Semantic Integration of XML Heterogeneous Data Sources

Chantal Reynaud
LRI-Univ. Paris XI &Univ. Paris X
cr@lri.fr

Jean-Pierre Sirot
Xyleme S.A.
jean-pierre.sirot@xyleme.com

Dan Vodislav
CNAM/CEDRIC, Paris
vodislav@cnam.fr

Abstract

With the current explosion of data, retrieving and integrating information from various sources is a critical problem. The designer has to specify a mediated schema providing a homogeneous view on the sources. In this paper, we report on an initial work toward automatically generating mappings between elements in the sources and in the mediated schema. Information sources we are interested in are XML documents in respect with a Document Type Definition (DTD). We describe the Xyleme project, which is the context of this study. We present our approach implemented in the SAMAG system to automatically find mappings on the basis of semantic and structural criteria. Finally, we report the first results of an experiment where SAMAG has been applied to XML documents in the cultural domain.

1. Introduction

Internet and Intranet architectures allow an access to multiple, distributed and heterogeneous information sources. Each of these sources may contain relevant and complementary information. In this paper, we report on an initial work to allow the user to express queries at an abstract and semantic level and to automatically reformulate the queries in terms of the different information sources.

Information sources we are interested in are XML documents. XML is an emerging standard for data representation and exchange on the World-Wide-Web [11]. Tagged elements are the building blocks of XML. An XML document may be accompanied by a Document Type Definition, (DTD), essentially a grammar for restricting the tags and structure of the document. So, query processors for XML-based data can support expressive query languages exploiting the name of the tags and the structure of the documents. Our work is a part of the Xyleme project whose aim is to query a huge repository of XML documents.

Format homogeneity among distributed information systems makes the construction of an integrated system easier. However, semantic heterogeneity still remains. The studied DTDs come from various sources. Different persons have written them and all these persons have made personal choices concerning the name of the tags and their structure. Solutions to semantic heterogeneity in the context of information retrieval are based on the introduction of global domain descriptions, viewed as interfaces between end-users and information sources. The role of this mediated layer is twofold. First, it provides to the end-users a vocabulary for querying the system. Second, it connects all the information sources.

Currently, the connection between information sources and domain descriptions corresponding to the global model of the system can be established in two different ways [20]. The global model can be defined as a collection of views over the sources [8]. In that case, query rewriting algorithms are very efficient, but it leads to update the global model whenever any of the sources change or when new sources are added. In more recent approaches, sources are defined as views of the global model [14] [7] [2]. In such a case, adding new sources is quite straightforward because each source is defined independently from others. However the algorithms to rewrite a domain level query into a source level query involves testing containment of views, which is computationally expensive. So, none of these two approaches is perfectly suitable. This leads us to advocate a different and scalable approach with two objectives: (1) efficiency of query processing, (2) flexibility of adding new documents.

In our approach, we propose a pre-compile step where DTDs associated to XML documents and the global domain descriptions are connected through matching relationships, called mappings. These mappings computed off-line, in a batch mode, are stored and then exploited at run-time. Such an approach combines flexibility in the description of a new document XML with the run-time efficiency.

In this paper, we describe the problem of the connection between XML documents and domain description. Our aim is to find a connection method and techniques that can be automated the most as possible. First, we describe the Xyleme project, which is the context of this study. We present the overall system architecture, the semantic layer that must be connected with XML documents and the nature of the XML documents and of the DTDs studied in our work. We present in section 3 a review of our approach for the connection problem. In section 4, we report and analyze the first results of an experiment. To end with, we give an overview of related work in section 5 and we conclude.

2. The Xyleme Project

Xyleme is an initiative of INRIA which involves the following participants: the VERSO group of INRIA (F), the IASI research group of the LRI at the Paris-South university (F), the Vertigo research group of CNAM in Paris (F) and a research group of the university of Mannheim (G). This project has been initiated in September 1999. The goal was to look for innovative solutions for Web information retrieval, assuming that the XML language will become a standard for data representation and exchange on the WWW.

