Information Systems 56 (2016) 36–54

Contents lists available at ScienceDirect

Information Systems

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

A tool for producing structured interoperable data from product features on the web

Tuğba Özacar*

Department of Computer Engineering, Celal Bayar University, Muradiye, 45140 Manisa, Turkey

ARTICLE INFO

Article history:
Received 11 March 2015
Accepted 7 September 2015
Recommended by: F. Carino Jr.
Available online 25 September 2015

Keywords:
Information extraction
GoodRelations
Protégé
Web scraping
Ontology
Rich snippets

ABSTRACT

This paper introduces a tool that produces structured interoperable data from product features, i.e., attribute name–value pairs, on the web. The tool extracts the product features using a web site-specific template created by the user. The value of the extracted data is maximized by using GoodRelations, which is the standard vocabulary for modeling product types and their features. The final output of the tool is GoodRelations snippets, which contain product features encoded in RDFa or Microdata. These snippets can be embedded into existing static and dynamic web pages in a way accessible to major search engines like Google and Yahoo, mobile applications, and browser extensions. This increases the visibility of your products and services in the latest generation of search engines, recommender systems, and other novel applications.

1. Introduction

The web contains a huge number of online shops which provide excellent resources for product information. Besides, the data of e-commerce is growing at a rapid speed [1]. Information in e-commerce includes technical specifications and descriptions of products. If we present this information in a structured way, it will significantly improve the effectiveness of many applications [2].

The vast majority of web content consists of different kinds of textual documents, which are provided in a number of different formats and vary from plain text to semi-structured documents containing data records. This makes different methods of bringing structure and semantics to the web (including web information extraction) an active research field [3]. Although the web has a dynamic nature, Etzioni has argued for that “information on the web is sufficiently structured to facilitate effective web mining” [4]. Since a big portion of web content subject to web information extraction is created from data repositories, a web information extraction system rediscovers the structure that was encoded in a web page.

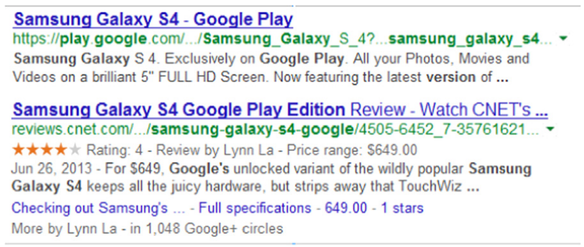
This paper introduces a tool[1] that produces structured interoperable data from product features, i.e., attribute name–value pairs, on the web. It extends the previous work of the author [5] in two ways. First it supports tree nodes that define text operations (e.g. concatenate, contains, fragment, lower, upper, replace, substring, and trim) on tree nodes. Second it presents a user-based evaluation accomplished using 15 different “real world” scenarios.

Designed as a plug-in for the open source ontology editor Protégé [6], the proposed tool exploits the advantages of the ontology as a formal model for the domain knowledge.

* Tel.: +90 236 2012103; fax: +90 236 2412143.
E-mail address: tugba.ozacar@cbu.edu.tr

[1] Download link: https://github.com/tugbaozacar/iris

http://dx.doi.org/10.1016/j.is.2015.09.002
0306-4379/© 2015 Elsevier Ltd. All rights reserved.



Samsung Galaxy S4 - Google Play
https://play.google.com/.../Samsung_Galaxy_S_4?...samsung_galaxy_s4...
Samsung Galaxy S 4. Exclusively on Google Play. All your Photos, Movies and Videos on a brilliant 5" FULL HD Screen. Now featuring the latest version of ...

Samsung Galaxy S4 Google Play Edition Review - Watch CNET's ...
reviews.cnet.com/.../samsung-galaxy-s4-google/4505-6452_7-35761621...
★★★★☆ Rating: 4 - Review by Lynn La - Price range: $649.00
Jun 26, 2013 - For $649, Google's unlocked variant of the wildly popular Samsung Galaxy S4 keeps all the juicy hardware, but strips away that TouchWiz ...
Checking out Samsung's ... - Full specifications - 649.00 - 1 stars
More by Lynn La - in 1,048 Google+ circles

Fig. 1. A regular snippet and a rich snippet.

Another promising feature of the tool is support for building an ontology that is compatible with GoodRelations Vocabulary [7]. GoodRelations is the most powerful vocabulary for publishing all of the details of your products and services in a way friendly to search engines, mobile applications, and browser extensions. In [8], GoodRelations product ontology is defined as a “product atlas” describing specifications, marketing copy, catalog data, photos, videos, manuals, installation instructions, updates, responses to issues, prices and reviews of over 1 million products from various sources. It updates twice a week, so companies can enter all their descriptions and prices and the information will flow through the thousands of e-commerce systems within a few days. The goal is to have extremely deep information on millions of products, providing a resource that can be plugged into any e-commerce system without limitation.

If you have GoodRelations in your markup, Google, Bing, Yahoo, and Yandex will or plan to improve the rendering of your page directly in the search results. Rich snippets—the few lines of text that appear under every search result—are designed to give users a sense for what is on the page and why it is relevant to their query. Fig. 1 shows the difference between a regular and a rich snippet. The first search result is a regular snippet and the second one is a rich snippet. The proposed tool supports marking up your content with RDFa or Microformats for creating rich snippets of the extracted products.

In addition to seeing rich snippets in the search results, rich markup indicates the relevance of your page for a particular query. You provide information to the search engines so that they can rank up your page for queries to which your offer is a particularly relevant match. For many popular shop applications including Drupal Commerce, Magento, Prestashop, and Rakuten.de/Tradoria.de, Joomla/Virtuemart, there exist free extension modules that make adding GoodRelations RDFa for semantic SEO as simple as a few mouse-clicks.

The tool supports “Open Standards” including RDFa, Microdata and JSON. International Telecommunications Union (ITU-T) specifies that “Open Standards” facilitate interoperability and data exchange among different products or services and are intended for widespread adoption. In [9] the key benefits of interoperable data are listed as follows: enabling information sharing with trusted partners, enhancing system capabilities and longevity, lowering overall costs of information applications, improving the breadth and quality of information, increasing the speed and accuracy of decisions, improving transparency and speed of disclosure of information to valid constituents, preserving data for future uses.

The organization of the article is as follows: In Section 2, we review background information and related work. Section 3 includes an overview of the system's architecture, features and settings and a scenario based quick-start guide. Section 4 presents a user-based evaluation accomplished using 15 different “real world” scenarios. Finally, Section 5 concludes the article with a brief talk about possible future work.

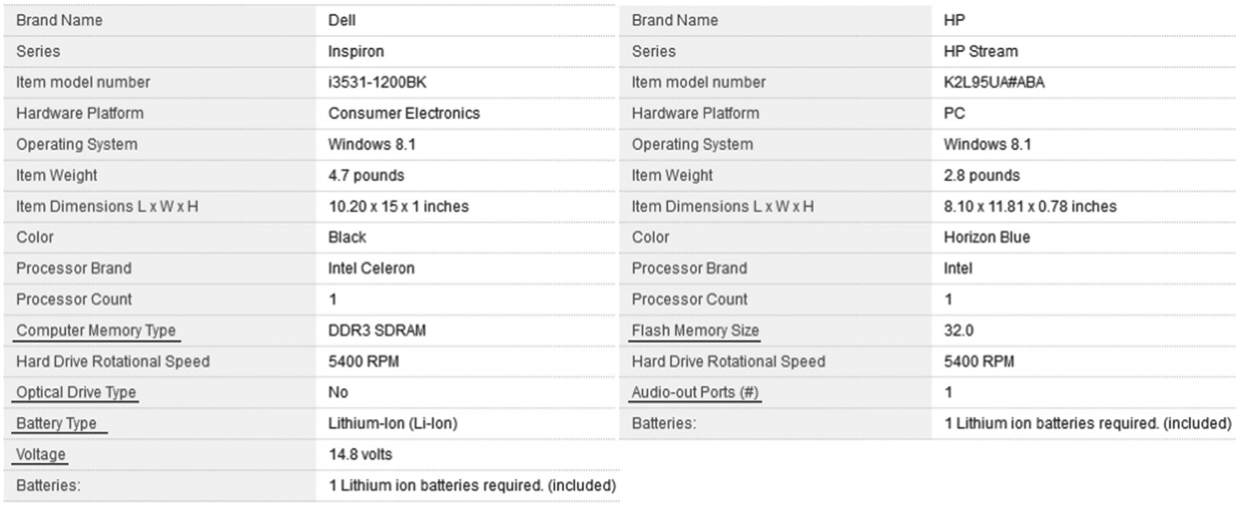
2. Background knowledge and related work

The information extraction systems can be divided into following three categories [10]:

- *Procedural wrapper*: The approach is based on writing customized wrappers for accessing required data from a given set of information sources. In these systems the extraction rules are coded into the program.
- *Declarative wrapper*: These systems consist of a general execution engine and declarative extraction rules developed for specific data sources.
- *Automatic wrapper*: These systems use machine learning techniques to learn extraction rules by examples.

In [11] information extraction systems are divided with respect to the level of automation of wrapper creation into *manual*, *semi-automatic* (i.e., based on user interaction with user interface) and *automatic* (using supervised machine learning techniques). In this paper, we present a tool that has two main components: a wrapper and an ontology builder.

The main component of the proposed tool is a declarative and manual wrapper, which has a general rule engine that executes the extraction rules specified in a template file created manually by the user. The extraction rules are specified using our domain-specific language (Appendix D) based on XML Path Language (XPath) [12]. XPath is a query language that can be used for selecting arbitrary parts of HTML documents. In addition to specifying the XPath query, these rules also specify how to convert the extracted information to the internal object format. The template file also provides information on how the HTML documents should be acquired.



Brand Name	Dell
Series	Inspiron
Item model number	i3531-1200BK
Hardware Platform	Consumer Electronics
Operating System	Windows 8.1
Item Weight	4.7 pounds
Item Dimensions L x W x H	10.20 x 15 x 1 inches
Color	Black
Processor Brand	Intel Celeron
Processor Count	1
Computer Memory Type	DDR3 SDRAM
Hard Drive Rotational Speed	5400 RPM
Optical Drive Type	No
Battery Type	Lithium-Ion (Li-Ion)
Voltage	14.8 volts
Batteries:	1 Lithium ion batteries required. (included)

Brand Name	HP
Series	HP Stream
Item model number	K2L95UA#ABA
Hardware Platform	PC
Operating System	Windows 8.1
Item Weight	2.8 pounds
Item Dimensions L x W x H	8.10 x 11.81 x 0.78 inches
Color	Horizon Blue
Processor Brand	Intel
Processor Count	1
Flash Memory Size	32.0
Hard Drive Rotational Speed	5400 RPM
Audio-out Ports (#)	1
Batteries:	1 Lithium ion batteries required. (included)

Fig. 2. The underlined properties differ for the example items of the same kind.

