International Journal of Trade, Economics and Finance, Vol.2, No.1, February, 2011
2010-023X

An Optimization Approach for Semantic-based XML Schema Matching

Jaewook Kim, Yun Peng, Nenad Ivezic, and Junho Shin

Abstract—**We propose a novel solution for semantic-based XML schema matching, taking a mathematical programming approach. This method identifies the globally optimal solution for the problem of matching leaf nodes between two XML schema trees by reducing the tree-to-tree matching problem to simpler problems of path-to-path, node-to-node, and word-to-word matching. We formulate these matching problems as maximum-weighted bipartite graph matching problems with different constraints, which are solved by different mathematical programming techniques, including integer programming and dynamic programming. Solutions to simpler problems provide weights for the next stage until the optimal tree-to-tree matching solution is obtained. The effectiveness of this approach has been verified and demonstrated by computer experiments.**

Index Terms—**E-business, XML schema matching, maximum-weighted bipartite graph, semantic similarity, mathematical programming.**

I. Introduction

Over the past two decades, the eXtensible Markup Language (XML) [1] and XML schemas [2] have been widely used in the electronic business (e-Business) transactions among enterprises to exchange business documents with their partners (e.g., suppliers and customers in the supply chain) [3]–[5]. Many enterprises and organizations have defined their own XML schemas to describe the structure and content of the business documents to be used in the transactions. Many organizations have also published standard XML schemas to be shared in the transactions within specific industry domains (e.g., e-manufacturing, e-government, and e-health industries) [6]–[8].

The popularity of XML leads to an integration problem as different enterprises or organizations often choose different XML representations for the same or similar concepts [4], [5]. One of the most critical steps to achieving the seamless exchange of information between heterogeneous e-Business systems is schema matching. Schema matching is a process that takes as input two heterogeneous schemas and possibly some auxiliary information, and returns a set of dependencies, so called mappings that identify semantically related elements and attributes [9]. This process has largely been manual and is known to be costly and error-prone [10], [11].

An XML schema defines a set of discrete elements and attributes for a class of XML documents, aiming at defining the structure, content and semantics of XML documents [2]. XML documents that attempt to adhere to an XML schema are said to be instances of that schema (i.e., XML instances). XML schemas or instances are typically viewed as labeled trees (i.e., rooted acyclic graphs) where each node represents a data element or an attribute named by a label of English word or concatenation of words or their abbreviations. Most schema matching approaches analyze the similarity between these labeled trees based on their syntactic and structural information [9], [10]. For the structural similarities, they analyze the differences in hierarchical tree structures. For semantic similarities, they typically analyze the meaning (semantics) of nodes in the labeled tree. Those semantics are often obtained by lexical analysis of English words in the labels of nodes.

XML schemas can be classified into two types according to the types of the e-Business standard schemas. The first type is the component schema. This type of schema contains a set of global type components that can either be extended or reused by other components (e.g., OAGIS UBL Common Core Component schema [12]). The term "components" here refers to either elements or types [2]. Component schemas can be thought of as a collection of labeled trees, each of which corresponds to a global type component. The second type of XML schemas is the document schema. It defines the syntax and structure of a single global type element for a class of valid XML instance (e.g., Purchase Order Document Schema). The document schema can reuse or extend the components defined by the component schemas. It can be viewed as a single labeled tree.

The component schema matching primarily seeks to identify the relations between two sets of labeled trees (i.e., two sets of global type components), whereas the document schema matching identifies relations between nodes (elements or attributes) of two labeled trees (i.e., two schemas).

The document schema matching problems can be further classified according to their purposes. If two document schemas need to be fully matched to create an integrated schema, every node in one schema should be matched to some nodes in the other schema. On the other hand, if the

Manuscript received January 30, 2011. This work was supported in part by NIST award 70NANB9H9145.

Jaewook Kim is with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250 USA (e-mail: jaewook2@umbc.edu).

Yun Peng is with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250 USA (e-mail: ypeng@umbc.edu).

Nenad Ivezic is with the National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (e-mail: nivezic@nist.gov).

Junho Shin is with the National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (e-mail: junho.shin@nist.gov).

matching is to determine how to transform one instance into another, only leaf nodes in the schema trees need to be matched. The leaf nodes are also called *atomic nodes* because it cannot be further decomposed. In this paper we focus on the latter which identifies matching between all atomic nodes of two schemas or instances based on their semantics (meaning of nodes). We call this problem *tree-to-tree matching* as it attempts to matching *all* atomic nodes between two schema trees. Matching between those atomic nodes helps to determine how a certain value in one XML instance can be transformed to certain value of the other for successful exchange of information. Also, we only consider pair-wise matchings of 1-to-1 cardinality (e.g., any atomic node in the source schema can match no more than one atomic node in the target schema).

We propose new innovative techniques to address two challenging problems in this type of schema matching. First, due to synonyms (different words meaning the same thing) and multi-senses (one word having different meanings in different contexts) found in natural languages, the meaning of an atomic node cannot be determined solely by the words in its label. Although XML does not provide means to formally define the semantics, the *semantic ambiguity* can be reduced by contextual information such as the labels of its neighboring nodes. In this paper, we concentrate on one type of context for an atomic node: the nodes along the path from the root to the leaf in the schema tree.

Second, it is difficult to correctly identify the *best set* of matching pairs for all atomic nodes between two schema trees. This is because a leaf in one tree may match more than one leaf in the other tree (with different semantic similarities) and best-matching pairs identified in isolation do not necessarily form the globally optimal set of matchings for all atomic nodes. We propose to use mathematical programming techniques to solve this combinatorial optimization problem. To further reduce the computational complexity, we propose to decompose the global problem into simpler matching problems such as path-to-path, node-to-node, and word-to-word matching. We formulate the sequence of matching problems as *maximum-weighted bipartite matching problems* with different sets of constraints. We solve these optimization problems by different mathematical programming techniques, including *integer programming* [13] and *dynamic programming* [14]. Solutions to simpler problems provide weights for the next stage until the optimal tree-to-tree matching is obtained.

The remainder of the paper is organized as follows. Section II provides a brief survey of the related works. The detailed algorithms of the proposed approach are described in Sections III. Section IV reports the experiments and results. Section V concludes with the directions for future research.

