# A SEMANTIC SCRAPING MODEL FOR WEB RESOURCES
## *Applying Linked Data to Web Page Screen Scraping*

José Ignacio Fernández-Villamor, Jacobo Blasco-García, Carlos Á. Iglesias, Mercedes Garijo
*Departamento de Ingniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain*
*jifv@dit.upm.es, j.blasco@alumnos.upm.es, cif@dit.upm.es, mga@dit.upm.es*

Keywords: Information extraction, Linked Data, screen scraping

Abstract: In spite of the increasing presence of Semantic Web Facilities, only a limited amount of the available resources in the Internet provide a semantic access. Recent initiatives such as the emerging Linked Data Web are providing semantic access to available data by porting existing resources to the semantic web using different technologies, such as database-semantic mapping and scraping. Nevertheless, existing scraping solutions are based on ad-hoc solutions complemented with graphical interfaces for speeding up the scraper development. This article proposes a generic framework for web scraping based on semantic technologies. This framework is structured in three levels: scraping services, semantic scraping model and syntactic scraping. The first level provides an interface to generic applications or intelligent agents for gathering information from the web at a high level. The second level defines a semantic RDF model of the scraping process, in order to provide a declarative approach to the scraping task. Finally, the third level provides an implementation of the RDF scraping model for specific technologies. The work has been validated in a scenario that illustrates its application to mashup technologies.

# 1 INTRODUCTION

A growing amount of data is available to users in the web. Web users can enjoy plenty of services and information in e-commerce web sites, electronic newspapers, blogs and social networks. Although this data is available for its consumption by users, its format is not suited for automated agents and computer programs. This has favoured the research in several fields such as web content mining (Kosala and Blockeel, 2000) or Semantic Web (Berners-Lee et al., 2001), that seek manners to build linked interoperable data that can be automatically processed by software systems.

Several approaches such as Linked Data initiative (Bizer et al., 2009) are favouring the publication of annotated data in web resources, so that automatic processes can actually consume this data and perform other operations. Similarly, other research fields attempt to take advantage of this amount of available information, such as mashup applications. However, ontologies and applications that expose their data are not widespread, constraining the Linked Data initiative, mashups and service composition.

The field of Web Content Mining applies data mining techniques to the discovery and extraction of information available on the Web. Web Content Mining comprises several research fields such as Information Extraction or Natural Language Processing, which research related techniques that are used to extract data from web documents (Chang et al., 2006).

Approaches to the problem of extracting information out of HTML documents considers processing either the DOM tree or the resulting rendering information. The first approach involves defining an extractor or *wrapper* (Kushmerick, 1997) (Kushmerick, 2000) that selects the relevant information out of the DOM tree. The latter is a vision-based approach that attempts to provide a more general solution to the problem by assuming that similar content types have similar visual features (Wei et al., 2006) (Cai et al., 2003).

In this paper, we define a framework for web scraping for the extraction of RDF graphs that represent content in HTML documents. This framework

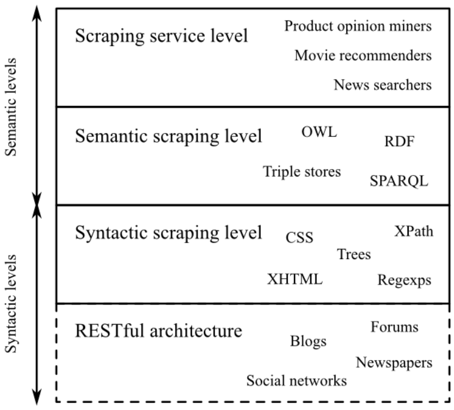

Figure 1: Semantic scraping framework

allows defining services based on screen scraping by linking data from RDF graphs with contents defined in HTML documents. We have used this model to build a semantic scraper that uses RDF-based extractors to select fragments and data from web documents and build RDF graphs out of unstructured information. The model enables to generate graphs in different representations by keeping the original sources in the resulting graph.

The paper is organized as follows. First, the framework for scraping of web resources is defined in section 2. The development of a scenario with semantic scrapers that uses the model is described in section 3. Finally, related work is compared with the one presented in this paper, and the main conclusions and future research lines are presented.