Manual approaches are known to be tedious, time-consuming and require some level of expertise concerning the wrapper language [13]. Besides, when web sites change, updating of specification files have to be done manually by user. However, manual and semi-automatic approaches are currently better suited for creating robust wrappers than the automatic approach. The maintenance costs of current automatic approaches are also comparable to manual and semi-automatic approaches, since in the automatic approach the user has to annotate new training samples when the wrapped web pages are modified [11].

Writing a template for the tool is considerably easier than most of the existing manual wrappers; however, it requires users' understanding of the HTML parse tree, the identification of the separating tags for rows and columns in a table, etc. Thus, such kinds of approaches are classified as systems that require special expertise of users [14].

Various researchers have proposed methods and developed tools toward the web information extraction task [15]. In the manual approach (e.g. Jedi [16], TSIMMIS [17], and W4F [18]), users create general extraction rules by analyzing a representative set of web pages, and they are responsible for updating the specification files when necessary. In automatic generation (e.g. WIEN [19], SoftMealy [20], and Stalker [21]), machine learning techniques are applied to the training examples provided by the user. Semi-automatic approaches, such as XWRAP [22] and Lixto [23], try to make wrapper generation easier through mappings between the visual and text or DOM views.

All of the work above is about extracting information from web pages. However, there are works which directly focus on the problem of this paper—product feature extraction from web. The first one [24] uses a template-independent approach to extract product features from web. They use a co-training algorithm with naive Bayesian classifier to identify the candidate pairs. This approach makes hypothesis to identify the specification block but since some detail product pages may violate these hypothesis, the pairs in these pages cannot be extracted properly. Besides, this approach needs training data. The second work [25] needs two predefined ontologies (table ontology and domain ontology) to extract product features from a web page. The table ontology is a standard ontology for every web page. On the other hand the domain ontology is built according to the contents of the page and it is not an easy task to build the domain ontology from scratch for every change in the page content. Finally, the third work [2] is based on unsupervised adaptive template construction. This approach is evaluated using data corpus in two different domains.

Portia is an emerging tool[2] that allows you to visually scrape web sites without any programming knowledge required. With Portia you can annotate a web page to identify the data you wish to extract, and Portia will understand based on these annotations how to scrape data from similar pages. For example, a product may have fields named name, price, manufacturer and so on, annotations are used to extract data from the page into each of these fields. You can define annotations by highlighting or clicking elements on the page. The problem lies with the fact that some pages containing the same item type may have a different layout or fields missing, and you will need to accommodate those pages by creating a template for each layout variation (see Fig. 2).

The tool presented in this paper differs from all the above works in many ways. First, it semi-automatically transforms the extracted information to an ontology [26] in order to share and reuse common understanding of structure of information among users or software agents.

Second, it provides a domain-specific language, which has a very concise but powerful syntax based on XPath. This language primarily serves to make it easy for users to create web-site specific templates.

[2] https://github.com/scrapinghub/portia

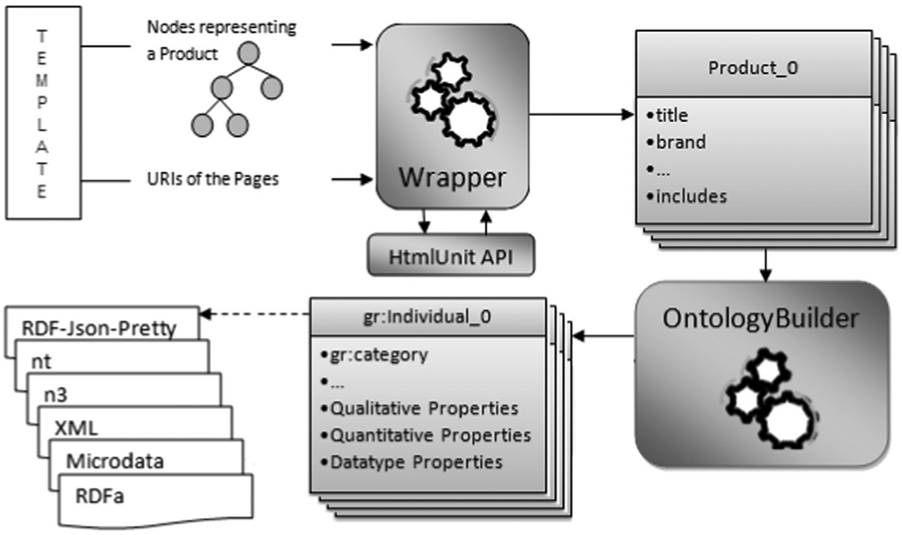


Fig. 3. Architecture of the system.

Third, according to [27], the proposed tool is the first Protégé plug-in that is used to extract product features from web pages. As an extension to Protégé, it exploits the advantages of the ontology as a formal model for the domain knowledge and profits from the benefits of a large user community (currently 230,914 registered users), a library of reusable components and a flexible architecture.

Fourth, over time, many of web sites have started to use composite web documents (using frames and JavaScript to combine multiple HTML document), complex server interaction paradigms (such as AJAX), advanced user interface elements (based on a wide use of JavaScript and Flash components, as well as on the rich event model) and different types of stateful design patterns. The problem of extracting information from web sites using these technologies (referred to as complex data-intensive web sites) is addressed in [3].

The tool uses HtmlUnit WebDriver [28] for parsing the web page into DOM tree. HtmlUnit is currently the fastest implementation of WebDriver, a pure Java solution and so it is platform independent and has fairly good JavaScript support and is able to work even with quite complex AJAX libraries. As a result, the system partially handles the problem of extracting information from complex-data intensive web sites.

Fifth, the tool supports marking up your product information in a way that is compatible with GoodRelations Vocabulary [7], which is currently supported by Google and Yahoo. Search engines use your markup to augment the preview of your product in the search results preview which certainly will increase the visibility of your product among other results.

Finally, as an open source Java Application, the tool can be further extended, fixed or modified according to the needs of the individual users.

3. Scenario-based system specification

The proposed tool gathers semi-structured product information from an HTML page, applies extraction rules specified in the template file, and presents the extracted product data in an ontology that is compatible with GoodRelations Vocabulary. It has two main components: the wrapper and the ontology builder (Fig. 3).

The wrapper extracts the product data from the web page using the template file. The HTML page is first parsed into a DOM tree using HtmlUnit, which is a web driver that supports walking the DOM model of the HTML document using XPath queries. The template file includes a tree that specifies the paths of HTML tags around the product features. The system evaluates the nodes in the template and queries the HtmlUnit for the required product properties. At the end of this process, the system returns a list of product objects.

Then the user classifies each extracted property using the property categories of the GoodRelations ontology and the ontology builder automatically builds a GoodRelations-compliant ontology. Finally, the tool serializes the ontology into a series of structured data markup standards using RDF Translator API [29].

The following subsections describe building a template file and a GoodRelations-compliant ontology.

3.1. Building a template file

Creating a template file is the fundamental step for using the tool. The information collected using the template is mapped to the attributes of the *Product* object which is represented in Fig. 4. *Title*, *brand*, *productID*, *description*, and *imgLink* are single-valued attributes whereas *feature*, *includes*, *propertyName* and *propertyValue* are multi-valued attributes. For the *propertyName* and *propertyValue* arrays, the elements at the same index (*propertyName*[*x*]-*propertyValue*[*x*]) constitute a *property name–value* pair.

A template has two parts: the first part contains the tree that specifies the paths of HTML tags around the product features and the second part contains the information on how the HTML documents should be acquired.

The first example uses "amazon.com" pages that contain the search results about laptops. By the time this example was written, the search results contained 17,714 laptops spread across 400 pages. Fig. 5(a) shows the first product on the first page. We create path expressions to select the parts of the HTML document that contains the product features. There may be more than one path to select a specific part of the HTML document.

One possible path to select the title of this product is given in Fig. 5(b). This path shows the *h3* heading tag having a *class* attribute with the value "a-size-base s-inline s-access-title a-text-normal" and nested in the HTML list having an *id* attribute with the value "*result*_0".

Possible paths to select the image link and the link that goes to the web page that contains the details of the product are given in Fig. 5(c) and (d) respectively. The first path shows the image link having a *src* attribute and nested in the HTML list having *id* attribute with the value "*result*_0". The second path shows the link having *class* attribute with value "*a-link-normal s-access-detail-page a-text-normal*" and nested in the HTML list having *id* attribute with the value "*result*_0".

When you follow the link in Fig. 5(d), the destination page includes the HTML table containing the details of the first product (Fig. 6(a)). We create the path expressions to select the parts of the HTML document that contains the cells of the table. Cells having a *class* attribute with the value "*label*" contain the property names of the product. Cells having a *class* attribute with value the "*value*" contain the property values of the product. One possible path to select the property names is given in Fig. 6(b) and a possible path to select the property value is given in Fig. 6(c).

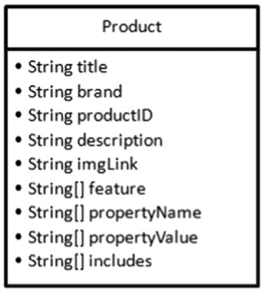


Fig. 4. The attributes of the *Product* object.

