An Efficient Schema Matching Approach Using Previous Mapping Result Set

Hongjie Fan[1], Junfei Liu[2(✉)], Wenfeng Luo[1], and Kejun Deng[1]

[1] School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{hjfan,1201214087,kejund}@pku.edu.cn
[2] National Engineering Research Center for Software Engineering, Peking University, Beijing, China
liujunfei@pku.edu.cn

Abstract. The widespread adoption of e**X**tensible **M**arkup **L**anguage pushed a growing number of researchers to design XML specific Schema Matching approaches, aiming at finding the semantic correspondence of concepts between different data sources. In the latest years, there has been a growing need for developing high performance matching systems in order to identify and discover such semantic correspondence across XML data. XML schema matching methods face several challenges in the form of definition, utilization, and combination of element similarity measures. In this paper, we propose the XML schema matching framework based on previous mapping result set (*PMRS*). We first parse XML schemas as schema trees and extract schema feature. Then we construct *PMRS* as the auxiliary information and conduct the retrieving algorithm based on *PMRS*. To cope with complex matching discovery, we compute the similarity among XML schemas semantic information carried by XML data. Our experimental results demonstrate the performance benefits of the schema matching framework using *PMRS*.

Keywords: Schema matching · XML · Previous mapping result set

1 Introduction

eXtensible Markup Language, because of the flexibility of self-description, has become a standard information representation and exchange of data in a wide range of scenarios [1,2]. XML has been widely used in many domains, such as biology [3], business [4], chemistry [5], and geography/geology [6], to name a few. To make data exchange easier, organizations like the World Wide Web Consortium (W3C) are increasingly committed to define an advanced languages to describe the structure and content of XML data source, such as DTD/XSD. Despite the presence of powerful languages, the achievement of the full interoperability among applications based on XML data is often illusory. One of the biggest obstacles to the development of this technology is how to effectively

H. Gao et al. (Eds.): DASFAA 2016 Workshops, LNCS 9645, pp. 285–293, 2016.
DOI: 10.1007/978-3-319-32055-7_23

identify and correspondence between the semantic nodes, called Schema Matching [7]. One of the most important steps of schema matching between source and target data is to select an appropriate measure which can best calculate an amount of similarity between documents based on their representation, but these measures are time consuming.

A promising approach to improve both the effectiveness and efficiency of schema matching is reusing of previous match results [7]. Exploiting the reuse potential requires a comprehensive repository to maintain previously determined correspondences and match results. Schema matching tools such as COMA [8] and its successor COMA++ [9] apply a so-called *MatchCompose* operator for a join-like combination of two match mappings to indirectly match schemas. [10] is the corpus-based match approach uses a domain specific corpus of schemas and focuses on the reuse of element correspondences. They augment schema elements with matching elements from the corpus and assume that two schema elements match if they match with the same corpus element(s), and use a machine learning approach to find matches between schema and corpus elements. The OpenII project is developing an infrastructure, *Harmony*, for information integration of schemas to permit their reuse [11]. [12] describes an approach called *schema covering* to partition the input schemas such that the partitions can be matched to schema fragments in the repository. Such techniques are not yet common in current match systems, and more research is needed in reuse of previous determined matching result.

In this paper, we develop and implement a schema matching framework based on previous mapping result set, *PMRS*. This matching framework consists of three phases. (1)*Parse Schema and Extract Feature*. During this step, we preprocess the XML data and extract the specific features, such as name, attribute, and comment. (2)*Construct the Matching Framework*. Similarity among XML schemas are determined by exploiting semantic information. In this step, we develop the schema matching framework. We construct *PMRS* as the auxiliary information, and conduct the retrieving algorithm based on *PMRS*. To cope with complex matching discovery, we need to compute the similarity among XML schemas. (3)*Experiment Demonstration*. We carried out a set of experiments to evaluate the proposed framework. Our experiment results show that the proposed framework is useful and efficient in heterogeneous XML data matching issue, especially for the situation of reusing previous match results.

