Toward Semantic XML Clustering

Andrea Tagarelli, Sergio Greco *

Abstract

The increasing availability of heterogeneous XML informative sources has raised a number of issues concerning how to represent and manage semistructured data. Although XML sources can exhibit proper structures and contents, differently annotated XML documents may in principle encode related semantics due to subjective definitions of markup tags. Discovering knowledge to infer semantic organization of XML documents has become a major challenge in XML data management. In this context, we address the problem of clustering XML data according to structure as well as content features enriched with lexical ontology knowledge. We propose a framework for clustering semantically cohesive XML structures based on a transactional representation model. Experiments on large real datasets give evidence that the proposed approach is highly effective in detecting groups of XML data that exhibit structure and/or content affinities.

1 Introduction

XML is touted as the driving-force for representing and exchanging data on the Web. Indeed, the semistructured and self-describing physiognomy of XML makes it feasible to model a broad variety of data as XML documents, in order to fulfill the promises of the next-generation Web.

XML data sources exhibit different structures and contents. Markup tags, which play a basic role to impose the structure of a document, reflect subjective factors that brand the authorship in coding information. As a consequence, differently annotated XML data may be "semantically related" at a certain degree.

In such a context, a challenge is inferring semantics from XML documents according to the available syntactic information, namely structure and content features. This has several interesting application domains, such as integration of data sources and query processing, that can be seamlessly generalized to any kind of semistructured data. For example, detecting structural and content affinities among XML data can aid to conceive techniques for indexing such data, thus narrowing the search space and improving the design of query plans.

As a fundamental exploratory data mining task, clustering represents the natural solution to discover common characteristics and specific facets exhibited by XML documents. However, the complexity intrinsic to semistructured data requires nontrivial effort to define an effective clustering framework. Extracting significant features, modeling document structures and contents, defining an appropriate notion of homogeneity between documents are only some of the issues to be addressed.

Contribution. In this paper we investigate how to cluster semantically related XML data through an in-depth analysis of content and structural specifics in the data. A major novelty of our proposal for mining XML data is the introduction of the notion of *tree tuple* in the definition of an XML representation model that allows for mapping XML document trees into transactional data. The notion of tree tuple lends itself particularly well to identify semantically cohesive substructures from XML documents; moreover, it enables a flat, relational-like XML representation that is well-suited to meet the requirements for clustering XML data according to structure and content information. Our contributions can be summarized as follows:

1. We devise suitable features for XML data, focusing on content information extracted from textual elements and structure information derived from tag paths. Both kinds of syntactic information are enriched with knowledge provided by a *lexical ontology*. In particular, for the structure case, we propose a novel *word sense disambiguation* method to select the most appropriate sense for each tag name in the context of an XML tree path. XML features are conveyed by XML tree tuple items.

2. We conceive a *transactional model* for representing the XML tree tuples extracted from a collection of XML documents. Such a model is at the basis of a *semantic XML clustering* framework. We adopt an effective partitional approach based on an algorithm designed for the XML transactional domain, although the proposed framework is conceived to be parametric w.r.t. any method of clustering.

*A. Tagarelli and S. Greco are with the DEIS Dept. of University of Calabria, Via Bucci 41c, 87036 Rende (CS) – Italy. E-mail addresses: {tagarelli,greco}@deis.unical.it

3. We conduct on large real datasets several experiments aimed at assessing the ability of the framework in performing structure- or content-driven clustering, as well as discovering clusters of "hybrid" type. Empirical evidence argues that the proposed framework is highly effective and shows a good scalability.

Related work. Several approaches to mining XML data have been recently devised, mostly focusing on clustering XML documents by structure. This is mainly motivated by several applications in the management of semistructured data, especially in the Web environment [1, 17], which have raised the demand for effective and efficient solutions to the problem of identifying structural similarities among semistructured data. In this context, [12] proposes an XML-aware edit distance to measure structural similarity among XML documents, and applies a standard hierarchical clustering algorithm to evaluate how closely cluster documents correspond to their respective DTDs. However, in general, computing tree edit distances turns out to be unpractical, as it requires a quadratic number of comparisons between document elements.

An important insight into the task of supervised classification of XML data from the structure viewpoint is provided in [19]. The authors propose a machine learning technique that exploits mining of substructures frequently occurring in XML documents in order to define classification rules.

More recently, it has been raised the importance of summarizing XML data, mainly for the purpose of defining syntheses for clusters of structurally similar XML data. Approaches exploiting graph-based representation models [11] as well as tree matching [5] have been proposed. XSketch [13] is a refined synopsis model that addresses the problem of approximating path-expression selectivities in the general setting of graph-structured XML data with element values. The construction of an accurate synopsis is based on some statistical assumptions that compensate for the lack of detailed path and value information in the synopsis. However, building an effective XSketch has been demonstrated to be an NP-hard problem, thus a heuristic synopsis refinement strategy must be used.

The need for organizing XML data according to both their content and structural features has become challenging, due to the increase of heterogeneity of XML sources. However, mining XML data from a content/structure combination viewpoint is still in a preliminary stage, and no existing approach provides effective capabilities for semantic XML clustering. A first attempt is given in [6], which proposes to apply a partitional clustering technique to XML documents represented in a vector-space model by textual and tag-based features.

An alternative representation, called BitCube, is presented in [18] as a 3-dimensional bitmap index of triplets ⟨document, XML-element path, word⟩. BitCube indexes can be manipulated to partition documents into clusters, by exploiting bit-wise distance and popularity measures. In order to speed up query answering, slice/dice/project operations are performed to sub-cubes resulting from the clustering phase. However, no critical discussion is provided by the authors about possible improvements of document clustering. In general, the approach suffers from typical disadvantages of boolean representation models, such as the lack of partial matching criteria and natural measures of document ranking.

Generation of features for XML data is more deeply investigated in [16], where annotations, structure and ontological information are combined together. However, the focus here is on building appropriate features for purposes of supervised classification of XML data.

Plan of the paper. The rest of the paper is organized as follows. Section 2 provides terminology and notations useful for our purposes, and necessary background on the notions of tree tuple, item, and transaction for the domain of XML data. Section 3 describes how to generate *XML features* by semantically enriching syntactic information in XML tree tuple items with lexical ontology knowledge. Particular emphasis in this section is posed on a novel method for tag sense disambiguation. Section 4 presents a framework for clustering XML transactions. Section 5 reports experimental evaluation stating the effectiveness of the clustering framework. Section 6 presents concluding remarks and promising points for future research.

2 Background

2.1 XML trees and paths. A *tree* T is a tuple $T = \langle r_T, N_T, E_T, \lambda_T \rangle$, where $N_T \subseteq \mathbb{N}$ denotes the set of nodes, $r_T \in N_T$ is the distinguished root of T, $E_T \subseteq N_T \times N_T$ denotes the (acyclic) set of edges, and $\lambda_T : N_T \mapsto \Sigma$ is a function associating a node with a label in the alphabet Σ. Let Tag, Att, and Str be alphabets of tag names, attribute names and strings, respectively. An *XML tree* XT is a pair $XT = \langle T, \delta \rangle$, such that: *1)* T is a tree defined on the alphabet $\Sigma = Tag \cup Att \cup \{\mathtt{S}\}$, where symbol $\mathtt{S} \notin Tag \cup Att$ is used to denote the `#PCDATA` content model; *2)* given $n \in N_T$, $\lambda_T(n) \in Att \cup \{\mathtt{S}\} \Leftrightarrow n \in Leaves(T)$; *3)* $\delta : Leaves(T) \mapsto Str$ is a function associating a string to a leaf node of T.