2.1. Xyleme architecture

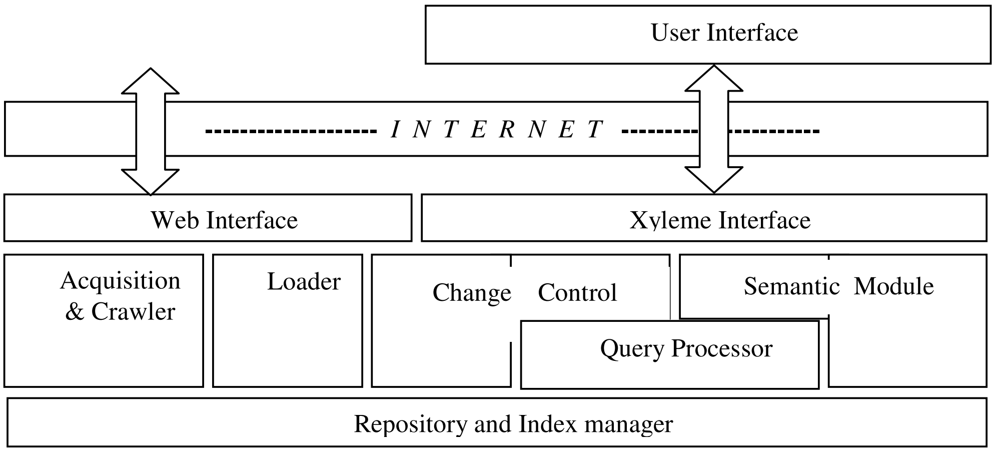

Figure 1. Xyleme Architecture

The Xyleme project deals with data integration when data sources are XML documents. All XML documents are stored in a repository. In this way, the system is efficient even when several data issued from distributed sources must be combined to answer queries. A semantic layer plays the role of interface between end-users and XML documents which, by definition, come from several sources and are semantically heterogeneous. Figure 1 presents the overall architecture of the system.

- The Repository and Index Manager module is the lowest level in Xyleme. It enables to store and index XML documents.
- The Acquisition and Crawler module inspects the Web and collects all the XML documents, which are loaded in the repository by the Loader module.
- The Change Control module is responsible of specific functionalities, such as monitoring of document changes, version management and subscription of temporal queries.
- The Semantic module is the component we focus on in this paper. It provides a homogeneous view on the heterogeneous XML documents stored in the repository.
- The Query Processor module enables to query the XML repository as a database. In particular, it translates a query in terms of the semantic layer into another one computable on the stored documents.

2.2. The semantic layer

The aim of the semantic layer is twofold: (1) to provide a homogeneous view over the repository, in order to allow a natural expression of queries, "at a semantic level" without having to be aware of the structure and the syntax of the real DTDs describing the documents; (2) to define the connection between this semantic layer and the actual data described by the DTDs of the XML documents in the repository.

The semantic layer provides simple descriptions of domains (e.g. culture, tourism), called "abstract DTDs". An abstract DTD can be considered as an abstract merger of the DTDs associated to the set of XML documents of that domain. The abstract DTD is the basis of the semantic layer.

We model an abstract DTD as a tree whose nodes are terms of the corresponding domain (Culture, Tourism). Figure 2 shows a fragment of an abstract DTD in the cultural domain. The label of the root is the domain name (Culture). The internal nodes are labelled with terms that represent either concepts of the domain (Art, Architecture, Movie) or properties of those concepts (Name, Title).

There are two main reasons for the choice of representing the abstract DTD as a tree. First, we want it to be compatible with a real XML DTD, because it is intended to be the DTD of all the XML documents of the domain. More precisely, query results are XML files whose type is given by the abstract DTD. Next, the abstract DTD must be simple enough, because it is the support of the visual query interface tool, intended to be used by non-programmer users.

The links between a node and its sons in the abstract DTD have no strict semantics. They may correspond to a relation of specialisation between concepts, or to a relation of composition, or simply express different viewpoints on a concept. A link also exists between a concept and a property of it. We do not distinguish between those different types of links. In other words, a node labelled with t2 being a son of a node labelled with t1 just means that the term t2 has to be interpreted in the scope of meaning of t1. Thus, the different occurrences of a same term do not have the same meaning. This is obvious for terms that represent properties (e.g. "Name" can appear different times, and may represent the name of different entities), but it is also the case for terms representing concepts (e.g. "Author" under "Movie" means director, while "Author" under "Painting" means painter).

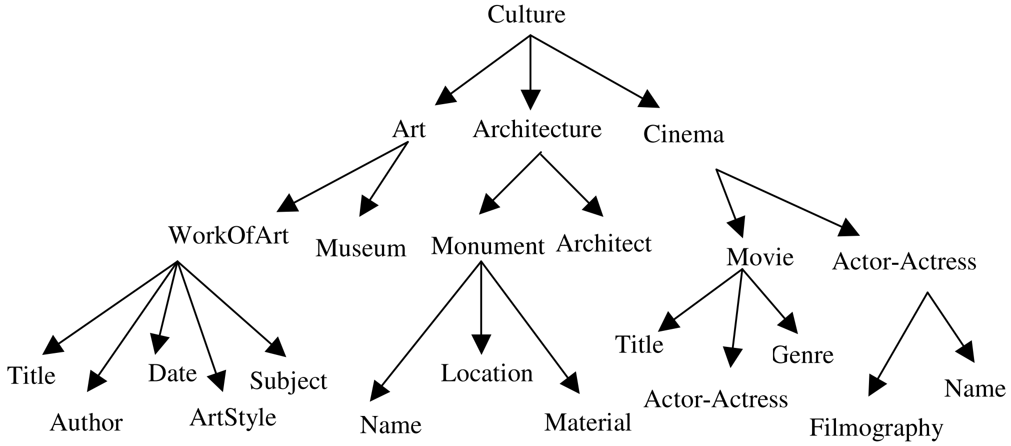
Culture
Art
Architecture
Cinema
WorkOfArt
Museum
Monument
Architect
Movie
Actor-Actress
Title
Author
Date
ArtStyle
Subject
Name
Location
Material
Title
Actor-Actress
Genre
Filmography
Name

