DOI: 10.3969/j. issn. 1673-5188. 2015. 04. 004
http://www.cnki.net/kcms/detail/34.1294.TN.20151130.0926.002.html, published online November 30, 2015

A Novel Data Schema Integration Framework for the Human-Centric Services in Smart City

Ding Xia, Da Cui, Jiangtao Wang, and Yasha Wang
(Peking University, Beijing 100871, China)

Abstract

Human-centric service is an important domain in smart city and includes rich applications that help residents with shopping, dining, transportation, entertainment, and other daily activities. These applications have generated a massive amount of hierarchical data with different schemas. In order to manage and analyze the city-wide and cross-application data in a unified way, data schema integration is necessary. However, data from human-centric services has some distinct characteristics, such as lack of support for semantic matching, large number of schemas, and incompleteness of schema element labels. These make the schema integration difficult using existing approaches. We propose a novel framework for the data schema integration of the human-centric services in smart city. The framework uses both schema metadata and instance data to do schema matching, and introduces human intervention based on a similarity entropy criteria to balance precision and efficiency. Moreover, the framework works in an incremental manner to reduce computation workload. We conduct an experiment with real-world dataset collected from multiple estate sale application systems. The results show that our approach can produce high-quality mediated schema with relatively less human interventions compared to the baseline method.

Keywords

schema matching; schema integration; smart city; human-centric service

1 Introduction

Human-centric service is an important domain of the smart city and includes rich applications that help residents with shopping, dining, transportation, entertainment, and other daily activities. While offering services, these systems have generated a large amount of hierarchical data. Usually data from each system is incomplete, and one system complements another. Take the data of second-hand housing for example. There are many second-hand housing information sharing application systems containing data for a given city in China. In Beijing, such systems include lianjia.com, 5i5j.com, 58.com, fang.com, iwjw.com, and many other local forums. Since each system only contains certain information, a resident who wants to buy a second-hand house or apartment needs to browse the systems one by one and pull together all the parts of the information by themselves. In another scenario, if a city planning or market investigation department wants to know the situation of the second-hand house market of the whole city, they also need to integrate data from different systems. As the data schemas of different systems are diverse, the schema integration, whose goal is to establish a whole and unified schema for all the multi-sourced datasets, is necessary and crucial.

Traditional schema integration techniques are usually used in scenarios where the number of data schemas is small, the structure or semantic of the schemas is well understood, and the linguistic resources for semantic matching across different schemas are sufficient. Typical application scenarios include the evolution of product directories in the domain of e-commerce, data system integration caused by the company merges, and so on [1]–[5]. However, data integrating in the domain of human-centric service of smart city is much more challenging due to its distinct characteristics as follows:

1) Broad application domains and lack of domain knowledge. Human-centric service is about almost everything in a resident's daily life. There has not been any standard or knowledge base providing support for the semantic matching in this domain. For example, in second-hand housing data sets, there are dozens of terms for the sales agent, such as agent, broker, advisor, secretory, or housekeeper across various systems. Therefore, we need a domain specified term dictionary to help us figure out whether two elements from two schemas

This work is funded by the National High Technology Research and Development Program of China (863) under Grant No. 2013AA01A605.

refer to the same concept. However, no such dictionary is available, and it is time-consuming to build one. Also, traditional string-based matching algorithms perform poorly for Chinese labeled elements.

2) Large number of schemas. Even in a fine-grained sub domain such as second-hand housing, there are dozens of systems from which data schemas need to be integrated. Traditional schema integration methods mostly involve studying how to match or integrate two schemas, and if we simply use them to work on every pair of many schemas, it will cost too much time and has poor extendibility.

3) Label Incompleteness. On the one hand, some data in human-centric service domain is acquired from web tables and sometimes the labels of schema elements are missing. This makes traditional element-level matching techniques unsuitable. On the other hand, instance data is usually available and can be used to assist schema matching. However, instance data from different systems do not always overlapped, so we won't get satisfying result if we simply calculate instance overlapping level to represent similarity between two elements.

