A Wrapper&Mediator Prototype for Web Data Warehouses

Verónica Giaudrone vgiaudrone@adinet.com.uy
Marcelo Guerra mguerra@adinet.com.uy
Marcelo Vaccaro mvaccaro@adinet.com.y

Regina Motz rmotz@fing.edu.uy
Adriana Marotta amarotta@fing.edu.uy

Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Julio Herrera y Reissig 565, 5to piso
11300 Montevideo, Uruguay
Fax: (+598) (2) 7110469

Abstract

There is a lot of information published on the Web that can be useful for decision-making. The work reported in this paper focuses on how to extract and integrate this information in order to construct a Data Warehouse that makes it available. The manual process of extracting and integrating information is expensive and complex. That is the reason why we suggest the development of a tool, based on Wrappers and Mediators, which allows the extraction of information from the Web and integrate it automatically. Wrappers are in charge of information extraction, which is based on page's structure and a query enriched using a domain ontology. Mediators perform data integration in order to combine information from several sources, solving the conflicts that may appear due to contradictory information, taking into account the trust of the sources. An important characteristic we consider in our proposal is that information contained in the Web changes constantly; therefore a mechanism that supports system evolution becomes essential. For this reason we propose the generation of metadata that keeps the traceability of the process, allowing managing the impact of the source changes on the whole system.

Keywords: Wrapper, Mediator, Ontology

Workshop: Workshop de Ingeniería de Software y Bases de Datos

1. Introduction

Nowadays, there is a need for looking up information in the Web and we must inevitably use a search engine, such as Google. Therefore, we enter some key words, and as a result we obtain a group of Web pages that, we expect, contain the information we are looking for.

Generally, the result does not meet our requirements, because it contains a lot of pages that include the words required, but we have to analyze them to find information useful to us. The result is usually characterized for containing a large number of Web pages, most of which do not contain all the information we requested. There are also inconsistencies that we have to evaluate from different sources. It is becoming more and more necessary to obtain the information we need in the most exact and concrete way as possible, in order to use them for decision making.

As a solution it was proposed an extraction and integration architecture based on Wrappers and Mediators techniques that allow the load of a Data Warehouse with information from the Web. In this work we present the framework that we developed to achieve this end.

One of the most relevant aspects of our proposal is the intensive use of web technologies for the extraction and integration of data. In particular, Ontologies are used for domain specification, description of relevant data to be extracted for web pages and resolution of integration heterogeneities.

Moreover, information that is published on the Web changes constantly, so there is a great need for managing information evolution within the system, changes in information sources must impact in the resulting information. Considering this, metadata is generated to keep process traceability. To this end, our framework follows the architecture proposed in [1].

Figure 1 shows the complete framework to construct a web data warehouse. However, for space reasons, this paper focuses on the initial process of extraction from web pages and data integration. Documents on the remainder process can be found in the web page of the project http://www.fing.edu.uy/inco/grupos/csi/Proyectos/WDW.htm

