



**Queensland University of Technology**  
Brisbane Australia

This may be the author's version of a work that was submitted/accepted for publication in the following source:

[Nayak, Richi & Iryadi, Wina](#)

(2007)

XML Schema Clustering with Semantic and Hierarchical Similarity Measures.

*Knowledge-Based Systems*, 20(4), pp. 336-349.

This file was downloaded from: <https://eprints.qut.edu.au/13994/>

**© Consult author(s) regarding copyright matters**

This work is covered by copyright. Unless the document is being made available under a Creative Commons Licence, you must assume that re-use is limited to personal use and that permission from the copyright owner must be obtained for all other uses. If the document is available under a Creative Commons License (or other specified license) then refer to the Licence for details of permitted re-use. It is a condition of access that users recognise and abide by the legal requirements associated with these rights. If you believe that this work infringes copyright please provide details by email to [qut.copyright@qut.edu.au](mailto:qut.copyright@qut.edu.au)

**Notice:** *Please note that this document may not be the Version of Record (i.e. published version) of the work. Author manuscript versions (as Submitted for peer review or as Accepted for publication after peer review) can be identified by an absence of publisher branding and/or typeset appearance. If there is any doubt, please refer to the published source.*

<https://doi.org/10.1016/j.knosys.2006.08.006>

# XML schema clustering with semantic and hierarchical similarity measures

Richi Nayak and Wina Iryadi

School of Information Systems, Queensland University of Technology  
Brisbane, Australia [r.nayak@qut.edu.au](mailto:r.nayak@qut.edu.au)

**Abstract.** With the growing popularity of XML as the data representation language, collections of XML data have exploded in numbers. The methods are required to manage and discover the useful information from them for improved document handling. We present a schema clustering process by organising heterogeneous XML schemas into groups. The methodology considers not only the linguistic and the context of the elements but also the hierarchical structure similarity. We support our findings with experiments and analysis.

**Keywords:** Clustering; Data mining; Document mining; XML; semi-structured data; semantic similarity; structural similarity; schema matching

## 1. Introduction

XML has become a standard for information exchange and retrieval [34]. With the continuous growth in XML data, the ability to manage massive collections of XML data and to discover knowledge from them becomes essential for Web-based information systems [15, 25]. A possible solution is to group similar XML data based on their context and structure. The clustering of XML data facilitates a number of advanced applications such as improved information retrieval, data and schema integration, document classification analysis, structure summary and indexing, and query processing and optimization [6, 23].

The clustering data mining process categorizes the XML data based on their similarity without having a prior knowledge on the taxonomy. There exist a number of clustering methods dealing with (unstructured) database objects and text data [3, 36]. The XML data is different – semistructured and hierarchical [34]. There are two types of XML data: XML documents and XML schemas. A XML schema describes the structure of the XML document. Usually, XML's schema can be obtained separately without scanning the whole document. Therefore, a method to cluster XML documents should take advantage of their schema.

Similarity of correspondence elements between XML documents can be conducted efficiently using relevant XML schemas. The document schema provides a definitive description of the document, while document instances only give a snapshot what the document may contain. The document definition outlined in a schema holds true for all document instances of that schema. So the result produced from clustering of schemas will hold true for all document instances of those schemas, and can be reused for any other instances. On the contrary, the result of clustering of document

instances will hold true for included document instances only. The clustering process is to be repeated for any other document instances.

This paper presents the XMine methodology that quantitatively determines the similarity between heterogeneous XML schemas by considering the semantic as well as the hierarchical structure similarity of elements. The similar schemas are clustered into separate meaningful classes. Whilst there are several XML documents and schema clustering techniques available [4, 6, 9, 11, 24, 26], this paper enhances this task by adding hierarchical similarity in clustering by addressing the element level hierarchical positions. The XMine methodology can deal with varying structure of schemas and with varying aspects of semantic differences in schema elements.

The contributions of this paper are (1) combining the semantic and syntactic relationships to calculate the linguistic similarity between two element names; (2) calculating the structural similarity between two elements by considering the ancestor-child relationship along with parent-child relationship in maximal similar paths; and then (3) generalizing a suitable schema class hierarchy to determine the relationship between the discovered schemas in the XMine methodology.

The performance of XMine is demonstrated using a number of heterogeneous schemas derived from several application domains. The empirical results demonstrate that the semantic, syntactic and hierarchical relationships of schema elements play important roles for producing good quality of clustering results. Most importantly, it discovers that syntactic similarity measure is more useful than semantic similarity measure.

### **1.1 Potential Applications of the XMine methodology**

The result of schema class composition hierarchy can serve as a basis for a number of XML application processes. The clusters of schemas provide a hint for building an index structure. Indexing based on structural similarity support many applications. For example in information retrieval field, the XML-based search engines can improve the speed and accuracy in retrieving the relevant portions of XML data by using efficient indexes. Moreover, several databases tools that are developed to deliver, store, integrate and query XML data [5, 12, 21, 33], require indexing based on structural similarity to support an effective document storage and retrieval

Moreover, the schema class composition hierarchy can be viewed as a generalization of the training sets of schemas to a super-class that is useful for further XML document classification analysis. A number of heterogeneous sources of schemas can be classified into this set of predefined classifications of schemas. This process will improve the XML document handling and achieve more effective and efficient searches of relevant XML documents.

The method of association rule mining can also be applied to find interesting correlation relationships of all metadata available in schemas belonging to the same schema class. The element tags that frequently occur together within a schema class can be used to maximally distinguish one class of schema from others. This would derive a set of association rules associated with each schema class. This schema element tag-based association analysis is also useful for discovering common XML structures for a specific domain.

In addition, the schema class hierarchies can also facilitate a difficult task of schema integration process on heterogeneous schemas. The integration on similar schemas within each schema class would provide an easier task than reconciling schemas that are different in structure and semantics, which would involve complex restructuring process.

The similarity between two structures is also a notion tied to a challenging task of reusing XML or semi-structured documents. In XML document content reuse, a document (or a part of document) structured under one schema must be restructured into an instance of a different schema. The identification of common paths between two instances of schema helps to avail this restructuring.

<pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;Companies&gt;   &lt;Company&gt;     &lt;Symbol&gt; Eagle.img &lt;/Title&gt;     &lt;Name&gt; EagleFarm &lt;/Name&gt;     &lt;Industry&gt; Dairy &lt;/Industry&gt;     &lt;Profile&gt;       &lt;MarketCap&gt; 1000 &lt;/ MarketCap &gt;       &lt;EmployeeNo&gt; 20 &lt;/ EmployeeNo &gt;       &lt;Address&gt;         &lt;State&gt; QLD &lt;/State&gt;       &lt;/Address&gt;       &lt;Description&gt; gdsfkl &lt;/Description&gt;     &lt;/Profile&gt;   &lt;/Company&gt;   &lt;!-- Some more instances --&gt;   .... &lt;/Companies&gt; </pre>	<pre> &lt;!DOCTYPE Companies [   &lt;!ELEMENT Companies (Company+)&gt;   &lt;!ELEMENT Company (Symbol, Name,     Sector?, Industry, (Profile))&gt;   &lt;!ELEMENT Profile (MarketCap,     EmployeeNo, (Address),     Description)&gt;   &lt;!ELEMENT Address (State,City?)&gt;   &lt;!ELEMENT Symbol(#PCDATA)&gt;   &lt;!ELEMENT Name (#PCDATA)&gt;   &lt;!ELEMENT Sector (#PCDATA)&gt;   &lt;!ELEMENT Industry (#PCDATA)&gt;   &lt;!ELEMENT MarketCap (#PCDATA)&gt;   &lt;!ELEMENT EmployeeNo (#PCDATA)&gt;   &lt;!ELEMENT State (#PCDATA)&gt;   &lt;!ELEMENT City (#PCDATA)&gt; ]&gt; </pre>
--	--

**Figure 1:** Example of a XML document and its respective DTD

## 2. Background Knowledge on XML Data

XML is a flexible representation language. There are two varieties of XML data: XML documents and XML schemas. A XML schema provides the data definitions and structure of the XML document [1]. While XML documents are the instances of a schema giving a snapshot of what the document may contain. A schema includes what elements are (not) allowed; what attributes for any elements may be and the number of occurrences of elements; etc. A schema for a document may be included as both internally and externally (located within the same file or a different file, respectively).