Figure 2. Fragment of a cultural abstract DTD

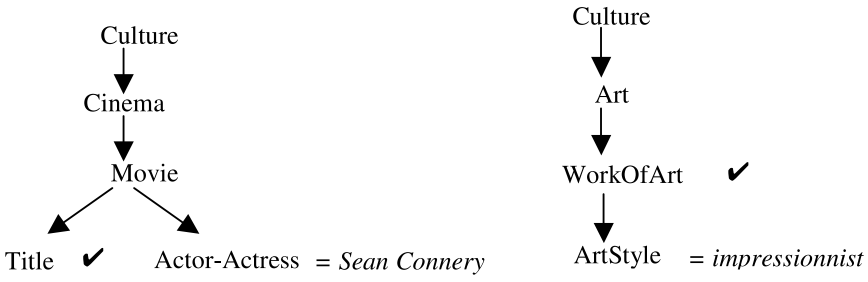
Culture
Cinema
Movie
Title ✔
Actor-Actress = Sean Connery
Culture
Art
WorkOfArt ✔
ArtStyle = impressionnist

Figure 3. Examples of user queries

Such representations are very simple and structured. They enable end-users to build semantic queries from different viewpoints in an easy and natural way, using both the vocabulary and the structure of the abstract DTD. Abstract queries over a domain are also modelled with trees, more precisely sub-trees of the abstract DTD of the domain.

For example, the sub-tree on the left in figure 3 models a query in the cultural domain asking the titles of all the films in which the actor Sean Connery is acting. The query on the right in figure 3 asks the description of all impressionist works of art. These examples illustrate the basic method for building abstract queries: the user marks in the abstract DTD tree the nodes to be included in the result (projection nodes, e.g. Title, WorkOfArt) and the nodes constrained with conditions (selection nodes, e.g. Actor-Actress, ArtStyle). The query itself is given by the sub-tree of the abstract DTD connecting these nodes.

It is known that in data retrieval systems, we can increase precision by adding structure to the representation formalism [10]. By exploiting structure and semantics, we can improve both precision and recall.

Such techniques are studied in the Xyleme project and are described in section 3.

Currently, the construction of the abstract DTDs is not automated.

2.3. XML documents and DTD

The XML documents we are interested in are valid documents, instances of a DTD that defines their structure. Figure 4 presents a XML document, instance of the DTD "MyDTD", represented in figure 5.

```
<MyDTD>
<Movie>
<Actor>Sean Connery</Actor>
<Actor>Lois Maxwell</Actor>
<Title>Goldfinger</Title>
<Category>Action</Category>
</Movie>
<Painting>
<Title>La course</Title>
<Author>Picasso</Author>
</Painting>
</MyDTD>
```

Figure 4. An XML document whose structure is described in MyDTD (figure 5)

In this paper, we only consider acyclic DTDs without attributes nor entities or notations. Therefore, a DTD can be represented as an oriented acyclic graph, with one node for each element definition. The links between the nodes are composition links: a link n1 → n2 means that elements of type n1 contain instances of elements of type n2. The number of such instances is described by structural occurrence indicators ("*" means 0 or more, "?" means 0 or 1, "+" means 1 or more and nothing means 1). Figure 6 represents the graph associated with the DTD "MyDTD".

```
<!ELEMENT MyDTD (Movie* , Painting*)>
<!ELEMENT Movie(Actor*, Title, Abstract?,
Category)>
<!ELEMENT Actor (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Abstract (#PCDATA)>
<!ELEMENT Category (#PCDATA)>
<!ELEMENT Painting (Title, Museum?, Author)>
<!ELEMENT Museum (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
```

Figure 5. The DTD "MyDTD"

Because the DTDs are acyclic, their associated graph may be represented as a forest of trees, whose roots correspond to entry points in the graph (nodes without predecessors). Nodes shared in the graph by several trees (e.g. Title in figure 6) are duplicated in order to make these trees independent of each other. For example, the graph in figure 6 produces a single tree, rooted in MyDTD, where the Title node has been duplicated as son of Movie and of Painting. Therefore, the problem of connecting an abstract DTD to concrete DTDs is reduced to the problem of connecting trees of terms. In the following, we will consider DTDs only as trees.

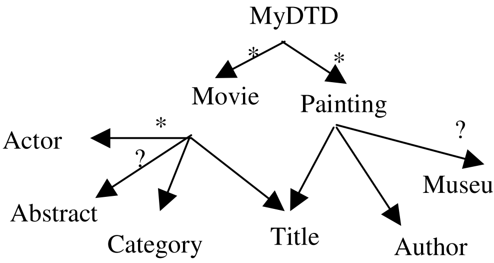

Figure 6. The graph associated with "MyDTD"

3. The connection between the semantic layer and the XML documents

The abstract DTD provides a homogeneous and simple user interface for querying a very large amount of heterogeneous XML documents stored in the repository. In order to obtain relevant answers for queries issued against the semantic layer (e.g. the abstract queries in figure 3), Xyleme must identify the XML documents concerning a given query. The solution chosen by Xyleme is to create and maintain a correspondence between the terms of the abstract DTD and the elements of the real DTDs that describe the structure of the XML documents. Then, each abstract query can be translated into a set of concrete queries over real documents, by simply replacing the abstract terms of the query with the corresponding real element names.

