A Test Platform for the INEX Heterogeneous Track

Serge Abiteboul[1], Ioana Manolescu[1], Benjamin Nguyen[3], and Nicoleta Preda[1,2]

[1] INRIA Futurs & LRI, PCRI, France, firstname.lastname@inria.fr
[2] LRI, Université de Paris-Sud, France
[3] PRiSM, Université de Versailles Saint-Quentin, France

Abstract. This article presents our work within the INEX 2004 Heterogeneous Track. We focused on taming the structural diversity within the INEX heterogeneous bibliographic corpus.
We demonstrate how semantic models and associated inference techniques can be used to solve the problems raised by the structural diversity within a given XML corpus. The first step automatically extracts a set of *concepts* from each class of INEX heterogeneous documents. An *unified set of concepts* is then computed, which synthesizes the interesting concepts from the whole corpus. Individual corpora are connected to the unified set of concepts via *conceptual mappings*. This approach is implemented as an application of the KADOP platform for peer-to-peer warehousing of XML documents. While this work caters to the structural aspects of XML information retrieval, the extensibility of the KADOP system makes it an interesting test platform in which components developed by several INEX participants could be plugged, exploiting the opportunities of peer-to-peer data and service distribution.

1 Context

Our work is situated in the context of the INEX Heterogeneous Track (which we will denote as *het-track* throughout this paper). The het-track is very young: it has been held in 2004 for the first time. The het-track has built a collection of *heterogeneous data sets*, all representing bibliographic entries structured in different XML dialects. In keeping with the INEX terminology, throughout this paper, we will use the term *heterogeneous* to qualify a set of XML documents featuring different sets of tags (or conforming to different DTDs), and this shall not cause confusion.

The het-track collection includes:

- Berkeley: library catalog of UC Berkeley in the areas of computer and information science. The particularity of this data set is to include *several* classifications or codes for each entry.
- CompuScience: Computer Science database of FIZ Karlsruhe.
- BibDB Duisburg: Bibliographic data from the Duisburg university.

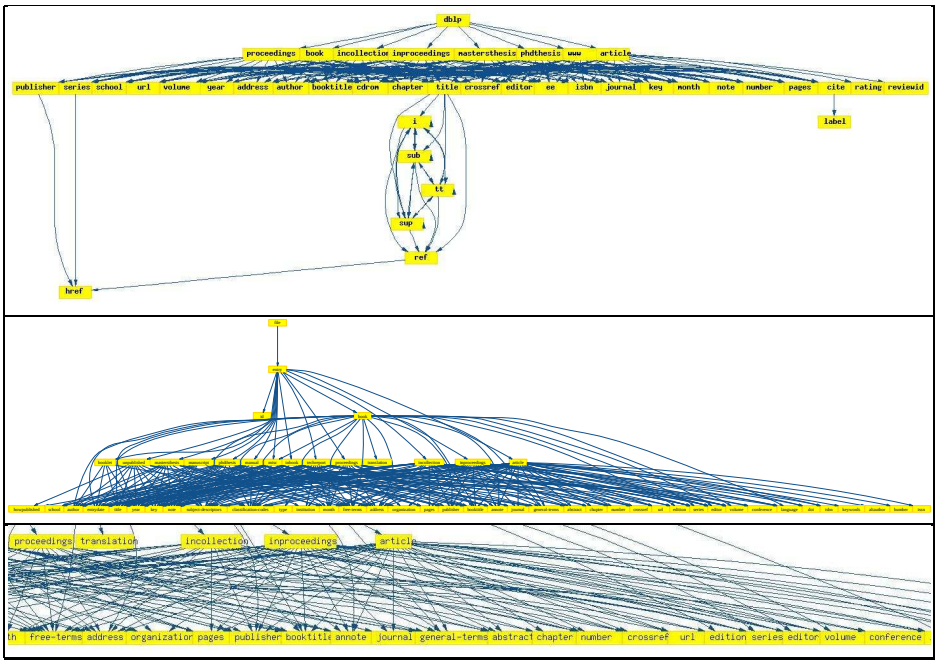

Fig. 1. XSum drawing of the DBLP DTD (top), Duisburg DTD (middle), and zoom-in on Duisburg DTD articles (bottom).

