Heterogeneous Web Data Extraction using Ontology

Hicham Snoussi
Centre de Recherche Informatique de Montréal
550 rue Sherbrooke, suite 100, Montréal, Canada H3A 1B9
hsnoussi@crim.ca

Laurent Magnin
Centre de Recherche Informatique de Montréal
550 rue Sherbrooke, suite 100, Montréal, Canada H3A 1B9
lmagnin@crim.ca

Jian-Yun Nie
Université de Montréal
C.P. 6128, succ CENTRE-VILLE
Montreal, H3C 3J7 Canada
nie@iro.umontreal.ca

ABSTRACT
Multi-agent systems can be fully developed only when they have access to a large number of information sources. These latter are becoming more and more available on the Internet in form of web pages. This paper does not deal with the problem of information retrieval, but rather the extraction of data from HTML web pages in order to make them usable by autonomous agents. This problem is not trivial because of the heterogeneity of web pages. We describe our approach to facilitate the formalization, extraction and grouping of data from different sources. To do this, we developed a utility tool that assists us in generating a uniform description for each information source, using a descriptive domain ontology. Users and agents can query the extracted data using a standard querying interface. The ultimate goal of this tool is to provide useful information to autonomous agents.

Keywords : data extraction, WEB, ontology, agent, XML.

1. INTRODUCTION
The Internet contains a huge number of information sources of different kinds. Even if a user has the possibility of browsing the Internet, the search of relevant information is still a difficult task. Most search engines use keywords to identify possible answers to a user's query, and return a list of links to the documents. Many of the returned documents are not relevant to the query, and the user often has to browse the returned document list to find relevant information.

Although a search engine provides useful help for users to identify relevant information, it cannot be used by a software agent to obtain reliable data to fulfil its tasks. This is mainly due to the lack of precision and standard formalism of the returned answers. In addition, current search engines are more focussed on static data on the web rather than dynamic data that constantly change, such as weather forecast, stock exchange information, etc. Such data are more and more required by intelligent agents. If there is a way to extract and reorganize such data, then they can be fully exploited by agents. Our goal is to develop a method to extract reliable data from web pages that intelligent agents can use.

2. PROBLEM DESCRIPTION AND SUMMARY OF OUR APPROACH
Data on the Web are usually included in HTML pages, and they do not correspond to pre-established schema. While a human user can understand the data in a page, it is impossible to do it by a machine. Therefore, extracting data from web pages for agents requires knowledge on both the structure and the contents of the web pages. There have been mainly two approaches to deal with this problem in data extraction from web pages:

- The first approach relies on a natural language processing (NLP). It is known that current NLP is not accurate and powerful enough to recognize the contents of unrestricted web pages. Therefore, this approach has been used in some limited domains.
- The second approach tries to associate a web page with some semantic markers (or tags) when it is created. For example, one may use personalized markers. The limitations of such an approach are well known: the markers are personalized, they can be hardly generalized [2];

As the original data are structured in different ways, it is necessary to restructure them according to a common model that is independent from the information sources. Our approach is based on this idea. In particular, we will focus on data extraction from semi-structured web pages that present constantly changing data, but with a fixed structure (e.g. stock exchange cotes). Our approach makes use of an ontology to model the data to be extracted. The data in a web page is first converted into XML, then mapped with the data model. The definition of the data model and the mapping are done manually. Then an automatic process is carried out to perform the real extraction task. The final result is a XML document that contains a standardized and queriable data set.

3. CHARACTERISTICS OF DATA SOURCES

Exploitable information sources[1] in our context may be classified into three categories: structured, semi-structured and unstructured [14].

- An information source is "structured" if one can query the information using a predefined query language. Database and knowledge base are typical examples of structured information sources.
- An information source is "semi-structured" if it contains a structure that enables querying, however without a querying language as in databases. [12] considers that a source is semi-structured if the information can be retrieved using a formal grammar. HTML pages, Web forms and languages of text description are examples of semi-structured sources.
- A source is unstructured if it does not have any form of standard organization and there is no precise relations among the data.

It is difficult to extract data from unstructured sources. Extensive NLP will be required, yet the result will be uncertain. Therefore, we do not aim to extract information from such sources in this paper. Rather, we want to extract information from semi-structured and structured sources. This is not only because the task is easier, but also because there is a growing number of sites offering semi-structured and structured data. If we can successfully extract information from such sources, a great amount of useful information can be exploited by agents.

Moreover, we do not intend to deal with all kinds of data available on the Web. In fact, only a part of the sources can be submitted to an automatic data extraction, namely those that provide dynamic data with regular updating but in a stable structure (e.g. sites that provide exchange rates).

4. SOME RELATED WORK ON DATA EXTRACTION

In the literature, one often considers that the process of data integration is a mediator between a user requesting information and an information source. Mediation is thus a process that allows a client to obtain data without caring about data identity, storage, structure and access. This is the key to data integration because it enables a transparent access and an interface with the sources [14]. Mediation has been included in almost all the work in data integration.

A necessary condition for integration is a standard syntactic and semantic description of the source.

For structured sources, this means to transform the conceptual schema of the source to another one with which the source will be accessed. The new structure should provide an explicit or declarative description. With a declarative description, one tries to identify possible ways to query the source. These ways correspond to different views (or relations) constructed over the data source [14]. This type of description has been used in Information Brokers. If an explicit description is used, then the source contents are also described and related. Systems that use this kind of description (e.g. InfoSleuth and SIMS) usually make use of a domain model that covers the application area [1] [15].

For semi-structured and unstructured sources, such a model has to be created by the system. Examples are the work of [16] on WMA (Web Mining Agent) and that of [12] (project Ariadne). The model is then understandable to all the elements in the system. After the creation of the model, no further transformation will be necessary for a semi-structured or unstructured source.