II. Related Works

Many schema matching methods have been proposed [9], [10]. Typically, these methods first attempt to identify semantic similarity between the elements of two schemas. So our survey starts with the existing semantic similarity techniques that have been used to assist in matching between two schemas.

A. Semantic similarity techniques

To compute the semantic similarity, string-based similarity metric is commonly used to analyze the linguistic context of names and name descriptions of schema entities. There are a variety of string-based similarity metrics. Hamming distance is one of the simplest metrics, which measures between two strings of equal length the minimum number of substitutions required to change one string into the other [15]. Levenshtein distance, often called edit distance, provides an extended version of hamming distance by measuring the amount of difference between two string sequences [16]. Jaccard similarity coefficient [17], a well-known statistical method for similarity measure between two sets, is defined as the size of the intersection divided by the size of the union of the two sets: $J(A,B) = |A \cap B| / |A \cup B|$. Cosine coefficient [18] is a common vector space similarity metric similar to Jaccard coefficient in which the input string is transformed into vector space so that the Euclidean cosine rule [19] can be used to determine similarity. N-gram (q-gram) [20], [21] can be also used to determine similarity. A string-distance can be measured by counting the number of the occurrences of different n-grams, i.e., the substrings of length n, in the two strings. The more similar the strings are, the more n-gram they will have in common. [22].

The string-based similarity metrics can be enhanced using natural language preprocessing techniques for the input string, such as tokenization, lemmatization, and elimination [23]. To further enhance the string-based metrics, document corpus resources can be utilized for more accurate and less ambiguous semantics (e.g., synonyms or hyponyms) for words in the node labels. One of the important resources is the lexical taxonomy among the words (e.g., parents, children, ancestor, and descendant relationships). Common knowledge corpora, such as WordNet [24] and domain-specific corpora, can be used to help to determine the meaning of the words. Based on those corpora, several methods have been proposed [25], [26].

A corpus also provides statistical information related to the importance of words and the relationships between words. The information content (IC)-based metric was proposed to utilize this statistical information [27]–[29]. This approach measures the similarity between two entities – two words, two objects, or two structures – A and B based on how much information is needed to describe $common(A,B)$, the commonality between them. Examples of commonality include the features or hypernyms the two words share. According to information theory [29], entities that appear widely in many objects carry less information than rarely appearing ones, and thus are considered less important insemantic similarity measures. In other words, more specific entities carry more information than generic and common entities. Therefore, the more specific the $common(A,B)$ is, the more similar A and B will be. The information content of a concept or word C is defined as [29]:

$$I(C) = -\log P(C) \quad (1)$$

The $common(A,B)$ can then be measured by the information content of the most specific common hypernyms

of A and B. Applying this approach to tree-like IS-A taxonomies [29], one can measure the similarity between A and B as

$$sim_{IC}(A,B) = \max_{C \in S(A,B)} I(C) = \max_{C \in S(A,B)} (-\log P(C)), \quad (2)$$

where $S(A, B)$ is the set of all concepts that subsume both A and B, $I(C)$ is the information content of C, and $P(C)$ is based on the frequency of C in a corpus.

Lin [28] proposed a normalized information content based measure. In a general form, this measure is defined as

$$sim_{I}(A,B) = \frac{I(comman(A,B))}{I(description(A,B))}, \quad (3)$$

where $description(A,B)$ is the sum of $common(A,B)$, and $difference(A,B)$.

For tree-like IS-A taxonomies, Lin also suggested:

$$sim_{IC}(A,B) = \frac{2 \cdot \log P(C)}{\log P(A) + \log P(B)}, \quad (4)$$

where C is the most specific subsumer of A and B with the smallest prior probability and the probabilities can be obtained according to the frequencies in a corpus. Equation (4) can be seen as a normalized version of (2).

Information contents of words or concepts can also be used as their weights when computing composite similarity measure between groups of words.

Based on these semantic similarity techniques, many schema matching methods have been developed [9], [10]. We now look at the state of the art schema matching approaches.

B. The state of the art schema matching approaches

Several hybrid and composite matching approaches have been proposed recently. Reference [30] proposed a hybrid matching approach for component schema matching, called *layered semantic similarity metrics*. To compute the semantic similarity between two global data elements defined in two XML schemas, this approach divides the tree structure of each schema into three layers (i.e., top, inner, and atom layers) and applies different similarity metrics to these layers. The layered approach is motivated by the fact that each layer represents a unique aspect of the semantics of the global element.

As an example of hybrid matching, the LSD system [31] uses machine-learning techniques to match a pair of schemas. The LSD is based on the combination of several match result obtained by independent learners. The predictions of individual learners are combined by a so called meta-learner, which weighs the predictions from a learner according to its accuracy shown during the training phrase.

For structure-level matching, a variety of graph-based metrics have been proposed [9], [10]. Typically, these metrics quantify the commonality between nodes by taking into account the lexical and structural similarities of super and sub-nodes (e.g., ancestors and descendents all the way to leaf nodes).

Because most schemas can be viewed as labeled trees, many matching algorithms have been developed based on either top-down or bottom-up traversal techniques [10]. As an example of the top-down approach, TransScm [32] provides a schema matching method for data translation and conversion based on the syntactic analysis of the structures. The matching is performed node-by-node, considering 1-to-1 matching cardinality in a top-down fashion. Tess [33] is another example of a top-down algorithm, which deals with schema evolution. Tess takes definitions of the old and new types and identifies pairs of types as matching candidates. It then recursively tries to match their substructure in a top-down fashion.

Similarity flooding (SF) [34] provides bottom-up matching based on similarity propagation. This method begins with a string-based comparison of the schema elements and analyzes the structure-level relationships on the assumption that if two nodes from two schemas are similar, then their neighbors may also be somehow similar.

A generic schema matching method, called Cupid [35], was proposed by Microsoft Research [36]. It is comprised of element- and structure-level matching approaches, and it computes the similarity with domain-specific thesauri as the linguistic information resources. The Cupid algorithm provides an effective algorithm to traverse the tree in a combined bottom-up and top-down manner.

