MuMIe: A New System for Multimedia Metadata Interoperability

Samir Amir, Yassine Benabbas, Ioan Marius Bilasco and Chabane Djeraba
LIFL UMR CNRS 8022, University of Lille1 and Telecom-Lille1
Villeneuve d'Ascq, France
{samir.amir, yassine.benabbas, marius.bilasco, chabane.djeraba}@lifl.fr

ABSTRACT

The recent growth of multimedia requires an extensive use of metadata for their management. However, a uniform access to metadata is necessary in order to take advantage of them. In this context, several techniques for achieving metadata interoperability have been developed. Most of these techniques focus on matching schemas defined by using one schema description language. The few existing matching systems that support schemas from different languages present some limitations. In this paper we present a new integration system supporting schemas from different description languages. Moreover, the proposed matching process makes use of several types of information (linguistic, semantic and structural) in a manner that increases the matching accuracy.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Systems Metadata, Standards, Interoperability
; H.2.4 [**Database Management**]: Systems—*Multimedia databases*

General Terms

Algorithms, Performance, Experimentation, Verification

Keywords

Metadata integration, interoperability, semantic similarity

1. INTRODUCTION

The ubiquitous presence of multimedia resources in our lives has generated an increasing interest in the use of metadata to improve the efficiency and effectiveness of retrieval, filtering and managing procedures of these types of contents. Metadata is a machine processable data that describes different types of information: multimedia contents (e.g., video, image, audio, etc.), semantics of these contents (e.g., concept in image, context of video, etc.), characteristics of devices consuming or transmitting these contents (e.g., networks, TV, mobile, etc.) and consumer characteristics (e.g., consumer profile, consumer preference, etc.).

By anticipating the increase of metadata in the upcoming years, we can foresee that it will become more and more difficult to achieve a uniform access to media objects. This is due to the number of independent metadata communities, which combine terms from multiple vocabularies and use different structures for metadata descriptions. In this context, *metadata interoperability* becomes crucial.

Over the last decades, a variety of interoperability techniques has been proposed. Introducing a standardized metadata schema (e.g., Dublin Core, CIDOC/CRM, MPEG-7, MPEG-21, FGDC, IMS, etc.)[10] and standardized schema definition language (e.g., XML Schema, UML, OWL, RDF Schema, etc.) [9] is one of them. But due to strategical or political reasons it is impossible to adhere to a single standard or schema definition language. Consequently, several metadata integration solutions have been proposed and are analyzed in [9]. Most of these works focus on the creation of a core ontology which contains common information on metadata to be integrated. This ontology acts as a mediated schema, which is the common interface used for querying all metadata encoded in different formats. After designing this core ontology, a manual mapping is performed between the latter and other metadata formats.

The works mentioned above have not been a real success since the integration process is performed manually, which is costly and time consuming. Besides, the integration must be updated every time a new metadata format appears. In this context, several semi-automatic techniques have been developed to facilitate the integration process. *Schema matching* techniques play a central role in these approaches [5].

Due to the complexity of schema matching, it was mostly performed manually by human experts. However, manual reconciliation tends to be a slow and inefficient process especially in large-scale schemas (e.g., MPEG-7, MPEG-21, etc.) and dynamic environments such as multimedia where new metadata standards are appearing constantly. Another problem that should be taken into consideration is that metadata are heterogeneous on two levels: *schema* and *schema definition language*. The structural and semantic heterogeneity on the schema level occurs when the same information is represented differently on different schemas (e.g., naming conflicts, multilateral correspondence, abstraction level incompatibility, etc.). The schema definition language heterogeneity is due to the substantial structural and semantic discrepancies of schema definition languages. For instance, several XML Schema structural descriptions can not be expressed using RDF Schema (RDFS) and several semantic descriptions of the latter are not supported by XML Schema (XSD) [9].

This paper aims at automating the integration process of metadata in order to achieve interoperability. This is done by proposing a new integration system named MuMIe (Multi-level Metadata Integration). The proposed system occurs on the schema level and schema definition language level. Moreover, our system combines several types of semantic and structural information in a manner that increases the matching accuracy. The paper is organized as follows: Section 2 discusses the limitations of existing schema matching techniques and describes related work. In Section 3, we describe the proposed integration system. An evaluation study is presented in Section 4. Finally, Section 5 gives concluding remarks and future work.

2. RELATED WORK

Since metadata heterogeneity can occur on the schema definition language and the schema levels, it is necessary to specify the mappings for each level. To do so on the schema definition languages level, several methods have been developed. Examples include the work done in [7][16], in which the authors propose conversion rules for XML Schemas to OWL. These rules insure the capture of a part of XML Schema implicit semantics. The Ontology definition metamodel specification [12] offers a set of metamodels and mappings for translating between the UML metamodel and ontology languages such as RDF Schema and OWL. The authors in [15] present an approach for automatically generating RDF Schema from XML Schema specification. Other approaches for language translation have been proposed, some of them are analyzed in [9]. Because of the substantial structural and semantic discrepancies among schema definition languages, the works mentioned previously have not been very successful as the translation from one language into another causes the loss of valuable semantic and structural information.