WMA uses its own description language to represent and identify data inside HTML pages. XML documents are used as templates for data extraction and transformation. Extracted data are then stored in a relational database. One have to learn WMA description language to be able to use this tool. Also, software agents must query databases to get data, however, in our approach, they can extract data directly from Web pages.
(Question: quelle est la différence de WMA avec "notre approche"? Il y a l'extraction a partir de HTML dans les deux cas. Il y a aussi l'acces des données dans une BD - ou XML dans notre cas, mais c'est conceptuellement la même chose. Donc, les différences soulignées ici me semble pas très convainquantes)

5. CONSTRUCTION OF AN ONTOLOGY FOR DATA EXTRACTION

5.1 The need for an ontology

We notice that all the above mentioned work assumes that all information provided by different sources to be integrated is covered by a domain model. However, information is not necessarily presented in

[1] By "information source" we refer to any system that can provide any type of data.

the same way. Due to this fact, information exchange is not an easy task if different actors (producers or consumers of information) have not agreed on the semantic of data. It is then necessary to define an "alphabet" to ensure a good interpretation and understanding of exchanged data. The role of the ontology is to provide a common model that ensures the minimal requirements for this purpose. In fact, such a model allows one to construct a common view of different sources. The elements in the model are described in a way independent from the particularity of the data source.

The granularity of the model is dependent of the application domain. The more an application domain is restricted, the more it is possible to elaborate a precise description of the domain with the help of an ontology, and the more the processing may be refined. This is achieved mainly with the help of metadata proper of a domain.

There is no unique definition of ontology in literature. Each definition is placed in an expected use and its domain. Below are several definitions we can find:

[8] [7] defines ontology as follows: "An ontology is an explicit specification of some topic. It is a formal and declarative representation which includes the vocabulary (or names) for referring to the terms in a specific subject area and the logical statements that describe what the terms are, how they are related to each other... Ontologies therefore provide a vocabulary for representing and communicating knowledge about some topic …"

In [6], ontology is defined as follows: "A domain conceptualisation (called ontologies) names and describes the entities that may exist in that domain and the relationships among those entities. It therefore provides a vocabulary for representing and communicating knowledge about the domain."

Ontology is a way to discompose a world into objects, and a way to describe these objects. This is a partial description of the world, depending on the objectives of the designer and the requirements of the application or system. For each domain, there may be a number of ontologies [3]. The use of ontology differs from an application to another, so are its design and its formalism of representation.

5.2 Modeling of ontology

Our intended use of ontology is to describe a data model, rather than knowledge. Therefore, it is not necessary to include inference and reasoning mechanism to produce new knowledge at the current stage. We will use a modeling of ontology close to object-oriented (OO) modeling. We believe that with OO paradigm, we can express an ontology in an explicit way and generate software elements that are easily exploitable by other applications.

We propose a design of ontology that uses a 3-level model: basic objects, meta-model and model.

We consider four types of object: entity, attribute, relation and constraint. These basic objects are used in the definition of the model, as well as meta-model.

- Entity: Entities describe concepts (elements of the domain studied) and provide a logical representation of them.
- Attribute: An attribute corresponds to a property that characterizes an entity.
- Relation: It describes links between objects in the model (i.e. entities and attributes).
- Constraint: A constraint is a condition that the designer imposes on entities, attributes or relations.

The use of a meta-model allows us to define interactions between objects previously defined. The role of meta-model is to provide a way for a designer to express his own constraints on the basic objects. The meta-model can thus be used to refine the basic objects and to define properties on them. The meta-model layer has been introduced and experimented in [22] where the goal is to describe axioms of ontology. We use the same concept here, but in a more general framework. We use this approach to express specific design requirements of the user, as in [8] and [22]. Particularly, we will allow a redefinition of the roles of the elements of the model by the designer, according to the particular requirements of the application. Each user can specify his own meta-model of his ontology.

An example of meta-model that we used to model ontology is shown in figure 1. We can notice following facts in this meta-model:

- An entity is a representation of a concept or an object of real world.
- An attribute is unitary. It cannot be decomposed.
- We include in the meta-model the relations between entities, and between entities and attributes. A relation is Binary (entity-entity or entity-attribute) or Multiple (entity-entities or entity-attributes). A binary relation between an entity and an attribute is called Property. It means that the attribute is a property of the entity. A binary relation between an entity and another entity is either a specification relation IS-A or a composition relation HAS-A. A multiple relation Properties connects an entity with a set

of attributes. A multiple relation between an entity and a set of entities is a composition relation HAS-A.

- We choose to set constraints on relations. They correspond to elementary requirements in terms of occurrence number (Cardinality), a precise sequence of decomposition (Sequence) or a choice among a given set (Choice). The constraints Sequence and Choice only apply to multiple relations to require a sequence or a choice among entities or attributes. The Cardinality constraint applies to binary or multiple relations to define the possible occurrence numbers for entities and attributes that form this relation.

