Semantic and Structure Based XML Similarity: The *XS*^{3} Prototype

Joe Tekli
LE2I Laboratory CNRS
University of Bourgogne
21078 Dijon Cedex
France
joe.tekli@khali.u-bourgogne.fr

Richard Chbeir
LE2I Laboratory CNRS
University of Bourgogne
21078 Dijon Cedex
France
richard.chbeir@u-bourgogne.fr

Kokou Yetongnon
LE2I Laboratory CNRS
University of Bourgogne
21078 Dijon Cedex
France
kokou.yetongnon@u-bourgogne.fr

1. Introduction

Due to the ever-increasing web availability of XML-based data, an efficient approach to compare XML documents becomes crucial in information retrieval. Such comparison of XML documents has applications in version control (finding, scoring and browsing changes between different versions of a document), change management and data warehousing (support of temporal queries and index maintenance) [3, 4, 5], XML query systems (finding and ranking results according to their similarity in order to retrieve the best results possible) [10, 12] as well as in the classification/clustering of XML documents gathered from the web against a set of DTDs declared in an XML database (just as schemas are necessary in traditional DBMS for the provision of efficient storage, retrieval, protection and indexing facilities, the same is true for DTDs and XML repositories) [1, 2, 8].

Here, we present our XML comparison prototype *XS*^{3} (XML Structural and Semantic Similarity) able to integrate IR semantic similarity assessment in an edit distance structural similarity algorithm, seeking to amend similarity judgments when comparing heterogeneous[1] XML-based documents.

2. System architecture

XS^3 prototype (implemented using C#) is made up of four components: a v*alidation component,* an *edit distance component,* a *synthetic XML data generator*, and a *taxonomic analyzer*. They are respectively presented in the following subsections.

Figure 1 depicts the overall architecture of our system. XML documents (synthesized via our *XML generator* or attained from some external source) are tested using our *validation component* before being passed to the *edit distance component* for similarity computations. Semantic similarity values between pairs of words/expressions would have to be computed by the *taxonomic analyzer* prior to the comparison phase, so as to be used (via our SCM) in identifying the semantic relatedness between element/attribute node labels.

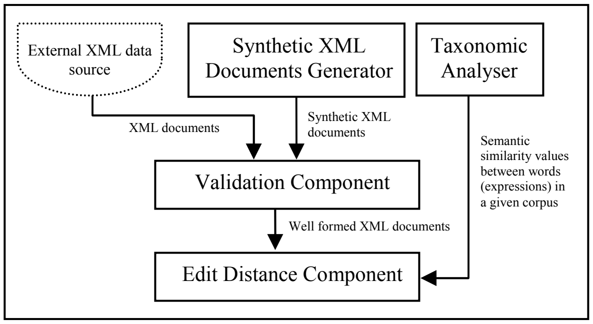

External XML data source
Synthetic XML Documents Generator
Taxonomic Analyser
XML documents
Synthetic XML documents
Semantic similarity values between words (expressions) in a given corpus
Validation Component
Well formed XML documents
Edit Distance Component

Fig. 1. Overall system architecture

2.1 Validation component

The *validation component* verifies the integrity of XML documents, testing if the documents are well formed with respect to the basic XML syntactic rules [9]. Note that we have considered the following simplifications in order to gain in computation and time complexity:

- XML documents to be treated by the *validation component* should have attributes appearing as children of their containing elements.
- Empty elements should be handled as PCDATA elements with the constraint of having null contents.

In addition, the *validation component* transforms documents to their Ld-pair representations [11] in order to be treated by the e*dit distance component.*

[1] We note by *heterogeneous* XML document, one that doesn't conform to a given grammar (DTD or XML Schema), which is the case of a lot of XML documents found on the web [7].

2.2 Edit distance component

The *edit distance component* in XS^3 undertakes XML similarity computations following the process developed in our study [11]. It basically comprises of an original operation cost model (SCM), utilized with Chawathe's classical edit distance algorithm [5], in order to take into account the semantic relatedness of XML element/attribute labels in the comparison process.

2.3 Synthetic XML documents generator

A *synthetic XML data generator* was also implemented in order to produce sets of XML documents based on given DTDs. The synthetic XML generator accepts as input: a DTD document, a MaxRepeats[1] value designating the maximum number of times a node will appear as child of its parent (when * or + options are encountered in the DTD), as well as an NbDocs value underscoring the number of synthetic XML documents to be produced.

