International Journal on Artificial Intelligence Tools
Vol. 21, No. 6 (2012) 1250032 (20 pages)

DOI: 10.1142/S0218213012500327

World Scientific
www.worldscientific.com

FIRST-ORDER LOGIC RULE INDUCTION FOR INFORMATION EXTRACTION IN WEB RESOURCES

JOSÉ IGNACIO FERNÁNDEZ-VILLAMOR, CARLOS ÁNGEL IGLESIAS
and MERCEDES GARIJO

Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid
Avenida Complutense, 30 Madrid, 28040, Spain
{jifv,cif,mga}@dit.upm.es

Received 26 July 2011
Accepted 7 May 2012

Information extraction out of web pages, commonly known as screen scraping, is usually performed through wrapper induction, a technique that is based on the internal structure of HTML documents. As such, the main limitation of these kinds of techniques is that a generated wrapper is only useful for the web page it was designed for. To overcome this, in this paper it is proposed a system that generates first-order logic rules that can be used to extract data from web pages. These rules are based on visual features such as font size, elements positioning or types of contents. Thus, they do not depend on a document's internal structure, and are able to work on different sites. The system has been validated on a set of different web pages, showing very high precision and good recall, which validates the robustness and the generalization capabilities of the approach.

Keywords: Information extraction; first order logic; machine learning; semantic web.

1. Introduction

The vast amount of information available on the Web turns it into an important knowledge source for many different domains. Semantic Web standards[1] and the Linked Data initiative[2] propose the annotation of web resources with metadata, which allows the processing of web resources by automated agents. Despite the growth in adoption of standards of this kind, many web sites still do not provide means to retrieve their contents according to a known, structured schema. For example, out of 17 popular electronic newspapers surveyed,[a] none of them provide semantic annotations of a Semantic Web standard.

Examples of applications that make use of web data can be travelling mashups, which scan web pages for flights, hotels and trains, and provide the best trip plan

[a]The surveyed papers were New York Times, Wall Street Journal, The Guardian, The Telegraph, Spiegel, Bild, Frankfurter Allgemeine Zeitung, Le Monde, L'Équipe, ABC, El Mundo, El País, ADN, 20 Minutos, Público, Marca, and As.

according to a user's preferences. Those flight, hotel and train web sites that adopted the Linked Data initiative would publish metadata that allows a simple extraction of these sites' data. However, in order to get data from other sites that do not publish appropriate metadata, it would be neccesary to use Screen Scraping techniques to get access to data that is published in an unstructured way.[3,4]

Traditional scraping approaches are based on some kind of Document Object Model (DOM)[b] tree processing. Usually, techniques such as tree-to-tree edit distance[5–7] and wrapper induction[8,9] are used to, either manually[10] or automatically,[11] build wrappers that allow extracting data from web resources. The main limitation of DOM tree processing is that these wrappers are specific to one web site, and therefore do not show generalization capabilities for extracting data from other visually similar web sites. Wrappers also require being rebuilt, as part of a maintenance process, when a web resource layout changes.[12] Alternatively, other approaches consider processing visual properties of DOM elements when rendered by a web browser.[13,14] The advantage of these kinds of approaches is its generalization across different sites.

In this paper we describe a system that performs extraction of Linked Data out of web resources and which shows high generalization capabilities and robustness. Semantic information in a web resource is a graph that, following Semantic Web's standards, can be represented using the Resource Description Framework (RDF).[15] Therefore, extracting RDF data implies building the associated graph out of the information present in the web page. We propose using first-order logic rules to extract RDF graphs. To build these rules, we have built an algorithm which follows a specific-to-general basis. First, the information to be extracted is manually identified in web pages and with these samples a set of overfitting rules are built. Then, the algorithm combines and generalizes rules progressively. This supervised first-order logic classifier makes use of web elements' visual properties. Therefore, the knowledge acquired by the classifier generalizes across web sites and is robust to layout changes on them.

The paper is organized as follows. Section 2 describes the problem of extracting Linked Data out of web resources. In section 3, our approach based on visual features and first-order logic rules is proposed. Section 4 describes the evaluation process that was used to validate the approach. Related works are summarized in section 5 and, finally, some conclusions and future works are gathered in section 6.

2. Problem Statement

The Web is a hypermedia system that follows the Representational Stateless Transfer (REST) architectural style.[16] When a client accesses a web resource on a server, the server returns a representation of the resource. Usually, these representations are formatted in HyperText Markup Language (HTML), a language

[b]http://www.w3.org/TR/DOM-Level-2-Core/

that allows defining the structure of a document for its rendering on a web browser. HTML documents are structured as a DOM tree, which defines the logical structure of the HTML document (see Fig. 1) that will be used for rendering the representation on a web browser. In order to have information about the resource's content and not about its rendering structure, Linked Data proposes using resources' representations that include metadata, by enhancing HTML with semantic annotations or by providing RDF representations, such as in Fig. 1. Such figure shows the RDF representation of a piece of news by using Semantically-Interlinked Online Communities Project (SIOC) ontology[17] and Dublin Core (DC) schema,[18] ontologies chosen due to their high adoption and popularity among the Semantic Web community.