Concerning the heterogeneity problem on the schema level, tools and mechanisms are needed in order to achieve interoperability. These tools and mechanisms must resolve the semantic and structural heterogeneity of schemas and align terms between metadata. To do so, several schema matching techniques have been proposed over the last decade, but up to now few existing systems aim at being general and support schemas from different languages, among them we can note: Similarity Flooding [14](relational, XML and RDF), Cupid [4](relational and XML) and S-Match [5] (relational, XML and Ontologies). These approaches do not make use of most of the structural and semantic information (e.g., equivalence properties, generalization features, disjointness classes, etc.). Consequently, it is difficult to detect the complex matching such as one-to-many and many-to-many. Moreover, the main drawback of these methods is how to use the structural information. For instance, the authors in [4] consider that much of the information content is represented in leaves and that the leaves have less variations between schemas than the internal structures. Thus, the similarity of inter-nodes is based on the similarity of their leaf sets. This is not always true as we can find equivalent concepts occurring in completely different structures, and completely independent concepts that belong to isomorphic structures. The drawback of the method developed in [14] that it is based on the idea of similarity propagation, the basic concept behind the algorithm is that adjacency contributes to similarity propagation. Thus, the algorithm will perform unexpectedly in cases when the adjacency information is not preserved. The work done in [8] is limited to tree-like structure and does not consider properties or roles.

Motivated by the above challenges, we discuss below a simplified approach for achieving the interoperability on two levels. The approach can be used to integrate several heterogeneous schemas, defined using different definition languages. We combine several syntactic, semantic and structural resources available on schemas to detect the mappings between metadata terms. Moreover, our approach overcomes the structural gap of the matching systems mentioned above by using several similarity contexts [11].

3. PROPOSED APPROACH

This section shows the two main parts of the proposed system. The first one is the *projection* step which is responsible for achieving interoperability on the language level. This is done by the transformation of all schemas into a common representation model. The second part is the *matching* step which is responsible for finding the correspondences between metadata terms by using several types of information.

3.1 Projection

As mentioned in Section 2, up to now, few existing matching systems support schemas from different languages. In this section, we propose transformation rules that allow the projection of different schema languages on the same representation space. This space is an abstract model that serves as a foundation to represent conceptually several types of schemas whatever their description language. We model schemas as a directed labeled graph (Section 3.1.1) and save semantic and structural information in order to use them for the matching process (Section 3.1.2). Figure 1 illustrates the projection process. Note that examples on this paper are limited to three description languages (XML Schema, RDF Schema and OWL) [10], since they are commonly used by metadata communities and present a significant heterogeneity (a high structural description for XSD and a high semantic expressiveness for RDF Schema and OWL).

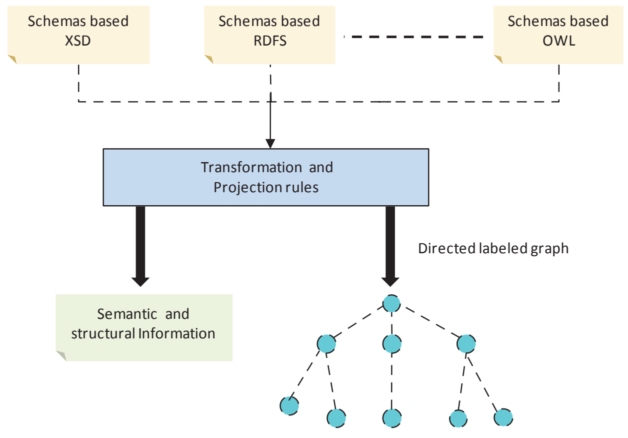


Figure 1: Projection.

3.1.1 Schema graph representation

Due to the high heterogeneity of schema definition languages, it is not possible to find one representation model which supports all schema features. In our approach, we model these schemas as a directed labeled graphs representing only basic schema concepts and properties linking between them. These two entities are the common and basic information for all description languages. In order to create the graph, the description concepts of schema definition languages must be known. For instance, all classes (*rdfs:class*) and properties (*rdf:property*) in RDF Schema must be represented by nodes. Table 1 shows the basic concepts and properties for the three different languages. Concerning the languages based on a taxonomy classification (e.g., XML Schema), we use the idea proposed in [15] to capture the semantics of implicit properties. So, the name of property node is the combination of parent and child nodes.

We categorize nodes into *normal nodes* and *property nodes*. Normal nodes correspond to the basic concepts (e.g., *xsd:complexType*, *rdfs:class*, *xsd:element*, etc.) and property nodes correspond to properties linking between schema concepts (e.g., *rdf:property*, *owl:ObjectProperty* , etc.).