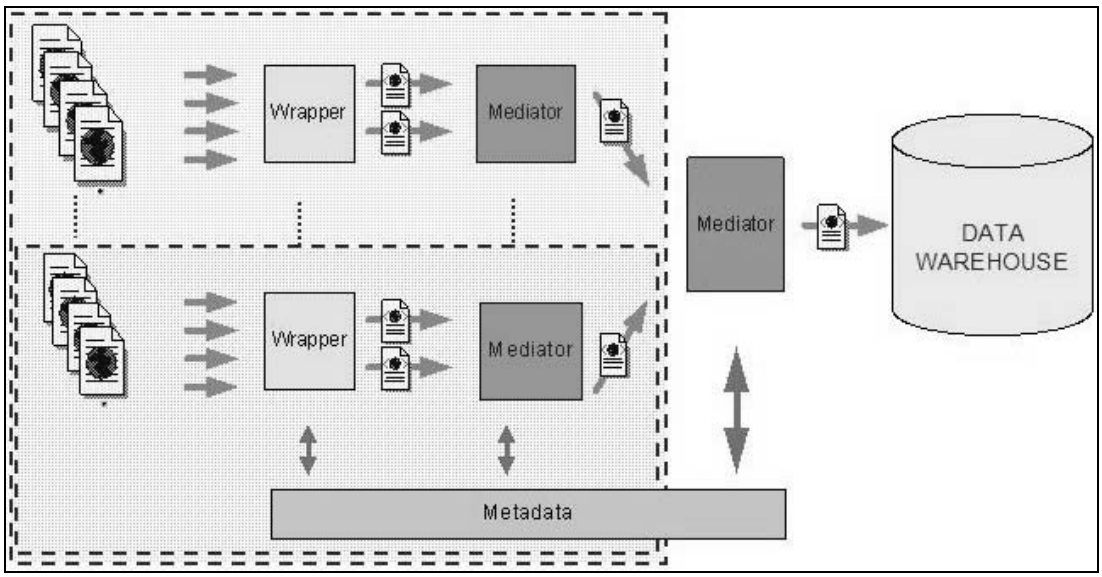


Figure 1 - System architecture.

It can be noted that, architecture remains the same for every group of pages, so our system implements it once and can be instantiated for each group. This paper contributes with the

specification an implementation of a wrapper that can be instantiated with different domain descriptions. Thi spaper also presents the specification and implementation of a mediator that resolves data conflicts. Our development reached certain quality level, following guidelines stated in [2].

The remainder of the paper is organized as follows. In Section 2 we describe our system, including requirements and test case characteristics. In section 3 we explain the main information extraction process and in Section 4 we explain the main data integration process. In section 5 we comment our developing strategy. Finally, in Section 7 we expose our conclusions and future work.

2. System Description

The system presented in this paper is part of another one that covers the whole process, which starts at the sources and ends at the DW. The present system implements the extraction and data- integration modules. In Figure 1 we show the architecture of the whole system and we mark with dotted lines the part that corresponds to the here-described system. The complete architecture presents two kinds of mediators. The first ones only perform the data integration, since the input schemas are identical. The second one performs data and schema integration.

The present system was divided in three components: classification module, extraction module and mediator. The implemented system has two possible kinds of users: one that uses the system previously configured for a particular domain, and other, that may be the system administrator, who manages XML associated technologies (like X-Path, etc.) and ontologies, and is capable of configuring the system for different domains.

The functionalities provided by the system are the following:

- Web pages classification. The system selects Web pages that are relevant for the DW information requirements.

- Information extraction from a Web page. The system selects relevant information from each classified page, returning a XML document, based on a given schema (which was derived from the DW information requirements).

- Data integration. The information extraction module returns a document for each relevant Web page, and all of them are based on the same XML Schema. Data integration receives these documents and returns one document with the requested information, after solving possible data conflicts.

- Metadata generation for evolution support. The system generates metadata in order to register how the information selection process was done (traceability). This is useful for solving how changes on the pages affect the extracted information (propagation of changes sources-mediator).

We selected a particular domain to support the tool development, which also was used for test cases. This domain determined the inputs for the system: Web pages, ontology and XML Schema.

The test case that oriented the development was about movies information. We used movies and actors as related domains, each of which is processed independently by the extractor and by the data integration mediator.

We chose 11 Web pages from a search in Google about movies, including Uruguayan cinema's pages. Some pages had information about a particular movie and others about many movies that were being shown. We especially looked for pages about "Harry Potter and the prisoner of Azkaban" and "Troy".

We use domain ontologies for enriching the vocabulary from the information request with domain knowledge. We did not find an ontology that met our requirements, so we constructed one with the necessary information, using OWL language. Concepts were declared as classes under the namespace movies ("películas" in Spanish), we defined equivalent and different concepts as equivalent and disjoint classes, a similar thing was done for instances. We used synonyms as equivalent concepts. Our tool does not consider hierarchies, but it could be extended to use them. We defined movie names in the ontology as instances of title, because we could not find a standard way for title presentation on the Web pages, so we could not identify them if they were not in the ontology. A Web page or database could have been defined as reference for obtaining new movie titles, or regularly update the ontology.

