Supervised Data Extraction

N. Georgiev, J.M. Labat, Jean-Luc Minel, L. Nicolas

Supervised Data Extraction

N.Georgiev[1], J.M.Labat [2], J.L.Minel[3], L.Nicolas[4]

[1]CRIP5, Nielsen // NetRatings
Nikolay.Georgiev@netvalue.fr
[2]CRIP5, Université Paris 5
Jean-Marc.Labat@math-info.univ-paris5.fr
[3]LaLICC, UMR 8139 CNRS - Université Paris 4
Jean-Luc.Minel@paris4.sorbonne.fr
[4]Nielsen // NetRatings
67, rue Anatole France
92 300 Levallois-Perret, France
Laurent.Nicolas@netvalue.fr

Abstract. The process of **data extraction** from internet sources have been originating the interest of the scientific society for the past years. However there are still no well established standards because of the heterogeneous nature of the information in the Global Network. Nevertheless there is still something in common – all the data is available in HTML format for compatibility reasons. This article presents our methodology and the prototype system we've created to extract data from HTML pages. We use **XPath** as data extraction language and have developed a methodology for **visual wrapper generation**. Our approach takes advantage of the implicit correlation between the data and the surrounding structure. Some evaluation tests are given also in order justify our methods.

1. Introduction

Several years after the birth of the World Wide Web 80 % of the information presently available in the internet is in the form of HTML documents. Nevertheless there is still no well established standards for inter-computer communication thought the global network and the majority of the data shared is accessible only in a human readable form. That's why the initial design of the Web has become obsolete very fast. The human capacity of processing the information had been rapidly overtaken by the growing volume of the shared information in the Net. Even that the search engines covers only the uppermost layer of the Web (*see DeepWeb [3]*) and thus some 2-3 billions documents this still represents a giant quantity of data which is not manageable by an unarmed man.

A lot of people see now the ultimate solution in the migration to the semantic web[2]. Some promises had been made that the web will become in both user and computer readable form. However the technologies around the semantic web are still in the research area and will not be finalized as fast as we need them. Meanwhile the web still improves it's current technologies giving more access of program based

agents to the resources shared world wide. The first step in this evolution are the XML technologies[27]. Direct examples is the migration to XHTML/CSS and the WS[1].

As this process is about to take a while we need to deal with the web as it is today. That's why the need of "Data Extraction Systems" [5,6] compatible with the current form of the web is very present. Having the possibility to query the web as we do with the relational data bases will have an unlimited application.

Unfortunately the information in the web documents is not structured. Even that there exists formal structural languages like the HTML, the data it-self is not semantically structured in the absence of some Data Description Record (DTD). This means that the structure, if any is there only to organize visually the data. Indeed that was the basic idea behind the HTML language. It defines the formatting of the data but does not give us meta-information about its meaning. That's why the information is readable only by a human.

There exists yet some computer based techniques [9, 10] starting from simple pattern matching, lexical analyzers up to some complex semantic analyzers capable of identifying concepts and retrieving information from text. Some approaches based on auto-learning non-deterministic finite-state automata (*like those using hidden Markov's chains/models*), can give good-to-average results also. However all those methods are rather complex and still not a hundred percents trustable. They do not take advantage of the semi-structured aspect of the HTML pages neither.

Today there is still no well established model for direct retrieval of data stored in the form of HTML or XHTML documents. Meanwhile the most of the web pages have gone dynamic. This means that the data inside had already been structured in some kind of database behind the scenes, so that the HTML pages have been automatically generated from it. This is a very strong argument why we should seriously take in consideration the structure of the pages when doing the data extraction.

In this paper we are about to present one approach of data extraction from HTML / XHTML pages along with its implementation and will justify our method with some real world tests.

We had turned our research towards an approach which consists of finding the relation between the visual formatting of the data and it's semantic signification in order to take use of the HTML structure to extract the needed data. Our implementation represents a semi-structured data extraction prototype with supervised rules creation features. Enabled with an ontology based knowledge base with auto learning capabilities for assisted data recognition and rules creation. It takes advantage of the page structure to create extraction rules and store them in a wrapper[2] model [7,10] (*think for a wrapper as a "template" to apply*) capable of extracting the same type of data from "*common type*" of pages.