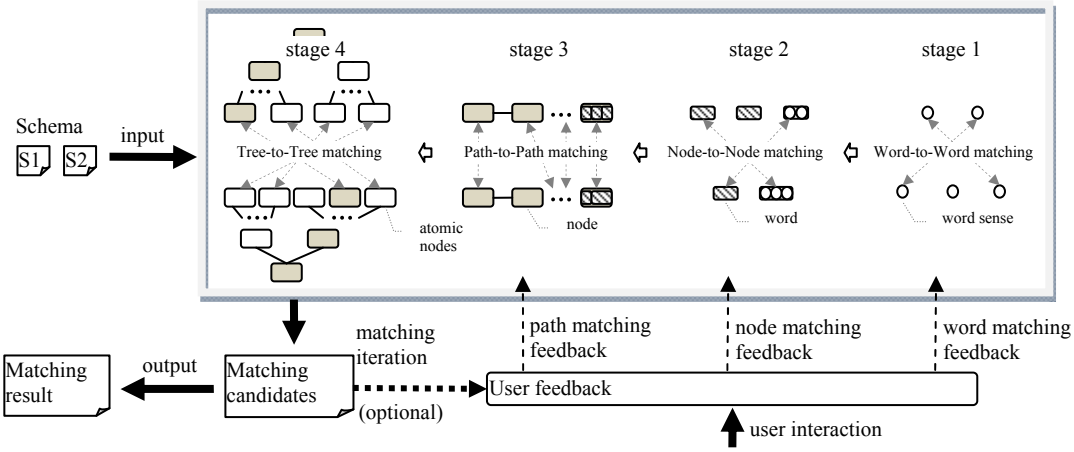


Figure 1. Matching algorithm overview.

Another effective bottom-up method, called S-Match [37], follows a graph-based matching algorithm, which decomposes the tree matching problem into a set of node matching problems. Each node matching problem is translated into a propositional formula, which can then be efficiently resolved using state of the art propositional

satisfiability deciders.

In general, top-down approaches are less expensive but can be misled if the top-level schema structures are very different [10]. On the other hand, bottom-up approaches take a more comprehensive view [9]. However, existing bottom-up methods identify optimal matches in an ad hock manner without solid theoretical foundations.

Another group of graph-based metrics is based on terminological taxonomy that can be applied to 'IS-A' hierarchies such as ontologies. The edge counting approach is well-known traditional approach based on conceptual distance in a taxonomy [38]. The principle of the edge counting is simple and intuitive. It computes the shortest path between two nodes in the taxonomy, presents the most intuitive method to evaluate the semantic similarity in a hierarchical taxonomy. Another taxonomy-based approach, known as bounded path matching [39], takes two paths, with links between classes defined by the hierarchical relationships, compares terms and their positions along these paths, and identifies similar terms.

To identify the matching between different paths, it is common to compare the similarities of nodes that compose these two paths. Work in [40] introduced the concept of the *path context coefficient* (PCC) to capture the degree of similarity for two paths. The algorithm, called LocalMatch, finds the best 1-to-1 matching pair of elements within two path contexts by summing up the linguistic similarities for all of the matched elements. This solution can be inaccurate if there are additional nodes within the path that were not matched.

To find the optimal path similarity, [41] proposes criteria for matching the paths between XML query and documents. To calculate the similarity between two paths, [41] also proposes a similarity score for each criterion and combines them with the given weights. References [42], [43] apply these criteria for the path similarity measure to the schema matching solution. Work in [42] only considers identical string matches between nodes, and [43] investigates further to deal with the string-based similarities between nodes. However, they do not utilize other semantic information such as linguistic resources. In addition, their scoring algorithm only considers adding weight to the higher levels of the tree without considering the differences in importance of nodes.

III. Algorithm

We propose a novel schema-based matching algorithm to solve the combinatorial optimization problem of matching atomic nodes between two schemas. Our approach finds the globally optimal set of pair-wise matchings between *atomic nodes* in a principled manner by mathematical programming. We obtain the semantic similarities between words using WordNet.

A. Matching algorithm overview

The matching algorithm takes two schemas as input and identifies the set of matching pairs of all atomic nodes with the highest semantic similarity among all possible sets of pair-wise matchings. Fig. 1 illustrates the matching process of our approach for two input schemas, *S1* and *S2*.

The algorithm breaks the complex combinatorial optimization problem into four matching stages: tree-to-tree (between the sets of atomic nodes of the two schemas), path-to-path (between the sets of nodes on the paths of two atomic nodes), node-to-node (between sets of words in the labels of two nodes), and word-to-word (between multiple senses of two words) matchings. As can be seen in Fig. 2, each stage works on a bipartite graph, consisting of two sets of vertices and a weight matrix between them, with the objective of finding the 1-to-1 matching between vertices in one set to the other with the highest combined weight. Therefore, we formulate these sub-problems as *maximum-weighted bipartite graph matching* problems [44].

Except for the word-to-word matching at the bottom stage, the weight matrix between edges for each stage is a similarity matrix calculated by the previous stage. For example, the similarity matrix for tree-to-tree matching stage is provided by path-to-path matching stage. The word-to-word matching stage uses WordNet to compute the semantic similarity between two words by identifying the optimal matching pairs for their respective senses.

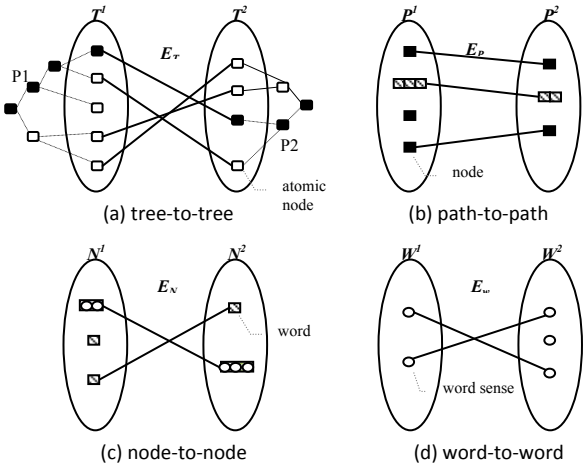


Figure 2. Weighted bipartite graph modeling for different types of nodes in two labeled trees.