We argued in the previous section that both the abstract DTD and the concrete DTDs are trees of terms. For each term of the abstract DTD, the objective is to keep the set of elements having the same meaning in the concrete DTDs. Actually, because the abstract and the concrete DTDs are trees, each term is identified by its path in the tree, called *abstract path* for a term of the abstract DTD and *concrete path* for a term of a concrete DTD. We call a mapping the relationship between an abstract path and a concrete path. The distinction between terms and paths is necessary, because a term may occur several times in a DTD with different meanings, while a path always identifies a unique element.

In conclusion, the connection between the semantic layer and the XML documents stored in Xyleme is expressed through a set of mappings of the form *abstract_path* ↔ *concrete_path*. These mappings capture the heterogeneity of the various data sources. To each concept of the abstract DTD corresponds a set of heterogeneous terms from the concrete DTDs, having the same meaning. The example in figure 7 shows the mappings between the abstract DTD for Culture and the concrete DTD *MyDTD*. Figure 8 presents the translation of an abstract query by using these mappings. The result is a concrete query over the concrete DTD *MyDTD*.

Culture/Cinema/Movie ↔ MyDTD/Movie
Culture/Cinema/Movie/Actor-Actress ↔ MyDTD/Movie/Actor
Culture/Cinema/Movie/Title ↔ MyDTD/Movie/Title
Culture/Cinema/Movie/Genre ↔ MyDTD/Movie/Category
Culture/Cinema/Actor-Actress/Name ↔ MyDTD/Movie/Actor
Culture/Cinema/Actor-Actress/Filmography ↔ MyDTD/Movie
Culture/Art/WorkOfArt ↔ MyDTD/Painting
Culture/Art/WorkOfArt/Title ↔ MyDTD/Painting/Title
Culture/Art/WorkOfArt/Author ↔ MyDTD/Painting/Author
Culture/Art/Museum ↔ MyDTD/Painting/Museum

Figure 7. Possible mappings between the abstract DTD Culture and the concrete DTD "MyDTD"

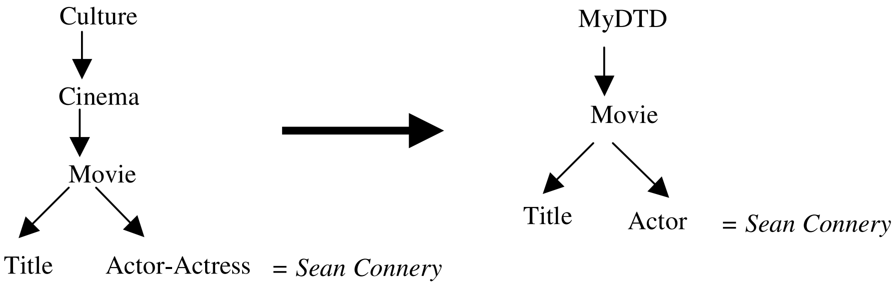

Figure 8: Translation of an abstract query into a concrete query

In Xyleme, the difficulty of connecting the semantic layer and the concrete DTDs comes from the very large number of data sources. Therefore, it is absolutely necessary to generate mappings automatically. Below, we present the principles of our method for automatic mappings generation in Xyleme. Other issues, such as the query translation from abstract to concrete by using mappings, fall out of the scope of this paper. A detailed presentation of query translation in Xyleme is presented in [4].

The automated generation of mappings is based on two kinds of criteria: syntactic/semantic and structural, described respectively in section 3.1 and 3.2.

3.1. Syntactic / semantic matching

Automatic mappings generation is based on term matching. The idea is that a mapping *abstract_path* ↔ *concrete_path* is generated only if the abstract and the concrete terms identified by these paths are semantically related. We check two kinds of relations between terms: syntactic and semantic matching.

Syntactic matching concerns the syntactic inclusion of a term, or of a part of a term, into another one. For example, the concrete term "Actor" is syntactically similar to the abstract term "Actor-Actress". This method also includes techniques for detecting abbreviations, such as "nb" for "Number", etc. For more details, see [19].

Semantic matching is based on the use of extra knowledge available through existing ontologies or thesauri. In our experimentation, we chose the WordNet thesaurus [16].

WordNet groups English nouns, verbs, adjectives and adverbs into sets of synonyms that are linked through semantic relations. Each set of synonyms (synset) represents a concept. A word may belong to several synsets, each one representing a particular sense. Our approach only exploits the semantic links between nouns because all the abstract DTD terms are nouns. The relationships used in our prototype are listed and illustrated in figure 9.