An *XML path* p is a sequence $p = s_1.s_2\ldots .s_m$ of symbols in $Tag \cup Att \cup \{\mathtt{S}\}$. Symbol s_1 corresponds to the tag name of the document root element. An XML path can be of two types: *tag path*, if $s_m \in Tag$, or *complete path*, if $s_m \in Att \cup \{\mathtt{S}\}$. We denote as $\mathcal{P}_{XT}$ the set of complete paths in XT.

Let $XT = \langle T, \delta \rangle$ be an XML tree, and $p = s_1.s_2\ldots .s_m$ be an XML path. The application of p to XT identifies a set of nodes $p(XT) = \{n_1, \ldots, n_h\}$ such that, for each $i \in [1..h]$, there exists a sequence of nodes, or *node path*, $np_i^p = [n_{i_1}, \ldots, n_{i_m}]$ with the following properties: *1)* $n_{i_1} = r_T$ and $n_{i_m} = n_i$; *2)* $n_{i_{j+1}}$ is a child of n_{i_j}, for each $j \in [1..m\text{-}1]$; *3)* $\lambda(n_{i_j}) = s_j$, for each $j \in [1..m]$.

Moreover, we say that the application of a path to an XML tree yields an *answer*, which is defined depending on the type of path. In case of a tag path p, the answer of p on XT is exactly the set of node identifiers $p(XT)$, that is $\mathcal{A}_{XT}(p) \equiv p(XT)$. For a complete path p, the answer of p on XT is defined as the set of string values associated to the leaf nodes identified by p, that is $\mathcal{A}_{XT}(p) = \{\delta_T(n) \mid n \in p(XT)\}$.

2.2 XML tree tuples. Tree tuples resemble the notion of tuples in relational databases and have been proposed to extend functional dependencies to the XML setting [2, 8]. In a relational database, a tuple is a function assigning each attribute with a value from the corresponding domain. According to [8], we provide the following definition.[1]

Definition 2.1. *Given an XML tree XT, a* tree tuple *τ is a maximal subtree of XT such that, for each (tag or complete) path p in XT, the answer $\mathcal{A}_\tau(p)$ contains at most one element.*
We denote as $\mathcal{T}_{XT}$ the set of tree tuples from XT.

Intuitively, a tree tuple is a (sub)tree representation of a complete set of distinct concepts that are correlated according to the structure semantics of the original tree. Moreover, tree tuples extracted from the same tree maintain identical structure while reflect different ways of associating content with structure as they can be naturally inferred from the original tree.

Example 1. *Consider the XML tree shown in Fig. 1, which represents two journal articles from the DBLP archive. Any internal node has a unique label denoting a tag name, whereas each leaf node is labeled with either name and value of an attribute, or symbol* `S` *and a string corresponding to the* `#PCDATA` *content model. Path answers can be easily computed: for example, path* `dblp.article.title` *yields the set of node identifiers $\{n_8, n_{22}\}$, whereas path* `dblp.article.author.S` *yields the set of strings {'Hartmut Liefke', 'Dan Suciu'}.*

Three tree tuples can be extracted from the example tree (Fig. 2). One tree tuple is extracted starting from the right subtree rooted in the `dblp` *element. Two tree tuples are instead extracted starting from the left subtree rooted in* `dblp`*, as in this subtree there are two paths* `dblp.article.author`*, each of which yields a distinct path answer corresponding to a paper author.*

2.3 A transactional model for XML tree tuples. In the huge amount of available structured data, a relevant portion is represented by *transactional data*, i.e. variable-length sequences of objects with categorical attributes. The interest in analyzing this particular domain arises from many challenging application problems (e.g. analysis of Web access logs) that can be formalized using transactional data. Given a set $\mathcal{I} = \{e_1, \ldots, e_m\}$ of distinct categorical values, or *items*, a transactional database is a multi-set of transactions $tr \subseteq \mathcal{I}$.

In our setting, the item domain is built over all the leaf elements in a collection of XML tree tuples, that is the set of distinct answers of complete paths applied to the tree tuples. A transaction is modeled with the set of items associated to the leaf elements of a specific tree tuple. The intuition behind such a model lies mainly on the definition of tree tuples itself: each path applied to a tree tuple yields a unique answer, thus each item in a transaction indicates information on a concept which is distinct from that of other items in the same transaction.

Definition 2.2. *Let τ be a tree tuple and $p \in \mathcal{P}_\tau$ be a complete path in τ. The pair $e = \big(p, \mathcal{A}_\tau(p)\big)$ defines a* tree tuple item *in τ.*

Let $\mathcal{I}_\tau = \{\big(p, \mathcal{A}_\tau(p)\big) \mid p \in \mathcal{P}_\tau\}$ denote the set of items of a tree tuple τ. Given an XML tree XT, the associated set of items is defined as $\mathcal{I}_{XT} = \bigcup_\tau \mathcal{I}_\tau$, with $\tau \in \mathcal{T}_{XT}$. Analogously, given a collection $\mathcal{XT} = \{XT_1, \ldots, XT_n\}$ of XML trees, the associated set of items, or *item domain*, is defined as $\mathcal{I}_{\mathcal{XT}} = \bigcup_{XT \in \mathcal{XT}} \mathcal{I}_{XT}$.

The set $\mathcal{I}_\tau$ of tree tuple items in τ is referred to as the *XML transaction* associated to τ. Given a collection $\mathcal{XT} = \{XT_1, \ldots, XT_n\}$ of XML trees, the *XML transaction dataset* $\mathcal{S}$ is defined as $\mathcal{S} = \bigcup_{XT \in \mathcal{XT}} \mathcal{S}_{XT}$, where $\mathcal{S}_{XT} = \{\mathcal{I}_\tau \mid \tau \in \mathcal{T}_{XT}\}$.

Some remarks can be highlighted regarding the impact of XML transactions on critical aspects such as

[1] In [8], an XML document is assumed to conform to a predefined DTD. However, in our context, the presence of an XML schema is not necessary to extract tree tuples.