This research have been hold for the needs of the Nielsen // NetRatings company. A leading American company in the internet measuring business. The methodologies and the prototype are still in research phase.

---

[1] **WS** – Web Services

[2] **Wrapper** - Procedure to transform one type of data to another

2. Requirements

The need of the company is to take usage of some data present in the HTML documents but not taken in count until now because of lack of technologies. We are intending to develop this technology in order to be able to offer a richer gamut of products. A good application example is the retrieving of the data found in the electronic transactions going through the global network. This includes shopping charts, banking accounts, products information and so on. This gives the immediate possibility of identifying order confirmations, retrieving the amounts spent and calculating the BPI[3], for example.

NetRatings has well established technologies and some specific needs and constraints which applies to our research project as well.

The company dispose of hundreds of thousands RDD[4] panels and is present in fourteen countries, covering over 90% of the internet universe. Some 2 milliards of displayed pages for month are referenced in our databases. Our platform parses about 100 milliards tags of meta information each month.

A particular constraint in this environment is the scalability of the system. That's why some complex and no linear algorithms lacking of performance are not suitable for use here. We need an approach capable of processing 5 to10 million pages per month, as this is the estimated number of pages to handle each month. They are selected mainly on a domain basis by applying some URL filters, but also by taking into account some keywords, where available. The criteria of the selection depends on the current survey.

The higher reliability is also a requirement. The retrieved data will be used behind to calculate some statistics, where the main condition is the accuracy of the extracted data. We cannot allow to introduce any noise along the extraction process.

A clear advantage in here is the quality of the source documents. We are looking for to extract data from professional sites and not from personal ones. This automatically faces us towards some automatically generated HTML pages, often based on XHTML/CSS technologies. In this type of pages the internal structure and attributes are in strong semantic relation with the displayed data. This facilitate the data extraction process and opens the possibility of generating the extraction wrapper from a visual interface without the need of advanced skills in the Internet/XML technologies. This an important requirement also – the simplicity of usage and therefore the reduced maintenance and support costs.

3. Similar Works

There exists already some commercial information extraction systems[4, 18-21]. Non of them however has an extensible architecture capable to interface with our platforms. Their main drawback is the lack of performance, often related to it's non

[3] **BPI** - Buyer Power Index

[4] **RDD -** Random Digit Dialing**,** a selection of telephone numbers where the digits in the numbers are picked by chance, often by a computer.

determinist algorithms. Even that there are some solutions that can respond to our expectations, they showed to use some very complex data extraction languages that were not suitable for automatic wrapper generation. None of them had an appropriate GUI for constructing the wrappers by simple visual interactions or was not robust enough to ensure the data retrieval even when the contents has changed. One of the famous toolkits for generating wrappers which we were capable of testing are the: "W4F" from Tropea Inc., "X-Fetch Wrapper" from Republica, "WebDataKit" from Loton Tech, "Visual Web Task" from Lencom Software, "RoboSuit" from Karpow Technologies, "TextPipe" from Crystal Sftware, "Lixto Wrapper" from Lixto, "WebL" from Compaq and "XWrap".

They all use some formal script language to describe the extraction process. They are named as follows: W4F – **HEL (HTML Extraction Language)**, X-Fetch – **DEL (Data Extraction Language)**, WebDataKit – **SQL for HTML**, RoboSuit – **RegEx**, **TextEx**, TextPipe – **Perl, VBScript, Jscript, RegEx, Python**, Lixto – **ELOG** (**Declarative extraction language**), WebL – **WebL** (**Compaq's Web Language**, based on RegEx and Perl).

In comparison with these our HTML Wrapper uses the **XPath** as Data Extraction Language.

4. The Data Extraction Approach

After an extended research in the domain of the data extraction techniques from semi-structured sources finally we've chosen to base our approach on some standard XML technologies. This ensures, we believe, the future compatibility of the system with the fast growing on technological level web.

In general our method consists of parsing the page and constructing a DOM[5] [26] from it. After that we are using the Xpath[6] [24, 25] language to create extraction rules based on visual markers, attributes and structure patterns. Finally a wrapper is created and exported in XML [27] or XSLT [28] format.

