S. Krajči (ed.): ITAT 2018 Proceedings, pp. 152–156
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Peter Gurský, Matej Perejda, and Dávid Varga

Semi-automatic annotation of e-shops

Peter Gurský, Matej Perejda ,Dávid Varga
Institute of Computer Science
Faculty of Science, P.J.Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia
peter.gursky@upjs.sk, {matej.perejda, david.varga}@student.upjs.sk

Abstract. *Extraction from web pages becomes a very popular way to acquire important data for better decision process. Acquisition of structured data from a new web portal requires an annotation of web pages of the portal to allocate the location and the type of the information. We present methods for semi-automatic annotation of e-shops' content, to create rules for extraction. The methods are implemented in Chrome extension named Exago. The aim of these methods is to generate XPaths and regular expressions. We use positive and negative examples to further specify which of the generated XPaths should be used for extraction. The annotation methods are tested on real data and the results show a high success rate.*

1 Introduction

Web scraping is a complex process that consists of web pages annotation, crawling, data extraction and data processing. This kind of data acquisition is very valuable for decision process, because it can provide data from various sources and process them together. Project Kapsa [1] deals with extraction and unification of information from web pages, focusing on products on e-shops. The aim of the project is a creation and management of a collection of products which are offered by e-shops. In this paper, we focus on the first step of the web scraping process, the web pages annotation.

The annotation has two main goals: recognition of relevant page on portal and identification of positions of relevant pages, where the data of our interest are.

The positions of relevant data are usually specified by XPaths of HTML source or by regular expressions, which are used by extractor on each relevant page. It is also possible to extract data using a procedural script. Writing complex XPaths or regular expressions, as well as a creation of scripts is not an easy task. It requires the annotator to be an IT expert.

Our goal is to make the annotation process of e-shops easier and possible for ordinary person. We would like to offer in our Chrome extension Exago[1] satisfactory results of web scraping with annotation made only by mouse clicking.

2 State of the art

Web annotation and extraction systems can be categorized to four groups [2]. Manual systems [3, 4] require programming in some (pseudo) language. Automatically constructed extractors [5, 6] create extraction system based on complete user annotation and examples of extracted data from several pages. Automatically constructed extractors with partial user support [7, 8] create extraction system without the need of extraction examples. Automatic extractors with no user support [9, 10] analyze repeating patterns on web pages, and extract every data that seem to be interesting. Our tool Exago can be included in automatically constructed extractors with partial user support.

Currently there are more than 50 web scrapers available on the internet. We have tried to use all of them to product data extraction from two e-shops Alza.sk and Heureka.sk. The majority of them were not able to extract the product data. The web scrapers that are at least partially applicable to product data extraction are [11-28], so we examined these web scrapers more deeply. Majority of these tools provide the generation of one XPath by clicking on element on the page. Some tools hide this functionality and do not show the final XPath [12, 18, 19, 21, 22, 24, 25, 26], the others [11, 13, 15, 17, 20, 23, 27, 28] allow the modification of the XPath manually. The rest of them [14, 16] provides manual insertion of XPath only. None of the tools provide regular expression generation, but some of them allow writing regular expressions manually.

3 Methods for annotation

Having an HTML element containing the relevant data we can easily create an XPath as a path from the root element. The XPath points to the target element and it is used during the extraction process on this page. However on the similar page (e.g. the page about different product on e-shop), the created XPath can be unsuccessful – either it finds no element or an element with different kind of data, because the HTML source tree can be slightly different (e.g. missing subtree, different highlighting of texts etc.) than the one of the annotated product.

Fortunately, many XPaths can be created that point to the same element. The tree navigation of XPaths can be based on elements' attributes, order of element between its siblings, various conditions and more. Choosing the right navigation of XPath increases the success rate of extraction considerably. The problem is that an unexperienced user is not able to choose the right XPath from the hundreds of possibilities. Therefore we provide semi-automatic XPath and regular expression generation and interactive selection of the right rules to make the annotation process easier.

3.1 XPath generation

XPath is a query language that is used to select nodes of XML DOM model. All browsers create a DOM model out of HTML file as the first step of page processing. XPath language provides various approaches to specify a starting node(s) and traversal of the DOM model. Creator of XPath expression can utilize tag names, tag attributes and their values, order of the elements, navigation functions, and conditions with build-in functions. Such variability allows many XPaths to localize the same node.

Annotator's goal is to specify the position of relevant data that can be universal for all pages of the same type (created from the same HTML template). In our case, we

focus primarily on pages, where the details of the products are located. Unfortunately, when the template is combined with the structured data of products, to create final detail pages, the differences between result pages eventuate in variety of HTML tree structures. They can vary in element attributes as well as in absence of whole subtrees. Such differences cause many XPaths, which work on one page, fail on other page. They can point to different or no nodes, while the corresponding data is still present somewhere on the page.