- DBLP: The well-known database and logical programming data source.
- HCIBIB: Bibliographic entries from the field of Human-Computer Interaction.
- QMUL: Publications database of QMUL Department of Computer Science.

A set of topics have also been proposed, which are largely similar (in structure and scope) to those formulated within the relevance feedback track. The topics include:

- *Content-only (CO)* topics, of the form "database query". The answer given by an IR engine to a CO topic consists of XML fragments pertinent to the specified keywords.
- *Content-and-structure (CAS)* topics, such as
  //article[about(.//body, "XML technology")]
  The answer given by an IR engine to a CAS topic consists of XML fragments pertinent to the specified keywords, and using the structure criteria of the topic as hints.

Answering an IR query on a structurally heterogeneous corpus raises two main challenges. First, the relevance of a data fragment for a given keyword or set of keywords must be computed; this task is no different from the main relevance

assessment track. Second, the structural hints present in the topic, in the case of CAS topics, must be taken into account.

In the presence of a heterogeneous set of XML documents, the second task becomes particularly difficult. This is due to the fact that semantically similar information is structured in different XML dialects; furthermore, DTDs may or may not be available for the documents. The work we present has specifically focused on this second task.

Contributions Our work within the het-track makes the following contributions.

First, we present an approach for *integrating the heterogeneous structures* of the heterogeneous data sources under an *unified structure*. This approach relies on simple semantic-based techniques, and on our experience in building semantic-based warehouses of XML resources [2, 3]. The result of this integration on the het-track corpus is *a unified DTD, and a set of mappings* from individual sources to this DTD. CAS topics against the het-track corpus can now be expressed in terms of the unified DTD, and get automatically translated into a union of topics over each data set. Thus, solving a CAS topic on a heterogeneous corpus is reduced to solving several CAS topics against the individual component data sets.

Second, we present XSum [20], a free XML and DTD visualization tool, that we developed as part of our work in INEX. XSum helped us get acquainted with the complex structure of the heterogeneous collection, and devise semi-automatic integration strategies.

Finally, we outline the architecture of a peer-to-peer platform for processing XML queries or IR searches, over a set of distributed, potentially heterogeneous XML data sources. This platform has the advantage of being *open* and *inherently distributed*, allowing to take advantage of the data sources and capabilities of each peer in the network in order to solve a given query or search. In particular, we show this platform may be used as a testbed for the XML IR methodologies developed within the het-track, by allowing to test and combine the various implementations of the about functions developed by INEX participants.

This document is structured as follows. Section 2 describes our semantic-based approach for XML information retrieval over a heterogeneous corpus. Section 3 details the result we obtained by applying this approach on the INEX het-track corpus. Section 4 outlines the peer-to-peer generic platform we propose, and describes how it could be used as a testbed for the het-track in the future. Section 5 compares our work with related ones, while Section 6 draws our conclusion and outlines future work.

2 Approach

Dealing with structural diversity in heterogeneous sources has been a topic of research in the field of databases and in particular of *data integration*; the basic concepts in this area can be found, e.g., in [16]. The purpose of a data integration system is to provide the user the illusion of a single, integrated database, on

which the user can pose queries (in our case, IR queries, or topics). Behind the uniform interface, the system will process these queries by translating them into the formats specific to each data source, processing them separately, and integrating the results into a single one.

Traditionally, data integration operates at the level of *schemas*. A source schema characterizes the structure of each data source, and an integrated schema is provided to the user. This approach has been thoroughly investigated in the case of relational data sources and schemas [11, 6].