The schema for the information request can include the element to look for, in this case "movie", and as many attributes as needed. Figure 2 shows the used schema. A required attribute is used as key during the integration process, in this case "title". We specified a data type for each attribute, to check the extracted information domain.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
      attributeFormDefault="unqualified">
  - <xs:element name="pelicula">
    - <xs:complexType>
        <xs:attribute name="título" type="xs:string" use="required" />
        <xs:attribute name="título_original" type="xs:string" />
        <xs:attribute name="país" type="xs:string" />
        <xs:attribute name="año" type="xs:gYear" />
        <xs:attribute name="duración" type="xs:duration" />
        <xs:attribute name="género" type="xs:string" />
        <xs:attribute name="productora" type="xs:string" />
        <xs:attribute name="audiencia" type="xs:string" />
        <xs:attribute name="idioma" type="xs:string" />
        <xs:attribute name="recaudación_total" type="xs:long" />
        <xs:attribute name="actores" type="xs:string" />
        <xs:attribute name="directores" type="xs:string" />
        <xs:attribute name="productores" type="xs:string" />
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

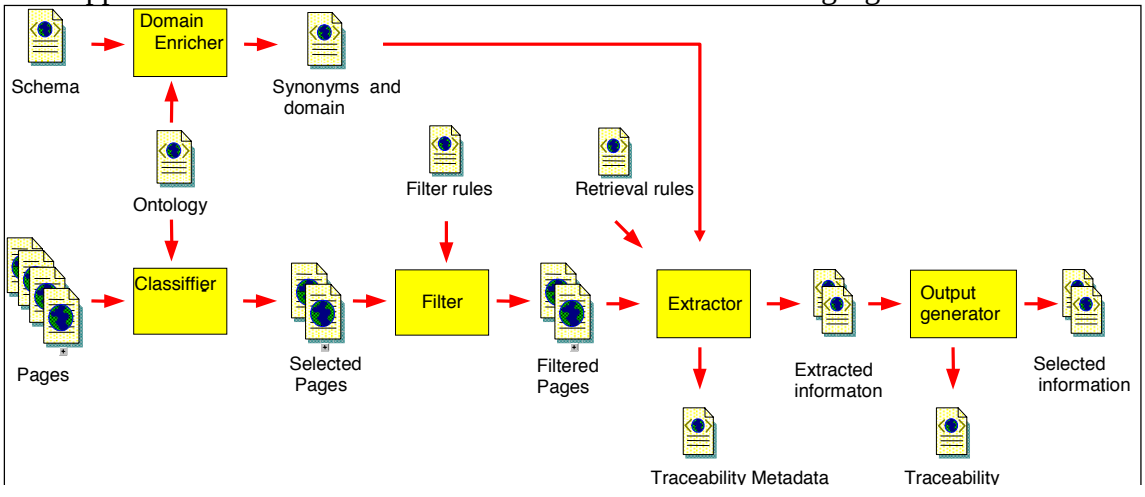
Figure 2 - XML Schema for the information request

3 A Wrapper for Information Retrieval

An important part of the developed system consists on information extraction from Web pages based on a request. There is no standard criterion among Web designers on how to display information and therefore there are many ways of doing it.

This section shows how the system deals with this problem and tries to obtain the information. Information search will be oriented by the schema and domain knowledge provided by an ontology. Heuristics will determine where to find the data to extract.

The wrapper consists of five main modules shown in the following figure:



Domain enricher: The information request presented as a XML Schema enters the wrapper using particular terms that we call "representative", this term is selected by who generated the schema. However, the way in which this information will appear does not necessarily include that representative; it may contain some variant of it (like a synonym). In order to enrich this request we propose two variants of it.
The first one consists on using synonyms of the concepts (movie → film), the second on using instances of the concepts (movie → Harry Potter).