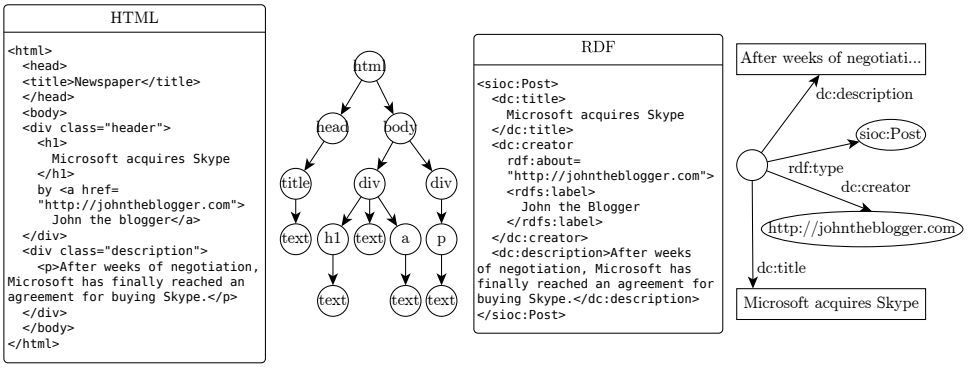

Fig. 1. HTML vs RDF documents.

Whenever a resource provides unannotated HTML, a technique that processes the DOM tree in some way needs to be used to identify the structure of the data present in the HTML document and build the associated RDF graph. This process is known as Screen Scraping, and it implies solving the following problems:

- Identifying what pieces of information are relevant in a web resource. Usually, in a web resource there are DOM fragments that do not provide information, such as advertisements, headers, footers, or decorative elements, while other fragments such as posts or comments have valuable information.
- Identifying what relations are stated in a web fragment. For instance, a heading in a piece of news might represent the news title.

The conceptual model behind the process of building an RDF graph out of an HTML page is shown in Fig. 2 and serves as a basis for addressing the problem of web resource screen scraping. It shows the elements involved in the process of Screen Scraping in order to familiarize the reader with the process. The figure shows the relations among these elements and how a scraper requires different pieces of information to complete the process of converting a web page into a set of RDF

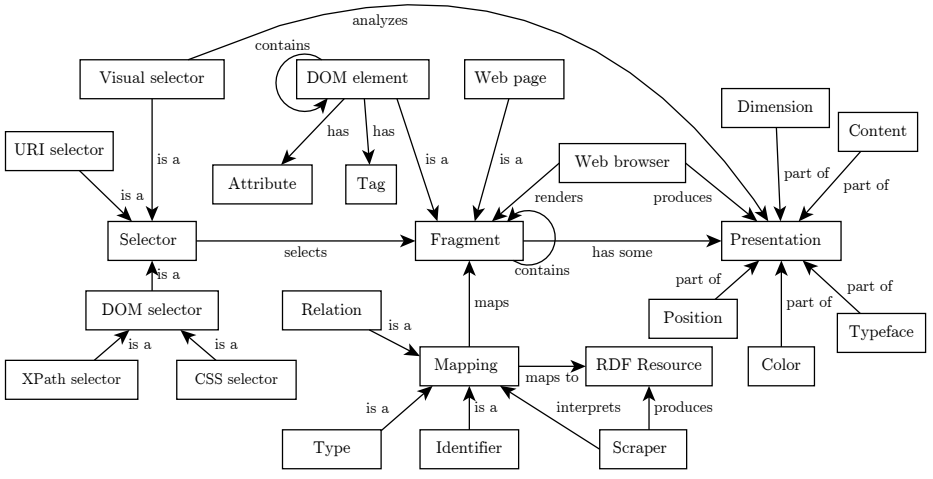

Fig. 2. Scraping conceptual model.

resources. As shown, the main elements involved in the problem of Screen Scraping are:

Web page. The HTML representation that is returned by a web browser when attempting to retrieve a web resource. Web pages are designed to be used by human users through a web browser.

Fragment. Any web fragment inside a web page, or a web page itself. A web page fragment usually shows information about one or more concepts, such as a blog post, a flight, a web result, etc.

Selector. Any mean to identify a fragment inside a document. Usually, web scrapers use regular expressions or Content Style Sheets (CSS) or XPath selectors to achieve these tasks.

DOM element. Each of the elements in the DOM tree of an HTML document. They represent the hierarchical structure of the document, and can be referenced through the usage of DOM selectors.

Presentation. The output of a web browser when rendering a web fragment, which consists of a set of properties such as typeface, color or dimensions. This output is used by users to interpret the contents of a web page, and can be used by visual selectors as well to identify web fragments according to their visual properties.

Mapping. The mapping that exists between a fragment inside an HTML document and the RDF resource it represents. A mapping might consist of stating a predicate

about a resource or that a resource has a particular Uniform Resource Identifier (URI).

Scraper. An automated process that is able to interpret mappings to produce RDF data. An RDF document that defines the extraction mappings on web fragments can be used by the appropriate scraper to extract the data from a web resource in RDF format.

Several challenges are involved behind the problem of Screen Scraping. The main difficulties are listed next:

- Identification of data to extract. First, the desired data to be extracted needs to be defined. Either tools for manual annotation or automated approaches that compare similar pages or analyse documents' structure are used for this task.
- Definition of selectors. After the data to extract have been identified, appropriate selectors need to be constructed. Either regular expressions, wrappers, CSS, XPath, or visual selectors can be used. The type and quality of the defined selector will affect its applicability to other sites.
- Changes in web pages. Whenever a web page's layout changes, the defined selectors can be not valid anymore. The consequences are usually badly extracted data or no extracted data at all. Quality checks and better selectors help to prevent this from happening.
- Dynamic, JavaScript-intensive web pages. Some web sites change the layout after page load or after interaction with the user. A solution consists of executing JavaScript and reproducing the interactions with mocked-up users to access those data, although overcoming this problem is out of the scope of this paper.

Scraping mappings contain the selectors that identify data and their generalization capabalities. Therefore, when employing mappings to tackle the problem of Screen Scraping, the main problem is defining quality mappings that allow a scraper to extract the data from a web resource. Then, we identify two desireable aspects on the definition of a mapping, which are obtaining *robustness* and *generalization*:

- A *robust* mapping is one that extracts the same data even with changes in the DOM tree of the web resource. If a mapping is not robust, it might stop working once the layout of a web site is changed by its web administrator on a redesign stage.[12]
- A mapping that *generalizes* is one that is valid for all the web resources that contain the same kind of data. If a mapping is only valid for the web resource (or resources) that it was defined for, it does not generalize across different resources. The main limitation of wrapper induction is that wrappers are only valid for the web pages they were designed for Ref. 8.

