USING ONTOLOGIES TO PROSPECT OFFERS ON THE WEB

Rafael Cunha Cardoso, Fernando da Fonseca de Souza, Ana Carolina Salgado
Federal University of Pernambuco, C.P. 7851 CEP 50732-970, Recife, Brazil

Keywords: Semantic Web, Ontologies, Information retrieval/extraction systems, Web programming

Abstract: Nowadays, information retrieval and extraction systems perform an important role getting relevant information from the World Wide Web (WWW). Semantic Web, which can be seen as the Web's future, introduces a set of concepts and tools that are being used to insert "intelligence" into contents of the current WWW. Among such concepts, Ontologies play a fundamental role in this new environment. Through ontologies, software agents can cover the Web "understanding" its meaning in order to execute more complex and useful tasks. This work presents an architecture that uses Semantic Web concepts allied to Regular Expressions (regex) to develop a device that retrieves/extracts specific domain information from the Web (HTML documents). The prototype developed, based on this architecture, gets data about offers announced on supermarkets Web sites, using Ontologies and regex to achieve this goal.

1 INTRODUCTION

The World Wide Web appeared in the end of the eighties, time where the impact of this new technology on the society was unknown yet (Berners-Lee et al., 1994). The increasing use of the Internet has caused a permanent growth of the amount of data available on the Web. The intrinsic characteristics of the Web generate the need of specialized tools to achieve efficient management, qualified information retrieval and extraction procedures (Baeza-Yates & Ribeiro-Neto, 1999).

In this context, efforts point to the second generation of the Web, called Semantic Web (Berners-Lee et al., 2001). The Semantic Web can be thought as an extension of the current Web, where data receives a formal expression of its own meaning. The main goal of the Semantic Web is to insert some sense into WWW resources. This will enable that software agents "understand" and process Web contents, in a more intelligent way.

This work introduces an architecture that allies Semantic Web concepts (Ontologies, RDF) (Decker et al, 2002) with stabilized techniques (Regular Expressions) to execute information retrieval/extraction from the Web. To test such architecture, a study case, based on it, was developed. The prototype identifies and extracts relevant information from a specific knowledge domain (expressed by one ontology), structuring the extracted information in a friendly format.

The organization of this paper is as follows. Section 2 is dedicated to motivations of this work. Section 3 introduces the system's architecture and processes. Section 4 focuses the techniques that must be used to develop a prototype based on the architecture. The case study implementation is discussed in section 5. Section 6 presents related works and, section 7, concludes the work.

2 MOTIVATION

The Internet's explosion is responsible for the huge amount of data that increases daily on the Web. The immense quantity of data makes the Web the biggest repository ever seen. Most of these resources are spread on the Web without any worries about categorization rules or properties descriptions.

Such characteristics may cause troubles on the Web such as: delay in the information location; failure in finding the requested information due to URL (Universal Resource Locators) changes; retrieval of a raised number of unexpected resources due to ambiguity problems (Lopatenko, 2001).

In this situation, where unstructured data is easily found, tools specialized in information location become a challenge to the scientific community. The

Cunha Cardoso R., da Fonseca de Souza F. and Carolina Salgado A. (2005).
USING ONTOLOGIES TO PROSPECT OFFERS ON THE WEB.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 200-207
DOI: 10.5220/0002555902000207
Copyright © SciTePress

efficacy of such tools depends directly of the way the resources were described on the Web. In accordance with Moura (Moura, 2001), search tools are classified in two main classes:

·*Research in Directories:* Tools that were introduced when the content of the Web was small enough to be collected in a non-automatic way. The documents are classified manually according to a taxonomy.

·*Search Engines:* Tools that are worrier with its database's size. The gathering of documents in such systems is done by software agents, which cross the Web to collect data.

The aspects involved in these classes of tools, guide the development of mechanisms which tries to perform searches with meaning inlaid in the keywords. The problem is that such search tools do not consider the semantic aspects involved in the keywords that were submitted. They just analyze the words syntactically.

A complementary field to Information Retrieval is Information Extraction, which aims at extracting relevant information from semi-structured documents, and organizes it in a friendly format. NLP, Wrappers Development, Ontology-based methods, among others, are techniques that can be used to execute Information Extraction (Laender, 2002).

