NAME SIMILARITY MEASURES FOR XML SCHEMA MATCHING

Ali El Desoukey  
Mansoura University, Mansoura, Egypt  
Amany Sarhan, Alsayed Algergawy  
Tanta University, Tanta, Egypt  
Seham Moawed  
Middle Delta Company for Electricity, Mansoura, Egypt  
Seham_25jan@yahoo.com

ABSTRACT

Schema matching plays a central role in identifying the semantic correspondences across shared data applications, such as data integration etc. One of the great challenges in the field of XML schema matching is the definition, adoption, and utilization of element similarity measures. We classify these measures either as internal or external. The internal element measures are based on element features while external measures rely heavily on the element context. The element name is considered as the most important feature of XML schema elements and a vital source of semantic information for schema matching. Therefore, we aim in this study to concentrate on the name similarity measures as an initial step towards building efficient and effective matchers. We strive to validate the effect of existing string functions and their combinations on schema matching performance. The experimental evaluation shows good and encourage results towards building robust schema matchers utilizing different schemas.

Keywords: XML Schema, Similarity measures, Syntactic Matching

1 INTRODUCTION

Nowadays XML becomes a popular standard for effective data presentation and interchange through Web. This is due to XML’s inherent data self-describing capability and the flexibility of organizing data [1]. This helps applications on the Web understand the content of XML documents published by other applications [2]. As the demand of using XML is increasing, a huge amount of XML schemas also has been created [3]. Consequently, it is urgent to develop good information integration technology for the growing number of XML data sources. Identifying and discovering semantic correspondences among heterogeneous data sources is the biggest obstacle for developing such an integrated schema.

The demand of schema matching is great in a diverse number of data application scenarios: in data integration, to identify and characterize inter-schema relationships across multiple heterogeneous schemas; in data warehousing, to map data sources to a warehouse schema; in E-business, to help to map messages between different XML formats; in the Semantic Web, to establish semantic correspondences between concepts of different ontologies [4]; in data migration, to migrate legacy data from multiple sources into a new one [5]; and in XML data clustering, to determine semantic similarities between XML data [6].

To cope with XML schema matching, a lot of heterogeneities should be resolved. Such heterogeneities increase the difficulties of the schema matching process [7]. Some of these heterogeneities are as follows: Representational problems involve two kinds of heterogeneity, (i) the same domain schemas are designed in different possible representation models and (ii) same concepts are represented using different names and structures (synonyms). Semantic problems, indeed extracting semantics information from data sources is often extremely bulky. Besides, either schemas or data clues, the basis of the matching process, are often unreliable and incomplete. Computational cost problems, we have to examine each element in one schema against all other elements in the other schema. This global nature adds substantial cost to the matching process. Due to this complexity, manual schema matching becomes expensive, extremely tedious, and error prone. Therefore, efforts are vested in the development of automated schema matching systems. Though it is impossible to fully automate schema matching, myriad of matching algorithms have been proposed and many systems
for semi automatic schema matching have been developed that bring important improvements to the schema matching process such as Cupid [8], Similarity Flooding (SF) [9], COMA/COMA++ [10,11], LSD [12], BTreeMatch [13], OntoBuilder[14], S-Match [15] and XPruM System [7].

In order to resolve the mentioned heterogeneities in XML schema matching, a lot of element similarity measures have been proposed. We classify them into two main kinds; internal and external. Internal element measures exploit the element features, such as names, data types, constraints, etc. to compare elements from different schemas. On the other hand, external measures make use of structural properties which are extracted from the context within which the element resides. These include the number of child or sibling elements, depth of the element within the schema, properties of the path leading from the root to the element in concern, path cardinalities, and referential and identity constraints [7].

Element names represent an important source for assessing similarity between schema elements. This can be done syntactically by comparing the name strings or semantically by comparing their meanings. To the best of our knowledge, no study has been done to evaluate name similarity measures for XML schemas independent of their matching systems. The work presented in the paper applies a set of syntactic measures on XML schemas based on the use of an open source tool Java toolkit of name-matching methods [25] which includes a variety of different techniques. To validate the performance of the name similarity measures, we conducted a set of experiments to gain more knowledge about the most appropriate element name similarity measure which can be applied for XML schema matching.

The rest of the paper is organized as follows: Section 2 presents general schema matching steps. We describe different string metrics to measure the similarity between schema element names in Section 3. Section 4 shows how to combine the individual similarity values resulted from the various name matching techniques. Section 5 presents the experimental evaluation Section 6 concludes the paper.