3.2. Dealing with structure

Syntactic / semantic matching between words is not enough to obtain precise mappings. The meaning of a term depends on the place it occupies in the DTD tree, which defines the interpretation context of the word. More precisely, a term may occur several times in the same DTD with different senses. Its meaning may be influenced by the meaning of its predecessors in the tree.

Semantic relationship	Linked terms
Synonymy	Movie - Film, Picture - Image These couples of nouns belong to the same synset
Hyponymy / Hypernymy	Painting - Portrait because portrait is a kind of painting. Portrait is a hyponym of Painting and Painting is a hypernym of Portrait.
Meronymy / Holonymy	Movie - Scene because a movie is composed of several scenes. Scene is a meronym of Movie and Movie is a holonym of Scene.
Hyponymy composed with hypernymy	Painting - Dance because Art is a hypernym of both Painting and Dance.

Figure 9. Illustration of the semantic relationships exploited in our prototype

Our approach is based on a very simple hypothesis: the terms of a concrete DTD tree are either object names (e.g. WorkOfArt), or property names (e.g. Name). We consider that property names have a meaning only in the context of the object that they characterise. So, an object, which is characterised by a set of properties, defines the interpretation context of its properties.

To define if a term is an object or a property node, we use heuristics based on the translation of a DTD into a conceptual database schema. In the relational database context, a conceptual schema is built according to the Entity-Relationship formalism, that naturally helps to determine the classes, called entities, in a domain. An entity is a class of instances that share the same properties called attributes. In our approach, we consider that object nodes are similar to entities and that property nodes are similar to attributes. Given a link A → B, one heuristics used in the translation process is that B can refer to an entity according to the definition given above only if B has son nodes, these ones being interpreted as B's attributes. Such a heuristics leads to consider that leaves in a concrete DTD always are property nodes (they can't refer to entities, then they can't be object nodes). Other heuristics are used to qualify internal nodes in a concrete DTD. The exhaustive translation mechanism is described in [19]. However, very often, internal nodes are object nodes.

A mapping *abstract_path* ↔ *concrete_path* for a concrete path identifying a property name will be valid only if the object characterized by that property is mapped to a predecessor of *abstract_path* (and of course, there is a syntactic/semantic matching between the abstract and concrete terms). This introduces a context constraint into the mapping generation process, because a property name *P* of an object *O* may only be mapped to the subset of abstract terms placed below the abstract terms mapped to *O*, i.e. the context of the object is transmitted to the property. This also means that mappings for object names must be computed first.

For instance, in the concrete DTD *MyDTD*, the terms Title, Author and Museum are properties of the object Painting. The mappings *Culture/Art/WorkOfArt/Title* ↔ *MyDTD/Painting/Title* and *Culture/Art/WorkOfArt/Author* ↔ *MyDTD/Painting/Author* (figure 7) are valid, because there exists a mapping between the object *MyDTD/Painting* and the abstract term *Culture/Art/WorkOfArt*. However, the automatic generation algorithm will not find the mapping *Culture/Art/Museum* ↔ *MyDTD/Painting/Museum* because there is no mapping from *MyDTD/Painting* to a prefix of *Culture/Art/Museum*. The reason, considering our hypothesis, is that it is not clear if *Culture/Art/Museum* is a museum with paintings, because we didn't match Painting with Art in our example in figure 7.

4. Experimentation and results analysis

A prototype, SAMAG (Semi-Automatic MApping Generation), has been implemented in Java. We report here the first results of this work.

The corpus of concrete DTDs used in our experiment is related to the cultural domain. An abstract DTD on this domain has been written and mappings have been established manually between this abstract DTD and all the concrete DTDs of the corpus. We used these manual mappings as a set of reference for evaluating the results of the SAMAG system.

SAMAG is a semi-automatic tool. A user-validation is necessary each time a syntactic relationship between an abstract term and a concrete term is detected. Indeed, two terms linked by such a relationship may refer to different concepts and may not have the same meaning. Only a human is actually able to guarantee the semantic consistency of such a mapping.

4.1. Conditions of the experimentation

SAMAG has been implemented as a modular system. Thus, it can be parameterised. This makes the evaluation of our method easier because results obtained with various parameters from the same inputs can be compared. SAMAG's parameters are:

1. The semantic relationships used by the system to detect a semantic link between abstract and concrete terms, such as synonymy, hypernymy, holonymy, etc.
2. The structural constraints to apply on terms of the concrete DTDs. One can decide to apply or not these constraints. Furthermore, they can be applied only on property nodes in order to consider that property names have a meaning only in the context of the object that they characterise (§ 3.2). But they can also be applied on all the nodes of a concrete DTD considering that the meaning of object names is also influenced by the meaning of their predecessors in the tree.

Considered semantic relationships	All	Synonymy	All
Application of structural constraints	none	All	Property nodes on concrete DTDs
Number of automatically generated mappings	683	32	267
Number of relevant mappings	145 = 21 %	28 = 87,5 %	115 = 43 %
Number of mappings manually and automatically generated	74 among 111 = 66,5 %	18 among 111 = 16 %	52 among 111 = 47 %

Figure 10. Behaviour of the SAMAG system

Notice that in all the cases, the system tests the existence of a syntactic relationship between an abstract term and a concrete term.