Note that for the sake of computational simplicity, the DTDs to be treated by our generator should respect certain criterion in order to yield well formed documents:

– Attributes must be introduced as children of their containing elements:
  - <!ELEMENT Employee (#PCDATA)>
    <!ATTLIST Employee Name CDATA>
    The preceding should be replaced by the following:
    <!ELEMENT Employee (Name)>
    <!ELEMENT Name (#PCDATA)>.

– Since attributes are treated as sub-elements, we disregard empty elements:
  - Declarations such as <!ELEMENT Employee (EMPTY)> should not appear in the DTDs presented to our generator.

– We also disregard elements without restrictions:
  - Declarations such as <!ELEMENT Employee (ANY)> should not appear in the DTDs presented to our generator.

– The "OR" operator ("|"), representing an *alternative* of elements, is not considered in our generator. We only consider the "AND" operator, representing *sequences*.
  - Declarations such as <!ELEMENT University (Faculty | Department)> are not analysed by our generator.
    Consequently, declarations containing overlapping brackets should not appear in our input DTDs:
    <!ELEMENT University (Faculty, ((A, Section) | Laboratory)).

– Input DTDs should not contain entity declarations:
  - <!ENTITY … >.

– Input DTDs should not contain prologues:
  - <!DOCTYPE … >.

– Input DTDs should not contain comments:
  - <!-- … -->.

Figure 2 shows a typical DTD that can be treated by our XML documents generator.

2.4 Taxonomic analyzer

Furthermore, a *taxonomic analyzer* was also introduced so as to compute semantic similarity values between words (expressions) in a given taxonomy. Our *taxonomic analyzer* accepts as input a hierarchical taxonomy and corresponding corpus-based word occurrences. Consequently, concept frequencies are computed and, thereafter, used to compute semantic similarity (via Lin [6]) between pairs of nodes in the knowledge base.

```
<!ELEMENT University (Faculty+)>
        <!ELEMENT Faculty (Department+, Laboratory, Section*)>
                <!ELEMENT Department (#PCDATA)>
                <!ELEMENT Laboratory (Professor, Student?)>
                        <!ELEMENT Professor (#PCDATA)>
                        <!ELEMENT Student (#PCDATA)>
                <!ELEMENT Section (#PCDATA)>
```

Fig. 2. Sample DTD

Semantic similarity computational details are given in the main paper [11].

Note that we utilized an indexed Oracle 9i DB[2] table to store, and subsequently access similarity values.
In fact, we had considered computing semantic similarity each time it was needed (practically with each edit distance operation execution). However, pre-computing similarity values for each pair of nodes in the taxonomy at hand and, subsequently, managing them via an indexed table proved to be less time consuming. For example, an average of 0.25 seconds per pair-wise semantic similarity assessment was saved, when exploiting a 677 words WordNet-based taxonomy, owing to that procedure.

3. XS^3 comparison modules

XS^3 enables four comparison functionalities, made available via dedicated comparison modules.

3.1 *One to One* (1/1) comparison module

It is the basic module of the system upon which are built all other three modules. Its function comes down to comparing one XML document X_1 to another document X_2, the user being able to choose between the intuitive cost model ICM (assigning unit operation costs) mostly used in the literature and our semantic cost model SCM [11] (cf. *Figure 3*).

[1] A greater MaxRepeats value increases the probability of attaining greater size and variability when generating synthetic XML documents