There are several XML schema languages, but only two are commonly used. They are DTD (Document Type Definition) and XML Schema or XML Schema Definition (XSD), both of which allow the structure of XML documents to be described and their contents to be constrained [32]. A DTD specifies the structure of an XML element by specifying the names of its sub-elements and attributes. Sub-element structure is specified using operators \* (zero or more elements), + (one or

more elements), ? (optional), and | (or), as well as with properties type (PCDATA, ID, IDREF, ENUMERATION).

The DTD language is considered limited as it only supports limited set of data types, loose structure constraints, limitation of content to textual, etc. To overcome the above limitations of DTD, XSD provides novel important features, such as simple and complex types, rich datatype sets, occurrence constraints and inheritance. An XML Schema is usually comprised of a set of schema components, such as type definitions and element declarations. They can be used to assess the validity of well-formed element information items.

It is believed that XSD will soon take over DTD due to its flexibility [13]. Therefore, the XMine methodology clusters the XML schemas represented in both schema languages. Throughout this paper, we use the term 'schema' to express both XML-DTD and XML-Schema unless clearly specified.

Figure 1 illustrates a simple example of XML document and its corresponding DTD. Figure 2 shows a respective XML Schema.

```
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>
  <xsd:element name="Companies" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Company" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Symbol" type="xsd:string"/>
              <xsd:element name="Name" type="xsd:string"/>
              <xsd:element name="Sector" type="xsd:string"/>
              <xsd:element name="Industry" type="xsd:string"/>
              <xsd:element name="Profile" >
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="MarketCap" type="xsd:string"/>
                    <xsd:element name="EmployeeNumber" type="xsd:unsignedInt"/>
                    <xsd:element name="Address" >
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="State" type="xsd:string"/>
                          <xsd:element name="City" type="xsd:string"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="Description" type="xsd:string"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </element>
</xsd:schema>
```

**Figure 2:** Example of the respective XSD of the above document

### 3. The XMine Methodology

Figure 3 illustrates the overall architecture of the XMine methodology. This is deployed in three phases, namely *preprocessing*, *data mining*, and *postprocessing*.

The focus of the preprocessing phase is to determine the common and similar features between various schemas in automated manner to effectively facilitate the clustering process. It includes four stages to address various issues involved in measuring the similarity of schemas. Firstly, the *structure analyser* analyses the structure of a schema and transforms it into a *labelled and directed acyclic tree graph*. The *element analyser* then measures the similarity between the arbitrary elements in different schemas primarily based on the element names. Next, the *maximally similar paths finder* determines the common and similar hierarchical structure of the elements defined in schema by using the adapted *sequential pattern mining algorithm*. Lastly, the overall degree of similarity between schemas is computed by taking the element and structure similarity into consideration.

The XMine methodology then proceeds for data mining. Schemas similar in structure and semantics are grouped together to form a hierarchy of schema classes using an agglomerative clustering algorithm. The clustering result is visualized in the final phase of the methodology. The visualization is also a critical verification of the clustering results, which assist the generalization and specialization on the schema classes to develop a schema class hierarchy.

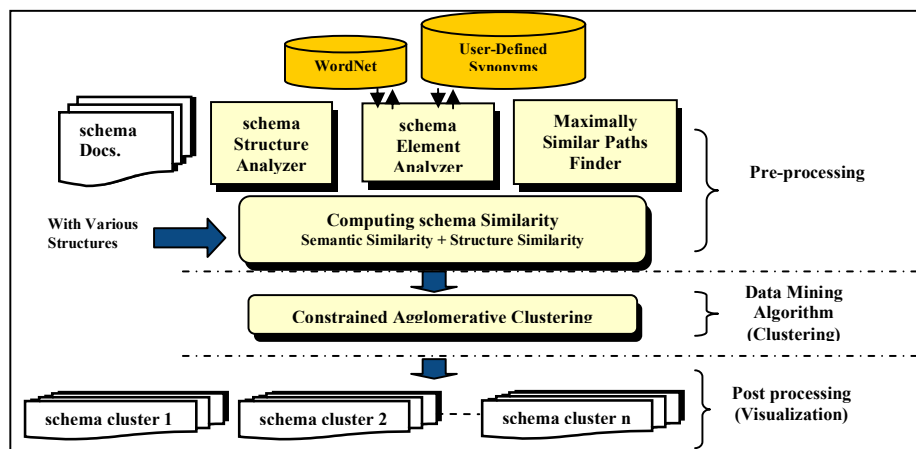


Figure 3: The architecture of XMine methodology

#### 3.1 Preprocessing: Structure Analyser

This module represents a schema into a labelled and ordered tree. This module also performs simplification analysis of the schema trees in order to deal with nesting and repetition problems. XMine handles both the common types of XML schemas: DTD (document type definition) and XSD (XML Schema definition). A schema is composed of hierarchical elements, wherein for each element it is possible to specify

whether: it is optional ('?'); it occurs several times ((maxOccurs="unbounded") in XSD or ('+') or ('\*') in DTD); subelements are alternatives with respect to each other (('xsd:choice') in XSD or ('|') in DTD); or subelements are grouped in a sequence (('xsd:sequence') in XSD or ('.') in DTD).

The constraint features of a schema serve as the primary elements for the construction of the tree representation. Each node in the tree contains its properties such as name, data type and cardinality. In addition, each node in the tree corresponds to an element or an attribute, or to an element operator with edges denoting the nested relationship between element and its subelement or operator. Moreover, there can be more than one edge outgoing from a node, only if the edge incoming to that node is labeled by AND or OR operator. The elements that have basic property types of #PCDATA or ANY in a DTD, or 'type' in a XSD are considered as leaves of the tree (e.g. fName, mName, lName). Attributes are treated as special elements that have an atomic property.

According to [18], it is difficult to determine the degree of similarity of two elements that have AND-OR operators in their content representation. Therefore these details of a schema are normalized into a simplified schema according to a series of predefined transformation procedures similar to those in [18].

An example of representing DTDs as a tree form is shown in Figure 5.

### 3.2 Preprocessing: Element Analyser

This module addresses the issue that schemas from same domains may have naming differences, and they may model non-identical but similar content. The element analyser measures the elements (tag names) similarity (*linguistic similarity coefficient: lSim*) by comparing each pair of elements of two schemas primarily based on their names, assuming the same names bear the same semantic meaning.

It considers the equality of canonical name representations after stemming and element preprocessing. This is important to deal with special prefix or suffix symbols (e.g. CName → customer name, EmpNo → employee number). In addition, the element names in different schemas might not be exactly the same, provided they are stems or similar enough. Hence, the other consideration is the equality of synonyms between elements (e.g. car → automobile, movie → film) and similarity of elements based on common string edit distance operation (e.g. chtitle → title). We use of WordNet thesaurus [10] to exploit synonyms (e.g., movie → film) and the user-defined dictionaries in order to identify abbreviations (e.g. Emp → Employee), acronyms (e.g. DOB → Date of Birth), and user-defined synonyms.

The steps to measure the *linguistic similarity coefficient (lSim)* are as follows:

1. Parse the compound element name into a set of tokens based on customizable delimiters such as, uppercase, punctuation, and special symbols, e.g., PONumber → {PO, Number}.
2. Expand the tokens into a linguistic set (*lingSet*) using the user-defined dictionary with acronyms and abbreviations, e.g., {PO, Number} → {Purchase, Order, Number}.

$T = \text{Set of the tokens} = \text{lingSet}(w)$  where  $w$  is an element name.