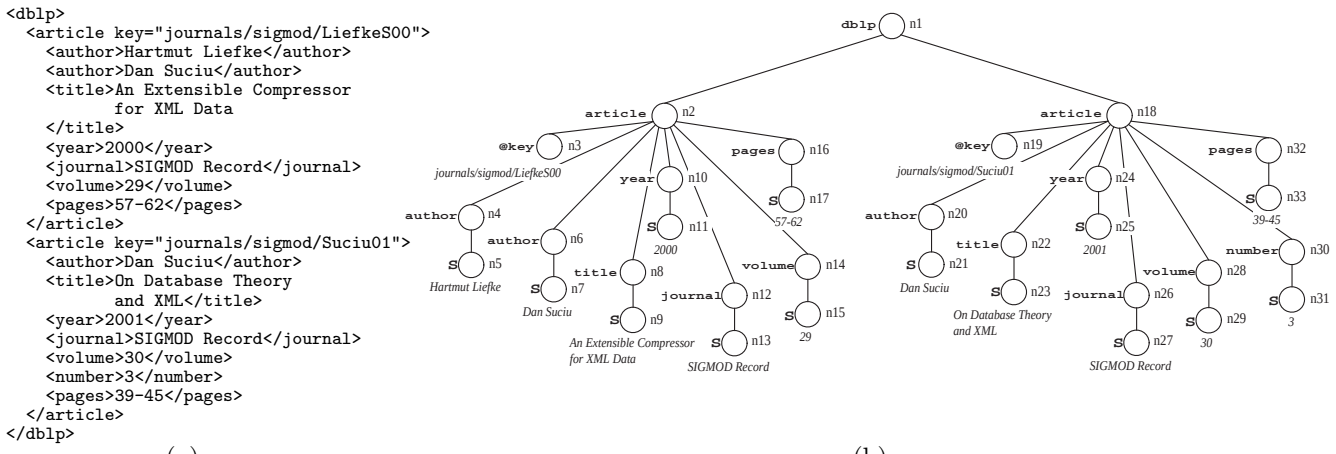

Figure 1: Example DBLP XML document and its tree

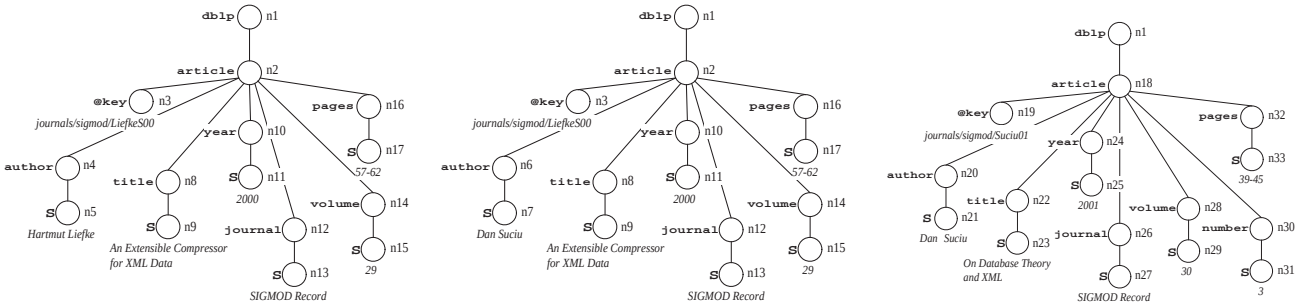

Figure 2: The tree tuples extracted from the XML tree of Fig. 1(b)

missing elements and fragmentation. High heterogeneity of relational schemes, which can be derived from XML schemes under some mapping criteria, generally brings about many missing values; transactions overcome this traditional drawback since items always refer to non-null values. Concerning fragmentation, this may be a negative outcome of excessive decomposition of original XML documents, and cause serious overhead in query evaluation; however, XML tree tuples substantially act as "features" for clustering semistructured data, whereas further query processing can be carried out on original XML documents.

EXAMPLE 2. *In order to model tree tuples as transactions, each tuple can be decomposed into distinct paths and respective answers, as shown in Fig. 3(a). For example, the application of* `dblp.article.@key` *yields the attribute value 'journals/sigmod/LiefkeS00' corresponding to node n_3 of τ_1; item e_1 is then associated to this pair path-answer. Yet, the answer of* `dblp.article.journal.S` *is the string 'SIGMOD Record' corresponding to nodes n_{27} of τ_3 and n_{13} of τ_1 and τ_2.*

Once the item domain has been completely defined, a transaction is assigned with each tree tuple by mapping its pairs path-answer into the corresponding items. A transactional representation of the tree tuples of Fig. 2 is shown in Fig. 3(c).

3 Extracting XML Features

3.1 Structure features. Tag paths represent the natural basis for extracting structural features from XML data. However, although important information can be inferred from XML tags, subjective factors reflect the style of document authors in coding information to XML, thus informative consistency among XML data is not usually guaranteed. The key idea lies in going beyond a context-free use of mere terms, i.e. tag names, by mapping them into semantically related concepts. Each concept belongs to an ontological space and is represented by a lexical meaning, or *sense*, to be associated to a tag name. *Lexical ontology knowledge* can be hence exploited to semantically enrich features to be extracted from XML tag names.

path (p)	$\mathcal{A}_{\tau_1}(p)$	*node ID*
dblp.article.@key	'journals/sigmod/LiefkeS00'	n_3
dblp.article.author.S	'Hartmut Liefke'	n_5
dblp.article.title.S	'An Extensible Compressor ...'	n_9
dblp.article.year.S	'2000'	n_{11}
dblp.article.journal.S	'SIGMOD Record'	n_{13}
dblp.article.volume.S	'29'	n_{15}
dblp.article.pages.S	'57-62'	n_{17}

path (p)	$\mathcal{A}_{\tau_2}(p)$	*node ID*
dblp.article.@key	'journals/sigmod/LiefkeS00'	n_3
dblp.article.author.S	'Dan Suciu'	n_7
dblp.article.title.S	'An Extensible Compressor ...'	n_9
dblp.article.year.S	'2000'	n_{11}
dblp.article.journal.S	'SIGMOD Record'	n_{13}
dblp.article.volume.S	'29'	n_{15}
dblp.article.pages.S	'57-62'	n_{17}

path (p)	$\mathcal{A}_{\tau_3}(p)$	*node ID*
dblp.article.@key	'journals/sigmod/Suciu01'	n_{19}
dblp.article.author.S	'Dan Suciu'	n_{21}
dblp.article.title.S	'On Database Theory ...'	n_{23}
dblp.article.year.S	'2001'	n_{25}
dblp.article.journal.S	'SIGMOD Record'	n_{27}
dblp.article.volume.S	'30'	n_{29}
dblp.article.number.S	'3'	n_{31}
dblp.article.pages.S	'39-45'	n_{33}

(a)

item ID	*associated node IDs*
e_1	n_3
e_2	n_5
e_3	n_9
e_4	n_{11}
e_5	$n_{13},\ n_{27}$
e_6	n_{15}
e_7	n_{17}
e_8	$n_7,\ n_{21}$
e_9	n_{19}
e_{10}	n_{23}
e_{11}	n_{25}
e_{12}	n_{29}
e_{13}	n_{31}
e_{14}	n_{33}

(b)

tr_1	$e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7$
tr_2	$e_1\ e_8\ e_3\ e_4\ e_5\ e_6\ e_7$
tr_3	$e_9\ e_8\ e_{10}\ e_{11}\ e_5\ e_{12}\ e_{13}\ e_{14}$

(c)

Figure 3: Transactional representation of the tree tuples of Fig. 2: (a) tree tuples with paths and answers, (b) item domain, and (c) transaction set

Due to its increasing scope and public availability, the lexical database *WordNet* [7] is used in this work as ontology knowledge base. WordNet groups words with the same meaning into equivalence classes, called *synsets* (sets of synonyms). Each synset represents a concept and is described by a short textual description, called *gloss*. Synsets are explicitly connected through predefined relations. In particular, noun synsets are connected through *is-a* (hypernymy/hyponymy) and *part-of* relations (meronymy/holonymy). We expect to use mainly the noun portion of WordNet, since nouns are much more heavily used to annotate XML data.