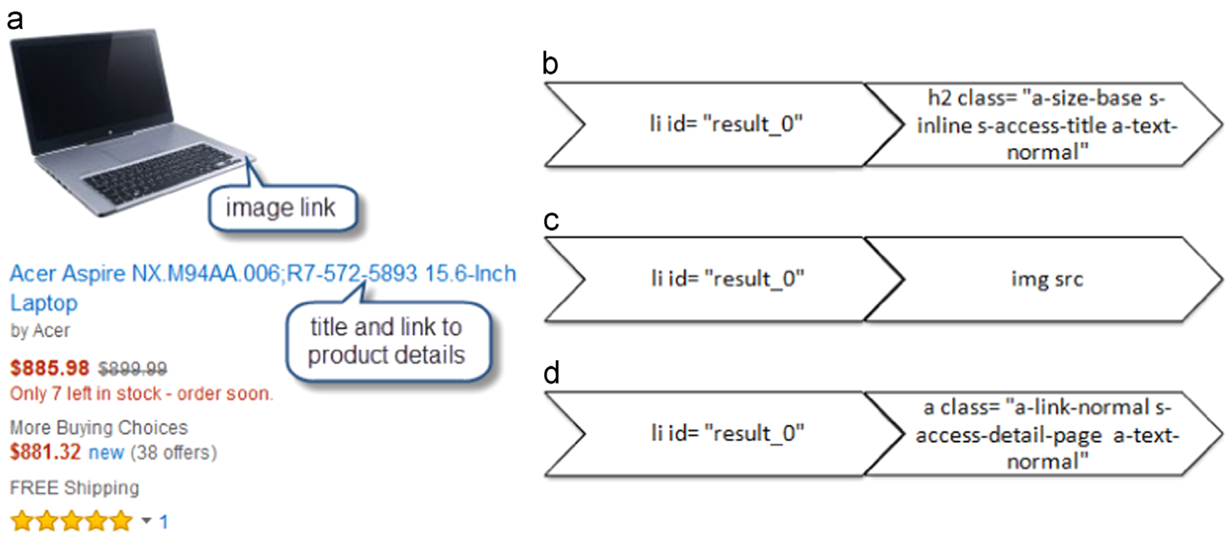
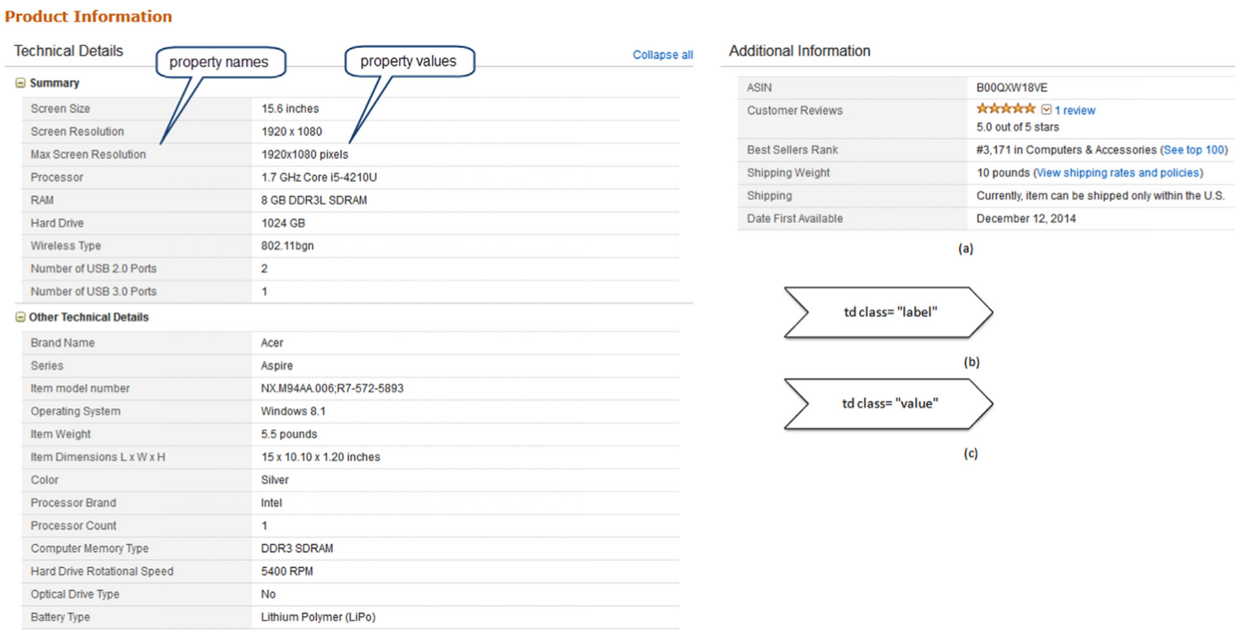

a
image link
Acer Aspire NX.M94AA.006;R7-572-5893 15.6-Inch Laptop
by Acer
$885.98 $899.99
Only 7 left in stock - order soon.
More Buying Choices
$881.32 new (38 offers)
FREE Shipping
1
title and link to product details
b
li id= "result_0"
h2 class= "a-size-base s-inline s-access-title a-text-normal"
c
li id= "result_0"
img src
d
li id= "result_0"
a class= "a-link-normal s-access-detail-page a-text-normal"

Fig. 5. The paths selecting the product title, product image link and the link that goes to the web page that contains the details of the product.



Product Information

Technical Details

Summary

Screen Size	15.6 inches
Screen Resolution	1920 x 1080
Max Screen Resolution	1920x1080 pixels
Processor	1.7 GHz Core i5-4210U
RAM	8 GB DDR3L SDRAM
Hard Drive	1024 GB
Wireless Type	802.11bgn
Number of USB 2.0 Ports	2
Number of USB 3.0 Ports	1

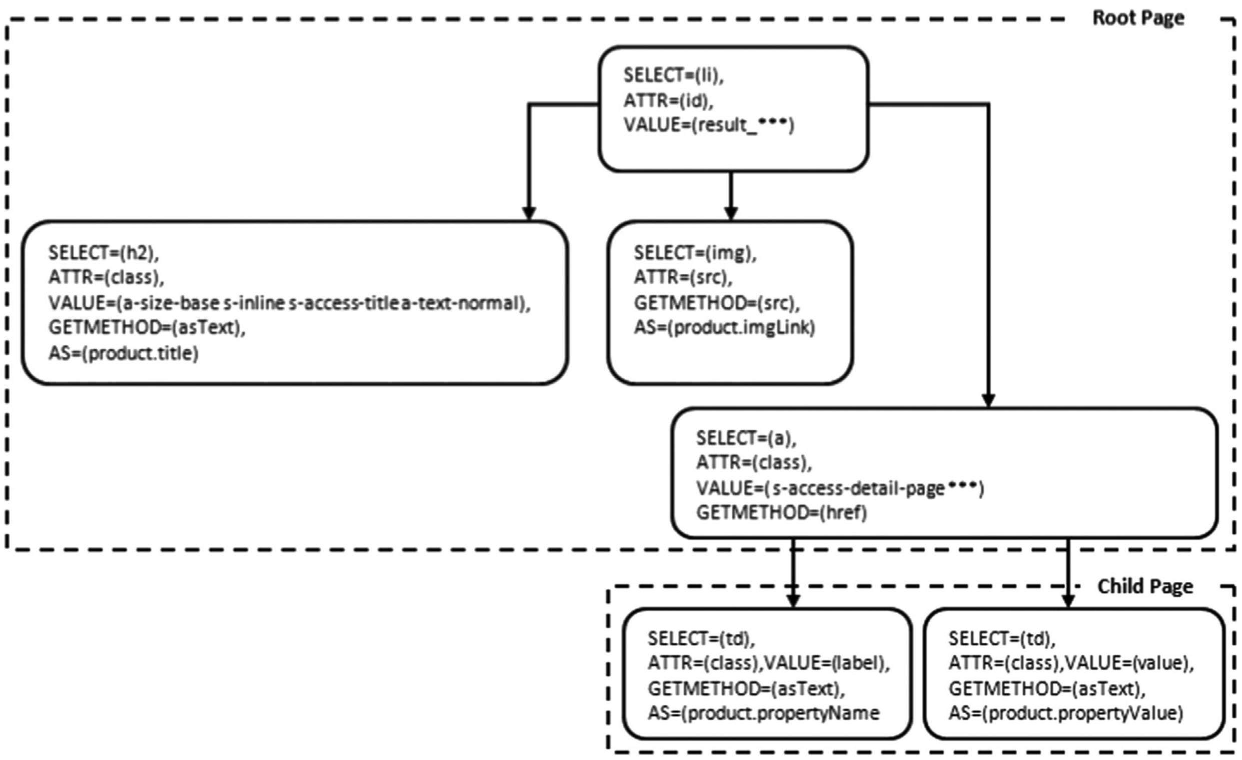
Other Technical Details

Brand Name	Acer
Series	Aspire
Item model number	NX.M94AA.006;R7-572-5893
Operating System	Windows 8.1
Item Weight	5.5 pounds
Item Dimensions L x W x H	15 x 10.10 x 1.20 inches
Color	Silver
Processor Brand	Intel
Processor Count	1
Computer Memory Type	DDR3 SDRAM
Hard Drive Rotational Speed	5400 RPM
Optical Drive Type	No
Battery Type	Lithium Polymer (LiPo)

Additional Information

ASIN	B00QXW18VE
Customer Reviews	★★★★★ 1 review 5.0 out of 5 stars
Best Sellers Rank	#3,171 in Computers & Accessories (See top 100)
Shipping Weight	10 pounds (View shipping rates and policies)
Shipping	Currently, item can be shipped only within the U.S.
Date First Available	December 12, 2014

Fig. 6. The paths selecting the cells of the HTML table with details of the first product.



```
SELECT=(li),
ATTR=(id),
VALUE=(result_***)
```

```
SELECT=(h2),
ATTR=(class),
VALUE=(a-size-base s-inline s-access-title a-text-normal),
GETMETHOD=(asText),
AS=(product.title)
```

```
SELECT=(img),
ATTR=(src),
GETMETHOD=(src),
AS=(product.imgLink)
```

```
SELECT=(a),
ATTR=(class),
VALUE=(s-access-detail-page***)
GETMETHOD=(href)
```

```
SELECT=(td),
ATTR=(class),VALUE=(label),
GETMETHOD=(asText),
AS=(product.propertyName
```

```
SELECT=(td),
ATTR=(class),VALUE=(value),
GETMETHOD=(asText),
AS=(product.propertyValue)
```

Fig. 7. The tree nodes for extracting “Laptops” from “http://www.amazon.com”.

After analyzing the web pages about the first product and defining the paths to the relevant information about the product, we built the tree in Fig. 7 for extracting product features from the web page.

The full path of the relevant information is built by ordering all the nodes from root to leaf. Since there may be more than one path to the relevant information, it is possible to create alternative trees for extracting the same information.

Each node of the tree specifies an HTML element in the web page. The nodes of the tree are specified in the template file in Appendix A according to our domain-specific language, whose EBNF notation is represented in Appendix D. SELECT-ATTR-VALUE fields are used to specify the HTML element. The root node defines the HTML elements containing the information of a specific product on the page. In this example, each product is an HTML list item. The first item has an *id* attribute with the value *result*_0, the second one has an *id* attribute with the value *result*_1, and so on. Therefore to specify the path to a product on the page, we select HTML list items, which have *id* attributes with values containing the string *result*_. Three asterisk characters in the VALUE field of the root node (Fig. 7) substitute for zero or more characters at the beginning and the end of the value.

GETMETHOD field collects the proper values in the selected HTML element. This field is used only with the leaf nodes or the internal nodes returning an HTML link element. If you want to get the textual representation of the element, you define the value of the GETMETHOD field as "*asText*". On the other hand if you want to get the value of an element attribute then you specify the name of that attribute as the value of this field.

The AS field is only used with leaf nodes. The value which is collected from a leaf node using GETMETHOD field is mapped to the product attribute specified in the AS field.

The first part of the template (Appendix A) contains the tree in Fig. 7 encoded using the syntax described above. The second part of the template contains the information on how the HTML documents should be acquired. The fields in the second part of the template are as follows:

- NEXT_PAGE: If the information about products is spread across multiple web pages, we use this field to specify the HTML link of the next page. The link is determined by a link element in the HTML code (see Appendix A) of the page or by an open parameter in the URL of the page. In the second case, the page number is specified in the URL of the page, e.g. $\&page = 1$. For navigating to the next page, the page number is increased by a number n. For example, if the page parameter is $\&page =$ and n equals 1 then the value of the next page field is $[\&page =][+1]$.
- PAGE_RANGE: This field stores the page number or the range of pages including the information to be extracted.
- BASE_URI: This field represents the base URI of the web site.
- PAGE_URI: This field stores the URI of the first page including the information to be extracted.
- CLASS: This field stores the name of the class that represents the products to be extracted.