Except for the path-to-path matching stage, optimal matching at each stage can be obtained according to the general *Maximum-weighted Bipartite Matching* algorithm (*MBM*) [45]. The path-to-path matching requires an additional *ordering criterion* [41] that path P^1 includes most of the nodes of path P^2 in the correct order as shown in Fig. 2(b), and is called *Ordered Maximum-weighted Bipartite Matching* (*OMBM*) problem. Algorithms for solving the *MBM* and the *OMBM* problems are described in the following sections.

B. Maximum-weighted bipartite matching algorithm

Tree-to-tree, node-to-node, and word-to-word matching stages can be formulated as the general *weighted bipartite graph matching* problems. Let G be a weighted bipartite graph with two sets of vertices, $U = \{u_1, u_2, ..., u_m\}$ and $V = \{v_1, v_2, ..., v_n\}$, and the set of edges E. Edge e_{ij} in the graph connects the vertices u_i and v_j whose weight w_{ij} is given in the weight matrix W. Vertices of the same set are not connected.

A matching M of graph G is a subset of E such that no two

edges in M share a common vertex. In other words, the matching M consists of a set of pair-wise matchings of 1-to-1 cardinality. The *maximum-weighted bipartite matching* is a matching whose sum of the weights of the edges is the highest among all possible sets of pair-wise matchings. The optimal matching M can be found by integer programming defined below:

Maximize: $$\sum_{e_{ij}\in E} w_{ij}x_{ij} \quad (5)$$

subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j = 1,\cdots,|V| \quad , \quad \sum_{j=1}^{n} x_{ij} = 1, \forall i = 1,\cdots,|U| \quad ,$$

$x_{ij} \in \{0,1\}$, where x_{ij} is 1 if $e_{ij} \in M$ and 0 otherwise.

Because integer programming is typically NP-hard (i.e., harder than a nondeterministic polynomial-time problem and for worst case with running time exponential to the problem size) [46], we approximate it by a simple greedy algorithm as follows:

```
algorithm MBM-greedy (U,V,W)
  m=|U|, n=|V|; M = ∅;
  sort W;
  while (|U|>0 and |V|>0)
     Choose vertices u and v connected with an edge e that has the
     highest weight w in the weight matrix W;
     if edges in M share neither u nor v
        then M := M ∪{e}, U := U-{u}, V := V-{v}, wsum := wsum + w;
     W [u,v] := 0;
  sim := 2 * wsum / (m + n);
  return {M, sim};
```

Figure 3. Greedy algorithm for maximum-weighted bipartite matching.

The greedy algorithm simply sorts the weight matrix W in descending order and at each iteration it chooses an edge with the highest weight. The initial weight matrix W is calculated by the previous matching stage. The chosen edge will be the matching candidate if it shares no vertex with edges in M. This process is repeated until there is no vertex to be matched in either U or V. The algorithm returns the optimal matching M and the average weight of all edges in M as the measure of similarity between U and V. In this greedy algorithm, the most expensive step is the sorting of the weight matrix W of size $|U|\times|V|$. We use a quicksort algorithm [47] that takes $O(k\log(k))$ to sort k items. Thus, the complexity of this greedy algorithm is $O(|U||V|\log(|U||V|))$.

C. Ordered maximum-weighted bipartite matching algorithm

Some have suggested using the longest common sequence (LCS) to address the ordering criterion in the path-to-path matching [41]–[43]. However, none of the suggestions utilizes the semantic similarities of the nodes on the two path contexts. To consider semantic similarities of the nodes, we have developed the *ordered maximum-weighted bipartite matching* algorithm based on dynamic programming.

Let G be a weighted bipartite graph with two *ordered* sets of vertices $U = \{u_1, u_2, ..., u_m\}$ and $V = \{v_1, v_2, ..., v_n\}$, and the set of edge E. The core algorithm, *OMBM* (U, V), finds the optimal matching M between U and V by recursively partitioning it into smaller sub-problems until the solution becomes trivial.

For a sequence $S=s_1s_2...s_d$, a sequence shortened from the end is denoted $S_k=s_1s_2...s_k$, where $k \le d$. We call S_k the *prefix* of S. The *prefixes* of U are $U_1, U_2, ..., U_m$, and the *prefixes* of V are $V_1, V_2, ... V_n$. Let *OMBM* (U_i, V_j) be the function that finds the optimal matching of *prefixes* U_i and V_j. This problem can be reduced to three alternative simpler sub-problems with shortened *prefixes* and returns the one with maximum sum of weights:

1) u_i and v_j match each other. Then, the optimal matching for U_i and V_j can be formed by attaching edge e_{ij} to the optimal matching of two shortend sequences U_{i-1} and V_{j-1}, denoted (*OMBM* (U_{i-1}, V_{j-1}), e_{ij}).
2) u_i and v_j do not match each other. Then, either of them can be removed to shorten one of the matching sequences and *OMBM* (U_i, V_j) is reduced to either *OMBM* (U_{i-1}, V_j) or *OMBM* (U_i, V_{j-1}).

Thus *OMBM* (U_i, V_j) can be computed by the following recursive function:

$$OMBM(U_i,V_j) = \begin{cases} \varnothing & \text{if } i=0 \text{ or } j=0 \\ max\begin{pmatrix} OMBM(U_{i-1},V_j), \\ OMBM(U_i,V_{j-1}), \\ (OMBM(U_{i-1},V_{j-1}),e_{ij}) \end{pmatrix} & \text{otherwise} \end{cases} , \quad (6)$$

where the function *max*() returns the optimal matching among the three matchings from the sub-problems based on the similarity scores returned by *OMBM*; it returns empty if either U_i or V_j is reduced to nill ($i = 0$ or $j = 0$). The similarity score calculated by *OMBM* (U_i, V_j), denoted $sim_{OBMB}(U_i, V_j)$, is the average weight of all edges in the matching as follows:

$$sim_{OMBM}(U_i,V_j) = \sum_{e_{ij}\in OMBM(U_i,V_j)} w_{ij} \cdot \frac{2}{i+j} . \quad (7)$$

The optimal matching M of two sets of ordered vertices U and V, $|U| = m$, $|V| = n$, is then computed as:

$$M = OMBM(U,V) = OMBM(U_m,V_n) . \quad (8)$$

The similarity score of M, denoted $sim_{OBMB}(U, V)$, is the average weights of all edges in M:

$$sim_{OMBM}(U,V) = \sum_{e_{ij}\in M} w_{ij} \cdot \frac{2}{|U|+|V|} . \quad (9)$$

The example below shows how the optimal matching and similarity score between two simple path contexts is calculated by (8) and (9).