When the mappings generation process is finished, all the results are analysed according to (1) the number of relevant mappings among those automatically generated and (2) the number of manually generated mappings retrieved by the system. In particular, we study the number of the following mappings:

1. Mappings that are both manually and automatically generated, such as Culture/Art/WorkOfArt/Artist ↔ School/Painters_List/Painter. Such a mapping is generated thanks to a synonymy relationship between Painter and Artist,
2. Mappings that are manually generated, but not retrieved by the SAMAG system, such as Culture/Art/Artist/ArtMovement ↔ School/Name. Here, there is no semantic relationship in WordNet between ArtMovement and Name, and the detection of a similarity relationship between an abstract term and a concrete term is a precondition to generate a mapping,
3. Relevant mappings that are automatically but not manually generated such as Culture/Art ↔ W4F_DOC/Painting. Indeed writing mappings manually is both labour-intensive and error-prone. Interesting and relevant mappings may have been forgotten,
4. Irrelevant mappings that are automatically but not manually generated, such as Culture/Cinema/Movie/Picture↔W4F_DOC/Painting/image. Such mappings raise the problem of dealing with the polysemy of terms. Indeed, although it exists a synonymy relationship between Picture and Image, Picture is used in a cinema context and Image is used in a painting context. SAMAG is unable to detect such context differences.

4.2. Experimental results

The table in figure 10 gathers some significant statistical results about the mappings generated by the SAMAG system.

The first column in the table describes the results obtained from SAMAG when all the semantic relationships in WordNet are used and no structural constraints are applied on terms of the concrete DTDs. With this configuration, one obtains the maximum number of mappings that SAMAG can generate. The only condition to generate a mapping is the existence of a similarity relationship (semantic or syntactic) between abstract and concrete terms. In such a configuration, the system does not consider the interpretation context of terms, that is why a lot of mappings are not relevant.

The second column in the table describes the results when SAMAG is highly constrained. We consider that the meaning of all terms in the tree is influenced by the meaning of its predecessor. Moreover, only one semantic relationship, synonymy, is checked between terms. It is the relation that introduces the lowest distance between terms. In this configuration, almost all the generated mappings are relevant. However, few mappings are generated, so few manually generated mappings are retrieved.

The last column in the table seems a good trade-off between the number of relevant mappings and the number of mappings automatically retrieved by SAMAG among manually generated mappings. In this configuration, all the semantic relationships in WordNet are used and only property names are considered to have a meaning in the context of their predecessor in the tree.

This experimentation raises a major issue when automatically generating mappings: How can the interpretation context of terms involved in mappings be considered in an automated process ? We proposed a solution based on structural constraints and the analysis of our results must be viewed as a contribution to better understand how to take context into account.

First, we conclude that structural constraints are necessary to limit the number of generated mappings and to produce relevance ones. The number of irrelevant mappings is too high when no structural constraint is applied (145 relevant mappings among 683 when the configuration in column 3 gives 115 relevant mappings among 267).

Our results show also that applying contextual or structural constraints only on property nodes is not enough restrictive. In column 3 of the table in figure 10, more than 50 % of automatically generated mappings are not relevant. An analysis of the irrelevant mappings shows that most of terms involved in bad mappings are objects nodes. Thus, structural constraints must also be applied on object nodes. However, if we apply structural constraints on all the nodes (see results in column 2 in figure 10), the system becomes too constrained and doesn't generate enough mappings.

The main problem in our approach based on structural constraints seems to be the definition of the interpretation context for object nodes. Indeed, the results indicate that property nodes respecting our structural constraints produce relevant mappings, whereas matching object nodes is not satisfactory. In many cases, the context of interpretation of an object node is not only given by its predecessor node within the associated path. For an object node, the number of nodes within the path that are significant to define its context of interpretation depends on it, so this number is variable from an object node to another. For instance, the context of interpretation of the object node Actor in *Culture/Cinema/Movie/Actor* is {Movie} whereas the context of interpretation of the object node Museum in *Culture/Painting/Oil/Museum* is {Painting, Oil}.

The quality of the mapping depends also on the extra knowledge used to establish semantic relationships between terms. WordNet was used as a resource to look for semantic relationships among terms. This choice has been principally motivated by the fact that data in WordNet are easily accessible for automatic applications. It offers a large coverage of general lexicon in English. However, because a term in natural language has often many meanings, WordNet returns synsets for all the senses of a given term. But we don't know how to select automatically the good sense of a term, because it directly depends of its interpretation context. So when the system, for instance, asks to WordNet the set of synonyms of a given term, this one, being not domain-specific, takes synonyms of all the senses of the term. A future work issue will be the choice of the right meaning of a term, based on the interpretation context given by the structural constraints.