2.1 The Web

In the beginning of the nineties, the first efforts were done to develop the WWW as we know currently. Tim Berners Lee, the web's idealizer, had faced several challenges before its project was understood by the scientific community (Fernández, 2001). However, his efforts were rewarded, once the Web was consolidated as the mean of information distribution with the faster expansion in the worldwide history. Its intensive use allied to exponential growth provided a radical change in the life of the people who access the Web. On the other hand, this fast expansion turned the Web a content deposit as huge as disorganized. Such troubles generate constant disappointment to users with small experience on the Web, especially when they are involved with searches for specific information.

One of the factors that complicate this situation is the pattern language used to create Web pages, the HTML (Hypertext Markup Language) language. HTML does not specify any kind of semantic relative to the page's contents. It is just responsible for document's presentation. Consequently, it appears a gap, between the information available to Web services processing and the information that exists for people reading. The lack of meaning in Web documents caused the need of insertion of some "intelligence" to current WWW resources. The idea of inserting meaning into the Web documents can be summarized by one term: "Semantic Web" (Berners-Lee et al., 2001).

2.1.1 The Semantic Web

The Semantic Web enables the evolution from a Web composed by documents, to a Web formed by information where every data possesses a well defined meaning that can be interpreted, "understood" and processed by people and software cooperatively.

To understand the Semantic Web let's assume that someone is looking for pages about specie of bird, an eagle, for instance. Typing "eagle" in a search engine, several answers will be retrieved, getting besides the requested information about eagles, also pages about the "War Eagles Air Museum", or about the American football club "Philadelphia Eagles", among others results. This happens because the software just analyzes the word syntactically, do not discerning the football club, from the birds or from the museum. But, if these resources were marked up with a Semantic Web language, this would not occur, because the words could be distinguished semantically.

To develop the Semantic Web, an architecture formed for a set of layers was proposed by Tim Bernners-Lee (Berners-Lee et al, 2001). In this paper, the RDF and ontology layer are considered.

2.1.2 Ontology

Ontology is a term vastly known in areas such as Philosophy and Epistemology meaning respectively, a "subject's existence" and a "knowledge to know" (Chandrasekaran et al, 1999). Recently this term is being also used in the Artificial Intelligence (AI) to describe concepts and relationships used by agents. In the Database (DB) community, ontology is a partial specification of a domain, which expresses entities, relationships between these entities and integrity rules (Mello et al, 2000). From these definitions, an ontology can be defined as a conceptual data model that describes the structure of the data stored in a DB, in a high abstraction level.

Through these definitions an ontology will be able to provide a common/shared understanding about concepts on specific knowledge domains. This work uses a task ontology designed to aid the performance of the extraction process defined in the system architecture. Such architecture is introduced in section 4.

3 RELATED WORKS

In the literature several studies and works related to data retrieval and extraction from the Web can be found. The importance of this area is the possibility of manipulating the data as traditional data structures (e.g., relational databases) after the extraction process. There are some methods to perform the data extraction from Web documents: Natural Language Processing (NLP), Ontology-based, Machine-Learning techniques, among others. Some of this works are briefly commented next.

WebQL (Arocena & Mendelzon, 1998) consists in a language of declarative consultation capable of locating pieces of interesting data. Trying to find these data, an HTML-generic *wrapper* analyzes the page sent as data entry, and creates an *hypertree*, an abstract tree that represents the HTML document that is being analyzed. Using the syntax of the language it is possible to write queries that searches for data represented in the *hypertree*, and present the results in a tabular form.

WHISK is a tool that uses the NLP techniques for data extraction from text documents (Soderland, 2001). A set of extraction rules is induced from a collection of training documents. WHISK starts with an empty set of rules, and at each interaction, it selects and presents to the user a batch of instances to be tagged. An interface is present to users allowing them to add a tag for each attribute of interest from the instance. WHISK then uses the tagged instances to create rules and test the accuracy of the proposed rules.

ShopBot is another related work which functions with a set of URLs that addresses several on-line stores in the Web (Doorenbos et al, 1997). The ShopBot prototype operates in 2 phases; the first is the *Learning Phase*, where the ShopBot applies a learning algorithm in the sites that are listed in its register. ShopBot creates profile of each store present in its URL list. This stage of learning examines sites to “learn” a logical description of each site. The second phase is called *Comparison phase*, in which an assistant of purchases uses the profiles (acquired in the 1st phase) to help the user to carry through purchases in real time. The assistant uses the profiles to sail each site, following the appropriate structure, discovering the best price for a specific product desired by the user.