Table 1: Language Mapping Rules

Node Types	XSD	RDFS	OWL
Normal Nodes	xsd:element xsd:attribute xsd:attributeGroup xsd:group	rdfs:class	owl:class
Property Nodes	parentName and childName	rdf:property	owl:objectProperty

3.1.2 Semantic and structural information

Different types of structural and semantic information can be specified with the different schema definition languages. However, not all of this information is necessary for the matching process. In our approach, we make use of a part of this information that can help to detect mappings between metadata terms. The datatype comparison for instance is ignored because we can find a node n_i which matches another node n_j but one of them corresponds to a leaf node (simple type) and the other is a complex node (complex type). Moreover, datatype is not available on all schema definition languages (e.g., all simple types are *rdfs:literal* for RDF Schema). Element cardinalities are also another example of ignored information. Table 2 shows the semantic and structural information used in our matching process. Examples of utilization of some types of this information are given in Section 3.2.2

3.2 Matching System

In this section, we describe the different steps of the proposed matching system as shown in Figure 2. The system is composed of three main parts: *pre-processing*, *linguistic* and *structural similarity computation*. The system takes as input two schemas presented as directed labeled graphs as well as the structural and semantic information got from the projection step. Then, all irrelevant node names in both schema graphs are eliminated and the useful words are normalized

Table 2: Semantic and structural information

XSD	xsd:restriction, xsd:abstraction, xsd:extension, xsd:substitutionGroup
RDFS	rdfs:seeAlso, rdfs:isDefinedBy, rdfs:subClassOf
OWL	owl:unionOf, owl:complementOf, owl:intersectionOf, owl:TransitiveProperty, owl:someValuesFrom, owl:equivalentClass, owl:disjointWith, rdfs:subClassOf owl:distinctMembers, owl:differentFrom owl:AllDifferent, owl:sameAs, owl:equivalentProperty

(Section 3.2.1). After the *pre-processing* step, the system calculates the linguistic similarity between nodes in both schemas by exploiting the semantics of their corresponding names and comments as well as semantic information obtained from the projection step (Section 3.2.2). Finally, the structural similarity is computed and the correct mappings are selected according to the linguistic and structural similarity scores (Section 3.2.3).

3.2.1 Pre-Processing

In this step, we start by parsing all entities involved in the matching process, including normal nodes, property nodes as well as comments (i.e: textual description available on *xsd:documentation*, *rdfs:comment*, etc.) corresponding to these entities. Then, node names and comments are normalized in order to make their semantics useful for the linguistic similarity calculation step. To do so, several filtering techniques such as tokenization, lemmatization are used [1].

3.2.2 Linguistic similarity computation

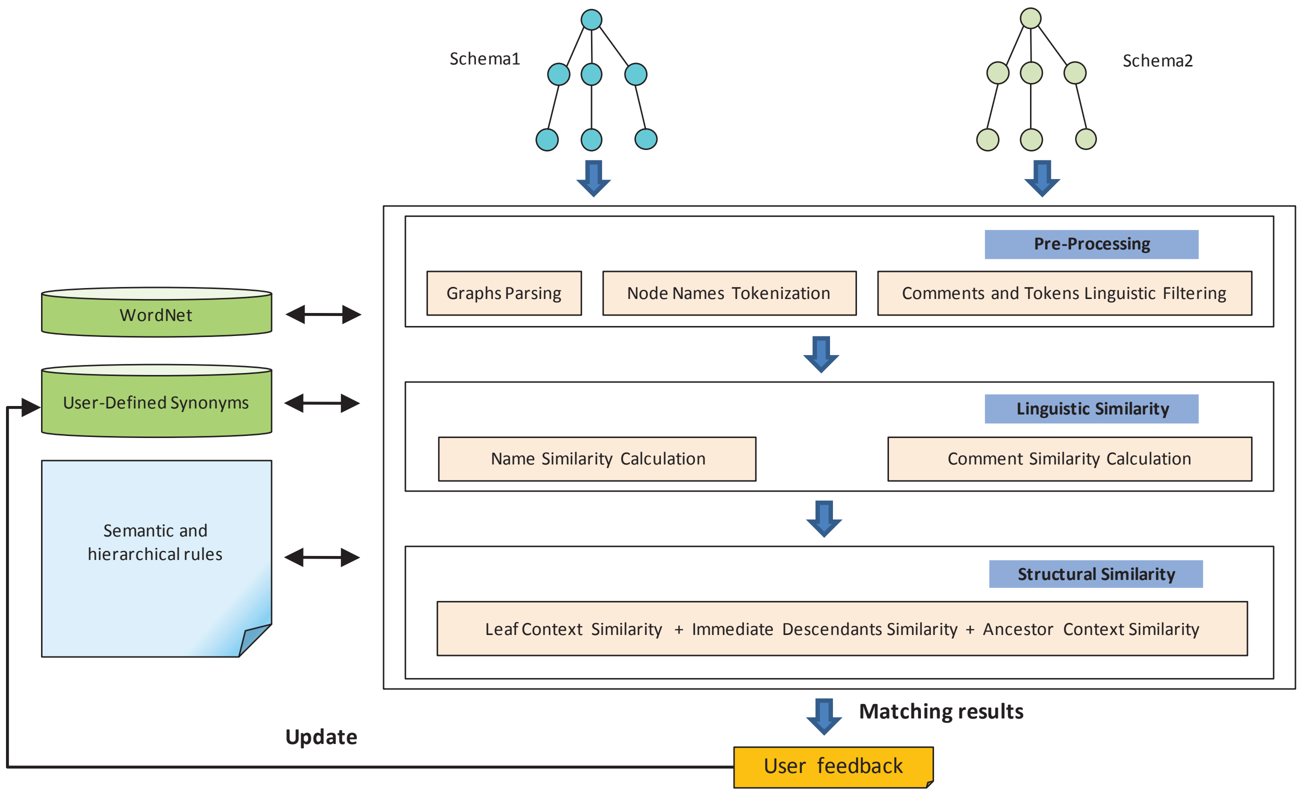
This phase is concerned with the linguistic similarity computation between every graph node pairs belonging to schemas to be mapped. In order to form the linguistic similarity matrix, a string-based technique is used to map the node names. WordNet is used for the explicitation of the words meaning [6]. In addition to the node names, comments are used as a second semantic resource for the matching process. We apply the TF/IDF technique [1] to these comments in order to extract the most pertinent information. The linguistic similarity score is a weighted sum of both similarities. Finally, the semantic and structural information obtained from the projection step are used in order to deduce other mappings.