It is impossible to know, which parts of the template are on all result pages, without complex analysis of pages, because the templates are not public. Therefore we don't know which elements or attributes can be used as navigation points of universal XPath. Annotation experts usually examine various HTMLs of result pages and create the universal XPath manually.

```
<h1>Samsung 24FDX</h1>
<h2>Specification</h2>
<table id="product parameters">
  <tbody>
    <tr>
      <td style="color:powderblue;">Producer</td>
      <td style="color:pink;">Samsung</td>
    </tr>
    <tr>
      <td style="color:powderblue;">Model</td>
      <td style="color:pink;">24FDX</td>
    </tr>
  </tbody>
</table>
```

Fig. 1. Part of HTML source example

Our approach generates several possible XPaths as possible candidates to final universal XPath. Consider that annotator wants to create an XPath leading to element with value "24FDX" in HTML source on Figure 1. In this example, the result of the method used for generating XPaths is a set of 36 different correct XPaths, which point to the same element. The shortest ones of them are listed below:

- //tr[2]/td[2]
- //tr[last()]/td[last()]
- //tr[last()]/td[2]
- //tr[2]/td[last()]
- //tr[2]/*[@style="color:pink;"]
- //tr[last()]/*[@style="color:pink;"]

XPaths are generated gradually along the path from the clicked element to a specified root element occurs. If there is no root element specified, the root element of HTML document is chosen to be the element where generation stops. At every current element during the generation, all attributes, the name and the order between siblings of this element are combined to create different XPaths.

The extractor process, which extracts the data from all similar pages, needs only one XPath per each value position. Our approach in Exago tool [1] chooses the shortest XPath as the result candidate by default. The annotation process is a process where annotator determines whether the candidate XPath is the most universal one, to be chosen for extraction.

During the annotation process, the annotator can navigate to other similar page (e.g. the page about other product) and check out the success of the chosen XPath. There are three possibilities:

1. The element is correctly found using the XPath and no changes need to be done.
2. The XPath addresses no element and annotator can mark the correct element as **the positive example**.
3. The XPath addresses different element and annotator can mark the addressed element as **the negative example**.

When the annotator marks the positive or negative example, by clicking on an HTML element, the method for generating XPaths generates new set of XPaths to this element. Let this set of XPaths be named as B, and the original set as A. In case of the positive example, the new set of XPaths, that work fine on both pages is A∩B. In case of negative example, the new set is A-B.

After some iterations of this procedure, the result set contains only XPaths that work on all pages. The annotator can choose any of them to be part of final extraction rule, or just keep the default one. As it was mentioned before, our Exago tool chooses the shortest XPath by default.

3.2 Regular expression generation

XPath is a very capable language at locating the whole element of HTML source. There are cases, when we want to extract a value only from a part of the element, or a value spreads across two or more elements. In this case, XPath is unusable.

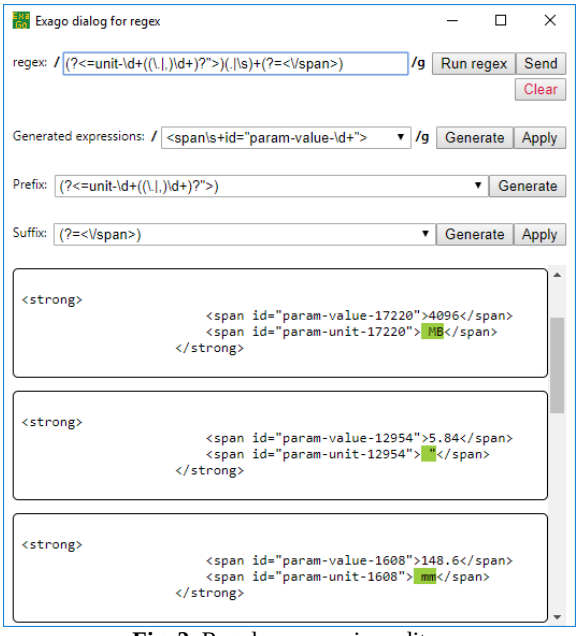

Fig. 2. Regular expression editor

In Exago, we combine XPaths and regular expressions, if needed. XPath localizes the HTML part in which the regular expression can be used. It is also possible, that the XPath localizes more than one element, and regular

