XML Document Comparison: Appendix

Joe Tekli and Richard Chbeir

LE2I Laboratory UMR-CNRS, University of Bourgogne
21078 Dijon Cedex France
{joe.tekli, richard.chbeir}@u-bourgogne.fr

Technical Report - Mai 2010

1. Semantic Similarity between Groups of Words

Methods for evaluating the semantic similarity between pairs of words have been widely studied in the literature [7, 12]. Nonetheless, assessing the semantic similarity between two groups of words/expressions (e.g., node labels of two sub-trees) has not been effectively covered yet. To our knowledge, two complementary approaches have tackled the issue, [5] developed in the context of concept similarity in ontology management systems, and [3] for concept similarity in geographical information systems.

While theoretically sound, our analysis of the solution provided in [3, 5] showed that the approach is not practical (of minimum *O(N!)* complexity). With respect to sub-tree semantic similarity, the main idea in [3, 5] is to compare all nodes of both sub-trees being compared and identify the node pairs that semantically match, i.e., that are most similar. Theoretically, one could proceed as follows.

To evaluate the similarity between all pairs of nodes in both sub-trees being compared (sub-trees being treated as sets of nodes), the *Cartesian product* between the corresponding sub-tree node sets is considered and all sets of pairs of nodes are selected such that there are no two pairs sharing the same node. Such sets would be identified as *restricted pair-wise Cartesian sets* (in comparison with the natural *pair-wise Cartesian set* corresponding to the result of the *Cartesian product*).

Definition A.1 – Restricted pair-wise Cartesian set: given two sub-trees $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, a restricted pair-wise Cartesian set of A and B, denoted $S_{A, B}$, is a set of node pairs $\{(a_1, b_1), \ldots, (a_n, b_n)\} \in \{A \times B\}$ such that $\forall\, i, j \in [1, n]$, $a_i \neq a_j$ and $b_i \neq b_j$.

Then, for each set $S_{A,B}$ corresponding to the *restricted pair-wise Cartesian product* of two sub-trees *A* and *B*, the sum of the semantic similarity values between the labels of each node pair is computed. Consequently, the set of pairs of attributes that yields a maximum semantic similarity score would underline the semantic resemblance between the sub-trees being compared. Note that the maximum semantic similarity value would be divided by the cardinality of the corresponding set $S_{A,B}$ so as to attain a similarity score comprised in the *[0, 1]* interval:

$$Sim_{Set}(A,B) = \underset{S_{A,B}}{Max}\left(\frac{\sum_{i,j \in S_{A,B}} Sim_{Label}(A[i].l, B[j].l)}{|S_{A,B}|}\right) \quad \textbf{(1)}$$

Consider for instance sub-tree sets A'_1 = *{Academy, Professor, PhD Student}* and B'_2 =*{College, Lecturer, Scholar}* of Figure A.1. Evaluating the semantic similarity between A'_1 and B'_2, following [3, 5] comes down to identifying the corresponding *restricted pair-wise Cartesian sets* prior to computing pair-wise similarity values:

- {(Academy, College), (Professor, Lecturer), (PhD Student, Scholar)}
- {(Academy, College), (Professor, Scholar), (PhD Student, Lecturer)}
- {(Academy, Lecturer), (Professor, College), (PhD Student, Scholar)}
- {(Academy, Scholar), (Professor, Scholar), (PhD Student, College)}
- {(Academy, Scholar), (Professor, College), (PhD Student, Lecturer)}
- {(Academy, Scholar), (Professor, Lecturer), (PhD Student, College)}

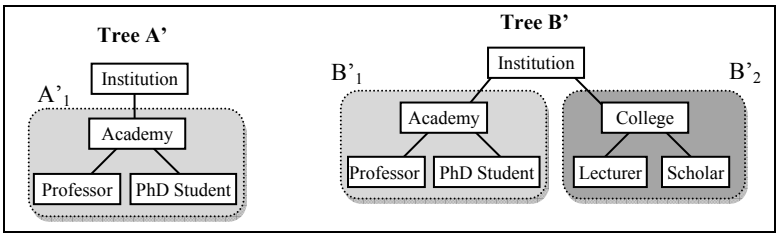


Figure A.1. Sample XML trees with sub-tree semantic similarities.