Finally, the problem of missing good matches remains even if the sense resolution procedure would work perfectly. According to our tests, 66,5% is the maximum ratio of manual mappings retrieved by SAMAG (see first column of the table). So, about 33% of mappings in this case are never retrieved by the system, because no similarity relationship can be automatically detected between terms. First, we miss mappings such as *Culture/Art/WorkOfArt/Subject* with *Painting/Title*, or *Culture/Art/WorkOfArt/ArtStyl*e with *School/Name* because Subject and Title, respectively ArtStyle and Name, are not semantically similar. The fact that these mappings are relevant is based on the knowledge of the underlying domain. Next, the quality of the real DTDs is very important too. Tags are not always significant nouns, but sometimes they are simple abbreviations. For example, in the Inventory DTD of the French Ministry of Culture, the tag tl-dgn corresponds to title of "document general". It is very difficult to find syntactic / semantic similarities for such tags.

An important remark concerns the influence of mapping generation on query translation. Of course, if a relevant mapping is missed by SAMAG, the query processor will miss the results concerning that mapping. However, additional irrelevant mappings rarely produce irrelevant answers, because query translation need all the mappings in the translation of the query tree to be compatible (i.e. to form a tree that respects the descendant relationship among terms). Yet, reducing the number of irrelevant mappings is important for storage saving reasons, which are critical for web-scale domains.

5. Related Work

The connection problem between information sources and global domain models raises in all information retrieval systems through heterogeneous and independent sources. Substantial work has been made in information integration leading to develop systems such as TSIMMIS [8] or the Information Manifold [14] but until now, little work has really focused on automating the semantic matching process.

Some researchers have considered the automatic generation of mappings between parts of knowledge represented in various representation formalisms, such as between terminological knowledge based systems and relational databases in the context of Federated Databases [9] or also between Object-based and relational systems

[15]. These works focus on the translation of specific formats. Other works whose aim is data translation from one source to another adopt a schema-based approach and automate the connecting process between schemas. So, the TranScm system [17] suggests a (partly) automated matching between schemas based on the name and the structure of the schema elements using a set of predefined generic rules. The schemas being represented as tree structures, the match of two nodes is conditional on the matching of their label (a dictionary is used to detect synonyms) and on the matching of their child nodes. However, this approach assumes that much of the structure of the data in the sources is very similar. So one can specify generic rules usable to automatically connect schemas, many of the structure differences being rather standard.

Other researchers have studied the semantic connection between data issued from multiple information sources. In the Ariadne project [3], building a functional source that normalises the textual format of the data solves such a problem. In the WHIRL approach, this problem is solved by representing the information sources into a highly structured collection of small fragments of texts. Database-like queries are approximated by replacing the equality tests by tests of textual similarities. The system allows queries that integrate information from multiple Web sites, without requiring the extraction and the normalisation of object identifiers that can be used as keys [5]. Finally, some works compute semantic mappings between schemas with a learner-based approach. The idea is that after a set of data sources have been manually mapped to a mediated schema, the system should be able to glean significant information and to successfully propose mappings for subsequent data sources. In the LSD (Learning Source descriptions) system [6], one-to-one mappings for the leaf elements of source schemas are searched this way.

One can also establish a parallel between our objective and research works in Information Retrieval (IR) where relationships between the information needs of users and information in documents are looked for. This can be viewed as a connection problem too. However, IR approaches are based on document indexing and an important task is to derive terms describing the content of documents from the text itself using NLP-based mechanisms [13].

In our work, we are interested in connecting domain descriptions (abstract DTDs) and real DTDs, which can be viewed as schemas of XML documents, all these descriptions being represented as tree structures. So, we don’t manipulate text and we have not a problem of matching primitives issued from different languages. However, DTDs come from various sources. This raises a problem of semantic matching. To solve it, our aim is to match elements of the two descriptions, whatever the place of the elements in trees modelling the descriptions is. So, we have to deal both with heterogeneous names and structures, a problem which arises in language translation when translating text document [18]. Yet, in our case, the inputs are structured representations, not informal representations, then similar techniques are not usable.

Furthermore, an important point in our work in that we need an approach that scales well to large number of sources. This has two effects. First, it is necessary to automate mappings generation. Second, given that the search space becomes very large when the number of queried documents is important, it is necessary to adopt an approach that does not increase the cost of query processing. This explains our choice of generating mappings at compile-time, the mappings being then used at run-time. Such an approach has also been adopted in the SIMS project [1] but not in a context of semantic heterogeneity. For efficiency, source definitions are precompiled into a set of domain axioms. The domain axioms compactly express the possible ways of obtaining the data for each class in the domain model (the best combinations of documents that must be queried). Based on the axioms compiled off-line, the system computes at run-time the most appropriate rewriting for answering a query by simply instantiating the corresponding axioms.

6. Conclusion

We aim to build a data integration system that can explore a repository of XML documents and automatically generate mappings between a mediated schema (the abstract DTD) and schemas of the sources (real DTD). Towards this end, we have implemented a prototype system, called SAMAG, that uses a thesaurus and a structure-based approach to automatically match elements in the two schemas. SAMAG was tested over eight sources (real DTDs) in the cultural domain. The experimental results demonstrate the feasibility of our approach and many interesting open issues in research (e.g. machine learning system to generate only relevant mappings or natural language techniques to automatically detect the interpretation context of terms).