a) Name matching

The purpose of this phase is to find an initial matching by calculating the similarity distance between the names of all node pairs in the two schemas to be mapped. Each node is represented by a set of tokens. We first start with the explicitation of tokens meaning by using WordNet [1]. Several synonyms can be found for a given term, this helps to resolve problems of terminological conflicts. Each node n_i represented by a set of tokens M_i will have a set of synonyms *synset* for each token m_i after the explicitation step. $M_i^{'}$ is the final result that regroups all synsets returned by M_i explicitation.

$$M_i^{'} = M_i \bigcup \{m_k | \exists m_j \in M_i \bigcap m_k \in synset(m_j)\} \tag{1}$$

Once the explicitation step is performed, we compute the similarity S_{name} between all node pairs belonging to the two



Schema1
Schema2
WordNet
User-Defined Synonyms
Semantic and hierarchical rules
Pre-Processing
Graphs Parsing
Node Names Tokenization
Comments and Tokens Linguistic Filtering
Linguistic Similarity
Name Similarity Calculation
Comment Similarity Calculation
Structural Similarity
Leaf Context Similarity + Immediate Descendants Similarity + Ancestor Context Similarity
Matching results
Update
User feedback

Figure 2: Matching process phases.

schemas to be mapped. To do so, for each node pair (n_1, n_2) we calculate S_{name} by using Jaro-Winkler metric (JW) [1] between each token $m_i \in M_1^{'}$ and all tokens $m_j \in M_2$ (and vice versa). We take the maximum score (MJW) of each token m_i:

$$MJW(m_i, M^{'}) = max_{m_j \in M^{'}} JW(m_i, m_j) \quad (2)$$

Finally, the average of the best similarities is calculated to get the name similarity between nodes:

$$S_{name}(n_1, n_2) = \frac{\Sigma_{m_i \in M_1} MJW(m_i, M_2^{'}) + \Sigma_{m_j \in M_2} MJW(m_j, M_1^{'})}{|M_1| + |M_2|} \quad (3)$$

b) Comment matching

We apply the TF/IDF technique in order to calculate the similarity between comments. To do so, all comments on the two schemas to be mapped are considered as documents, each node will be represented by a vector $v = (w_1, w_2, ..., w_P)$ whose coordinates are the results of TF/IDF. More details about the technique can be found in [1]. Hence, the similarity $S_{comment}$ between two nodes (n_i, n_j) is the distance between vectors corresponding to their comments.

$$S_{comment}(n_i, n_j) = \frac{\sum_{k=1}^{P} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{P}(w_{ik})^2 * \sum_{k=1}^{P}(w_{jk})^2}} \quad (4)$$

The result of above processes is a linguistic similarity matrix *lSim*, where:

$$lSim(n_i, n_j) = \mu_1 * S_{name}(n_i, n_j) + \mu_2 * S_{comment}(n_i, n_j) \quad (5)$$

c) Semantic and structural information utilization

Structural and semantic information obtained from the projection step (Section 3.1) are used in our system to detect other mapping candidates, especially complex ones (n:m mappings). In the following, we present some examples of such features and show how they can be used to deduce other mapping candidates:

- **Generalization Features:** Generalization relationship between two types indicates that one type is a subset of another (e.g., *xsd*:*extension*, *rdfs*:*subClassOf*, *xsd*:abstraction, etc.)[10]. This information can successfully help the matching algorithm to infer complex matches. Let us consider that the attribute *ms*:*Agent* is defined at the schema to be matched with MPEG-7 schema, this attribute linguistically map to *mpeg7*:*AgentType* complex element (*lSim* between both node is greater than a given threshold). In this context, the union of two complex elements *mpeg7*:*PersonType* and *mpeg7*:*OrganizationType* is also considered as a mapping candidates for *ms*:*Agent*. This is done because *mpeg7*:*OrganizationType* and *mpeg7*:*PersonType* are extension of *mpeg7*:*AgentType*. Other structural properties are used (e.g., *owl*:*disjointWith*, *owl*:*unionOf*, etc.)

- **Semantic Features:** Let us consider that n_i and n_j are two nodes linguistically match, and n_k is another node corresponding to a class having an equivalence property with that corresponding to n_i (e.g.,*owl*:

equivalentProperty, *owl*:*equivalentClass*, *xsd*: *substitutionGroup*, etc.)[10]. This information allows us to deduce that n_k is another mapping candidate for n_j. Other semantic properties are also used (e.g., *owl*: *sameAs*, *owl*:*differentFrom*, etc.)

The use of semantic and structural information helps to discover more mappings but may also increase the number of false positive mappings that can be eliminated in the structural similarity computation step (Section 3.2.3).