The problem of identifying the *restricted pair-wise Cartesian sets* comes down to evaluating, in a roundabout way, the determinant of the $CP(A'_1, B'_2)$ matrix corresponding to the *Cartesian Product* of sub-tree sets A'_1 and B'_2, following the Leibniz formula [6]:

$$\det(CP(A'_1, B'_2)) = \sum_{\sigma \in Sn} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} CP(A'_1, B'_2)_{i,\sigma(i)} \quad \textbf{(2)}$$

Having:

$$CP(A'_1, B'_2)= \begin{bmatrix} \text{Academy, College} & \text{Academy, Lecturer} & \text{Academy Scholar} \\ \text{Professor, College} & \text{Professor, Lecturer} & \text{Professor, Scholar} \\ \text{PhD Student, College} & \text{PhD Student, Lecturer} & \text{PhD Student, Scholar} \end{bmatrix}$$

Here, the determinant is not to be computed mathematically since the matrix is not made of numerical numbers. In fact, the *restricted pair-wise Cartesian sets* can be identified w.r.t. to the various permutations that are obtained when applying the Leibniz determinant method (*Formula 2*).

Thus, the major problem with this approach is in the number of possible permutations. The number of permutations, and consequently of number of *restricted pair-wise Cartesian sets* that is obtained, is *N!* (Factorial) for a $N \times N$ matrix. Thus, identifying the *restricted pair-wise Cartesian sets* for large sub-trees is extremely complex and hence impractical. In other words, evaluating the semantic similarity between two groups of words/expressions, following [3, 5], is not practically feasible, despite being theoretically sound.

2. Semantic Resemblance Between Sub-trees: The *Sem_RBS* Algorithm

Algorithm *Sem_RBS* is shown in Figure A.2. It consists of building the vector space corresponding to the XML sub-trees being compared as well as computing the semantic and cosine measures as explained in the main manuscript. In short, after transforming XML sub-trees into semantically weighted vectors, the semantic relatedness between two sub-trees can be evaluated using a measure of similarity between vectors such as the inner product, the cosine measure, the Jaccard measure, etc. Here, we adopt the cosine measure widely exploited in information retrieval:

$$Sem\text{-}RBS(SbT_i, SbT_j, \overline{SN}) = \mathrm{Cos}(\vec{V}_i, \vec{V}_j) = \frac{\sum_{l_r \in SbT_i \cup SbT_j} w_{SbTi}(l_r) \times w_{SbTj}(l_r)}{\sqrt{\sum_{l_r \in SbT_i \cup SbT_j} w_{SbTi}(l_r)^2 \times \sum_{l_r \in SbT_i \cup SbT_j} w_{SbTj}(l_r)^2}} \in [0,1] \quad \textbf{(3)}$$

Consequently, algorithm *Sem_RBS* takes as input the sub-trees SbTi and SbTj to be compared, and the reference semantic network $\overline{SN}$, and generates the sub-tree semantic similarity score ($\in [0, 1]$).

Algorithm *Sem_RBS*

Input: SbT_i and SbT_j // sub-trees in ld-pair,
$\overline{SN}$ // weighted semantic network

Output: Sem_RBS(SbT_i, SbT_j) //Semantic resemblance between SbT_i and SbT_j

Begin

```
VS = Generate_Vector_Space(SbT_i, SbT_j)                                    1
V_i = Generate_Occurrence_Vector(SbT_i, VS)   // Vectors with simple node   2
V_j = Generate_Occurrence_Vector(SbT_j, VS)   // occurrence weights: 0/non null, 3
                                              // taking into account node depths 4
For each node label l_r in V_i   // Computing semantic weights for vector V_i  5
{                                                                           6
    If (w_Vi(l_r) == 0)                                                     7
    {                                                                       8
        For each node label l_s in SbT_i                                    9
        {                                                                   10
            SemWeight = Sim_Label(l_r, l_s, SN) × D-factor(l_r)   // Semantic weight 11
            If (SemWeight > w_Vi(l_r) ) { w_Vi(l_r) = SemWeight }   // Max  12
        }                                                                   13
    }                                                                       14
}                                                                           15

For each node label l_s in V_j   //Computing semantic weights for vector V_j  16
{                                                                           17
    If (w_Vj(l_s) == 0)                                                     18
    {                                                                       19
        For each node label l_r in SbT_j                                    20
        {                                                                   21
            SemWeight = Sim_Label(l_s, l_r, SN) × D-factor(l_s)   // Semantic weight 22
            If (SemWeight > w_Vj(l_r) ) { w_Vj(l_s) = SemWeight }   // Max  23
        }                                                                   24
    }                                                                       25
}                                                                           26

ScalarProduct = 0                                                           27
Module_i=0                                                                  28
Module_j=0                                                                  29

For each label l_r in VS   // Computing the cosine measure Cos(V_i, V_j)    30
{                                                                           31
    ScalarProduct = Product + w_Vi(l_r) * w_Vj(l_r)                         32
    Module_i = Module_i + w_Vi(l_r)^2                                       33
    Module_j = Module_j + w_Vj(l_r)^2                                       34
}                                                                           35

                Product
Return ------------------------------   // Sem_RBS(SbT_i, SbT_j, SN) = Cos(V_i, V_j)  36
       sqrt(Module_i + Module_j)

End
```

Figure A.2. Algorithm *Sem_RBS* for evaluating the semantic resemblance between two sub-trees.

