyglasses.gr/	44	ul	11	11	11	17	15	20	1	1	1	4	3	4	58
http://zyloeyewear.com/	33	div	9	9	9	10	10	10	1	1	1	2	2	2	45
http://www.tokotoukan.com/	17	div	8	8	8	9	9	18	1	1	1	3	3	6	82
http://www.adidas.de/	44	div	13	13	13	28	28	56	1	1	7	5	3	7	97
http://batterypark.gr/	18	div	12	12	12	58	56	58	1	1	1	3	3	3	974
http://www.you.gr/	20	div	11	11	11	32	31	34	1	1	1	6	6	6	258
http://www.fk-shop.es/	96	div	7	7	7	38	36	39	3	2	4	7	6	7	108
http://la-shop.es/	9	div	8	8	8	19	19	19	1	1	1	4	4	4	299

to the minimum number of parent elements one or more product record elements have.

- Column 7-9: The most frequent number of children elements of the HTML elements which represent the product records. That means, most of the product record elements have this number of children. The maximum number of children elements shows the maximum number of children one or more of the product record elements have. The minimum number of parent elements correspond to the minimum number of parent elements one or more product record elements have.
- Column 10-12: The most frequent number of image elements (<*img*> tag) included in the HTML elements which represent the product records. That means, most of the product record elements include this number of images. The maximum number of image elements shows the maximum number of images one or more of the product record elements contain. The minimum number of images elements correspond to the minimum number of image elements one or more product record elements include.
- Column 13-15: The most frequent number of an-

chor elements (<*a*> tag) included in the HTML elements which represent the product records. That means, most of the product record elements include this number of anchors. The maximum number of anchor elements shows the maximum number of anchors one or more of the product record elements contain. The minimum number of anchors elements correspond to the minimum number of anchor elements one or more product record elements include.

- Column 16: The average length of included text shows the average length of text included in a product record built over all product record elements on the product overview page.

The analysis of the collected product record data led to the following results:

- There are seven different tag types including the product records in the selected page set for the analysis. With 63.3% the most product records are represented by a <*div*> tag and still 20% are included in a <*li*> tag. Additionally, the product records were represented once by a <*tr*> tag, a <*table*> tag, a <*form*> tag, a <*section*> tag and a <*ul*> tag. Since the tag type of the product records vary in almost 40% of the selected pages there cannot be made an assumption about the type of tag including the product records.
- Considering the parent elements of the product record elements of the selected pages the number and path of parent was exactly the same for all product records inside a page for 100% of the analysed pages. That leads to the conclusion that all product records of one page are located in the same data region. Thus, if the parent path of one or more product records could be identified the remaining product records can be obtained based on the path built from parent elements.
- 36.7% of the product records of the selected pages include the same children whereas even 63.3% contain a different number of child elements. The number of included child elements in the considered page set comprises a range from 6 to 158 children. For this reason the product records cannot be identified based on analysing their child element structure.
- 66.7% of the product records of the considered pages include exactly one image element. 80% of the product records include the same number of image elements as the other product record elements of the page, whereas this number differs for 20%. The range of the number of image elements varies from 1 to 20 elements. On this account one can only assume that a product record will usually include at least one image element, but no assumption can be made about the number of image elements.
- 73,3% of the product records include the same number of anchor elements as the other product record elements of the page, whereas it differs for 26,7%. The range of the number of anchor elements varies from 1 to 15 elements. Therefore it can be only assumed that a product record will usually include at least one anchor element, but it cannot be made an assumption about the number of anchor elements.
- Each product record element of the selected pages contains text. The average length of the text included in one product record element comprises a range from 33 to 1,740 characters. Thus, no assumption of the text length inside a product record element can be made, but it can be expected that a product record contains some text.

4 APPROACH

The proposed approach, called LightExtraction, is a lightweight method for automatically detecting and extracting product records from e-shop websites.

