Knowledge Extraction from an XML Data Flow: Building a Taxonomy based on Clustering Technique

Ernesto Damiani **Maria Cristina Nocerino** **Marco Viviani**
Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano
Via Bramante, 65 – 26013 Crema (CR) – Italy
{edamiani, mnocerino, mviviani}@crema.unimi.it

Abstract Similarity between XML documents can be studied in different ways, in particular depending of the fact that we take into consideration only the structure of the document, its content, or both of them. In this paper we choose a structure-based approach, based on the fuzzyfication of the XML documents representing instances of a given XML data model, belonging to a common domain of interest. After a process of data flow analysis we obtain similarity values between documents, used to create similarity classes representing the domain. Using a clustering technique and a graph based approach it is possible build a taxonomy that can be represented through an ontology language.

Keywords: Knowledge Management, Fuzzy bags, Taxonomy Building, Ontology Extraction.

1 Introduction

The process of building a hierarchy from a data flow, the final task of knowledge extraction process, is strictly related to the vision of the Semantic Web. It is necessary that data items are described in such way that the meaning of them can be exploited, e.g. because general concepts associated to data can be related to other concepts. Specifications of relevant concepts and of relations among them are called ontologies [1].

2 The encoding problem for XML data

XML (*eXtensible Markup Language*) is a mark-up language for information interchange, often used to enable semantics-aware tagging of World Wide Web information. XML information shares many features of semi-structured data: its structure can be irregular, is not always known ahead of time, and may change frequently and without notice. Several models have been proposed for semi-structured data including the well-known OEM (*Object Exchange Model*)

[2], a graph-based, self-describing object instance model. In the same way XML has its own data model, related to OEM, called Information Set (*Infoset*) [3]. A more abstract representation of this model [4], useful in our work, is a *multi-labeled directed graph* $G = (V, E, L, f, g)$, whose node set V comprises both nodes representing tags and nodes representing text/multimedia content and attributes. Arcs belonging to $E \subseteq V \times V$ may represent, according to their labeling (given as usual by a function $f : E \to L$, where $L = \{$*e-contains*, *a-contains*, *link*, *id-idref*$\}$ is a set of relation labels) tag and attribute inclusion, hypertext links, and ID-IDREF relationships. Another function $g : V \to I^*$ (where I^* is the set of strings built over a suitable alphabet I) represents the value or content associated to a terminal element or attribute. The sub-graph representing (element and attribute) containment alone is in most cases a tree, where leaf nodes represent content and values, while non-leaf nodes correspond to tags. For the sake of simplicity in this paper we only refer to such sub-graph. Under this hypothesis, it is possible consider every document belonging to the data flow as an instance of the particular model described before. A first step of this work is comparing documents encoded as fuzzy bags (fuzzy multi-set). Our aim is to obtain a good method of comparison fuzzyfying instances leaving unchanged the model.

2.1 Fuzzy bags and fuzzyfication of documents

In our problem we have to deal with a collection of documents in which duplicates are significant. A collection of elements which may contain duplicates is called a *bag*. So we model XML documents as bags due to the fact that the same tag can be repeated at different levels and/or with different cardinalities at the same level. In particular we use a fuzzy technique to model a document as a bag, obtaining a *fuzzy bag*. In our approach we only take into consideration tag names and not their value (following an approach *value-blind*) because information we want to extract is related to typical structures, later to be mapped into classes. The encoding of an XML tree as a fuzzy bag can be performed as follows: supposing that every tag x_i has associated a membership degree $\mu(x_i)$, initially equal to 1 ($\mu(x_i) = 1$), we divide every membership degree by its nesting level L, obtaining a lossy representation of the tree, taking into consideration the original nesting level of tags, but not the brotherhood relations (Fig. 1). This process is called *encoding process*: many encoding techniques can be used, the simple one we just described is called *flattening*.