Tag sense disambiguation. Mapping tag names into an ontological concept space needs to address the issue of deciding the most appropriate sense for each tag name. This can be accomplished by performing *word sense disambiguation* (WSD), that is assigning a word with a sense based on the context in which the word appears.

The proposed approach to the disambiguation of senses for tag names consists in selecting the most appropriate senses w.r.t. a *path-context*. A path-context is represented by a semantic network built on all the possible senses associated to the tags of a specific path. For each tag path $p = t_1.t_2\ldots\ .t_n$, a directed weighted graph $SG(p)$, called *synset graph* of p, is used to disambiguate the senses of the tags in p. $SG(p)$ is defined as follows:

- Nodes are pairs $\langle t_i, \sigma\rangle$, with $i \in [1..n]$ and $\sigma \in senses(t_i)$, where $senses(t_i)$ denotes the set of senses for word (tag) t_i; moreover, additional nodes *source* and *sink* are given for purposes of convenient visits through the graph.
- Edges are connections between nodes of contiguous tags ($\langle t_i, \sigma\rangle$, $\langle t_{i+1}, \rho\rangle$), with $i \in [1..n\text{-}1]$; moreover, edges $(source, \langle t_1, \sigma\rangle)$ and $(\langle t_n, \rho\rangle, sink)$ hold, for each $\sigma \in senses(t_1)$, $\rho \in senses(t_n)$.
- Edge weights are computed to reflect the semantic relatedness between the concepts associated to any

two nodes $\langle t_i, \sigma \rangle, \langle t_{i+1}, \rho \rangle$; weights on edges involving either *source* or *sink* are set to 0.

Once $SG(p)$ has been built, the disambiguation of the tag names in p is accomplished by finding the *maximum-weight path* in $SG(p)$, so that the most appropriate senses are those corresponding to this graph path. In case of multiple maximum-weight graph paths, the preferred one can be computed by exploiting the dictionary-supplied linear order of synsets based on the lexicographer identifiers for senses. It is easy to observe that computing a maximum-weight path is linear in the number of edges, due to the layered form of the synset graph.

The crucial aspect in the construction of the synset graph is how to compute the edge weights, that is how to compute the semantic relatedness between senses. We now investigate this aspect in detail.

In dictionary-based WSD the assumption is that the most plausible sense to assign to multiple co-occurring words is the one that maximizes the relatedness among the chosen senses. Within this view, the pioneer Lesk method [10] disambiguates a target word by choosing the meaning whose gloss shares the largest number of words with the glosses associated to neighboring words.

The use of a lexical ontology, like WordNet, allows for capturing the semantic relationships lying on concepts by exploiting also hierarchies of concepts besides dictionary glosses. The basic Lesk algorithm can be hence enhanced in order to take advantage of the network of relations provided in WordNet. This idea has been formalized in a measure of semantic relatedness between word senses based on the notion of *extended gloss overlap* [4], which has the merit of considering phrasal matches and weighting them more heavily than single word matches. The extended gloss overlap measure takes as input two concepts (i.e. two WordNet synsets) and computes a gloss overlap score, hereinafter denoted with *score*, as the sum of the squared sizes of the distinct overlaps between the glosses. An *overlap* is detected whenever a shared maximal sequence of words occurs. To finally measure the semantic relatedness between two synsets a and b, the gloss overlap scoring function is used to compare not only a's gloss with b's gloss but also pairs of glosses of those synsets to which a and b are directly connected through a certain WordNet relation. More precisely, comparisons include combination of a's hypernym with b's hypernym, a's hyponym with b's hyponym, a's hypernym with b, and a with b's hypernym. Further details can be found in [4].

Following on this direction, we maintain the above mechanism of scoring of gloss overlaps, while we differently exploit the WordNet concept taxonomy to determine the relatedness between synsets. On one hand, we consider different combinations of relations over synsets. Indeed, tag names in an XML path are related by an order that allows for inferring concept hierarchies, or even lattices. On the other hand, we do not take into account only direct connections but also indirect connections to a target synset.

Let $\mathcal{WSR}$ denote a selected set of WordNet synset relations. We assume that $\mathcal{WSR}$ consists of the following relations: *hypernymy*, *hyponymy*, *meronymy*, and *holonymy*. Given a relation $r \in \mathcal{WSR}$ and a synset a, we define a function ω such that, applied to r and a, yields the set $\omega(r, a)$ of synsets directly connected to a through the WordNet relation r.

Function ω can be extended to include synsets that are indirectly connected to a target synset at a given distance in the WordNet taxonomy. Given two synsets a and a' and fixed a WordNet relation, there exists a unique path in the WordNet taxonomy that leads from a to a'. Given a relation $r \in \mathcal{WSR}$, a synset a and an integer value $\overline{d}$, the set of pairs (a', d) such that a' is a synset connected to a, through r, across a path of length $d \leq \overline{d}$ is defined as

$$\omega^*(r, a, \overline{d}) = \begin{cases} \bigcup_{a' \in \omega(r,a)} \omega^*(r, a', \overline{d} - 1) & \text{if } \overline{d} > 1 \\ \omega(r, a) & \text{if } \overline{d} = 1 \end{cases} .$$

Let $a = \langle t_i, \sigma \rangle$ and $b = \langle t_{i+1}, \rho \rangle$ be two synsets corresponding to nodes in a synset graph, and let $\overline{d}$ be an integer value. The weight on the edge between a and b is computed as:

$$\begin{aligned} &weight(a, b, \overline{d}) = score(a, b) \\ &+ \textstyle\sum_{(b',d) \in \omega^*(\text{hype}, b, \overline{d}) \ \vee \ (b',d) \in \omega^*(\text{holo}, b, \overline{d})} score(a, b') \times \mathrm{f}(d) \\ &+ \textstyle\sum_{(a',d) \in \omega^*(\text{hypo}, a, \overline{d}) \ \vee \ (a',d) \in \omega^*(\text{mero}, a, \overline{d})} score(a', b) \times \mathrm{f}(d), \end{aligned}$$

where $\mathrm{f}(d)$ is a function monotonically decreasing for increasing values of d. In our experimental setting, we fix $\mathrm{f}(d)$ to an inverse exponential function, while the maximum distance value $\overline{d}$ is set to 3. Notice that $weight(a, b)$ is not the same as $weight(b, a)$. This asymmetry is justified in the context of XML path — a corresponds to a tag name that typically represents a hypernym or holonym of the tag name associated to b.

3.2 Content features. Content features are generated by discovering patterns from text elements. We refer to *textual content unit* (for short, TCU) as the text extracted from any leaf node of an XML tree (i.e. a `#PCDATA` element content or an attribute value). TCUs can be represented by adopting a conventional bag-of-words model, and are subject to both lexical and semantic analysis. The former aims at selecting syntactically significant features (index terms), by means of language-specific text preprocessing operations, such as

removal of stopwords and word stemming [3]. In addition, semantic analysis can be applied to leverage the lexical ambiguity problem by examining the degree of polysemy of terms.