Another approach used is ontology-based. The Data Extraction Group at Brigham University (BYU) develops work in this area. In their tool, ontologies are previously constructed to describe the data of interest (Embley et al, 1999). By parsing a particular ontology the tool is able to produce a database by recognize and extract data present in pages given as input. To work properly this tool requires the careful build of an ontology, a job that must be done manually by a specialist in the domain.

Infomaster (Genesereth et al, 1997; Kosala & Blockeel, 1997) accesses information stored in databases or ACL (Agent Communication Language) knowledge bases. The first information available in Infomaster is about renting of houses and apartments in the San Francisco Bay area. The system extracts announcements from Web sites of several periodicals that present renting information in this region. These ads are separated in individual announcements, analyzed in a structured format and then loaded into a KIF knowledge base.

The system we developed uses practices adopted by some related works like Ontology, tree-parsing techniques and regular expressions. It uses the same premises of Infomaster and ShopBot, i.e., extracts information from web documents, using techniques like Ontology, tree-parsing analysis, and database storing.

4 ARCHITECTURE

The main goal of this work is to define an architecture to execute information retrieval and extraction, using an ontology-based model to assist this process. The architecture was organized a three-layer architecture:

·*Interface Layer:* associated to users browsers. Contains the Web interface, provided to get access to information extracted by the prototype;

·*Business Layer:* contains the modules related to information retrieval/extraction. The ontology is part of this layer too; and

·*Data Layer:* contains functionalities related to database tasks (storage and querying tasks).

Figure 1 shows the basic components that constitute the architecture, divided into its three layers. Such processes are: Resource Retrieval; Data analysis/extraction; Information Storing; and querying process.

4.1 Resource Retrieval

Software agents or *bots*, executes the first stage inside the context of the information extraction system (Heaton, 2002). In Figure 1, this step is represented by the “Resource Retrieval” and by phases A (sending the *bot* to the Web) and B (getting documents from the Web). After this, the gathered data is submitted to a parser analysis.

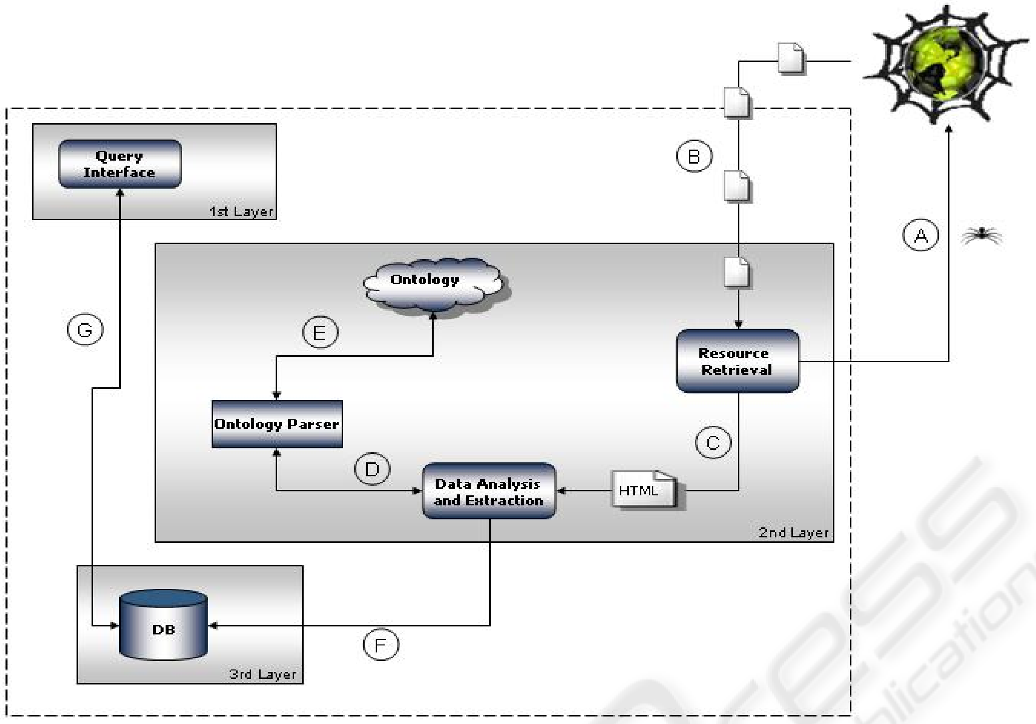


Figure 1: System's architecture

This process gets a set of pre-defined URL, connects to the Web and downloads all files from these sites. After retrieved, these resources are saved in a local folder, allowing starting the second stage performed by the architecture: Data Extraction.