The existing approaches presented in Section 2 need many steps to identify and extract relevant data records from web pages. The MDR algorithm is a slim approach, but for the identification of the relevant datasets an additional method is needed. In contrast, LightExtraction automatically detects and extracts the product records of an e-shop Web page through only a few steps.

The functionality of the developed algorithm is based on the results of the analysis of the product records presented in Section 3. The LightExtraction algorithm is shown in form of pseudo code in Figure 1. LightExtraction uses a clustering technique based on a special tag path representation of the elements in the HTML page tree of a web page for identifying and extracting product records.

The input for the algorithm is the URI of a web page, which is retrieved and rendered in the first step e.g. by using Selenium WebDriver[8]. All information like CSS or information created by JavaScript code is made available in the HTML page tree. Since the majority of e-shop websites are automatically generated by modern e-shop software which uses templates for viewing the content of a database the product records

[8]http://docs.seleniumhq.org/projects/webdriver/

```
render web page

for each element in HTML page tree:
  remove style & script elements
  if product record filter matches:
    generate tag path
    add element to tag path cluster

get tag path of cluster with max. elements

results = elements with identified tag path
results += elements with same parent path

return results
```

Figure 1: LightExtraction Algorithm.

are usually located in the same data region of the e-shop website and contain similar or even the same elements. For this reason the page elements are filtered by analysing their basic structure as described in Section 4.1 and then the elements are clustered based on a special tag path representation as depicted in Section 4.2. The elements are extracted by the created tag paths as described in Section 4.3. The output of the algorithm are the HTML elements of the web page including the product records.

4.1 Element Filtering

After the rendering process LightExtraction runs through all elements inside the HTML page tree. The algorithm rejects all elements having only a styling purpose like <*b*>, <*strong*> or <*em*> or elements including JavaScript code like <*script*>. In the next step LightExtraction checks if the element probably is a product record by using a special filter. The filter compares the structure of the element to a basic element structure, which is expected for a product record. The filter of LightExtraction assumes that a product record (1) contains at least five child nodes and additionally, that it (2) includes some text (product name and description) and (3) an image tag (<*img*>, product image) as well as (4) an anker tag (<*a*>, hyperlink to product detail page). In this way LightExtraction prevents the detection of single record items or items of large navigation menus as product records. The filter was implemented as a simple if-statement which checks the elements for having the mentioned structure.

4.2 Element Clustering

For the element clustering a special tag path is built for each element. The first part of the tag path describes the path from root element (<*html*>) of the web page to the actual element (including the actual element). An asterisk in square brackets is added after the actual element for marking it. The second part of the tag path consists of the tag paths of all child elements from the child element to its last element. That means the tag paths of all child elements are connected together to one long tag path. In order to be able to distinguish different elements which would have the same paths (e.g. the *tr* and *td* elements of the same table) as well as to store the information which part is the parent path an asterisk in square brackets is used to highlight the actual element.

Figure 2a shows the HTML snippet of an example product record. The Web browser view of this product is presented in Figure 2b and its tag paths created by LightExtraction is shown in Figure 2c. The first line of Figure 2c represents the tag path of the searched list element including the product record data. The tag path until the asterisk in square brackets represents the tag path from the root element to the actual element, the path after this marking shows the element paths of the actual element's children which were compound to one long path string. The tag path built by LightExtraction does not properly represent the HTML tree of the element since the tag paths of the child elements are put together to a single long tag path string not respecting the structure of the children in the HTML tree. But the structure of the element's children is not important for the clustering of the HTML elements since the goal is to cluster elements with the same tag name, having the same path from root and containing the same child elements.

The elements are clustered based on the created tag path. That means each element cluster comprises elements having exactly the same tag path. Thus, an element cluster contains only the same element types (e.g. <*div*>), which are probably elements of the same data region of the web page and which include the same child elements.