3. Measure the *lSim* of two sets of name tokens *T1* and *T2* to find how linguistically close two element names are. It is the average of the best similarity of each token with a token in the other set. It is calculated as follows:

$$lSim(t_1, t_2) = \frac{\sum_{t_1 \in T_1} \int_{t_2 \in T_2}^{\max} sim(t_1, t_2) + \sum_{t_2 \in T_2} \int_{t_1 \in T_1}^{\max} sim(t_2, t_1)}{|T_1| + |T_2|}$$

*sim* (*t*<sub>1</sub>, *t*<sub>2</sub>) is a combined measure (as formulated in figure 4) that calculates the semantic relationship (e.g. movie → film) as found in *WordNet thesaurus* [10] and the syntactic relationship (e.g. ctitle → title) using the *string edit distance function* [27]. The semantic relationship is first applied for exploiting the semantic similarity degree between two tokens by looking up in the WordNet. If the WordNet does not identify common elements, the syntactic relationship is then applied. Similarity thresholds ( $\delta$  and  $\mu$ ) are set to represent the minimal degree of similarity required for semantic and syntactic measures respectively.

```

Function sim (t1, t2)
  sim = SemanticSim (t1, {t2}, 1); /* Semantic Relationship with the WordNet*/
  if sim ≥  $\delta$  then return sim;
  else /* Syntactic Relationship */
    sim =  $\frac{edit\_distance(t_1, t_2)}{\max(length(t_1), length(t_2))}$ 
    if sim ≥  $\mu$  then return sim;
  return 0; /* No match */

```

**Figure 4: Algorithm to compute Linguistic Similarity of two words**

Following is an **example** showing the calculation of *lSim*: consider two elements *w1-author\_fname* and *w2-writerName*. Tokens are derived: *T1*- {*author*, *fname*} and *T2*- {*writer*, *name*}. Similarity between each pair is measured:

1. *sim* (*author*, *name*) = 1 (using the semantic similarity measure)
2. *sim* (*name*, *author*) = 1 (using the semantic similarity measure)
3. *sim* (*fname*, *name*) = 0.8 (using the string edit function due to the semantic similarity less than  $\delta$  - Assuming  $\delta$  is set as 0.7.)
4. *sim* (*name*, *fname*) = 0.8 (using the string edit function due to the semantic similarity less than  $\delta$ )

$$\text{Linguistic Similarity Coefficient (lSim): } \frac{(1 + 0.8) + (1 + 0.8)}{2 + 2} = 0.9$$

### 3.3 Preprocessing: Maximally Similar Paths Finder

This module identifies the paths and elements that are common and similar between each pair of tree schemas. The assumption is that similar schemas have more common paths. We adapt the sequential pattern mining algorithm [2] to infer similarity between elements and paths. The sequential pattern mining algorithm considers the frequent occurrences of elements as well as the sequences of elements.



The structure of a schema tree is represented by a set of path expressions (or paths). Each path expression is viewed as a sequence. A path expression is represented by a unique sequence of elements following the links from the root node to a leaf node by traversing through the nodes in that path. A path expression,  $p$ , is denoted as  $\langle x_1, x_2, \dots, x_n \rangle$  where  $x_1$  is a name of the root node and  $x_n$  is a name of the leaf node. Let the set of path expressions,  $PE$ , in a schema tree be  $\{p_1, p_2 \dots p_m\}$  where  $m$  is the number of unique paths in the tree. Using the terminology of sequential pattern mining, a sequence (or a path) is contained by another if it is a subsequence of that sequence. A sequence (or a path) is frequent if it occurs in the set more times than the user defined threshold (or support). In a set of paths, a path  $p_j$  is *maximal* if it is not contained by any other path expression or no super path of  $p_j$  is frequent.

The task is to find the maximal frequent paths among the set of path expressions in two schema trees. Each such maximal frequent path represents a common structure between the pairs of trees. Unlike other data mining applications, the minimum support for finding the maximal frequent paths between two trees must be 100% since similar paths must be in both structures. Another variation in this process is that support count for an element should be incremented only one per schema even if the schema contains the same elements in two different paths.

The five phases of the sequential mining algorithm [2] are modified to facilitate the finding of maximal similar paths (MPEs) between two trees (a base tree  $T_B$  and a query tree  $T_Q$ ):

1. *Sort Phase*. Elements contained in each path are sorted according to their hierarchical position in the tree levels. The first element appearing in a path always represents the root node of the corresponding schema tree. The remainder of the elements in the path are then denoted as the descendent of the root node in order.
2. *Transformation Phase*. The elements of path expressions are mapped into integer representation to facilitate faster sequential mining process. Elements in the path expressions defined as *similar* according to the linguistic similarity coefficient ( $lSim$ ) is mapped into the same integer representation.
3. *Litemset Phase*. In this phase, the set of *large 1-paths* are found by considering the element matching. Every similar element in the two path expressions is included in the *large 1-paths* set. The *large 1-path* is a set of all expressions that have only one element and that is frequent.
4. *Sequential Phase*. This includes the multiple passes over a collection of large paths sets in order to determine new larger paths progressively such as the large 2-paths, large 3-paths and so on, until large  $n$ -paths are found.
5. *Maximal Phase*. The maximal similar paths (MPE) are found by using the *backward* phase [2] to all the large paths obtained in the sequential phase. All sub-paths contained in large paths are pruned out until maximal paths are found.

### 3.4 Preprocessing: Schema Similarity Matrix Processor

A method to compute the similarity between schemas is presented by making use of: (1) the element semantic similarity as explained in section 3.2; and (2) the element structural similarity obtained as the maximal large paths in section 3.3. The element structural similarity includes the hierarchical position of an element in the schema.

This covers the context of an element defined by its ancestor (if it is not a root) and its descendants positioned as in path expressions. This is included in XMine by determining similar elements in two trees based on the common paths. This serves the basis of structural computation. The element semantic similarity includes the linguistic and constraint similarity between each pair of elements contained in two maximal similar paths. The overall degree of similarity based on the element and structure similarity is then computed in the schema similarity matrix processor.

Let us assume two schemas: base schema (schema<sub>B</sub>) and query schema (schema<sub>Q</sub>) that are to be compared. Base tree  $T_B$  and query tree  $T_Q$  are the corresponding simplified trees. A unique set of path expressions are obtained by traversing both the base and query trees, denoted as  $PE^B$  and  $PE^Q$  respectively. A set of maximal similar path expressions ( $MPE$ ) represents a number of common paths that exist in both base and query tree. The corresponding path expressions that contain a  $MPE$  from the  $PE^B$  and  $PE^Q$  sets are identified.

**Structural similarity:** Once all the corresponding common path expressions from both trees have been obtained, the similarity coefficient of all maximal similar paths,  $maxpathSim$ , is measured. The  $maxpathSim$  aggregates the similarity coefficient of two corresponding base and query path expressions, refers to as path similarity coefficient,  $pathsim$ . The following is the formalization of  $maxpathSim$ :

$$maxpathSim(MPE_k) = \frac{\sum_{PE_i^B \in MPE_k} \sum_{PE_j^Q \in MPE_k} pathSim(PE_i^B, PE_j^Q, Threshold)}{Max_{PE^B \in MPE_k, PE^Q \in MPE_k} (|PE^B|, |PE^Q|)}$$

Similarity between two path expressions ( $pathSim$ ) is computed by measuring the linguistic, constraints, and path name similarity of each element of  $PE_i^B$  against elements of  $PE_j^Q$ . This checks a one-to-one mapping of elements in the path expressions, that is an element in  $PE_i^B$  matches, at most, one element in  $PE_j^Q$ .

$$pathSim(PE_i^B, PE_j^Q, Threshold) = \frac{\sum_{b=1}^{|PE_i^B|} \sum_{q=1}^{|PE_j^Q|} baseSim(e_b, e_q) * PNC(e_b, PE_i^B, e_q, PE_j^Q, Threshold)}{Max(|PE_i^B|, |PE_j^Q|)}$$

where the base element similarity coefficient,  $baseSim$ , represents the semantic similarity between two names. The path name coefficient,  $PNC$ , measures the degree of similarity of elements in two given paths.

**Semantic similarity:** The base element similarity coefficient,  $baseSim$ , is obtained by the weighted sum of linguistic similarity coefficient,  $ISim$  and the constraint similarity coefficient,  $constraintSim$  of the elements, shown as below:

$$baseSim(e_1, e_2) = w_1 * lSim(e_1, e_2) + w_2 * constraintSim(e_1, e_2)$$

where weights  $w_1 + w_2 = 1$ .

The linguistic similarity coefficient, ***lSim*** is defined in section 3.2. The cardinality constraint coefficient, ***constraintSim*** of two elements is determined from the cardinality constraint compatible table (Table 1) as used in [18] for DTD. Table 1 shows the compatibility between two operators. XSD schema is more flexible than DTD in terms of cardinality operations by using minOccurs and maxOccurs. We show the mapping between the cardinality operators of DTD and XSD in Table 2 and utilise the values of Table 1 for each equivalent mapping. For the operators outside this list, if their data types are identical then 1 is returned or else 0 is returned. The constraint coefficient is ranged between [0, 1].