Example 1: Consider two path contexts *PO / BillTo / Zip* and *PurchaseOrder / Customer / Address / Postal*. Let *P1* and *P2* be the set of nodes on these two paths:

$P1 = \{n_{PO}, n_{BillTo}, n_{Zip}\}$, and

$P2 = \{n_{PurchaseOrder}, n_{Customer}, n_{Address}, n_{Postal}\}$.

Suppose that the similarity scores between nodes are as follows:

$sim^N(n_{PO}, n_{PurchaseOrder}) = 1.0$, $sim^N(n_{BillTo}, n_{Customer}) = 0.6$,

$sim^N(n_{BillTo}, n_{Address}) = 0.4$, $sim^N(n_{Zip}, n_{Postal}) = 1.0$.

The similarities between all other pairs are 0. By (6), (7), and (8), the *OMBM* (*P1*, *P2*) between the two paths returns the optimal matching

$M = \{(n_{PO}, n_{PurchaseOrder}), (n_{BillTo}, n_{Customer}), (n_{Zip}, n_{Postal})\}$.

By (9), the similarity score is

$sim_{OMBM}(P1, P2) = (1.0 + 0.6 + 1.0) \times (2/(3+4)) = 0.74.$

To efficiently execute the algorithm, we use a bottom-up approach [13]. The algorithm is as follows:

```
algorithm OMBM-A (U,V,W)
  for i from 1 to |U|
    for j from 1 to |V|
      A [i,j] := maximum of A[i-1,j], A[i,j-1], and A[i-1,j-1]+W [i,j];
  sim := 2*A[|U|,|V|] / (|U| + |V|);
  return {A, sim};
```

Figure 4. Bottom-up dynamic programming algorithm for ordered maximum-weighted bipartite matching.

This algorithm starts from the simplest matching between U_1 and V_1 and continues to more complex matching problems. The calculated similarity scores for the optimal matchings (average weights by (7)) are stored in a two dimensional array $A[i,j]$ in Fig. 4 for future use to avoid repeated calculations of smaller problems. The weights of W are calculated by the previous matching stage (i.e., node-to-node matching stage). The complexity is only $O(|U||V|)$, i.e., linear to the size of table A.

The following algorithm deals with a simple matter of finding the matching between two sets of nodes U and V that is identified by bottom-up dynamic programming algorithm of Fig. 4.

```
algorithm OMBM (U,V,W)
M = ∅;
{A, sim} := OMBM-A (U,V,W);
  i := |U|, j :=|V|;
  while i > 0 and j > 0
    if A[i,j] equal to A[i-1,j] then i--;
      elseif A[i,j] equal to A[i,j-1] then j--;
        else M := M ∪ {e_ij}, i--, j--;
  return {M, sim};
```

Figure 5. Dynamic programming algorithm for ordered maximum-weighted bipartite matching.

How this bottom-up dynamic programming algorithm works is illustrated below using the same example defined in Example 1.

Example 2: Consider *P1* and *P2* for the two path contexts defined in Example 1. The weight matrix W is initialized by the similarity scores between nodes as shown below:

TABLE I. AN EXAMPLE OF WEIGHT MATRIX

W		3	2	1
		n_{PO}	n_{BillTo}	n_{Zip}
4	$n_{PurchaseOrder}$	1.0	0.0	0.0
3	$n_{Customer}$	0.0	0.6	0.0
2	$n_{Address}$	0.0	0.4	0.0
1	n_{Postal}	0.0	0.0	1.0

The array $A[]$ for calculating the matching similarity scores can be represented as follows.

TABLE II. AN EXAMPLE OF MATCHING SIMILARITY SCORE TABLE FOR BOTTOM-UP APPROACH

A	3	2	1	0
4	1.0+1.6=2.6	1.6	1.0	0.0
3	1.6	0.6+1.0=1.6	1.0	0.0
2	1.6	0.4+1.0=1.4	1.0	0.0
1	1.0	1.0	1.0	0.0
0	0.0	0.0	0.0	0.0

Note that values in the array are not normalized. According to the algorithm *OMBM-A* () in Fig. 4, the value of $A[i,j]$ is obtained from the maximum of the three values: $A[i-1,j]$, $A[i,j-1]$, and $A[i-1,j-1] + W[i,j]$, where $1 \le i \le 3$ and $1 \le j \le 4$. The initial values of $A[]$ set to zero.

The calculation starts from the simplest matching array $A[1,1]$. The algorithm compares three values: $A[1,0] = 0$, $A[0,1] = 0$, and $A[0,0] + W[1,1] = 1.0$ and the maximum score 1.0 is chosen. To find the optimal matching by the *ordered maximum weighted bipartite matching algorithm*, look at the first entry $A[3,4]$. It is calculated by the maximum value among three matching scores: $A[2,4] = 1.6$, $A[3,3] = 1.6$, and $A[2,3] + W[3,4] = 1.6 + 1.0 = 2.6$. The maximum value is 2.6, telling us the normalized similarity by the average length of two paths is $2.6 \times (2/(3+4)) = 0.74$, which is the same as what was calculated in Example 1.