Due to the abovementioned characteristics, the existing approach is incapable of handling the data schema integration for human-centric service in smart city. Therefore, we propose a novel approach of schema integration with data from domain of human-centric services in this paper. In our approach, we use a mediated schema to help integrate multiple schemas in a quick manner. Every schema will be matched and integrated to the mediated schema only once, i.e., one iteration, and the mediated schema is updated and extended after each iteration. During each iteration, a depth-first-search algorithm is used to control element-matching and integration order. Five matchers that utilize both schema metadata and instance data are combined for schema matching. We introduce a similarity entropy based interactive method of human intervention controlling to make matching results more precise. After schema matching, a set of conflict resolution strategy is adopted to solve all kinds of complex conflicts and then form a better and more complete new mediated schema.

The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 gives the framework of our approach. Section 3 describes detailed algorithms. Section 4 designs and conducts experiments to do validation. Section 5 concludes this paper and analyzes future work.

2 Related Work

Schema integration can be divided into two parts: schema matching and mediated schema generation. Schema matching involves forming mapping between elements in different schemas, which have the same or similar semantics, while the mediated schema generation is to generate a whole and unified schema for all the integrated schemas based on the result of schema matching.

2.1 Schema Matching

According to the input, schema matching can be divided into two categories: metadata based matching and instance-based matching. Metadata-based matching uses the metadata, including labels and structures of the elements in the input schemas, as the input of the matching. The most basic matching algorithms in this category are called element-level matchers, which maps the elements from different schemas based on the their labels according to the string similarity [1], [2], [6], linguistics-based semantic relationship [7], [8], or word co-occurrence in schemas [9], [10]. Another kind of metadata-based matching algorithm is called structure-level matchers, which not only take the element information into consideration but also the structure of the element information. Typical structure-level matchers include graph-based matchers [11], [12] and path-based matchers [3], [13]. Although metadata-based matching algorithms are fast, they may be unfeasible when the metadata of the schemas are incomplete. On the other hand, instance-based matching algorithms do not depend on the metadata of schemas. Instance-based matchers dig similarity between elements from instance data. Typical instance-based matchers calculate the similarity among elements according to instance statistical features [14] or the overlapping instances [15], [16].