The above two tasks of text analysis determine the relevance weighting for index terms. Indeed, we can consider syntactic relevance and semantic relevance. The former is typically weighed according to information on the frequency of occurrence of a term, w.r.t. a content context. On the other hand, syntactic information can be "semantically enriched" by resorting to the notion of *semantic rarity* of terms. We shall address this point in the next section.

4 XML Transactional Clustering

4.1 XML tree tuple item similarity. In our setting XML features are embedded in tree tuple items. The notion of similarity between tree tuple items is hence function of the similarity between their respective structure and content features.

DEFINITION 4.1. *Let e_i and e_j be two tree tuple items. The* tree tuple item similarity *function is defined as*

$$sim(e_i, e_j) = f \times sim_S(e_i, e_j) + (1 - f) \times sim_C(e_i, e_j),$$

where sim_S (resp. sim_C) denotes the structural (resp. content) similarity between the items, and $f \in [0..1]$ is a factor that tunes the influence of the structural part to the overall similarity.

Moreover, let $\gamma \in [0..1]$ be a similarity threshold. We say that two tree tuple items e_i and e_j are γ-*matched* if and only if they have a "match" at a degree greater than or equal to γ, that is $sim(e_i, e_j) \geq \gamma$.

Notice that tree tuple item similarity is defined as a linear function since intuitively it enables us to easily control and understand the different contributions coming from structure and content features. In the following we gain an insight into the notions of structural and content similarity between tree tuple items.

Similarity by structure. Structural similarity between two tree tuple items e_i and e_j is evaluated by comparing their respective tag paths and computing the average similarity between the senses of the respective *best matching tags*. Given two paths p_i, p_j and a tag $t \in p_i$, we denote as $bm(p_j, t) = \{t' \in p_j \mid \nexists t'' \in p_j,\ t'' \neq t',\ sim(t.\sigma, t''.\sigma) > sim(t.\sigma, t'.\sigma)\}$ the set of tags of p_j with which t has the best match. We assume that each tag t is associated with the most appropriate sense $t.\sigma$ that has been selected by performing a process of tag sense disambiguation, as described in Sect. 3.1.

DEFINITION 4.2. *Let e_i and e_j be two tree tuple items, and $p_i = t_{i_1}.t_{i_2}\ldots\ .t_{i_n}$, $p_j = t_{j_1}.t_{j_2}\ldots\ .t_{j_m}$ be their respective tag paths. The* structural similarity *between e_i and e_j is defined as*

$$sim_S(e_i, e_j) = \frac{1}{n+m}\Bigg(\sum_{t \in p_i} \sum_{t' \in bm(p_j,t)} \frac{sim(t.\sigma, t'.\sigma)}{|bm(p_j, t)|} + \sum_{t \in p_j} \sum_{t' \in bm(p_i,t)} \frac{sim(t.\sigma, t'.\sigma)}{|bm(p_i, t)|} \Bigg),$$

where $sim(t.\sigma, t'.\sigma)$ computes the similarity between the senses respectively assigned to tags t and t'.

In order to define a suitable measure of similarity between tag name senses, we exploit both lengths of paths in the concept taxonomy and co-occurrences of senses. This approach lies on two main observations.

In a hierarchy of concepts, path-based measures allow for determining the degree to which two concepts are related. However, path lengths should be interpreted differently depending on where the concepts are located in the hierarchy, since the higher a concept in a hierarchy the more general itself. A suitable path-based measure is defined in [14], which focuses on the notion of lowest common subsumer (lcs) to compute the most specific sense shared by two senses.

Besides the semantic relatedness among word senses, we are also interested in estimating the sense specificity. More precisely, we estimate the specificity of a sense σ through the occurrences of every tag name associated with σ in a given corpus.

Faced with the above observations, the similarity between tag senses σ_1 and σ_2 is computed by combining path-based and co-occurrence measures.

DEFINITION 4.3. *Let $\mathcal{S}$ be a set of XML transactions, and σ_1, σ_2 be two tag senses. The* sense similarity *between σ_1 and σ_2 w.r.t. $\mathcal{S}$ is defined as:*

$$sim(\sigma_1, \sigma_2) = \frac{2 \times depth(lcs(\sigma_1, \sigma_2))}{depth(\sigma_1) + depth(\sigma_2)} \times \frac{freq(\sigma_1, \sigma_2, \mathcal{S})}{freq(\sigma_1, \mathcal{S}) + freq(\sigma_2, \mathcal{S}) - freq(\sigma_1, \sigma_2, \mathcal{S})},$$

where $depth(\sigma)$ is the distance from the concept node for σ to the root of the hierarchy, $freq(\sigma_i, \mathcal{S})$ is the number of XML transactions in $\mathcal{S}$ containing a tag t_i such that $t_i.\sigma = \sigma_i$, and $freq(\sigma_1, \sigma_2, \mathcal{S})$ denotes the number of XML transactions in $\mathcal{S}$ containing both a tag t_i and a tag t_j such that $t_i.\sigma = \sigma_1$ and $t_j.\sigma = \sigma_2$.

Similarity by content. Index terms extracted from TCUs can be weighted according to both syntactic and semantic relevance. From the syntactic viewpoint, we consider two statistical criteria: term density in a text and term rarity in the text collection. The common

tf.idf weighting function is just defined to take both criteria into account [3]. In our setting, given any term w_j and any TCU u_i, the *tf.idf* weight is computed as

$$tf.idf(w_j, u_i) = freq(w_j, u_i) \times \log\left(\frac{N}{n_j}\right),$$

where $freq(w_j, u_i)$ denotes the number of occurrences of w_j in u_i, N is the total number of TCUs counted over the collection of tree tuples, and n_j is the number of TCUs containing w_j.

Conventional term weighting functions take into account only lexical information to assess the relevance of a term. Our idea is to enrich the *tf.idf* function by exploiting the *semantic rarity* of a term. We define this notion by resorting to the degree of polysemy of a term, in such a way that the relevance of a term is reduced as many meanings as the term has. Formally, the semantic rarity of a term w is evaluated as

$$s\text{-}rarity(w) = \log(\frac{\text{MAX\_POLYSEMY}}{|senses(w)|}),$$

where $senses(w)$ denotes the set of meanings of w and MAX_POLYSEMY is a constant denoting the number of meanings of the most polysemous word in WordNet. The logarithmic function incorporates the effect of favoring terms with typical lower polysemy. Moreover, we assume that any term has at least one meaning, even if that term is not present in the reference dictionary.

The semantic rarity function is invariant w.r.t. the term location in the collection of textual content units. Therefore, for each term w_j, the same value $s\text{-}rarity(w_j)$ can be combined with the *tf.idf* value associated to any pair (w_j, u_i).

DEFINITION 4.4. *The* relevance weight *of a term w_j w.r.t. a textual content unit u_i is computed as*

$$relevance(w_j, u_i) = tf.idf(w_j, u_i) \times s\text{-}rarity(w_j).$$

A measure particularly suitable for assessing similarity between documents is the *cosine similarity* [3]: it computes proximity in terms of the cosine of the angle that two generic documents form in a multidimensional space. In our context, any TCU u_i can be associated with a term vector $\vec{u}_i$ whose j-th component contains the value $relevance(w_j, u_i)$. Given two XML elements e_i and e_h, the content similarity between e_i and e_h is computed by measuring the cosine similarity between the term vectors associated with their respective TCUs: $sim_C(e_i, e_h) = \frac{\vec{u}_i \cdot \vec{u}_h}{\|\vec{u}_i\| \times \|\vec{u}_h\|}$.