2 Framework

This section gives an overview of our proposed method and our processing can be divided into two major steps: XML Preprocess including feature extraction and Retrieve Process.

Figure 1 depicts the overall framework of our method. During the matching, we present the previous result as the auxiliary information. Data entity e from source schema X_s firstly retrieve from the *PRMS*. If we find the semantic correspondence in this step, then output the mapping result directly, and delete e

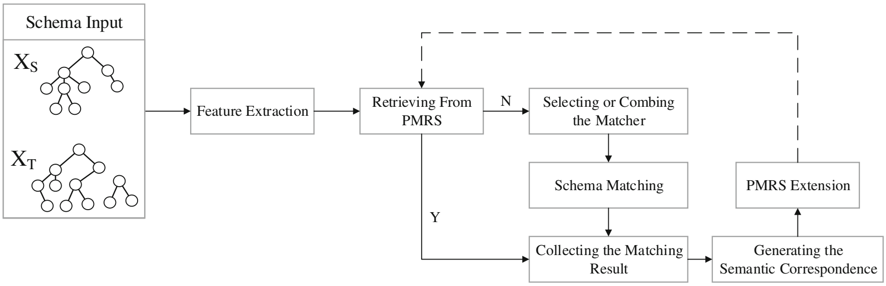


Fig. 1. The framework of our method

from *Entity Set* consequently. This action would increasingly reduce the number of matching entity candidates and boost the matching speed with extension of *PMRS*. If there is no semantic correspondence in this step, we switch into the normal matching workflow. As we known, after collecting the matching result, we generate the semantic correspondence in succession, and can put these collections as *PRMS*. With the process of matching, the *PRMS* will be expanded and enriched. Finally we can achieve the well performance of matching.

The *PMRS* is composed of two parts:

(1) the *Entity Set*(ES) of reference schema and source schema. Entity is the basic matching element. We construct the $ES(E_i)$ for storing all the matching entities E_i from reference schema.
The structure of (ES) is represented as $ES(E_i)$=$\{E_i$,1,e_1,S_{e_1},e_2,S_{e_2},...,e_n,$S_{e_n}\}$. In this structure, e_i means the entity match E_i. S_{e_i} means the original matching similarity, and calculated as S_{e_i}=$Sim(e_i$,$E_i)$. Because entity E_i has three types: *concept*, *attribute*, and *individual*, all these types storing in *PMRS*, the entity belongs to specific type could not match with entity from other types.

(2) the *Comment Key Words Library* ($CKWL$). We need to maintain the ($CKWL$) storing e_i, keywords after splitting *comment*, and the frequency of these keywords. we need to point out if the frequency exceed the threshold, this keyword is called *stopping word* and will be blocked.

3 Quick Retrieving Algorithm Construction

Each XML schema may contain a large scale of computational attributes. Considering some attributes information would be meaningless, in this paper, we use three typical attributes: *name*, *label*, and *comment*. We pick up these features in data preprocess, and construct them as e_i= $<$ *name*, *label*, *comment* $>$. During the quick retrieving algorithm, we calculate the similarity using *e.name*, *e.label*, and *e.comment* with *E.name*, *E.label*, and *E.comment*, e represented as source entity and E represented as result entity. The similarity calculation between e and E as:

$$\begin{cases} Sim(e,E) = \lambda_1\, Sim(e.name, E.name) + \lambda_2\, Sim(e.label, E.label) + \\ \qquad\qquad\quad \lambda_3\, Sim(e.comm, E.comm) \\ \\ \sum_{i=1}^{3} \lambda_i = 1 \end{cases} \tag{1}$$

(1) *Name/Label Simiarity Computation*

We present $Sim(e.name{,}E.name)$ and $Sim(e.label{,}E.label)$ to caculate the entity similarity of name/label between source schema and reference schema. The values of *e.name* and *E.name* are often the words, but may contain some special symbol such as "-". We process all these issues including format the letters in lowercase, get rid of special symbol. Then we use *Levenshtein Distance Algorithm* [13] to calculate the similarity these strings. It is the basic programming algorithm for computing the edit distance. Several variants of the edit distance have been proposed, such as the normalized edit distance [14]. There are many methods to compare strings depending on the way the string is coded (as exact sequence of characters, an erroneous sequence of characters, a set of characters, etc.) [15–17].