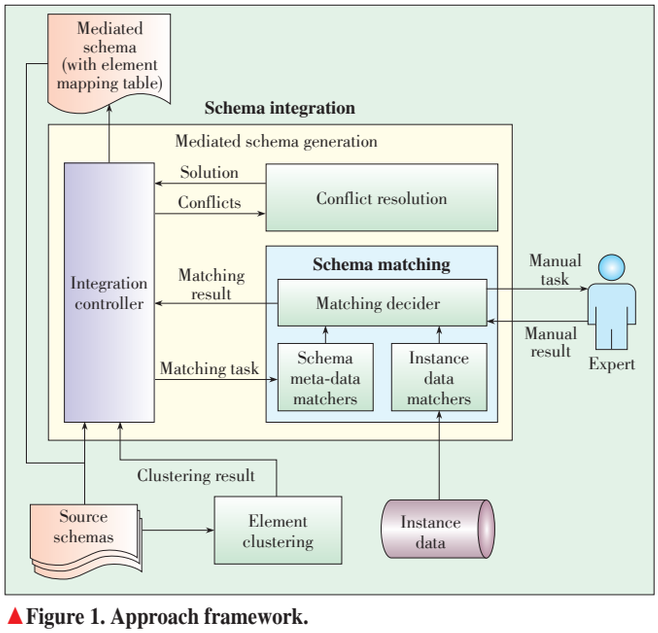
2.2 Mediated Schema Generation

Based on the results gain from the schema matching, mediated schema generation algorithms try to resolve conflicts in element naming, definition, and structure inconsistency from different schemas. Then, such algorithms form a mediated schema containing all of the elements in the integrated schemas to make the heterogeneity of schemas transparent to the data users [4]. In addition to the conflict resolution, the number of input source schemas has great impact on mediated schema generation. Traditional resolution, such as XSIQ [4], is applied to two source schemas. However, when there are many schemas to be integrated, techniques that apply to two schemas greatly increase complexity. XINTOR [17] provided a global schema integration technique for multiple source schemas. It uses the statistical characteristics of element structure to generate a mediated schema. XINTOR greatly increases the efficiency of matching multiple schemas but it has poor extensibility. Another idea is taking mediated schema as an intermediate product and integrating all the other schemas into it. This increases both efficiency and extensibility. In some research [18], human experts are required to develop a mediated schema. However, under the background of human-centric services which have massive multi-source heterogeneous datasets, these techniques pose too much burden on experts. Automatic generation of mediated schema will avoid large number of human effort, for example, PORSCHE [19]. PORSCHE integrates one schema with

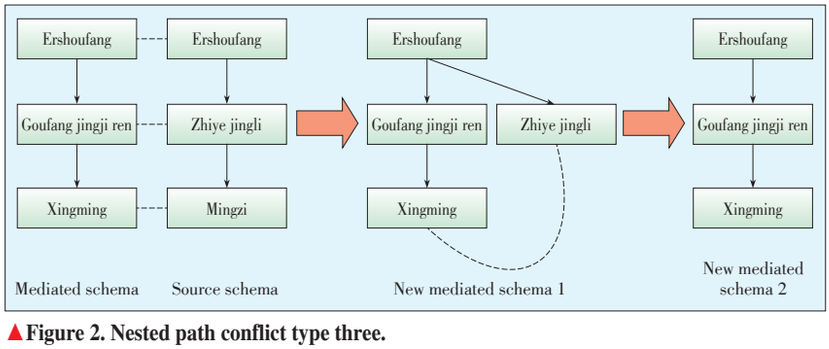
mediated schema for one time and in the meantime updates the mediated schema. It only takes advantage of schema information, which results in limited accuracy. Moreover, it has simple conflict resolution strategy and some conflict types are not considered. Hence, the quality of mediated schema generated by PORSCHE is low.

3 Framework Overview

The framework of our approach is shown in **Fig. 1**. There are three kinds of input into the system. The first kind is schema metadata of the data sources, which defines the structure of the data, and the elements making up the structure. For example, the source schema in **Fig. 2** is part of the schema metadata of the web application called 'Anjuke', and it consists of three string labels. The second kind is the instance data of every leaf element, providing the actual useful information, such as the leaf element 'Mingzi' in the source schema in Fig. 2. The instance data are real names of second-hand housing agents. The third kind is human intervention. The major output is the mediated schema. In our approach, the mediated schema serves as



▲**Figure 1. Approach framework.**



▲**Figure 2. Nested path conflict type three.**

intermediate product and all the other source schemas will be matched and integrated to it sequentially, resulting in its update and expansion.

Element clustering module takes all the source schemas as input and uses element level matcher to calculate the similarity of any two elements from different schemas. Based on these similarity values, all elements will then be classified into several clusters, which are used to reduce the complexity of the following computation.

The schema integration module takes source schemas, current mediated schema, and instance data as input to complete schema integration task under the support of experts' knowledge and elements clustering result. In this module, first, one of n source schemas is chosen as the initial mediated schema. Then, the left n-1 source schemas are integrated into the mediated schema sequentially in an n-1 iteration. Each iteration is a complete schema integration process: the integration controller submodule traverses the matching source schema in depth-first-search order and for every element that is traversed and finds its candidate matching elements in the mediated schema using the clustering result. Then schema matching submodule calculates similarities between the traversed element and its candidates only, thus avoiding the similarity calculation of all element pairs. In the schema matching submodule, element level matcher, ancestor path matcher, tree edit distance matcher, which make use of schema metadata, and statistic based instance matcher, content based instance matcher, which make use of instance data are used. Some of these matchers are modified or redesigned to adapt to Chinese labels, and different matchers are reasonably and efficiently combined according to their characteristics. Also, human intervention based on similarity entropy is introduced. Questions are generated and sent to experts if the schema matching algorithm cannot automatically decide which candidate is the most appropriate one to the traversed element. The matching decider submodule sends these questions, gathers experts' feedback, and determines the final matched element pair. When matched, the conflict resolution submodule will be used to solve conflicts between the two matched elements. The mediated schema is updated and expanded after every iteration, and at the end of the last iteration, the final mediated schema and element mapping table is obtained.