In some cases, it is required to define text operations (e.g. concatenate, contains, fragment, lower, upper, replace, substring, and trim) on tree nodes. For instance "color" property and its value "black" can be organized in the same text node as follows: ⟨span⟩color: red⟨/span⟩. The values in this text node should be fragmented and "color" should be mapped to property name and "black" should be mapped to property value.

Appendix B represents a template to collect kids' books from "barnesandnoble" pages. Leaf node in grey represents a text node which contains both property name and property value. Therefore the AS field has two values "product.propertyName" and "product.propertyValue" respectively.

OPERATORS field represents the operations to be executed on the value of the text node. In this example, the first operation is "split", the first operand is "U+003A" which is the hexadecimal code of the character ":". This returns a string array that contains the substrings in the value of the text node that are delimited by ":". The second operand "0" gets the array element at index 0. Finally, the third operand "0" assigns this array element to the value in the AS field with index 0, namely "product.propertyName". The second operation is "split" again, and the first operand is "U+003A" which is the hexadecimal code of the character ":". This returns a string array that contains the substrings in the value of the text node that are delimited by ":". The second operand "1" gets the array element at index 1. Finally, the third operand "1" assigns this array element to the value in the AS field with index 1, namely "product.propertyValue". Appendix C represents the whole list of operators that are supported by the tool.

OPERATORS field is also used for text node fragmentation based on HTML text formatting (bold, important, italic, emphasized, marked, small, deleted, inserted, subscripts and superscripts). For instance "color" property and its value "black" can be organized in the same text node as follows:

⟨span⟩⟨strong⟩color⟨/strong⟩black⟨/span⟩.

The values in this text node can be fragmented in the following way:

$$\begin{aligned}&SELECT = (span), GETMETHOD = (asText),\\&\quad AS = (\{product.propertyName\}\{product.propertyValue\}),\\&\quad OPERATORS = (\{strong\}\{\{0\}\}\{exceptStrong\}\{\{1\}\})\end{aligned}$$

We first get the text element between ⟨strong⟩ tags and assign this text to the value in the AS field with index 0, namely "product.propertyName". Then we get the all text except the text between ⟨strong⟩ tags and assign it to the value in the AS field with index 1, namely "product.propertyValue".

We can also apply more than one operator on a specific value in the text node. The operators with a common last operand value are applied in order. In the following HTML code, there is an asterisk character at the beginning of the property name which is formatted in bold:

⟨span⟩⟨strong⟩*color⟨/strong⟩black⟨/span⟩.

To get the property name properly, we first get the text between ⟨strong⟩ tags and remove the "*" character at the beginning of the string as follows:

$$SELECT = (span), GETMETHOD = (asText),$$
$$AS = (\{product.propertyName\}\{product.propertyValue\}),$$
$$OPERATORS = (\{strong\}\{\{0\}\}\{exceptStrong\}\{\{1\}\}\{substring\}\{\{1\}\{0\}\})$$

As you can see above, operators "strong" and "substring" have the last operand value 0, therefore these operators are executed in order. The output of the strong operator is the input of the substring operator and the result will be assigned to the value in the AS field with index 0 (the value of the last operand). We apply the substring operator to the text between

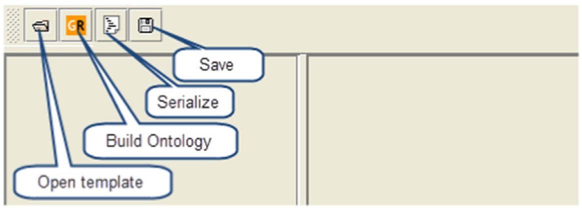
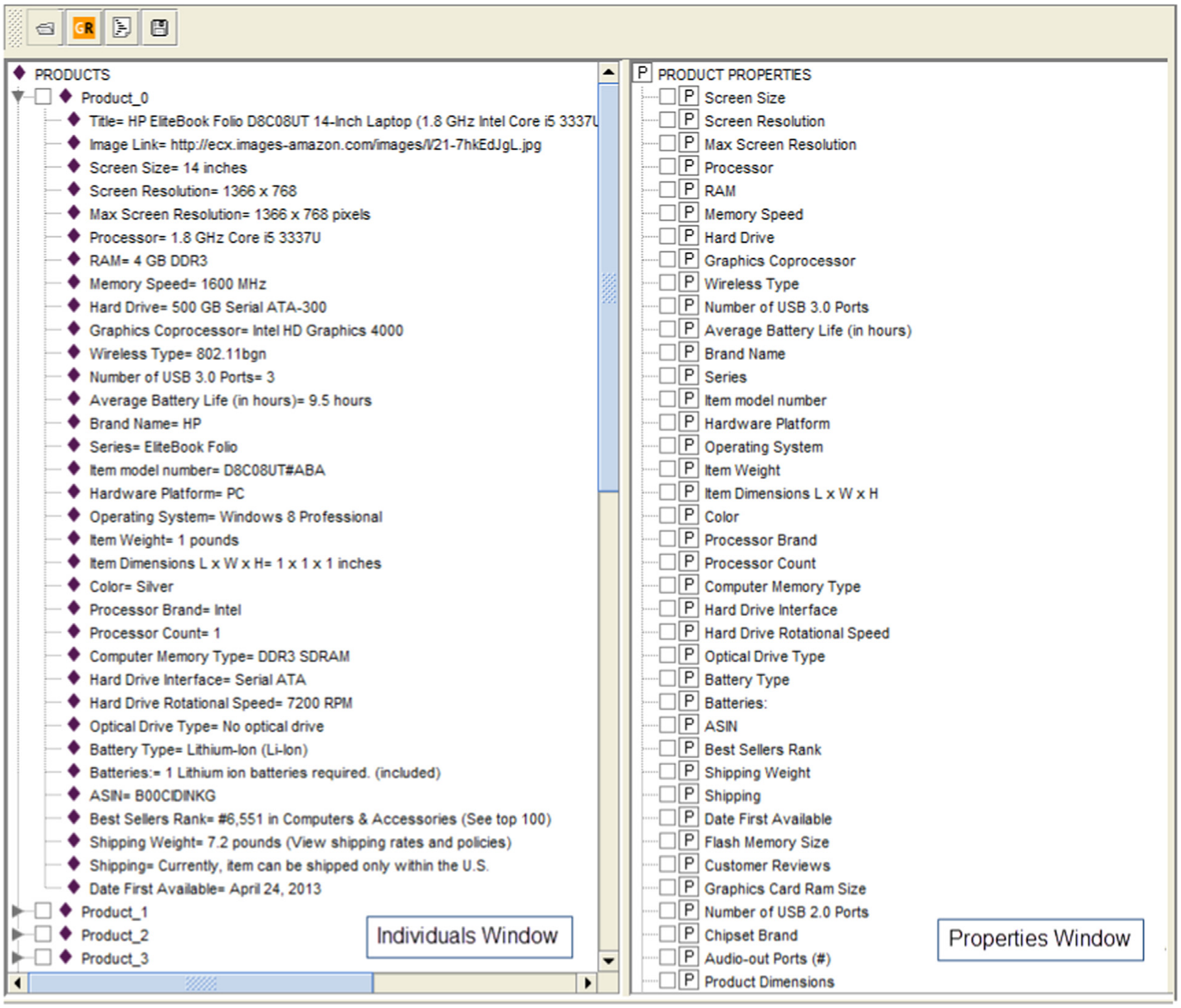


Fig. 8. The new tab in Protégé Ontology Editor.



PRODUCTS
Product_0
Title= HP EliteBook Folio D8C08UT 14-Inch Laptop (1.8 GHz Intel Core i5 3337U
Image Link= http://ecx.images-amazon.com/images/I/21-7hkEdJgL.jpg
Screen Size= 14 inches
Screen Resolution= 1366 x 768
Max Screen Resolution= 1366 x 768 pixels
Processor= 1.8 GHz Core i5 3337U
RAM= 4 GB DDR3
Memory Speed= 1600 MHz
Hard Drive= 500 GB Serial ATA-300
Graphics Coprocessor= Intel HD Graphics 4000
Wireless Type= 802.11bgn
Number of USB 3.0 Ports= 3
Average Battery Life (in hours)= 9.5 hours
Brand Name= HP
Series= EliteBook Folio
Item model number= D8C08UT#ABA
Hardware Platform= PC
Operating System= Windows 8 Professional
Item Weight= 1 pounds
Item Dimensions L x W x H= 1 x 1 x 1 inches
Color= Silver
Processor Brand= Intel
Processor Count= 1
Computer Memory Type= DDR3 SDRAM
Hard Drive Interface= Serial ATA
Hard Drive Rotational Speed= 7200 RPM
Optical Drive Type= No optical drive
Battery Type= Lithium-Ion (Li-Ion)
Batteries:= 1 Lithium ion batteries required. (included)
ASIN= B00CIDINKG
Best Sellers Rank= #6,551 in Computers & Accessories (See top 100)
Shipping Weight= 7.2 pounds (View shipping rates and policies)
Shipping= Currently, item can be shipped only within the U.S.
Date First Available= April 24, 2013
Product_1
Product_2
Product_3

PRODUCT PROPERTIES
Screen Size
Screen Resolution
Max Screen Resolution
Processor
RAM
Memory Speed
Hard Drive
Graphics Coprocessor
Wireless Type
Number of USB 3.0 Ports
Average Battery Life (in hours)
Brand Name
Series
Item model number
Hardware Platform
Operating System
Item Weight
Item Dimensions L x W x H
Color
Processor Brand
Processor Count
Computer Memory Type
Hard Drive Interface
Hard Drive Rotational Speed
Optical Drive Type
Battery Type
Batteries:
ASIN
Best Sellers Rank
Shipping Weight
Shipping
Date First Available
Flash Memory Size
Customer Reviews
Graphics Card Ram Size
Number of USB 2.0 Ports
Chipset Brand
Audio-out Ports (#)
Product Dimensions

Fig. 9. The tool imports all laptops from the specified "http://www.amazon.com" pages.

