A Framework for Goal-Oriented Discovery of Resources in the RESTful Architecture

José Ignacio Fernández-Villamor, Carlos A. Iglesias, and Mercedes Garijo

***Abstract*—One of the challenges facing the current web is the efficient use of all the available information. The Web 2.0 phenomenon has favored the creation of contents by average users, and thus the amount of information that can be found for diverse topics has grown exponentially in the last years. Initiatives such as linked data are helping to build the Semantic Web, in which a set of standards are proposed for the exchange of data among heterogeneous systems. However, these standards are sometimes not used, and there are still plenty of websites that require naive techniques to discover their contents and services. This paper proposes an integrated framework for content and service discovery and extraction. The framework is divided into several layers where the discovery of contents and services is made in a representational stateless transfer system such as the web. It employs several web mining techniques as well as feature-oriented modeling for the discovery of cross-cutting features in web resources. The framework is used in a scenario of electronic newspapers. An intelligent agent crawls the web for related news, and uses services and visits links automatically according to its goal. This scenario illustrates how the discovery is made at different levels and how the use of semantics helps implement an agent that performs high-level tasks.**

***Index Terms*—Agents, contents, discovery, information extraction, representational stateless transfer system (REST), services.**

I. Introduction

THE Web 2.0 phenomenon has caused a mushrooming of websites and applications that allow users to produce content without any sort of technical skill. This has caused the web to become a highly rich source of contents in the so-called digital age, thanks to users' collaboration and other content providers. Also, with the emergence of mashup technologies, developers are able to combine existing services and data sources to quickly build new applications, in a fashion similar to Web 2.0.

The vast amount of information and services available on the web makes it a platform for the development of mashed-up applications with intensive information usage. The research field of the Semantic Web [1] and the linked data initiative [2] have defined techniques and standards for the semantic representation of information, which have been increasingly adopted in the web.

However, there are still plenty of websites that do not provide appropriate semantic metadata in the resources they publish. This causes their services and contents to be processable only by the human users who visit these websites. The reasons for this can be various: because of limited knowledge by developers of the Semantic Web standards, or because of the limited effort that developers can spend on these tasks.

Also, the so-called deep web contains many web resources that are not discoverable by crawlers. The deep web is the subset of the web that is only accessible behind web forms. It, thus, requires a different treatment for reaching its information and accessing its contents and services, which makes this part of the web hidden from most automatic agents.

This paper proposes a framework for the discovery of services and contents in the web. Our framework allows intelligent focussed crawling [3] and agents' accessing the deep web. Therefore, an agent architecture that uses this discovery framework for goal-oriented discovery of resources is also described. This agent architecture allows implementing agents that intelligently crawl and use services for retrieving contents in the web that correspond to some top goals, usually, stated by the user. This has let us implement an agent that covers a use case in a related news search scenario. The agent searches the web and combines different information sources to carry out effective reasoning that determines its decisions and behavior during the crawling.

This paper is structured as follows. Section II defines the discovery framework that is followed in the rest of the paper. Section III proposes the agent model that builds the plans for smart discovery of contents, and Sections IV and V describe, respectively, service- and content-level discovery. Section VI describes a scenario in which the agent has been put to use. Section VII summarizes related work that deals with the intelligent discovery of services and contents. Finally, Section VIII draws a conclusion and indicates possible future research.

II. Discovery Framework

An integrated framework for content and service discovery is defined in this section. We understand discovery as the process of identification and construction of an element's

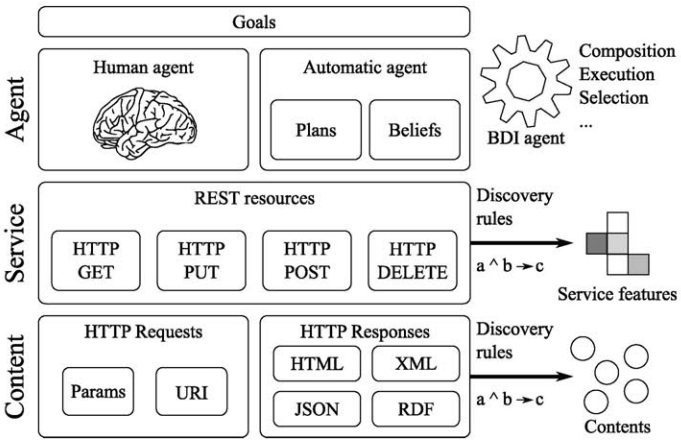

Fig. 1. Discovery framework.

semantically meaningful description at some particular level. The framework is shown in Fig. 1.

It is stacked on top of the representational stateless transfer (REST) architectural style [4], the architectural style on which the World Wide Web is based. This framework uses three levels of abstraction on the content and services that are available on the web. They are listed next from top to bottom.