Only an element clustering process and another iteration process of schema integration are needed when a new source schema is added in. All the updates and expansions brought by the new added source schema are incremental and the pre-existing results of schema integration will not be reversed.

4 Algorithm Design

Here, we describe the algorithms of the four

most important modules: element clustering module, integration controller submodule, schema matching submodule, and conflict-resolution submodule respectively. In schema matching submodule, the five different matchers, the combination design and the human intervention design are introduced in order.

4.1 Element Clustering

This module clusters elements according to the similarity calculated by the element level matcher. We use and modify the Kruskal algorithm which is originally used to calculate the minimum spanning tree, to do the clustering job. First, we calculate the similarity values of every element pair using the element level matcher, which is the quickest among all matchers. Then we sort them in ascending order and every time pick an element pair that has the smallest similarity to be clustered. The intuition behind this algorithm is to make the difference among clusters to be as large as possible.

4.2 Integration Controller

During each round of the iteration, the current source schema is matched and integrated into the mediated schema. We traverse the current source schema in depth-first-search order to ensure that whenever an element is about to be matched, its father element is already matched. The detailed steps are described as follows:

1) The root elements of all source schemas should all be matched because all the source schemas are from the same domain. We traverse the current source schema from the root's first child element.
2) Denote the currently traversed element as element *a*. Find the cluster containing *a* from the clustering result, denoted as *A*. If the size of *A* is one, go to step 7.
3) Extract all elements which exist in the mediated schema from *A* to form another set *B*. If the size of *B* is zero, go to step 7.
4) For each element *bi* in *B*, invoke schema matching submodule to calculate its comprehensive similarity with *a*, denoted as *Si*, *i* is from one to the size of *B*.
5) Calculate *Entropy(a)* using *S1* to *S|B|*. If *Entropy(a)* is greater than *T*ln|B|*, send question to experts to receive human intervention and get the right matching element *bm*, otherwise, choose the element that has the greatest comprehensive similarity with *a*, denoted as *bm* as well.
6) Match *a* to *bm* and invoke conflict resolution submodule to solve possible conflicts. Update the mediated schema and record elements mapping relationships. Go to step 8.
7) If no element in the mediated schema can be matched to *a*, find the father element of *a*, denoted as *fa*. Denote the element that has been matched to *fa* as *fb*, and add *a* to the mediated schema as *fb*'s right-most child element.
8) If *a* is not a leaf element, get the first child element of *a* and repeat step 2. Otherwise get *a*'s next brother element and repeat step 2. If *a* is the last element in depth-first-search order, algorithm ends.

4.3 Schema Matching

In our approach, we use five matchers for schema matching. Both schema metadata and instance data are utilized, and different matchers are reasonably and efficiently combined according to their characteristics. Interactive human intervention based on similarity entropy is introduced to improve matching accuracy.

4.3.1 Element Level Matcher

Traditional element level matchers primarily acquire similarity value between elements according to their string-based similarity and semantic-based similarity. Commonly used string-based algorithms perform poorly whereas processing Chinese label, and semantic-based algorithms rely deeply on external domain synonyms dictionary. Therefore, we use Word2Vec [20] to help calculate element level similarity. Word2Vec is a tool to map words to *K*-dimension real vector space, so the similarity between two words can be represented by the cosine value of the two *K*-dimension vector. If trained by a large corpus in a particular domain, similarity values calculated can not only represent the string-based similarity of words but also cover parts of their semantic-based similarity. This matcher runs fast and can be served as a standalone matcher in most cases, but it only has an ordinary accuracy, not satisfying enough. **Algorithm 1** shows the process of this matcher.