Therefore, the problem that is addressed in this paper is building robust and generalizable extraction mappings that allow scraping web resources. Metrics to measure robustness and generalization will be given in section 4.1.

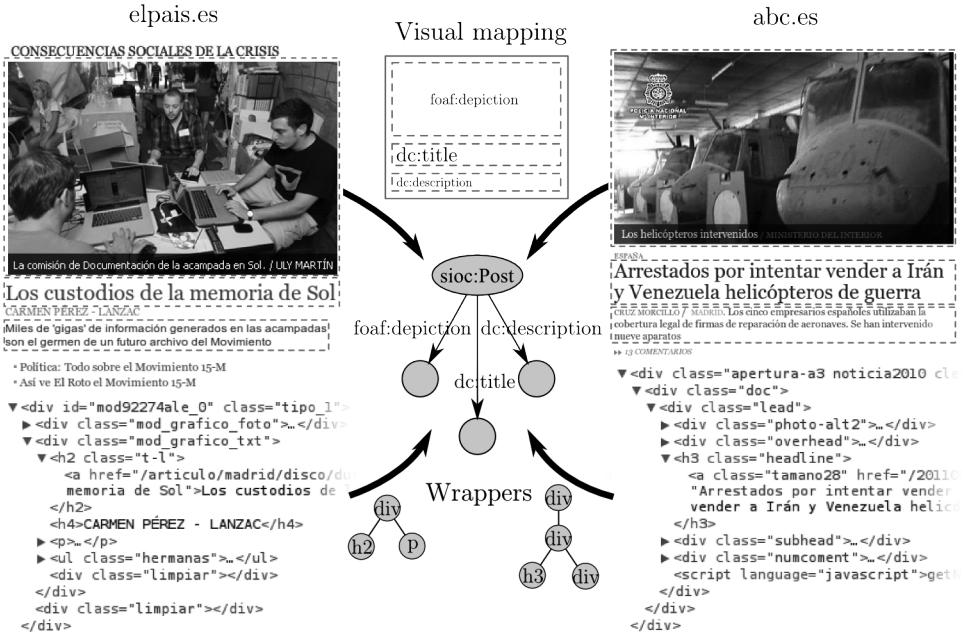

Fig. 3. Conversion of a DOM tree into an RDF graph.

3. First-Order Logic Rules for Data Extraction

In this section, we describe an approach to automatically create mappings for extracting RDF data from HTML documents while attempting to solve the issues described above. The approach uses visual features of the elements displayed in a web browser in a way as shown in Fig. 3, which compares the approach of using wrappers against visual selectors. The figure shows two Spanish news sites (Abc and El País), which have different layouts but similar looks. When using the techniques behind wrapper induction, it is required to define a new wrapper for each different web site, as they are based on the DOM tree structure of a web site, which is rarely shared among different web sites. Using techniques that are based on visual features allows mappings to generalize accross different web resources, as these web resources share similar looks.

The Scraping Ontology[c] is an RDF schema that allows defining mappings between HTML elements and the RDF data the mappings represent,[19] and is used in this paper to represent the RDF mappings. This ontology contains the terms that were defined in the conceptual model.

The mappings defined in this ontology are sequences of fragments with the RDF data that they represent. An example of the mappings that are considered by