The main conclusion of this experiment is that structural constraints, and more generally the interpretation context of the terms, is very important for the automatic generation of relevant mappings. If some mappings are inevitably missed because of the quality of data and of the lack of knowledge about the domain, the relevance of the generated mappings strictly depends on the structural constraints. The main problem about structural constraints is the difficulty of automatically detecting the interpretation context of object nodes. The maximum precision is obtained if we manually specify for each node in the DTD tree the other nodes that define its interpretation context. This is practically impossible for real DTDs, because their number is supposed to be important. The solution we imagine for the future is to direct context definition toward the abstract DTD, which has a limited number of nodes. Each abstract DTD node will be annotated with the set of nodes that defines its context. The principle of the structural constraints and the algorithm will remain very similar, with much more precise results.

References

[1] Ambite José Luis, Knoblock Craig A., Muslea Ion, Philpot Andrew, "Compiling Source Descriptions for Efficient and Flexible Information Integration", *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, (Orlando, FL, USA), AAAI Press, 1999.

[2] Arens Y., Chee C., Hsu C., Knoblock C. "Retrieving and Integrating Data from Multiple Information Sources", *International Journal on Intelligent and Cooperative Information Systems*, 2(2), pp. 127-158, 1993.

[3] Barish Greg, Knoblock Craig A., Chen Yi-Shin, Minton Steven, Philpot Andrew, Shahabi Cyrus, "The TheaterLoc Virtual Application", *AAAI 2000.*

[4] Cluet. S., Veltri. P., Vodislav D. "Views in a Large Scale XML Repository", *INRIA Verso report 185*, 2000 ftp://ftp.inria.fr/INRIA/Projects/verso/VersoReport-185.ps.gz

[5] Cohen William W., "The WHIRL Approach to Integration : An Overview", in *Proc of Workshop on AI and Information Integration, AAAI 98*, AAAI Press, 1998.

[6] Dan A., Domingos P., Levy A. "Learning Source Descriptions for Data Integration", *In Proc. of the Int. Workshop on the Web and Databases (WebDB),* 2000.

[7] Duschka O., Genesereth "Query planning in Infomaster", *In proceedings of the 1997 ACM Symposium on Applied Computing*, San Jose, USA, 1997.

[8] Garcia-Molina H., Papakonstantinou, Y., Quass D., Rajaraman A., Sagiv Y., Ullman J., Widom J. "The TSIMMIS project: Integration of heterogeneous information sources", *In Journal of Intelligent Information Systems*, 8(2):117-132, 1997.

[9] Goni Alfredo, Blanco José Miguel, Illarramendi Arantza, "Connecting Knowledge Bases with databases: a complete mapping relation", *In Proceedings of the 8th ERCIM Workshop*, Trondheim, Norway, 1995.

[10] Guarino N., Masolo C., Vetere G. "OntoSeek: Content-based Access to the Web", *IEEE Intelligent Systems*, p. 70-80, 1999.

[11] http://www.w3.org/XML/

[12] Knoblock Craig A., Minton Steven, Ambite Jose Luis, Ashish Naveen, Modi Pragnesh Jay, Muslea Ion, Philpot Andrew G., Tejada Sheila, "Modelling Web Sources for Information Integration", Madison, WI, *AAAI 1998.*

[13] Levis D., Jones K. "Natural Language Processing for Information Retrieval", *Communications of the ACM*, vol.39, n°1, 1996.

[14] Levy A. Y., Rajamaran A., Ordille J. "Querying heterogeneous information sources using source descriptions", *In Proc. of the Int. Conf. On Very Large Data Bases (VLDB)*, p.251-262, 1996.

[15] Mena E., Illarramendi A., Blanco J.M. "MAGIC: An Interface for Generating Mapping Information between Object-based and Relational Systems", In *Information Systems Design and Hypermedia*, Cepadues Edition, 1994.

[16] Miller G. A. "WordNet: A lexical Database for English", *Communications of the ACM*, Vol. 38, n°11, November 1995.

[17] Milo T., Zohar S. "Using Schema Matching to Simplify Heterogeneous Data Translation", *In Proc. of the 24th Int. Conf. On Very Large Data Bases (VLDB98*), p. 122-133, 24-27 Aug. 1998.

[18] Mitamura T., Nyberg E., Torrejon E., Igo R. "Multiple Strategies for Automatic Disambiguation in Technical Translation", in *Proceedings of the 8th International Conference on Theoretical and Methodological Issues in Machine Translation*, Chester, UK, August 1999.

[19] Sirot J.-P. "Documents XML et serveurs d'information: approche, techniques et outil d'indexation basés sur l'utilisation d'une ontologie", *Rapport de stage du DEA informatique I3*, Université Paris-Sud, Juin 2000.

[20] Ullman J. "Information Integration Using Logical Views", *In Proceedings of the International Conference on Data Transaction ICDT'97*, 1997.