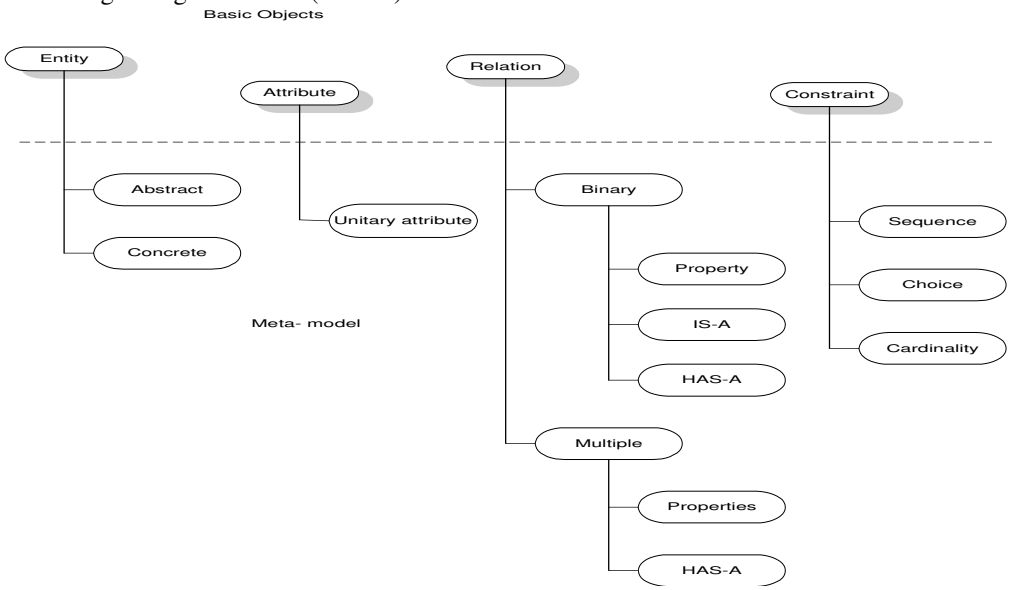


Figure 1. Example of meta-model

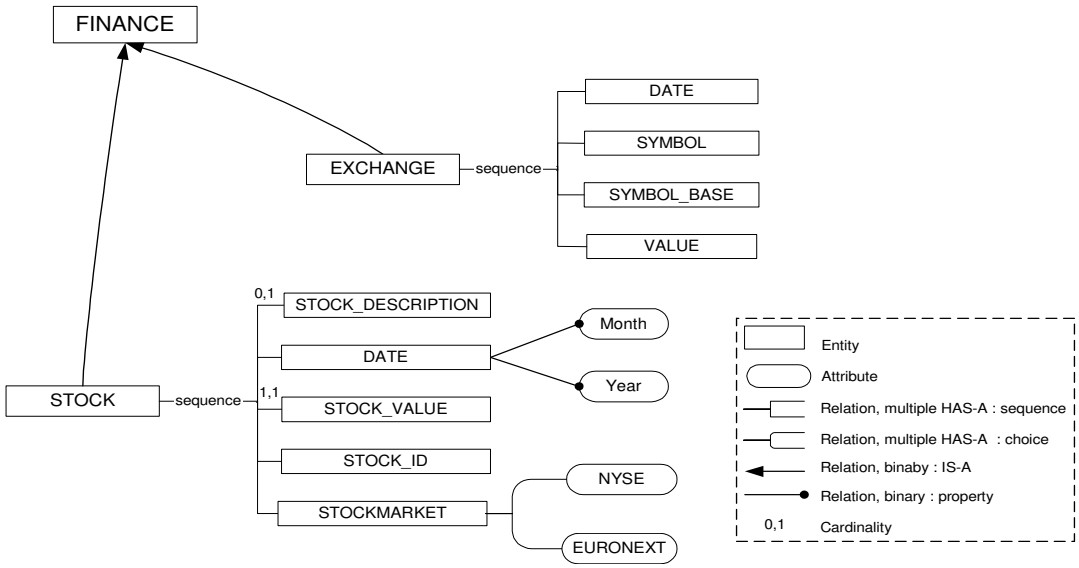


Figure 2. Example of a simplified model

In figure 2, we give a simple example of ontology model in finance area. This model makes use of the previous meta-model. For example, the entity STOCK is in relation *HAS-A* with entities STOCK_DESCRIPTION, DATE, STOCK_VALUE, STOCK_ID and STOCKMARKET. Since the presence of these last entities is necessary to describe and represent the entity STOCK, the relation *HAS-A* which binds the entity STOCK and the set (STOCK_DESCRIPTION,DATE, STOCK_VALUE, STOCK_ID and STOCKMARKET) takes the form of a composition sequence. In practice, this means that these entities have to be put together. The presence of an entity (STOCK) depends on the presence of the all entities that compose it. The creation of this kind of dependence makes it possible

to control the completeness of data extracted and the validation of documents. In addition, entity STOCKMARKET is in relation *CHOICE* with two attributes NYSE and EURONEXT. So one of the attributes has to be selected.

6. TOOLS AND LANGUAGES FOR MODELING

We used the language SOX (Schema for Oriented-Object XML), a language of definition of schemas for XML documents [20] for ontology definition and data modeling. SOX is developed by *Commerce One* to use XML in electronic commerce [4]. A schema means a set of rules that defines the structure of an XML document. A DTD (Document Type Definition) is a special case of language of schema used to define XML documents. Given a schema, one can create instances of XML documents that agree with the schema. The verification is done by a *parser*. When using SOX to define an ontology, the data extracted from Web sites are organized as XML documents that correspond to the ontology (expressed as SOX schemas). See appendix 1 for an example of SOX schema.

The choice of SOX is motivated by the fact that one can use it to express the requirements of the meta-model that we defined previously. SOX has been developed to overcome the insufficiencies of DTD. It is close to the OO paradigm and introduces concepts of OO-programming into XML documents. SOX can also define data types. In addition of the basic data types, the user can define his own data types. For example, the user can specify that an attribute can only take value from a given interval. This is not possible with DTD. The defined schema ensures some control of the model and makes the model more reliable.

We used XDK (XML Developer's Kit) from Commerce One to parse a SOX document [4].

7. Extraction process

An important step in data integration is to determine the methods for their access and extraction. If the retrieval of data from a database is usually easy to do (e.g. using a query language such as SQL), this operation is difficult for semi-structured data because of the lack of data schema. Semi-structured data are not directly queriable. It is necessary to first construct a view or a model for the data source by an automatic tool. In other words, we have to find a way that takes advantage of the existing organization in the semi-structured data.

Let us first review some existing data extraction tools before describing our approach.