4.2 The *XTrK-means* algorithm.

XML tree tuples modeled as transactions can be efficiently clustered by applying a partitional algorithm devised for the XML transactional domain.

A partitional clustering problem consists in partitioning a set $\{x_1, \ldots, x_n\}$ of objects in k non-empty groups each containing a homogeneous subset of objects. An important class of partitional approaches is based on the notion of cluster *centroid*, or *representative*: each object x_i is assigned to a cluster C_j according to its distance from a value c_j, called centroid of C_j.

A simple but effective partitional algorithm for clustering generic transactional data is *TrK-means* [9]. This algorithm consists of two main phases. In the first phase, it works as a traditional centroid-based method to compute $k+1$ clusters: starts choosing k objects as the initial cluster centroids, then iteratively reassigns each remaining object to the closest cluster until all cluster centroids do not change. The $(k+1)$-th cluster, called *trash cluster*, is created to contain unclustered objects, i.e. objects having an empty intersection with each cluster centroid and so are not assigned to any of the first k clusters. The second phase is in charge of recursively splitting the trash cluster into a small number of clusters.

The TrK-means algorithm provides a suitable notion of cluster centroid exploiting a compact representation model based on a refinement of the intersection of transactions within a cluster. Indeed, when we compare two transactions we are searching for the "shared" items. Intuitively, we could evaluate the similarity between two transactions as directly proportional to the number of common items and inversely proportional to the number of different items. The basic TrK-means exploits the *Jaccard coefficient* to compute the similarity between two sets as the ratio of the cardinality of their intersection to the cardinality of their union.

However, computing exact intersection between transactions of tree tuple items may turn out to be not effective. Indeed, two items may share relevant structural or content information even though they are not identical. For this purpose, we enhance the notion of standard intersection between sets of items with one able to capture even minimal similarities from content and structure features of XML elements.

DEFINITION 4.5. *Let tr_1 and tr_2 be two XML transactions, and $\gamma \in [0..1]$ be a similarity threshold. The set of* γ-shared items *between tr_1 and tr_2 is defined as*

$$match^{\gamma}(tr_1, tr_2) = match^{\gamma}(tr_1 \rightarrow tr_2) \cup match^{\gamma}(tr_2 \rightarrow tr_1),$$

where

$$match^{\gamma}(tr_i \rightarrow tr_j) = \{e \in tr_i \mid \exists e_h \in tr_j,\ sim(e, e_h) \geq \gamma, \nexists e' \in tr_i, sim(e', e_h) > sim(e, e_h)\}.$$

The set of γ-shared items resembles the intersection between transactions at a degree greater than or equal

Input:
A set $\mathcal{S} = \{tr_1, \ldots, tr_n\}$ of XML transactions,
the desired number k of clusters, a similarity threshold γ.
Output:
A partition $\mathcal{C}$ of $\mathcal{S}$ in $k + l$ clusters, where $l \geq 0$.
Method:
$\mathcal{C} := \emptyset$;
select $Seeds = \{tr_{i_1}, \ldots, tr_{i_k}\}$ from distinct original trees;
for each $tr_i \in Seeds$ **do** /* *initializes partition of clusters* */
$C_i := \{tr_i\}$; $c_i := tr_i$;
$\mathcal{C} := \mathcal{C} \cup C_i$;
repeat
$C_j := \{tr_i | sim_J^\gamma(tr_i, c_j) > sim_J^\gamma(tr_i, c_h), h \in [1..k]\}, \forall j \in [1..k]$;
$C_{k+1} := \{tr_i | sim_J^\gamma(tr_i, c_j) = 0, \forall j \in [1..k]\}$;
$c_j :=$ computeRepresentative(C_j), $\forall j \in [1..k]$;
until $Q(\mathcal{C})$ is maximized;
/* *partitions trash cluster* */
apply previous steps (except one for further trash clustering)
to partition C_{k+1} into $l = \sqrt{k}$ clusters;
$\mathcal{C} := \{C_1, \ldots, C_k, C_{k+1,1}, \ldots, C_{k+1,l}\}$;
return $\mathcal{C}$;

Function computeRepresentative$(C) : rep$;
$R := \emptyset$;
for each $tr_i \in C$ **do**
$r_i := \bigcup_{tr_j \in C} match^\gamma(tr_i, tr_j)$; $R := R \cup \{r_i\}$;
$I_C := \{e \mid e \in r,\ r \in R\}$;
rank items in I_C by decreasing frequency;
let $I'_C \subseteq I_C$ be the set of items in I_C with the highest freq.;
$rep := conflateItems(I'_C)$;
/* *refines representative* */
$s_0 := \sum_{tr \in C} sim_J^\gamma(tr, rep)$; $rep' := rep$;
while $(I_C - I'_C \neq \emptyset)$ **do**
add the next item $e \in I_C - I'_C$ to rep';
$rep' := conflateItems(rep')$;
$s' := \sum_{tr \in C} sim_J^\gamma(tr, rep')$;
if $(s' \geq s_0)$ **then**
$I'_C := I'_C - \{e\}$;
$s_0 := s'$;
else
$I_C - I'_C := \emptyset$;
$rep := rep' - \{e\}$;
return rep;

Figure 4: The *XTrK-means* algorithm

to a similarity threshold γ. This notion of (enhanced) intersection is also at the basis of the following similarity function.

DEFINITION 4.6. *Let tr_1 and tr_2 be two XML transactions, and $\gamma \in [0..1]$ be a similarity threshold. The* XML transaction similarity *function between tr_1 and tr_2 is defined as*

$$sim_J^\gamma(tr_1, tr_2) = \frac{|match^\gamma(tr_1, tr_2)|}{|tr_1 \cup tr_2|}.$$

We have adapted the TrK-means algorithm focusing our effort mainly on conceiving suitable notions of proximity among XML transactions and cluster representative. Fig. 4 sketches the main phases of the resulting algorithm, called *XTrK-means*, whose salient characteristics are discussed in the following.

Firstly, the XML transaction similarity function replaces the traditional Jaccard coefficient as a measure of proximity among transactions.

Secondly, the representative of any cluster C is computed by starting from the set of γ-shared items among all the transactions within C. More precisely, for each transaction in C, the union of the γ-shared item sets w.r.t. all the other transactions in C is obtained; this guarantees no dependence of the order of examination of transactions. Then, a raw representative is computed by selecting from these union sets the items with the highest frequency. The raw representative, however, may not have the form of a tree tuple, as some items therein may refer to the same path but with different answers. Function $conflateItems$ is applied to a set of items and, for each subset I of items sharing the same path p, yields one item that has p as path and the concatenation of contents of the items in I as its content. Finally, a greedy heuristic refines the current representative by iteratively adding the remaining most frequent items until the sum of pair-wise similarities between transactions and representative cannot be further maximized. Again, any refinement must guarantee that the resulting representative is a tree tuple.

Moreover, in *XTrK-means* tree tuples selected as initial cluster centroids are constrained to come from different XML documents. This simple strategy aims to favor the construction of clusters with low inter-similarity.