According to algorithm *OMBM* () in Fig. 5, the optimal matching result can be obtained by following the traces to reach the first entity $A[3,4]$. As highlighted in Table II, the entities used to calculate $A[3,4]$ are $A[3,4]$, $A[2,3]$, $A[1,2]$, and $A[1,1]$. Then, the entities added their similarity scores are selected as matching: $A[3,4]$, $A[2,3]$, and $A[1,1]$, which lead to the optimal matching

$$M = \{(n_{PO}, n_{PurchaseOrder}), (n_{BillTo}, n_{Customer}), (n_{Zip}, n_{Postal})\}.$$

Algorithm *OMBM-A* and *OMBM* are further enhanced by considering the differences in importance for the individual nodes measured by their information contents. We collect each node's frequency-of-occurrence in the schema trees and compute the information contents by (1). Fig. 6 shows the modified algorithm of *OMBM-A*.

```
algorithm OMBM-A-IC (U,V,W)
  for i from 1 to |U|
    ic-sum := ic-sum + ic(u_i);
  for j from 1 to |V|
    ic-sum := ic-sum + ic(v_j);
  for i from 1 to |U|
    for j from 1 to |V|
      ic_w = W [u_i, v_j]*(ic(u_i)+ic(v_j));
      A[i,j] := maximum of A[i-1,j], A[i,j-1], and A[i-1,j-1] + ic_w;
  sim := A[|U|,|V|]/ ic-sum;
  return {A, sim};
```

Figure 6. Algorithm enhanced by information contents.

Algorithm *OMBM-A-IC* gives more weights to higher-level nodes because lower-level nodes are typically generic entities that appear widely as the descendants of the higher-level nodes. In addition, it also considers the differences in importance of nodes at the same level. The complexity of this algorithm is still $O(|U||V|)$.

D. Overall schema matching algorithm

Fig. 7 gives the algorithm for overall schema matching and how each stage obtains the weight matrix by calling the optimization algorithm for the previous stage.

The algorithm views matching two schema trees as matching two sets of atomic nodes with their respective path-contexts. Each path consists of a sequence of nodes along the path from the root to the leaf of the schema tree. Each node represents either an element or an attribute named

by a label of English word or concatenation of words or their abbreviations. To compute semantics similarities between words, we analyze optimal pair-wised matchings between multiple senses of two words.

```
algorithm T2T-matching (T1, T2)
   for i from 1 to |T1|
      for j from 1 to |T2|
         t2t-smatix[i,j] :=
            P2P-matching (path of T1's i_th atom, path of T2's j_th atom);
   return MBM-greedy (T1's atoms, T2's atoms, t2t-smatix);

algorithm P2P-matching (P1, P2)
   for i from 1 to |P1|
      for j from 1 to |P2|
         p2p-smatix[i,j] := N2N-matching (P1's i_th node, P2's j_th node);
   return OMWM-IC (P1's nodes, P2's nodes, p2p-smatix);

algorithm N2N-matching (N1, N2)
   for i from 1 to |N1|
      for j from 1 to |N2|
         n2n-smatix[i,j] := W2W-matching (N1's i_th word, N2's j_th word);
   return MWM-greedy (N1's words, N2's words, n2n-smatix);

algorithm W2W-matching (W1, W2)
   if wordnet definitions for W1 and W2 exists then
      for i from 1 to |W1|
         for j from 1 to |W2|
            w2w-smatix[i,j] := word-sense-sim (W1's i_th sense, W2's j_th
            sense);
      return MWM-greedy (W1's senses, W2's senses, w2w-smatix);
   else
```

Figure 7. Overall schema matching algorithm.

The word-to-word matching algorithm uses two semantic similarity measure functions: *word-sense-sim*() based on WordNet taxonomy and *word-desc-sim*() based on textual description. In WordNet, nouns are organized into taxonomies in which each node has a set of synonyms (a synset), each of which representing a single sense [24]. If a word has multiple senses, it will appear in multiple synsets at various locations in the taxonomy. To compute the semantic similarity between two words (two sets of senses), we use the *MBM-greedy* () algorithm with the input of two set of senses for words *W1* and *W2*, respectively, and the similarities between the senses are calculated by (4).

If a word does not exist in WordNet, we extract the textual description of a given word from the internet and then use string-similarity measures, such as the cosine similarity [18], to calculate the similarity between the two textual descriptions of the two words

IV. Experiments and Results

We have implemented a prototype system of our approach based on Java and Java WordNet Library (JWNL) [48] for experimental validation. In the experiments, we used five real world XML schemas for purchase orders (i.e., CIDX, Apertum, Excel, Norris, and Paragon) from [49], [50]. Table III summarizes the characteristics of those XML schemas.

TABLE III. Characteristics of PO XML Schemas

Schemas	CIDX	Apertum	Excel	Norris	Paragon
max depth	4	5	4	4	6
# nodes	40	145	55	65	80
# leaf nodes	33	116	42	54	68

In the experiment, as suggested in [50], we compute the tree-to-tree similarity of the ten pairs of the five XML schemas. Then for each schema, we accept a matching to any of the other four if the similarity score is above a fixed threshold 0.6. To evaluate the quality of our match result, we used several performance metrics including Precision, Recall, F-measure, and Overall [50], [51], against the results from manual matching [50]. These measures are then compared with the performances of other approaches for the same setting [35], [43], [50]. Note that the Overall measure, proposed by [50] to estimate the post-match efforts, varies in [-1,1] and other three vary in [0,1].

Precision, Recall, F-measure, and Overall for our results are 0.85, 0.85, 0.85, and 0.69. To increase the precision, we used a relative threshold which is chosen as the similarity of the matching with the largest gap to the next best matching, among matching candidates with similarities ranging from 0.5 to 0.6. Fig. 8 shows the performance analysis of the matching result that our solution produced.

The experimental results show that our matching performances of average Precision, Recall, F-measure, and Overall are 0.93, 0.83, 0.88, and 0.77, respectively. Comparing to the previous results that use a fixed threshold, the Recall is slightly decreased while the Precision is significantly increased. The relative threshold also helps to increase F-measure and Overall. For comparison purposes, the average scores of performance metrics by some other methods are given in Fig. 9.