[c]http://lab.gsi.dit.upm.es/scraping.rdf

the algorithm is shown next:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:sc="http://lab.gsi.dit.upm.es/scraping.rdf#"
   xmlns:sioc="http://rdfs.org/sioc/ns#">
  <sc:Fragment>
    <sc:type rdf:resource="http://rdfs.org/sioc/ns#Post"/>
    <sc:selector>
      <sc:VisualSelector>
        <sc:max_height>139</sc:max_height>
        <sc:max_relative_x>508</sc:max_relative_x>
        <sc:max_relative_y>1084</sc:max_relative_y>
        ...
      </sc:VisualSelector>
    </sc:selector>
    <sc:subfragment>
      <sc:Fragment>
        <sc:max_cardinality>1</sc:max_cardinality>
        <sc:min_cardinality>1</sc:min_cardinality>
        <sc:type
           rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
        <sc:relation
           rdf:resource="http://purl.org/dc/elements/1.1/title"/>
        <sc:selector>
          <sc:VisualSelector>
            <sc:font_family>serif</sc:font_family>
            <sc:max_font_size>24</sc:max_font_size>
            ...
          </sc:VisualSelector>
        </sc:selector>
      </sc:Fragment>
    </sc:subfragment>
    ...
  </sc:Fragment>
</rdf:RDF>
```

As it can be seen, not only is the output of extraction mappings expressed in RDF, but also the mappings themselves, as a result of using the Scraping Ontology. In the previous RDF document, a fragment that represents a news *post* (according to SIOC ontology) is defined. The fragment is selected out of a web resource thanks to a visual selector, for which some visual conditions are defined. Additionally, this news post has other subfragment, which is also selected thanks to another visual selector. This subfragment is related to the parent through a *title* relation (according to DC schema).

These mappings can be represented as rules that are applied to web resources. If the rule succeeds, data is extracted from the web page. Two examples of rules are the following ones:

$$width(x) > 200 \wedge width(x) < 300 \wedge font\_size(x) < 14 \Rightarrow rdf{:}type(x, sioc{:}Post) \tag{1}$$

$$width(x) > 200 \wedge width(x) < 300 \wedge font\_size(x) < 14 \wedge parent(x, y) \wedge font\_size(y) > 16 \wedge font\_weight(y) > 400 \Rightarrow rdf{:}type(x, sioc{:}Post) \wedge dc{:}title(x, y) \tag{2}$$

Equation (1) shows a rule which states that an HTML fragment x represents a blog post if some font and size conditions are evaluated as true. The rule shown in Eq. (2) is a more complex one, and makes a statement about a post by considering it also has a subfragment that represents a title. If the conditions are evaluated as true, then RDF triples (i.e. data structures that consist of a subject, a predicate and an object) are built and thus a post with a title is generated as output. The rules make use of the already mentioned SIOC and DC ontologies to model the extracted data.

As seen, rules have a left-hand side with conditions about the visual features of HTML fragments (e.g., $width(x) > 200$ or $font\_size(x) < 14$), as well as structural conditions about how these HTML fragments are organized in the DOM tree (e.g., $parent(x, y)$). The right-hand side contains RDF statements about the HTML fragments involved in the left-hand side. More formally, the rules have the following structure:

$$\bigwedge_i c_i(x_i \in \mathcal{X}) \wedge \bigwedge_{i,j} parent(x_i \in \mathcal{X}, x_j \in \mathcal{X}) \Rightarrow \bigwedge_i T_i(x_1, ..., x_N) \tag{3}$$

where:

- $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is the set of HTML fragments involved in the rule.
- $c_i(x_j)$ is a condition on attributes of the HTML fragment x_j (e.g. $width(x_3) < 300$).
- $parent(x_i, x_j)$ is a binary predicate that states that x_i has to be a parent of x_j in the DOM tree (i.e. HTML fragment x_j is contained in fragment x_i).
- $T_i(x_1, \ldots, x_N)$ is an statement about the RDF resource represented by $x_i \in \mathcal{X}$ which can involve one or more of the variables x_i that are used in the left-hand side (e.g. $rdf{:}type(x_2, sioc{:}Post)$ or $dc{:}title(x_2, x_3)$).

3.1. *Training attributes and classes*

As said before, robustness and generalization are aspects that are desireable to have in extraction mappings. An RDF mapping can be defined using different selectors. Selectors such as CSS or XPath might result in extraction mappings that can only be applied on a reduced set of web resources. The usage of visual selectors allows extraction mappings to work on different web sites. The algorithm makes use of several visual features of the DOM tree elements present in a web resource, as listed next:

- Continuous attributes:
  - Positioning (X and Y).
  - Width and height.
  - Font size and weight.
- Discrete attributes:
  - Font family: sans, sans-serif or monospace.
  - HTML tag: link, image or other.

The values of these attributes are captured by using a web browser, which renders web pages according to CSSs and other files, such as images.

The classes of the samples can be any RDF triple. As have been shown previously, the rule examples used RDF properties such as *rdf*:*type*, *sioc*:*Post* or *dc*:*title*, but any other kind of RDF triple could be employed.

3.2. *Induction algorithm*

The algorithm builds a rule set in a specific to general basis, by using overfitting rules that are combined into more general ones. This decision is taken in order to reduce the search space; in top-down induction of logical rules[20] all the possible combination of conditions need to be explored, while in rule combination approaches, such as ours, the search space is reduced to the possible combinations among similar rules.[21]

The algorithm requires a supervised dataset as input. There are many techniques that could be used to obtain such supervised database by performing an extraction from a set of web resources. It can be done by using a manually defined wrapper, so the typical techniques for wrapper induction can be used for this purpose. Once this is done, a training dataset is obtained and used as input for the induction algorithm.

Then, overfitting rules are built out of the results of the supervised extraction. An example of the overfitting rules is given with after the following training sample, classified as a *sioc*:*Post*:

$$\begin{aligned} width(x) &= 100 \\ height(x) &= 200 \\ font\_size(x) &= 12 \\ font\_type(x) &= sans \\ rdf{:}type(x, sioc{:}Post) & \end{aligned} \tag{4}$$

As it can bee observed, the sample represents an HTML fragment x which has some visual properties about size and font. This HTML fragment x is converted into the following overfitting rule:

$$\begin{gathered} width(x) \geq 100 \wedge width(x) \leq 100 \wedge \\ height(x) \geq 200 \wedge height(x) \leq 200 \wedge \\ font\_size(x) \geq 12 \wedge font\_size(x) \leq 12 \wedge \\ font\_type(x) = sans \\ \Rightarrow rdf{:}type(x, sioc{:}Post) \end{gathered} \tag{5}$$

Afterwards, the set of overfitting rules is iteratively reduced by grouping similar rules into more general ones, and by simplifying rules by generalizing conditions. A rule r^* is considered more general than rule r according to the following definition:

$$more\_general(r^*, r) \iff lhs(r)(x_1, \ldots, x_n) \Rightarrow lhs(r^*)(x_1, \ldots, x_n)) \tag{6}$$

We use $rhs(r)$ to denote the right-hand side of a rule r and $lhs(r)$ for the left-hand side.

Algorithm 1 Rule generalization algorithm