4.3 Product Record Extracting

LightExtraction assumes that the cluster containing the maximum number of elements includes the majority of the product records. Thus, LightExtration takes the tag path of that cluster for identifying all clusters containing data records by searching for all elements in all clusters having the same element tag path. The elements with the same element tag path are the elements of the same data region as the elements of the cluster including the maximum number of elements. LightExtraction considers the result set of that last step as the product records of the web page and extracts them.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
 ...
    <body>
        <div>
            <ul class="product-list" id="product-list">
                ...
                <li class="item">
                    <a href="http://www.shop.com/shampoo250.html">
                        <img src="img1.jpg" alt="Shampoo 250ml">
                    </a>
                    <div class="col">
                        <h2>
                            <a href="http://www.shop.com/shampoo250.html">Shampoo 250ml</a>
                        </h2>
                        <em class="discount_price">
                            <span class="price">&#36;10.00</span>
                        </em>
                        <a href="http://www.shop.com/shampoo250.html">Details</a>
                    </div>
                </li>
                ...
            </ul>
        </div>
    </body>
</html>
```

(a) HTML snippet.

Shampoo 250ml

$10.00 Details

(b) Web browser view.

```
...
/html/body/div/ul/li[*]/a/img/div/h2/a/span/a
/html/body/div/ul/li/a[*]/img
/html/body/div/ul/li/a/img[*]
/html/body/div/ul/li/div[*]/h2/a/span/a
/html/body/div/ul/li/div/h2[*]/a
/html/body/div/ul/li/div/h2/a[*]
/html/body/div/ul/li/div/span[*]
/html/body/div/ul/li/div/a[*]
 ...
```

(c) Tag path snippet.

Figure 2: HTML snippet, Web browser view and tag path snippet of a product record.

5 EXPERIMENT

For evaluation the proposed approach is compared to the two existing approaches MDR and ClustVX by an experiment. We have chosen MDR since it is one of the most popular approaches for the automated detection and extraction of data records from web pages. ClustVX was selected as a recent approach promising very good results. The experiment comprises the identification and extraction of product records from the same data set of web pages by each approach and the comparison of the extracted results by evaluation metrics.

5.1 Experimental Setup

For the experiment we implemented the LightExtraction approach in Python[9]. The rendering of the HTML page tree is done by Firefox Selenium Driver

[9]https://www.python.org/

for Python[10]. For the navigation in the HTML page tree we use Beautiful Soup[11].

For MDR we use the MDR implementation available on the MDR Website[12] of the Department of Computer Science of the University of Illinois at Chicago (UIC).

Since the ClustVX Demonstration Website[13] is not available we have made our own implementation in Python. For the rendering of the HTML page tree and the navigation in the HTML tree we use Firefox Selenium Driver for Python and Beautiful Soup.

According to (PostNord, 2014) in 2014 the five product categories most often bought online in Europe were clothes, books, home electronics, cosmetics and CDs. We have created an experimental dataset containing randomly selected web pages. The dataset has to be a mixture of web pages of at least three different countries and each web page must include one of the most popular product categories. The result dataset is shown in Table 2.

Since MDR can identify and extract data regions as well as the included data records but it is not able to decide which is the data region containing the relevant data records, we manually identify the relevant data region (if extracted) and count the correctly and incorrectly extracted data records for the evaluation.

5.2 Evaluation Metrics

For the evaluation of the results and the comparison of the different approaches we use the precision and recall measures, which are common metrics in the field of information retrieval. The definition of precision and recall in the context of information retrieval is given in Equation 1 and Equation 2 (Rijsbergen, 1979).