In the case of heterogeneous, complex, potentially schema-less data sources, this approach is no longer applicable. Instead, we chose to draw from the experience obtained in *semantic-based data integration* [6, 5], to integrate sources pertinent to a specific domains, such as the het-track corpus, under a single *conceptual model*. The building bricks of our conceptual model are:

- *Concepts*, which are the notions of relevance for a given application. For instance, in the het-track corpus, useful concepts are: publication, author, etc.
- *IsA* relationships represent specialization relationships between concepts. For instance, book IsA publication represents the fact that books are a kind of publication. IsA relationships are also known under the name of *hyponymy*.
- *PartOf* relationships represent composition (aggregation) relationships between concepts. For instance, title PartOf book represents the fact that a title is a component of a book. PartOf relationships are also known under the name of *meronymy*

Since its inception, XML has been hailed as a *self-descripting* data format: since the set of markup tags is by definition extensible, typical XML applications use semantically meaningful tag names. This was part of the indended usage for XML in data management, and indeed this choice is made in most popular XML applications. An outstanding example is XSL, the XML-based stylesheet language: an XSL program is written based on a set of XML types with meaningful names, such as if/else, apply-templates, select etc. An inspection of the DTDs found on the "XML Cover Pages" leads to the same conclusion: XML flexibility is typically exploited to encode meaning into tag names. Furthermore, tag nesting is often used to reflect nesting of application objects. Thus, while tags by themselves are insufficient to capture complex application semantics, they are at least a first step in that direction.

As a consequence of this "pseudo-semantic description" approach, in several data management applications the need arises for integrating data sources of which we only know their XML syntax (or their DTDs). This approach has been taken for instance in [5], and has been further studied and refined in [8].

Thus, our approach starts by extracting a conceptual model from each source. For the sources for which DTDs are available, the process is straightforward: we extract a concept for each type in the DTD, including element and attributes (among which we do not make a distinction). For sources for which DTDs are not available, we start by extracting a "rough" DTD, including all elements and

attributes. Whenever we encounter in the data an element labeled l_1 as a child of an element labeled l_2, we mention in the DTD that the type l_1 can appear as a child of the type l_2. After having extracted this DTD, we compute from it a set of concepts as in the previous case.

At the end of this stage, we have obtained a set of conceptual data source models. Our purpose then is to construct a unified conceptual model characterizing all sources, and mappings between each conceptual model to the unified one.

Extracting the unified conceptual model To build the unified conceptual model, we identify groups of concepts (each one in different conceptual source models) that represent semantically similar data items. We do this in a semi-automatic manner, as follows.

First, the names of concepts from different source models are compared for similarity, to identify potential matches. This is done automatically, with the help of WordNet [7]. We have used WordNet outside the peer; it can be easily wrapped as a Web service local to the integration peer. If simple matches such as the one between book (DBLP) and book (HCI BIB) can be automatically detected, more subtle ones such as the similarity between editor (HCI BIB) and Edition (Berkeley) require the usage of tools such as WordNet. Having identified clusters of concepts which potentially represent the same thing, we create one concept in the unified model, for each cluster of source model concepts above a given similarity threshold; human intervention is required at this point in setting the similarity threshold.

At the end of this process, it may happen that some source model concepts have not been clustered with any others. This may be the case, for instance, of concepts called Fld012, Fld245, etc. from the Berkeley data source. These concepts are difficult to cluster, since their names (standing for Field number 012, Field number 245, etc.) do not encapsulate the meaning of the concept, instead, this meaning is included in plain-text comments preceding to the corresponding element type definition. To deal with such concepts, we need to capture the DTD comments preceding the type definition, and feed those descriptions to the word similarity-based clustering. This way, we may learn that Fld245 stands for "Title Statement", and cluster Fld245 with similarly named concepts from other DTDs.

Once the clusters of similar (and supposedly semantically close) concepts have been extracted, we create a concept for each such cluster, in the unified conceptual model.

Extracting mappings between the source and unified conceptual models We add an IsA relationship going from each source model concept, to the unified model concept that was derived from its clusters. If a source model participates to several clusters, this may yield several IsA relationships.

3 Contributions

In this section, we report on the results that we obtained in our work in the framework of the het-track.