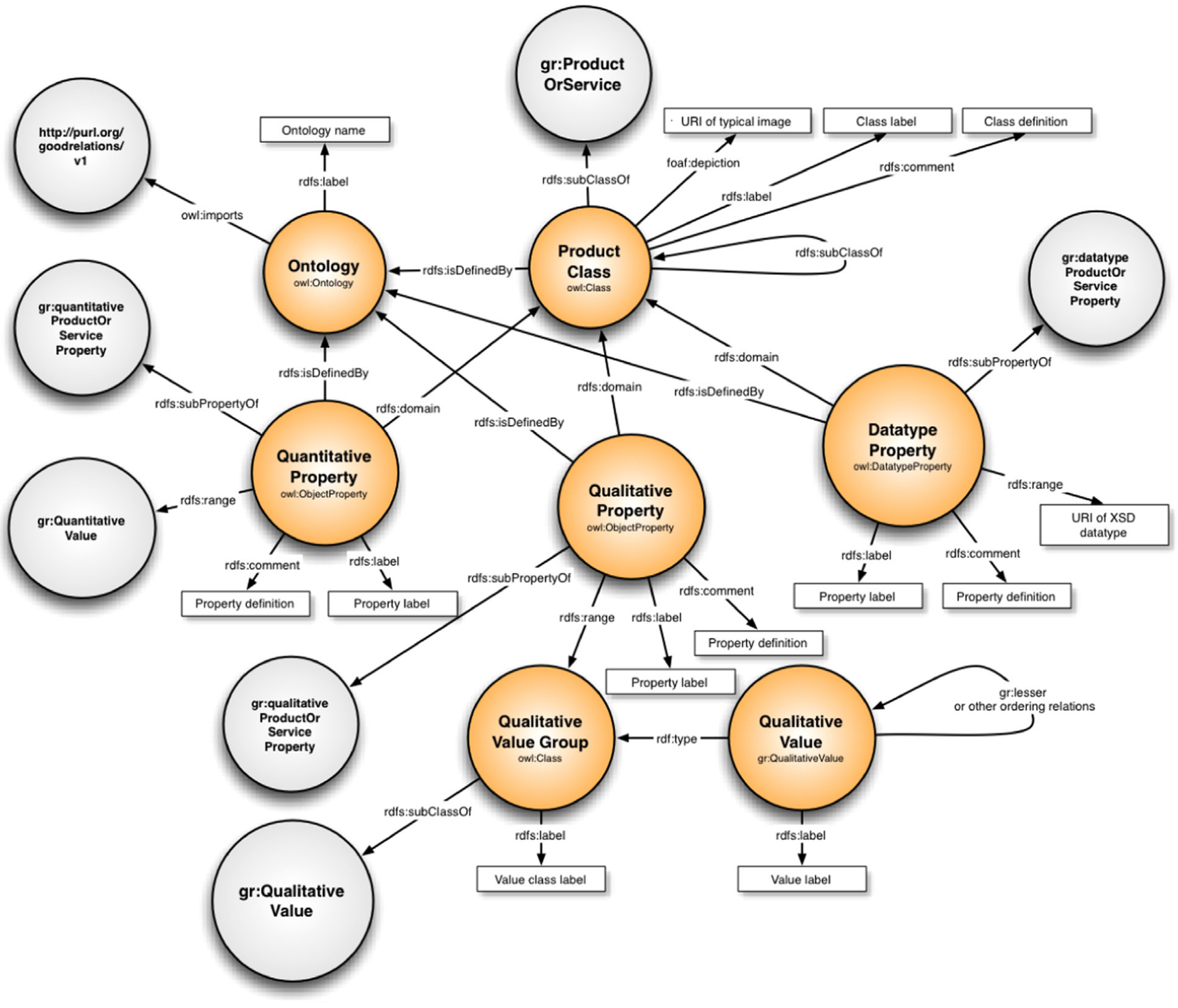
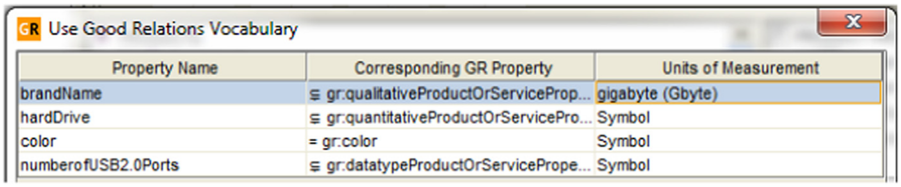


Fig. 10. Pattern for GoodRelations-compliant product ontologies.



Property Name	Corresponding GR Property	Units of Measurement
brandName	⊆ gr:qualitativeProductOrServiceProp...	gigabyte (Gbyte)
hardDrive	⊆ gr:quantitativeProductOrServicePro...	Symbol
color	= gr:color	Symbol
numberofUSB2.0Ports	⊆ gr:datatypeProductOrServicePrope...	Symbol

Fig. 11. Selecting the types and units of properties using "Use GoodRelations" wizard.

the strong tags ("*black"). The substring function returns the substring beginning with the character at the index specified in the first operand. The first operand of substring is 1, therefore we get the substring "black" and assign this value to the "product.propertyName".

3.2. Building the ontology

The tool is developed as a plug-in for Protégé Ontology Editor therefore the user interface of Protégé is extended with additional tabs to access the modules of the tool. First the user opens an empty ontology in Protégé Editor and chooses

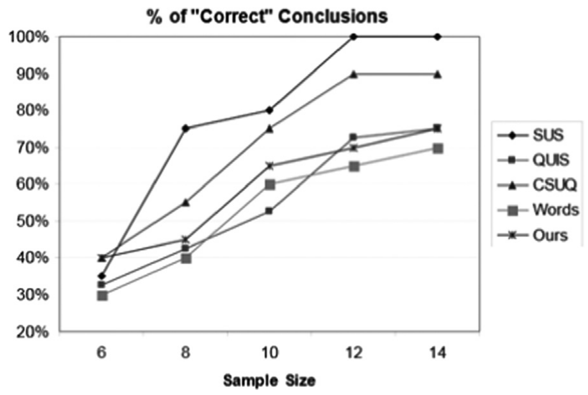


Fig. 12. Comparison of the significance of the different usability questionnaires.

Table 1
The experience levels of participants.

Subject	*N*	AVG	DEV	MED	MAX	MIN
HTML	15	4.13	0.92	4	5	2
Ontology-based systems	15	3.20	1.61	4	5	1
Protégé environment	15	2.67	1.76	2	5	1

Table 2
The three internal subscales of CSUQ.

Score name	Average to responses to
Overall	Questions 1–19
System usefulness	Questions 1–8
Information quality	Questions 9–15
Interface quality	Questions 16–18

Table 3
CSUQ evaluation results.

Question	*N*	AVG	DEV	MED	MAX	MIN
Overall, I am satisfied with how easy it is to use this system	15	5.00	0.65	5	6	4
It was simple to use this system	15	4.73	0.70	5	6	4
I can effectively complete my work using this system	15	5.60	1.12	6	7	4
I am able to complete my work quickly using this system	15	5.67	0.82	5	7	5
I am able to efficiently complete my work using this system	15	5.93	1.16	6	7	4
I feel comfortable using this system	15	5.07	1.33	5	7	2
It was easy to learn to use this system	15	4.87	1.30	5	7	3
I believe I became productive quickly using this system	15	5.93	0.96	6	7	4
The system gives error messages that clearly tell me how to fix problems	15	3.13	0.92	3	5	2
Whenever I make a mistake using the system. I recover easily and quickly	15	3.67	1.40	4	6	2
The information (such as online help, on-screen messages, and other documentation) provided with this system is clear	15	4.80	1.08	5	6	3
It is easy to find the information I needed	15	5.00	1.00	5	7	3
The information provided for the system is easy to understand	15	5.40	0.91	6	7	4
The information is effective in helping me complete the tasks and scenarios	15	5.53	0.92	6	7	4
The organization of information on the system screens is clear	15	5.53	1.06	6	7	4
The interface of this system is pleasant	15	5.00	1.00	5	7	4
I like using the interface of this system	15	5.20	0.86	5	7	4
This system has all the functions and capabilities I expect it to have	15	5.80	0.77	6	7	4
Overall. I am satisfied with this system	15	5.27	0.88	5	7	4

Table 4
CSUQ single participant evaluation scores.

Respondent no	Overall	System usefulness	Information quality	Interface quality	Average	Score value
1	4.53	5.25	3.29	5.00	4.52	64.51
2	4.95	5.00	4.71	5.33	5.00	71.41
3	4.42	5.00	3.57	5.00	4.50	64.26
4	5.11	5.25	4.71	5.67	5.18	74.06
5	4.84	5.25	4.57	4.33	4.75	67.85
6	5.16	5.00	4.71	6.33	5.30	75.73
7	5.11	5.88	4.00	5.33	5.08	72.55
8	5.21	5.13	5.14	5.67	5.29	75.52
9	5.68	5.88	5.00	6.67	5.81	82.95
10	4.89	4.63	5.14	5.33	5.00	71.41
11	5.32	5.38	5.86	4.00	5.14	73.39
12	5.95	6.63	5.43	5.00	5.75	82.15
13	5.37	5.13	5.43	5.67	5.40	77.10
14	5.26	5.38	5.29	5.00	5.23	74.73
15	4.89	5.50	4.00	5.67	5.02	71.65
Average	5.11	5.35	4.72	5.33		
Score value	73.03	76.43	67.48	76.19		

Table 5
Items of the SUS questionnaire.

No	Question
1	I think that I would like to use this system frequently
2	I found the system unnecessarily complex
3	I thought the system was easy to use
4	I think that I would need the support of a technical person to be able to use this system
5	I found the various functions in this system were well integrated
6	I thought there was too much inconsistency in this system
7	I would imagine that most people would learn to use this system very quickly
8	I found the system very cumbersome to use
9	I felt very confident using the system
10	I needed to learn a lot of things before I could get going with this system

"Project - Configure…" to bring up the "Configure" dialog. The new tab will be listed on the TabWidgets panel. The user checks the "Visible" checkbox and click the OK button. The system will now display the tab as in Fig. 8.

After opening the ontology, the user selects the template that is be used to import product features from the web. In this example, we choose the amazon.txt by clicking the "Open template"' button in Fig. 8. Then the tool imports all laptops from the "amazon.com"' pages specified in the PAGE_RANGE field of the template in Appendix A (Fig. 9). The imported individuals are listed in the "Individuals Window". Fig. 9 shows the expanded version of the Product_0 node, whose children store all properties of Product_0 and their corresponding values. The "Properties Window" lists all properties of the individuals in "Individuals Window".

Next the user specifies how the new ontology must be designed in order to be compatible with the GoodRelations ontology in Fig. 10. All the extracted properties should be categorized under the appropriate GoodRelations product category in the following way:

- p_x is mapped to p_y, if the property p_x is semantically equivalent of the property p_y and p_y is a property from the following list: "gr:category", "gr:color", "gr:condition", "gr:depth", "gr:description", "gr:hasBrand", "gr:hasEAN UCC-13", "gr:hasGTIN-14", "gr:hasGTIN-8", "gr:hasMPN", "gr:hasMakeAndModel", "gr:hasManufacturer","gr:hasStockKeepingUnit", "gr:height", "gr: isAccessoryOrSparePartFor", "gr:isConsumableFor", "gr:isSimilarTo", "gr:name", "gr:serialNumber", "gr:weight", "gr:width".
- All properties that specify quantitative characteristics, for which an interval is at least theoretically an appropriate value, are defined as subproperties of "gr:quantitativeProductOrServiceProperty". Allowed values for a "quantitativeProductOrService-Property" are "gr:QuantitativeValue", "gr:QuantitativeValueFloat" and "gr:QuantitativeValueInteger". A quantitative value is a numerical interval that represents the range of a certain "gr:quantitativeProductOrServiceProperty" in terms of the lower and

Table 6
SUS evaluation results.