```
1: procedure GENERALIZE(R, D)
2:     T ← ∅
3:     R* ← R
4:     repeat
5:         if SCORE(R*, D) ≥ SCORE(R, D) then
6:             R ← R*
7:         end if
8:         R* ← R
9:         for (r1, r2) ∈ R × R do
10:            if RHS(r1) = RHS(r2) ∧ ¬((r1, r2) ∈ T) then
11:                T ← (r1, r2)
12:                P ← {θi ∈ LHS(r1), θi(x, y) = parent(x, y)}
13:                lhs* ← ⋀θi∈P θi
14:                for (c1, c2) ∈ LHS(r1) × LHS(r2) do
15:                    if (c1 = (a(x) > th1)) ∧ (c2 = (a(x) > th2)) then
16:                        lhs* ← (lhs* ∧ (a(x) > min(th1, th2)))
17:                    else if (c1 = (a(x) < th1)) ∧ (c2 = (a(x) < th2)) then
18:                        lhs* ← (lhs* ∧ (a(x) < max(th1, th2)))
19:                    else if (c1 = (a(x) = v)) ∧ (c2 = (a(x) = v)) then
20:                        lhs* ← (lhs* ∧ (a(x) = v))
21:                    end if
22:                end for
23:                r* ← (lhs* ⇒ rhs(r1))
24:                R* ← R \ {r1, r2} ∪ {r*}
25:                break for
26:            end if
27:        end for
28:    until R* = R
29:    return R
30: end procedure
```

The process of generalization is shown in Algorithm 1, which accepts a ruleset $\mathcal{R}$ and a set of HTML documents $\mathcal{D}$ on which perform the extractions. A generalization operation is considered valid if the resulting ruleset produces a score as high as or higher than with the previous ruleset. Regarding the score function, we have used F-score, which will be defined in section 4.1, although other score function could be employed. The algorithm finishes when no more generalization operations can be performed, returning a new ruleset.

This approach is similar to other machine learning techniques that perform rule pruning to reduce overfitting.[22] Our system thus requires a *building subset* out of the training dataset during the training phase. This building dataset is the one that

is used to build the set of overfitting rules. Then, the whole training dataset is used by the Algorithm 1 to generalize the obtained rules.

As said, Algorithm 1 progressively merges rules into new, more general ones. Lines 10–26 attempt to group two rules into a new one whenever two rules share the same structure. The requirement for this operation is that the two rules share the same right-hand side. Otherwise, the rules are not grouped and no new rule is built. We use $RHS(r)$ to denote the set of terms that appear on $rhs(r)$, and similarly $LHS(r)$ for the left-hand side. If two rules r_1 and r_2 are to be grouped, lines 14–22 state that for each of the terms c_1, from left-hand side of rule r_1, and c_2, from left-hand side of rule r_2, are combined into a more general one so that $c_1(x) \vee c_2(x) \Rightarrow c^*(x)$. This allows to produce a new rule r^* which is more general than the original rules r_1 and r_2. I.e., after combining r_1 and r_2, the following condition meets:

$$lhs(r_1)(x_1, \ldots, x_n) \vee lhs(r_2)(x_1, \ldots, x_n) \Rightarrow lhs(r^*)(x_1, \ldots, x_n) \tag{7}$$

It can be proved that Eq. (7) implies $more\_general(r^*, r_1)$ and $more\_general(r^*, r_2)$.

An example of combining two rules is shown using the next ruleset:

$$\begin{gathered} r_1 = (width(x) > 100 \wedge width(x) < 300 \wedge font\_size(x) < 14 \wedge \\ parent(x, y) \wedge font\_size(y) > 16 \\ \Rightarrow rdf{:}type(x, sioc{:}Post) \wedge dc{:}title(x, y)) \\ \\ r_2 = (width(x) > 100 \wedge width(x) < 400 \wedge font\_size(x) < 14 \\ \Rightarrow rdf{:}type(x, sioc{:}Post)) \\ \\ r_3 = (width(x) > 250 \wedge width(x) < 400 \wedge font\_size(x) < 14 \wedge \\ parent(x, y) \wedge font\_size(y) > 14 \wedge font\_weight(y) > 500 \\ \Rightarrow rdf{:}type(x, sioc{:}Post) \wedge dc{:}title(x, y)) \end{gathered} \tag{8}$$

Only rules r_1 and r_3 can be combined because of sharing their right-hand sides. After combining term by term according to lines 14–22, the resulting rule r^* is:

$$\begin{gathered} r^* = (width(x) > 100 \wedge width(x) < 400 \wedge font\_size(x) < 14 \wedge \\ parent(x, y) \wedge font\_size(y) > 14 \\ \Rightarrow rdf{:}type(x, sioc{:}Post) \wedge dc{:}title(x, y)) \end{gathered} \tag{9}$$

The new rule is more general than the previous ones, as the condition from Eq. (7) is satisfied:

$$\begin{gathered} (width(x) > 100 \wedge width(x) < 300 \wedge font\_size(x) < 14 \wedge \\ parent(x, y) \wedge font\_size(y) > 16) \vee \\ (width(x) > 250 \wedge width(x) < 400 \wedge font\_size(x) < 14 \wedge \\ parent(x, y) \wedge font\_size(y) > 14 \wedge font\_weight(y) > 500) \\ \Rightarrow (width(x) > 100 \wedge width(x) < 400 \wedge font\_size(x) < 14 \wedge \\ parent(x, y) \wedge font\_size(y) > 14) \end{gathered} \tag{10}$$

Afterwards, the algorithm checks the new score with the resulting ruleset. If the score is as high as the previous one, the new ruleset is kept, otherwise the ruleset is rolled back. In all cases, a different pair of rules is tried in the next iteration (in order to achieve this, the already tried rule combinations are stored in set $\mathcal{T}$). The algorithm finishes when no more rules can be combined.

3.3. *Wrapper conversion*

When a scraper processes a visual mapping, it will obtain a set of RDF resources which are mapped to the fragments in a particular web resource. In our system, we will use this intermediate output to induct a wrapper with traditional wrapper induction techniques. This has the advantages explained next.

First, it improves the results of the visual patterns. Wrapper induction techniques require only a few correctly supervised samples to induct a wrapper.[23] When inducting a wrapper, all extracted data in a list can be extracted even if not all the samples are marked for extraction during the supervision process. This lets increase the recall of the visual patterns, as long as all news with a similar DOM tree will be selected whenever the visual extractor produces an acceptable amount of data as output.

Finally, wrappers are more lightweight, as they do not require visual features to be computed. The visual attributes enumerated above require a web browser to render the web resource and load Content Style Sheets and images. By converting the visual patterns into a wrapper, a web browser is not necessary anymore for using the mapping.

4. Evaluation

The algorithm has been implemented as part of Scrappy,[d] an Open Source Semantic scraper that uses the Scraping Ontology for its mappings.

The system has been evaluated on a set of web pages. It has been trained to extract news posts with title, description and image by using Friend of a Friend (FOAF),[24] DC and SIOC ontologies, chosen because of their high adoption and popularity. An example of extracted piece of news in RDF is the following:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/spec/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sioc="http://rdfs.org/sioc/ns#">
  <sioc:Post
    rdf:about="http://abc.es/20110629/internacional/abci-bolivia-coca-201106291637.html">
    <dc:description>La Convencion de Viena considera la hoja de coca un estupefaciente,
      pero su masticado es una practica ancestral de los indigenas del pais sudamericano
    </dc:description>
    <dc:title>Bolivia denunciara la convencion de la ONU que prohibe el masticado de coca
    </dc:title>
```