3 Matching fuzzy bags representing XML fragments

In order to compare the fuzzy bags representing the XML documents belonging to the data flow, we have taken into account measures of comparison studied in [5]. Having a data flow and no reference documents to start with, we need a symmetric measure to compare all documents to each other. So we used a *measure of resemblance*:

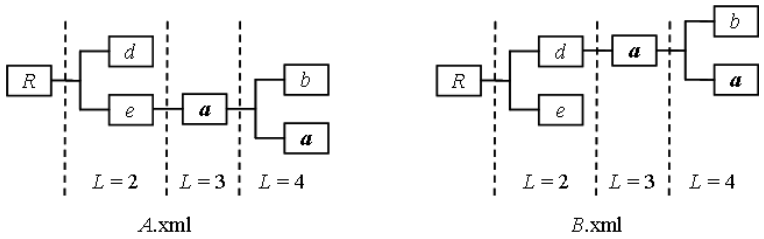

Figure 1: Example of the encoding process. Applying the method described before to the tree representation of generic XML documents A.xml and B.xml, we obtain the same fuzzy bag $A = B = \{R/1, a/0.33, a/0.25, b/0.25, d/0.5, e/0.5\}$

$$S(A,B) = M(A \cap B)/M(A \cup B) = \\ = M(A \cap B)/(M(A \cap B) + M(A -_1 B) + M(B -_1 A))$$

Where:

$$M = \sum_{count}(A) = \sum_{x \in X} \mu_A(x),$$

$$\mu_{A-_1 B}(x) = max(0, \mu_A(x) - \mu_B(x)),$$

$$f_{A \cap B}(x) = \mu_{A \cap B}(x) = min(\mu_A(x), \mu_B(x)).$$

Applying the extension principle, as discussed in [6], it is possible to extend to fuzzy bags the measure studied in [5] for fuzzy sets. One of the main reason to adopt a model based on bags rather than sets is making our technique sensitive to differences in cardinality between tags of the same name. Ideally, similarity obtained with this sort of encoding should be inversely proportional to differences in cardinalities.

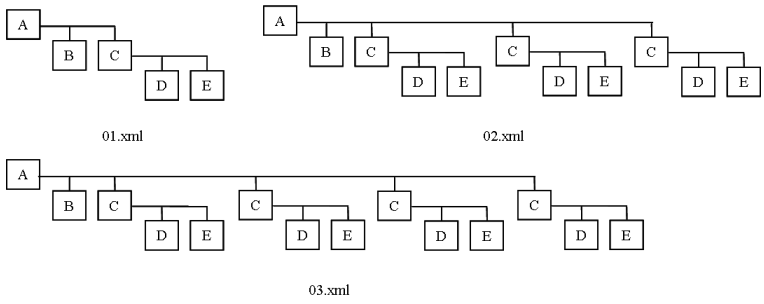

Figure 2: Tree representation of three generic XML documents, different for tag cardinality

The following fuzzy bags (01, 02 and 03) represent the encoded XML documents shown in Fig. 2.

$$01 = \{\{1/0, 1/1\} * a, \{1/0, 0.5/1\} * b, \{1/0, 0.5/1\} * c, \\ \{1/0, 0.33/1\} * d, \{1/0, 0.33/1\} * e\}$$

$$02 = \{\{1/0, 1/1\} * a, \{1/0, 0.5/1\} * b, \{1/0, 0.5/3\} * c, \\ \{1/0, 0.33/3\} * d, \{1/0, 0.33/3\} * e\}$$

$$03 = \{\{1/0, 1/1\} * a, \{1/0, 0.5/1\} * b, \{1/0, 0.5/4\} * c, \\ \{1/0, 0.33/4\} * d, \{1/0, 0.33/4\} * e\}$$

As we can see below documents having a similar number of occurrences of the same tags (02 and 03) present an higher similarity value.