(2) *Comment Computation*

We present $Sim(e.comment{,}E.comment)$ to calculate the similarity of comment between source schema and reference schema. Considering *comment* always contain phrase or sentence, we need to split *comment* into keywords set. During this step, another issue, blocking the *stopping words* which exceed threshold, should be pay attention to. The *stopping words* are composed of two parts: *Static Stopping Words* and *Dynamic Stopping Words*. *CKWL* store the frequency of keyword(kw), represented as f_{kw}. The frequency is calculate as $p(kw){=}f_{kw}{+}N_g$, while N_g is the number of entities in reference schema. If $p(kw)$ is large than τ (τ is the threshold we set), we judge kw as stopping word. After that, Then we use the classic *Cosine Method* to calculate the similarity. The cosine similarity is a well known measure from information retrieval. It computes the cosine of the angle between the two d-dimensional vectors $\overrightarrow{\sigma_1}$ and $\overrightarrow{\sigma_2}$ of the two string σ_1 and σ_2. The d dimensions of these vectors correspond to all d distinct tokens that appear in any string in a given finite domain. For example, we assume that σ_1 and σ_2 originate from the same attribute A. The Cosine similarity is calculated as $Cosine(\sigma_1{,}\sigma_2){=}\frac{\overrightarrow{\sigma_1}\cdot\overrightarrow{\sigma_2}}{||\overrightarrow{\sigma_1}||\cdot||\overrightarrow{\sigma_2}||}$. Several variants of the Cosine similarity have been proposed for comment computation, such as the *TF-IDF* [18] *Soft-TFIDF* [16].

(3) *Quick Rretrieving Algorithm*

Based on above algorithm, we calculate the transfer similarity for all same type entities between e from source schema and E_i from *PMRS*. We pick up the highest similarity, and compare it with τ_{PMRS}. If it is larger than τ_{PMRS}, we output it as the final matching result. The transfer similarity is represented as $Sim_{trans}(e,e_i) = Sim(e,e_i)^{*}S_{e_i}$, while $Sim_{trans}(e,e_i)$ is caculated from the context similarity, S_{e_i} is the original similarity which denote the similarity between e and E.

Algorithm 1. Fast Retrieving Algorithm

Require: Input: Entity e.
  Output: $< e, E_i, Sim_{trans} >$.
while $ES(E_i)$ in all ES **do**
  while e_i in all $ES(E_i)$ **do**
    $Sim(e,e_i)$.
    $Sim_{trans}(e,e_i)$=$Sim(e,e_i)$*S_{e_i}.
if $MAX(Sim_{trans}(e,e_i))$>τ_{PMRS} **then**
  Return $< e, E_i, Sim_{trans} >$.

There are two points need to pay attention to in the algorithm implementation process: (1) Only retrieve the same type *ES*. *Concept*, *attribute* and *individual* are the different types of entity, which cannot match with each other. We need to determine e type with *ES*. (2) Entity e firstly need to match entities from reference schema, so we calculate the transfer similarity rather than the similarity between entity e and e_i. The existing result utilized in quick retrieval is an intermediary, so under this circumstance the transfer similarity is used as confidence of similarity equivalence.

(4) PMRS *Extension*

We set the similarity between e and E_i as original similarity, put (e,S) in $ES(E)$, and add *e.comment* into *CKWL*. Since the error information of *PMRS* will effect the final match result, we need to set τ_{PMRS} and τ_{ePMRS} precisely in order to ensure the accuracy of the existing result. We set an expansion threshold τ_{ePMRS}, if similarity is bigger than τ_{ePMRS} after matching between source schema X_S and reference schema X_R, we determine it as the matching pair (e,E_i).

There is no previous matching result in *PMRS* at the time *PMRS* constructed initially. The algorithm is facing the "*cold start*" problem. In order to properly use *PMRS*, we put the entities of reference schema into *PMRS*, and set the original similarity manually. So the quick retrieval algorithm would be operated normally without "*cold start*".