[d]http://github.com/josei/scrappy

```
    <foaf:depiction
      rdf:resource="http://www.abc.es/Media/201106/29/10557169--229x229.jpg"/>
  </sioc:Post>
</rdf:RDF/>
```

In the evaluation, a set of experiments are run to check the performance of the solution. Section 4.1 details the metrics that will be obtained out of each experiment. Section 4.2 explains what training and testing datasets have been used, and, finally, in section 4.3 the results are presented and discussed.

4.1. *Evaluation metrics*

In order to evaluate our system, a set of metrics will be calculated out of each test. Typically, recall and precision are the most common metrics used in information extraction, and they will be used in our evaluation along with F-score, a combined metric of precision and recall. These metrics are defined next.

Given a single extraction of a set of data out of a web resource, let n^+ be the number of triples that were extracted right, let n be the number of triples that were extracted, and let N be the number of triples that should have been extracted. With these variables, we get the following formulae for *precision* and *recall*:

$$precision = \frac{n^+}{n} \tag{11}$$

$$recall = \frac{n^+}{N} \tag{12}$$

Precision and recall are separate metrics that provide an idea of the performance of an information extraction or retrieval test. Precision indicates the ratio of results that are correct, while recall indicates the ratio of correct results that are extracted.

In order to have a global indicator that combines both metrics, *F-score* metric is defined. We define F-score as the harmonic mean of precision and recall, which lets us write:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{13}$$

As of section 2, it is desirable to measure the robustness and generalization of the system. To do so, we will split the testing data into a robustness test data and a generalization test data, so robustness and generalization can be defined as follows.

We define *robustness* as the mean of F-scores on a set $\mathcal{S}_R$ of DOM-altered web resources r_i:

$$robustness = \frac{\sum_{r_i \in \mathcal{S}_R} F(r_i)}{|\mathcal{S}_R|} \tag{14}$$

A DOM-altered web resource is a web resource taken from the training dataset which has been subject to changes in its DOM tree. This makes traditional wrapper-

based approaches to fail, as they base the extraction on a particular structure of the DOM tree. Examples of variations performed in the DOM tree can be renaming CSS classes, removing parent nodes, or relabelling HTML nodes. These variations affect only the internal structure of the document, while the visual aspect experiences less changes.

We also define *generalization* as the mean of F-scores on a testing set $\mathcal{S}_T$ of web resources r_i which belong to web sites that were not used at training time.

$$generalization = \frac{\sum_{r_i \in \mathcal{S}_T} F(r_i)}{|\mathcal{S}_T|} \tag{15}$$

Because these resources have completely different DOM trees from the resources used in the training phase, wrapper-based techniques cannot be applied, as they would require a new wrapper to be constructed. Our approach is based on visual features, so it will be able to extract data, as for usability reasons web sites often share similar visual aspects.

4.2. *Evaluation datasets*

The datasets used for the evaluation are home pages of three Spanish newspapers: Abc,[e] El País[f] and El Mundo.[g] These web sites have some visual aspects in common and therefore comprise a suitable dataset for our evaluation.

We manually built wrappers to obtain the supervised data of these web sites. Sample home pages from different days for each newspaper were selected and a set of RDF triples were extracted out of them. Table 1 summarizes the data in the training dataset, while Table 2 shows some samples with their attributes and classes. Table 3 summarizes the testing datasets.

Abc and El País home pages are used as training datasets, while El Mundo is used as testing dataset for the generalization test. Regarding the robustness test, El País newspaper performed a layout redesign on 23rd May, 2011, as of Spanish elections held the day before, in order to better present the elections' results. This

Table 1. Training dataset.

	elpais.es	abc.es
sioc:Post	326	122
dc:title	343	126
dc:description	343	123
foaf:depiction	175	56
Total triples	1193	427

[e]http://www.abc.es
[f]http://www.elpais.es
[g]http://www.elmundo.es

Table 2. Training samples.

	X	Y	Width	Height	Font Size	Font Type	...	Triple
x_1	118	5552	310	243	N/A	N/A	...	$rdf{:}type(x_1, sioc{:}Post)$
x_2	119	5558	310	26	18	serif	...	$dc{:}title(x_1, x_2)$
x_3	118	5736	310	28	16	sans-serif	...	$dc{:}description(x_1, x_3)$
x_4	787	332	300	703	N/A	N/A	...	$rdf{:}type(x_4, sioc{:}Post)$
x_5	787	507	284	53	22	sans-serif	...	$dc{:}title(x_4, x_5)$
x_6	787	569	300	34	12	sans-serif	...	$dc{:}description(x4, x_6)$
x_7	114	1247	390	301	N/A	N/A	...	$rdf{:}type(x_7, sioc{:}Post)$
x_8	114	1474	299	26	22	sans-serif	...	$dc{:}title(x_7, x_8)$
x_9	114	1509	390	34	12	sans-serif	...	$dc{:}description(x_7, x_9)$
x_{10}	114	1258	390	194	N/A	N/A	...	$foaf{:}depiction(x_7, x_{10})$

Table 3. Testing datasets.