1) *Agent level*: This layer comprises the orchestration of services for fulfilling a user goal. Searching blogs to obtain relevant information about a product, look for the best price, and suggest to the user which are the best ones, is an example of an orchestrated plan that is executed by an agent (either human or machine) attempting to reach a particular goal.
2) *Service level*: This layer comprises the services that agents are able to use on the web, which nowadays vary from search services to booking services, social networking services, and so on. These services are generally orchestrated at a higher layer. They make use of contents in the lower layer by exchanging requests and responses with representations of resources that are present on the web.
3) *Content level*: This level comprises the requests and responses that are exchanged between clients and servers when interacting with web resources. In the REST architectural style, requests consist mainly of a verb and a uniform resource identifier (URI), optionally, with parameters, while responses vary in format, although in the web they will usually be in hyper text markup language (HTML).

Discovery will take place at these levels. At the service level, REST resources would be the discoverable elements to be analyzed, with semantic service descriptions being obtained. Discovery of data at the content level would produce meaningful semantically-annotated information.

Therefore, a method to extract semantic descriptions out of unstructured data at every level of the framework will be described. A uniform approach that employs first-order logic rules is employed to model discovery rules that allow identifying features at all levels. The combination of these features will comprise a semantic description for a discovered element.

In the next sections, we will describe the agent architecture that operates at the orchestration layer and composes plans for fulfilling goals. We will also describe the techniques that allow performing discoveries for already existing pieces of contents and services.

III. Agent Model

This section defines an agent model on top of the service level layer in the proposed service discovery framework. This agent model makes use of the REST architectural style through semantically annotated services and contents. As shown in the general framework in Fig. 1, the agent is designed to perform the same tasks as a regular, human user of the web.

This problem statement leads to the agent's ability to browse the web and run services in just the same way a human user would, with discovery rules as the means for extracting semantic descriptions from regular resource representations. The agent would attempt to achieve a top level goal in the same way as a human user, e.g., finding a fact, a place, or a picture for a particular person.

Furthermore, the agent is able to manage the lower levels of discovery logic, i.e., manage the service and content discovery rules. As stated in Section II, the discovery rules are the result of a machine learning algorithm applied to supervised data. This supervised data can be added manually as a training dataset that is processed in a later stage. Furthermore, by introspecting into the discovery rules, the agent can anticipate the content and services that can be extracted out of resources, thus modifying its behavior without interacting with those resources.

A. Architecture

The agent follows the belief–desire–intention (BDI) pattern, which differentiates the independent modules that comprise a reactive system interacting with other systems. In our case, beliefs are resource description framework (RDF) contents that are extracted from web pages. The agent has plans that represent the possible actions that it can perform, such as executing discovered services or visiting links, while the intentions are stacks of these plans to reach a particular goal. These top-level goals therefore represent discovery targets, i.e., contents which the agent attempts to add to its knowledge base.

Thus, both beliefs and goals are sets of RDF triples, while intentions are stacks of plans that are fired upon the creation or deletion of triples in the beliefs and goals triple sets. This makes up a naive adaptation of AgentSpeak's agent model [5] to RDF, which results in an agent model, which is similar to approaches that integrate RDF and Semantic Web standards with BDI agents [6]. The resulting architecture is shown in Fig. 2.

B. Plans

As long as the agent makes use of a RESTful architecture, it is able to interact with resources by exchanging requests

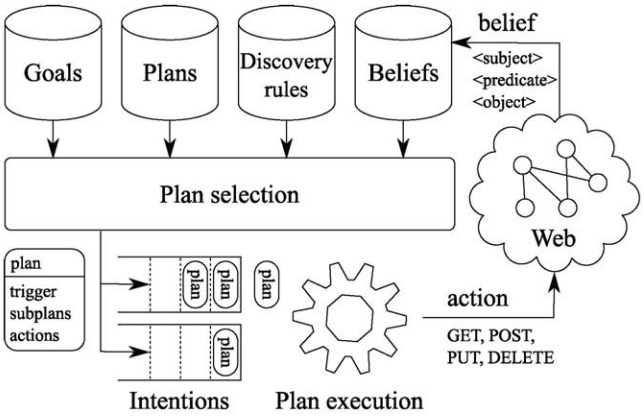

Fig. 2. Agent model.

and responses. This allows following hyperlinks and executing services such as web forms. Therefore, the four main hyper text transfer protocol (HTTP) methods make up the set of actions employed by the agent. Plans are defined around these actions to specify the possible behavior of the agent to reach its goals. They consist of a triggering condition and a set of consequences, either subplans or actions.

The set of plans can be extended, adapting to different domains, in order to establish domain-specific behavior, especially, in the presence of certain services in the system considered. However, a base set of plans will be defined next in order to provide the basic discovery capabilities of the agent.

1) *Focused Crawling Plan:* Whenever a resource's representation is expected to have contents with triples that are present in the goal set (trigger), perform a GET on that resource (action). This way, the agent will crawl the web on a greedy basis, looking to fulfil its top-level goals. This can be done thanks to the content discovery rules in the knowledge base, which allow anticipating the contents to be found in a resource's representation. In AgentSpeak language, this plan is represented as[1]

$$+![x, rdf{:}type, t] : content\_rule(y, x) \wedge [y, sc{:}type, t] \leftarrow get(x). \quad (1)$$

Note that, according to the scraping ontology [7], a *sc*:*type* predicate indicates that in a triple $< a, sc{:}type, b >$, the HTML fragment a is of type b after the extraction is performed.