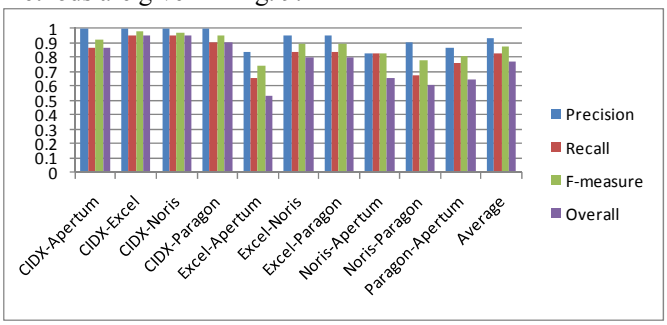
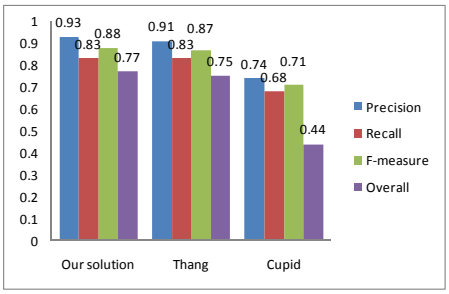
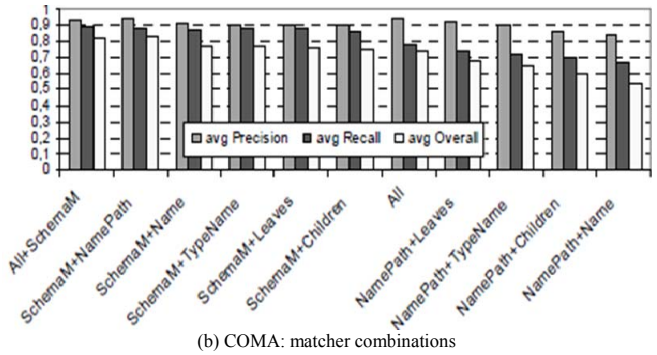


Figure 8. Performance analysis.



(a) Thang and Cupid



(b) COMA: matcher combinations

Figure 9. Performance analysis.

The first comparison, as illustrated in Fig. 9(a), is with Thang [43] who proposed an XML schema matching solution that combines linguistic, data type, and path-context similarity measures. He also implemented the Cupid [35] algorithm for comparison purpose. We compared our result to both algorithms. In general, all performance metrics of our approach are slightly better than Thang's and significantly better than Cupid's.

The second comparison is with COMA (COmbination MAtch) [50], which used various ways for combining different matchers. Because COMA only provides performance graphs without the specific scores as shown in Fig. 9(b), it is difficult to compare the performances with our result precisely. However, comparison between Fig. 8 and Fig. 9(b) shows that our result is, in general, at least equal to or slightly better than COMA's results even if some of their matchers used the manual matching called *SchemaM* [50].

V. Conclusions

In this paper, we have described a solution to identify semantic-based optimal XML schema matching using mathematical programming. This solution identifies the optimal matching between two XML schemas on the assumption that the tree-to-tree matching problem can be globally optimized by reducing it to simpler problems, such as path-to-path, node-to-node, and word-to-word matching. We have implemented a prototype system for our solution and conducted the experiments with actual industry XML schemas. We compared our result to some other XML schema matching approaches. The results were encouraging. The average matching performances of Precision, Recall, F-measure, and Overall were 0.93, 0.83, 0.88, and 0.77, which are better than or at least equal to other approaches'.

Although our approach primarily targets the XML schema matching problem, the solution can be applied to other matching problems - such as XML instance matching if the instances can be represented as labeled trees. Our solution is limited to the assumptions that only 1-to-1 matching cardinality is considered and that schema designers correctly use the English terminologies when labeling the elements/attributes in the schemas. These limitations call for further research. Other directions of research include methods to improve the performance by utilizing domain specific terminology and taxonomy, ontologies with formally defined concept semantics, and user feedback.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by National Institute of Standards and Technology (NIST), nor does it imply that these products are necessarily the best available for the purpose.

References

[1] W3.org "Extensible Markup Language (XML) 1.1 specification". Available: http://www.w3.org/TR/xml11/
[2] W3.org, "XML schema 1.1 specification". Available: http://www.w3.org/TR/xmlschema11-1/
[3] R. Kalakota, and M. Robinson, E-business: roadmap for Success, Addison-Wesley, Reading, MA, 1999.
[4] J.M. Nurmilaakso and P. Kotinurmi, "A review of XML-based supply-chain integration," Production Planning and Control, vol. 15, no. 6, Sep. 2004, pp. 608–621, doi: 10.1080/09537280412331283937.
[5] R. Skinstad, "Business Process Integration through XML", Netfish Technologies, 2000. Available: http://www.infoloom.com/gcaconfs/WEB/paris2000/S10-03.HTM
[6] S.Y. Shim, V.S. Pendyala, M. Sundaram, and J.Z. Gao, "Business-to-business e-commerce frameworks," IEEE Computer, vol. 33, no. 10, Oct. 2000, pp. 40-47, doi: 10.1109/2.876291.
[7] B. Medjahed, B. Benatallah, A. Bouguettaya, A. Ngu, and A. Elmagarmid, "Business-to-business interactions: issues and enabling technologies," VLDB Journal, vol. 12, no. 1, May 2003, pp. 59–85, doi: 10.1007/s00778-003-0087-z.
[8] C. Bussler, "B2B protocol standards and their role in semantic B2B integration engines," Bull Tech Comm Data Eng, vol. 24, no. 1, 2001, pp. 3–11.
[9] P. Shvaiko, and J. Euzenat, "A survey of schema-based matching approaches," Journal on Data Semantics IV, LNCS 3730, 2005, pp. 146-171, doi: 10.1007/11603412_5.
[10] E. Rahm and P.A. Bernstein, "A survey of approaches to automatic schema matching," VLDB Journal, vol. 10, no. 4, 2001, pp. 334-350, doi: 10.1007/s007780100057.
[11] A. Gal, "Why is schema matching tough and what can we do about it?," ACM Sigmod Record, vol. 35, no. 4, 2006, pp. 2-5. doi: 10.1145/1228268.1228269.
[12] B. Meadows, and L. Seaburg, "Universal Business Language (UBL) 1.0," 2004. Available: http://docs.oasis-open.org/ubl/cd-UBL-1.0/
[13] K.H. Elster, Modern mathematical methods of optimization, Vch Pub. 1993, ISBN 3-05-501452-9.
[14] R.E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957, Republished 2003: Dover, ISBN 0486428095.
[15] R.W. Hamming, "Error detecting and error correcting codes," Bell System Technical Journal, vol. 29, no. 2, 1950, pp. 147–160, MR0035935.
[16] V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals PDF," Soviet Physics Doklady, vol. 10, no. 8, 1966, pp. 707–710.
[17] C.J. Van Rijsbergen, Information retrieval (2nd ed)., London: Butterworths, 1979.
[18] H.A. Sneath, "Then application of computers to taxonomy," Journal of General Microbiology, vol. 17, no. 1, 1957, pp. 201-226, doi: 10.1099/00221287-17-1-201.
[19] Euclid, Euclid's Elements, Sir Thomas Little Heath, New York, Dover, 1956.
[20] J. R. Ullmann, "A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words," The Computer Journal, vol. 20, no. 2, 1977, pp. 141–147.
[21] E. Ukkonen, "Approximate string matching with q-grams and maximal matches," Theoretical Computer Science, vol. 92, no. 1, 1992, pp. 191-211.
[22] E. Sutinen and J. Tarhio, "On using q-gram locations in approximate string matching," In Proceedings of Third Annual European Symposium on Algorithms (ESA'95), 1995, pp. 327–340.
[23] B. Jeong, "Machine learning-based semantic similarity measures to assist discovery and reuse of data exchange XML schemas," Ph.D. thesis, Department of Industrial and Management Engineering, Pohang University of Science and Technology, June 2006.