Table 1. Annotation accuracy analysis

	first genarated XPath			our approach		
URL of e-shop	Successful annotation of elements	Success rate	Use of regular expressions	Successful annotation of elements	Success rate	Use of regular expressions
gymbeam.sk	10 out of 14	71%	No	11 out of 14	79%	No
vivantis.sk	8 out of 12	67%	No	10 out of 12	83%	Yes
bestbuy.com	5 out of 14	36%	No	12 out of 14	86%	Yes
martinus.sk	8 out of 14	57%	No	12 out of 14	86%	Yes
nay.sk	10 out of 14	71%	No	12 out of 14	86%	Yes
insportline.cz	9 out of 14	64%	No	12 out of 14	89%	Yes
target.com	6 out of 9	67%	No	8 out of 9	89%	Yes
eshop.eta.cz	8 out of 11	73%	No	10 out of 11	91%	Yes
mall.sk	9 out of 11	82%	No	10 out of 11	91%	Yes
notino.sk	4 out of 11	36%	No	10 out of 11	91%	Yes
pantarhei.sk	8 out of 11	73%	No	10 out of 11	91%	Yes
obi.cz	3 out of 14	21%	No	13 out of 14	93%	Yes
andreashop.sk	9 out of 11	82%	No	11 out of 11	100%	No
decathlon.sk	12 out of 13	92%	No	13 out of 13	100%	No
hej.sk	9 out of 11	82%	No	11 out of 11	100%	No
heureka.sk	5 out of 12	42%	No	12 out of 12	100%	Yes
hornbach.sk	5 out of 10	50%	No	10 out of 10	100%	Yes
radioshack.com	9 out of 12	75%	No	12 out of 12	100%	No
rajdazdnikov.sk	0 out of 11	0%	No	11 out of 11	100%	Yes
zoohit.sk	5 out of 11	45%	No	11 out of 11	100%	Yes
Total	**142** out of **240**	**59%**	-	**221** out of **240**	**93%**	-

expression can be used in all of them to find out the target value.

Regular expression is an effective tool for extraction of a substring based on complex conditions with quite complicated syntax. Sometimes, even IT experts have a hard time at constructing more complex regular expressions.

In Exago, we created regular expression editor (Fig. 2) that generates multiple regular expressions using mouse events. . The process of generating regular expressions can be performed only by clicking on buttons in the editor and highlighting the text needed for extraction. Therefore, the editor allows a less experienced user to create regular expressions, and see the result. On the other hand, there is still a possibility to edit generated regular expressions or write custom ones.

Our editor supports two approaches. First, user can select the target text and click on the first Generate button. Exago generates several regular expressions, collected in the combo box. The method that generates the expressions tries to generalize the two most common text parts:

- spaces and other whitespace characters are converted to expression *\s* or *\s+*,
- numbers are converted to *\d+* or *\d+((\.|,)\d+)?*, which covers also decimal numbers.

The second approach to regular expression generation is selection of prefix and suffix of the target value and hitting the appropriate "Generate" button. In Figure 2, user selected text "*unit-17220>*" as the prefix and "**" as the suffix and generated the related regular expressions. Using the chosen regular expressions, a combined expression is created and written in the text field on the top of the screen. The result of the regular expression search is emphasized on the bottom with green background.

4 Experiments

In this section we analyze the accuracy of annotation of our new version of Exago. The new version is using the semi-automatic annotation based on positive and negative examples and generating regular expressions. The tests compare our approach with the usual approach of element localization used in other web scrapers and our previous version of Exago, i.e. generation of one XPath per value.

This analysis has been done on 20 different e-shops. During the testing phase, every annotation has been realized only by clicking with mouse on HTML elements and Exago components. No manual editing of XPaths and regular expressions has been used, in order to compare the old and the new approach. The table on Table 1 shows the results of each annotation per e-shop ordered by success rate.

We measure following aspects for both approaches:

- successful annotation of elements – the number of successfully annotated elements or values compared

to the number of all elements or values that could be annotated,
- success rate – percentage of correctly addressed elements and values among annotated elements and values
- use of regular expressions – information about the use of generated regular expressions during annotation; with every use of regular expressions in Exago, regular expressions have been used in combination with XPaths.

With the new approach, we have achieved success rate of 100 % in 8 internet shops. In remaining 12 e-shops we have not been able to achieve the 100 % success rate because of the following reasons:

- in 5 cases, the HTML structures of detail web pages in each e-shop have varied too much,
- in 5 cases, lists of product parameters have been divided into more elements unrelated to each other,
- in 3 cases, complications occurred during annotation of images that used dynamic styles of their presentation on a web page,
- in 3 cases, we have not been able to annotate prices of products, because e-shops displayed sale prices in different elements compared to non-sale prices, whilst both of these prices have been present on a web page,
- in 3 cases, we have not accomplished to annotate product ratings represented by pictures without any further information given, for example, amount of stars awarded.

Success rate average of annotation of information about products has been 59% in the case of common approach. With the new version of Exago we have achieved the average success rate of 93%. This growth has been achieved with the help of generating regular expressions and the functionality of positive and negative examples.