	Robustness Test elpais.es	Generalization Test elmundo.es
sioc:Post	79	529
dc:title	79	546
dc:description	79	545
foaf:depiction	39	223
Total triples	276	1843

affected the performance of our manually constructed wrapper, and therefore makes it an interesting testing sample for the robustness test.

4.3. *Results and discussion*

Table 4 shows the results of the generalization test. As the testing samples belong to a news site, they share some visual aspects with the training data, so the system managed to extract most pieces of news right. Regarding precision, the system failed to extract properly some pieces of news that are shown in Fig. 4. According to the learned patterns, the top text in a piece of news should represent the title and the lowest one the news description, although in this case the top text is just a news category. Also, the system could not extract some pieces of news in the middle column, as their size is smaller than the pieces of news that were used to learn the patterns, which affected the results of the recall metric.

In the case of the technique of wrapper induction, it requires a new wrapper to be constructed for each new site. Therefore, generalization results do not include wrapper induction as long as it does not have generalization capabilities.

Table 5 shows the results of robustness test. The system shows top precision and very high recall. The high precision is achieved because all the news that appear in the web site have a similar one whose pattern was already learned in the training

Table 4. Evaluation of generalization.

	Wrapper	Rules
Triples	–	1843
Extracted triples	–	1423
Correct triples	–	1325
sioc:Post precision	–	98.07%
sioc:Post recall	–	76.94%
sioc:Post F-score	–	86.23%
dc:title precision	–	98.13%
dc:title recall	–	76.74%
dc:title F-score	–	86.13%
dc:description precision	–	83.60%
dc:description recall	–	76.70%
dc:description F-score	–	80.00%
foaf:depiction precision	–	100.00%
foaf:depiction recall	–	36.32%
foaf:depiction F-score	–	53.29%
General precision	–	**93.11%**
General recall	–	**71.89%**
General F-score	–	**81.14%**

Fig. 4. Extraction errors on a new web site.

Table 5. Evaluation of robustness.

	Wrapper	Rules
Triples	276	276
Extracted triples	196	238
Correct triples	196	238
sioc:Post precision	100.00%	100.00%
sioc:Post recall	68.35%	86.08%
sioc:Post F-score	81.20%	92.52%
dc:title precision	100%	100.00%
dc:title recall	67.50%	86.25%
dc:title F-score	80.60%	92.62%
dc:description precision	100.00%	100.00%
dc:description recall	67.50%	86.25%
dc:description F-score	80.60%	92.62%
foaf:depiction precision	100.00%	100.00%
foaf:depiction recall	91.89%	86.49%
foaf:depiction F-score	95.77%	92.75%
General precision	100.00%	**100.00%**
General recall	71.01%	**86.23%**
General F-score	83.05%	**92.61%**

phase. Regarding recall, the system managed to extract news that were published under a new layout, while the manually built wrapper failed to achieve so.

The tests prove the robustness of our system, which makes it a suitable tool for automatic maintenance of wrappers. Additionally, the system shows good generalization capabilities, which turns it into a useful tool for the automatic, unsupervised generation of wrappers in unforeseen web sites. The evaluation has been performed on a specific domain. In a different domain, it is expected that the rules would extract those HTML fragments that resemble pieces of news according to the evaluation performed. Mismatches would only happen in those cases where fragments visually appear to be pieces of news but which are not because of their contents.

5. Related work

Plenty of approaches have already dealt with the problem of extracting information out of web sites which do not publish metadata that describe them.[4,3]

As said, some systems provide tools that allow the manual annotation of web resources and induct wrappers for information extraction. Some examples of these tools are Piggy Bank,[25] Reform,[26] Thresher[10] or Marmite.[27] These tools work very well on web sites that have been annotated by a user. The main limitation is that they require user supervision for every new web site, as long as a wrapper that was built for one web site cannot be applied to a different site.

Also, there are studies that provide solutions to the problem of wrapper maintenance,[28] some of them using machine learning techniques for this purpose.[12] However, they do not address the problem of wrapper generalization to different web sites.

Other approaches perform automatic unsupervised wrapper induction. These systems attempt to identify repetitive patterns that occur in a document to build extraction patterns. This technique is applied by different extractors[11,29] by identifying differences among neighbour resources, as well as in a supervised basis.[30] These approaches do not rely on user annotated document fragments, and therefore require no interaction from a user. Some approaches integrate external knowledge to annotate the data type, such as Wikipedia,[31] to classify and identify the type of the extracted structured data. In comparison to our approach, these systems provide a structured syntactic output, where no semantic relations are obtained for each of the fields extracted out of the web resource.

Similarly, the systems that are based on visual features do not require manual supervision when applied to new web sites. Some approaches[32,13] perform this by defining some heuristics that are based on the HTML tags used for layouts, such as tables or titles. This limits the approach to the web sites that follow these design guidelines. Finally, another tool[14] uses visual information to extract data from web pages, which makes it work in web pages without preliminar knowledge about them. The main difference is that that tool only builds a hierarchical structure of the web page based on titles and sections.