3.1 Unified model for the het-track corpus

In this section, we discuss the techniques used to construct a unified DTD in order to query the documents of the heterogeneous track. An important factor is that the documents are all about the same application domain: scientific bibliography. Five of them are quite verbose, and the labels are self descriptive, while one (Berkeley data set) has labels that convey no semantic signification whatsoever. Some examples of such labels would be: Fld001, Fld002, etc.

In this article, we do not take into account this DTD, therefore the unified DTD we propose does not include elements from the Berkeley data set for the moment. We report on our experience with the Berkeley data set, based on using our XSum tool, in Section 6.

The method used in order to determine a unified DTD is the following :

- We first of all create mappings between elements whose tags are similar, which feature similar children element types, but that originate from different DTDs. For instance, we might find two article elements, one from DBLP, the other from BibDB Duisburg. If there are several elements with the same (or very similar) type definition in multiple DTDs, we group them all together. These are one to one mappings, and the output of this phase is a group of clusters of syntactically close elements.
- For each cluster, we then check the parent nodes, and group them together in a new parent cluster.
- For all these automatically constructed clusters, we manually check the correctness of these groupings, and cho0se a name for the cluster, generally of the form nameOfElementC.

We provide the unified DTD on our website [18].

Using the unified DTD : The Unified DTD is to be used when asking queries over the heterogeneous data set. The querying mechanism is as follows.

- The INEX queries must be written taking into account the unified DTD. The names of the elements appearing in the unified DTD should be used to formulate queries against the whole het-track corpus. We call such a query a *generic query*.
- The generic query is then converted into specific queries, with a specific structure for each database, and the queries are then run seperately on all the databases.
- Given the unified DTD, the answers returned are clustered together in a common structure, in order to use only a single DTD for browsing means.

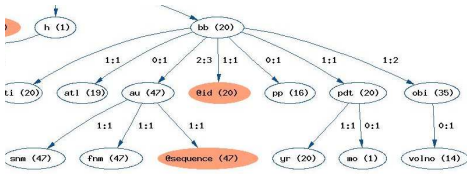

Fig. 2. Fragment of a path summary computed from an article in the INEX main corpus (IEEE CS).

3.2 XSum: a simple XML visualization tool

We have developed a simple XML visualization tool, called *XSum* (for *XML Summary Drawer*). XSum can be used in two ways.

Drawing path summaries Given an XML document, XSum extracts a tree-shaped structural summary of the document, and draws it. This structural summary contains a node for each distinct path in the input document [13], and is the equivalent of a strong DataGuide [9] for XML data (DataGuides were initially proposed for graph-structured OEM data). XSum enhances this structural representation with:

- Node counts: XSum records the number of nodes on a given path in the XML document, and correspondingly may show this number in the summary node corresponding to that path.
- Leaf types: XSum attempts to "guess" the type (String, integer or real number), of each leaf node (which can be either a text node, or an attribute value), and depicts the corresponding node in a color reflecting its type.
- Edge cardinalities: for every path p/l, where p is a path and l is a label, XSum records the minimum and maximum number of children on path p/l that an element on path p may have. This holds also for text children.

A sample summary representation produced by XSum from a XML-ized article from the INEX IEEE CS corpus is depicted in Figure 2. The fragment shown here reflects the references at the end of an article, including authors, titles, and publication information for the relevant references. If a an element type is recursive, the summary drawn by XSum will unfold the recursion: it will construct several nodes corresponding to the recursive type, nested within each other, up to the maximum nesting level actually encountered in the data.

Drawing DTDs When given a DTD, XSum draws a simple graph, representing each attribute or element type from the DTD as a node, and adding an edge from a node to another whenever a type may appear inside another in the DTD. If a an element type is recursive, the DTD graph will only include one node for this type, and that node will participate in a cycle.

Figure 1 shows the drawing extracted by XSum from: the DBLP DTD, the DTD of the Duisburg data source, and a zoomed-in fragment around the node corresponding to the "article" type in the Duisburg data source.