The key point of the approach is the construction of robust XPath expressions for intra-document referencing[15, 16, 17]. Actually our idea was to focus on local and relative expressions in order to ensure a maximum robustness trying at the same to keep the precision and the recall high.

To accomplish our task we have developed a tool called **HTML Wrapper**. It is composed by three main modules:

The first is the **data extraction** module which is a server application that receives in a batch some pre-selected web pages, decrypts them and tries to apply different wrappers in order to extract the searched information. Before the extraction can take place the pages are sorted by domain and stored in a queue. For each domain there is a limited number of wrappers to test. The number of wrappers to check against the page is further limited by presetting routine based on session details, url

[5] **DOM** - Document Object Model, W3C standard for representing an XML document
[6] **XPath** - XML Query Language mainly used by the XSLT

particularities, keywords and meta-data found in the page. We are also looking at the possibility to use some LSA[7] or Bayesian's classification algorithms here.

After the page has been decrypted and acknowledged it is passed through the extraction engine which tries to retrieve the data from it using the correct wrapper and then stores the results using the specified in the wrapper database mapping. If the extraction process is incomplete or fails for some reason, the analyzed page is transferred to a second queue which is the entry point to the second module.

This second module, called **visual wrapper creation module** is charged to fine-tune the wrapper for the current page or to crate a completely new one. This process goes through an operator validation, as we need a very high precision in the data extraction. In fact the module is a client-server application with a graphical interface where the analyzed pages are displayed. The system then tries to give a proposition about were the relevant data is expected to be. This estimation is based over the closest matching wrapper if any, over some reattachments algorithms[15] or is given by the third module.

This third module is called a **proposition module** and is the most advanced part the system. It is capable to calculate an approximation and point out where the relevant data is located. This is possible due to a kind of "knowledge base" learned from examples. The process of learning is actually happening when the operator points out the data in the graphical interface. The system recognizes the structural specificities (*page structure, attributes, elements etc.*) and assigns them to a thematic classification for this kind of data. We can think of it as an domain ontology where each class has some structure patterns (*think of them as micro-wrappers*) associated. The ontology defines also the scope of application [11,12,13] for these micro wrappers. Once the page is recognized to correspond to certain category (*match a given domain ontology*), the proposition module tries to guess which parts of the page contain a relevant data in the context of the current domain ontology. This is obtained by matching these micro wrappers. We called them "**information couples**" as it will be explained further.

Using the Xpath [25] syntax we can construct different type of expressions to point to data nodes in the HTML document. The XPath however works only with well formatted XML documents. So an HTML aware DOM[5] parser is needed to process the document in background [22, 23]. There are several examples out there, but the best choice is to use the one that comes with the browser that has visualized the page. This will ensure that the structure will be kept the same, as every HTML parser deal with the badly formatted HTML using it's own logic which normally gives slightly different results between the browsers.

We've adopted an hybrid approach here. It takes use of the real browser, accessing directly it's DOM[5] in the memory and retrieving the parsed document while still displayed. This ensures the accuracy of the data and the correct interpretation of some badly written html at the same time. The displayed page is immediately retrieved from the target browser which guarantees that the analyzed page will be the same as the one that had been displayed by the user. In contrast with a crawler based approaches, where the page is retrieved later by it's url and does not guarantee to be the same, especially when there are sessions restrictions applied.

[7] **LSA** - Latent semantic analysis

In fact we retrieve the serialized version of the displayed page, already parsed by the browser. This solves one major problem - the proper interpretation of the real world html. This is possible because our client side panel technology called NetMeter can access the loaded in the browser document directly. Additionally we use a third party HTML DOM[5] Parser to build back the DOM[5] of the page from it's serialized copy. Then we can further normalize and tidy the differences in the structures between the different browsers if needed.

Once the page had been normalized and a DOM[5] had been constructed we can start applying our extraction expressions in order to access the data in the document. Using the XPath language we can then address the nodes with a given type and attributes, the ones which are found in a particular structure or which matches a given pattern[1]. Thanks to the flexible syntax of the Xpath we can define different type of expressions for different type of data to extract.

We've tried to define one common way of rules creation adapted to work fine for some general and most frequent case[1]. Further the process of rule creation can be customized through some additional settings and parameters.