2) *Deep Web Crawling Plan:* Whenever a keyword-filtered retrieval resource is expected to provide results that meet one or more triple in the goal set (trigger), perform a GET on that resource. This allows using search forms to look for desired contents. The keywords that are entered in the form are the label of the resource, as a naive conversion of RDF into natural language, which is subject to finer grain definition in domain-specific plans. In AgentSpeak, this plan is represented as

$$+![x, rdf{:}type, y] : [z, ms{:}has\_feature, ms{:}Retrieval] \wedge [z, ms{:}has\_feature, ms{:}KeywordFiltered] \wedge [z, ms{:}outputs, y] \wedge [x, rdfs{:}label, l] \leftarrow get(z, \{(keywords, l)\}). \quad (2)$$

IV. Service Level

The discovery at service level involves building semantic service descriptions out of the HTTP interaction data and the discovered semantic contents from the lower level in the proposed discovery framework. The input service model, therefore, comprises the output content model and the HTTP interaction data, i.e., the involved URI, the HTTP verb, and the HTTP parameters.

The output service model used for service discovery applies ideas inspired by mixins, aspect-oriented, and feature-oriented programming paradigms to semantic service description. These paradigms extend the paradigm of object-oriented programming by allowing the modeling of secondary concerns in an isolated way. Feature oriented programming (FOP) [8], [9] is a composition model that allows refining classes through the definition of features, i.e., subclasses with core functionality. Mixins [10], or abstract subclasses, are separate groups of methods, which can be inserted into a class to override its original behavior but which cannot be instantiated on their own. Therefore, mixins serve to implement features in FOP [11]. FOP can be seen as a generalization of the traditional class inheritance in object-oriented programming, and is used to develop the so-called software product lines, i.e., programs that provide different combinations of features [12].

Feature and aspect orientation have inspired other modeling approaches, such as feature-oriented model driven development [13], role-oriented programming [14], and subject-oriented programming [15]. At the service level of the proposed discovery framework, the idea of separating features and concerns is employed to enable feature-oriented service modeling. A service description is a composition of features that allows the reuse of feature descriptions in a similar way in which features, aspects, and abstract classes are reused respectively in FOP, aspect oriented programming (AOP), or mixins paradigms.

A service is modeled by the set of features f_1, f_2, ..., f_i that it has. In web applications, some examples of service features are performing a retrieval operation, requiring user authentication, performing a storage operation, handling images, and outputting a set of resources. The library of features constitutes the output service model.

Service discovery rules are used to provide a mapping between the service feature set and the input model: discovery rules map a set of features f_1, ..., f_k to a set of conditions, defined using the output content model and the HTTP interaction data. For example, given the features f_1 (outputting a set

[1]The syntax [*subject*, *predicate*, *object*] has been used to represent RDF triples not considered in the AgentSpeak language.

of resources) and f_2 (handling images), some discovery rules could be the following:

$$\forall x(|Output(x)| > 1 \Rightarrow ms{:}has\_feature(x, f_1)) \quad (3)$$

$$\begin{aligned} \forall x, y(\ & y \in Output(x) \rightarrow \\ & rdf{:}type(y, foaf{:}Image)\ ) \Rightarrow \\ & ms{:}has\_feature(x, f_1) \wedge \\ & ms{:}has\_feature(x, f_2)\ ). \end{aligned} \quad (4)$$

The discovery rule (3) formalizes feature f_1 by stating that the service's output cardinality has to be greater than one. Meanwhile, the discovery rule (4) formalizes that all output resources are images, and is applicable to services that 1) output a set of resources (as of feature f_1), and 2) handle images (feature f_2). Allowing definitions that are activated in the presence of more than one feature might be regarded as an unnecessary complexity. However, this serves to resolve the issue of feature interaction, already identified in FOP [8].

The output service model is therefore a vocabulary of terms that can be extended, with each term representing a feature. As the framework follows the REST architectural style, the terms for HTTP GET, POST, PUT, or DELETE requests are already part of the vocabulary [16].

V. Content Level

The web is a hypermedia system that follows the REST architectural style [4]. When a client accesses a web resource on a server, the server returns a representation of the resource. Usually, these representations are formatted in HTML, a language that allows defining the structure of a document for rendering on a web browser. An HTML document is structured as a document object model (DOM) tree that defines the logical structure of the HTML document 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 resource representations that include metadata, by enhancing HTML with semantic annotations or by providing RDF representations.

Whenever a resource provides unannotated HTML, a technique that in some way processes the DOM tree needs to be used to identify the structure of the data present in the HTML document and build the associated RDF graph. Also, in a web resource there are DOM fragments that do not provide information, such as advertisements, headers, footers, and decorative elements, while other fragments, such as posts or comments, have valuable information. In our framework, discovery rules will be employed to identify which pieces of information are relevant in a web resource and to identify which relations are stated in a web fragment. For instance, a heading in a piece of news might represent the news title. A discovery rule will use content style sheets (CSS) information, rendering information, or natural language processing (NLP) to identify the relevant data in the resource's representation.