3.2.3 Structural similarity somputation

Linguistic similarity computation may provide several node candidates. There can be multiple matching candidates which differ in the structure but have a high linguistic similarity value. For instance, *mpeg7:MediaRelTimePoint* and *mpeg7:MediaRelIncrTimePoint*[10] are two elements which may map to the same entity *ms:MediaTimePoint* defined in mediated schema. Thus, in order to deal with this case, the structural similarity is computed in order to prune these false positive candidates. Our structural matching algorithm is based on the node context, which is reflected by its ancestors and its descendants. In this paper, as in [11], we consider three kinds of node contexts depending on their position in the ontology tree: *ancestor context*, *immediate descendant context* and *leaf context*. The context of a node is a combination of these three contexts.

a) Ancestor context similarity measure

The ancestor context of a node n_i is defined as the path p_i extending from the root node of the schema to n_i. Consequently, in order to compare two ancestor contexts, we essentially need to compare their corresponding paths. The authors in [11] have introduced the concepts of path context coefficient to capture the degree of similarity in the paths of two elements. However, this solution has no high matching accuracy. For this reason, we use the ideas of path similarity measure described in [3]. The authors in [3] have relaxed the matching condition by allowing the matching of paths even if their source nodes do not match and their nodes appear in a different order. In addition, paths can also be matched even if there are additional nodes within the path, meaning that the child-parent edge constraint is relaxed into ancestor-child constraint. Such relaxations are inspired by ideas in query answering to approximate answering of queries. The authors in [3] consider two paths p_1 and p_2 being matched, p_2 is the best matching candidate for p_1 if it fulfils the following criteria:

- The path p_1 includes most of the nodes of p_2 in the right order.
- The occurrences of the p_1 nodes are closer to the beginning of p_2 than to the tail, meaning that the optimal matching corresponds to the leftmost alignment.
- The occurrences of the p_1 nodes in p_2 are close to each other, which means that the minimum of intermediate non matched nodes in p_2 are desired.
- If several matching candidates that match exactly the same nodes in p_1 exist, p_2 is the shortest one.

In order to answer to the criteria mentioned above, four parameters have been defined in [3]: $lcs_n(p_i, p_j)$, the longest common subsequences between two paths normalized by the length of the first path, $pos(p_i, p_j)$ which considers that the optimal matching between (p_i, p_j) is the matching that starts on the first element of p_i and continues without gaps, $gap(p_i, p_j)$ used to ensure that the occurrences of two path nodes are close to each other, and $ld(p_i, p_j)$ used to give higher values to source paths whose length is similar to target paths.

Based on the criteria for the paths matching mentioned above, we introduce a new adaptation and relaxation of the approach as follows:

- For the fourth parameters, the *longest common subsequence*(lcs) must be calculated according to the linguistic similarity matrix computed in the previous step. It means that two nodes (n_i, n_j) are considered identical if their $lSim(n_i, n_j)$ is greater than a given threshold e.g. 0.80. That is, ordinary string comparison is now relaxed into string and text comparison ($lSim$) that is based on the similarity threshold.
- The parameters in [3] have been defined for XML which is based on a taxonomy classification (relations between nodes are not labeled, generally containment type). Therefore, the transformation of all schemas into a directed labeled graphs needs more relaxation for the lcs calculation. For instance, let us consider the two RDF triples (*Objet*, *ContentOn*, *URL*) and (*Object*, *hasReference*, *URL*). We can easily note that the predicate value of the first triple matches the subject value (*Content* is equivalent to *Object*). However, the predicate value in the second triple matches the object (*Reference* is equivalent to *URL*). We introduce another type of relaxation by considering triple predicates (property nodes) as nodes which are switchable with the immediate child node or the immediate parent node (object or subject according to RDF interpretation). To do so, we consider each path as a set of segments $p = \{g^1, ..., g^N\}$, each segment is composed of two adjacent nodes (n_k, n_{k+1})(one of them in a property, see Figure 3). In this context, $lcs^{'}$ is the new value of *longest common subsequence* between two paths after the relaxation step. Its calculation is showed in Algorithm 1.

  After the relaxations introduced above, the new version of equations presented in [3] becomes:

$$lcs_n(p_i, p_j) = |lcs'(p_i, p_j)|/|p_i| \tag{6}$$

$$pos(p_i, p_j) = 1 - ((ap - aop)/(|p_j| - 2 * aop + 1)) \tag{7}$$

$$gap(p_i, p_j) = gaps/(gaps + lcs'(p_i, p_j)) \tag{8}$$

$$ld(p_i, p_j) = (|p_j| - lcs'(p_i, p_j))/|p_j| \tag{9}$$

If we consider the two paths showed in Figure 3 as the path being matched, the *longest common subsequence* is 5 instead of 0 after the two steps of relaxation introduced above.