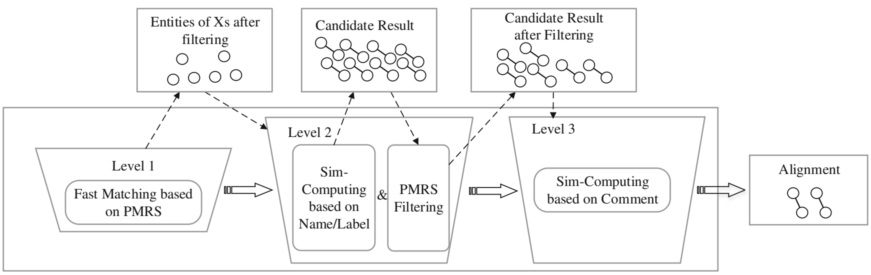


Fig. 2. Matching Workflow by 3-level Filter

Figure 2 depicts the overall matching workflow by 3-level filter. In *level-1*, the entity e firstly match with entity E from reference schema using *Quick Retrieving Algorithm*. If the threshold exceeds τ_{PMRS}, then we output the mapping result directly, and delete data entity e from *Entity Set* consequently. The rest entities take participate in the following matching. In *level-2* and *level-3*, we compute the similarity among XML schemas using similarity measures based on *name*, *label*, and *comment*, respectively. The main reason of schema matching in this way is to reduce the computation cost. During such three level filter, we can get the optimistic matching result in well time cost.

4 Experiment Demostration

In this section, we present the experimental results to demonstrate efficiency and effectiveness of our method. *F-measure* combined precision and recall to present the ratio of error match and missing correct match to evaluate the matching result comprehensively.

4.1 Datasets & Setup

We use OAEI benchmark as our experiment data set to evaluate our schema matching algorithm with other algorithms. All experiments are implemented in Java. Our method is based on disk, and our experiment is conduct on a machine with Intel Corei7 CPU processor, 8G RAM memory and running Ubuntu 14.04 LTS (64bits). Here, we provide more details and statistics about the datasets.

OAEI[1] Since 2004, OAEI organizes evaluation campaigns aiming at evaluating ontology matching technologies, and benchmark is an important part oriented to specific domains. The datasets include 51 schemas come from reference bibliography fields. #101 is the example schema; #101 and #1XX are substantially the same; #2XX lacks some essential elements; #3XX comes from real existing schemas. Figure 3 gives the suitable range of each schema.

Testing purpose	Number	Explanation
Easy test	#103-104	Similar with #101
Comprehensive test	#201-266	To test the robustness and availability of the algorithm. Some elements from the schema (*name*, *attributes*, *constraints*) are excluded.
Application test	#301-304	Real existing schemas. The purpose is to test the performance and availability of the algorithm in practical application.

Fig. 3. Testing Purpose using OAEI Dataset

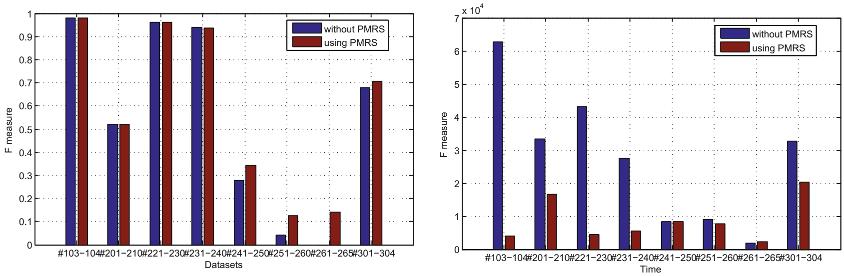
[1] http://oaei.ontologymatching.org/.

4.2 Computational Result and Efficiency

We designed a comparative experiment using *PMRS* and without using *PMRS* to calculate the similarity between #101 schema and #103–304 schemas. Figure 4 presents results for our two quality metrics, the *F-measure* and *time-cost*, respectively.