	*	+	?	None
*	1	0.9	0.7	0.7
+	0.9	1	0.7	0.7
?	0.7	0.7	1	0.8
None	0.7	0.7	0.8	1

**Table 1: Cardinality constraint compatibility table adapted from [18]**

Cardinality Operator	minOccurs	maxOccurs	No. of child element(s)
[none]	1	1	One and only one
?	0	1	Zero or one
*	0	Unbounded	Zero or more
+	1	Unbounded	One or more

**Table 2: Cardinality Mapping between XSD and DTD**

**Path similarity coefficient:** The path name coefficient, ***PNC***, measures the degree of similarity of elements in two given paths. The goal of this computation is to differentiate elements that are present in both paths but are different in their context (e.g., a patient's name and a physician name). Consider two common paths that have two elements with the same name but appearing in different level position (e.g., *book.name* and *book.author.name*). The context in which an element appears in the hierarchical structure of a schema strongly contributes to determine the information that element models [25]. The context of an element  $e$  is given by the path from *root* element (that is the first element in the path expression) to the element  $e$ , denoted as  $e.path(root) = \{root, e_{pi}, \dots, e_{pj}, e\}$ .

The similarity of two path names is obtained by summing up all the *baseSim* values between each pair of elements in two paths then normalizing it with the maximum number of elements contained in the two paths of the element names.

$$PNC = \frac{\sum baseSim}{Max(|dest_1.path(source_1)|, |dest_2.path(source_2)|)}$$

**Schema similarity:** Having obtained the similarity between all the maximal similar paths (MPEs) of two trees, the similarity between two schemas is computed by combining all *MPE similarity coefficients*:

$$schemaSim(schema_B, schema_Q) = \frac{\sum_{k=1}^{|MPE|} MaxpathSim(MPE_k)}{\max(|PE^B|, |PE^Q|)}$$

The similarity between each pair of schemas is mapped into the *schema similarity matrix*. This matrix becomes the input to the next phase.

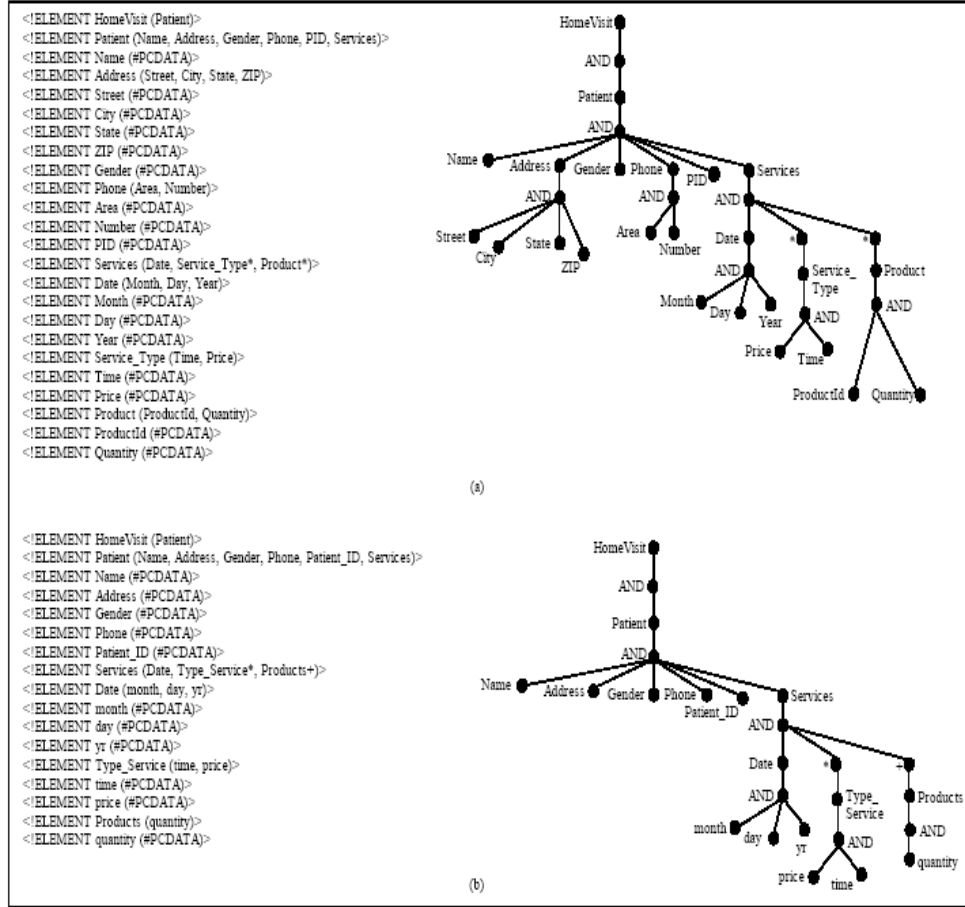
### 3.5 An example showing the process of preprocessing

Figure 5 shows two schemas and their respective tree representation related to the health care system. Let us consider one of them (a) as ‘base’ and another one (b) as ‘query’. The objective is to find the similarity between them. A unique set of path expressions are obtained by traversing both the base and query trees denoted as  $PE_B$  and  $PE_Q$  respectively. Every element contained in the path expression is sorted according to their hierarchical position in the tree.

Similar elements of paths are mapped into same integer representation by referencing the linguistic similarity table. For instance, the abbreviation of  $PID \leftrightarrow Patient\ ID$  is defined similar in the user defined library, so both elements are assigned to the same integer in any path expression. In addition, the use of WordNet thesaurus is able to resolve the abbreviations such as  $yr \leftrightarrow year$  and  $Qty \leftrightarrow Quantity$ . The tokenizer during the element pre-processing is able to recognize the similarity between the element names  $Service\_Type \leftrightarrow Type\_Service$ .

Table 3 shows the  $PEs$  of both trees. The maximal similar path expressions (MPEs) are determined from these  $PEs$  according to the process described in section 3.3. In the first iteration of the adopted apriori-algorithm, each distinct element in both sets of path expressions is a member of the set of the candidate 1-paths,  $C_1$ . The algorithm simply scans the elements that are similar in both sets of path expressions. The set of large 1-paths,  $L_1$ , is then determined. It consists of the candidates 1-paths that exist in both  $PE_B$  and  $PE_Q$ . To discover the set of large 2-paths,  $L_2$ , the algorithm uses joining  $L_1 \times L_1$  to generate a candidate set of 2-paths,  $C_2$ . Then the algorithm scans  $C_2$  to obtain the 2-large-paths that are contained common in  $PE_B$  and  $PE_Q$ . The algorithm iterates this process until it finds all the large paths.

In our example, the algorithm terminates in the sixth pass. Table 4 shows some of the Large Paths. The *backward phase* is now used to find the maximal similar paths among the set of large paths. Starting from  $L_5$ , no paths are deleted since there are no path sequences contained in some other large paths. Then moved on to  $L_4$ , delete those paths that are subsequences of the paths in  $L_5$  and thus all the 4-large paths are pruned out. Next, the paths in  $L_3$  that are subsequence of the 5-large paths are pruned out. The 5-large paths in  $L_3$  are found to be maximal. They are the first five rows shown in Table 5. Finally all the paths in  $L_2$  and  $L_1$  are pruned out since they are contained in the larger paths. Table 5 lists all the MPEs for these two schemas with the corresponding  $PE_B$  and  $PE_Q$  that contain them.