2 SCHEMA MATCHING STEPS

In this section we present the steps of a generic schema matching process. A first step in almost all schema matching systems is to transform input schemas into a common data model in order to apply the matching algorithm. Most current schema matching approaches use graph or tree data structure. After representing input schemas as schema graphs/trees, the second step is to identify elements from different data sources that have to be matched. The third step, matching phase, is the most important step in which matching algorithms are applied to identified elements. Our study is devoted towards this step. Finally, the set of identified correspondences should be transformed into the application domain.

The matching step, as shown in Figure 1, has three main phases as follows:

1) Element Matcher: The similarity between each pair of schema elements is computed using various hints with each hint implemented as a single element matcher. E.g., one might compare element names, another compares data types, and so forth. Given two schemas S1 and S2 with n and m elements respectively, K element matchers, the output from this stage is a K×n×m similarity cube.

2) Similarity Combiner: It calculates a single similarity value for each pair of schema elements. This value is computed by combining the similarity values by different element matchers. The result is a n×m similarity matrix.

3) Similarity selector: It selects the most plausible corresponding elements based on a set of selection criteria.

![Figure 1: Match step phases](image)

The element matcher phase is mainly based on schema element names. To this context, the rest of the section is devoted to present an overview of string–based measures in some of recent matching systems.

In Cupid system [8], element features are used by a linguistic matcher which exploits name and data type information. The linguistic matching goes through three phases; normalization, categorization and comparison. Similarity Flooding [9] uses a String Match operator to compute the initial node name similarity. The StringMatch operator compares common prefixes and suffixes in words. This is an atomic, local, and schema based hint. The output of this computation is used in the next hint, the iterative fixed point computation in a similarity propagation graph. This hint exploits the structural context of nodes.

COMA++ (combining matching algorithms) [10, 11] exploit different kinds of element properties and use different kinds of matching algorithms. Its matcher library consists of simple matchers, hybrid matchers and auxiliary information. In COMA, four simple approximate string matchers have been
implemented; Affix, n-gram, EditDistance and Soundex. COMA++ supports two hybrid element-level matchers, Name and TypeName, and three hybrid structural matchers, NamePath, Children and Leaves.

LSD applies different types of base learners [12] which train on their own training examples from the extracted data to construct an internal classification model that helps it match new examples such as Name Matcher, Content Matcher and Naïve-Bayes Learners. The Name Matcher matches an XML element using its tag name (expanded with synonyms and all tag names leading to this element from the root element). The name matcher stores all training examples on the form (tag-name,label).

GLUE is an extension version of LSD which is devoted to discover semantic matching between ontologies [17]. Both are based on machine learning techniques for individual matchers and an automatic combination of match results. GLUE has two base learners, Content Learner and Name Learner and a meta learner that is a linear combination of the base learners. The Name Learner makes predictions using the full name of the input instance instead of its content. The full name of an instance is the concatenation of names leading from the root of the taxonomy to that instance.

iMap [18] aims at detecting complex mappings between schemas. It consists of three main modules: match generator, similarity estimator, and match selector. Since the accuracy reported by the searchers in match generator module may not be very accurate, there has to be the similarity estimator. It employs two evaluator modules, each of which exploits a specific type of information to suggest a score, and then combines the suggested scores into final one. These modules called name-based evaluator and Naive Bayes evaluator. The name-based evaluator computes a score for a match candidate based on the similarity of its name to the name of the target attribute. The name of a match candidate is the concatenation of the names of the attributes appearing in that candidate, together with the names of the tables that contain the attributes.

In XPruM System [7], schemas are parsed using a SAX parser and represented internally as schema trees. Then using the Prufer encoding method, it extracts label sequences and number sequences. Then the schema matching part discovers the set of matches between two schemas utilizing both sequences. First, a degree of of linguistic similarity is automatically computed for all element pairs using linguistic matcher phase. The aim of this phase is to obtain an initial similarity value between the nodes of the two schema trees based on the similarity of their labels. It applies three different kinds of string measures; Levenshtein distance, JaroSimilarity and N-gram distance.

3 NAME SIMILARITY MEASURES

The most important component to achieve the task of schema matching is the definition, adoption, and utilization of element similarity measures. Through this section, we introduce how we can make use of the various element name similarity measures.