Structural Clusterering of DTDs Graphs corresponding to DTDs tend to have relatively few nodes, but large number of edges, crossing each other in the drawing (as shown in Figure 1), which may make the image difficult to read. To cope with this problem, we have devised a *structural DTD clustering technique* which reduces the number of nodes (and edges) in a DTD graph. The clustering is performed as follows. All DTD nodes sharing the same set of parents are clustered together; an edge is drawn between two clusters, if some nodes in the parent cluster are parents of all the nodes in the child cluster. Notice that the structural clustering performed by XSum takes place *within* a single DTD; it has nothing in common with the semantic clustering performed *across* DTDs and described in the previous section.

Clustered DTD graphs are much more readable than simple DTD graphs. As an example, Figure 3 shows the clustered graph produced for the Duisburg DTD. This readability comes at the price of some loss of precision (since they do no longer show the exact set of parents of each node).

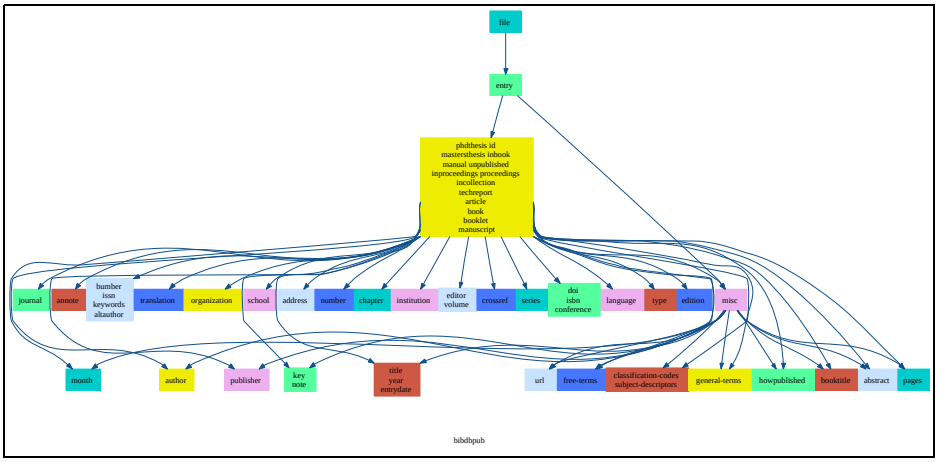

Fig. 3. Clustered DTD graph for the Duisburg data set.

From our experience using XSum with the INEX standard and heterogeneous corpus, we draw some remarks. Both DTD and summary drawings tend to be large for documents of the complexity we are dealing with, typically larger than the screen or a normal printer format. Understanding the image requires "sliding" over it to see one part at a time. To simplify path summaries, we have introduced in XSum options allowing to omit leaf nodes and/or cardinality annotations, which simplifies the graphs. To simplify DTD graphs, we introduced structural clustering. We welcome the feedback of INEX participants on how to enhance XSum's XML and DTD drawing logic, to produce more helpful images.

XSum is implemented in Java, and is based on GraphViz, a well-known free graph drawing library developed at AT&T. XSum is freely available for download from [20].

Further info and graphs The graphs produced by XSum, for all DTDs in the het-track corpus, are available at [18].

4 The KadoP platform

In this section, we briefly describe the KadoP peer-to-peer XML resources management platform, which serves as the framework for our work. A more detailed presentation can be found in [3].

The KadoP platform allows constructing and maintaining, in a decentralized, P2P style, a warehouse of *resources*. By resource, we mean: data items, such as XML or text documents, document fragments, Web services, or collections; semantic items, such as simple hierarchies of concepts; and relationships between the data and semantic items. KadoP's functionality of interest to as are:

- *publishing* XML resources, making them available to all peers in the P2P network;
- *searching* for resources meeting certain criteria (based on content, structure as well as semantics of the data).

KadoP leverages several existing technologies and models. First, it relies on a state-of-the art Distributed Hash Table (DHT) implementation [19] to keep the peer network connected. Second, it is based on the ActiveXML (AXML) [17] platform for managing XML documents and Web services. A full description of ActiveXML is out of the scope of this work, see [1]. For our purposes here, AXML is an XML storage layer, present on each peer.