5. Semi-automatic supervised rules creation

One of the strongest sides of our system is the assisted process of the wrapper creation. It has been designed with the idea to be used by any general user. That's why the creation of the wrapper can be very easy and intuitive. All it have to do the operator is to point-and-click the desired data and to associate it with some category by drag-and-drop. The remaining is managed by the system behind the scenes. The ending result is the creation of a wrapper for the given type of page.

The creation logic is based on several concepts. These are "**Extraction Model**", "**Information Couple**" and "**Local Structure**". We will try explain further the exact place of each one in our approach.

The "**Extraction Model**" is the model from which the Wrapper is generated using different methodologies. One particular methodology is proposed here[1]. The model is created in memory as a result of the user manipulation in the graphical interface. It is a tree structure model which can be looked at as a simpler representation of the DOM of the page analyzed, containing only the required nodes, parameters and paths. In fact it is a object model which contains all the variables needed to access the selected data in the DOM of the page and to construct extraction rules according to different methodologies. The model is composed by several type of objects. The main container is the wrapper object which contains objects of type "local structure", "record", "information couple", "keyword", "data" and some others. This model can be serialized in XML or exported to XSLT and is stored back to a database.

The "**Information Couple**" is the smallest entity in our model. It represents a type of data found in the web page. The couple contains all the needed parameters to identify and to access to the data node in the DOM of the page, as well as the mapping back to the database. It keeps information like the type of the data node, some structure relations or/and attributes as well as some visual markers like strings (*keywords, links, etc..*) or graphical elements (*separators, images, etc..*).

Several couples can be regrouped in an "record" object to form a data record. This object is optional and is present only when there is a list of items to extract. By default all the couples and the records (*if any*) are regrouped in an object called "**Local Structure**". This is done mainly to restrain the scope of application of the local extraction expressions. Notice the local expressions are more robust than the global ones but had the disadvantage to bring out some noise. By limiting their application to the "**Local Structure**" we eliminate successfully any possible ambiguity all in keeping the robust aspect of the local expressions.

The couples contains also some mapping information about where the data goes after the extraction as well as it's cardinality. The cardinality specifies if the couple is mandatory (*1..1*) or not (*0..1*), but it gives also it's multiplicity i.e. is it a single result (*1..1*) or a list of results (*1..5, 1..N*).

The couple is therefore a set of variables retrieved from the DOM by the rules creation module as result of the operator's manipulations, at one hand. But also the result of the association (*categorization*) of the data to some predefined domain ontology at other hand.

All this constructions are expressed in syntax XPath. For example the following expressions gives all the data contained in the 3-th column of the table called "devis", excluding the first line (*the header in this case*).

Example:

```
table[@name='devis']/td[position()>1 and last()]/td[3]
```

The next expression for instance, looks for all the text nodes which contains the string '€' and returning everything before the currency (*the price in this case*).

Example:

```
substring-before(/descendant::text()
                 [normalize-space()]
                 [contains(.,'€')][2],'€')
```

To make the things more comprehensible we will proceed with the following example. Consider the following domain ontology:

Fig. 1. Products Order Domain Ontology

This domain ontology defines the searched data for a given type of page[11,12,13]. In this case it's a general product's order. This data can be found in some

confirmation page often at the end of the user session. The scheme shows that for a web page of type “product order” we are looking to extract at least one product name but also some other useful data if available.

The system reads the ontology and creates couples for each ontology class with the given cardinality. After that it is up to the user to make the relation between the newly created couples and the real data on the web page in order for the system to generate the wrapper for the specified type of page.