**Figure 5:** Base (a) and query (b) documents with their corresponding trees.

PE	ID	Original path expressions	Transformed path expressions
PE <sub>B</sub>	1	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Name}\rangle\rangle$	$\langle\{1\}\{2\}\{3\}\rangle$
	7	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Phone}\rangle\langle\text{Area}\rangle\rangle$	$\langle\{1\}\{2\}\{10\}\{11\}\rangle$
	8	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Phone}\rangle\langle\text{Number}\rangle\rangle$	$\langle\{1\}\{2\}\{10\}\{12\}\rangle$
	16	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Services}\rangle\langle\text{Product}\rangle\langle\text{Quantity}\rangle\rangle$	$\langle\{1\}\{2\}\{14\}\{22\}\{24\}\rangle$
PE <sub>Q</sub>	1	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Name}\rangle\rangle$	$\langle\{1\}\{2\}\{3\}\rangle$
	4	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Phone}\rangle\rangle$	$\langle\{1\}\{2\}\{10\}\rangle$
	11	$\langle\langle\text{HomeVisit}\rangle\langle\text{Patient}\rangle\langle\text{Services}\rangle\langle\text{Product}\rangle\langle\text{Quantity}\rangle\rangle$	$\langle\{1\}\{2\}\{14\}\{22\}\{24\}\rangle$

**Table 3:** Equivalent transformed path expressions for both trees.

Large 1-path (L1)	Large 2-path (L2)	Large 5-path (L5)
Total elements: 17	Total elements: 46	Total elements: 6
All: $\langle\{1\}\rangle, \langle\{2\}\rangle, \langle\{3\}\rangle, \langle\{4\}\rangle,$	Sample: $\langle\{1\}\{2\}\rangle, \langle\{1\}\{3\}\rangle,$	All:
$\langle\{9\}\rangle, \langle\{10\}\rangle, \langle\{13\}\rangle, \langle\{14\}\rangle,$	$\langle\{1\}\{4\}\rangle, \langle\{1\}\{9\}\rangle, \langle\{1\}\{10\}\rangle,$	$\langle\{1\}\{2\}\{14\}\{15\}\{16\}\rangle, \langle\{1\}\{2\}\{14\}\{15\}\{17\}\rangle,$
$\langle\{15\}\rangle, \langle\{16\}\rangle, \langle\{17\}\rangle, \langle\{18\}\rangle,$	$\langle\{1\}\{13\}\rangle, \langle\{1\}\{14\}\rangle, \langle\{1\}\{15\}\rangle,$	$\langle\{1\}\{2\}\{14\}\{15\}\{18\}\rangle, \langle\{1\}\{2\}\{14\}\{19\}\{20\}\rangle,$
$\langle\{19\}\rangle, \langle\{20\}\rangle, \langle\{21\}\rangle, \langle\{22\}\rangle,$	$\langle\{1\}\{16\}\rangle, \langle\{1\}\{17\}\rangle, \langle\{1\}\{18\}\rangle,$	$\langle\{1\}\{2\}\{14\}\{19\}\{21\}\rangle, \langle\{1\}\{2\}\{14\}\{22\}\{24\}\rangle,$
$\langle\{24\}\rangle.$	$\langle\{1\}\{19\}\rangle, \dots\dots\dots$	

**Table 4:** Large paths in base and query documents

<i>MPE</i>	Corresponding <i>PE<sub>B</sub></i> and <i>PE<sub>Q</sub></i>
<i>MPE<sub>1</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Name})\rangle$	<i>PE<sub>B 1</sub></i> , <i>PE<sub>Q1</sub></i>
<i>MPE<sub>2</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Address})\rangle$	<i>PE<sub>B 2</sub></i> , <i>PE<sub>B 3</sub></i> , <i>PE<sub>B 4</sub></i> , <i>PE<sub>B 5</sub></i> , <i>PE<sub>Q2</sub></i>
<i>MPE<sub>3</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Gender})\rangle$	<i>PE<sub>B 6</sub></i> , <i>PE<sub>Q3</sub></i>
<i>MPE<sub>4</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Phone})\rangle$	<i>PE<sub>B 7</sub></i> , <i>PE<sub>B 8</sub></i> <i>PE<sub>Q4</sub></i>
<i>MPE<sub>5</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{PID})\rangle$	<i>PE<sub>B 9</sub></i> , <i>PE<sub>Q5</sub></i>
<i>MPE<sub>6</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Services}) (\text{Date}) (\text{Month})\rangle$	<i>PE<sub>B 10</sub></i> , <i>PE<sub>Q6</sub></i>
<i>MPE<sub>7</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Services}) (\text{Date}) (\text{Day})\rangle$	<i>PE<sub>B 11</sub></i> , <i>PE<sub>Q7</sub></i>
<i>MPE<sub>8</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Services}) (\text{Date}) (\text{Year})\rangle$	<i>PE<sub>B 12</sub></i> , <i>PE<sub>Q8</sub></i>
<i>MPE<sub>9</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Services}) (\text{Service\_Type}) (\text{Price})\rangle$	<i>PE<sub>B 13</sub></i> , <i>PE<sub>Q9</sub></i>
<i>MPE<sub>10</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Services}) (\text{Service\_Type}) (\text{Time})\rangle$	<i>PE<sub>B 14</sub></i> , <i>PE<sub>Q10</sub></i>
<i>MPE<sub>11</sub></i> $\langle(\text{HomeVisit}) (\text{Patient}) (\text{Services}) (\text{Product}) (\text{Quantity})\rangle$	<i>PE<sub>B 16</sub></i> , <i>PE<sub>Q11</sub></i>

**Table 5:** Corresponding base and query path expressions, *PE<sub>B</sub>* and *PE<sub>Q</sub>*, for each *MPE*.

For each *MPE*, the path similarity coefficient between each pair of base and query paths is computed by measuring the *baseSim* and *PNC* of all pairs of elements in both paths. Let us compute the maximal similarity path coefficient of

*MPE<sub>4</sub>* =  $\langle (\text{HomeVisit}) (\text{Patient}) (\text{Phone}) \rangle$  that consists of:

*PE<sub>B7</sub>* =  $\langle (\text{HomeVisit}) (\text{Patient}) (\text{Phone}) (\text{Area}) \rangle$

*PE<sub>B8</sub>* =  $\langle (\text{HomeVisit}) (\text{Patient}) (\text{Phone}) (\text{Number}) \rangle$

*PE<sub>Q4</sub>* =  $\langle (\text{HomeVisit}) (\text{Patient}) (\text{Phone}) \rangle$

Here, the pairs of element names with no semantic similarity are not shown. We have:

$$pathSim(PE_{B7}, PE_{Q4}, 0.3) = \frac{(1.0) + (1.0) + (1.0)}{Max(4,3)} = 0.75$$

$$pathSim(PE_{B8}, PE_{Q4}, 0.3) = \frac{(1.0) + (1.0) + (1.0)}{Max(4,3)} = 0.75$$

$$maxpathSim(MPE_4) = \frac{(0.75) + (0.75)}{Max(2,1)} = 0.75$$

The *maxpathSim* for each MPE is calculated and the *schemaSim* is determined by combining them all.

### 3.6 Data mining: Clustering the schemas according to their similarity

The constrained hierarchical agglomerative clustering method is used for grouping similar schemas. This method uses a bottom-up strategy that initially assigns each object to its own cluster and then pairs of clusters are repeatedly merged until the number of classes is sufficiently small or until certain termination conditions are satisfied [36]. The reasons to use this method are manifold. Firstly, similarity of clusters is based on the number of common elements that the schemas share. There may be schemas that form small and reasonably cohesive clusters, as well as the schemas that are not part of particularly cohesive groups. The type of clusters desired is therefore globular in nature. This algorithm has been shown to be very powerful at discovering arbitrarily shaped clusters.

Secondly, the algorithm repeatedly merges the pair of clusters to form a final solution. Therefore this clustering process can be analysed in the post-processing phase to form a hierarchy of schema classes. Thirdly, the algorithm must be resistant to noise and outliers. Since the data collection can have a schema that may not be related to other schemas, outliers may be present. This algorithm uses a k-nearest neighbour graph in the partitioning phase that ensures to reduce the effects of noise and outliers. Fourthly, the algorithm should not require the number of clusters to be pre-determined because the relationships between data are unknown. Finally, because the volume of query data can be very large, the algorithm should be scalable.

We use the wCluto web-enabled data clustering application [31] for clustering the XML data. In order to use Wcluto, XMine first generated a matrix containing the *schemaSim* coefficient (common path similarity coefficient) between the trees in the data source (pair-wise similarity) using path similarity threshold of 0.7. The Wcluto takes in the *schema similarity* matrix and performs the clustering process. The 'Complete-Link' merging criterion function is chosen for computing the distance between clusters.