4.2 Data Analysis/Extraction

After collected, the resources are sent to the *Data analysis/Extraction* module. It identifies and extracts relevant data from the HTML files. To analyze the files, it is necessary to provide a way that makes it possible to examine the resources properly. This is done via parsers that provide an intermediate representation of the files, allowing an adequate resource analysis.

Using these tools, it is possible to navigate the structure of the HTML/XML files. Navigating through this structure it is possible to get access to words or expressions that are part of the document for posterior comparisons with terms of a determined domain, described by the specific ontology. The architecture foresees the use of three API to parse three different formats of files: HTML, XML and specific ontology files.

This module must execute some steps in order to achieve the task of extracting information. Each of these steps are depicted in the next subsections.

4.2.1 Excluding unnecessary files

As the *bot* retrieves all files that are part from the site, the first step executed by this module consist in deleting data that can not be analyzed by the prototype. The system must examine the type of file it is dealing with, maintaining only HTML files to the next step.

4.2.2 Initial HTML parsing

This step aims at determining if the HTML retrieved by the *bot*, is according to the domain addressed by the system's ontology. This is done to avoid that pages out of context be deeply analyzed. To make this verification an HTML parser is used to access the HTML nodes that are part of the file. Navigating and accessing the nodes values, it is possible to make a *pattern matching* operation between the node value and the *context keywords* that are in the ontology. If any of these *keywords* is found in any HTML node, the file is sent to the next phase; otherwise the HTML file is deleted.

4.2.3 Creating a temp file

This step transposes the nodes that are part of the HTML files to a *XML temp file*. This is done to avoid problems that appear when we are dealing with HTML files.

The creation of the *XML temp file* is executed internally using the HTML parser to access the nodes and the XML parser to create the *XML temp file*. This temp file contains all data present un the HTML file.

4.2.4 Identifying and extracting data

This step is responsible for extracting the required information. To do this, the system accesses the ontology through a specific parser getting the regex(es) that was designed to extract data from the site that is currently analyzed. The regex is applied over the nodes of the *XML temp file*, recuperating the required information from it (if the regex finds it). Once identified, instances of the extracted data are inserted into a new XML file (called *XML final file*). Process D and E (figure 1) represent the extraction process done through all this steps, representing also the interactions executed with the ontology during the operations.

4.3 Storing and querying

After extracted, the information must be stored in a database (process represented by phase F). To achieve this, the *XML final file* is read through XML API and populates the relational DB that is used to store the extracted data. Once read, the XML data is transformed into a SQL statement, which populates de DB.

After populated, the DB is capable to receive query from users, through a Web interface, retrieving information about the domain that is modeled in the ontology. The querying process is done through a Web interface (Process G).

5 TECHNIQUES USED

To develop a prototype based on this architecture it is essential to use some techniques that permit to implement a study case.

5.1 Regular expressions

Once the architecture predicts to extract information from HTML files it is important to consider two aspects of this kind of file. First, they do not have any kind of semantics related to its TAGS. Second, the data presented by HTML files are written textually.

These two characteristics turn necessary to use techniques that allow identifying and extracting the required data from the files. Regular Expressions (*regex*) can be used to do this. Regex are a way to describe sets of strings, permitting to find relevant data in texts through *pattern matching* operations. Using *regex* it is possible to define expressions that represent a pattern that regularly appears in a text, allowing finding the required data. Regex performs an important role in this work, since they permit to recognize data from the HTML files analyzed. These regex must be putted in the ontology used by the prototype.

5.2 Ontology

A domain-specific ontology is used by the prototype to aid the identification/extraction processes from a web page. The ontology was defined through three basic steps:

a) Definition of the domain to be modeled;

b) Graph modeling, that reflects the environment addressed by the application;

c) Ontology creation, using a specific language to this kind of task.

The modeling through graphs is done because the project uses a Semantic Web language to define the domain ontology. Ontologies created from a Semantic Web view are modeled in graphs, since ontologies have structures of graphs, e.g., ontologies may be seen as sets of elements related to each other. This modeling allows a direct mapping between the modeling and the formal definition of a Semantic Web ontology.