The KadoP data model comprises the types of resources that can be published and searched for in our system. We distinguish two kinds of resources: *data items*, and *semantic items*. *Data items* correspond to various resource types:

- A *page* is an XML document. Pages may have associated *DTDs* or *XML schemas* describing their type; we treat DTDs as sources of semantic items (see further). Other formats such as PDF can be used; we ignore them here.
- We consider data with various granularities. Most significantly, we model: *page fragments*, that is, results of an XML query on a page, and *collections*, as user-defined sets of data items. Collections can be viewed as an extension of the familiar concept of Web navigator bookmarks: they are defined by the user who gives them a name, and can gather in a collection any kind of data items which, from the viewpoint of the user, logically belong together. Inside pages, we also consider element *labels*, attribute *names*, and *words*.
- Finally, a *web service* is a function taking as input types XML fragments, and returning a typed XML fragment.

Any data item is uniquely identified by an PID (peer ID) and a name. The PID provides the unique name (logical identifier) of the peer that has published the data item, and where the item resides; names allow distinguishing between data items within the peer. Data items are connected by PartOf relationships, in the natural sense: thus, a word is part of a fragment, a fragment part of a page etc. Furthermore, any type of data items can be part of collections. A data item residing on one peer may be part of a collection defined on another peer (just like for bookmarks, adding a page to a collection does not entail its replication).

Semantic items consist of *concepts*, connected by two types of relationships: IsA, and PartOf. A graph of concepts, connected via IsA or PartOf links, is called a *concept model*. We derive a source concept model from each particular data source, as described in Section 2.

InstanceOf statements connect data items with concepts. In particular, all elements from an XML document, of given type τ (obtained as the result of the XPath expression $//\tau$), are implicitly connected by InstanceOf statements to the concept derived from the type τ.

The KadoP query language allows retrieving *data items*, based on constraints on the data items, and on their relationship with various concepts. Queries are simple tree patterns, and return the matches found for a single query

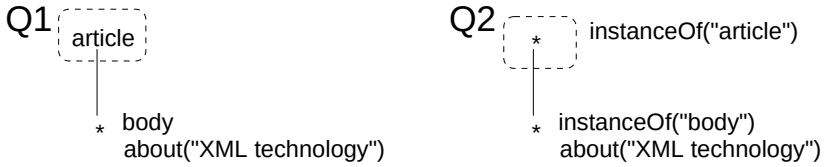

Fig. 4. Sample KadoP queries.

node (in the style of XPath and the CAS INEX topics). For instance, the query in Figure 4 at left allows retrieving all "article" elements such that they have a "body" element, and the body is about XML technology. This corresponds to the sample CAS topic in Section 1. The dashed box designates the node for which matches will be returned.

Such a query, however, needs specific element tag names for its nodes. In the case of the heterogeneous corpus, such queries are no longer helpful, due to the presence of different element tag names corresponding to semantically similar data objects.

The approach we take for solving INEX heterogeneous CAS topics is based on the unified conceptual model. The idea is to drop name conditions from the queries, and instead use conditions of the form "instanceOf c", where c is a concept from the unified model. On our example query, this leads to the KadoP query at right in Figure 4, where we assume that "article" and "body" are part of the unified conceptual model. This query is processed as follows:

1. The elements directly declared as instance of the concepts "article" and "body" are found.
2. We search for concepts c_a such that c_a IsA "article", and concepts c_b such that c_b IsA "body". This will lead to retrieving all the concepts from the source concept models, which have been mapped to the unified concepts "article" and "body".
3. We search for elements declared as instances of the concepts c_a and c_b obtained as above.

These steps lead to matching the structural conditions posed by the CAS query against the heterogeneous corpus. They do not, however, apply the "about" condition, since implementing this condition is out of the scope of our work. We next explain how others' implementations of the "about" function could be plugged in our work.