Therefore, the input model that the discovery rules will use at this level comprise HTML fragments, which identify relevant pieces of data in a document, and selectors, which are means for identifying a fragment inside a document. Usually, web scrapers use regular expressions or CSS or XPath selectors for these tasks, while the output of a web browser when rendering a web fragment, which consists of a set of properties, such as typeface, color, or dimensions, can also be used through visual selectors.

On the other hand, the output model comprises the different types of contents available on the web. Ontologies such as semantically-interlinked online communities project (SIOC), friend of a friend (FOAF), and Dublin core (DC), address this issue by defining schemas for the modeling of blog posts, relationships between users, or annotations of metadata in publications, thus constructing the output content model of our discovery framework. Table I summarizes the input and output models of the service and content levels of the discovery framework.

Next is an example of a content discovery rule.

$$\begin{aligned} \forall x, y(\ & uri(x, http{:}//nytimes.com) \wedge \\ & parent(x, y) \wedge \\ & css(y, ".story\ h2\ a")) \Rightarrow \\ & rdf{:}type(sioc{:}Post, y)\ ), \end{aligned} \quad (5)$$

This example defines that all DOM tree nodes that satisfy a particular CSS selector in the New York Times home page are posts, according to the SIOC ontology [17].

The main challenge at the content level is defining discovery rules that are robust and generalizable: a robust rule is one that extracts the same data even with changes in the DOM tree of the web resource. If a rule is not robust, it might stop working once the layout of a web site is changed by its web administrator in some redesign stage [18]. A rule that generalizes is one which is valid for all web resources that contain the same kind of data. If a rule is only valid for the web resource (or resources) that it was defined for, it does not generalize to different resources. The main limitation of wrapper induction is that wrappers are only valid for the web pages for which they were designed [19]. Using NLP and visual selectors as input content model improves the generalization capabilities of the discovery rules [20].

VI. Scenario

A scenario that makes use of the aforementioned agent model and discovery framework has been defined. A bare-bones implementation of the agent has been developed, based on the Open Source tool Scrappy,[2] a screen scraper that was extended with the previously defined intelligent agent model. The result is an agent which is able to address top-level goals for discovering contents and services on the web and will be here used in a scenario to validate the proposed framework. The agent is then configured to perform high-level tasks under a scenario involving electronic newspapers.

A. Description

This scenario will address the task of comparing similar pieces of news when surfing the web. Electronic newspaper

[2]http://github.com/gsi-upm/scrappy

TABLE I

SUMMARY OF TERMS EMPLOYED IN DISCOVERY RULES

Term	Description	Content input	Content output	Service input	Service output
$css(x, y)$	Set of Hyper Text Markup Language (HTML) fragments that fit Representational Stateless Transfer (CSS) selector x in document y	✓	×	✗	×
$xpath(x, y)$	Set of Hyper Text Markup Language (HTML) fragments that fit XPath selector x in document y	✓		✗	×
$font(x)$	Font size of Hyper Text Markup Language (HTML) fragment x	✓		✗	×
$dimension(x)$	Dimension $d \in \Re^2$ of Hyper Text Markup Language (HTML) fragment x	✓		✗	×
$position(x)$	Position $p \in \Re^2$ of Hyper Text Markup Language (HTML) fragment x inside its Hyper Text Markup Language (HTML) document	✓		✗	×
$rdf{:}type(x, y)$	An Resource Description Framework (RDF) node x being of Resource Description Framework (RDF) type y	✗	✓	✓	×
$rdf{:}Statement(s, p, o)$	An Resource Description Framework (RDF) triple with subject s, predicate p, and object o	×	✓	✓	×
$Output(x)$	Set of output contents extracted in the Hyper Text Transfer Protocol (HTTP) response of resource x	×	✗	✓	×
$Input(x)$	Set of input parameters extracted from the Hyper Text Transfer Protocol (HTTP) request to resource x	✗	×	✓	×
$status(x)$	Hyper Text Transfer Protocol (HTTP) status of the Hyper Text Transfer Protocol (HTTP) request to resource x	✗	✗	✓	×
$method(x)$	Hyper Text Transfer Protocol(HTTP) method of the Hyper Text Transfer Protocol (HTTP) request to resource x	✗	×	✓	✗
$ms{:}has\_feature(x, y)$	Service x being described by feature y	✗	×	✓	

readers often read the same piece of news in several sources, to cross-check the views of the different newspapers, therefore, spending a considerable amount of time searching the web for news. Also, users often browse news by similar topics, as they represent their own interests. Although most electronic newspapers provide recommendations to guide users when they are browsing their websites, they do not provide outlinks to other newspapers to help the users easily contrast different sources and how the news has been edited.

A contribution to the solution of this problem is to use an agent that searches various electronic newspapers on behalf of the reader. For this purpose, a browser plugin has been developed in the form of a button that triggers the agent's execution. After clicking the button, the agent is launched and searches contents which might be useful to the user.

The main challenges that are addressed in this scenario are as follows.