Finally, the exit criterion in *XTrK-means* requires the quality of the resulting cluster partition is maximized. As usual in a clustering task, this means that overall cluster cohesiveness is maximized while similarity between clusters is minimized. By exploiting pair-wise similarities between transactions, which compose or represent a cluster, intra- and inter-similarity (for a partition $\mathcal{C}$ of k clusters) are respectively defined as follows:

$$IntraSim(\mathcal{C}) = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{tr \in C_i} sim_J^\gamma(tr, c_i)$$

$$InterSim(\mathcal{C}) = \frac{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} sim_J^\gamma(c_i, c_j)}{\frac{1}{2}k(k-1)}.$$

The *quality* of a clustering $\mathcal{C}$ is defined as $Q(\mathcal{C}) = IntraSim(\mathcal{C}) - InterSim(\mathcal{C})$. If we assume that both intra-similarity and inter-similarity have values within [0..1] and that $IntraSim(\mathcal{C}) \geq InterSim(\mathcal{C})$ holds for good clusterings, then $Q(\mathcal{C})$ also ranges within [0..1].

Clearly, an alternative, less expensive exit criterion consists in checking whether clusters are stable, that is checking whether cluster centroids in the current iteration are not changed w.r.t. the previous iteration.

However, the use of an internal cluster validity criterion allows us to both evaluate and drive a clustering task.

5 Experimental Evaluation

Data description. To evaluate the proposed clustering framework, we considered real XML datasets having different characteristics according to three main aspects: dataset composition (single document or collection of documents), structure complexity in terms of degree of element nesting, and content impact w.r.t. size and number of textual elements. The latter aspect is also a crucial reason we had to leave synthetic datasets out of consideration: indeed, they do not provide elements containing coherent natural language texts, but at most (e.g. XMark [15]) use an automatic generator that takes frequently occurring words in a fixed prose text as feed to mimic a real statistical text distribution.

Three real XML data sources were used to conduct experiments: the DBLP archive, the Reuters RSS news channel, and PubMed. *DBLP* (http://dblp.uni-trier.de/xml/) is a digital bibliography mainly concerning journal articles, conference papers, books, book chapters, and theses on computer science. The XML version of this archive is represented by a single huge document, which can be decomposed (according to its DTD) in several thousands of XML documents. The *Reuters RSS* news channel (http://www.microsite.reuters.com/rss/) concerns topics such as business, entertainment, technology, science, sport. Any Reuters RSS news consists of a list of items, each containing a headline, a description, and a link to access the full-text article. *PubMed* (http://www.ncbi.nlm.nih.gov/entrez/) is a service of the National Library of Medicine, and includes over 15 million citations for biomedical articles. These citations are from Medline and additional journals in the fields of biology and medicine. We composed a collection by gathering the XML documents dynamically obtained as results of the query 'protein' posed against the site search engine and accessing all available fields in PubMed.

The three collections propose different types of content and structure, as they reflect different informative services delivered by their originating sources. Reuters RSS is a good example of RSS (Really Simple Syndication) service, which is becoming popular for sharing short Web content like news headlines. Documents in Reuters RSS collection have a very regular structure, with average depth close to the maximum depth. DBLP and PubMed refer to scientific bibliographies. From the structure perspective DBLP shows more variety, although it is characterized by a small average depth and offers quite short text descriptions limited to paper titles, event topics (e.g. full name of conference), and author names. PubMed is the hardest of the three collections, as it exhibits deeply nested elements and includes elements containing long texts reporting journal abstracts. Notice that molecular biology and medicine subjects may be extremely complex: they typically contain terms shared across related species, as well as several terms used specifically for certain biological species, and new connections may be discovered among previously unrelated subjects.

Table 1: Statistics on test sets

data	*size*	*#docs*	*#trans.*	*#items*	*#terms*
DBLP	1444KB	3000	5140	6789	7329
Reuters RSS	3428KB	572	5653	7725	9251
PubMed	4247KB	1000	5331	6409	14380
Hybrid-Data	3229KB	1170	2893	6042	12935

Experimental setting and results. Experiments were performed in order to test the ability of the proposed framework in achieving the following goals:

1. *structure-driven clustering*: distinguishing structurally homogeneous classes of XML tree tuples;
2. *content-driven clustering*: identifying classes of XML tree tuples that share the same contents;
3. *structure/content-driven clustering*: this is the most general goal and encompasses many different scenarios, ranging from detecting common structures across different topics, or vice versa, to identifying classes of tree tuples that both cover common topics and belong to the same structural category – for example, DBLP tree tuples concerning works on 'computational logics' should be grouped in distinct clusters depending on whether they correspond to conference papers, journal articles, or books.

From each of the three collections previously described, we selected one test set. Moreover, we assembled a hybrid dataset by picking up documents from the *DBLP*, *Reuters RSS*, and *PubMed* datasets. Table 1 shows statistics about each test set, including information available from the tree tuple extraction phase: number of transactions (i.e. tree tuples) derived from the original documents in the collection, size of the item domain, and number of index terms. Index terms are content features for the TCUs within the extracted tree tuples, which were subject to a preprocessing phase involving lexical analysis,[2] removal of stopwords, and word stemming.

[2]We retained alphanumerical strings as they appeared in the text, whereas we removed isolated strings of digits.

Table 2: Summary of key results of clustering

data	*type*	f	γ	*#clust.*	*quality*
DBLP	content	0.1-0.2	0.6	65	0.97
DBLP	hybrid	0.5	0.6	83	0.955
DBLP	structure	0.7-0.8	0.7	5	1.0
Reuters RSS	content	0.1-0.2	0.55-0.65	144	0.968
Reuters RSS	hybrid	0.4-0.5	0.65-0.7	177	0.97
PubMed	content	0.1-0.2	0.65	11	0.91
PubMed	hybrid	0.5	0.7	10	0.895
Hybrid-Data	content	0.1-0.2	0.55	94	0.955
Hybrid-Data	hybrid	0.4	0.65	58	0.953
Hybrid-Data	structure	0.8-0.9	0.7	7	0.99

In the following we discuss the main experimental results, which are summarized in Table 2. Structure-driven clustering is obtained when factor f equals or is above 0.7, whereas content-driven clustering requires f equal to or below 0.3. Middle values of f cause a structure/content-driven clustering behavior of the framework. For the sake of brevity, Table 2 reports only the most representative result for each type of test. For instance, the first row of the table reports the best result provided by *XTrK-means* when performed content-driven clustering on *DBLP*: this result refers to a specific setting ($f \in [0.1..0.2]$, $\gamma = 0.6$); actually, very similar quality results and comparable numbers of computed clusters were achieved, for example, with γ values close to 0.6.