Integrating "about" functions In the KADOP framework, "about" can be integrated as a Web service, offered by one or several peers. The implementation of this function is typically complex. From the KADOP perspective, all that is needed is that one or several participants make available a Web service named "about", obeying to a well-defined interface. Then, the KADOP query processor can invoke one of these services to evaluate the pertinence of an XML fragment for a given set of keywords. The user may specify which service to use; this is helpful when we want to compare the results of different implementations. Or, she may let the system choose an implementation.

It is worth stressing that the KADOP framework is based on a concept of *openness* and *extensibility*: new data sets, new concepts, or new semantic statements can be added by any participant, and refer to any data item on any peer. Finally, the KADOP framework is by nature distributed: any Web service (thus, any "about" function) can be invoked on XML fragments originating from any peer.

5 Related work

We have proposed in this paper a method of integrating heterogeneous DTDs related to the same domain, into a unified DTD. Our work is related to projects on semi-automatic schema matching. In the domain of semi-automatic schema matching, we may distinguish three main research directions related to our work:

- given two schemas S_1 and S_2, compute the matching that associates label elements in schema S_1 with other label elements in schema S_2.
- given two schemas S_1 and S_2, create mappings between the elements of S_1 and the elements of S_2 in the form of views.
- given a set of schemas, generate one or more integrated schemas.

Our approach is related to the first direction as mappings between two DTD sources are derived based on the semantic and syntactic resemblance between

nodes. The KADOP query engine exploits not only mappings between the unified DTD and each DTD source, but also mappings between two DTD source schemas.

We have proposed a semi-automatic method of computing a unified DTD. This work, related to the third research direction, is based on clustering similar elements. As related integrating schemas system based on clustering techniques, we mention ARTEMIS [4].

The particularity of our integration problem consists in the type of our input schemas: DTD schemas. These schemas do not have rich semantics associated with the edges between concept nodes. In the XSum project, we are now investigating, heuristics in order to add more semantic information to the DTD structure. Conceived first as a method of pretty drawing a DTD, we have defined a clustering method that has good properties of grouping related concepts together. This is interesting, because it implies that semantic relationships between nodes may be verified on a smaller graph instance. Other heuristics that transform the DTD-s into ER database schemas have been investigating in [6], [14], [12].

For the het-track collection, we have defined a single (unified) abstract schema, and mappings between concepts in the in the unified DTD and concepts in the various DTD sources, as each DTD was referring to the same topic.

In the case of a collection that contains schemas and resources of different domains (which may be the case of a peer to peer application), we may build a tool that semi-automatically defines unified DTDs by clustering DTDs of the same domain. We may benefit of works done in the ONION [15] project, that is heavily based on the existence of rich semantic relationships between the nodes of the schemas.

The second research direction hasn't been investigated in this paper, although the KADOP query language may handle mappings in the form of a concept node in the DTD associated to a view (KADOP query). An automatic method of deriving such mappings may benefit of works done in Xyleme [6] (path to path mappings), or CLIO [10].

6 Conclusion and perspectives

We have presented our work focused on building an unified DTD for the data sets of the het-track. We have produced an unified DTD including all but the Berkeley data set, and we have developed a simple XML visualization tool which helped us get acquainted with the various data sets. We have furthermore presented an approach for formulating CAS INEX topics against a heterogeneous data corpus, based on our KADOP platform.

Our next step is merging the DTD of the Berkeley data set into the unified DTD.Aas explained in Section 3.1, the tag names encountered in this data set are meaningless; however, tag meaning can be found in comments appearing in the DTD, just before the element type definition appearing in the DTD.

We attempted to cluster the DTD, but we were not able to parse it; thus, we extracted our own DTD, and clustered this one. We made several remarks.

First, the original DTD features much more element types (around 700) than the data set actually uses (around 200). Thus, the extracted DTD is easier to grasp.