1) Extracting the data from the addressed contents. By following the proposed discovery framework, discovery rules are used to perform this extraction. Defining the discovery rules for the information extraction is done semiautomatically, thanks to rule induction algorithms [21].
2) Using services for relating the news posts and identifying the recommendations. Using the discovery framework, the agent can use services by using feature-oriented descriptions of these services. The services will be used by the agent according to the plans that are available in the agent's plan set.

As a result, two stages take place during the agent's lifecycle.

1) *Feeding phase*: The agent is provided with a set of HTML pages from the addressed newspapers that are annotated with the RDF data that their HTML represent. The agent then inducts discovery rules for extracting the semantic representation of the newspapers' resources. Also, service discovery rules are provided for newspapers' search forms and OpenCalais [22] service. OpenCalais is used to enrich the semantic descriptions of news posts in order to identify recommendations. OpenCalais is a service that returns linked data information about a piece of text, returning disambiguated entities about places or people mentioned in the text.
2) *Execution phase*: A plugin that is installed into the web browser[3] allows launching the agent with the goal of returning services that are related to the current news post being browsed. After clicking the button, the agent performs its focussed crawling and discovers a set of results, which are then returned to the web browser so that the user can review the related news that the agent found in the newspapers.

These phases are shown in Fig. 3, which illustrates how the agent is managed and used.

[3]In our implementation, the plugin is simply a bookmark that links to a web service triggering the execution of the agent.

TABLE II

EXAMPLE OF RULES USED IN THE SAMPLE SCENARIO

Level	Rule
Content	$uri(x, \text{"http://abc.es"}) \wedge css(x, \text{".lead"}) \wedge parent(x, y) \wedge css(y, \text{".headline"}) \Rightarrow sioc{:}Post(x) \wedge dc{:}title(x, y)$
Content	$width(x) > 70 \wedge ... \wedge parent(x, y) \wedge font\_size(y) > 12 \wedge separation\_y(x, y) > 5 \wedge ... \Rightarrow sioc{:}Post(x) \wedge dc{:}title(x, y)$
Content	$css(x, \text{"form.search"}) \wedge s = attr(x, \text{"action"}) \Rightarrow ms{:}has\_feature(s, \{ms{:}Retrieval, ms{:}KeywordFiltered, ms{:}News\})$
Service	$method(s) = get \wedge status(s) = 200 \Rightarrow ms{:}has\_feature(s, ms{:}Retrieval)$
Service	$x \in Input(s) \wedge y \in Output(s) \wedge ctag{:}tagged(y, x) \wedge \Rightarrow ms{:}has\_feature(s, ms{:}KeywordFiltered)$
Service	$x \in Output(s) \wedge rdf{:}type(sioc{:}Post, x) \Rightarrow ms{:}has\_feature(s, ms{:}News)$
Service	$x \in Input(s) \wedge rdf{:}type(sioc{:}Post, x) \wedge y \in Output(s) \wedge calais{:}subject(x, y) \Rightarrow ms{:}has\_feature(s, ms{:}Related)$
Agent	$+[x, rdf{:}type, sioc{:}Post] : [z, ms{:}has\_feature, ms{:}Related] \wedge [x, sioc{:}content, c] \rightarrow get(z, (body, c))$
Agent	$+[x, calais{:}subject, y] : true \rightarrow +[z, rdfs{:}label, y] \wedge +![z, rdf{:}type, sioc{:}Post]$

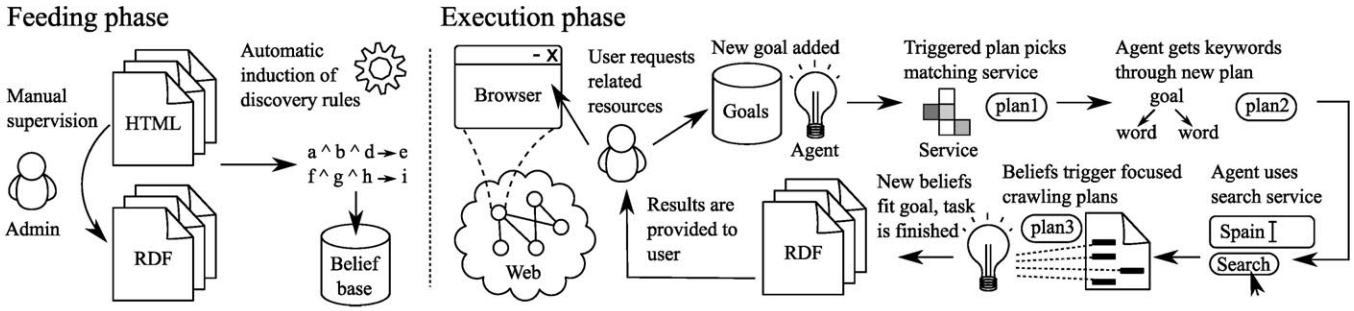

Fig. 3. Agent's lifecycle and interaction with scenario.

B. Results

Table II shows the rules used to configure the agent. As said, it was configured with content rules to extract newspapers' contents, with some content extraction rules automatically generated using rule induction techniques. One of them is shown in the table and uses visual features such as the font size and the width and height. Furthermore, some feature definitions were added to the agent's service-level knowledge base. The related feature is used to describe OpenCalais service, which returns related entities about a piece of news. In the case of agent-level rules, a plan for using OpenCalais service on news retrieval was defined. A second rule is defined for using search services to retrieve news for a particular entity, where a goal for spotting a piece of news is set after an entity's label.