[2] Oracle uses the *B-Tree* indexing technique

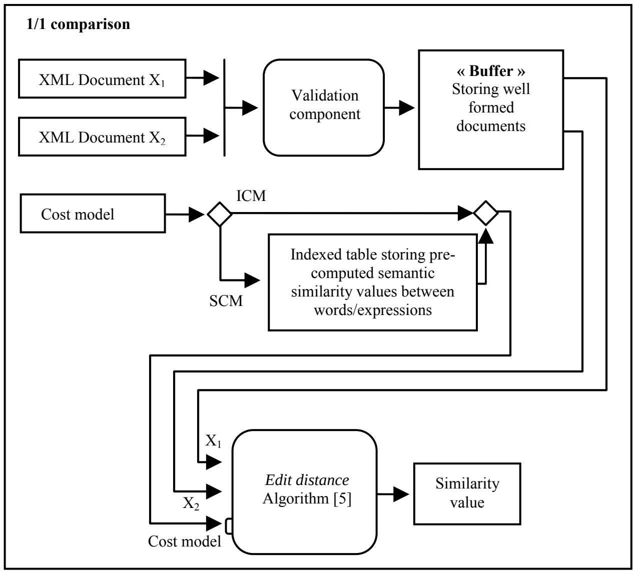

1/1 comparison
XML Document X_1
XML Document X_2
Validation component
« Buffer » Storing well formed documents
Cost model
ICM
SCM
Indexed table storing pre-computed semantic similarity values between words/expressions
X_1
X_2
Cost model
Edit distance Algorithm [5]
Similarity value

Fig. 3. Simplified UML Activity diagram describing the 1/1 comparison functionality of *XS*^{3}.

3.2 *One to Many* (1/∞) comparison module

It utilizes the *one to one* module in order to compare one XML document X_1 to the set of XML documents contained in the same folder. This functionality allows the ranking of documents according to their similarity to X_1 (cf. *Figure 4*).

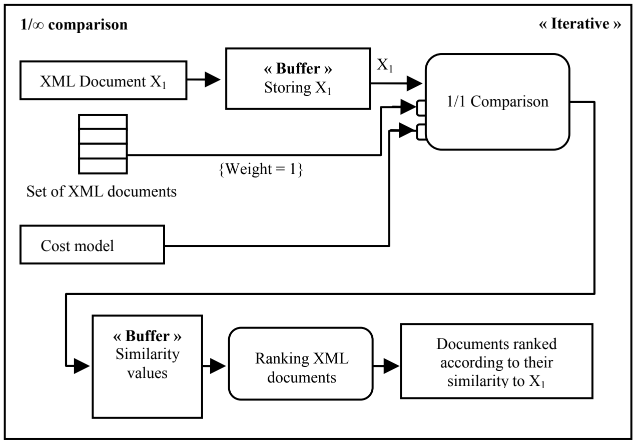


Fig. 4. Simplified UML Activity diagram describing the 1/∞ comparison functionality of *XS*^{3}.

3.3 *Many to Many* (∞/∞) comparison module

It uses the *one to many* comparison module in order to compare XML documents contained in the same folder, one by one. This functionality would allow the clustering of similar documents, the clustering phase not being implemented in our prototype (cf. *Figure* 5).

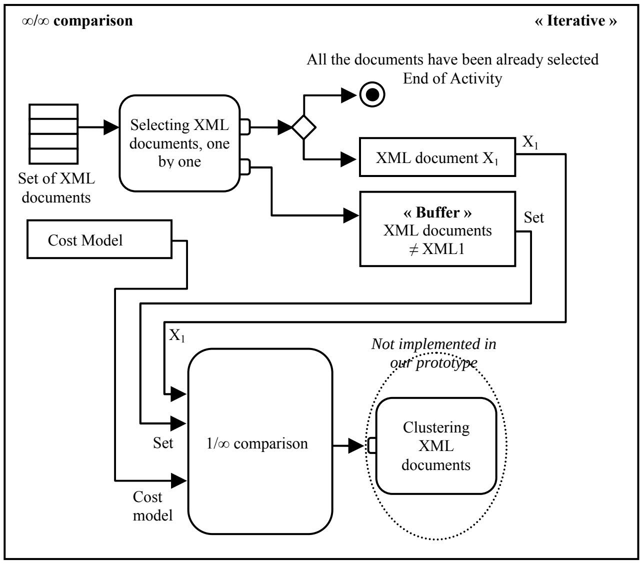


Fig. 5. Simplified UML Activity diagram describing the ∞/∞ comparison functionality of *XS*^{3}.

3.4 *Set* comparison module

It compares sets of XML documents by computing corresponding average similarity scores (cf. *Figure 6*). This functionality was proven useful while undertaking our experiments in order to validate our approach [11].