# 2 SEMANTIC SCRAPING FRAMEWORK

In this paper, a framework for using semantic extracted data from the web is defined, which is shown in figure 1. The model considers three levels of abstraction in order to provide an integrated model for semantic scraping:

- Scraping service level. This level comprises services that make use of semantic data extracted from unannotated web resources. Possible services that benefit from using this kind of data can be opinion miners, recommenders, mashups that index and filter pieces of news, etc.
- Semantic scraping level. This level defines a model that maps HTML fragments to semantic web resources. By using this model to define the mapping of a set of web resources, the data from the web is made available as knowledge base to scraping services. This level provides semantics to the syntactic scraping capabilities of the level below.
- Syntactic scraping level. This level gives support to the interpretation to the semantic scraping model. Wrapping and Extraction techniques such as DOM selectors are defined at this level for their use by the semantic scraping level.

The framework is stacked on top of the REST architectural style (Fielding, 2000). This architectural style is the one the World Wide Web is based on, and defines a hypermedia distributed system. The additional semantics and data mappings that are necessary to allow information scraping on a RESTful architecture are defined by the upper levels of our framework.

In this section, the levels defined in the framework are described in further detail.

## 2.1 SCRAPING SERVICE LEVEL

This level comprises all services and applications that make use of the semantic scraping level by providing value to an end user. Services such as opinion miners, recommenders, mashups, data mining applications or any agent-based service benefit from an increased level of knowledge. Other approaches that make use of the semantic scraping facilities can be automatic service composition for automatic generation of new applications out of existing services. Scraping technologies allow getting wider access to data from the web for these kinds of services.

The paradigm behind scraping services has subtle differences from that behind traditional Semantic Web applications or knowledge-based systems. While annotated data in the Semantic Web allows automatic knowledge extraction and retrieval by automatic agents, data in unstructured web documents require prior supervision of some kind to allow information extraction. This implies that when designing a scraping service some of the following steps might be required:

- Scraping data identification. Data that wants to be scraped and merged with other knowledge is identified in this task. Target web sites and resources are identified for fragment extraction.
- Data modelling. A model to represent the extracted data is defined in this task. Either existing ontologies might be available or new ones should be defined. The result from this step is an ontology that fits the data that needs to be extracted. A bounded context, i.e. a conceptual context where a domain model has a non-ambiguous meaning,

should be identified in order to separate domain models of similar fields. Methodologies for the definition of ontologies can be useful for this task.

- Extractor generalization. In order to perform massive extractions, enough samples need to be collected to generalize an appropriate extractor. This collection of samples needs to be provided to a human administrator or an automated or semi-automated module. Using this data set, one or more extractors are defined at the semantic scraping level and serve to provide additional knowledge to the scraping service.

Consider a movie recommender that requires extracting data from the Internet Movie Database. Data about reviews are added to the recommender's knowledge in order to enable collaborative filtering of movies. Reviews and user reviewers are therefore the identified data to scrape. As long as an existing movie ontology is defined, no new modelling would be needed for this task. Also, in case extractors are built automatically using a machine learning approach, data samples should belong to the bounded context of cinema and movies. Therefore, considering our framework, the movie recommender is a scraping service that makes use of additional knowledge available in unstructured resources from the web.

## 2.2 SEMANTIC SCRAPING LEVEL

This level defines the mapping between web data and semantic web resources. An RDF model that allows formalizing this mapping has been defined.

Applying the model to the definition of extractors of web resources allows separating the declarative from the procedural model in the web content extraction process. This enables implementing technology-independent extractors or automating certain tasks such as focused and personalized scraping, as will be described in section 3.

The proposed model allows to reference HTML fragments in RDF and define web content extractors, being a basis for the programmatic definition of extractors for screen scraping. This requires bridging the gap between both RDF and HTML's data models. HTML is a markup language for documents with a tree-structured data model. On the other hand, RDF's data model is a collection of node triples, defined by a subject, a predicate, and an object. Each node can be a text literal, a resource (identified by a URI) or a blank node.

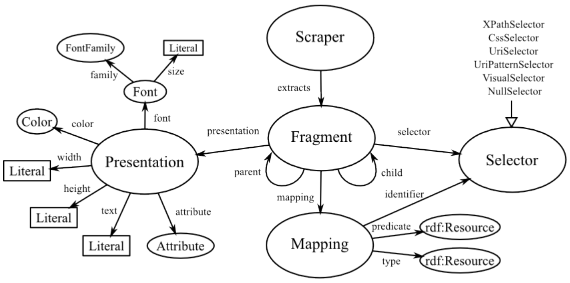

Figure 2: Semantic scraping RDF model

A model comprisen of a vocabulary of RDF terms has been defined to represent HTML fragments and their mapping to RDF resources. This serves as a model for the semantic scraping level. A summary of the model is shown in figure 2. The basic classes of the model are described next:

**Scraper** A scraper is an automatic agent that is able to extract particular fragments out of the web.

**Fragment** Any element of an HTML document. It serves to represent and traverse a whole subtree of a document.

**Selector** A condition that indicates which this element is. Different selector terms are defined for each selector type. Selectors can be XPath expressions, CSS selectors, URI selectors, etc. Selectors are means to identify a web document fragment.