$$|01 \cap 02| = \{1/1, 0.5/3, 0.33/5\} \cong 2.66$$
$$|01 \cup 02| = \{1/1, 0.5/5, 0.33/11\} \cong 4.98$$
$$|01 \cap 02|/|01 \cup 02| \cong 2.66/4.98 \cong 0.53$$

$$|01 \cap 03| = \{1/1, 0.5/3, 0.33/5\} \cong 2.66$$
$$|01 \cup 03| = \{1/1, 0.5/6, 0.33/14\} \cong 6.14$$
$$|01 \cap 03|/|01 \cup 03| \cong 2.66/6.14 \cong 0.43$$

$$|02 \cap 03| = \{1/1, 0.5/5, 0.33/11\} \cong 4.98$$
$$|02 \cup 03| = \{1/1, 0.5/6, 0.33/14\} \cong 6.14$$
$$|02 \cap 03|/|02 \cup 03| \cong 4.98/6.14 \cong 0.81$$

Besides cardinality, our approach is also sensitive to differences in structure between XML documents containing the same group of tags in different position. The more the topological structure of documents changes, the more similarity between them decreases. Fuzzy bags 04, 05 and 06 represent XML documents shown in Fig. 3.

$$04 = \{\{1/0, 1/1\} * a, \{1/0, 0.5/1\} * b, \{1/0, 0.5/1\} * c, \{1/0, 0.33/1\} * d, \\ \{1/0, 0.33/1\} * e, \{1/0, 0.33/1\} * f, \{1/0, 0.33/1\} * g, \{1/0, 0.25/1\} * h, \\ \{1/0, 0.25/1\} * i, \{1/0, 0.25/1\} * l, \{1/0, 0.25/1\} * m, \{1/0, 0.25/1\} * n, \\ \{1/0, 0.25/1\} * o, \{1/0, 0.25/1\} * p, \{1/0, 0.25/1\} * q\}$$

$$05 = \{\{1/0, 1/1\} * a, \{1/0, 0.33/1\} * b, \{1/0, 0.5/1\} * c, \{1/0, 0.5/1\} * d, \\ \{1/0, 0.33/1\} * e, \{1/0, 0.33/1\} * f, \{1/0, 0.33/1\} * g, \{1/0, 0.25/1\} * h, \\ \{1/0, 0.25/1\} * i, \{1/0, 0.25/1\} * l, \{1/0, 0.25/1\} * m, \{1/0, 0.25/1\} * n, \\ \{1/0, 0.25/1\} * o, \{1/0, 0.25/1\} * p, \{1/0, 0.25/1\} * q\}$$

$$06 = \{\{1/0, 0.25/1\} * a, \{1/0, 0.33/1\} * b, \{1/0, 0.5/1\} * c, \{1/0, 0.33/1\} * d, \\ \{1/0, 0.33/1\} * e, \{1/0, 0.33/1\} * f, \{1/0, 0.5/1\} * g, \{1/0, 0.25/1\} * h, \\ \{1/0, 0.25/1\} * i, \{1/0, 0.25/1\} * l, \{1/0, 0.25/1\} * m, \{1/0, 0.25/1\} * n, \\ \{1/0, 0.25/1\} * o, \{1/0, 0.25/1\} * p, \{1/0, 1/1\} * q\}$$