5.3 Specific Parsers

The project uses three different kinds of parsers to manipulate appropriately the files handled by the application.

HTML pages are usually written without following rules established to XML documents creation. Because of this, XML and HTML files must be analyzed using different parsers. To allow the ontology's analysis, it is used Jena, a toolkit for specifically developing Semantic Web applications. Jena has an Ontology API that makes possible to navigate and to access the ontology elements.

6 STUDY CASE

Following the architecture it is possible to develop a study case. The study case extracts information about offers announced by a set of supermarket Web sites. Such offers are presented in different ways, varying from supermarket to supermarket. The announcements are done using several techniques since images, animations, or pure HTML. This study case is limited to HTML analysis, discarding other types of ads.

6.1 Specific Domain Ontology

The ontology was organized as a conceptual data model. This model was used to define the classes, subclasses, properties and instances of the ontology.

The domain ontology was written using the DAML+OIL language (McGuinness et al 2003).

The knowledge domain described by the ontology is the "supermarket offers" in Web sites. Besides modeling the environment, the ontology contains also *context words*. These *words* are expressions that characterize the domain. They will help to define if a page may have the required information or not. The ontology still is formed by a set of *regex*, which allow to identify and to extract the required information, i.e., promotional data, in this case. Each site must have at least one associated to it.

6.1.1 Ontology structure

The ontology was modeled containing five classes: Regex, Promocao (Promotion), Produto (Product), Termos_promocionais (Promotional terms) and Supermercado (Supermarket). The classes Promocao, Produto and Supermercado, are used to guide the storing of the data in the DB. The other two classes (Regex and Termos_promocionais) are responsible to maintain, respectively:

a) Instances of *context words* that will be used to determine the relevance (or not) of the documents that were downloaded by the prototype;
b) Instances of the regex related to each supermarket that later, will be used to identify the data that must be extracted.

The following code shows the definition of the Promocao class in DAML+OIL:

```
<daml:Class rdf:about="#promocao">
      <rdfs:label>promocao</rdfs:label>
   </daml:Class>
```

In DAML+OIL, properties are described separated from its class. There are two types of properties DAML+OIL. The first one is represented by `daml:DatatypeProperty` TAG. In this case, the property has as the "range" value, a datatype (for example string or decimal). The code below shows a definition of the property "temNomeSuper" (hasSuperName), which is related to the "super" class. The range tag determines which type of data this property accepts. In the example, it accepts the *string* datatype.

```
<daml:DatatypeProperty rdf:about="#nome">
     <rdfs:label>nome</rdfs:label>
     <rdfs:domain>
        <daml:Class rdf:about="#super"/>
     </rdfs:domain>
     <rdfs:range>
        <xsd:string/>
     </rdfs:range>
</daml:DatatypeProperty>
```

The relation between classes is made by properties `daml:ObjectProperty`. The definition of this kind of property is similar to Datatype Properties. The difference is the value of its "range" attribute that is a reference to another class, defined in the data model. Besides classes definitions there is also the definition of instances of specific classes, like regex and promotional terms, which will be used in the extraction process to discover the required information.

To allow access to data described in the ontology, it is used Jena, toolkit specifically developed to build Semantic Web applications.

6.2 Implementation Issues

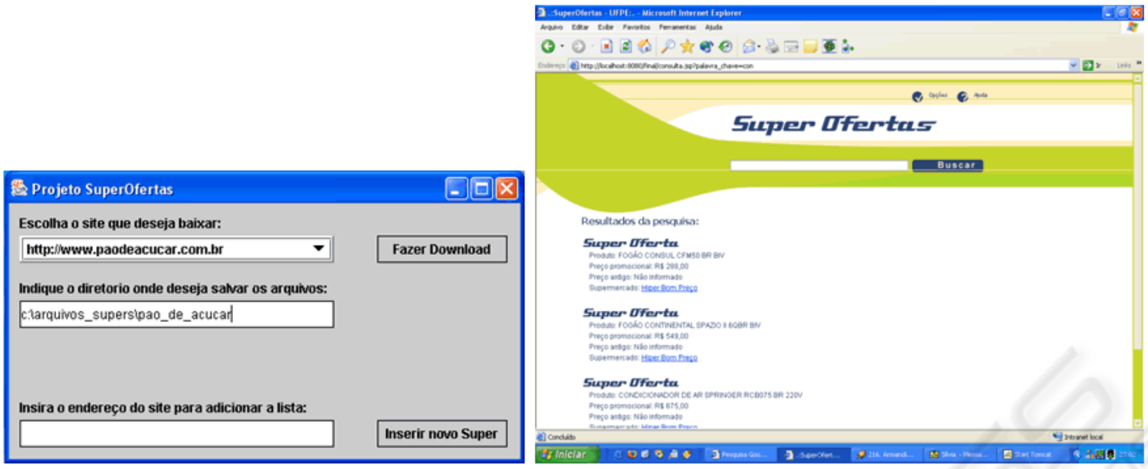
The Java language was used to build the prototype. The *Resource Retrieval* module was developed containing a set of URL of sites that can be visited by the *bot*. The *bot* connects these sites via HTTP (Hypertext Transfer Protocol) protocol, downloading all files that are part of the sites.