**Mapping** The mapping between a fragment and an RDF resource or blank node. An identifier is defined to map the fragment to a URI. A predicate between the parent's mapped fragment and this is defined to produce an RDF triple. Also, an RDF class can be assigned to the mapped resource of this fragment.

**Presentation** The representation of a fragment. This includes HTML attributes as well as visual parameters such as color, size or font.

The proposed vocabulary serves as link between HTML document's data and RDF data by defining a model for scraping agents. With this RDF model, it is possible to build an RDF graph of HTML nodes given an HTML document, and connects the top and lowest levels in the scraping framework to the semantic scraping level.

## 2.3 SYNTACTIC SCRAPING LEVEL

This level defines the required technologies to extract data from web resources. It provides a basis for a interpretation of the semantics defined in the semantic scraper level's RDF model. Some of the considered scraping techniques in this level are the following:

- Content Style Sheet selectors. Content Style Sheets define the visual properties of HTML elements. These visual properties are mapped to elements through the use of CSS selectors, defined through a specific language. Therefore, CSS is one technology that serves to select and extract data.

- XPath selectors. Similarly to CSS selectors, the XML Path Language[1] is a different language for HTML node selection.
- URI patterns. URI patterns allow to select web resources according to a regular expression that is applied on the resource's URI. While XPath or CSS selectors are able to select an element at document level, URI patterns allow selecting documents, i.e. resources representations, according to the resource's URI.
- Visual selectors. Visual information can be used to select nodes. HTML nodes are rendered with a set of visual properties given by the used browser. It is common that human users prefer uniform web designs. Web designers thus make elements of a same kind to be rendered with similar visual properties to help identification. A visual selector is a condition that combines several visual properties of an element to identify the element's class.

Other kinds of selectors that process HTML's inner text are available as well and fit into the model. This way, extractions from natural language parsing or text tokenization are possible.

Selectors at the syntactic scraping level allow to identify HTML nodes. Either a generic element or an unambiguously identified element can be selected using these techniques. Their semantics are defined in the upper semantic scraping level, allowing to map data in HTML fragments to RDF resources.

An example of the usage of selectors for a news scraper is shown in figure 3. In this case, a scraper is defined that is able to scrape a set of posts (by using the SIOC ontology (Breslin et al., 2006)) from a specific URI. A sample mapped RDF graph is shown in the figure, too.

The scraper can be retrieved by using the following SPARQL[2] query at a scraper store:

```
SELECT ?scraper
WHERE { ?scraper rdf:type sc:Scraper
        ?scraper sc:extracts ?fragment
        ?fragment sc:mapping ?mapping
        ?mapping sc:type sioc:Post
      }
```

With that, all the stored scrapers that extracts posts will be retrieved, allowing further queries to them to obtain the desired information.

# 3 VALIDATION

A semantic scraper that follows the defined framework has been developed[3]. The defined RDF model is used at the semantic scraping level. Open Source Webkit library[4] has been used to implement the scraper at the syntactic scraping level. Both a scraping API and an HTTP proxy pattern interface are supported by the scraper.

The considered scenario points out some of the benefits of using a semantic scraper. The scenario has the goal of showing the most commented sports news on a map, according to the place they were taken. The main techonological challenges behind this scenario are (i) the lack of semantic annotations in the sports news web sites, (ii) the potential semantic mismatch among these sites and (iii) the potential structural mismatch among these sites.

The rationale to use semantic scraping resides in that many of the available sport news web sites do not provide semantic annotations nor microformats, and do not include some relevant information in their RSS feeds, such as location, users' comments or ratings.

The general approach to realise this scenario would consist of (i) defining the data schema to be extracted from selected sports news web sites, (ii) defining and implementing these extractors and (iii) defining the mashup by specifying the sources, mashup operators and renderers. This paper is focused on the first two steps. First, the extracted data is modelled using an ontology derived from SportsML[5], a model for representing sports data.