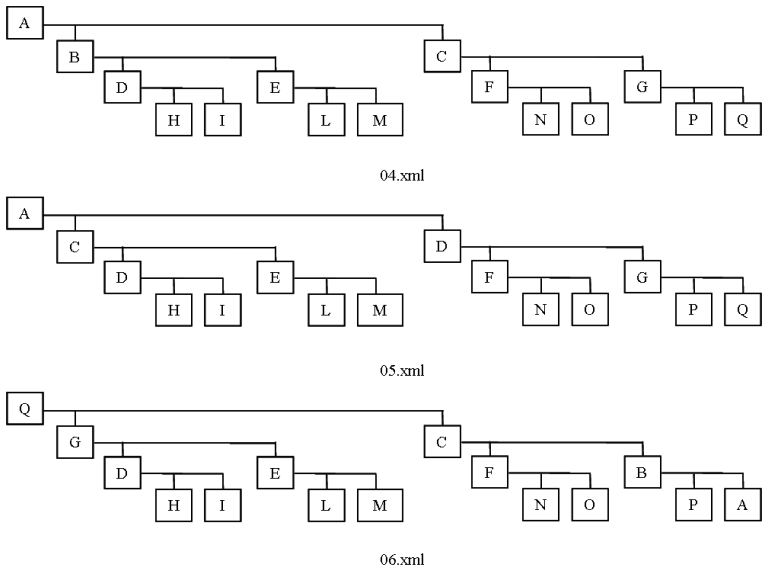

Figure 3: Tree representation of three generic XML documents different in structure

Also in this case the proposed approach gives the expected result: 04 and 05 having a "not so different" structure present an higher similarity value.

$|04 \cap 05| = \{1/1, 0.5/2, 0.33/7, 0.25/15\} \cong 5.15$
$|04 \cup 05| = \{1/1, 0.5/4, 0.33/7, 0.25/15\} \cong 5.49$
$|04 \cap 05|/|04 \cup 05| \cong 5.15/5.49 \cong 0.92$

$|04 \cap 06| = \{1/0, 0.5/1, 0.33/6, 0.25/15\} \cong 4.40$
$|04 \cup 06| = \{1/2, 0.5/5, 0.33/8, 0.25/15\} \cong 6.24$
$|04 \cap 06|/|04 \cup 06| \cong 4.40/6.24 \cong 0.70$

$|05 \cap 06| = \{1/0, 0.5/1, 0.33/6, 0.25/15\} \cong 4.40$
$|05 \cup 06| = \{1/2, 0.5/5, 0.33/8, 0.25/15\} \cong 6.24$
$|05 \cap 06|/|05 \cup 06| \cong 4.40/6.24 \cong 0.70$

Comparing a large set of data, it is reasonable that some documents may have common features. So we compare only documents belonging to a subset of the available data set. This subset is provided automatically or by a domain expert identifying documents with different global and local cardinality. In the first case we choose documents which have different number of attributes, in the second we consider tag cardinality, which can be different for the same tag in different documents. Similarity values obtained from the comparison of this subset of documents are then used to populate a matrix. The matrix in Fig. 4

contains data computed from the comparison of a subset of documents belonging to a musical domain. Here, an α-cut has been performed and original fuzzy values have been replaced by crisp value 1 where the similarity value is higher than threshold α. We rely on the fact that multimedia files have associated a certain number of optional information. For example mp3 files have a fixed (now growing) number of predefined tags (author, title, genre, ...). This information can be organized in XML fragments depending on the view we want to have of the musical domain. With our approach is possible for instance cluster fragments describing single albums (i.e. CDs containing a single track) and distinguish them with respect to other fragments describing complete albums (i.e. CDs containing multiple tracks).

Doc.	03	07	12	14	15	17	19	20
03	1	0	0	***1***	0	0	0	0
07		1	0	0	0	***1***	0	0
12			1	0	0	0	***1***	0
14				1	0	0	0	0
15					1	0	0	***1***
17						1	0	0
19							1	0
20								1

The matrix is symmetrical

Figure 4: Similarity values between documents grouped in a matrix

4 Building a Taxonomy

Once similarity values between XML documents have been obtained, it is not difficult create a taxonomy as the outcome result of a pattern discovery procedure. In other words, we compute a collection of blocks composed of documents close to a pattern, again based on similarity. Though our blocks are not the result of proper clustering, since the chosen similarity measure lacks the mathematical property of a distance, we shall call them *clusters* in the following. Intuitively, patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster. The rationale behind this choice is that we prefer to use a measure of comparison (not always symmetrical) rather than a distance, the latter being too restrictive to represent complex objects such as XML documents and relationships between them, e.g. is-a relationships going from the general concepts to more specific ones. The first task is to group in the same cluster documents with a similarity value equal to 1 (we remember that we have replaced fuzzy values with crisp values 0 and 1 in the matrix for the elements lower and greater than a threshold α). Referring to Fig. 4 and considering only half of the matrix (because the measure chosen for the comparison process is symmetrical), we obtain the subdivision in clusters shown in Fig. 5.