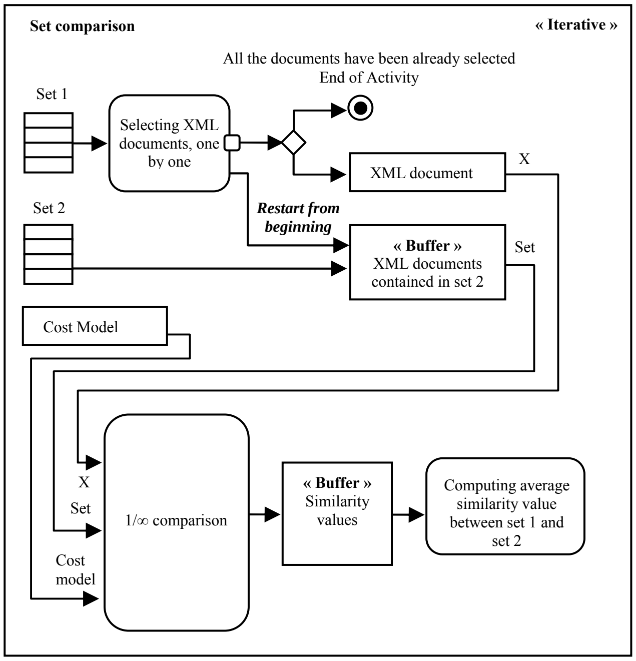
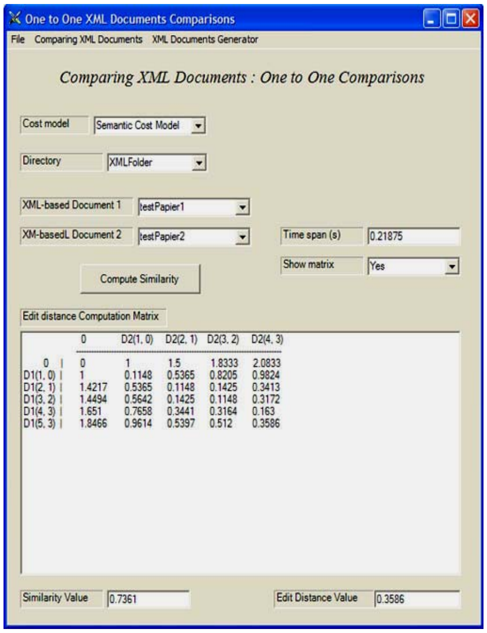

Set comparison
« Iterative »
All the documents have been already selected
End of Activity
Set 1
Selecting XML documents, one by one
XML document
X
Restart from beginning
Set 2
« Buffer » XML documents contained in set 2
Set
Cost Model
X
Set
Cost model
1/∞ comparison
« Buffer » Similarity values
Computing average similarity value between set 1 and set 2

Fig. 6. Simplified UML Activity diagram describing the set comparison functionality of *XS*^{3}.

4. XS^3 Interfaces

XS^3 provides a set of visual interfaces easy to be used and managed. *Figure 7* is a screen shot of its 1/1 comparison interface. The user starts by identifying the cost model to be employed and the XML documents to be compared. Detailed edit distance computations can be depicted if the user so wishes. In addition to the edit distance value and the corresponding similarity value, the time span needed to perform the comparison process is also reported on screen.



Comparing XML Documents : One to One Comparisons

Cost model: Semantic Cost Model

Directory: XMLFolder

XML-based Document 1: testPapier1

XM-basedL Document 2: testPapier2

Time span (s): 0.21875

Show matrix: Yes

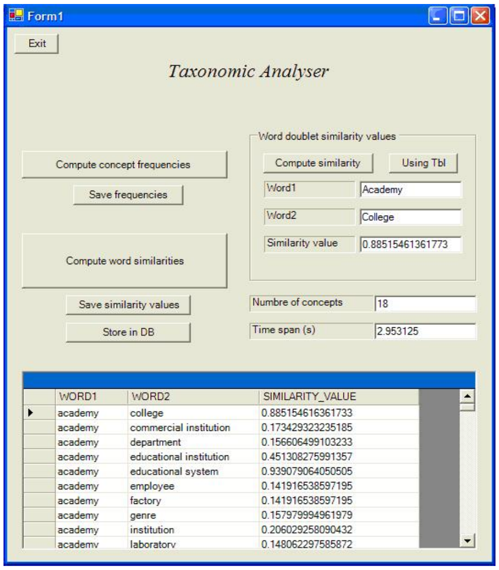
Edit distance Computation Matrix

```
            0        D2(1, 0)  D2(2, 1)  D2(3, 2)  D2(4, 3)
          ----------------------------------------------
      0  |  0        1         1.5       1.8333    2.0833
D1(1, 0) |  1        0.1148    0.5365    0.8205    0.9824
D1(2, 1) |  1.4217   0.5365    0.1148    0.1425    0.3413
D1(3, 2) |  1.4494   0.5642    0.1425    0.1148    0.3172
D1(4, 3) |  1.651    0.7658    0.3441    0.3164    0.163
D1(5, 3) |  1.8466   0.9614    0.5397    0.512     0.3586
```