Finally, the similarity ps between two paths (p_i, p_j) is obtained by combining the scores resulting from the equations (6, 7, 8 and 9):

$$ps(p_i, p_j) = \delta lcn_n(p_i, p_j) + \varphi pos(p_i, p_j) - \theta gap(p_i, p_j) - \lambda ld(p_i, p_j) \tag{10}$$

Algorithm 1 lcs' calculation

```
1: Input parameters $(p_1, p_2)$.
2: Each path is a set of segments $p_1 = \{g_1^1, ...., g_1^N\}$, $p_2 = \{g_2^1, ...., g_2^M\}$.
3: Each segment is a composed of two adjacent nodes $g_i = (n_i, n_{i+1})$.
4: int $lcs' = 0$. int i = 0.
5: calculate $lcs(p_1, p_2)$.
6: $lcs' = lcs(p_1, p_2)$.
7: for all segments $g_i^1$ $(i \in N)$ do
8:   switch nodes $(n_i, n_{i+1})$ of segment $g_i^1$ and update $p_1$.
9:   calculate $lcs(p_1, p_2)$
10:  if $(lcs(p_1, p_2) > lcs')$ then
11:    $lcs' = lcs(p_1, p_2)$
12:    pass to the after next segment (i := i + 2).
13:  else
14:    return $(n_i, n_{i+1})$ to their previous positions and update $p_1$.
15:    pass to the next segment (i := i + 1).
16:  end if
17: end for
18: return $lcs'$
```

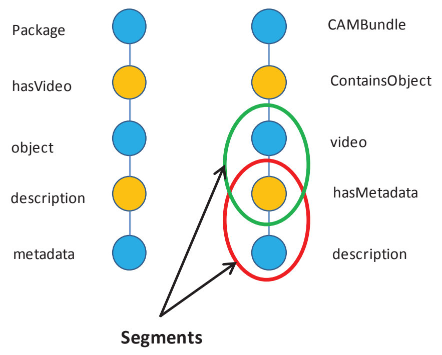


Figure 3: Example of paths and segments.

where δ, φ, θ and λ are positive parameters representing the comparative importance of each factor. These parameters are given based on experimental results from [3]: $\delta = 0.75$; $\varphi = 0.25$; θ=0.25; λ=0.2.

Finally, the ancestor context similarity *ancSim* between two nodes (n_i, n_j) is the path similarity of their ancestors (p_i, p_j) weighted by their linguistic similarity $lSim(n_i, n_j)$:

$$ancSim(n_i, n_j) = ps(n_i, n_j) * lSim(n_i, n_j) \qquad (11)$$

b) Immediate descendant context similarity measure

To obtain the immediate descendant context similarity *immSim* between two nodes (n_i, n_j), we compare their two immediate descendant sets $S = \{s_1, s_2, \cdots, s_n\}$ and $S' = \{s_1', s_2', \cdots, s_m'\}$ (immediate descendant nodes are normal nodes, property nodes are included only for path similarity calculation). This is done by using the similarity *SimI* between each pair of children in the two sets. Where:

$$SimI(s_i', s_j) = ps(s_i', s_j) * lSim(s_i', s_j) \qquad (12)$$

$ps(s_i', s_j)$ is the similarity between paths extending from (s_i', s_j) to (n_i, n_j).

Then, we select the matching pairs with maximum similarity values.

$$MaxSimI(s_i, S') = max_{s_j' \in S'} SimI(s_i, s_j') \qquad (13)$$

$$MaxSimI(s_i', S) = max_{s_j \in S} SimI(s_i', s_j) \qquad (14)$$

Finally, the average of the best similarity values is taken to get the immediate descendant similarity value *immSim*:

$$immSim(n_i, n_j) = \frac{\sum_{i=1}^{|S|} MaxSimI(s_i, S') + \sum_{i=1}^{|S'|} MaxSimI(s_i', S)}{|S\prime| + |S|} \qquad (15)$$

c) Leaf context similarity measure

The leaf context of a node n_i is defined as the set of leaf nodes of subtrees rooted at n_i. If $l_i \in$ *leaves(n_i)* is a leaf node, then the context of l_i is given by the path p_i from n_i to l_i. The leaf context is given by:

$$leafSim(l_i, l_j) = ps((p_i, p_j)) * lSim(l_i, l_j) \qquad (16)$$

To obtain the leaf context similarity between two leaves $l_i \in$ *leaves(n_i)* and $l_j \in$ *leaves(n_j)*, we compute the leaf similarity *leafSim* between each pair of leaves in the two leaf sets. We then select the matching pairs with the maximum similarity values. The average of the best similarity values is taken (see Section 3.2.3b)

d) Node similarity

The node similarity *nodeSim* between normal nodes (property nodes are included only for path similarity calculation) can be obtained by the combination of *ancestor context*, *immediate descendant context*, and *leaf context* similarities unless one of the two nodes being compared is a leaf node or a node child of root. In this case, the node similarity calculation considers that the context of both nodes depends only on their ancestors or descendants. The node similarity is given by:

$$nodeSim(n_i, n_j) = \alpha * ancSim(n_i, n_j) + \beta * immSim(n_i, n_j) + \gamma * leafSim(n_i, n_j) \qquad (17)$$

where $\alpha + \beta + \gamma = 1$ and $(\alpha, \beta, \gamma) \geq 0$

Once the structural similarity computation is made, the system returns for each source node n_i the K target node candidates that have the maximum values of *nodeSim*. These nodes must have *nodeSim* value greater than a given threshold. Finally, the user selects the correct mappings and the user defined synonyms is updated according to the feedbacks.