$$Precision = \frac{|Relevant\ Records \cap Retrieved\ Records|}{|Retrieved\ Records|} \quad (1)$$

$$Recall = \frac{|Relevant\ Records \cap Retrieved\ Records|}{|Relevant\ Records|} \quad (2)$$

Since LightExtraction and the other approaches have to classify the elements of the web pages into relevant data records (product record) and other elements (irrelevant data) we can use the terms True Positives, False Positives, True Negatives and False Negatives for calculating precision and recall.

A True Positive (TP) is a correct hit, which is a correctly extracted data record (in our case: a product record). False Positives (FP) are incorrect hits or false alarms, which are incorrectly extracted data records. True Negative (TN) means a correct rejection, which is a correctly rejected data record. In our context True Negatives cannot be measured since the number of all negative data records on website is unknown. Incorrect rejections or missing hits are defined as False Negatives (FN), which are incorrectly rejected data records (product records, which have not been detected).

Expressing the equations using the terms True Positive (TP), False Positive (FP) and False Negative (FN) Equation 1 leads to Equation 3 and Equation 2 to Equation 4.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (4)$$

5.3 Experimental Results

The results of the experiment are shown in Table 3. The number of product records available on each page of the experiment is given in the column “total”. MDR obtains a precision of 39.77% and a recall of 11.99%, ClustVX reaches a precision of 98.99% and a recall of 67.47%, while LightExtraction achieves a precision of 98.43% and a recall of 85.96%. The results show that both LightExtraction and ClustVX achieve much better results than MDR. LightExtraction obtains a similarly good precision as ClustVX and even a better recall.

The reason for the missing product records (False Nagatives) of row 1 and row 10 is that these are located in a second product data region which LightExtractor is not recognizing. The product records of row 5 were not identified since LightExtractor has detected some promotion product records in the website menu contains 47 records which is a higher number of elements than the number of the “real” product records which is 24. The False Positives of row 3 and 10 appear since the elements are located in the same data region as the product records.

In order to avoid such incorrect results the filter of the LightExtraction algorithm has to be improved in

[10] http://selenium-python.readthedocs.org/en/latest/api.html
[11] http://www.crummy.com/software/BeautifulSoup/
[12] http://www.cs.uic.edu/liub/WebDataExtraction/MDR-download.html
[13] http://clustvx.no-ip.org/

Table 2: Experimental Dataset.

No.	URI for Data Extraction	Product Category
1	http://www.barnesandnoble.com/u/new-books-fiction-nonfiction-bestsellers/379004022	Books
2	http://www.ebay.com/chp/Baitcasting-Reels-/108153	Miscellaneous
3	http://www.terrashop.de/	Books
4	http://www.very.co.uk/home-garden/curtains-blinds/made-to-measure-curtains-blinds/e/b/116982.end	Clothes
5	http://www.electricshop.com/televisions/televisions/icat/subtelevisions/iflt/tag-screentype%7C46_4kultrahd_2755	Home Electronics
6	http://www.alconeco.com/makeup/eyes	Cosmetics
7	https://thecomicbookshop.comicretailer.com/comics-sale	Comic Books
8	http://coozina.gr/store/home.php?cat=188	Housewares
9	http://atlasstoked.com/	Clothes
10	http://www.bestarabic.com/mall/ar/	CDs & DVDs

Table 3: Experimental Results. See Table 2 for the URIs of the experimental websites.

No.	Total	MDR algorithm					ClustVX					LightExtraction				
		TP	FP	FN	Precision	Recall	TP	FP	FN	Precision	Recall	TP	FP	FN	Precision	Recall
1	43	0	0	43	0.00%	0.00%	30	0	13	100.00%	69.77%	30	0	13	100.00%	69.77%
2	25	0	0	25	0.00%	0.00%	25	0	0	100.00%	100.00%	25	0	0	100.00%	100.00%
3	12	8	24	4	25.00%	66.67%	12	0	0	100.00%	100.00%	12	3	0	80.00%	100.00%
4	12	0	0	12	0.00%	0.00%	12	0	0	100.00%	100.00%	12	0	0	100.00%	100.00%
5	24	24	0	0	100.00%	100.00%	24	0	0	100.00%	100.00%	0	0	24	0.00%	0.00%