(1) Similarity Measure Quality

They show that schema matching using *PMRS* in most cases is better than normal matching method. To get better matching quality, different similarity measures have been used, such as similarity measure based on *comment*. Compared to the studied approaches, it improves *F-measure* by +7.3 % and +14.6 % for #251–260, #261–266, respectively. For the schemas #254–257 and #261–266, the original *F-measure* is barely null, but in our experiment *F-measure* is 0.12 and 0.14. It proves the schema matching method using *PMRS* can improve the similarity quality.



(a) *similarity measure quality with/without PMRS* (b) *runtime of executing matching with/without PMRS*

Fig. 4. Similarity measure quality and efficiency with/without PMRS

(2) Effectiveness of *PMRS*

We measure the runtime of computing and executing the schema matching. The results for different datasets are shown in Fig. 4(b). They show the schema matching algorithm using *PMRS* achieves stable running time and reduce the running time efficiently. The average saving runtime achieves at 18 s. Compared to the studied approaches, it reduce runtime by 58 s at most for #103–104. For the complex schema such as #301-#304, the improvement achieves at 12 s (reduce 38 % runtime). Our experimental results demonstrate the effectiveness of the schema matching framework using *PMRS*.

5 Conclusions

In this paper, we propose the XML schema matching framework based on previous mapping result set (*PMRS*). We construct *PMRS* as the auxiliary information and conduct the retrieving algorithm based on *PMRS*. To cope with complex matching discovery, we compute the similarity among XML schemas semantic information. Our experimental results demonstrate the performance benefits of the schema matching framework using *PMRS*. Future research is geared towards efficiently generating candidate queries for similarity evaluations.

Acknowledgements. This research is supported by The National Natural Science Foundation of China under Grant No. 61272159 and No. 61402125. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. XML schema part 1: Structures. http://www.w3.org/TR/xmlschema-1
2. De Meo, P., Quattrone, G., Terracina, G., Ursino, D.: Integration of XML schemas at various "severity" levels. Inf. Syst. **31**(6), 397–434 (2006)
3. Hucka, M., et al.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics **19**(4), 524–531 (2003)
4. Ebxml website. http://www.ebxml.org
5. Murray, P.: Chemical markup language: a simple introduction to structured documents. World Wide Web J. **2**(4), 135–147 (1997)
6. Gml website. http://www.opengis.net/gml/
7. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
8. Do, H.H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: Proceedings of 28th International Conference on Very Large Data Bases. VLDB 2002, 20–23 August 2002, Hong Kong, China, pp. 610–621 (2002)
9. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, 14–16 June 2005, pp. 906–908 (2005)
10. Aberer, K., Franklin, M.J., Nishio, S. (eds.). Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5–8 April 2005, Tokyo, Japan. IEEE Computer Society (2005)
11. Seligman, L., Mork, P., Halevy, A.Y., Smith, K.P., Carey, M.J., Chen, K., Wolf, C., Madhavan, J., Kannan, A., Burdick, D.: Openii: an open source information integration toolkit. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. SIGMOD 2010, Indianapolis, Indiana, USA, 6–10 June 2010, pp. 1057–1060 (2010)
12. Saha, B., Stanoi, I., Clarkson, K.L.: Schema covering: a step towards enabling reuse in information integration. In: Proceedings of the 26th International Conference on Data Engineering. ICDE 2010, 1–6 March 2010, Long Beach, California, USA, pp. 285–296 (2010)

13. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady, pp. 707–710 (1966)
14. Marzal, A., Vidal, E.: Computation of normalized edit distance and applications. IEEE Trans. Pattern Anal. Mach. Intell. **15**(9), 926–932 (1993)
15. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
16. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb 2003), 9–10 August 2003, Acapulco, Mexico, pp. 73–78 (2003)
17. Formica, A.: Similarity of XML-schema elements: a structural and information content approach. Comput. J. **51**(2), 240–254 (2008)
18. Joachims, T.: A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, 8–12 July 1997, pp. 143–151 (1997)