7.1 W4F and JEDI : existing tools for data extraction

W4F (*World Wide Web Wrapper Factory*) [18] and JEDI (*Java Extraction and Dissemination of Information*) [10] are two tools for data extraction from web pages. W4F is a development environment that allows users to construct a *Wrapper*, to compile it as a Java component and to include it in applications. A *Wrapper* is an interface to access the contents of web pages. It download HTML documents from the Internet, corrects them and extract data from them. Extracted data are connected with predefined variables [19].

JEDI is a set of tools for generating *Wrapper*s in order to extract data from textual sources. This may be applied to web pages as well as other textual files. The goal of a *Wrapper* is to indicate how to generate a representation in XML for the extracted data. (Commentaire: quelles difference avec la precédente definition de wrapper? S'agit-il de deux wrappers differents dans ces deux cas? Si oui, il faudrait dire: The goal of a *Wrapper* in JEDI …) Usually, a *Wrapper* is a text that contains a set of extraction transformation instructions, including rules and codes of control. A rule contains a syntactic constraint describing the data (character strings) to be extracted.

W4F and JEDI use two different extraction languages. Users have to learn the languages in order to write *Wrapper*s. The languages have their own syntax to define declarations, manipulations of variables and even scripts. Indeed they are similar to a programming language. This makes them difficult to use. On the other hand, it is also difficult to construct a tool to generate *Wrapper*s, due to the complexity of the language. Although W4F proposes a tool to assist the user to write *Wrapper*, the tool is only limited to help the user identifying element's path. It does not allow an automatic generation of *Wrapper*.

Due to these reasons, we do not use this type of language. We designed a similar, but much simplified approach (in terms of utilization) that offers possibilities for future improvements.

7.2 Our extraction method

Our approach does not rely on the identification of boundaries of character strings within HTML documents, as is the case in TSIMMIS [9]. Rather, we use a special processing of HTML pages and an appropriate querying language.

For each source, we construct a description that shows how the data can be found and extracted from a Web page. The edition of the description is done manually, with the help of an assistance tool.

7.2.1 *Characteristics of Web pages*

Web pages presenting dynamic data usually have a fixed structure: data of a given type is shown at a given place. This is mainly due to the fact that data shown in the page are usually provided by a database (or another structured representation) behind the page. However, the data base itself is not directly accessible. This is why it is necessary to extract data from an HTML document. We will exploit this "property" in our data extraction approach.

The HTML language does not offer any mechanism for direct querying. However, the W3C proposed a specification to access the contents of XML documents based on their hierarchical structure [23]. It is thus interesting to transform HTML documents into XML in order to take advantage of this possibility. Although HTML may be considered as an instance of XML, WEB pages often contain imperfect and incomplete structures. Navigators often ignore errors of these pages to allow users to see HTML documents in most cases. For our conversion purpose, we have to correct the HTML pages in order to satisfy the norm of XML.

7.2.2 *Transforming HTML to XML*

To correct errors in HTML documents, we used correction tools - HTML TIDY [17] and W4F [19] (correction and transformation module of W4F).