6	45	0	0	45	0.00%	0.00%	45	0	0	100.00%	100.00%	45	0	0	100.00%	100.00%
7	59	0	0	59	0.00%	0.00%	10	0	49	100.00%	16.95%	59	0	0	100.00%	100.00%
8	36	0	0	36	0.00%	0.00%	36	0	0	100.00%	100.00%	36	0	0	100.00%	100.00%
9	12	3	29	9	9.38%	25.00%	3	0	9	100.00%	25.00%	12	0	0	100.00%	100.00%
10	24	0	0	24	0.00%	0.00%	0	2	24	0.00%	0.00%	20	1	4	95.24%	83.33%
Total:	*292*	*35*	*53*	*257*	*39.77%*	*11.99%*	*197*	*2*	*95*	*98.99%*	*67.47%*	*251*	*4*	*41*	*98.43%*	*85.96%*

future work by adding new criteria as well as decision rules for product record identification which would make the filter more flexible as the current one.

6 CONCLUSIONS

This paper presents a lightweight approach called LightExtraction for automatically detecting and extracting product records from web pages of online shops. The proposed approach uses a filtering technique for rejecting irrelevant elements. It clusters the elements of the HTML page tree by their tag paths. For this purpose LightExtraction generates the tag path for each element of the HTML page tree and adds elements with identical tag paths to the same element cluster. The tag path comprises the path from the first parent node (<*html*> tag) to the last child node of the element. The element is marked in the tag path by an asterisk in square brackets. The element cluster including the maximum number of elements is expected to contain the majority of product records. LightExtraction uses the tag path of the maximum cluster for detecting all elements in all clusters, which are product records.

In an experiment the novel approach is compared to the existing approaches MDR and ClustVX. The results of the experiment show that LightExtraction can achieve much better results than MDR with similarly good precision and an even better recall than ClustVX, whereas LightExtractor needs significantly less process steps than ClustVX.

ACKNOWLEDGEMENTS

The work published in this article was partially funded by the SME E-COMPASS project of the European Union's Seventh Framework Programme for research, technological development and demonstration under the grant agreement no. 315637.

REFERENCES

Anderson, N. and Hong, J. (2013). Visually extracting data records from the deep web. In *Proceedings of the 22nd International World Wide Web Conference (WWW 2013)*, WWW '13, pages 1233–1238, New York, NY, USA. ACM.

Grigalis, T. (2013). Towards web-scale structured web data extraction. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 753–758, New York, NY, USA. ACM.

Grigalis, T. and Cenys, A. (2014). Unsupervised structured data extraction from template-generated web pages. pages 169–192.

Liu, B. (2006). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and*

Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Liu, B., Grossman, R., and Zhai, Y. (2003). Mining data records in web pages. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 601–606, New York, NY, USA. ACM.

McGovern, C. and Levesanos, A. (2014). Optimizing pricing and promotions in a digital world: From product-led to customer-centric strategies. Online.

Nagelvoort, B. et al. (2014). European b2c e-commerce report 2014. Website. http://www.adigital.org/sites/default/files/studies/european-b2c-ecommerce-report-2014.pdf.

PostNord (2014). E-commerce in europe 2014. Website. http://www.postnord.com/globalassets/global/english/document/publications/2014/e-commerce-in-europe-2014.pdf.

Real, R. and Vargas, J. M. (1996). The Probabilistic Basis of Jaccard's Index of Similarity. *Systematic Biology*, 45(3):380–385.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Simon, H. and Fassnacht, M. (2008). *Preismanagement: Strategie - Analyse - Entscheidung - Umsetzung*. Gabler Verlag, Wiesbaden.

Walther, M., Hähne, L., Schuster, D., and Schill, A. (2010). Locating and extracting product specifications from producer websites. In *ICEIS 2010 - Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 4, SAIC, Funchal, Madeira, Portugal, June 8 - 12, 2010*, pages 13–22.

Zhao, H., Meng, W., Wu, Z., Raghavan, V., and Yu, C. (2005). Fully automatic wrapper generation for search engines. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 66–75, New York, NY, USA. ACM.