4. EXPERIMENTAL EVALUATION

In this section, we describe the experiments that we have carried out to evaluate our proposed system. Firstly, we describe the data sets which we have used through the evaluation. Secondly, we show the experimental results which allow us to evaluate the proposed system.

4.1 Data Sets

The system has been tested using several metadata standards (MPEG-7, MPEG-21, EXIF, MIX, DIG35, PeCMan Ontology, TV-Anytime, DIG35 Ontology and CAM4Home) [10][2]. These standards have a significant heterogeneity on two levels: metadata specification is performed using different languages and schemas that present a high structural and semantical heterogeneity.

In order to compare our results with Cupid [13], we must first choose only metadata based XML Schema. To do so, we calculated the mapping between TV-Anytime and all standards based XSD. Secondly, so we can compare our system with SF [14] we calculated the mapping between CAM4Home metadata model presented in [2] and all metadata standards mentioned aboves. CAM4Home metadata model is based RDF Schema, it is a part of the CAM4Home ITEA2 project [1]. A group of twenty multimedia academic and industrial practitioners from TV, 3G and Internet application fields defined a large set of metadata requirements in order to support the convergence of multimedia contents in Digital Home environments. Metadata defined under CAM4Home project describes information related to content semantics, user characteristics and device profiles.

4.2 Experimental Results

The experimentation starts by setting the α, β and γ parameters to know the effect of each similarity context on the matching process. Then, the matching quality measure is showed in Section 4.2.2. Finally, a comparative study is given in Section 4.2.3.

4.2.1 Tuning parameters

In order to know the influence of each similarity context, we calculated the mapping between several metadata standards using several combinations of the α, β and γ parameters. The mapping results between (MPEG-7, DIG35) and (DIG35, EXIF) in terms of *F-Measure* [13] is showed in Figure 4 and 5 respectively.

The experimental evaluation shows that the greatest amount of structural information is contained in the ancestor context (Figure 4 and 5) where the highest values of F-Measure is located when $\alpha \in [0.45\ 0.6]$; this explains the interest of some matching strategies which consider that the context of nodes only depends on their ancestors [5]. Besides, the experimentations showed that the immediate descendant context and the leaf context have also an effect on the matching process, where the highest values of F-Measure are located when $\beta \in [0.1\ 0.15]$ and $\gamma \in [0.25\ 0.4]$. However, we believe that an automatic choice of the α, β and γ values for each pair of nodes according to the node positions in the graph can increase the matching accuracy. This will be part of our future works.

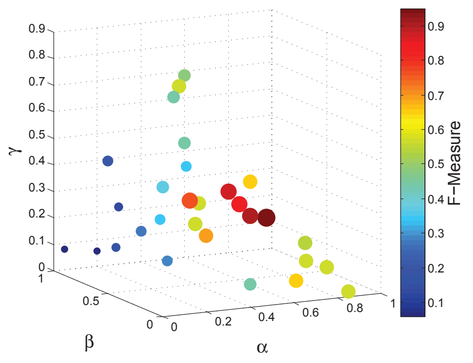


Figure 4: MPEG-7, DIG35 mapping results.

[1]http://www.cam4home-itea.org/

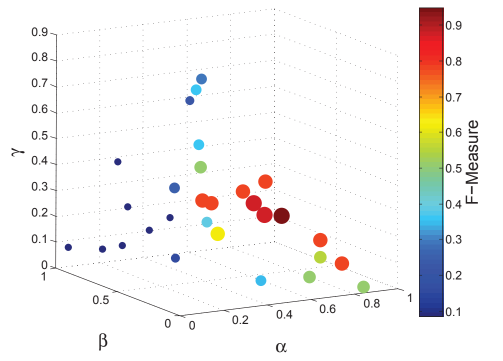


Figure 5: DIG35, EXIF mapping results

4.2.2 Matching quality measure

In order to calculate the similarities between nodes, equation (17) parameters are given according to the experiment results from the previous section ($\alpha = 0.55, \beta = 0.15, \gamma = 0.30$). The proposed solution has been evaluated by considering the matching quality based on the criteria described in [13][14], including *Precision*, *Recall*, *F-measure*. The experiment results of the mapping between TV-Anytime and other standards based XSD in terms of *Precision*, *Recall* and *F-measure* are shown in Table 3. Table 4 shows the mapping results between CAM4Home and other standards.

Table 3: Mapping results between TV-Anytime and standards based XSD

Metadata standards	Precision	Recall	F-measure
MIX	50%	55%	52%
EXIF	62%	65%	63%
MPEG-7	75%	70%	72%
MPEG-21	43%	47%	45%
DIG35	77%	82%	79%

Table 4: Mapping results between CAM4Home and other standards

Metadata standards	Precision	Recall	F-measure
MIX	92%	89%	90%
DIG35	87%	90%	88%
EXIF	90%	81%	85%
MPEG-7	77%	64%	70%
MPEG-21	72%	60%	65%
PeCMan Ontology	94%	89%	91%
DIG35 Ontology	85%	90%	87%
TV-Anytime	77%	72%	75%

4.2.3 Comparison with other systems

In order to compare the proposed solution with other systems, we have implemented SF [14] and Cupid [13]. The choice of Cupid and SF is done because they are schema based, and they all utilize linguistic and structural matching techniques. The result of this comparative study in terms of F-measure score is showed in Figure 6 and 7 where our solution is better than SF and Cupid for all tested metadata standards. This is due to the use of a single context for both systems (leaf context for Cupid and child context for SF). Whereas, in our solution, we have considered all of ancestor-context, child context and leaf-context. Besides our system is general and not limited to one or two schema definition languages, it has has successfully detected 68 % of complex matchings that are not detected by other systems.