After the definition of discovery rules and the already present base plans, the agent was able to mine and discover contents in different newspapers. Furthermore, the agent employed OpenCalais properly, to enrich the semantic description of contents. After retrieving related entities for a piece of news, the agent executes the newspapers' search services with these entities to retrieve related news.

To evaluate the agent, we provided users with the different news posts and the recommendations by the agent in the browser plugin shown in Fig. 4. The users were then asked to answer different questions about their experience when browsing the web using the agent. Questions about the number of recommendations, the quality, and the degree of relatedness to the original post were asked, and a rating between 1 and 5 was obtained for each question.

As can be observed in Table III, the users were overall satisfied with the agent's recommendations. This helps to validate that the agent's functionality is useful and that a system which is relevant to users has been built.

Fig. 4. Browser plugin for showing related news using an automated agent.

TABLE III

RESULTS OF USERS' SURVEY

Question	Result (1–5)
In general, is the news related?	3.6 (± 0.49)
Number of newspapers is OK	3.8 (± 0.75)
Number of sections is OK	4.1 (± 0.70)
Number of posts is OK	4.2 (± 0.60)
The agent provides useful information	4.1 (± 0.70)
Is it better to surf with the agent's help?	3.8 (± 1.54)

VII. RELATED WORK

There has already been plenty of research work on service and content discovery on the web, with several approaches

that usually have some differences from the one presented in this paper.

At the content level, scraping techniques have been used to extract data, in a similar way, as other information extraction approaches. We can point to the systems Piggy Bank [23], Reform [24], Thresher [25] and Marmite [26], Chickenfoot [27], or Denodo [28]. These approaches propose techniques that facilitate the scraping process, and which can complement our discovery rules. Also, gleaning resource descriptions from dialects of languages (GRDDL) [29] is the standard technology for extracting RDF from HTML documents. Unlike these approaches, ours provides a semantic framework that enables dereferencing the scraped data and reasoning about the scraping process. This enables enhancements in the scraping process, such as reasoning about visual aspects of the web resource, distributed scraping using multiagent systems, or focussed scraping and reasoning about the scraped resources.

Also, we have addressed the issue of describing services using lightweight semantics. WADL [30] defines a format for building and publishing RESTful semantic service descriptions that can be discovered and processed by automatic agents. Other similar RESTful approaches are SA-REST [31] and hRESTS [32], which allow building semantic service descriptions by annotating textual descriptions of service interfaces with RDFa and Microformats [33]. Also, RDForms [34] attempts to add to the Semantic web capabilities similar to HTML forms. In this approach, schemas for indexable, container and settable operations are defined, which represent HTTP GET, POST, and PUT methods. All these approaches focus on facilitating service description, but do not treat the automatic construction of these descriptions. Our approach is based on top of these standards, by allowing the induction of discovery rules to feed the standards with ready-to-use services.

Finally, many research fields have already researched service discovery [35] in parallel to the development of the web. Research behind Foundation for Intelligent Physical Agents (FIPA) agents has already analyzed the problem of service discovery, developing several protocols for diverse possible environments [36], [37], [38]. In pervasive computing, where services are widespread around *ad-hoc* networks, service discovery is a research field [39], [40]. These approaches differ greatly, because of the RESTless nature of the platform they are based on, i.e., often middlewares of different kinds. Some approaches [41], [42], [43], [44] consider Semantic web services, where the web is used more as a platform by following the web services architectures instead of RESTful services. [45] considers the RESTful architecture of the web and integrates an agent-based framework on it and the linked data initiative, though it does not treat the automatic construction of the semantic descriptions that are required for these kinds of solutions.

VIII. Conclusion and Future Work

This paper proposed a framework for the discovery of services and content in the web, describing an algorithm for the induction of rules for discovery, as well as its application to data and resource levels in the REST architectural style of the web. An agent architecture that fits the discovery framework was also defined for implementing combined services or contents in cases where discovery is required. A scenario that made use of an agent to discover related news items was described. The use case illustrated how the agent performs discovery tasks whenever appropriate and makes plans, thanks to the semantic descriptions of the discovered elements and the agent's ultimate goals. The behavior exhibited by the agent showed the potential of using semantics to express the data discovered on the web.

Future work will involve building a broader library of reusable feature descriptions and content types to enhance the output models at each discovery level. Here, a reduced subset of building blocks has been shown to solve a complex use case that involved typical elements in the web, such as news posts. Further training sets for the semantic definitions would improve the generalizability of the discovery rules to other scenarios, which could differ greatly from the ones considered in this paper, thus increasing the agent's versatility.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Sci. Am.*, vol. 284, no. 5, pp. 28–37, 2001.

[2] C. Bizer, T. Health, K. Idehen, and T. Berners-Lee, "Linked data on the web," in *Proc. 17th Int. Conf. WWW*, 2008, pp. 1265–1266.