Cluster1	*Cluster2*	*Cluster3*	*Cluster4*
07	15	03	12
17	20	14	19

Figure 5: Subdivision in clusters

Then it is necessary to extract a simple description of each cluster. This process of *data abstraction*, produces a compact description of each cluster in terms of cluster prototypes or representative patterns, that we call *cluster-heads*. A good candidate to be cluster-head should be the smallest fuzzy bag in the cluster (i.e. the one with the smallest number of elements whose membership is greater than 0) or, using a (non symmetrical) measures of inclusion [5], the bag more included in each other bag belonging to the same cluster. The cluster-heads hint to classes used to create the taxonomy. A sample is shown in Fig. 6.

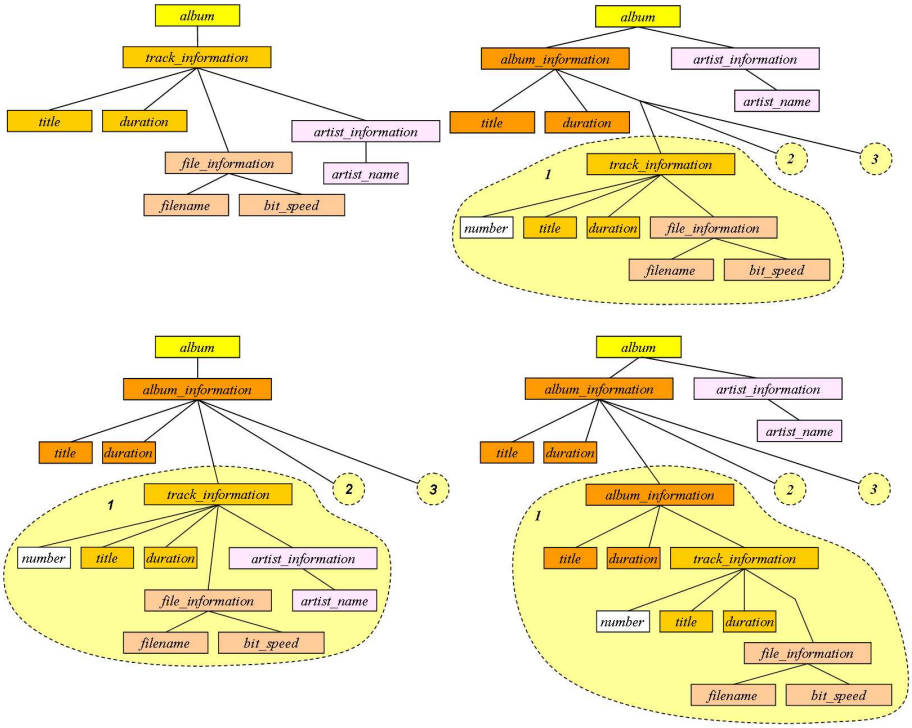

Figure 6: Cluster-heads representing the musical domain studied

The next step is to assign meaning to discovered blocks; this task is manually done by domain experts who relate derived class structure of data to domain theory, and, more specifically to the current problem solving situation.

Hopefully this process can also lead to discovery of new concepts that improve problem solving and extend domain theories.
We evaluate clustering validity in term of *class cohesion* or *intra-class similarity*. This is defined in terms of how individual documents match the prototypical description of the cluster they are assigned to by algorithm. Classes that exhibit high class cohesion lead to a better classification.
Once obtained classes representing our domain the last task of our work is discover relations between them. We represent classes as vertexes of a direct graph. Every edge joins the class with lower cardinality with classes with higher cardinality. Labels associated to the edges express the value of inclusion (calculated with the formula $S(A,B) = M(A \cap B)/M(B)$ [5]) between the connected classes (Fig. 7 (a)).