On the *DBLP* test set, the framework perfectly recognized the five structural classes covered by the documents in the set, that is journal articles, conference proceedings, papers in proceedings, books, and book chapters. Tests of content type offered a more variegate scenario, in which sixty-five main topics were captured to build differently sized clusters on. Looking at the representatives of computed clusters, we found out that terms appearing in the content of elements such as `title`, `booktitle`, `journal`, `series`, and sometimes `author` are particularly discriminative in raising the relevant topics. Cluster labels concerned general topics, like 'logic programming', 'Web databases', 'database transaction models', or 'nonmonotonic reasoning', but also more specific topics such as 'concurrency in software systems', 'tree automata and languages', 'evolutionary algorithms', 'object-oriented modeling of multimedia databases'. Even more interesting results were provided with "hybrid" clustering tests. In some cases, transactions sharing content features were grouped into separate clusters reflecting different structural categories. For example, conference papers and book chapters on 'multimedia database systems' were clustered in two distinct clusters. In other cases, the partitioning effect was due to the content specificity: conference papers on 'adaptive systems and interfaces' were grouped in different clusters depending on the specificity of the subtopics on 'adaptive systems and interfaces'.

Tests on *Reuters RSS* offered results relevant mainly from the content point of view. For this dataset, pure structural clustering has no much sense, since document structure conforms to one predefined template designed for a general news article. Unlike in *DBLP*, there are no tags that mark structural categories, although these are implicitly specified as titles of RSS channels. Documents in the selected test set are associated to a dozen of RSS channels (e.g. 'world', 'business', 'science', 'sports', 'politics', 'technology'). These channels can be considered as large themes; however, some themes are related and corresponding classes may overlap: for instance, 'domestic-news' and 'world-news' may be also contained in the 'top-news' channel, as well as the 'science-news' and 'health-news' channels may share some news articles. Computed clusters were finally built on subjects of topical interest ranging from 'tsunami aid' to recent 'NASA launches of space shuttle', from 'new services for mobile phones' to 'Live8 event', from 'Wimbledon 2005 tennis tournament' to 'North Korea's nuclear weapons ambitions'.

Concerning *PubMed*, we again focused on content-driven clustering. Cluster labels were generated according mainly to the content of `AbstractText`, `ArticleTitle`, and other elements related to information on cited Medline journals. Computed clusters covered topics concerning, for example, 'proteomic investigation' into 'neuroblastoma tumor' or into 'chromatographic selection', 'molecular cancer', and several subfields of 'physiology' (e.g. 'endocrinology and metabolism', 'heart and circulatory physiology', 'lung cellular and molecular physiology'). Despite the complexity of *PubMed* topics, the clustering framework still exhibited good effectiveness, as reported in Table 2.

The heterogeneity of *Hybrid-Data* set prompted us to initially perform structure-driven tests. Specifically, we expected seven classes, namely the five ones of *DBLP*, one class for *PubMed*, and one for *Reuters RSS*. Almost perfect quality results were provided by *XTrK-means*. Content-driven and hybrid tests also revealed high accuracy, following the major tendencies shown in the tests on the distinct datasets.

Besides f, the type of clustering turned out to be also functional to threshold γ. Indeed, high values of γ (e.g. 0.7) are sufficient to effectively perform structural clustering, especially when distinctions between structural categories are sharp; by contrast, more flexibility of grouping (i.e. lower values of γ) is required for cap-

turing the several semantic facets of a dataset.

A final noteworthy observation is about dealing with outliers in *XTrK-means*. Trash clusters were generated at each test, and we found out an increase of the size (number of clusters) of the resulting partition from 10% (in *PubMed* and *Reuters RSS*) to 25% (in *DBLP* and *Hybrid-Data*). Trash clustering always improved the overall partition quality, although the gained benefits may be small (2% in *DBLP*, 1% in *Hybrid-Data*) or quite irrelevant (below 1% in *PubMed* and *Reuters RSS*).

6 Conclusion and Future Work

We have presented a novel clustering framework for the semantic organization of XML data. We have investigated features for suitably representing both structural and content information from XML documents. Such features are enriched with the support of a lexical knowledge base, which plays a fundamental role in inferring XML semantics. We have exploited the notion of tree tuple to extract semantically cohesive structures from XML documents, and have shown that XML tree tuples can be easily modeled as transactions. A partitional clustering approach has been developed and applied to the XML transactional domain. Clustering evaluation has revealed very high effectiveness on large real datasets, arguing that XML tree tuple items are powerful features for effective semantic XML clustering.

There are some evident directions for future research. Some of these regard the consolidation of certain aspects of the framework, such as the development of a possibly novel clustering algorithm which is able to best fit the XML transactional model while satisfying at least the requirements of scalability, cluster discovery in subspaces, and browsing-aware cluster labeling. Moreover, the role of ontological knowledge in supporting the detection of semantic relatedness among XML data needs better investigation. Therefore, we shall look at incorporating application ontologies into our clustering framework to benefit from extended conceptual models describing not only objects represented in XML, and their relationships and constraints, but also "rules" regarding how objects may appear in an XML source.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: from relations to semistructured data and XML*. Morgan Kaufmann Publishers, 1999.

[2] M. Arenas and L. Libkin. A Normal Form for XML Documents. *ACM Trans. Database Systems*, 29(1):195–232, 2004.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books. Addison-Wesley, 1999.

[4] S. Banerjee and T. Pedersen. Extended Gloss Overlaps as a Measure of Semantic Relatedness. In *Proc. IJCAI*, pages 805–810, 2003.

[5] G. Costa, G. Manco, R. Ortale, and A. Tagarelli. A Tree-based Approach to Clustering XML Documents by Structure. In *Proc. PKDD*, pages 137–148, 2004.

[6] A. Doucet and H. A. Myka. Naive Clustering of a Large XML Document Collection. In *Proc. INEX Annual ERCIM Workshop*, pages 81–88, 2002.

[7] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[8] S. Flesca, F. Furfaro, S. Greco, and E. Zumpano. Repairs and Consistent Answers for XML Data with Functional Dependencies. In *Proc. Int. XML Database Symposium (XSym)*, pages 238–253, 2003.

[9] F. Giannotti, C. Gozzi, and G. Manco. Clustering Transactional Data. In *Proc. PKDD*, pages 175–187, 2002.

[10] M. Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to tell a pine cone from a ice cream cone. In *Proc. ACM SIGDOC Int. Conf. on Systems Documentation*, pages 24–26, 1986.

[11] W. Lian, D. W. Cheung, N. Mamoulis, and S.-M. Yiu. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. *IEEE Trans. Knowledge Data Engineering*, 16(1):82–96, 2004.

[12] A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *Proc. ACM SIGMOD WebDB Workshop*, pages 61–66, 2002.

[13] N. Polyzotis and M. Garofalakis. Structure and Value Synopses for XML Data Graphs. In *Proc. VLDB*, pages 466–477, 2002.

[14] P. Resnik. Semantic Similarity in a Taxonomy: An Information-based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[15] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, and R. Busse. The XML Benchmark Project. Technical report, INS-R0103, CWI, Amsterdam, The Netherlands, 2001.

[16] M. Theobald, R. Schenkel, and G. Weikum. Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. In *Proc. ACM SIGMOD WebDB Workshop*, pages 1–6, 2003.

[17] J. Widom. Data Management for XML: Research Directions. *IEEE Data Engineering Bulletin*, 22(3):44–52, 1999.

[18] J. P. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg. BitCube: A Three-Dimensional Bitmap Indexing for XML Documents. *Journal of Intelligent Information Systems*, 17(1):241–252, 2001.

[19] M. J. Zaki and C. C. Aggarwal. XRules: An Effective Structural Classifier for XML Data. In *Proc. ACM SIGKDD*, pages 316–325, 2003.