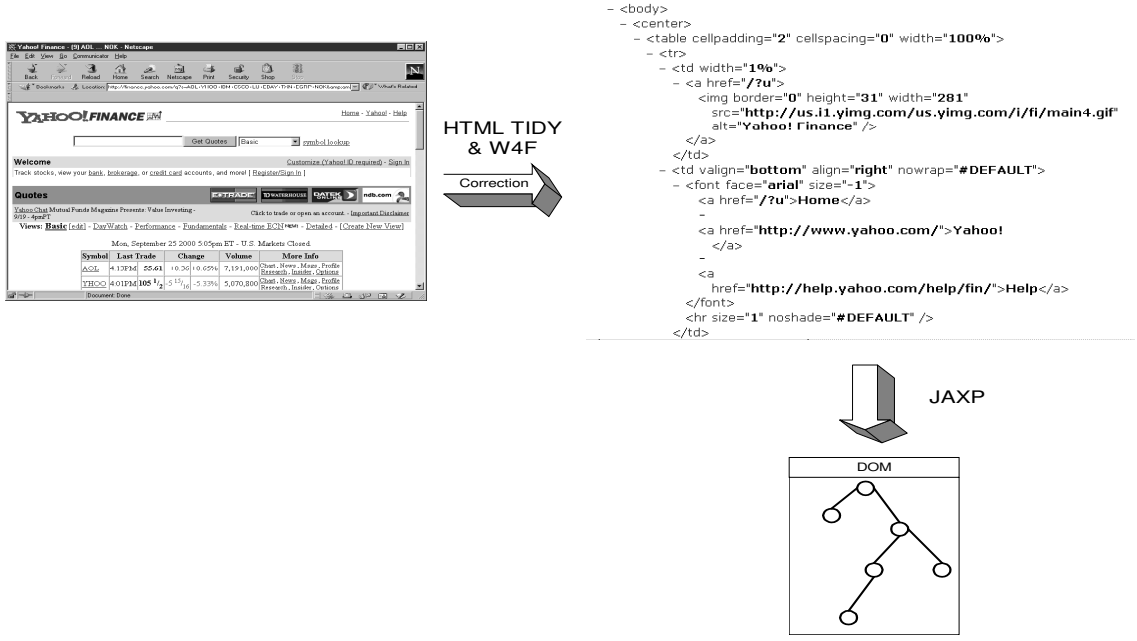


Figure 3. Conversion of an HTML page into XML

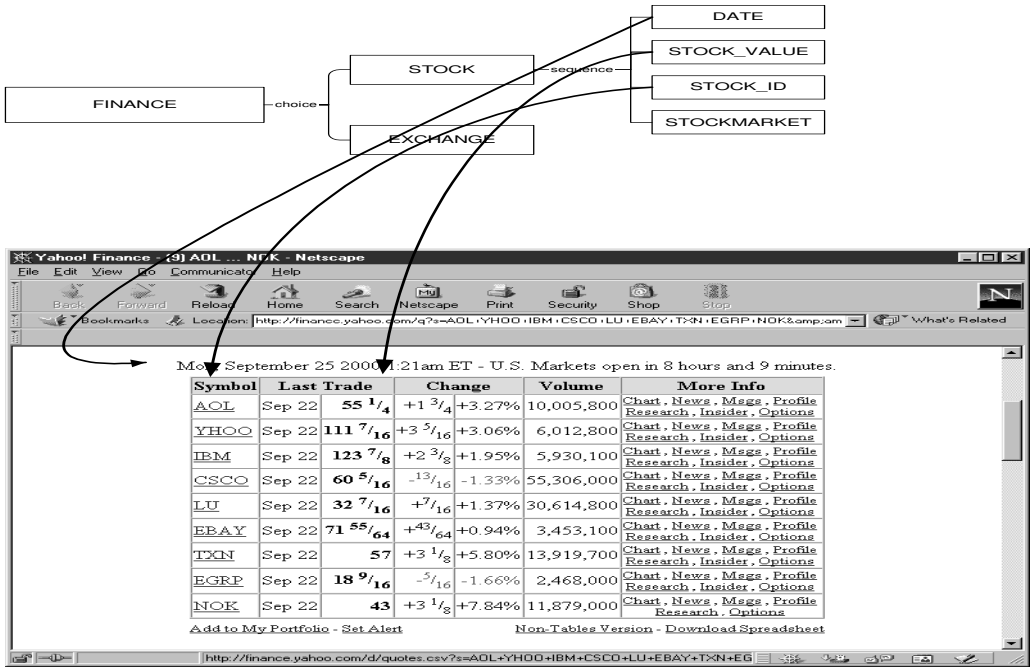


Figure 4. Mapping with the ontology

Once a web page is transformed into XML, data can be accessed using the parser DOM (Document Object Model) [5]. In our case, we used JAXP (from Sun Microsystems) as a DOM parser. The access to a data is done by following a path from the root in a hierarchy. The figure 3 illustrates this conversion process.

7.3 Mapping of date with ontology

As we explained earlier, our general approach is based on a description of the domain in form of an ontology. In fact, the data extracted will be linked to the corresponding elements of the ontology.

The figure 4 shows the construction of the element <STOCK>. This latter is part of the element <FINANCE>. The figure also shows the data in the Web page that correspond to the elements of the ontology.

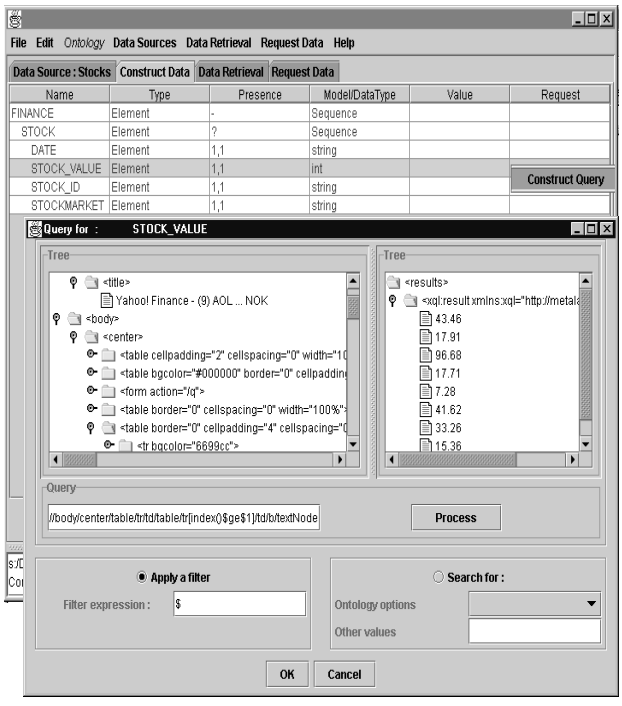
XQL is then used to retrieve data converted into XML [11]. This language has been proposed to access the contents of XML documents. It is simple, concise, and easy to use. Its characteristics would allow us to build queries graphically.

8. OUR PROTOTYPE

Our prototype is an assistance tool of data extraction. The extraction follows several steps: download a web page and convert it into XML, construct a data model using the ontology, and map the XML document with the elements in the ontology. The HTML-XML conversion is performed using the tools described in 7.2.2. This step can be done through our graphical interface.

In the construction of data model, the user chooses the elements of the ontology that correspond to his interests. The selection of elements by the user tries to answer the question "what to extract". The process of mapping and the underlying querying process answer the question "how to extract".

The figure 5 shows a snapshot of screen during the step of data model construction (background window). We can see that … (l'exemple de STOCK_VALUE). This window is also used for linking the entities of the ontology with the data at the source. The ontology elements are displayed in a table. The right column allows the user to specify an XQL query to get data corresponding to



specified ontology element. A new window (foreground window) will be used to generate XQL query. The lower frame in this window helps user to construct a query from a DOM tree view of HTML page (left part). The user navigate the tree to locate data he is interested in. He edit his XQL query on converted XML document to select data. The results are displayed in the right part. We are working on a new version that can help the user choosing his data graphically. The user gets data for each element of the ontology. These data are combined together in a way soecified in the ontology.

The result is a description that contains all the process in form of a specification (description) for a given source - see an example of a source description file in appendix 2. The specification contains, among other things, the names of entities/attributes, the queries, the transformations and the information about the structure of the final result (an XML document, see appendix 3). The advantage of making a specification is that it is reusable along time, and by an extraction program as well as an autonomous agent.

Figure 5. Screen snapshot - construction of data and linking to elements of the ontology

The figure 6 summarizes the main steps of extraction. The lower part can be integrated in a software agent. It is an automatic step since the agent uses only description (generated before) to recover the data. Since the extracted data respect the same ontology, the extraction results from several sources have the same structure. A simplistic manner to combine them would be to put them in a same XML document.