Classifier: Classification consists on identifying relevant pages from a set. Web page classification is the subject of previous projects like the one we used [3].

Filter: Once we have selected relevant pages, we have to deal with the fact that web pages are not necessarily XML files, for example, not all tags opened are closed. To solve this problem we used [4]. Moreover there is information in the pages that our system does not use (like images). Filter module removes this information.

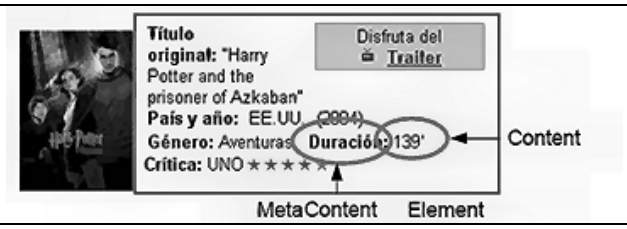
Extractor: There are various approaches for Web pages information retrieval. One approach is to use natural language based techniques. Another approach is to use page's structure. A third alternative consists on combining both previous techniques, the second to locate the information and the first to extract the information. This project implements an Extractor based on the second alternative. Retrieval algorithm evaluates rules for each element of information requested in order to locate it in the page. These rules are called "Retrieval Rules" and are based on the following concepts:

Element: Represents an element in XML Schema entered as information request to the system. (i.e.: film, actor, etc.)

Meta-content: Represents an attribute in XML Schema. Its location indicates the near presence of the information requested. (i.e.: in "year: 2005", "year: " indicates that the year is the following value)

Content: Represents the value with which the schema attribute will be filled. (i.e.: in "year: 2005", "2005" is the content).

We developed four rules in order to generate a prototype. Those rules are described according to the following schema:

Structure:	Paragraphs whitch contain Metacontent: Content, inside an HTML structure like <p>, , etc. {`<?> Metacontent: Content </?>`}
Element:	Cell in a table que the information is located. {`//td/descendant::text()[contains(.,'%synonym%')]/ancestor::td[1]`}
Meta-Content:	HTML tag where information is located. {`./descendant::text()[contains(.,'%synonym%')]`}
Content:	Same place as meta-content. {`.`}
Example:	
Structure:	Information in table with titkes. {`<td> Metacontent </td> <td> Content </td>`}
Element:	Each table row. {`//td/descendant-or-self::*[contains(text(),'%synonym%')]/ancestor::table[1]/descendant::tr[count(preceding-sibling::*)>%fila%][count(td)>1]`}
Meta-Content:	Table header. {`./ancestor::table/descendant-or-self::*/text()[.='%synonym%']`}
Content:	Cell in the corresponding column and row in the table (Evaluated from the element). {`./descendant::td[count(preceding-sibling::td)=%columna%]/descendant-or-self::text()[1]`}
Variables:	*Fila* = {`count(//td/descendant-or-self::*[contains(text(),'%synonym%')]/ ancestor::tr/preceding-sibling::tr)`} *Columna* = {`count(//td/descendant-or-self::*[contains(text(),'%synonym%')]/ ancestor-or-self::td[1]/preceding-sibling::td)`}
Example:	

Problems encountered:

What happens when no meta-content can be found (ie: 

)

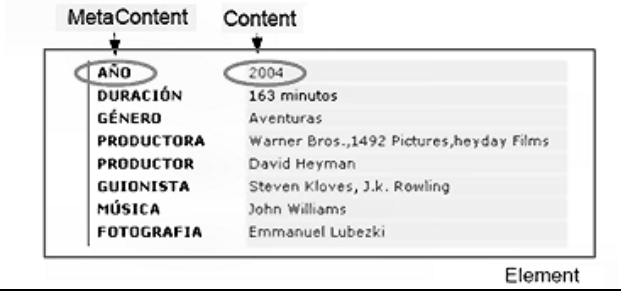
This problem has two approaches: (i) It is known that this is the title of a movie, or (ii) Context indicates that that is the title of a movie.