[24] G.A. Miller, "WORDNET: a lexical database for English," Communications of ACM, vol. 38, no. 11, 1995, pp. 39-41. doi: 10.1145/219717.219748.
[25] P. Qin, Z. Lu, Y. Yan, and F. Wu, "A new measure of word semantic similarity based on WordNet hierarchy and DAG theory," In Proceedings of International Conference on Web Information Systems and Mining, 2009, pp. 181-185, doi: 10.1109/WISM.2009.44.
[26] D. Yang, and D.M.W. Powers, "Measuring semantic similarity in the taxonomy of WordNet," In Proceedings of the 28th Australasian Computer Science Conference, 2005, pp. 315-322, doi: 10.1.1.87.678.
[27] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," In Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 448-453, doi: 10.1.1.55.5277.
[28] D. Lin, "An Information-theoretic definition of similarity," In Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 296-304. doi: 10.1.1.55.1832.
[29] T.M. Cover and J.A. Thomas, Elements of information theory, Wiley series in telecommunications and signal processing. Wiley, New York, 1991, ISBN 0-471-06259-6.
[30] J. Kim, Y. Peng, B. Kulvatunyou, N. Ivezik, and A. Jones, "A layered approach to semantic similarity analysis of XML schemas," in Proceedings of the IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, 2008, pp. 274 - 279.
[31] A. Doan, P. Domingos, and A. Halevy, "Reconciling schemas of disparate data sources: A machine-learning approach," In Proceedings of the International Conference on Management of Data (SIGMOD), 2001, pp. 509–520.
[32] T. Milo and S. Zohar, "Using schema matching to simplify heterogeneous data translation," In Proceedings of the 24th International Conference on Very Large Data Bases, 1998, pp. 122-133. doi: 10.1.1.30.2620.
[33] B.S. Lerner, "A model for compound type changes encountered in schema evolution," ACM Transactions on Database Systems, vol. 25, no. 1, 2000, pp. 83-127, doi: 10.1.1.105.1542.
[34] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding - a versatile graph matching algorithm," In Proceedings of 18th International Conference of Data Engineering, 2002, pp. 117-128. doi: 10.1.1.61.4266.
[35] J. Madhavan, P.A. Bernstein, and E. Rahm, "Generic schema matching with Cupid," In Proceedings of the 27th International Conference on Very Large Data Bases, 2001, pp. 49-58. doi: 10.1.1.17.4650.
[36] Microsoft Research, Microsoft Corporation, Available: http://www.research.microsoft.com/
[37] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-Match: an algorithm and an implementation of semantic matching," In Proceedings of the European Semantic Web Symposium (ESWS), 2004, pp. 61–75, doi: 10.1007/978-3-540-25956-5_5.
[38] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner, "Development and application of a metric on semantic nets," IEEE Transaction on Systems, Man, and Cybernetics, February 1989, pp. 17-30.
[39] N. Noy, and M. Musen, "Anchor-PROMPT: using non-local context for semantic matching," In Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI), 2001, pp. 63–70.
[40] L.L. Mong, Y.H. Liang, H. Wynne, Y. Xia, "XClust: clustering XML schemas for effective integration," In Proceedings in 11th ACM International Conference on Information and Knowledge Management (CIKM), McLean, Virginia, November 2002, doi: 10.1145/584792.584841.
[41] D. Carmel, Y. Maarek, Y. Mass, N. Efraty, and G. Landau, "An extension of the vector space model for querying XML documents via XML fragments," in ACM SIGIR 2002 Workshop on XML and Information Retrieval, Tampere, Finland, August 2002.
[42] A. Boukottaya and C. Vanoirbeek, "Schema matching for transforming structured documents," In DocEng, 2005, pp. 2-4, doi: 10.1145/1096601.1096629.
[43] H.O. Thang, V.S. Nam, "XML schema automatic matching solution," International Journal of Computer Systems Science and Engineering, vol. 4, no. 1, 2008, pp. 68-74.
[44] A.L. Dulmage and N.S. Mendelsohn, "Coverings of bipartite graphs," Canadian Journal of Mathematics, vol. 10, 1958, pp. 517–534.
[45] W.B. Douglas, Introduction to Graph Theory (2nd ed.), Prentice Hall, Chapter 3, 1999, ISBN 0-13-014400-2.
[46] C. H. Papadimitriou, "On the complexity of integer programming," J. ACM, vol. 28, 1981, pp. 765–768.
[47] C.A.R. Hoare, "Quicksort," Computer Journal, vol. 5. no. 1, 1962, pp. 10-15.
[48] Java WordNet Library (JWNL), Available: http://sourceforge.net/ apps/mediawiki/jwordnet
[49] BizTalk Server, Available: http://www.microsoft.com/biztalk/
[50] D. Aum¨uller, H.H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with COMA++," In Proceedings of the International Conference on Management of Data (SIGMOD), Software Demonstration, 2005.
[51] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, "Performance measures for information extraction," In Proceedings of DARPA Broadcast News Workshop, Herndon, VA, February 1999.