The second stage of the prototype performs *Data Analysis/Extraction*. To handle the files properly, the prototype uses specific parsing tools. The DOM model was proposed by the W3C (W3C, 2004) providing a standard way to access to data described in Web documents. This model uses a tree vision of the document that is being analyzed, supplying a way to navigate in all nodes that are part of the file.

Due to distinctions between XML and HTML files, two different API were used to get access to these formats of file. To analyze XML the prototype uses the JDOM (Java DOM) (McLaughlin, 2001), on the other hand, the HTML files are parsed using the NekoHTML parser (Clark, 2003), which besides providing a tree vision of the files it corrects some defects that usually occur in HTML files, as missing ending tags, for example.

When HTML pages arrive from a site the prototype the first thing the prototype does is determining if a page is relevant to it. Using NekoHTML parser each node is individually analyzed, trying to find any ontology *context word*. If the document has one of the terms expressed in the ontology (one instance of Termos_promocionais class), the HTML file is considered relevant, and otherwise it is discarded. If the document remains, a XML document is created, transposing all nodes that contain textual information in the HTML page (*XML temp file*).

This *XML temp file* passes to the next step of the extraction step. To do this, regex, present in the ontology (Regex class), are used. Regex supply a way to execute *pattern matching* over strings in an



Escolha o site que deseja baixar: http://www.paodeacucar.com.br

Indique o diretorio onde deseja salvar os arquivos: c:\arquivos_supers\pao_de_acucar

Insira o endereço do site para adicionar a lista:

Resultados da pesquisa:

Super Oferta
Produto: FOGÃO CONSUL CFM50 BR BIV
Preço promocional: R$ 288,00
Preço antigo: Não informado
Supermercado: Hiper Bom Preço

Super Oferta
Produto: FOGÃO CONTINENTAL SPAZIO II 6QBR BIV
Preço promocional: R$ 549,00
Preço antigo: Não informado
Supermercado: Hiper Bom Preço

Super Oferta
Produto: CONDICIONADOR DE AR SPRINGER RCB075 BR 220V
Preço promocional: R$ 675,00
Preço antigo: Não informado
Supermercado: Hiper Bom Preço

Figure 2: Administration tool and the Web interface, developed in the project.

intelligent way. Each site has one or more regex related to it. Such regex identifies the offers presented for the exact site. The specific regex are applied over all nodes of the XML files trying to recognize the data, which must be extracted. Once identified, the data is extracted by the regex. The information is then structured in a new XML file, called of *XML final file*. All this process is controlled by an administrative tool that permits to insert new supermarket's URL to be analyzed by the prototype. Besides, it allows managing all flux of files and processes that are being executed currently.

After extracted, the relevant data is stored in a database. MySQL SGDB (MySQL, 2003) is used by the project to store the data extracted from the files. The routines that allow inserting data into the DB reads the data described in the *XML final file*, and create SQL statements that insert the information into the DB.

After populated, the DB is ready to be queried by users through the Web interface. This interface was developed using JSP (Java Server Pages) technology (JSP, 2004). The interface retrieves the offers extracted by the prototype, about products the user is looking for, bringing all information available in the DB, in a friendly way.

Figure 2 presents both, the administrative tool and the Web Interface, used by the prototype to achieve the information extraction goal.

7 CONCLUSIONS

The Semantic Web is one of the W3C's long-term objectives. It is being developed in an environment of intelligent access to heterogeneous and distributed information.