3. Complexity Analysis

3.1. Time Complexity

The overall complexity of our integrated structural and semantic similarity approach simplifies to $O(|T_1|\times|T_2|\times|SN|\times Depth(SN))$, where $|T_1|$ and $|T_2|$ denote the cardinalities of the compared trees, $|SN|$ the cardinality of the semantic network exploited for semantic similarity assessment, $Depth(SN)$ its maximum depth. It is computed as follows:

- *Struct_CBS* algorithm for the identification of the structural commonality between two sub-trees is of complexity: $O(|SbT_i| \times |SbT_j|)$ where $|SbT_i|$ and $|SbT_j|$ denote the cardinalities of the compared sub-trees.
- *Sem_RBS* for identifying the semantic resemblance between two sub-trees is of complexity: $O(|SbT_i|\times|SbT_j|\times|SN|\times Depth(SN))$. Note that $O(|SN|\times Depth(SN))$ underlines the time complexity of the semantic similarity measure itself [7].
- *TOC* algorithm for computing the costs of tree insert/delete operations, which makes use of *Struct_CBS* and *Sem_RBS*, is of time complexity:

$$\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i|\times|SbT_j|\times|SN|\times Depth(SN))$$

Consequently, TOC's time complexity simplifies to $O(|T_1|\times|T_2|\times|SN)|\times Depth(SN))$.

Proof:

(1) Identifying the structural commonalities (via Struct-CBS) between each pair of sub-trees in the source and destination trees is of complexity:

$$\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i| \times |SbT_j|) = \sum_{i=1}^{|T_1|} O(|SbT_i|)\times\sum_{j=1}^{|T_2|} O(|SbT_j|).$$

However, $\sum_{i=1}^{|T_1|} O(|SbT_i|) = O(T_1\times min(Depth(T_1), Leaves(T_1))$ as shown in [11], where $Depth(T_1)$ and $Leaves(T_1)$ are respectively the depth and number of leaf nodes in T_1. This simplifies to $O(|T_1|)$ since XML documents are mostly shallow (the average depth of real XML documents usually hardly ever exceeds 8 levels [2, 8, 9]).

(2) Identifying the semantic resemblances (via Sem-RBS) between each pair of sub-trees in the source and destination trees is of complexity:

$$\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i|\times|SbT_j|\times|SN|\times Depth(SN)) = |SN|\times Depth(SN)\times\sum_{i=1}^{|T_1|} O(|SbT_i|)\times\sum_{j=1}^{|T_2|} O(|SbT_j|)$$

and simplifies to $O(|T_1|\times|T_2|\times|SN|\times Depth(SN))$ as show above.

(3) *Hence, identifying both structural commonalities (vis Struct-CBS) and semantic resemblances (via Sem-RBS) between each pair of sub-trees in the source and destination tree is of complexity:*

$$\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O((|SbT_i| \times |SbT_j|) + (|SbT_i| \times |SbT_j| \times |SN| \times Depth(SN))).$$

It simplifies to $\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i| \times |SbT_j| \times |SN| \times Depth(SN))$.

Consequently, following *(1)* and *(2)*, algorithm *TOC*'s overall complexity amounts to $O(|T_1| \times |T_2| \times |SN| \times Depth(SN))$ □

- The edit distance algorithm *TED* (an adaptation of the algorithm in [10]) which utilizes the results obtained by *TOC* (tree operations costs), is of complexity $O(|T_1| \times |T_2| \times |SN| \times Depth(SN))$, and simplifies to $O(|T_1| \times |T_2|)$ when disregarding semantics.

When disregarding semantic similarity assessment, i.e., when input parameter $\alpha=1$ (thus disregarding algorithm *Sem_RBS*), our approach simplifies to $O(|T_1| \times |T_2|)$, similarly to existing XML-based tree edit distance comparison approaches, e.g., [1, 4, 10].