Based on the clustering results, the discovered schemas classes serve as a basis for the visualization of the clustering solution and the generation of schema class hierarchy in the last phase of post-processing.

### 3.7 Post processing: Generating a hierarchy of schema classes

In the final phase, the discovered schema patterns are visualized as a tree of clusters called dendrogram (an example is shown in figure 10). The dendrogram shows the clusters that are merged together and the distance between these merged clusters. This facilitates the generalization and specialization processes of the clusters to develop an

appropriate schema class hierarchy. Each cluster, that contains a set of similar schemas, forms a node in the hierarchy, where all nodes (or clusters) are at the same conceptual level. Each cluster may be further decomposed into several schema sub clusters, forming a lower level of the hierarchy. Clusters may also be grouped together to form a higher level of the hierarchy.

A new schema can now be generalized. First, the schema is generalized to the identifier of the lowest subclass to which the schema belongs. The identifier of this subclass can then, in turn, be generalized to a higher-level class identifier by climbing up the class hierarchy. Similarly, a class or a subclass can be generalized to its corresponding superclasses by climbing up its associated schema class hierarchy.

Domain	No. of Sources	No. of Nodes	Nesting levels
Automobile	9	10-40	2-10
Property	16	20-50	5-15
Travel	52	20-50	2-16
Health	20	40-80	5-8
Flights	20	20-100	4-15
Publication	40	20-500	4-10
Hotel Messages	25	50-1000	7-20

**Table 6: The Input Data Set**

## 4 Empirical Evaluation and Discussion

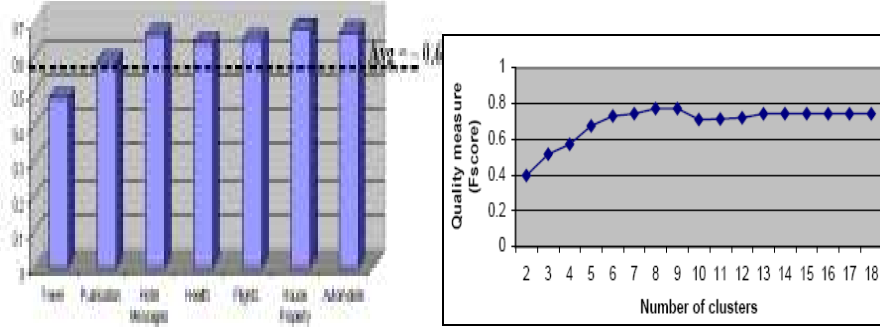
**Dataset:** Table 6 summarizes the major characteristics of the schema collection used in experiments. Each domain consists of a number of different domain categories that have structural and semantic differences. Hence, even though schemas are from the same domain, they might not be considered similar enough to be grouped into the same clusters. Figure 6 illustrates the average similarity degree (using *schemaSim* measurement) between schemas in the seven subject domain categories. The average similarity is estimated at approximately 0.6, showing that schemas are much different even though they come from the same domain.

**Evaluation measures:** The validity and quality of the XMine clustering solutions are verified using two common evaluation methods: (1) the intra-cluster and inter-cluster quality and (2) *FScore* measure.

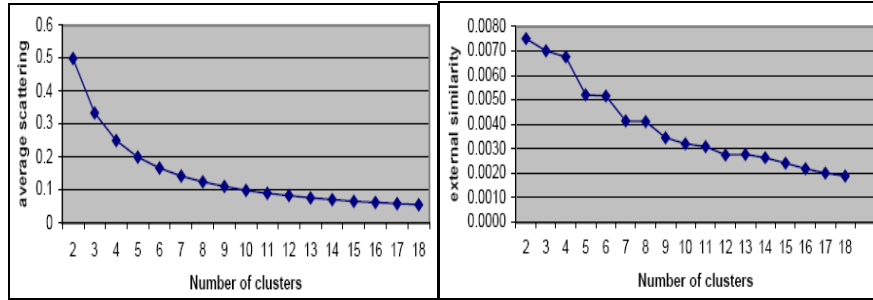
**Result and analysis:** Figure 7 shows the *FScore* of the dataset over the 18 different clustering solutions. The *FScore* result of the 9-clusters solution shows the best *FScore*. When the process reaches to the 13-clusters solution, the clustering quality is stabilized. The objective of clustering is to maximize the intra-class similarity in clusters and to find the compact clusters. XMine demonstrates (figure 8) this by the decreasing tendency in the average scattering compactness of clusters as the number of clusters increases. As the clustering process continues, clusters are further decomposed into smaller sub clusters that contain more highly similar schemas. Thus as the intra-cluster scattering compactness decreases, the more compact schemas result in the clusters. And, after achieving the optimum clusters, the solution is stabilised.



Another objective of clustering is to minimize the inter-class similarity or to find the well separated clusters. The figure 9 confirms that the average external similarity between clusters also decreases as the number of clusters increases. As the clustering process continues, clusters are produced consisting only of highly similar schemas. Based on these observations, the 13-clusters solution produces a better quality of clusters compared to the 9-clusters solution due to the lower intra-cluster scattering and inter-cluster similarity.



**Figure 6:** Average schema similarity coefficient **Figure 7:** FScore measure

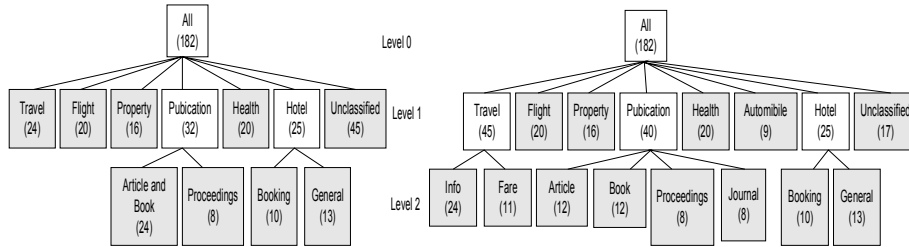


**Figure 8:** Intra-cluster Similarity

**Figure 9:** Inter-cluster similarity

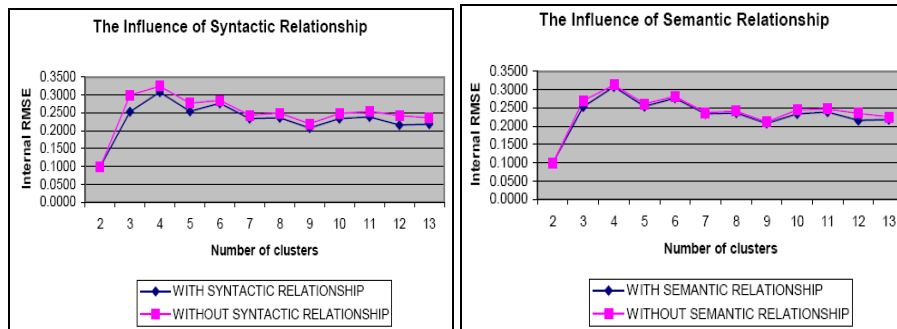
The members of the clusters are also important to examine the correct clustering of the similar schemas into related classes. Figure 10 displays the clusters decomposition for 9 and 13 numbers of clusters. The shaded nodes in the hierarchy represent the actual clusters of the schemas. The unshaded nodes represent the generalization class of the low-level schema classes. Each node is labelled with the class name and the size of the class.

Based on the cluster decompositions of all solution, we can say that the progression in clustering process achieves more disjoint and specific sub-groups (i.e., lesser unclassified patterns). However, the size of these classes becomes very small. In fact, these classes may not be sufficient to consider as an independent class. These clusters may only be holding one specific schema (as it happens in the case with 18 clusters), and this may be an outlier.