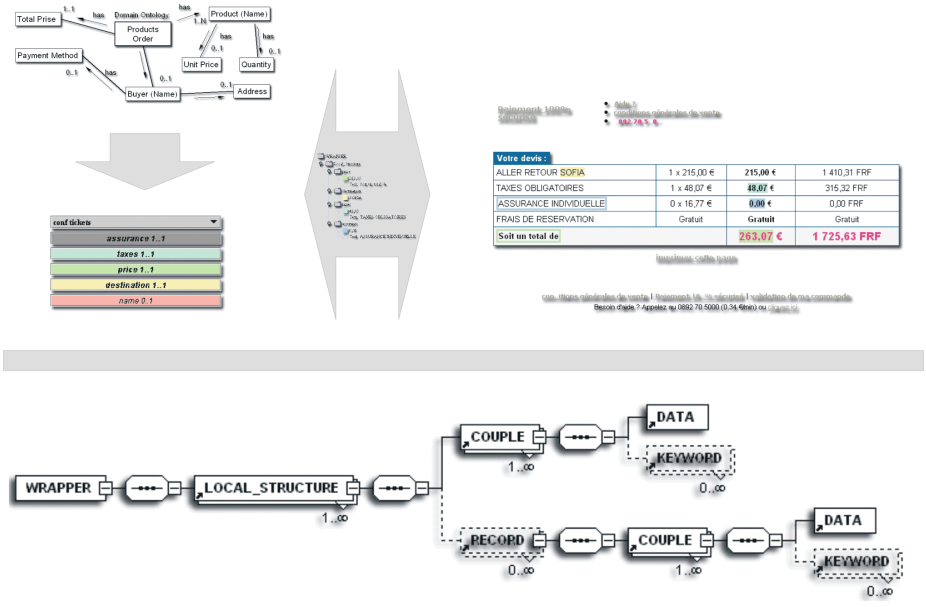

Fig. 2. Functional Scheme of HTML Wrapper

Let’s see now how the wrapper is functioning.

As we’ve mentioned earlier that the wrapper is first created in the memory and then serialized in the form of XML or XSLT. The XML version can be executed from the “Extraction Module” and the XSLT version can be executed using any XSL Processor. For the moment the XML version does provide full functionality and the XSLT is only provided as a portable export format. However the interesting thing in here are the extraction expressions generated for each data to extract. These are constructed from the wrapper by merging the local extraction expressions of the objects : “Wrapper”, “Local Strcuture”, “Couple”, “Keyword” and “Data”. The process is based on the following scenario:

1. The client connects to the server and loads the next page in the graphical interface.
2. The user selects a domain ontology to work with (*this will be further automated*).
3. The system generates the appropriate “information couples” and show them in the client’s interface in the form of colored buttons.

4. The users toggles a couple button and then points to the data in the web page to make the relation.
5. Then the user selects eventually some visual elements found to be descriptive for the data in question (*some labels, links, separators, images and so on*).
6. The system constructs the extraction model as the user adds more and more data. A visual representation of the model is shown at the clients interface.

After the validation of all the selections the wrapper is generated by the system and is stored back to the server to be available to the extraction module. The wrapper is generated taking into account different settings established previously. These defines the rules creation methodology. The selection of the methodology depends on the page structure and it's configuration is intended to happen automatically according to the acquired in the "KB" data patterns for previous examples.

6. Rules creation methodologies

Two main approaches have been taken and tested against real world pages to distinguish their qualities and drawbacks. The first one is a structure approach where structure patterns are recognized, stored in the wrapper and then used to retrieve the data. The second one is an enhanced version where in addition to the structure patterns we will take into account the contents of the "anchor" nodes as well.

We will try in our example to retrieve the payment details of the **amazon.com** electronic basket (*fig.3.*). Using the first methodology the wrapper have generated the following extraction expressions for retrieving the **"Items", "Shipping & Handling**" and "**Order Total**" entities:

Fig. 3. Tableau HTML

For the Local Structure :

```
//table[1] [tr[1][td[1] and td[2]] and
           tr[2][td[1] and td[2]] and
           tr[7]/b ]
```

For Items:

```
couple:   /tr[1][td[1] and td[2]]
data:     /td[2]/text()[normalize-space()]
```

Shipping & Handling:

```
Couple:   /tr[2][td[1] and td[2]]
data:     /td[2]/text()[normalize-space()]
```

Order Total:

```
couple: /tr[7]/b
data:   /text()[normalize-space()]
```

These "local expressions" will try to match a particular structure and extract some data form it. We call this method "**Structure Based**", as he is based completely on a structure pattern recognition. If we take a closer look at them we will notice their recursive nature. They are in fact composed by nested local expressions, where each is evaluated in the context of the others. That's because only one local expression is not specific enough to be evaluated in the document context on it's own. We can easily see here the reflection of the wrapper structure, the "local structure" from one hand and the couple (*data* + *keyword*) and the relative expression leading to the data at the other.