[3] S. Chakrabarti and B. Dom, "Focused crawling: A new approach to topic-specific web resource discovery," *Comput. Netw.*, vol. 31, pp. 1623–1640, Feb. 1999.

[4] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Berkeley, CA, USA, 2000.

[5] A. S. Rao, "Agentspeak (l): BDI agents speak out in a logical computable language," in *Proc. 7th Eur. Workshop Modelling Autonomous Agents MultiAgent World: Agents Breaking Away*, LNCS 1038. Eindhoven, The Netherlands: Springer, 1996, pp. 42–55.

[6] M. Laclavık, Z. Balogh, M. Babık, and L. Hluchỳ, "Agentowl: Semantic knowledge model and agent architecture," *Comput. Inf.*, vol. 25, no. 5, pp. 419–437, 2006.

[7] J. I. Fernández-Villamor, C. A. Iglesias, and M. Garijo, "A semantic scraping model for web resources–Applying linked data to web page screen scraping," in *Proc. Third Int. Conf. Agents Artif. Intell.*, 2011, pp. 451–456.

[8] C. Prehofer, "Feature-oriented programming: A fresh look at objects," in *Lecture Notes in Computer Science*, vol. 1241. Berlin, Germany: Springer, 1997, pp. 419–443.

[9] S. Apel, T. Leich, M. Rosenmüller, and G. Saake, "Combining feature-oriented and aspect-oriented programming to support software evolution," in *Proc. AMSE05 ECOOP05*, 2005, pp. 3–16.

[10] G. Bracha and W. Cook, "Mixin-based inheritance," in *Proc. Eur. Conf. Object-Oriented Programming Object-Oriented Programming Syst. Languages Appl.*, 1990, pp. 303–311.

[11] S. Apel, T. Leich, and G. Saake, "Aspectual mixin layers: Aspects and features in concert," in *Proc. 28th Int. Conf. Softw. Eng.*, 2006, p. 131.

[12] R. Lopez-Herrejon, "Understanding feature modularity in feature oriented programming and its implications to aspect oriented programming," in *Proc. ECOOP PhDOOS Workshop Doctoral Symp.*, 2005.

[13] S. Trujillo, D. Batory, and O. Diaz, "Feature oriented model driven development: A case study for portlets," in *Proc. 29th Int. Conf. Softw. Eng.*, 2007, pp. 44–53.

[14] F. Steimann, "On the representation of roles in object-oriented and conceptual modelling," *Data Knowledge Eng.*, vol. 35, no. 1, pp. 83–106, 2000.

[15] W. Harrison and H. Ossher, "Subject-oriented programming: A critique of pure objects," *ACM Sigplan Notices*, vol. 28, no. 10, pp. 411–428, 1993.

[16] J. I. Fernández-Villamor, C. Iglesias, and M. Garijo, "Microservices: Lightweight service descriptions for REST architectural style," in *Proc. 2nd Int. Conf. Agents Artif. Intell.*, 2010, pp. 186–189.