3.2. Space Complexity

As for memory usage, our approach requires RAM space to store the XML document trees being compared, as well as the distance matrixes and semantic vectors being computed. It simplifies to $O(|T_1| \times |T_2|)$ space (similarly to existing approaches, e.g., [1, 4, 10]) as:

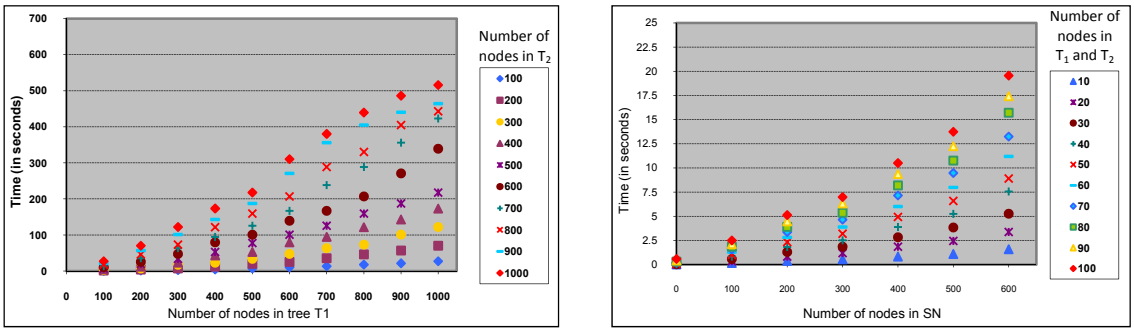
- *Struct_CBS* algorithm requires $|SbT_i| \times |SbT_j|$ space for storing the distance table when identifying the structural commonalities between any two sub-trees SbT_i and SbT_j. Hence, space complexity is of $O(|SbT_i| \times |SbT_j|)$.
- *Sem_RBS* also requires $2 \times (|SbT_i| + |SbT_j|)$ space for handling corresponding sub-tree vectors, each vector being of maximal dimension $|SbT_i| + |SbT_j|$. Hence, *Sem_RBS* is of $O(|SbT_i| + |SbT_j|)$. Note that the semantic network is not stored in local memory, but is stored on disk (and particularly managed via a database system), and thus does not contribute to space complexity.
- *TOC* is of $\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O((|SbT_i| \times |SbT_j|) + 2 \times (|SbT_i| + |SbT_j|))$ space, for storing the various distance trees (*Struct_CBS*) and sub-tree vectors (*Sem_RBS*) between each pair of sub-trees in the source and destination XML trees. It simplifies to $\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i| \times |SbT_j|)$.

Consequently, similarity to the complexity analysis in Section 3.1, *TOC*'s overall space complexity simplifies to $O(|T_1|\times|T_2|)$.

– The edit distance algorithm *TED* is of $O(|T_1|\times|T_2|)$ space complexity.

3.3. Time and Space Experiments

Timing experiments were carried out on a PC with an Intel Xeon 2.66 GHz processor with 1GB RAM. As shown in Section 3.1, our XML comparison method is of $O(|T_1|\times|T_2|\times|SN|\times Depth(SN))$ time complexity. It simplifies to $O(|T_1|\times|T_2|)$ when semantic similarity evaluation is disregarded (i.e., *Sem_RBS* is disregarded). We start by verifying our approach's polynomial (quadratic) dependency on tree size, i.e., $O(|T_1|\times|T_2|)$, which equally underlines a linear dependency on the size of each tree being compared (cf. Figure A.2.a). Therefore, despite being theoretically more complex, timing results demonstrate that our method's time complexity, w.r.t. structural similarity evaluation, is the same those of its alternatives, e.g., *N & J* [10], *DCWS* [4], and *Chawathe* [1]. Note that we do report timing experiments conducted on *N & J*, *DCWS* and *Chawathe* here since the algorithms were implemented in our prototype. In fact, such experiments should be conducted on the original algorithm implementations to obtain accurate results (for instance, our method and *N & J* yield similar timing results, following our prototype processes, whereas *N & J* might run faster/slower using its original implementation package).



a. Structural similarity evaluation (*Sem_RBS* deactivated). **b.** Integrating semantic similarity evaluation.

Figure A.2. Timing results for our XML document comparison approach.

On the other hand, when evaluating both structural and semantic similarity (i.e, when both *Struct_CBS* and *Sem_RBS* algorithms are considered), the size of reference semantic network, as well as the semantic similarity measure (e.g., Lin's measure [7]) being exploited to compute pair-wise XML node label similarity, come to play.