The second methodology of constructing the extraction expressions gives the following expressions for the same example:

For the Local Structure :

```
//table[tr[td[1][contains(text(),'Items')] and td[2]] and
       tr[td[1][contains(text(),'Shipping')] and
       td[2]]]
```

For Items:

```
couple:   /tr[td[contains(text(),'Items')] and td[2]]
data:     /td[2]/text()[normalize-space()]
```

Shipping & Handling:

```
couple:   /tr[td[contains(text(),'Shipping')] and td[2]]
data:     /td[2]
```

Order Total:

```
couple: /tr[img[@src='#pixel-grey.gif')] and tr[8]/b]
data:     /tr[8]/b
```

Here again we have the local structure, the couple and the data expressions merged together, but we do have also some content and attribute constraints which ensures the precision of the expression, especially when there are a few couples in the local structure. We call this approach "**Couple Based**". This gives the possibility to define local structures with only one couple, if necessary and solves some conceptual problems related to the previous approach.

One main disadvantage of the structure approach is that it's based on absolute positions in the local structure and in examples like the pervious one, where the structure components (*tags*) are repeating, it's impossible to detect the introduction of a new record without considering some other parameters.

An enhanced version called "**Loose Couple**" gives higher rate of flexibility. It uses the axes of the XPath to address relatively the data from it's anchor node. Such an expression have the following form:

For the Local Structure:

```
//table[contains(text(),'Items:') and
        contains(text(),'Shipping & Handling:') and
        contains(@src,'#pixel-grey.gif')]
```

For the couple:

```
//tr/td/img[@src='pixel-grey.gif')]
```

For the data:

```
/following::tr[1]/b/text()[normalize-space()]
```

Once again the three parts are the Local Structure Expression; the Anchor Expression of the couple; and the Data Expression of the couple; Merging all the three gives the extraction expression for the given data.

Different extraction expression constructions are possible depending on the factual structure of the page. It's necessary to be able to customize the expressions to match the particularities of the different type of pages. This can be done from the main rules construction interface.

Our solution is completely open and compatible with the upcoming XHTML/CSS techniques. The idea behind the CSS was to separate the formatting from the data, so that a properly constructed page will contain only semantically structured tags like div, span, ul, li, em, strong etc., with the given associated "classes" and "id" attributes. This will facilitate even more the extraction process using our HTML Wrapper and may make to disappear the need of the "Local Structure" concept, therefore simplifying even more the extraction model.

7. Evaluation tests

The three methods have been tested against several types of HTML pages coming from different domains. The most interesting was the amazon.com as their web pages are very rich and complex and so this was an excellent try for the system.

First we have created two wrappers to extract data from two type of Amazon's pages. The first results were based over **1254** web pages from the secured Amazon's sessions. This is an extract for one day activity of the US Panel for the 20 Oct 2003.

By applying some preliminary filters we preselect the suitable pages for the given wrapper. These filters are based on URL patterns and meta-data information found in the pages. An important criteria is the title of the page. By taking in consideration some keywords found in the title we can limit further the pool of pages down to the relevant ones. This is especially true in the case with the **amazon.com** pages.

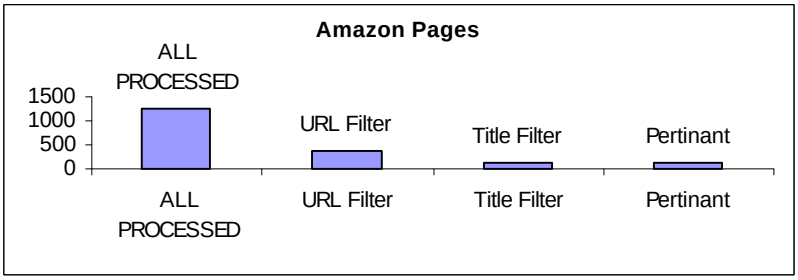

Fig. 4. HTML Pages for Amazon.com, processed, filtered and relevant

Generally the "**Structure Based**" approach is likely to introduce more noise in the results and so it's precision is not usually good. The recall is also not very high. Contrary to this, our "**Couple**" approaches, seams to respond better to our needs. They are able to give high precision, i.e. not introducing false results, but also a hundred percent of recall (*able to retrieve the data form all the pertinent pages*) in 60 % to 100 % of the time (*different pages*) depending on the selected approach (*couple or loose couple*).