This work, developed at UFPE, proposes a way of using some techniques related to Semantic Web (RDF, Ontologies) allied to techniques that allows syntactic analysis (Regular Expressions) to solve problems related to information extraction on the Web. The work introduces a general architecture that can be used to extract information about any domain, since a specific ontology be created to this.

To test such architecture, a study case was developed. The knowledge domain chosen was about offers in the Web. The prototype's intention is to get accurate information about products that are sold in supermarkets Web sites. This goal was reached through the development of SuperOfertas (SuperOffers). SuperOfertas (Interface in figure 2) retrieves to users

The prototype that avoids tedious tasks that users would do manually in the Web to find the information he looks for.

REFERENCES

Arocena, G. O.; Mendelzon, A. O., 1998. WebOQL: Restructuring documents, databases and webs. *In: Proceedings of the 14th International Conference on Data Engineering (Orlandos, FL, 1998), pp. 24-33.*

Baeza-Yates, R., Ribeiro-Neto, B., 1999. *Modern Information Retrieval*. ACM Press, Nova York.

Berners-Lee, T.; Cailliau, A.; Luotenem, A.; Nielsen, H. F.; Secret, A.. 1994. The World Wide Web. *In Communication of the ACM, 37(8): 76-82.* ACM Press.

Berners-Lee, T.; Hendler, J.; Lassila, O., 2001. *The Semantic Web.* Internet: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (March, 2003).

Chandrasekaran, B.; Josephson, J. R.; Benjamins, V. R., 1999. What Are Ontologies, and Why Do We Need

Them? *In IEEE Intelligent Systems. 1999. Magazine. pp 20 – 26.*

Clark, A., 2003. *CYBERNEKO HTML PARSER.* Internet: http://www.apache.org/~andyc/neko/doc/html/index.html (October 2003).

Decker, S.; Mitra, P.; Melnik, S., 2002. *Framework for the Semantic Web: An RDF Tutorial*. Internet: http://www.computer.org/internet/v5n6/ic-rdf.pdf (February, 2003).

Doorenbos, R. B.; Etzioni, O.; Daniel S. Weld, D. S., 1997. A Scalable Comparison-Shopping Agent for the World Wide Web. *In: Proceedings of the First International Conference on Autonomous Agents 1997.*

Embley, D. W.; Campbell, D. M.; Jiang, Y.S.; Liddle, S.W.; Kai Ng, U.; Quass, D.; Smith, R. D., 1999. Conceptual-model-based data extraction from multiple-record Web pages. *Data and Knowledge Engineering 31, 3 (1999), 227-251.*

Genesereth, M. R.; Keller, A. M.; Duschka, O. M., 1997 Infomaster: An information integration system. *In Proceedings of the ACM SIGMOD Conference, May 1997.*

Heaton, J., 2002. *Programming Spiders, Bots and Aggregators in Java*. Sybex Inx, Alameda.

JSP, 2004. JSP: Java Server Pages Technology. Internet: http://java.sun.com/products/jsp/ (January 2004).

Kosala, R.; Blockeel, H., 1997. Web Mining Research: A Survey. *In Proceedings of the ACM SIGMOD Conference, May 1997.*

Laender, A.H.F; Ribeiro-Neto B.A.; da Silva, A.S.; Teixeira J.S., 2002. A brief survey of web data extraction tools. *SIGMOD Record, 31(2): 84–93.*

Lopatenko, A. S., 2001. *Information Retrieval in Current Research Information Systems.* Internet: http://semannot2001.aifb.uni-karlsruhe.de/positionpapers/Lopatenko.pdf (July 2003).

McGuinness, D. L.; Fikes, R.; Hendler, J.; Stein, L. A., *DAML+OIL: An Ontology Language for the Semantic Web.* Internet: http://dsonline.computer.org/0211/f/x5mcg.pdf (October 2003).

McLaughlin B., 2001. *Java & XML, Second Edition.* O´Reilly, 2001, 528p.

Moura, A. M. C., 2001. *A Web Semântica: Fundamentos e Tecnologias.* Internet: http://www.udabol.edu.bo/biblioteca/congresos/cicc/cicc2001/datos/Tutoriales/Tutorial4/T4.pdf (June 2003).

MySQL, 2003. *MySQL: The World's Most Popular Open Source Database.* Internet: http://www.mysql.com (October, 2003).

Soderland, S., 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning 34, 1-3 (1999), 233-272.*

W3C, 2003. *W3C: World Wide Web Consortium.* Internet: http://www.w3c.org (October, 2003).