Some of the unsuccessful cases could be eliminated by manual editing of XPaths and regular expressions. For example, in the cases when internet shops presented more lists of product parameters, we would be able to manually create a regular expression that would address all types of elements representing these lists.

5 Conclusions

This paper deals with improvement of the commonly used data localization process in web annotation by implementing the semi-automatic annotation. We have created a plug-in module Chrome Extension named Exago containing this functionality. With the help of methods for generating XPaths, generating regular expressions and the functionality of positive and negative examples, we are able to annotate more information on detail web pages of products, compared to the previous version of Exago.

During the annotation accuracy analysis, we have been comparing the common approach with the new one, and discovered that the success rate has grown from 59% to 93%. We consider this result as notable, but there is still a room for improvement. One improvement could be creating a new component, which would be able to distinguish different types of detail web pages in one e-shop and use appropriate XPaths and regular expressions for annotation.

Another improvement could be more intelligent generation of XPaths by which we would generate XPaths faster and reduce the generation of very similar XPaths.

Designing methods for annotation of product ratings represented by pictures of stars or other objects would be also very benefiting. The solution to this problem could be counting matches of the regular expression addressing one star in the element containing these stars.

Adding a sale price as a new type of known value component would enable us to annotate sale prices as well as non-sale prices and grow our success rate per e-shop.

The new version of Exago could also be improved by solving a problem with image annotation, where images are being displayed in different dynamic styles. The solution could be some kind of a mechanism that would get all pictures from a detail web page, show them to the user, and the user would pick the image he wants. XPath addressing this image would be generated and used automatically.

References

[1] Project Kapsa, web page: http://kapsa.sk/
[2] B. Liu: Web Data Mining: Exploring Hyperlinks, contents and Using Data, Second edition, Springer 2011. ISBN 978-3-642-19459-7
[3] V. Crescenzi, G. Mecca: Grammars have exceptions. Information Systems, 23(8): 539-565, 1998.
[4] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, A. Sellers: OXPath: A language for sca-lable data extraction, automation, and crawling on the deep web. The VLDB Journal 22(1): 47-72, 2013
[5] C. Hsu, M. Dung: Generating finite-state transducers for semi-structured data extraction from the Web. Information Systems, 1998, 23(8): p. 521-538.
[6] Muslea, S. Minton, C. Knoblock: A hierarchical approach to wrapper induction. In Proce-edings of Intl. Conf. on Autonomous Agents (AGENTS-1999) 1999.
[7] C.-H. Chang, S.-C. Kuo: OLERA: A semi-supervised approach for Web data extraction with visual support. IEEE Intelligent Systems, 19(6):56-64, 2004.
[8] Hogue, D. Karger: Thresher: Automating the Unwrapping of Semantic Content from the World Wide. Proceedings of the 14th International Conference on World Wide Web (WWW), Ja- pan, pp. 86-95, 2005.
[9] V. Crescenzi, G. Mecca, P. Merialdo: RoadRunner: towards automatic data extraction from large Web sites. Proceedings of the 26th International Conference on Very Large Database Systems (VLDB), Rome, Italy, pp. 109-118, 2001.
[10] Arasu, H. Garcia-Molina: Extracting structured data from Web pages. Proceedings of the ACM SIGMOD International Conference on Management of Data, San Diego, California, pp. 337- 348, 2003.
[11] Content Grabber. Web scraper available on-line: (https://contentgrabber.com/)
[12] Data Miner. Web scraper available on-line: (https://data-miner.io/)
[13] Helium Scraper. Web scraper available on-line: (http://www.heliumscraper.com/)
[14] Import.io. Web scraper available on-line: (https://www.import.io/)
[15] Mozenda. Web scraper available on-line: (https://www.mozenda.com/)
[16] ParseHub. Web scraper available on-line: (https://www.parsehub.com/)

[17] Scrapinghub (Portia) . Web scraper available on-line: (https://scrapinghub.com/)
[18] Web Scraper. Web scraper available on-line: (http://webscraper.io/)
[19] Agenty. Web scraper available on-line: (https://www.agenty.com)
[20] Data Toolbar. Web scraper available on-line: (http://datatoolbar.com/)
[21] Dexi.io. Web scraper available on-line: (https://dexi.io/)
[22] Easy Web Extract. Web scraper available on-line: (http://webextract.net/)
[23] Fminer. Web scraper available on-line: (http://www.fminer.com/)
[24] GetData.IO. Web scraper available on-line: (https://getdata.io/)
[25] Grepsr. Web scraper available on-line: (https://www.grepsr.com/chrome-extension/)
[26] Instant Data Scraper. Web scraper available on-line: (https://webrobots.io/instantdata/)
[27] Visual Web Ripper. Web scraper available on-line: (http://visualwebripper.com/)
[28] Web Sundew. Web scraper available on-line: (http://www.websundew.com)