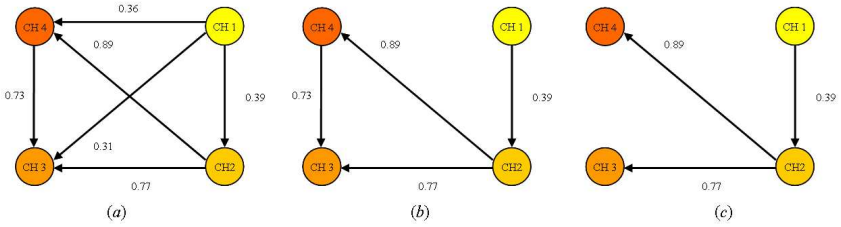

Figure 7: Building of a hierarchy

The initial direct cyclic graph has to be converted in a direct acyclic graph that better expresses inheritance relations. If it is possible, in Fig. 7 (b) and Fig. 7 (c) are shown two results of this process. In (b) for example, having a threshold $\gamma = 0.5$ we obtain a Directed Acyclic Graph (DAG). This kind of situation shows how not always is possible to obtain a tree as we can instead see in (c), where choosing an higher threshold or based on a decision by a domain expert, the edge between CH3 and CH4 is removed.

4.1 Representing taxonomy through OWL

Once the hierarchy has been obtained, the last task is its representation through an ontology language. Since the results our knowledge extraction process will need to be integrated with existing normative ontologies [1]. It is important to be able to translate them into restrictions on existing ontology classes. So we need a language allowing to express restrictions on some attributes properties (level, cardinality). For this reason we chose OWL [9], a language which allows to describe foundations of a description logic, because it makes possible to define logic constraints on a relation. Fig. 8 shows an OWL fragment showing the inferred taxonomy linked to the musical domain described before.

```
<owl:Class rdf:ID="Cluster4">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#int">1
            </owl:minCardinality>
            <owl:onProperty>
               <owl:ObjectProperty rdf:about=
                "file:/../Ontology/
                schema#has_artist_information"/>
            </owl:onProperty>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource=
    "file:/../Ontology/schema#Cluster2"/>
</owl:Class>
```

Figure 8: An OWL fragment representing a part of the inferred taxonomy

Acknowledgements

This work was partially supported by the Italian Ministry of Research Basic Research Fund (FIRB) - Project KIWI. Special thanks goes to D. Rocacher of the IRISA/ENSSAT - Lannion, for his help in understanding problems connected to operations on fuzzy bags.

References

[1] T. R. Gruber (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *International Journal of Human-Computer Studies*, 43(5/6), pages 907-928.

[2] D. Quass, A. Rajarman, Y. Sagiv, J. Ullman, J. Widom (1995). Querying semistructure heterogeneous information. In *International Conference on Deductive and Object Oriented Databases*, pages 319-344.

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

[4] E. Damiani, L. Tanca (2000). Blind Queries to XML Data. In *Proceedings of Database and Expert Systems Applications: 11th International Conference, DEXA 2000*, pages 345-356.

[5] B. Bouchon-Meunier, M. Rifqi, S. Bothorel (1996). Towards general measures of comparison of objects. In *Fuzzy Sets and Systems*, volume 84, pages 143-153.

[6] P. Ceravolo, M. C. Nocerino, M. Viviani (2004). Knowledge extraction from semi-structured data based on fuzzy techniques. In *Knowledge-Based and Emergent Technologies Relied Intelligent Information and Engineering Systems*, Part II, pages 1257-1263.

[7] A. K. Jain, M. N. Murty, P. J. Flynn (1999). Data Clustering: A Review. In *ACM Computing Surveys*, volume 31, no.3, pages 264-323.

[8] D. Rocacher (2003). On fuzzy bags and their application to flexible querying. In *Fuzzy Sets and Systems*, volume 140, no. 1, pages 93-110.

[9] http://www.w3.org/TR/2004/REC-owl-features-20040210/.