Similarity Value: 0.7361

Edit Distance Value: 0.3586

Fig. 7. The *One to One* comparison interface.



Taxonomic Analyser

Word doublet similarity values

Word1: Academy

Word2: College

Similarity value: 0.88515461361773

Numbre of concepts: 18

Time span (s): 2.953125

	WORD1	WORD2	SIMILARITY_VALUE
▶	academy	college	0.885154616361733
	academy	commercial institution	0.173429323235185
	academy	department	0.156606499103233
	academy	educational institution	0.451308275991357
	academy	educational system	0.939079064050505
	academy	employee	0.141916538597195
	academy	factory	0.141916538597195
	academy	genre	0.157979994961979
	academy	institution	0.206029258090432
	academy	laboratory	0.148062297585872

Fig. 8. Screen shot of the *taxonomic analyzer* interface.

Figure 8 shows a screen shot of the taxonomic analyzer interface while computing similarity values between words/expressions. After storing a given taxonomy, along with corresponding word occurrences, in dedicated Oracle 9i DB tables, the user utilizes the taxonomic analyzer interface to compute word frequencies as well as corresponding word similarity values. In addition, the user can test the speed of the system, comparing the time to compute the similarity value between a pair of words, with the time to access it in the indexed Oracle 9i DB table dedicated to storing similarity values (if the table is already populated).

XS^3 is available for research purposes and can be obtained from the authors.

References

1. Bertino E., Guerrini G., Mesiti M., Rivara I. and Tavella C., Measuring the Structural Similarity among XML Documents and DTDs, Technical Report, University of Genova, 2002, http://www.disi.unige.it/person/MesitiM.
2. Bertino E., Guerrini G., Mesiti M., A Matching Algorithm for Measuring the Structural Similarity between an XML Documents and a DTD and its Applications, *Elsevier Computer Science*, 29 (23-46), 2004.
3. Chawathe S., Rajaraman A., Garcia-Molina H., and Widom J., Change Detection in Hierarchically Structured Information. In *Proc. of the ACM Int. Conf. on Management of Data (SIGMOD)*, Canada, 1996.
4. Chawathe S., Comparing Hierarchical Data in External Memory. In *Proc. of the 25th VLDB conf.*, p. 90-101, 1999.
5. Cobéna G., Abiteboul S. and Marian A., Detecting Changes in XML Documents. In *Proc. of the IEEE Int. Conf. on Data Engineering*, p. 41-52, 2002.
6. Lin D., An Information-Theoretic Definition of Similarity. *In Proc. of the 15th Int. Conf. on Machine Learning*, 296-304, Morgan Kaufmann Pub. Inc., 1998.
7. Maguitman A. G., Menczer F., Roinestad H. and Vespignani A., Algorithmic Detection of Semantic Similarity. In *Proc. of the 14th Int. World Wide Web Conf.*, 107-116, Chiba, Japan, 2005.
8. Nierman A. and Jagadish H. V., Evaluating structural similarity in XML documents. In *Proc. of the 5th Int. Workshop on the Web and Databases,* 2002.
9. Ray E.T., Introduction à XML. *Edition O'Reilly, Paris, 327 p.*, 2001
10. Schlieder T., Similarity Search in XML Data Using Cost-based Query Transformations. In Proc. of *SIGMOD WebDB Workshop*, 2001.
11. Tekli J., Chbeir R., Yetongnon K., Semantic and Structure Based XML Similarity: An integrated Approach. In proc. of the *13th International Conference on Management of Data (COMAD)*, New Delhi, India, 2006.
12. Zhang Z., Li R., Cao S. and Zhu Y., Similarity Metric in XML documents. *Knowledge Management and Experience Management Workshop*, 2003.