Analyzing several cases we were unable to find a pattern so that we had to state that is the title of a movie.

What happens if information is within the same page but in different structures? We decided to state that if the information coming from two different structures within the same page is complementary then this information is united in one unique element.

Limitations: Preceding information implies that information presented in ways not supported by some rule will not be extracted.

Examples of these situations are: Information in paragraphs in sentences, Information not contained in the page (i.e.: contained in a page linked by the original), Information contained within an image.

Structure:	Information contained in a table where content and metacontent are in contiguous cells {`<td> Metacontent </td> <td> Content </td>`}
Element:	Table where the information is contained. {`//td/descendant-or-self::*/text()[contains(.,'%synonym%')]/ancestor::table[1]`}
Meta-Content:	Cell where the attribute synonym is located {`./descendant-or-self::*/text()[contains(.,'%synonym%')]`}
Content:	Cell next to the metacontent-cell {`./ancestor::td[1]/following-sibling::*/descendant-or-self::text()[1]`}
Example:	

The Wrapper Metadata consists on:
Rules schema: this file defines the structure a rule must follow, each rule in the system must follow this schema.

Rules: this file includes the definition of the rules the system currently uses, if new rules are required in the system, they should be included in this file. A same od rule specification is shown in Image 3.

```
<element location="//td/descendant::text()[contains(.,'%synonym%')]/ancestor::td[1]" >
  <attributes metaContentLocation="./descendant::text()[contains(.,'%synonym%')]" contentLocation="." />
</element>
```

Image 3 – Part of rule 1 XML specification

Used rules: shows the rules applied to a particular page and the result of this execution. Image 4 shows a sample of the used rules file.

There are also other files which help in the system execution (like filter definition, instanciated rules, etc.)

```
<elementRule location="/*[01]/*[02]/*[04]/*[01]/*[01]/*[01]/*[01]/*[01]/*[01]/*[02]/*[01]/*[01]/*[01]"
  aceptance="A0" friendlyLocation="//td/descendant::text()[contains(.,"%synonym%")]/ancestor::td[1]">
  <attributes metaContentLocation="./descendant::text()[contains(.,"%synonym%")]" contentLocation=".">
    <attribute name="título" type="xs:string" required="true'>
      <metaContent synonym="harry potter y el prisionero de azkaban" isInstance="true" />
      <content result="harry potter y el prisionero de azkaban" />
    </attribute>
    <attribute name="título_original" type='xs:string" required="false">
      <metaContent synonym="harry potter and the prisoner of azkaban" isInstance="true" />
      <content result="harry potter and the prisoner of azkaban" />
    </attribute>
    <attribute name="país" type="xs:string" required="false">
      <metaContent synonym="ee.uu" isInstance="true' />
      <content result="ee.uu" />
    </attribute>
    <attribute name="duración" type="xs:duration" required="false">
      <metaContent synonym="duracion" isInstance="false" />
      <content result="duracion:139' |" />
    </attribute>
```

Image 4 – Used Rules

5. A Mediator for Data Integration

For designing the architecture we considered the objective of the Mediator as well as its inputs and outputs. Its inputs are the outputs of the Extractor and the ontology that gives useful information for data integration to distinguish between similar terms.

Mediator receives many initial XML files (XSD instances) from different web pages and must integrate them returning only one instance with the required information.

This module also generates metadata which is stored in XML files like in the Extractor module.

The Image 5 shows the modules that conform the basic architecture of the Mediator.