Due to the wider popularity and usage of XSD (XML Schema Definition) as a descriptive language for the structure and the legal building blocks in XML documents, we only consider XSD schemas. To describe the operations of our study, we use two pairs of schemas extracted from the literature [22, 23], one pair of them is shown in figure 2(a, b). An XML schema can be modeled as a graph. It can also be represented as a tree by dealing with nesting and repetition problems using a set of predefined transformation rules [19]. Here, the schemas are first parsed using XSOOM parser and then represented internally using rooted ordered labeled trees, called schema trees.

In order to compare names of schema elements, we extracted them from each corresponding schema trees and we represent them as a set of label sequences (LS). Considering schema tree ST1 shown in fig. 2(a), the LS (ST1).name = (UnderGrad Courses, GradCourses, Name, Degree, AssistantProfessor, AssociateProfessor, Professor, Faculty, Staff, People, CSDeptUS). The various name similarity measures used through our work operate on the LS sequential representation of schema trees to discover syntactic correspondences between them.

Figure 2: Computer Science department
It should be noted that elements’ names can be syntactically similar or semantically similar. However, in this paper we consider the set of name similarity measures that syntactically determine the similarity between element names. These name measures determine the similarity based on the string representation of names (String-based measures). String similarity measures are divided into three categories: character-based, token-based and hybrid [20]. In the case of the first two, the similarity is calculated on character and token level respectively. In the case of hybrid measures, the similarity is first calculated on the character level, and then the obtained scores are used by a token based metrics. So, first, we need to normalize schema element names in the case of string measures that do not own self based-tokenized implementation. Normalization can go through three phases, (1) Tokenization: consists of segmenting strings into sequences of tokens by a tokenizer, which recognizes punctuation, cases, blank characters, digits, etc. E.g. UnderGrad Courses → {Under, Graduate, Courses}, (2) Expansion: Abbreviation and acronyms are expanded. E.g. Grad → Graduate, (3) Elimination: Tokens that are neither letters nor digits are eliminated and ignored during the comparison process. After decomposing each element name into a set of tokens, the name similarity between the two sets of name tokens T1 and T2 is determined as the average best similarity of each token with all tokens in the other set [7].

A brief description of some of the element name similarity measures is introduced and is given in [24]. Character-based E.g. (1) Levenshtein distance: it defines the minimal number of insertion, deletion, and substitution of characters to make two strings equal; (2) Jaro: the comparison is based on the number and order of the characters common to two strings. Jaro main steps are to compute the string lengths, to find the number of common characters in the two strings, and to find the number of transpositions. A variant of this measure from Winkler uses the length of the longest common prefix of the two strings; (3) N-gram distance: it is based on counting the number of common n-grams between two strings. A window of length n is slid over the characters of a string to create a number of ‘n’ length grams for finding a match. The ‘n’ length grams are then rated as number of n-gram matches within the second string over possible n-grams. When two strings are within a small edit distance of each other, they share a large numbers of n-grams in common. Token-based E.g. (1) Jaccard Similarity, the Jaccard similarity coefficient is a statistical measure of similarity between sample sets. For two sets, it is defined as the cardinality of their intersection divided by the cardinality of their union ;( 2) Overlap Coefficient: is a metric that determines to what degree one string is a substring of another. Hybrid E.g. Monge Elkan: this measure is a general text string comparison method based on an internal character-based similarity measure (e.g. edit distance) combined with a token level (i.e. word level) similarity measure.

4 COMBINING ELEMENT NAME SIMILARITY MEASURES

The similarity combiner computes a single similarity value for every pair of compared schema elements, by combining similarity values which are computed by the individual name matchers. The combining process is far from being trivial. Without a proper means of combining, the element similarity measures fail to produce correct correspondences across schema elements. Commonly, similarity combining can be done using an aggregation function that can be defined in [21]. In our work, we use a set of different combination strategies to combine the results from different matchers:

- Max. This strategy returns the maximal similarity value of any matcher. It is optimistic, in particular in case of contradicting similarity values.
- Min. This strategy shows the lowest similarity value of any matcher. As opposed to Max, it is pessimistic.
- Product. The product of the values obtained from the individual element similarly measures.
- Geometric mean. This method calculates the geometric mean for the different values of individual matchers.
- Weighted-sum. This strategy determines a weighted sum of similarity values of the individual matchers and needs relative weights which should correspond to the expected importance of the matchers.
- Average. This calculates the average similarity of all measures, i.e. considers them equally important. It is a special case of weighted sum.

Unfortunately, all the above aggregation functions methods focus on linear combination functions, which can not sufficiently explore the interdependencies between different element measures [7]. Moreover, these fixed linear methods do not consider the special features of each schema pair when combining similarities. To this context, we also evaluate the effect of the non linear methods as it takes the interdependencies between element measures into consideration.