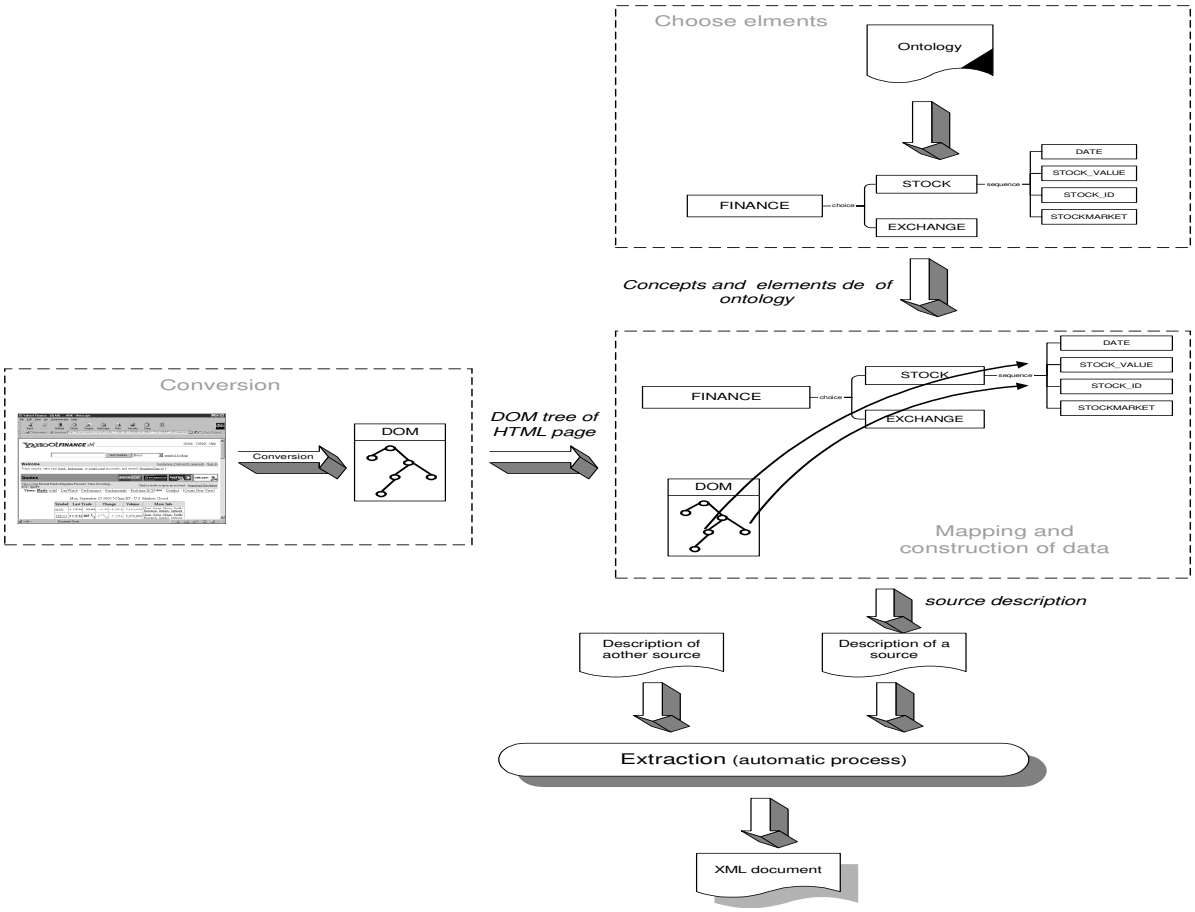


Figure 6 : Global view of the extraction

9. Autonomous agents using information extraction and integration

The progress in the development of agent technology at both theoretical and practical levels offer an interesting perspective of using agents. Data extraction and integration are prerequisite to many operations of agents. The implementation of services such as travel planning, information retrieval, e-commerce (buy, sell and negotiation) and stock exchange all require reliable data sources. Our tool is

a step forward to provide such data to be exploited by software agents.

A Multi-Agents System – MAS - that uses our extraction approach is under development in CRIM. Based on GUEST agents developed in CRIM [13], this application aims to monitor cotes in stock exchange. In particular, the agents follow the course of stock index and inform the user of the evolution of operations and the trends. The overall architecture is given in figure 7. See at the end a sample Java source code used by an agent to get a stock value - appendix 4. The agent uses gives a description source file and a stock id.

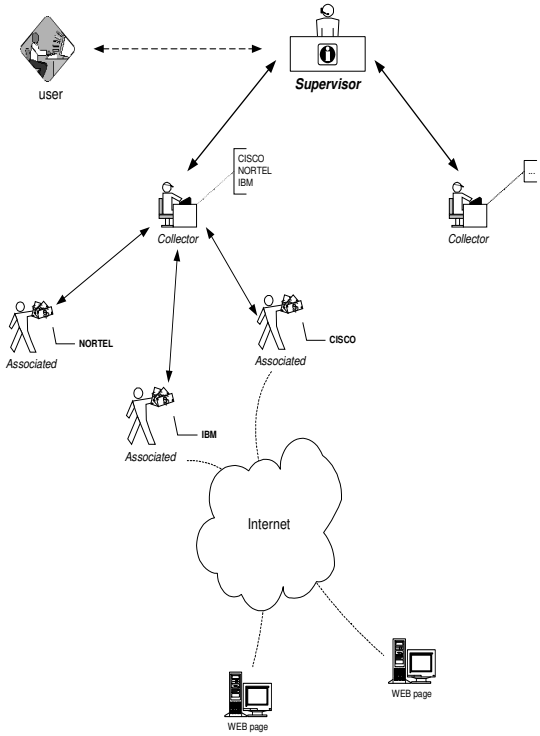


Figure 7. Global view of the extraction

10. Conclusions and discussions

In this paper, we dealt with the problem of data extraction from web pages and their integration in applications. In particular, our goal is to find a way to extract reliable data, and to convert them in a standard form.