Second, in some cases on the Berkeley data set, our structural clustering criteria has (quite surprisingly) clustered nodes representing semantically similar things. For instance, we obtained a cluster of 8 nodes representing variants of a publication's title, and a cluster of 33 nodes representing various numerical publication codes. However, in other cases, our parent-based clustering has grouped together nodes that do not *represent similar things*, but *are part of the same data subset*: namely, there is a "Main" cluster, grouping nodes such as Fld100 ("Main entry, personal name") and Fld111 ("Main entry, meeting name"), although personal and meeting names do not stand for the same thing. In this case, semantic clustering (using the comments, since the tag names themselves are meaningless) will disagree with structural clustering. Semantic clustering may correctly group "Main entry, personal name" with Fld700 ("Added entry, personal name"), since they represent similar things. However, this is only our intuition; a librarian's viewpoint may be quite different.

Third, we noticed also a (single, small) cluster where neither the tags, nor the accompanying comments convey any useful information. This is the case of a set of fields whose comments read "XXX Local Holdings Information for 9XXX"; we do not expect automatic processing of such data to yield meaningful results.

In a more general perspective, we intend to develop our approach into an easy-to-use integration platform, in order to include any other bibliographical semi-structured databases.

Acknowledgements The authors are grateful to the anonymous INEX referee, whose careful comments helped improve the readability of the paper.

References

1. Serge Abiteboul, Omar Benjelloun, and Tova Milo. The ActiveXML project: an overview. Gemo research report no. 344, 2004.
2. Serge Abiteboul, Gregory Cobéna, Benjamin Nguyen, and Antonella Poggi. Construction and maintenance of a set of pages of interest (SPIN). In *Bases de Donnees Avancees*, Evry, 2002. Informal proceedings only.
3. Serge Abiteboul, Ioana Manolescu, and Nicoleta Preda. Constructing and querying a peer-to-peer warehouse of XML resources. In *Proceedings of the Semantic Web and Databases Workshop (in collaboration with VLDB)*, Toronto, CA, 2004.
4. Silvana Castano, Valeria De Antonellis, and Sabrina De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, 2001.
5. Sophie Cluet, Pierangelo Veltri, and Dan Vodislav. Views in a large scale XML repository. In *VLDB*, pages 271–280, 2001.

6. Claude Delobel, Chantal Reynaud, Marie-Christine Rousset, Jean-Pierre Sirot, and Dan Vodislav. Semantic integration in Xyleme: a uniform tree-based approach. *IEEE Data and Knowledge Engineering*, 44(3):267–298, 2003.
7. Christine Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
8. Gloria Giraldo. Automatic ontology construction in mediator systems. Ph.D. thesis, University of Orsay, France, 2005.
9. R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, pages 436–445, Athens, Greece, 1997.
10. Laura M. Haas, Renée J. Miller, B. Niswonger, Mary Tork Roth, Peter M. Schwarz, and Edward L. Wimmers. Transforming Heterogeneous Data with Database Middleware: Beyond Integration. *IEEE Data Engineering Bulletin*, 22(1):31–36, 1999.
11. Alon Y. Levy. Logic-based techniques in data integration. *Logic Based Artificial Intelligence*, 2000.
12. Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *The VLDB Journal*, pages 49–58, 2001.
13. I. Manolescu, A. Arion, A. Bonifati, and A. Pugliese. Path Sequence-Based XML Query Processing. In *Bases de Données Avancées (French database conference)*, Montpellier, France, 2004. Informal proceedings only.
14. P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proc. of the 2nd Int. Conf. On Information FUSION'99*, 1999.
15. Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. *Lecture Notes in Computer Science*, 1777:86, 2000.
16. Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, 1992.
17. The ActiveXML home page. Available at www.activexml.net, 2004.
18. Gemo and PRiSM at the INEX heterogeneous track. Available at www-rocq.inria.fr/gemo/Gemo/Projects/ INEX-HET, 2004.
19. The FreePastry system. Available at www.cs.rice.edu/CS/Systems/Pastry/FreePastry/, 2001.
20. XSum: The XML Summary Drawer. Available at www-rocq.inria.fr/gemo/Gemo/Projects/SUMMARY, 2004.