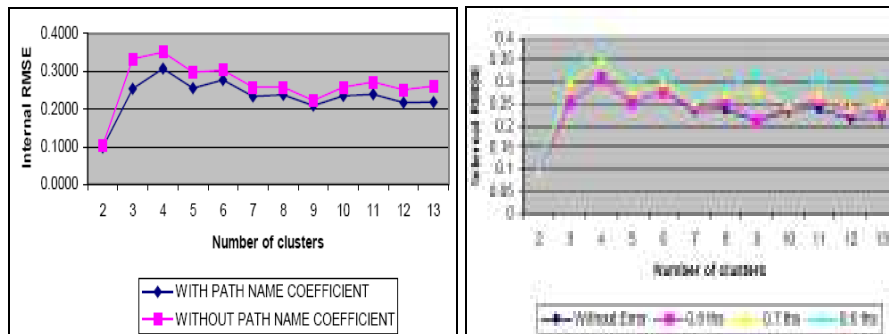


**Figure 10:** The cluster decomposition for 9 & 13 number of clusters

XMine is also examined to test the sensitivity in computing the schema similarity coefficient (*schemaSim*). Without the semantic relationship, XMine is still able to handle the linguistic similarity between element names relatively more effectively (figure 12) than without the syntactic relationship (figure 11). Therefore, syntactic similarity measure is more reliable than semantic similarity measure in measuring the linguistic similarity of two elements, for this particular data set.



**Figures 11 & 12:** Effect of Syntactic and semantic relationships on clustering



**Figure 13:** Influence of PNC

**Figure 14:** Thresholds in Clustering

Figure 13 shows that the PNC measure increases the correctness of the overall similarity of schemas. Without inclusion of *PNC*, the element names with the same

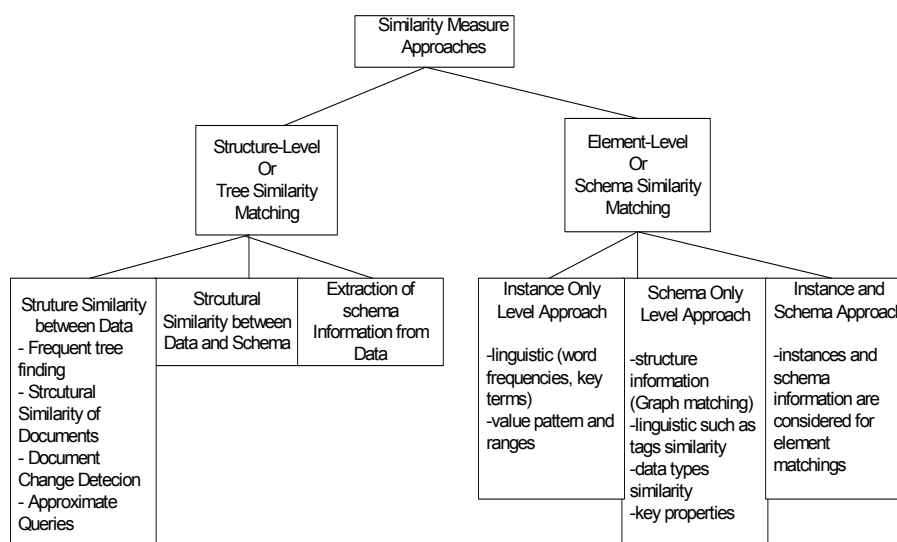
semantics but occurring in different position in the hierarchy path name (i.e. *book.title* and *book.author.title*) cannot be identified and discriminated. Hence the use of path name, *PNC*, shows a better quality of clustering solution compared to only considering single element name matching.

The sensitivity of the XMine in handling the semantic and syntactic similarity between elements depends on the setting of both semantic ( $\delta$ ) and syntactic ( $\mu$ ) threshold values. Figure 14 shows that 0.8 threshold yields the best values in this data.

The schemas with errors (grammatical or typo) would result in low matched values in terms of their element similarity. Hence, by adjusting the threshold values, two elements names with the semantic and syntactic errors can still be accepted as a matched candidate. However, the drawback of setting a low threshold value is a less restrictive matching process. The element pre-processing plays a significant role in element matching process. In XMine, parsing of element names into a set of tokens assists in the automatic selection of possible meanings of the erroneous words. Additionally, the alternative string comparison during the linguistic matching improves the semantic similarity measure.

## 5 Related Work

Research on measuring the structural similarity and clustering of XML data is gaining momentum. We show a taxonomy of these approaches in figure 15 as broadly classified into structure level and element level based similarity approaches.



**Figure 15:** A classification of Similarity Measure Approaches

The *structure-level similarity approaches* can be divided into three different research directions; (1) to detecting and measuring the structure and content similarities between data; (2) to detecting and measuring the structural similarity

between data and schema; (3) to determining the schema information from semistructured data relying on their structural similarities.

The approaches along the first direction can be further decomposed into approaches developed for (1) document clustering [11, 17, 19, 24, 26], (2) change detection in documents [30], and (3) approximate querying of documents [29]. Most of the works developed in these directions rely on the notion of tree edit distance developed in combinational pattern matching [7, 35]. Recently some researchers have developed techniques for frequent tree patterns mining [7]. However, none of these methods take into consideration the hierarchical information (i.e. the level of hierarchy at which an element locates) when representing frequent patterns. It prevents the use of level path information of similar elements to discover the synonym elements for quantifying the similarity between documents for clustering. Thus by ignoring the hierarchical position, these techniques become too restrictive and incompatible for clustering the similar hierarchical trees.

The XMine approach adapts the sequential mining approach [2] to find the maximal paths similar to Lee et al.[17]. [17] defines the structural similarity only based on the 'ratio' between the maximal similar paths and the paths of the base document. They however do not include the element level hierarchy position, leading in erroneous match between two names occurring at two different positions or with different context. XMine overcomes this by including *PNC* in calculation.

There are techniques [4, 28] that aim at measuring the structural similarity between data and schema in the context of XML. Some of these techniques present documents as edge-labelled graphs ignoring the constraints on the repeatability or alternatives of elements in XML schemas. Additionally, [4] can not be directly applicable to cluster documents without any knowledge of their schemas, and is not able to point out dissimilarities among documents referring to the same schema. However, this approach takes into account the context of element into calculation. This concept is adapted in XMine during the similarity computation process.

Nevertheless, majority of existing approaches measure structural similarity between XML documents and thus their goals are substantially different from the XMine methodology, which measures the structural similarity between a set of trees representing schemas. The tree-edit distance approach is also not sufficient enough to measure the semantic and hierarchical structure of the schemas, since it only concerns with the existence of different elements in two trees, but not the cardinality.

The *element-level similarity matching* approaches known as *schema matching* determines the semantic correspondences between elements of two schemas. The main difference between schema matching approach and tree editing problem is that in former, the primary component of determining the similarity between schemas is elements of the trees with respect to their semantic names and name structures similarity. On the other hand, tree editing problem concerns the whole tree structure similarity without concisely taking into account the detailed elements components in the tree. The tree edit problem treats the label of each node in the tree as a second preference. For instance, the cost of relabelling is assumed to be cheaper than that of deleting a node with the old label and inserting a node with the new label. Thus in other words, schema matching is more concerned on the internal matching of the tree, whereas tree edit problem is more concerned on the high-level tree matching.

Researchers have approached schema matching for XML data at three different levels as shown in figure 15. *Instance only level approaches* sometimes fail to capture the structure information of the XML data. Machine learning techniques are used to improve accuracy but can be very computationally expensive[16].

Schema matching at *schema only level approaches* can be used for mapping a collection of heterogeneous XML-Schemas [8, 14, 18, 20, 22]. The document community has also proposed the techniques to automate the process of schema matching to deduce the transform scripts which can rearrange and modify the associated data [6]. The drawback is that finding similar elements at this level can produce more mismatch of elements as no instance data is provided. Therefore the accuracy of the mapping is depended on the technique that is used for linguistic and structure matching at the schema only level approach. The instance or schema only level approach can have some drawback in finding similar elements between XML documents. Therefore some researchers have combined both the instance and schema information for schema matching [9]. These approaches however need both the XML documents and their associated schema definitions to be available for the mapping.

XMine comes closer to a number of schema only level approaches such as XClust [18], Deep [14], Cupid [20], COMA [8], SF [22]. However, the main difference between these approaches and XMine is that the structure similarity is derived based on the maximal similar paths obtained by using the adapted sequential pattern mining algorithm. Thus, this eliminates the element-to-element matching process, making XMine an efficient and accurate method.