The extraction of data consisted in two steps: converting an HTML page into XML and using XQL to query XML documents to extract the desired data. The extraction process is controlled by a specification file, which describes what elements of a web page to extract, and how they have to be extracted. As the user has a tight control on the extraction process, the extracted data is of high quality, thus can be exploited by other programs or software agents.

The integration of data is based on the use of an ontology, which provides a common model for information sources. The use of an ontology has greatly simplified our task of extraction and integration. The most critical point remained is the definition of an ontology. However, we cannot imagine an open system that exchanges data without using a norm of the domain. Even if on cannot construct a complete ontology, a standard will always be necessary to play a similar role.

We implemented a prototype that allowed us to extract some types of data from web pages (in finance). Although some steps have to be done manually, namely the construction of a specification of the extraction process, they are greatly facilitated by the graphical tool we constructed. The advantage of such a specification is that, once constructed, it can be reused for similar applications. Moreover, the same specification can be exploited by software agents to get data.

The most useful case of such an extraction process is on web pages that present dynamic contents, but with fixed structures. Examples are web pages that provide stock market exchange prices, money exchange rates, and so on. If an information site is restructured, the extraction process is no longer valid. We have to construct a new extraction process. This means that we have to monitor the validity of an extraction process.

However, one has to notice that:

- Many Web pages that provide dynamic data do not change often the page structure. This means that even the data shown in a page change, the extraction process is still valid. Our method aims to extract data from such information sites.
- Many Web pages are only a presentation page for data stored in a database. Even if the data change regularly, the structure remains the same.
- Finally, we extracted several types of data from web pages. According to our experience, these pages have not changed their structures during the whole period of our test (several months). This confirms the stability of structures of many information sites, and indicates that our method can be used in practice for many web sites.

This study shows that it is possible to exploit and use automatically the data presented in some web pages. However, some manual preparation is necessary. We argue that this manual step is always necessary if one's intention is to extract reliable data to be exploited by other programs or software agents. Despite the manual preparation, we believe that this

approach is appropriate for extracting data to be integrated in software agents.

Acknowledgement
This work has been supported by a scholarship of AUPELF (Association des Universités Partiellement ou Entièrement de Langue Française) to Hicham Snoussi, and a complementary scholarship of the NSERC (Natural Sciences and Engineering Research Council of Canada). We would like to take this opportunity to thank the two organizations.

11. References

[1] Arens, Y., Chee, C. Y., Hsu, C. N. and Knoblock, C. A., Retrieving and Integrating Data from Multiple Information Sources, International Journal of Intelligent and Cooperative Information Systems. Vol. 2, No. 2, 1993.

[2] Atzeni, P., Mecca, G. and Merialdo, P., To Weave the Web - In Proceedings of the 23rd International Conference on Very Large Databases (VLDB'97), 1997

[3] Bezivin, J., LES NOUVELLES CONVERGENCES : OBJETS, COMPOSANTS, MODÈLES ET ONTOLOGIES, JICAA'97, Roscoff France, Mai 1997.

[4] Commerce One, http://www.commerceone.com/

[5] Document Object Model,http://www.w3.org/DOM/

[6] Farquhar, A., Fikes, R., Pratt, W. and Rice, J., Collaborative Ontology Construction for Information Integration. Knowledge Systems Laboratory, Department of Computer Science, Technical Report KSL-95-63, August 1995.

[7] Gruber, T., Ontology definition, http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/what-is-an-ontology.html

[8] Gruber, T., Toward principles for the design of ontologies used for knowledge sharing, The International Workshop on Formal Ontology, March 1993.

[9] Hammer, J., Garcia-Molina, H., Cho, J., Aranha, R., and Crespo, A., Extracting Semistructured Information from the Web". In Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, May 1997.

[10] Huck, G., Fankhauser, P., Aberer, K. and Neuhold, E.J., JEDI: Extracting and Synthesizing Information from the Web, Conference on Cooperative Information Systems CoopIS'98, New York, August, 1998, IEEE Computer Society Press.

[11] Ishikawa, H., Kubota, K. and Kanemasa, Y., XQL: A Query Language for XML Data, Query Languages'98 (QL'98) workshop, Boston, Massachussets, December 1998.

[12] Knoblock, C. A., Minton, S., Ambite, J. L., Ashish, N., Modi, P. J., Muslea, I., Philpot, A. G. and Tejada, S., Modeling Web Sources for Information Integration. Proceedings of the National Conference on Artificial Intelligence, Madison, 1998.

[13] Magnin L., and Alikacem, E. H., GenA : Multiplatform Generic Agents, MaTa'99 First International Workshop on Mobile Agents for Telecommunication Applications, Ottawa, October 1999.

[14] Martin, D. L., Oohama, H., Moran, D. and Cheyer, A., Information Brokering in an Agent Architecture, Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, April 1997.

[15] Nodine, M., Fowler, J. and Perry, B., An Overview of Active Information Gathering in InfoSleuth, Technical Report, October 1998, http://www.mcc.com/projects/infosleuth/publications/TR98/INSL-114-98.ps

[16] OUAHID, H and KARMOUCH, A., An XML-Based WEB Mining Agent, Proceeding of MATA'99, Ahmed KARMOUCH and Roger IMPEY edts., World Scientific, Ottawa, October 1999.

[17] Raggett, D., HTML Tidy, http://www.w3.org/People/Raggett/tidy/

[18] Sahuguet, A. and Azavant, F., Building light-weight wrappers for legacy Web data-sources using W4F, International Conference on Very Large Databases (VLDB), Edinburgh - Scotland – UK, September 7 - 10 1999.