To our knowledge, timing analysis for Lin's measure [7] was not carried out previously. Theoretically, it can be estimated as $O(|SN|\times Depth(SN))$ [7] which is due to traversing the semantic network when searching for the lowest common ancestor between two taxonomic nodes. Thus, in order to reduce our method's overall complexity, we chose to pre-compute

semantic similarity for each pair of nodes in the taxonomy considered (which took about 30 seconds for the WordNet fragment depicted in Figure A.2.b, and more than 5 CPU hours for a 600 node semantic network) and store the results in a dedicated indexed table (Oracle 9i DB)[1]. Consequently, *Sem_RBS* would access the indexed table to acquire semantic values instead of traversing the taxonomy to compute semantic similarity each time it is needed (pair-wise similarity values are computed once, prior to comparing XML documents). Due to this process, we eliminated the impact of taxonomic depth on overall timing complexity. Timing results in Figure A.2.b show that our approach becomes linearly dependent on the size on the taxonomy considered, complexity simplifying from $O(|T_1| \times |T_2| \times |SN| \times Depth(SN))$ to $O(|T_1| \times |T_2| \times |SN|)$.

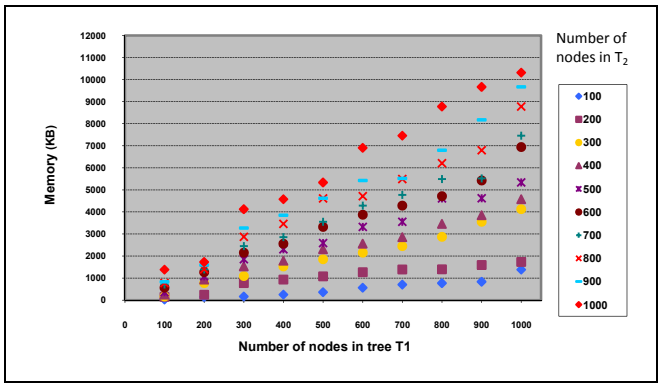


Figure A.3. Memory usage for our XML document comparison approach.

As for space complexity, memory usage results in Figure A.3 show that our approach is quadratic in the combined size of the trees being compares, $O(|T_1| \times |T_2|)$, which underlines a linear dependency on the size of each tree.

References

[1] Chawathe S., *Comparing Hierarchical Data in External Memory.* Proceedings of the International Conference on Very Large Data Bases (VLDB), 1999. pp. 90-101.

[2] Choi B., *What are Real DTDs Like?* Proceedings of the International Workshop on the Web and Databases (WebDB), 2003. pp. 43-48.

[3] D'Ulizia A.; Ferri F.; Formica A.; Grifoni P. and Rafanelli M., *Structural similarity in geographical queries to improve query answering.* Proceedings of the 2007 ACM symposium on Applied computing, 2007. pp. 19-23.

[4] Dalamagas T.; Cheng T.; Winkel K.; and Sellis T., *A Methodology for Clustering XML Documents by Structure.* Information Systems, 2006. 31(3):187-228.

[5] Formica A. and Missikoff M., *Concept Similarity in SymOntos: An Enterprise Ontology Management Tool.* The Computer Journal, 2002. 45(6), pp. 583-594.

[6] Knobloch E., *Beyond Cartesian limits: Leibniz's passage from algebraic to "transcendental" mathematics.* Historia Mathematica, 2006. 33(1):113-131.

[1] Oracle uses the *B-Tree* indexing technique.

[7] Lin D., *An Information-Theoretic Definition of Similarity.* Proceedings of the International Conference on Machine Learning (ICML), 1998. pp. 296-304. Morgan Kaufmann Pub. Inc.
[8] Mignet L.; Barbosa D. and Veltri P., *The XML Web: a First Study.* In Proceedings on the International World Wide Web Conference (WWW), 2003. 2, pp. 500-510.
[9] Mlynkova I.; Toman K. and Pokorny J., *Statistical Analysis of Real XML Data Collections.* Proceedings of the 13th International Confenrence on Management of Data (COMAD), 2006. pp., 20-31.
[10] Nierman A. and Jagadish H. V., *Evaluating structural similarity in XML documents.* Proceedings of the ACM SIGMOD International Workshop on the Web and Databases (WebDB), 2002. pp. 61-66.
[11] Shasha D. and Zhang K., *Approximate Tree Pattern Matching.* Pattern Matching in Strings, Trees and Arrays, Oxford University Press, 1995. chapter 14.
[12] Wu Z. and Palmer M., *Verb Semantics and Lexical Selection.* Proceedings of the 32nd Annual Meeting of the Associations of Computational Linguistics, 1994. pp. 133-138.