- Nonlinear methods. Since the similarity values are ranging between 0 and 1, so the
similarity combining function should be restricted to the second order [7].

The result from the combination process using any aggregation function is a match similarity matrix as shown in figure 1. Then an element selector is used to select matches from the given matrix. Several methods can be used but the simplest selection strategy is thresholding which we use in our implementation. It means that we return the matches with a similarity value exceeding a given threshold.

5 EXPERIMENTAL RESULTS

This section concerns with the experimental evaluation to test the performance of a myriad of name matching techniques in the XML schema matching field. We make use of an open source Java-toolkit of string-matching methods called symmetrics.jar. We experimented with two pair of schemas given in the literature [22, 23], one of them is given in Fig 2(a, b). Our aim is to discover the most appropriate name similarity measures which improve the system performance.

To measure the effectiveness of the matching result, we use the same measures used in the literature, including precision, recall and F-measure [7]. Precision evaluates the post-match effort needed to remove false positives, while recall evaluates the post-match effort needed to add true negatives from the final match result. So, neither precision nor recall alone can accurately assess the match quality. Hence, it is necessary to consider a trade-off between them. F-measure is one of the best measures used to combine precision and recall. It is determined as the weighted harmonic mean of precision and recall. It combines the effort of adding false negatives and the effort to remove false positives.

We conducted an intensive set of experiments to test both the performance of single name similarity measures and their combinations. In each test, the similarity values between schema tree elements are first computed. Then, the values are ranked and the ones that are higher than a predefined threshold are selected.

First, we tested with twenty single element name similarity measures and calculate the effectiveness of the matching. We ran these sets of experiments at different values of thresholds in ranges from 0.1 to 0.7. The measures are ranked according to threshold 0.4. This threshold is selected so that the similarity values between schema elements are appreciated. Results are reported in Figures 3 and 4.

Figure 3 represents the matching quality of different name similarity measures at different thresholds for data source 1 (DS1), while Figure 4 shows the results of data source 2 (DS2). In each case, we divided the matching quality of name similarity measures into two separate figures (to be readable). Figures 3(a, b) show that the best F-measure is 0.61 for DS1 using Levenstein distance measure at threshold 0.4. The best F-measure for DS2 is 0.75 using the Jaccard measure at the same threshold. It is known that the strings of element names differ in the way they are seen. Therefore, using a single name similarity measure is not sufficient to be used. Consequently, we test with different strategies of combining the name similarity measures. Of course it is difficult to test with 20 measures used in the first test. We selected the best three measures at each case. In the first case, we use Levenstein distance, N-gram distance of size 3 (i.e. tri-gram) and Cosine similarity while in the second case, we use Jaccard similarity, Dice similarity and Cosine similarity. Results of these set of experiments are reported in Figures 5 and 6.
The matching quality of data source 1 (DS) is represented in Figure 5, while Figure 6 gives the matching quality of data source 2 (DS2). A set of combining strategies has been used, and the matching quality (F-measure) has been evaluated and computed at different threshold values ranging from 0.1 to 0.7. Results shown in Figure 5 show that even if combining strategies are used to improve the matching quality, however, the best F-measure remains with the value of 0.61 at the selected threshold 0.4. Finally, from the figure, max, min, weighted-sum, average, and nonlinear combining strategies are the best among the others. Also, as shown in figure 6, using combining strategies results in a matching quality for data source 2 with value of 0.75. Max, min, Average, product, weighted sum and non linear method achieves the same value of effectiveness at threshold 0.5. This gives an indication that the tested two cases of XML schemas contain strong name heterogeneities and the string-based measures alone failed to discover correct correspondences. Furthermore, it gives us the motivation to use and exploit other features and structural properties of schema elements to correctly discover correspondences.

6 CONCLUSION

With the emergence of XML as a standard for information representation, analysis and exchange on the web, the need to develop high performance techniques to identify and discover semantic correspondences across XML data is always in great demand. The most dominant step is to define, adopt, and utilize similarity measures between XML data elements. In this paper, we study the effect of various name similarity measures and their combinations on the matching quality. To this end, we presented a set of well-known name similarity measures and their combinations. To validate the matching quality, we conducted a set of experimental evaluation. The results give which name measures and combining strategies can be used further to complete the development and the implementation of our XML schema matching system.
Also, it is deduced that the selection of the best syntactic similarity measure for a schema matching system is a domain specific problem and it may need some means of machine learning approaches to be automatically selected.

7 REFERENCES