The proposed algorithm for rule induction is similar in nature to RISE algorithm,[21] which also performs a progressive generalization of specific rules. Other first-order logic algorithms such as TILDE[20] perform a top-down rule induction, which results in a bigger, more exhaustive search space. Traditional rule induction algorithms such as C4.5[22] produce rules that cannot involve several variables (HTML fragments in our case), and therefore are not suitable for our problem, although are usually simpler and faster.

6. Conclusions and Future Work

In this paper, a system that performs induction of first-order logic rules to extract data from unstructured web resources has been described. Our system can be used to extract data from web sites with an unknown DOM tree structure, thanks to the fact that it is based on the visual features of the elements shown in the web browser. This allows extracting semantic information from unstructured web resources without external supervision, given a previous training stage.

The approach has been validated on an environment of newspapers that share some visual features but which have with very different DOM tree structures. The performance of the algorithm shows high precision and good recall. After a first training stage, the system is able to extract data from sites with similar appearance, as well as keep working even on the event of changes on a web resource's DOM tree.

This helps in solving the typical maintainability and generalization problems that exist in wrapper induction techniques. Thus, it is a step forward to solving the bootstrap problem of Linked Data, i.e., the lack of semantically annotated data in the Web.

Future works involve experimenting with other content-specific attributes, such as text category, type of content or Natural Language Processing (NLP) patterns. This should allow applying the algorithm to situations where visual features are not enough to identify the data, such as when identifying entities such as people, locations or dates in web pages.

Acknowledgments

This research work is funded by the European Commission under the R&D project OMELETTE (FP7-ICT-2009-5), and by the Spanish Government under the R&D project *Contenidos a la Carta* (TSI-020501-2008-114).

References

1. T. Berners-Lee, J. Hendler, O. Lassila *et al.*, The semantic web. *Scientific American* 284(5):28–37 (2001).
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *sbc* 14(w3c):9 (2009).
3. C. H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1411–1428 (2006).
4. R. Kosala and H. Blockeel. Web mining research: A survey. *ACM SIGKDD Explorations Newsletter* 2(1):1–15 (2000).
5. P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science* 337(1-3):217–239 (2005).
6. David T. Barnard, Gwen Clarke, and Nicolas Duncan. Tree-to-tree correction for document trees (1995).
7. Weimin Chen. New algorithm for ordered tree-to-tree correction problem. *J. Algorithms* 40(2):135–158 (2001).
8. Nicholas Kushmerick. Wrapper induction for information extraction (1997).
9. N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence* 118(1-2):15–68 (2000).
10. Andrew Hogue. Thresher: Automating the unwrapping of semantic content from the world wide web. In *Proceedings of the Fourteenth International World Wide Web Conference*, pages 86–95 (ACM Press, 2005).
11. Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, Universita Roma, Tre Universita, Basilicata Universita, and Roma Tre. Roadrunner: Towards automatic data extraction from large web sites. pages 109–118 (2001).
12. Kristina Lerman, Steven N. Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research* 18:2003, (2003).
13. Liu Wei, Xiaofeng Meng, and Weiyi Meng. Vision-based web data records extraction. In *WebDB* (2006).
14. F. Canan Pembe and Tunga Güngör. A tree learning approach to web document sectional hierarchy extraction. In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence* (2010).

15. O. Lassila and R.R. Swick. Resource description framework (rdf) model and syntax. *World Wide Web Consortium, http://www. w3. org/TR/WD-rdf-syntax*.
16. Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California (2000).
17. J.G. Breslin, A. Harth, U. Bojars, and S. Decker. Towards semantically-interlinked online communities. *The Semantic Web: Research and Applications*, pages 500–514 (2005).
18. S. Weibel. The dublin core: A simple content description model for electronic resources. *Bulletin of the American Society for Information Science and Technology* 24(1):9–11 (1997).
19. José Ignacio Fernández-Villamor, Jacobo Blasco García, Carlos A. Iglesias, and Mercedes Garijo. A Semantic Scraping Model for Web Resources — Applying Linked Data to Web Page Screen Scraping. In *Third International Conference on Agents and Artificial Intelligence* (2011).
20. H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence* 101(1-2):285–297 (1998).
21. Pedro Domingos. Rule induction and instance-based learning: A unified approach. pages 1226–1232 (Morgan Kaufmann, 1995).
22. J.R. Quinlan. *C4. 5: programs for machine learning*. (Morgan Kaufmann, 1993).
23. T. Anton. Xpath-wrapper induction by generalizing tree traversal patterns. *Lernen, Wissensentdeckung und Adaptivitt* (*LWA*), pages 126–133 (2005).
24. D. Brickley and L. Miller. Foaf vocabulary specification (2005).
25. D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):16–27 (2007).
26. Michael Toomim, Steven M. Drucker, Mira Dontcheva, Ali Rahimi, Blake Thomson, and James A. Landay. Attaching UI enhancements to websites with end users. *Conference on Human Factors in Computing Systems* pages 1859–1868 (2009).
27. Jeffrey Wong and Jason I. Hong. Making mashups with marmite: Towards end-user programming for the web. *Conference on Human Factors in Computing Systems*, page 1435 (2007).
28. J. Raposo, A. Pan, M. Álvarez, and Á. Viña. Automatic wrapper maintenance for semi-structured web sources using results from previous queries. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 654–659 (ACM, 2005).
29. Arvind Arasu. Extracting structured data from web pages. In *ACM SIGMOD*, pages 337–348 (2003).
30. Yanhong Zhai and Bing Liu. Extracting web data using instance-based learning. In *Proc. of 6th Intl. Conf. on Web Information Systems Engineering* (*WISE'05*), pages 318–331 (2005).
31. Philipp Cimiano, Siegfried Handschuh, Siegfried H., and Steffen Staab. Towards the self-annotating web (2004).
32. Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In *Proc.5 th Asia Pacific Web Conference*, pages 406–417 (2003).