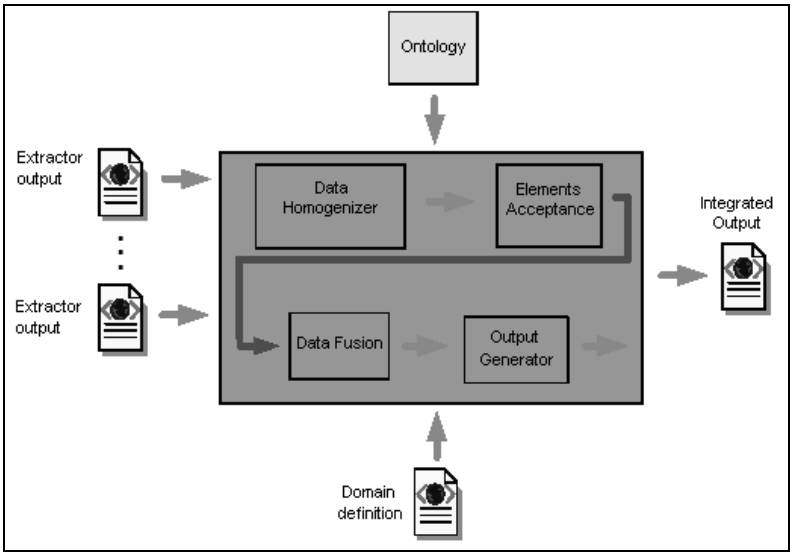


Image 5 – Mediator Architecture

Data Cleaner and Homogenizer Module: This module is in charge of preparing the data extracted for its integration, receiving the output of the extractor and domain definition as inputs and generating files with the homogenized information and necessary metadata as output. Domain

definition is combined with extractor output to apply some cleaning to the text and types checking to delete irrelevant information. I.e.: if in the country field contains “the country is EE.UU” we would rather have only “EE.UU”. Also we would like to be able to unify terms like “United States”, “EE.UU” and “USA”, and choose one of them like the representative of the group. The metadata indicates how the process was done and shows what cleaning occurred as shown in Image 6.

```
<root>
   <element name="pelicula">
         <attribute name="género" value="aventura" valueInstance="aventuras" originalValue="#aventuras# "/>
         <attribute name="país" value="ee.uu" valueInstance="estados unidos" originalValue=" #estados unidos# "/>
         <attribute name="título" value="harry potter y el prisionero de azkaban" valueInstance=" harry potter y el
prisionero de azkaban " originalValue="# harry potter y el prisionero de azkaban #"/>
      </element>
</root>
```

Image 6 – Cleaner and Homogenizer metadata.

Elements Acceptance Module: This module filters non-valid elements as a result of the cleaning and homogenizing process. At this point, it is necessary to make an element acceptance processing, because after the cleaning some attributes that had a valid value, now have an empty value (“”). An important issue for integration is that certain required values specified in the original scheme, can not have empty values so this process is required.