The graphic (*fig.5*) shows the results of applying the structure approach in four data extraction tasks, charged to extract data from the **amazon.com** order confirmation pages. In the first and second wrapper we've tried to extract the payment details form Fig.*4*. We should remind here that the structure of this table changes between the html pages, some fields change position, some may disappear and some may be added. After that we've tried in the third and the fourth wrappers to extract the "Name" and the "Address" of the client from the same HTML page (*not shown here*). We needed two wrappers because there were two main type of pages available. As we can see in fig. 5 the structure approach gives a lot of noise and does not manages to extract the data from the all relevant pages.

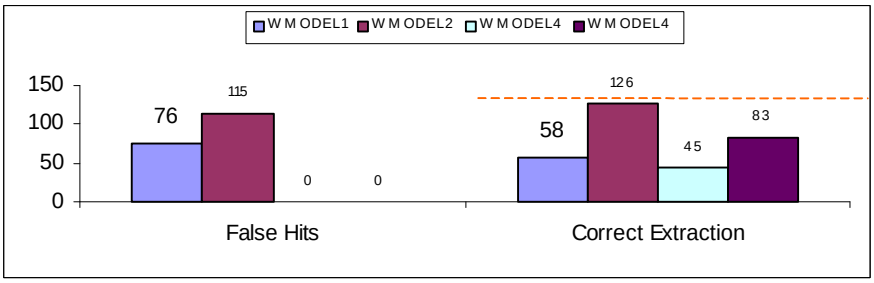

Fig. 5. *Structure approach, False Hits and Correct Extraction, the Max correct extraction is 128 (Relevant Documents)*

When looking further we can see that for the same pages the wrappers created using the "**Couple Based**" approach gives rather promising results. The wrappers gave zero false hits and the first and the second wrappers gave a hundred percent precision (*fig 6*). After passing to the "**Loose Couple**" approach the results were even better. All the four extraction tasks gave a hundred percent precision as well as hundred percents of recall. (*not shown here*) .

The approach was justified.

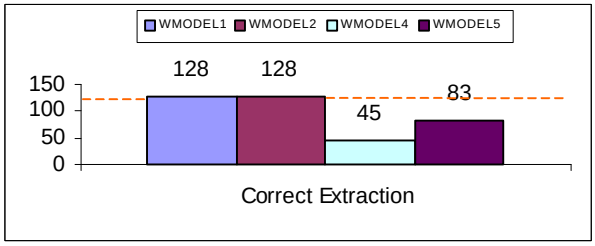

Fig. 6. *Couple approach, Correct Extraction, the Max correct extraction is 128 (Relevant Documents)*

8. Future Work

Finally we can enjoy a basic prototype implementation which justifies our expectations. Some satisfactory preliminary result were shown here. Our approach based on "**couple of information**" seams to be reliable and robust enough.

A lot of work to implement all the scheduled functionality is still ahead. As soon as our wrapper becomes stable, and starts filling the database with precious meta-information, we will hurry up to implement our proposition module. Then some extensible tests will be made in order to evaluate the supposed ability of the approach to discover relevant data in unknown type of pages.

9. Références:

[1] Extraction de données à partir de pages HTML par création semi-automatique de règles XSLT, N.Georgiev, J.M.Labat, J.L.Minel, L.Nicolas, 2002

[2] Laublet P., Reynaud C., Charlet J., Sur Quelques Aspects du Web Sémantique, Assises 2002 GdR I3, Nancy http://sis.univ-tln.fr/gdri3/

[3] M. K. Bergman, Deep Web : http://www.press.umich.edu/jep/07-01/bergman.html

[4] Toolkits for Generating Wrappers : A survey of Software for Automated Data Extraction from Web Sites, Stefan Kuhlins and Ross Tredwell, University of Mannheim, 2002

[5] Alberto H. F. Laender, Berthier A. RibeiroNeto, A Brief Survey of Web Data Extraction Tools, CiteSeer 2002, http://citeseer.nj.nec.com/laender02brief.html

[6] Line Kikvil, Information Extraction from World Wide Web : A Survey, Norwegian Computing Center, Oslo July 1999, http://citeseer.nj.nec.com/eikvil99information.html

[7] Naveen Ashish and Craig Knoblock, Semi-automatic Wrapper Generation for Internet Information Sources

[8] Appelt, D. E. and Israel, D. J. 1999. Introduction to Information Extraction Technology. In Proceedings of the 16th International doint Conference on Artificial Intelligence.