[19] Sahuguet, A. and Azavant, F., Looking at the Web through XML glasses, Conference on Cooperative Information Systems CoopIS'99, Edinburgh Scotland, September 2-4 1999.

[20] Schema for Oriented-Object XML, http://www.commerceone.com/xml/cbl/docs/

[21] Staab, S. and Maedche, A., Axioms are objects, too— ontology engineering beyond the modeling of concepts and relations. Technical Report 399, Institute AIFB, Univ. of Karl-sruhe, 2000.

[22] Staab, S., Erdmann, M. and Maedche, A., An extensible approach for Modeling Ontologies in RDF(S), Submitted to the 12th International Workshop on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, French Riviera, October 2-6, 2000.

[23] W3C, http://www.w3.org

Appendix1. Example of a SOX schema

```
<?xml version ="1.0"?>
<!DOCTYPE schema SYSTEM "urn:x-commerceone:document:com:commerceone:xdk:xml:schema.dtd$1.0">
<schema uri = "file:///S:/DataFiles/schemas/n1_0/Finance.sox" soxlang-version = "V0.2.2">
        <elementtype name = "FINANCE">
                <model>
                        <sequence>
                                <element type = "STOCK" occurs = "?"/>
                                <element type = "EXCHANGE" occurs = "?"/>
                        </sequence>
                </model>
        </elementtype>
        <elementtype name = "STOCK">
                <model>
                        <sequence>
                                <element type = "STOCKMARKET"/>
                                <element type = "STOCK_ID"/>
                                <element type = "STOCK_VALUE"/>
                                <element type = "DATE"/>
                        </sequence>
                </model>
        </elementtype>
        <elementtype name = "STOCKMARKET">
                <model>
                        <string/>
                </model>
        </elementtype>
        <elementtype name = "STOCK_ID">
                <model>
                        <string/>
                </model>
        </elementtype>
        <elementtype name = "STOCK_VALUE">
                <model>
                        <string datatype = "int"/>
                </model>
        </elementtype>
        <elementtype name = "DATE">
                <model>
                        < string/>
                </model>
        </elementtype>
        …
</schema>
```

Appendix2. Example of a description source file

```
<?xml version="1.0" encoding="UTF-8"?>
<SOURCE>
    <NAME>Stocks</NAME>
    <URL>http://finance.yahoo.com/q?s=AOL+YHOO+IBM+CSCO+LU+EBAY+TXN+NT+NOK&amp;amp;d=v1</URL>
    <ONTOLOGYFILE>file:///S:/DataFiles/schemas/n1_0/Finance.sox</ONTOLOGYFILE>
    <ITEMS>
        <ITEM>
            <INDEX>0</INDEX>
            <NAMEITEM>FINANCE</NAMEITEM>
            <TYPE>element</TYPE>
              <ITEM>
                <INDEX>1</INDEX>
                <NAMEITEM>STOCK</NAMEITEM>
                <TYPE>element</TYPE>
```

```
                <ITEM>
                    <NAMEITEM>DATE</NAMEITEM>
                    <TYPE>Element</TYPE>
                    <INDEX>2</INDEX>
                    <QUERY>//body/center/p[0]</QUERY>
                </ITEM>
                <ITEM>
                    <NAMEITEM>STOCK_VALUE</NAMEITEM>
                    <TYPE>Element</TYPE>
                    <INDEX>3</INDEX>
                    <QUERY>//body/center/table/tr/td/table/tr[index()$ge$1]/td/b/textNode() [0]</QUERY>
                </ITEM>
                <ITEM>
                    <NAMEITEM>STOCK_ID</NAMEITEM>
                    <TYPE>Element</TYPE>
                    <INDEX>4</INDEX>
                    <QUERY>//center/table/tr/td/table/tr[index()$ge$1]/td/a[@href$contains$'/q?']</QUERY>
                </ITEM>
              <ITEM>
                  <NAMEITEM>STOCKMARKET</NAMEITEM>
                  <TYPE>Element</TYPE>
                  <INDEX>5</INDEX>
                  <QUERY>//body/center/p[0]</QUERY>
              </ITEM>
            </ITEM>
        </ITEM>
    </ITEMS>
</SOURCE>
```

Appendix3. XML document of extracted data

```
<?xml version="1.0" encoding="UTF-8"?>
<RESULTS>
        <FINANCE>
                <STOCK>
                        <DATE>Thursday, March 29 2001 12:24pm ET </DATE>
                        <STOCK_VALUE>40.56</STOCK_VALUE>
                        <STOCK_ID>AOL</STOCK_ID>
                        <STOCKMARKET> U.S. Markets</STOCKMARKET>
                </STOCK>
        </FINANCE>
        <FINANCE>
                <STOCK>
                        <DATE>Thursday, March 29 2001 12:24pm ET </DATE>
                        <STOCK_VALUE>14 </STOCK_VALUE>
                        <STOCK_ID>YHOO</STOCK_ID>
                        <STOCKMARKET> U.S. Markets</STOCKMARKET>
                </STOCK>
        </FINANCE>
        <FINANCE>
                <STOCK>
                        <DATE>Thursday, March 29 2001 12:24pm ET </DATE>
                        <STOCK_VALUE>95.38</STOCK_VALUE>
                        <STOCK_ID>IBM</STOCK_ID>
                        <STOCKMARKET> U.S. Markets</STOCKMARKET>
                </STOCK>
        </FINANCE>
        …
</RESULTS>
```

Appendix4. Sample Java source code used by an agent

```
{
```

```
    …
    file = "http://www.crim.ca/~hsnoussi/sources/StockMarket.xml";
    xmlDocument = StockData.extract(file);
    stock_ID = "IBM";
    value = StockData.getValue(xmlDocument, stock_ID)
    return value;
}
```