Question no	Responder 1	Responder 2	Responder 3	Responder 4	Responder 5	Responder 6	Responder 7	Responder 8	Responder 9	Responder 10	Responder 11	Responder 12	Responder 13	Responder 14	Responder 15
1	3	4	2	3	3	3	2	2	4	3	3	4	4	4	4
2	3	4	3	3	3	4	3	4	3	4	4	4	3	4	3
3	2	3	2	3	2	3	3	3	3	3	3	3	3	3	3
4	4	3	3	3	4	4	4	4	4	4	3	4	3	3	4
5	4	2	4	3	3	4	2	3	4	3	4	3	3	4	3
6	4	4	4	4	4	4	3	4	4	4	4	4	2	4	4
7	2	3	2	3	2	2	3	4	2	2	2	4	4	3	2
8	3	4	2	3	4	3	4	4	4	3	3	4	4	4	3
9	1	3	1	2	2	3	3	3	4	2	3	4	2	3	3
10	4	4	3	3	3	2	3	4	3	3	3	3	2	2	3
Sum	30	34	26	30	30	32	30	35	35	31	32	37	30	34	32
Score value	75	85	65	75	75	80	75	87.5	87.5	77.5	80	92.5	75	85	80

Table 7
SUS scores with their corresponding adjective [33] and acceptability ratings [34].

SUS scores	Adjective ratings	Acceptability
89–100	Best imaginable	
84–88	Excellent	Acceptable
71–83	Good	
50–70	OK	Marginal
32–49	Poor	
20–31	Awful	Unacceptable
0–19	Worst imaginable	

Table 8
The complete set of words in the Microsoft Reaction Card.

Accessible	Creative	Fast	Meaningful	Slow
Advanced	Customizable	Flexible	Motivating	Sophisticated
Annoying	Cutting edge	Fragile	Not secure	Stable
Appealing	Dated	Fresh	Not valuable	Sterile
Approachable	Desirable	Friendly	Novel	Stimulating
Attractive	Difficult	Frustrating	Old	Straight forward
Boring	Disconnected	Fun	Optimistic	Stressful
Business-like	Disruptive	Gets in the way	Ordinary	Time-consuming
Busy	Distracting	Hard to use	Organized	Time-saving
Calm	Dull	Helpful	Overbearing	Too technical
Clean	Easy to use	High quality	Overwhelming	Trustworthy
Clear	Effective	Impersonal	Patronizing	Unapproachable
Collaborative	Efficient	Impressive	Personal	Unattractive
Comfortable	Effortless	Incomprehensible	Poor quality	Uncontrollable
Compatible	Empowering	Inconsistent	Powerful	Unconventional
Compelling	Energetic	Ineffective	Predictable	Understandable
Complex	Engaging	Innovative	Professional	Undesirable
Comprehensive	Entertaining	Inspiring	Relevant	Unpredictable
Confident	Enthusiastic	Integrated	Reliable	Unrefined
Confusing	Essential	Intimidating	Responsive	Usable
Collaborative	Efficient	Impressive	Personal	Unattractive
Consistent	Exciting	Inviting	Satisfying	Valuable
Controllable	Expected	Irrelevant	Secure	
Convenient	Familiar	Low maintenance	Simplistic	

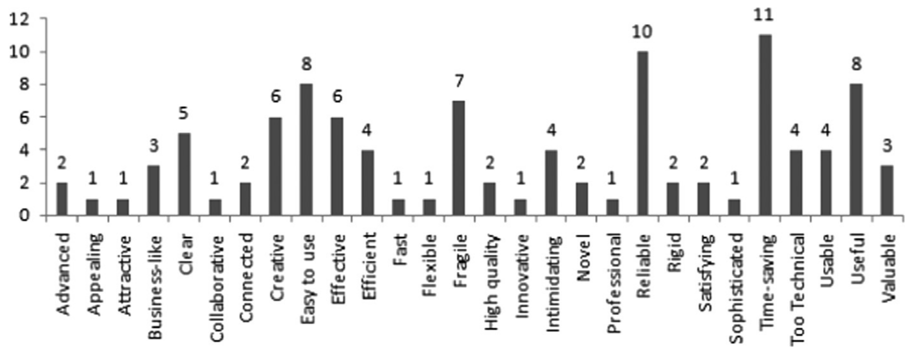


Fig. 13. Selected words in MPRC.

upper bounds for a particular “gr:ProductOrService”. It is to be interpreted in combination with the respective unit of measurement. Most quantitative values are intervals even if they are in practice often treated as a single point value. *Example:* a temperature between 60 and 67 °F. Note that some properties of the first category (“gr:depth”, “gr:height”, “gr:weight”, and “gr:width”) are defined as subproperties of “gr:quantitativeProductOrServiceProperty”.

- All properties for which value instances are specified are subproperties of “gr:qualitativeProductOrServiceProperty”. Allowed value for a “qualitativeProductOrServiceProperty” is “gr:QualitativeValue”, which is a predefined value for a product characteristic. For example, light bulbs have the qualitative property “hasEnergyEfficiency”, with predefined values from “A” to “G”.
- “gr:datatypeProductOrServiceProperty” is the superproperty for all pure datatype properties that can be used to describe a product and services instance. Only such properties that are not quantitative and that have no predefined value

instances are subproperties of this property. In practice, this refers to a few integer properties for which the integer value represents qualitative aspects, for string datatypes (as long as no predefined values exist), and for boolean datatype properties. For example, "isTouchscreenEnabled" is a "datatypeProductOrServiceProperty" of type boolean (yes/no).

For creating a GoodRelations-compliant ontology, user selects the individuals and properties that will reside in the ontology and clicks the "Use GoodRelations Vocabulary" button (Fig. 8). Using the newly opened wizard, user selects the corresponding GoodRelations property type and respective unit of measurement for each selected property. For the example, we choose "brandName", "hardDrive", "color", and "numberofUSB2Ports" properties and specify the property types and units as in Fig. 11. Note that the selected unit of measurement is saved as an UN/CEFACT Common Code in the ontology and "Symbol" in Fig. 11 means the property does not specified with a unit of measurement.

The last step is saving the ontology and serializing the ontology in one of the different standards for structured data markup. The user clicks the "Save" button (Fig. 8) to save the ontology in an owl file. Then the user clicks the "Export to a serialization format" button (Fig. 8) to view the ontology in one of the structured data markup standards.

4. Evaluation

We evaluate the usability of the tool using three questionnaires: Computer System Usability Questionnaire (CSUQ), System Usability Scale (SUS) and Microsoft's Product Reaction Card (MPRC). We primarily choose CSUQ and SUS for our experiments because these two approaches have a higher accuracy with an increasing sample size than the other questionnaires. These two formal approaches provided results to evaluate whether users can complete tasks. We supported these approaches with MPRC for measuring intangible aspects of the user experience such as "fun', "enjoyment" or whether the product is desirable enough to use.

Fig. 12 shows the results of the study by Tullis and Stetson [30]. They analyze the effectiveness of four standard usability questionnaires: SUS, QUIS, CSUQ, words that a variant of Microsoft's Product Reaction Cards, and one that they have used in their usability lab for several years. As one would expect, the accuracy of the analysis increases as the sample size gets larger. SUS and CSUQ reach asymptotes of 90–100% at a sample size of 12. Words jump up to about 70% accuracy at a sample size of 14. These results also indicate that, for the conditions of the study [30], sample sizes of at least 12–14 participants are needed to get reasonably reliable results. Therefore we have conducted a survey with 15 participants.

Two of the participants were working as software developers for an IT company. The other participants were 4 academics and 9 students (3 undergraduate students, 2 master students and 4 doctorate students) from computer science departments of two different universities. The participants rate their experience levels in HTML, ontology-based systems and Protégé Ontology Development Environment with 1 indicating no experience and 5 being an expert. The results are shown in Table 1, where *N* is the number of responses, AVG is the average value, DEV is the deviation value, MED is the median, MAX is the maximum value, and the MIN is the minimum value. Four of the participants had no experience with ontology based-systems, and six participants had no experience with Protégé Environment. One user had limited knowledge (level 2), and two users had average knowledge (level 3) with HTML. As a result of the study, users having no experience (level 1) with ontology-based systems or Protégé Environment could complete tasks with the tool. On the other hand, basic knowledge of HTML (level 2) is required to build a template.

The usability test was conducted as follows. First the participants received a brief introduction of 35 min of the structure of the evaluation, the basic functionality of the tool and example templates for different scenarios. After this introduction, the participants chose a web site which provides resources for product features. Then they built a template and run it without any further hint. The results for individual participants and the templates created are available on our repository[3]. At the end, each participant had to fill out a Computer System Usability Questionnaire, a System Usability Scale Questionnaire and a Microsoft Reaction Card. They also required answering questions about the level of previous HTML, Protégé and Ontology-based systems experiences and free text questions about most positive and negative aspects. The following subsections present the results of the questionnaires and discussion of these results.

4.1. CSUQ evaluation results

The Computer Usability Questionnaire (CSUQ) [31] contains 19 questions and users rate them from 1 to 7, where 1 is strongly agree and 7 is strongly disagree. Besides, participants asked about the most positive and negative aspects of the system. The three internal subscales of CSUQ (Table 2) are System Usefulness, Information Quality and Interface Quality. The first eight questions on CSUQ assess System Usefulness. These questions refer to the users' perception of the ease of use, learnability, speed of performance and effectiveness in completing tasks, and subjective feeling. Questions 9–15 can be used as a means of assessing the participants' satisfaction with the quality of the information associated with the system. "Information quality" includes the users' beliefs regarding error messages and error handling, information clarity, understandability, and utility more generally. Questions 16–18 provide a score for the interface quality.

[3] Download link: http://dropproxy.com/f/A94

The results of the CSUQ are shown in Table 3, where *N* is the number of responses, AVG is the average value, DEV is the deviation value, MED is the median, MAX is the maximum, and the MIN is the minimum.

Most of the participants appreciated that the system was easy-to-use and has a clear value and purpose. They found it to be a time-saving, fast and useful tool, which contributes to lowering the cost of data entry and improves productivity. It provides clear and reliable results even with complex web sites. They specified that at the beginning the system was intimidating but after analyzing the example templates it was easy to create another one. They felt comfortable about using the system for different scenarios and they did not need any extra knowledge other than the basics of HTML. As a negative aspect most of the participants explicitly mentioned that it is not easy to use the system for those who lack basic HTML knowledge.

The majority of participants enjoyed to use the interface and found it clear and pleasant. But two participants were not satisfied with the user-interface and required for a more user-friendly one. Five of the participants criticized that beyond being integrated as a plug-in to Protégé, the application should also be delivered as a Java library. Three of the participants specified that using a text file for creating the template is error-prone and they concerned about the lack of visual aids. One participant proposed a module to check the syntax of the template.