Then, the scraping rules are defined. Our framework allows the definition of these rules based on structural or visual properties, according to the model presented previously. In order to avoid massive scraping, semantic properties are used for defining this task. For example, scrape only web resources of this news category which is obtained from the newspapers' home page. Some examples of the expressivity of this approach are: identify the news title are the one with big font and a link to a detail page, or select only news with an associated image. Also, other types of rules are possible, such as the ones shown in figure 3.

Also, when considering scraping, recursive access to resources needs to be performed. For instance, a piece of news may show up as a title and a brief summary in a newspaper's homepage, but offers the whole content (including location, authors, and more) in its own URL. This can make scraping a time consuming

[1] http://www.w3.org/TR/xpath/

[2] http://www.w3.org/TR/rdf-sparql-query/

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

[4] http://webkit.org/

[5] http://www.iptc.org/cms/site/index.html?channel=CH0105

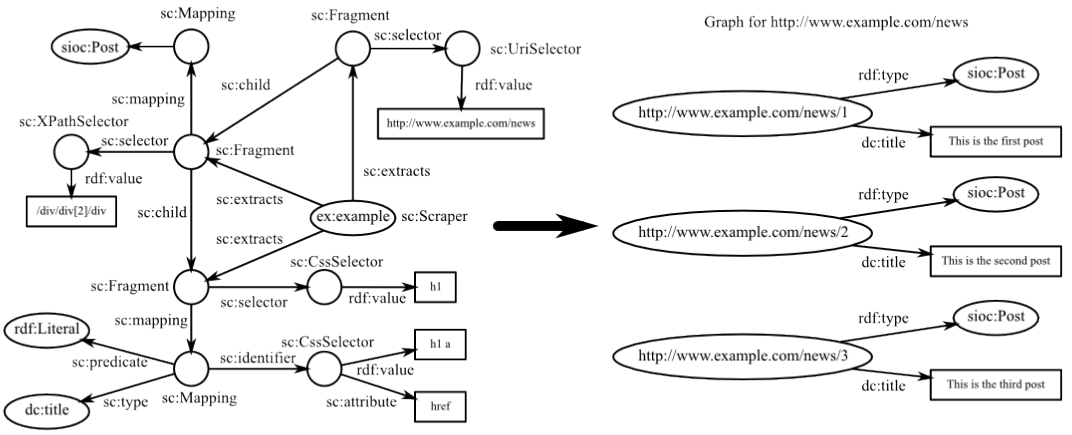

Figure 3: Example of semantic scraper

task. Our model allows focused scraping, i.e., scraping only the required resources, considering a given, high-level goal.

In this scenario, focused scraping is performed by limiting the scraping to sports news. Category information in newspapers' home pages is used to reduce the number of recursive steps that need to be performed in order to retrieve all the information.

Following this approach, the data is extracted and the news are placed on a map. This shows how the defined framework can be applied and how this research work can be integrated with mashup technologies.

# 4 RELATED WORK

The related work in web scraping technology can be classified attending to the targeted user, the end user or the programmer.

In the first category, we can point out the systems Piggy Bank (Huynh et al., 2007), Reform (Toomim et al., 2009), Thresher (Hogue, 2005) and Marmite (Wong and Hong, 2007). Piggy Bank (Huynh et al., 2007) provides also a web scraper that extracts RDF out of web pages. Nevertheless, their approach is based on a browser plugin that provides a visual interface for programming the scraper and produces JavaScript as a result. Reform (Toomim et al., 2009) proposes that scrapers can be attached to web pages by non programmers. The system is also integrated in Firefox web browser and allows users to annotate which parts of the web page contains a predefined pattern, and includes machine learning techniques for generalisating these annotations in the scraper. Thresher (Hogue, 2005) provides an interface where users can specify examples of semantic patterns by highlighting and assigning relevant features in the semantic web browser Haystack. The fact that these tools cannot be compatible, since their scrapers are based on JavaScript code, points out some of the benefits of our proposal. First, the definition of a general model for scrapers allows separating its programming model for its interface. In addition, our proposal defines scraping task goals at the knowledge level in RDF. In this way, scraping task goals can be queried, analysed and composed in RDF, which is not feasible if we specify these goals in an implementation language such as JavaScript.