## 5 Conclusions and Future Work

The potential benefits of the rich semantics of XML have been recognized widely for enhancing document handling. A schema clustering process improves the document handling process in digital libraries and XML repositories by organising heterogeneous schemas into groups. This paper presented the XMine methodology that accurately clusters the schemas by considering both structural and semantic information of elements. The element structural similarity is the hierarchical position of the element in the schema. XMine includes the structural information in similarity measurement by finding the maximal similar paths between schemas. The context of an element, which is defined by its level position among other elements in a path expression, is included in measuring similarity between maximal paths. This takes into account the elements with the same name but in different level position in the hierarchical tree. The element semantic similarity includes the linguistic and constraint similarity between elements contained only in the maximal large paths. Thus, this eliminates the element-to-element matching process of two trees and rather focuses only on elements those appear in maximal paths.

The evaluation shows the effectiveness of XMine in categorizing the set of heterogeneous schemas into relevant classes that facilitate the generalization of an appropriate schema class hierarchy. The sensitivity evaluation shows that the XMine pre-processing components influences the quality of clusters. The XMine's semantic and structural similarity measures ensure that equivalent concepts occurring in completely different structures, and completely independent concepts that belong to

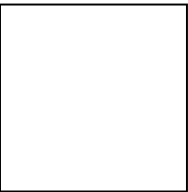
isomorphic structures, are recognised and considered appropriately during the clustering process.

This schema clustering approach can also easily be applicable to document instances after representing each document as a tree. Moreover, the methodology is applicable to general web documents after performing XHTML conversion, and then representing documents as trees.

## References

- [1] Abiteboul, S., Buneman, P., & Suciu, D. (2000). *Data on the Web: From Relations to Semistructured Data and XML*. California: Morgan Kaufmann.
- [2] Agrawal, R., & Srikant, R. (1996). *Mining Sequential Patterns: Generalizations and Performance Improvements*. Paper presented at the the 5th International Conference on Extending Database Technology (EDBT'96), France.
- [3] Berkhin, P. (2002). *Survey of clustering data mining techniques*: Technical Report, Accrue Software, San Jose, CA.
- [4] Bertino, E., Guerrini, G., & Mesiti, M. (2004). A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and its applications. *Information Systems*, 29(1), 23-46.
- [5] Boag, S., Chamberlin, D., Fernández, M., Florescu, D., Robie, J., & Siméon, J. *XQuery 1.0: An XML Query Language*. Retrieved September, 2005, from <http://www.w3.org/TR/2005/WD-xquery-20050915/>
- [6] Boukottaya, A., & Vanoirbeek, C. (2005, November 02-04). *Schema matching for transforming structured documents*. Paper presented at the the 2005 ACM symposium on Document engineering, Bristol, United Kingdom.
- [7] Chi, Y., Nijssen, S., & Muntz, R. (2005). Frequent Subtree Mining - An Overview. *Fundamenta Informaticae Special Issue on Graph and Tree Mining*, 66(1-2), 161-198.
- [8] Do, H. H., & Rahm, E. (2002 August). *COMA - A System for Flexible Combination of Schema Matching Approaches*. Paper presented at the 28th VLDB, Hong Kong, China.
- [9] Doan, A., Domingos, R., & Halevy, A. Y. (2001). *Reconciling schemas of disparate sources: a machine-learning approach*. Paper presented at the ACM SIGMOD, Santa Barbara, California, United States.
- [10] Fellbaum, C. (1998). WordNet: An Electronic Lexical Database. *MIT Press*.
- [11] Flesca, S., Manco, G., Masciari, E., Pontieri, L., & Pugliese, A. (2005). Fast Detection of XML Structural Similarities. *IEEE Transaction on Knowledge and Data Engineering*, 7(2), 160-175.
- [12] Guardalben, G. (2004). *Integrating XML and Relational Database Technologies: A Position Paper*. Retrieved May 1st, 2005, from [http://www.hitsw.com/products\\_services/whitepapers/integrating\\_xml\\_rdb/integrating\\_xml\\_white\\_paper.pdf](http://www.hitsw.com/products_services/whitepapers/integrating_xml_rdb/integrating_xml_white_paper.pdf)
- [13] *Introduction to XML Schema by Refsnes Data*. (2005, April 25). from [http://www.w3schools.com/schema/schema\\_intro.asp](http://www.w3schools.com/schema/schema_intro.asp)

- [14] Jeong, E., & Hsu, C.-N. (2001). *Induction of integrated view for XML data with heterogeneous DTDs*. Paper presented at the 10th International Conference on Information and Knowledge Management, Atlanta, Georgia, USA.
- [15] Koloniari, G., & Pitoura, E. (2005). Peer-to-peer management of XML data: Issues and research challenges. *SIGMOD Record*, 34(2), 6-17.
- [16] Kurgan, L., Swiercz, W., & Cios, K. (2002). *Semantic Mapping of XML Tags using Inductive Machine Learning*. Paper presented at the ICMLA.
- [17] Lee, J. W., & Park, S. S. (2004, October 20-24). *Finding Maximal Similar Paths Between XML Documents Using Sequential Patterns*. Paper presented at the ADVIS, Izmir, Turkey.
- [18] Lee, L. M., Yang, L. H., Hsu, W., & Yang, X. (2002, November). *XClust: Clustering XML Schemas for Effective Integration*. Paper presented at the 11th ACM International Conference on Information and Knowledge Management (CIKM'02), Virginia.
- [19] Leung, H.-p., Chung, F.-l., & Chan, S. C.-f. (2005). On the use of hierarchical information in sequential mining-based XML document similarity computation. *Knowledge and Information Systems*, 7(4), 476-498.
- [20] Madhavan, J., Bernstein, P. A., & Rahm, E. (2001). *Generic Schema Matching with Cupid*. Paper presented at the 27th VLDB, Roma, Italy.
- [21] Meier, W. (2002). *eXist: An open source native XML database*. Paper presented at the Web, Web-services, and database systems.
- [22] Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). *Similarity Flooding: A Versatile Graph Matching Algorithm*. Paper presented at the ICDE.
- [23] Nayak, R., Witt, R., & Tonev, A. (June 24-27 2002). *Data Mining and XML documents*. Paper presented at the The 2002 International Workshop on the Web and Database (WebDB 2002).
- [24] Nayak, R., & Xu, S. (2006). *XCLS: A Fast and Effective Clustering Algorithm for Heterogenous XML Documents*. Paper presented at the the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore.
- [25] Nayak, R., & Zaki, M. (Eds.). (2006). *Knowledge Discovery from XML documents: PAKDD 2006 Workshop Proceedings* (Vol. 3915). *Lecture Notes in Computer Science*: Springer-Verlag Heidelberg.
- [26] Nierman, A., & Jagadish, H. V. (2002, December). *Evaluating Structural Similarity in XML Documents*. Paper presented at the 5th International Conference on Computational Science (ICCS'05), Wisconsin, USA.
- [27] Rice, S. V., Bunke, H., & Nartker, T. A. (1997). Classes of Cost Functions for String Edit Distance. *Algorithmica*, 18(2), 271-280.
- [28] Suzuki, N. (2005, March 13-17). *Finding an optimum edit script between an XML document and a DTD*. Paper presented at the Proceedings of the 2005 ACM symposium on Applied computing, Santa Fe, New Mexico.
- [29] Theobald, A., & Wiekum, G. (2000). *Adding Relevance to XML*. Paper presented at the The 3th International Workshop on the Web and Databases (WebDB'00), Dallas.

- 
- [30] Wang, Y., DeWitt, D. J., & Cai, J. Y. (2003). *X-Diff: An Effective Change Detection Algorithm for XML Documents*. Paper presented at the The 19th IEEE ICDE.
  - [31] *wCluto: Web Interface for CLustering TOolKit*. (2003). Retrieved July 25, 2005, from <http://cluto.ccgb.umn.edu/cgi-bin/wCluto/wCluto.cgi>
  - [32] *XML Schema*. from <http://www.w3.org/XML/Schema>
  - [33] Xylem, L. (2001). *Xylem: A dynamic Warehouse for XML data of the Web*. Paper presented at the IDEAS.
  - [34] Yergeau, F., Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (2004). *Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation*. Retrieved February, 2004, from <http://www.w3.org/TR/2004/REC-XML-20040204/>
  - [35] Zhang, K., & Shasha, D. (1989). Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM Journal Computing*, 18(6), 1245-1262.
  - [36] Zhao, Y., & Karypis, G. (2002). *Evaluation of Hierarchical Clustering Algorithms for Document Datasets*. Paper presented at the The 2002 ACM CIKM, Virginia, USA.