In general the participants were satisfied with the functions and capabilities of the system. They appreciate the support of different output formats like Microdata and RDFa. They found it flexible to be able to carry out string operations (e.g. concatenate, substring, and trim) on leaf nodes. They verified that the system works with a wide variety of web sites with different technologies. One participant criticized that the system is based on HtmlUnit so it may not be able to handle web sites with complex JavaScript based interfaces. One participant specified the domain-specific language to build a template provided productivity and accessibility. Another participant found the GoodRelations E-commerce Vocabulary support very useful.

The quality of the error messages and the documentation was ranked negative in average. The participants criticized the inadequate, very generic and not self-explaining error messages. They reported that even soft errors cannot be recognized and recovered by the system. Most of the participants did not satisfy with the quality of the user documentation. So we plan to enrich the documentation with the 15 example templates produced by the participants. In addition to the above, the some of the participants required to be informed about the changes in the web site they used. Some participants required different output options like database output and csv output. One participant required a module to perform arithmetical and string operations and error correction on extracted data.

We further calculated the overall score and the three factor scores for System Usefulness, Information Quality, and Interface Quality for all participants as illustrated in Table 4. The overall assessment of the students about the usability of the tool was positive (73.03%). The results show that the system usefulness and interface quality ranked higher in 73% of the responses than the overall feedback. The score for information quality did poorly in 73% of the responses than the overall feedback.

4.2. SUS evaluation results

System Usability Scale (SUS) [32] is one of the popular questionnaires, used for the assessment of usability (Table 5). It is described as "quick and dirty" usability scale. The participant has to rate each question with a 5 point scale, where 5 is strongly agree and 1 is strongly disagree.

The score of positive questions (1, 3, 5, 7, and 9) is the rating minus 1. The score of negative questions (2, 4, 6, 8, and 10) is 5 minus the rating. To obtain the overall score on a scale of 0–100, you add up the score values and multiply the sum by 2.5.

The score values of the different participants are presented in Table 6; the final average SUS score is 79.67. Table 7 provides the SUS scores with their corresponding adjective and acceptability ratings. The results show that the proposed tool has a "good" and "acceptable" level of usability. Likewise, results in Table 6 show that all participants rated usability above 65.

4.3. Microsoft Product Reaction Card evaluation results

The participants also rated their satisfaction about the tool by selecting minimum five words from the Microsoft Product Reaction Card. Microsoft Reaction Card method [35] is developed by to check the emotional response and desirability of a design or product. The participant is asked to describe a design/product using any number of the 118 words (Table 8) with 60% positive and 40% negative or neutral words.

In this study, 28 different words were chosen by the 15 participants (Fig. 13). Five of those words were negative and the remaining 23 words were positive. On the positive side, 12 out of 15 participants (80%) chose the word "time-saving" to describe their experience on the tool. They found it to be a "useful" (53%), "creative" (40%), "effective" (40%), "efficient" (27%) and "usable" (27%) tool, which contributes to lowering the cost of data entry and improves productivity. Most of the participants appreciated that the system was "easy-to-use" (67%) and "valuable" (20%). They specified that the results are "clear" (33%) and "reliable" (67%). Two business users and an academic specified that it is a "business-like tool" (20%). The words "connected", "high quality", "novel" and "satisfying" selected twice each (13%). Two words regarding the user interface, "appealing" and "attractive", were selected only once (7%). The other positive words selected only once (7%) were "collaborative", "fast", "flexible", "innovative", "professional" and "sophisticated".

On the negative side, the words "intimidating" and "too technical" were selected four times each (27%). They specified that at the beginning the system was intimidating but after analyzing the example templates it was easy to create another one. As another negative aspect most of the participants explicitly mentioned that the tool may be "too technical" (27%) and "advanced" (13%) for users who lacks basic HTML knowledge. Regarding the ease of use, the negative word “rigid” is selected two times (13%). Concerning the insufficient error messages and poor error handling, most users selected the word “fragile” (67%).

5. Conclusion and future work

This paper introduces a Protégé plug-in that collects product features from web and transforms this information into GoodRelations snippets in RDFa or Microformats. The system attempts to solve an increasingly important problem: extracting useful information from the product descriptions provided by the sellers and structuring this information into a common and sharable format among business entities, software agents and search engines. It also presents a user-based evaluation accomplished using 15 different “real world” scenarios. During this evaluation work the tool is used in real business settings. A software company decided to use the tool in order to get the product features of clothing products. Another company began preparations for using the tool to monitor the price changes of products.

This work has a number of ongoing and future research directions. An ongoing work is integration with Google Semantics3's Products API [36], which is a direct replacement for Google's Shopping API. Semantics3 aims to organize the world's e-commerce data. It supports all of Google's Shopping API features and more, including additional features such as UPC Lookups, Merchant Product URL Lookups, Merchant Product Catalog Retrieval, Product Variation Lookups, Historical Pricing and Merchant Level Filtering. Semantics3 Product API gives developers comprehensive access to data across millions of products and prices. The primary use of the API is to retrieve detailed and structured data about a specific product. The queries can be performed against up to 50 different fields including “brand”, “name”’ and “price”. The data in Semantics3s database is being constantly refreshed so all information is current and up to date. Developers can even make millions of queries a day. The API is based on REST principles and request content is provided in JSON. We plan to improve the tool with an extension that gets user queries and sends them to Semantics3 API in order to build a GoodRelations compliant product ontology from the returned JSON responses.

Another potential future work is generating an environment for semi-automatic template construction. An environment that automatically constructs the tree nodes from the selected HTML parts will significantly reduce the time to build a template file. Beyond being integrated as a plug-in to Protégé, we also plan to deliver the application as a Java library. And yet another future work is diversify the supported input/output formats. Currently, the system accepts only HTML pages, but the choices can be increased with pdf and structured tabular formats (e.g. excel and csv). The output formats can also be extended with different options like database output and csv output.

Appendix A

```
SELECT=(li), ATTR=(id), VALUE= (result_***)
<
SELECT=(h2), ATTR=(class), VALUE=(a-size-base s-inline s-access-title a-text-normal),
GETMETHOD=(asText), AS=(product.title);
SELECT=(img), ATTR=(src), GETMETHOD=(src), AS=(product.imgLink);
SELECT=(a), ATTR=(class), VALUE=(s-access-detail-page***), GETMETHOD=(href)
<
SELECT=(td), ATTR=(class), VALUE=(label), GETMETHOD=(asText), AS=(product.propertyName);
SELECT=(td), ATTR=(class), VALUE=(value), GETMETHOD=(asText), AS=(product.propertyValue)
>
>
NEXT_PAGE:{SELECT=(a) , ATTR=(id) , VALUE=(pagnNextLink), GETMETHOD=(href)}
PAGE_RANGE:{1-2}
BASE_URI:{http://www.amazon.com}
PAGE_URI:{http://www.amazon.com/s/ref=sr_nr_n_1?rh=n%3A565108%2Ck%3Alaptop
          &keywords=laptop&ie=UTF8&qid=1374832151&rnid=2941120011}
CLASS:{Laptop}
```

Appendix B

```
SELECT=(li), ATTR=(id), VALUE=(search-result***)
<
SELECT=(div), ATTR=(class), VALUE= (image-block small)
<
SELECT(a), ATTR=(alt), GETMETHOD=(alt), AS=(product.title);
SELECT=(a), ATTR=(href), GETMETHOD=(href)
<
SELECT=(div), ATTR=(class), VALUE= (product-details box)
<
SELECT=(li), GETMETHOD=(asText), AS=({product.propertyName}{product.propertyValue}),
OPERATORS= ({split}{{U+003A}{0}{0}}{split}{{U+003A}{1}{1}})
>
>
>
>
NEXT_PAGE:{SELECT=(a) , ATTR=(title) , VALUE=(Next), GETMETHOD=(href)}
PAGE_RANGE:{1-3}
BASE_URI:{http://www.barnesandnoble.com}
PAGE_URI:{http://www.barnesandnoble.com/s?store=KIDS&SRT=sa&PUB=a&aref=1550}
CLASS:{Book}
```

Appendix C

charAt(int **index):** gets the char value at the specified index.
concat(String **str):** concatenates the specified string to the end of the value.
replace(hex char **oldChar,** hex char **newChar):** returns a new value resulting from replacing all occurrences of oldChar in the value with newChar.
replaceAll(String **regex,** String **replacement):** replaces each substring of the value that matches the regular expression with the given replacement.
replaceFirst(String **regex,** String **replacement):** replaces the first substring of the value that matches the given regular expression with the given replacement.
split(hex char **delimeter):** splits the value around the matches of the given delimiter.
substring(int **beginIndex):** returns a new value that begins at the specified index.
substring(int **beginIndex,** int **endIndex):** returns a new value that begins at the specified begin index and extends to the character at endIndex-1.
toLowerCase(): converts all of the characters in the value to lower case.
toUpperCase(): converts all of the characters in the value to upper case.
trim(): returns a copy of the string with leading and trailing whitespace omitted.
b(): returns the value between <b> tags.
exceptB(): returns the value except the text between <b> tags.
i(): returns the value between <i> tags.
exceptI(): returns the value except the text between <i> tags.
em(): returns the value between <em> tags.
exceptEm(): returns the value except the text between <em> tags.
br(): returns the value between <br> tags.
exceptBr(): returns the value except the text between <br> tags.
small(): returns the value between <small> tags.
exceptSmall(): returns the value except the text between <small> tags.
mark(): returns the value between <mark> tags.
exceptMark(): returns the value except the text between <mark> tags.
del(): returns the value between <del> tags.
exceptDel(): returns the value except the text between <del> tags.
ins(): returns the value between <ins> tags.
exceptIns(): returns the value except the text between <ins> tags.
sub(): returns the value between <sub> tags.
exceptSub(): returns the value except the text between <sub> tags.
sup(): returns the value between <sup> tags.
exceptSup(): returns the value except the text between <sup> tags.
strong(): returns the value between <strong> tags.
exceptStrong(): returns the value except the text between <strong> tags.