In the second category, we can point out Chickenfoot (Bolin et al., 2005), Denodo (Pan et al., 2002). Both define JavaScript libraries for scraping. This work fits in the implementation level of our framework for implementing the scraping of a given RDF scraping specification, which enables interoperability across scraping implementation libraries.

GRDDL (Hazaël-Massieux and Connolly, 2004) is the standard technology for extracting RDF out of HTML or XML documents. In order to extract the RDF semantic information, a GRDDL-aware agent should read the GRDDL annotations of the HTML/XML resources, which are usually provided as XSLT annotations. Our approach goes one step further than GRDDL, even though it is compatible with it. Our approach can address both GRDDL-annotated resources as well as non-annotated resources. GRDDL proposes a syntactic approach that can benefit from the reuse of scrapers among web sites. However, GRDDL specifies an unidirectional mapping between the implementation (XSLT scraper) and the web resource. In contrast, our approach provides a semantic framework which enables to dereference scraped data and reason about the scraping process. This enables enhacements in the scraping process, such as reasoning on visual aspects of the web resource, distributed scraping using multiagent systems, or focused scraping and reasoning on the scraped resources (for example, search a piece of sports news about Rafael Nadal taken in China).

# 5 CONCLUSIONS AND FUTURE WORK

In this paper, the problem behind web information extraction and screen scraping has been outlined, while the main approaches to it have been summarized. The lack of an integrated framework for scraping data from the web has been identified as a problem, and this paper presents a framework that tries to fill this gap.

An RDF model for web scraping has been defined at the semantic scraping level. The objective of this RDF model should not be confused with that of RDFa. RDFa defines a format for marking up HTML elements to extract an RDF graph. Our model complements RDFa by allowing RDF graphs to refer to data that is present in HTML fragments in an unannotated HTML document. This enables an open framework for web scraping. The tasks of building an RDF graph out of a web document has been shown. With this, a semantic screen scraper has been developed. The semantic screen scraper produces RDF graphs out of web documents and RDF-defined extractors, that offer interoperable scraping information.

Future works involve experimenting with the automatic construction of mappings out of incomplete ones or unstructured HTML resources. While some of the approaches to information extraction deal with wrapper induction or vision-based approaches, the modelling of web page fragments as web resources changes the paradigm behind this task. Approaches such as machine learning or graph analysis can be combined and applied to this different scenario.

# ACKNOWLEDGEMENTS

This research project 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 projects *Contenidos a la Carta* (TSI-020501-2008-114) and T2C2 (TIN2008-06739-C04-01).

# REFERENCES

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data-the story so far. *sbc*, 14(w3c):9.

Bolin, M., Webber, M., Rha, P., Wilson, T., and Miller, R. C. (2005). Automation and customization of rendered web pages. *Symposium on User Interface Software and Technology*, page 163.

Breslin, J., Decker, S., Harth, A., and Bojars, U. (2006). SIOC: an approach to connect web-based communities. *International Journal of Web Based Communities*, 2(2):133–142.

Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y. (2003). Extracting content structure for web pages based on visual representation. In *Proc.5 th Asia Pacific Web Conference*, pages 406–417.

Chang, C., Kayed, M., Girgis, M., and Shaalan, K. (2006). A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1411–1428.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California.

Hazaël-Massieux, D. and Connolly, D. (2004). Gleaning resource descriptions from dialects of languages (grddl). *World Wide Web Consortium, W3C Coordination Group Note NOTE-grddl-20040413*.

Hogue, A. (2005). 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.

Huynh, D., Mazzocchi, S., and Karger, D. (2007). 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.

Kosala, R. and Blockeel, H. (2000). Web mining research: A survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15.

Kushmerick, N. (1997). Wrapper induction for information extraction.

Kushmerick, N. (2000). Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118:2000.

Pan, A., Raposo, J., Álvarez, M., Montoto, P., Orjales, V., Hidalgo, J., Ardao, L., Molano, A., and Viña, A. (2002). The denodo data integration platform. *Very Large Data Bases*, page 986.

Toomim, M., Drucker, S. M., Dontcheva, M., Rahimi, A., Thomson, B., and Landay, J. A. (2009). Attaching UI enhancements to websites with end users. *Conference on Human Factors in Computing Systems*, pages 1859–1868.

Wei, L., Meng, X., and Meng, W. (2006). Vision-based web data records extraction. In *WebDB*.

Wong, J. and Hong, J. I. (2007). Making mashups with marmite:towards end-user programming for the web. *Conference on Human Factors in Computing Systems*, page 1435.