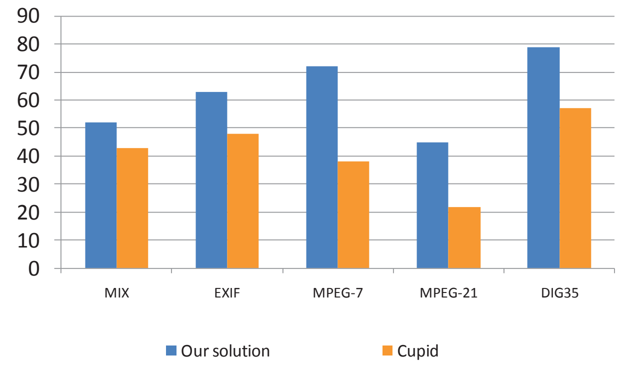


Figure 6: Comparative study with Cupid (F-measure).

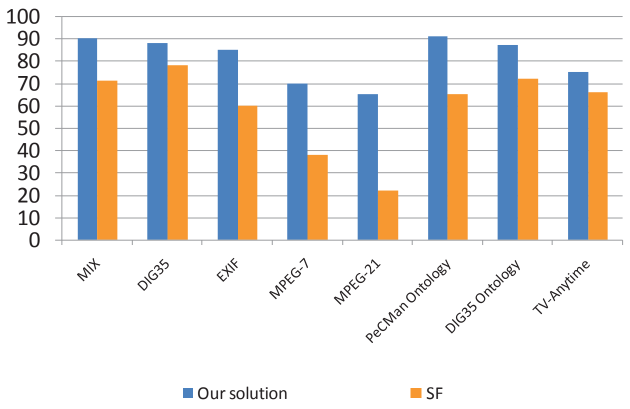


Figure 7: Comparative study with SF (F-measure).

5. CONCLUSION

Due to the extensive use of metadata and their semantic and structural heterogeneity, there has been a great interest to develop an integration system for achieving metadata interoperability. The existence of such model is crucial for media object uniform access fulfillment. To do so, we proposed and implemented in this paper a new integration technique for achieving metadata interoperability whatever the definition language. We essentially proposed a matching system that supports metadata schemas whatever their description language. It uses several types of syntactic, semantic and structural information to match between metadata. Our experiments showed that the combination of the linguistic and structural similarities plays a significant role in deriving a correct mapping.

In our ongoing works, we plan to enhance the proposed matching system through a better use of structural information. This can be achieved by adding a new structural matching technique to the system. We mainly explore the use of adjacency nodes to detect other mappings that cannot be detected by the current matching strategy. We also plan to enhance the proposed approach by taking into account the mappings already validated by the user as another structural information which may help to find other mappings.

6. ACKNOWLEDGEMENTS

This work was supported by the ITEA2 CAM4HOME European project.

7. REFERENCES

[1] S. Amir, I. M. Bilasco, T. Danisman, I. E. sayad, and C. Djeraba. Multimedia metadata mapping: Towards helping developers in their integration task. In *MoMM*, pages 13–19, 2010.

[2] I. M. Bilasco, S. Amir, P. Blandin, C. Djeraba, J. Laitakari, J. Martinet, E. Martínez-Gracía, D. Pakkala, M. Rautiainen, M. Ylianttila, and J. Zhou. Semantics for intelligent delivery of multimedia content. In *SAC*, pages 1366–1372, 2010.

[3] D.Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching xml documents via xml fragments. In *SIGIR*, pages 151–158, 2003.

[4] H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621, 2002.

[5] J. Euzenat and P. Shvaiko. *Ontology matching.* Springer-Verlag, Heidelberg (DE), 2007.

[6] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998.

[7] R. García and O. Celma. Semantic integration and retrieval of multimedia metadata. In *2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media*, pages 69–80, 2005.

[8] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *ESWS*, pages 61–75, 2004.

[9] B. Haslhofer and W. Klas. A survey of techniques for achieving metadata interoperability. *ACM Comput. Surv.*, 42(2), 2010.

[10] M. Hausenblas. Multimedia vocabularies on the semantic web, jul 2005.

[11] M.-L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: clustering xml schemas for effective integration. In *CIKM*, pages 292–299, 2002.

[12] W. Lee, T. Bürger, F. Sasaki, and V. Malaisé. Use cases and requirements for ontology and api for media object, July 2008.

[13] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, pages 49–58, 2001.

[14] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

[15] I. Miletic, M. Vujasinovic, N. Ivezic, and Z. Marjanovic. Enabling semantic mediation for business applications: Xml-rdf, rdf-xml and xsd-rdfs transformations. In *IESA*, pages 483–494, 2007.

[16] K. Yang, R. Steele, and A. Lo. An ontology for xml schema to ontology mapping representation. In *iiWAS*, pages 101–111, 2007.