Appendix D

```
<tree> := <node> (“<” <children> “>”)
<children> := <tree> |(<node>“;”)
<node> := <node_part1>(<node_part2> | empty)
<node_part1>:= “SELECT=” “(” <select_x> “)”
<node_part2>:= (“ATTR=” “(” <attr_x>“)”) (“VALUE=” “(” <terminal>“)”)
                 (“GETMETHOD=” “(” < getmethod_x>“)”)
                 (“AS=” “(” <as_x>“)”) (“OPERATORS=” “(” <operands_x>“)”)
<select_x> := “!–...-” | “!DOCTYPE” | “a” |“abbr” | “acronym”
           | “address” | “applet” | “area” | “article” | “aside”
           | “audio” | “b”| “base” | “basefont” | “bdi” | “bdo”
           | “big” | “blockquote” | “body”| “br” | “canvas”
           | “caption” | “center” | “cite” | “code” | “col”
           | “colgroup” | “command” | “datalist” | “dd” | “del”
           | “details” | “dfn”| “dialog” | “dir” | “div” | “dl”
           | “dt” | “em” | “embed” | “fieldset”| “figcaption”
           | “figure” | “font” | “footer” | “form” | “frame”
           | “frameset” | “h1” | “h2” | “h3” | “h4” | “h5” | “h6”
           | “head” | “header”| “hgroup” | “hr” | “html” | “i”
           | “iframe” | “img” | “input” | “ins” | “kbd”| “keygen”
           | “label” | “legend” | “li” | “link” | “map” | “mark”
           | “menu”| “meta” | “meter” | “nav” | “noframes”
           | “noscript” | “object” | “ol”| “optgroup” | “option”
           | “output” | “p” | “param” | “pre” | “progress” | “q”
           | “rp” | “rt” | “ruby” | “s” | “samp” | “script”
           | “section” | “select”| “small” | “source” | “span”
           | “strike” | “strong” | “style” | “sub”| “summary”
           | “sup” | “table” | “tbody” | “td” | “textarea” | “tfoot”
           | “th”| “thead” | “time” | “title” | “tr” | “track” | “tt”
           | “u” | “ul” | “var”| “video” | “wbr”
<attr_x>:= “accept” | “accept-charset” | “accesskey” | “action” | “align”
           | “alt” | “async” | “autocomplete” | “autofocus” | “autoplay”
           | “bgcolor” | “border” | “buffered” | “challenge” | “charset”
           | “checked” | “cite” | “class” | “code” | “codebase”
           | “color”| “cols” | “colspan” | “content” | “contenteditable”
           | “contextmenu” | “controls” | “cords” | “data” | “data-custom”
           | “datetime” | “default” | “defer” | “dir” | “dirname”
           | “disabled”| “download” | “draggable” | “dropzone” | “enctype”
           | “for” | “form”| “headers” | “height” | “hidden” | “high”
           | “href” | “hreflang”| “http-equiv” | “icon” | “id” | “ismap”
           | “itemprop” | “keytype”| “kind” | “label” | “lang”
           | “language” | “list” | “loop” | “low”| “manifest” | “max”
           | “maxlength” | “media” | “method” | “min”| “multiple” | “name”
           | “novalidate” | “open” | “optimum” | “pattern” | “ping”
           | “placeholder” | “poster” | “preload” | “pubdate” | “radiogroup”
           | “readonly” | “rel” | “required” | “revered” | “rows”
           | “rowspan”| “sandbox” | “spellcheck” | “scope” | “scoped”
           | “seamless” | “selected”| “shape” | “size” | “sizes” | “span”
           | “src” | “srcdoc” | “srclang”| “start” | “step” | “style”
           | “summary” | “tabindex” | “target” | “title”| “type” | “usemap”
           | “value” | “width” | “wrap”
<getmethod_x> := “asText” | <attr_x>
<as_x> := <as_term> | (“{”<as_term>“}”“{”<as_term>“}”{“{”<as_term>“}”})
<as_term> := “product.title” | “product.brand” | “product.productID”
           | “product. description”| “product.imgLink” | “product.feature”
           | “product.propertyName”| “product.propertyValue” | “product.includes”
<operands_x> := {“{”<function_name>“}”{“{”(terminal)“}”}“{”<number>“}”}
<function_name>:= “charAt” | “concat” | “replace” | “replaceAll” | “replaceFirst”
           | “split” | “substring” | “toLowerCase” | “toUpperCase” | “trim”
           | “b” | “i” | “em” | “br” | “small” | “mark” | “del” | “ins”
           | “sub” | “sup” | “strong” | “exceptB” | “exceptI” | “exceptEm”
           | “exceptBr” | “exceptSmall” | “exceptMark” | “exceptDel” | “exceptIns”
           | “exceptSub” | “exceptSup” | “exceptStrong”
<number> := <digit><digit>
<digit> :=“0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”
<symbol> := “[” | “]” | “” | “” | “(” | “)” | “<” | “>”
           | “'” | “"” | “=” | “♣ | “.” | “,” | “;”
<character> := <letter> | <digit> | <symbol> | “_”
<letter> := “A” | “B” | “C” | “D” | “E” | “F” | “G” | “H” | “I”
           | “J” | “K” | “L” | “M” | “N” | “O” | “P” | “Q” | “R”
           | “S” | “T” | “U” | “V” | “W” | “X” | “Y” | “Z”
<terminal> := “'” <character> { <character> } “'”
           | “"” <character> { <character> } “"”
```

References

[1] Q. Liu, H. Wang, H. Gao, Q. Lv, J. Fu, A recommendation method in e-commerce based on product taxonomy graph, in: Proceedings of the 2012 International Conference on Information Technology and Management Science (ICITMS 2012), Springer, Berlin, Heidelberg, 2013, pp. 411–422.
[2] W. Tang, Y. Hong, Y.-H. Feng, J.-M. Yao, Q.-M. Zhu, Simultaneous product attribute name and value extraction with adaptively learnt templates, in: Proceedings of the 2012 International Conference on Computer Science and Service System, CSSS '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 2021–2025.
[3] F. Dominik, Semi-automatic web information extraction (Ph.D. thesis), Poznan University of Economics Department of Information Systems, 2011.
[4] O. Etzioni, The world wide web: quagmire or gold mine? Commun. ACM 39 (1996) 65–68.
[5] T. Özacar, IRIS: a Protégé plug-in to extract and serialize product attribute name-value pairs, in: Proceedings of the 11th Extended Semantic Web Conference 2014, Crete, 2014.
[6] N. Noy, R. Fergerson, M. Musen, The knowledge model of protégé-2000: combining interoperability and flexibility, in: Knowledge Engineering and Knowledge Management Methods, Models, and Tools, Lecture Notes in Computer Science, vol. 1937, Springer, Berlin, Heidelberg, 2000, pp. 17–32.
[7] M. Hepp, Goodrelations: an ontology for describing products and services offers on the web, in: Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns, EKAW '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 329–346.
[8] D. Siegel, Pull: The Power of the Semantic Web to Transform Your Business, 1st edition, Portfolio Hardcover, 2009.
[9] P. O'Dell, Silver Bullets: How Interoperable Data Will Revolutionize Information Sharing and Transparency, AuthorHouse, 2010.
[10] J. Han, Design of web semantic integration system (Ph.D. thesis), Tennessee State University. Electrical & Computer Engineering, 2008.

[11] A. Firat, Information integration using contextual knowledge and ontology merging (Ph.D. thesis), Massachusetts Institute of Technology, Sloan School of Management, 2003.
[12] The xpath 2.0 Standard, W3c Recommendation ⟨http://www.network-theory.co.uk/w3c/xpath/⟩, 2007.
[13] I. Muslea, S. Minton, C. Knoblock, A Hierarchical Approach to Wrapper Induction, ACM Press, New York, NY, USA, 1999, 190–197.
[14] C.H. Chang, M. Kayed, M.R. Girgis, K.F. Shaalan, A survey of web information extraction systems, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1411–1428.
[15] C. Leondes, Neural networks, fuzzy theory and genetic algorithms, Intelligent Knowledge-Based Systems: Business and Technology in the New Millennium, vol. 5, Kluwer Academic, 2005.
[16] G. Huck, P. Fankhauser, K. Aberer, E.J. Neuhold, Jedi: extracting and synthesizing information from the web., in: Proceedings of the Third IFCIS International Conference on Cooperative Information Systems, IEEE Computer Society, New York, NY, USA, 1998, pp. 32–43.
[17] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom, The TSIMMIS approach to mediation: data models and languages, J. Intell. Inf. Syst. 8 (2) (2004) 117–132.
[18] A. Sahuguet, F. Azavant, Wysiwyg web wrapper factory (w4f), in: Proceedings of the Eighth International WWW Conference, 1999.
[19] N. Kushmerick, D.S. Weld, R. Doorenbos, Wrapper induction for information extraction, in: Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997.
[20] C. N. Hsu, M. T. Dung, Wrapping semistructured web pages with finite-state transducers, in: Proceedings of the Workshop on Learning from Text and the Web, Conference on Automated Learning and Discovery, 1998.
[21] I. Muslea, S. Minton, C. Knoblock, Stalker: Learning extraction rules for semistructured, in: Proceedings of the 15th Workshop on AI and Information Integration, 1998.
[22] L. Liu, C. Pu, W. Han, Xwrap: an xml-enabled wrapper construction system for web information sources, in: Proceedings of the 16th International Conference on Data Engineering, 2000, pp. 611–621.
[23] R. Baumgartner, S. Flesca, G. Gottlob, Visual web information extraction with lixto, in: Proceedings of 27th International Conference on Very Large Data Bases, 2001, pp. 119–128.
[24] B. Wu, X. Cheng, Y. Wang, Y. Guo, L. Song, Simultaneous product attribute name and value extraction from web pages, in: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol. 03, IEEE Computer Society, Washington, DC, USA, 2009, pp. 295–298.
[25] W. Holzinger, B. Krüpl, M. Herzog, Using ontologies for extracting product features from web pages, in: International Semantic Web Conference, Lecture Notes in Computer Science, vol. 4273, Springer, Athens, GA, USA, 2006, pp. 286–299.
[26] T.R. Gruber, A translation approach to portable ontology specifications, Knowl. Acquis. 5 (2) (1993) 199–220.
[27] Protégé Plug-in Library ⟨http://protegewiki.stanford.edu/wiki/Protege_Plugin_Library⟩ (last accessed: 2015-02-25).
[28] Htmlunit ⟨http://htmlunit.sourceforge.net/⟩.
[29] B.R.C. Alex Stolz, M. Hepp, RDF Translator: A RESTful Multi-Format Data Converter for the Semantic Web, Technical Report TR-2013-1, E-Business and Web Science Research Group 2013.
[30] T. S. Tullis, J. N. Stetson, A comparison of questionnaires for assessing website usability., in: Proceedings of UPA 2004, 2004.
[31] J. R. Lewis, Psychometric Evaluation of the Computer System Usability Questionnaire: The Csuq, Technical Report 54.723, 1992.
[32] J. Brooke, Sus: a “quick and dirty” usability scale, in: P.W. Jordan, B. Thomas, B.A. Weerdmeester, I.L. McClelland (Eds.), Usability Evaluation in Industry, Taylor and Francis, London, 1996.
[33] J.T. Miller, A. Bangor, P.T. Kortum, Determining what individual sus scores mean: adding an adjective rating scale, J. Usability Stud. 4 (3) (2009) 114–123.
[34] A. Bangor, P.T. Kortum, J.T. Miller, An empirical evaluation of the system usability scale, Int. J. Hum. Comput. Interact. 24 (6) (2008) 574–594.
[35] J. Benedek, T. Miner, Measuring desirability: new methods for evaluating desirability in a usability lab setting, in: Proceedings of UPA 2002 Conference ⟨http://www.microsoft.com/usability/UEPostings/DesirabilityToolkit.doc⟩, 2002.
[36] Semantics3 Inc., Semantics3—Apis for Products and Prices ⟨https://www.semantics3.com/googleshoppingapi/quickstart⟩, 2013.