[17] J. G. Breslin, A. Harth, U. Bojars, and S. Decker, "Toward semantically-interlinked online," in *Proc. 2nd Eur. Semantic Web Conf.*, LNCS 3532. 2004, pp. 500–514.
[18] K. Lerman, S. N. Minton, and C. A. Knoblock, "Wrapper maintenance: A machine learning approach," *J. Artif. Intell. Res.*, vol. 18, pp. 149–181, Jun. 2003.
[19] N. Kushmerick, "Wrapper induction for information extraction," University of Washington, 1997.
[20] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Extracting content structure for web pages based on visual representation," in *Proc. 5th Asia Pacific Web Conf.*, 2003, pp. 406–417.
[21] J. I. Fernández-Villamor, C. A. Iglesias, and M. Garijo, "First-order logic rule induction for information extraction in web resources," *Int. J. Artif. Intell. Tools*, vol. 21, no. 6, pp. 1250032-1–1250032-2, Dec. 2012.
[22] M.-G. Butuc, "Semantically enriching content using opencalais," in *Proc. Romanian Workshop Distributed Systems (EDITIA)*, Suceava, Romania, 2009, pp. 77–88.
[23] D. Huynh, S. Mazzocchi, and D. Karger, "Piggy bank: Experience the Semantic Web inside your web browser," *web Semantics*, vol. 5, no. 1, pp. 16–27, 2007.
[24] M. Toomim, S. M. Drucker, M. Dontcheva, A. Rahimi, B. Thomson, and J. A. Landay, "Attaching UI enhancements to websites with end users," in *Proc. Conf. Human Factors Comput. Syst.*, pp. 1859–1868, 2009.
[25] A. Hogue, "Thresher: Automating the unwrapping of semantic content from the World Wide Web," in *Proc. 14th Int. World Wide Web Conf.*, 2005, pp. 86–95.
[26] J. Wong and J. I. Hong, "Making mashups with marmite: Toward end-user programming for the web," in *Proc. Conf. Human Factors Comput. Syst.*, 2007, p. 1435.
[27] M. Bolin, M. Webber, P. Rha, T. Wilson, and R. C. Miller, "Automation and customization of rendered web pages," in *Proc. Symp. User Interface Softw. Technol.*, 2005, p. 163.
[28] A. Pan, J. Raposo, M. Álvarez, P. Montoto, V. Orjales, J. Hidalgo, L. Ardao, A. Molano, and A. Viña, "The Denodo data integration platform," in *Proc. 28th Int. Conf. Very Large Data Bases (VLDB)*, Aug. 2002, pp. 986–989.
[29] D. Connolly. (2007). *Gleaning resource descriptions from dialects of languages* [Online]. Available: http://www.w3.org/TR/grddl/
[30] M. J. Hadley. (2006). *web application description language* [Online]. Available: https://wadl.dev.java.net/wadl20061109.pdf
[31] A. P. Sheth, K. Gomadam, and J. Lathem, "SA-REST: Semantically interoperable and easier-to-use services and mashups," *IEEE Internet Comput.*, vol. 11, no. 6, pp. 91–94, Nov.–Dec. 2007.
[32] Wright State University. (2008). *HTML Microformat for Describing RESTful web Services and APIs* [Online]. Avialble: http://knoesis.wright.edu/research/srl/projects/hRESTs/#hRESTs
[33] Microformats community. (2008). *Microformats* [Online]. Available: http://microformats.org/
[34] M. Baker. (2005). *RDF Forms* [Online]. Available: http://www.markbaker.ca/2003/05/RDF-Forms/
[35] O. Etzioni, "Quagmire or gold mine?" *Commun. ACM*, vol. 39, no. 11, pp. 65–68, 1996.
[36] M. Pirker, M. Berger, and M. Watzke. (2004). An approach for FIPA agent service discovery in mobile ad hoc environments. *UbiAgents04* [Online]. Available: http://www.ift.ulaval.ca/~mellouli
[37] K. Jun, L. Bölöni, K. Palacz, and D. C. Marinescu, "Agent-based resource discovery," in *Proc. Heterogeneous Computing Workshop*, 2000, pp. 43–52.
[38] J. Cao, D. J. Kerbyson, and G. R. Nudd, "Use of agent-based service discovery for resource management in metacomputing environment," in *EuroPar 2001 Parallel Processing*, vol. 2150. Springer-Verlag, 2001, pp. 882–886.
[39] O. Ratsimor, D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service discovery in agent-based pervasive computing environments," *Mobile Netw. Applicat.*, vol. 9, no. 6, pp. 679–692.
[40] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, "Toward distributed service discovery in pervasive computing environments," *IEEE Trans. Mobile Comput.*, vol. 5, no. 2, pp. 97–112, Feb. 2006.
[41] Web Ontology Language and Semantic Web Services, "Dynamic discovery and coordination of agent-based Semantic web services," *IEEE Internet Comput.*, vol. 8, no. 3, pp. 66–73, Jun. 2004.
[42] S. Colucci, T. D. Noia, E. D. Sciascio, M. Francesco, M. Mongiello, G. Piscitelli, and G. Rossi, "An agency for semantic-based automatic discovery of web-services," in *Proc. Artificial Intelligence Applications Innovations (AI)*, 2004, pp. 315–328.
[43] N. S. Sidnal, R. S. Malashetty, and S. S. Manvi, "Service discovery using software agents in Semantic web," in *Proc. 11th Int. Conf. Control Automation Robotics Vision*, Dec. 2010, pp. 139–143.
[44] A. G. Neiat, M. Mohsenzadeh, R. Forsati, and A. M. Rahmani, "An agent-based Semantic web service discovery framework," in *Proc. 2009 Int. Conf. Comput. Model. Simulation*, Feb. 2009, pp. 194–198.
[45] O. Khriyenko and M. Nagy, "Semantic web-driven agent-based ecosystem for linked data and services," in *Proc. 3rd Int. Conf. Advanced Service Computing (Service Computation)*, 2011, pp. 110–117.

José Ignacio Fernández-Villamor received the Ph.D. degree in telecommunication engineering from the Universidad Politécnica de Madrid, Madrid, Spain, in 2012.

He has worked in the industry as a Technical Consultant and Open Source Software Developer. His current research interests include delivering information integration to the web. He has published several papers on topics such as information extraction, service discovery, and service modeling in RESTful architectures. He has taken part in several national and European projects and is currently involved in the European Project OMELETTE on web component discovery tasks.

Carlos A. Iglesias received the Ph.D. degree in telecommunications engineering from the Universidad Politécnica de Madrid, Madrid, Spain, in 1998.

He is an Associate Professor at the Universidad Politécnica de Madrid. His current research interests include multiagent systems, service engineering, and web engineering.

Mercedes Garijo received the M.D. and Ph.D. degrees in telecommunication engineering from the Universidad Politécnica de Madrid, Madrid, Spain, in 1982.

She is currently an Associate Professor at the School of Telecommunication Engineering, Universidad Politécnica de Madrid, where she instructs undergraduate, postgraduate, and doctorate students in computer science and communications engineering. She has been involved in several research and development projects funded by the EC and Spanish government. Her current research interests include telematic services engineering using techniques of software engineering and intelligent systems, especially ontologies, machine learning, and intelligent agents.