Data Fusion Module: This module is in charge of the integration, at the data level, of the different homogenized outputs, and returns one integrated output data and the metadata. For the integration process some kind of priority mechanism is necessary to determine which of the attribute value options will be chosen according to the selection criteria. In this case, we used the source trust criteria, where the value chosen is the value from the source with the greatest trust, but it could be extended with new custom integration criteria. The elements that have the same value for the required attribute are integrated. The integration process fills the other attributes solving possible conflicts using the selected integration criteria. The metadata shows for each attribute of each element, the value resulting of the integration, the source from where the data was obtained with its trust and the list of all possible value options. An example of this metadata is shown in Image 7.

```
<element name="pelicula">
- <attribute name="título" value="harry potter y el prisionero de azkaban" source="Cartelera Montevideo Shopping.htm"
trust="0.8">
   <options value="harry potter y el prisionero de azkaban" source="ADICTOSALCINE_COM - Harry Potter.htm" trust="0.1" />
   <options value="harry potter y el prisionero de azkaban" source="Cartelera Montevideo Shopping.htm" trust="0.8" />
   <options value="harry potter y el prisionero de azkaban" source="UruguayTotal Cartelera - Harry Potter y el prisionero de
Azkaban.htm" trust="0.7" />
   <options value="harry potter y el prisionero de azkaban" source="UruguayTotal Cartelera de Cine x Películas.htm" trust="0.6"
/>
   <options value="harry potter y el prisionero de azkaban" source="Yahoo! Cine - Harry Potter y el prisionero de Azkaban.htm"
trust="0.5" />
   </attribute>
- <attribute name="país" value="ee.uu" source="UruguayTotal Cartelera - Harry Potter y el prisionero de Azkaban.htm"
trust="0.7">
   <options value="ee.uu" source="ADICTOSALCINE_COM - Harry Potter.htm" trust="0.1" />
   <options value="ee.uu" source="UruguayTotal Cartelera - Harry Potter y el prisionero de Azkaban.htm" trust="0.7" />
   <options value="ee.uu" source="Yahoo! Cine - Harry Potter y el prisionero de Azkaban.htm" trust="0.5" />
   </attribute>
- <attribute name="género" value="aventura" source="Cartelera Montevideo Shopping.htm" trust="0.8">
   <options value="aventura" source="ADICTOSALCINE_COM - Harry Potter.htm" trust="0.1" />
   <options value="aventura" source="Cartelera Montevideo Shopping.htm" trust="0.8" />
   <options value="aventura" source="UruguayTotal Cartelera - Harry Potter y el prisionero de Azkaban.htm" trust="0.7" />
   <options value="aventura" source="UruguayTotal Cartelera de Cine x Películas.htm" trust="0.6" />
   <options value="aventura" source="Yahoo! Cine - Harry Potter y el prisionero de Azkaban.htm" trust="0.5" />
   </attribute>
</element>
```

Image 7 – Integrator Metadata

Finally, the output generator filters the metadata returning a file without metadata that match the format specified in the request information scheme.

6. Quality Properties of the Development

Our goal of development was the complete knowledge of system faults and being able to reproduce them (automated testing helped to improve performance of regression tests). To achieve this end we calculated for each test case the expected output, for automating testing, because analyzing each output and comparing it with the web page would take too much effort.

With this automation we built a test module entrusted with comparing the obtained output with the expected output.

This module produced a XML file indicating for each requested information item, the information that should have been found, the information that was actually found, an the information accuracy ratio and a result ratio.

Accuracy: is calculated as “the number of found characters in the output” divided by ”the number of expected characters”.

Result: is a qualitative ratio showing if was obtained extra information, missing information, not found information, correct information or incorrect information

```
<TestResult timeStamp="1100356298063" accuracy="0.8509804" result="NOT BAD">
  - <element name="pelicula" found="YES" accuracy="0.8509804" result="NOT BAD">
      <attribute name="título" expected="harry potter" found="harry potter" accuracy="1.0" result="OK" />
      <attribute name="país" expected="ee.uu" found="ee.uu de américa" accuracy="0.3125" result="EXTRA INFO" />
      <attribute name="actores" expected="D. radcliffe" found="radcliffe" accuracy="0.75" result="MISSING INFO" />
      <attribute name="año" expected="2004" found="" accuracy="0.0" result="NOT FOUND" />
      <attribute name="idioma" expected="" found="español" accuracy="0.0" result="WRONG INFO" />
  </element>
</TestResult>
```

The accuracy attribute in the element “pelicula” represents the average of accuracies of all its element attributes.

The accuracy attribute in the root element represents the average of accuracies of all elements on the test.

The extraction results are shown in Image 8. We measured the effectiveness of the extraction rules comparing the percentage of extracted information to the percentage of existing information in the Web pages we selected from four domains (Actors, Movies, Hospitals and Physicians).

We found more than 50% of the expected information in more than 90% of the pages.