[9] Raymond Kosala, Jan Van den Bussche, Maurice Bruynooghe, Hendrik Blockeel, Information Extraction in Structured Documents using Tree Automata Induction, 2002, http://citeseer.nj.nec.com/kosala02information.html

[10] Kushmerick, N.: Wrapper Induction for Information Extraction, Dissertation 1997, Dept of Computer Science & Engineering, Univ. of Washington, Tech. Report UW-CSE-97-11-04, Oct. 2002, http://www.cs.ucd.ie/staff/nick/home/research/download/kushmerickphd.ps.gz

[11] D.W.Embley, D.M.Campbell, A Conceptual-Modeling Approach to Extracting Data from the Web, CiteSeer 1998, http://citeseer.nj.nec.com/24307.html

[12] H. Snoussi, L.Magnin, J.Y.Nie, Heterogeneous Web Data Extraction using Ontology, CiteSeer, http://citeseer.nj.nec.com/550153.html

[13] David W. Embley, Douglas M. Campbell, Randy D. Smithn Stephen W. Liddle, Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents

[14] Chee-Yonh Chan, Pascal Felber, Minos Garofalakis, Rajeev Rastogi, Bell Laboratories, Luncent Technologies, Efficient Filtering of XML Documents with XPath Expressions

[15] Thomas A. Phelps, Roberts Wilensky, Robust Intra-document Locations, University of California, Berkeley

[16] A.J.Bernheim Brush, David Bargeron, Anoop Gupta, JJ Cadiz, Robust Annotation Positioning in Digital Documents, Technical Report MSR-TR-2000-95, Sep 22, 2000

[17] Masahiro Hori, MariAbe, Kouichi Ono, Extensible Framework of Authoring Tools for Web Document Annotations, IBM Tokyo Research Laboratory

[18] Ling Liu, Calton Pu, XWRAP, Liu, L. Pu, C. and Han, W.: XWrap – An XML-enabled Wrapper Construction System for Web Information Sources, Proceedings of the 16th International Conference on Data Engi-neering (ICDE'2000), San Diego CA, 2000, http://www.cc.gatech.edu/projects/disl/XWRAPElite

[19] Ling Liu, Calton Pu, Wei Han, XWRAP: An XML-enabled Wrapper Construction System for Web Sources, Oregon Graduate Institute of Science and Technology

[20] A.Sahuguet, Fabien Azavant, W4F, A.Sahuguet, Fabien Azavant, Building light-weight wrappers for legacy Web data-sources using W4F, Ecole Nationale Supérieure des Télécommunications, Paris, France, fabien.azavant@enst.fr,
http://cheops.cis.upenn.edu/W4F

[21] Robert Baumgartner, Sergio Flesca, Georg Gottlob, Supervised Wrapper Generation with Lixto, 2001

[22] Dave Ragget, Tidy : D.Raggett. Clean Up Your Web Pages with HTML TIDY, 1999, http://www.w3c.org/People/Raggett/tidy/

[23] World Wide Web Consortium (*2000*) : A Reformulation of HTML 4 in XML 1.0, ttp://www.w3.org/TR/xhtml1

[24] World Wide Web Consortium (*1999*) : XPath XML Path Language, http://www.w3.org/TR/xpath

[25] Miloslav Nic, Jiri Jirat, XPath Tutorial, ZVON Tutorials, http://www.zvon.org/xxl/XPathTutorial/General/examples.html

[26] World Wide Web Consortium: The Document Object Model, http://www.w3.org/DOM

[27] World Wide Web Consortium (*1998*) : XML Extensible Markup Language, http://www.w3.org/XML

[28] World Wide Web Consortium: The Extensible Stylesheet Language Family (XSL), http://www.w3.org/Style/XSL