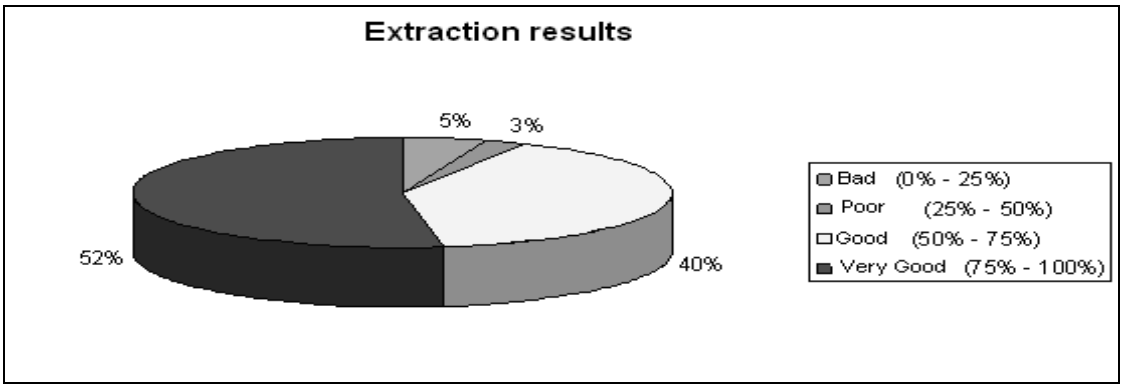


Image 8 – Extraction results

7. Conclusions and Future Work

We studied technologies and implemented a prototype covering the extraction and integration of information from Web pages. Our solution proposes mechanisms for doing the extraction of information from Web pages, based on pages structure, obtaining as much as information as possible considering its presentation. Mechanisms for doing data integration from the extracted information, using a criterion that chooses the best value considering options from different sources.

However there are a lot of work on wrappers and mediators we are not interested in the presentation of related work but in the presentation of a functional prototype that can be consulting in http://www.fing.edu.uy/inco/grupos/csi/Proyectos/WDW.htm

In the construction of this prototype we propose a set of information extraction rules (considering a set of Web pages) and XPath usage as language for expressing information location in rules. Moreover, we specify metadata generation allowing extensibility and evolution support.

Information extraction rules show good results over the pages we used. The extracted information allows us to do data integration. As there are many ways of presenting information on the Web, structures different from the ones we consider will need new rules to be created, to be able to extract information.

Integration process will make the user classify sources and assign them adequate trust values. It has to be taken into account that, all the information that is extracted from the most trustable source will be included in the result.

As this project will be continued we reach a quality level, and we emphasize our strategy of increasing test cases as well as increasing system functionality. We also want to highlight that, we created extraction rules using the first domain selected, and then modify them for the second one, but we did not change them for the third and fourth domain used. In spite of developing using a particular domain, the prototype should not lose generality.

We are planning as future work the following directions:

- Information extraction can be complemented using natural languages techniques, for example: extract paragraphs using pages structure approach and then process them using natural language recognition and differentiating data types.
- Create new criteria for solving conflict during data integration. For example: a criterion that returns the data that is proposed by the majority of the sources.
- Implement an automated mechanism for impacting source changes on the result, supported by the generated metadata.

8. References

[1] Marotta, R. Motz, R. Ruggia.Managing Source Schema Evolution in Web Warehouses International Workshop on Information Integration on the Web (WIIW '2001). Rio de Janeiro, Brazil. 2001

[2] Diego Vallespir – Master thesis: Actividades para mejorar la calidad Instituto de computación – Facultad de Ingeniería – Universidad de la República, Montevideo Uruguay http://www.fing.edu.uy/~dvallesp/Tesis/webActividades/index.htm Last visit: November 21, 2004

[3] Álvaro Fernández. Master thesis: Un Generador de Instancias de Wrappers Instituto de computación – Facultad de Ingeniería – Universidad de la República, Montevideo Uruguay, 2004.

[4] HTML TIDY http://www.w3.org/People/Raggett/tidy/ Last visit: November 21, 2004.

[5] FOLDC:http://www.si.umich.edu/UMDL/glossary.html Last visit: December 04, 2004.