Algorithm 1. Element level matcher

Input: *s*1, *s*2: labels of two elements to match
Output: *Element_Similarity*: element level similarity
1. **begin**
2. *TokenList*1 ← *Tokenize*(*s*1)
3. *TokenList*2 ← *Tokenize*(*s*2)
4. *Element_Similarity* ← 0
5. **for** each *token*1 ∈ *TokenList*1 **do**
6. **for** each *token*2 ∈ *TokenList*2 **do**
7. *Element_Similarity* += *Word2VectorSimilarity*(*token*1, *token*2)
8. **end for**
9. **end for**
10. *Element_Similarity* /= sizeof(*TokenList1*)*sizeof(*TokenList2*)
11. **end**

4.3.2 Ancestor Path Matcher

Ancestor path matcher is a kind of structure level matcher. The idea of this matcher is that two elements are similar to each other if they have matched ancestors and the paths from the matched ancestor to them are short enough. This matcher strictly control the matching relation of elements based on

their ancestor elements' matching relation. By using this matcher we can effectively avoid mismatching case such as falsely matching an element labeled 'Xingming' whose father element is labeled 'Zhiye Jingli' to another element labeled 'Xingming' whose father element is labeled 'Yezhu'. **Algorithm 2** shows the process of this matcher.

Algorithm 2. Ancestor path matcher

Input: a, b: two matching elements
Output: *AncestorPath_Similarity*: ancestor path similarity
1. **begin**
2. $c \leftarrow$ parentOf(a)
3. *AncestorPath_Similarity* $\leftarrow 0$
4. **while** $c \neq$ null **do**
5. $d \leftarrow$ MatchingNodeOf(c)
6. **if** d.isAncestorOf(b) = true
7. **then break**
8. **else**
9. $c \leftarrow$ parentOf(c)
10. **end if**
11. **end while**
12. *AncestorPath_Similarity* $\leftarrow$ 1/length(c, a)
13. **end**

4.3.3 Tree Edit Distance Matcher

Tree edit distance matcher is another kind of structure level matcher. The idea of this matcher is that two elements are similar if their subtrees are alike. To measure the likeness of two trees, we use tree edit distance, the definition of which is the minimum number of operations required to transform one tree into another [12]. We use dynamic programming to calculate it. This matcher runs slowly but can measure the similarity between elements according to their subtree structure. For example, elements labeled 'Zhiye Jingli' and 'Goufang Jingjiren' can be very hard to match using other matchers, such as element level matcher. However, we can obtain a pretty high similarity through tree edit distance because the two element both have child elements labeled 'Xingming', 'Dianhua', 'Gongsi'. One obvious weakness of this matcher is that useless for leaf elements, which have no child element.

4.3.4 Statistical Based Instance Matcher

A statistical-based matcher is a kind of matcher that uses element instance data and focuses on statistical indexes of this data. We first calculate Eigenvectors of every leaf element using its instance data, then we use the Back Propagation Neural Network Algorithm to get a classifier for every source schema each.

Thirteen features are chosen for Eigenvector. For instances of numeric type, features are computed using its value, while for instances of entity type, features are computed using its length. We have six data type features: integer, floating number, URI, date, string and text, and seven statistical feature: maximum, minimum, mean, standard deviation, mean-squared difference coefficient, number of bytes and precision.

When calculating the similarity value between two leaf elements a and b, we will put the eigenvector of a and b into the classifier of schema A and B respectively to get two similarity values, then we take the average of them as the statistical based instance similarity between a and b.

This matcher uses element instance data, but for instances of entity type, it only takes their lengths into account and neglects their content. This matcher has high recall rate but low precision rate when used alone, and it runs fast.

4.3.5 Content Based Instance Matcher

Content based matcher is another kind of matcher that utilizes elements' instance data, it focuses on the content of elements' instance data. For two leaf elements, the overlapping level between the contents of their instance data can represent a sort of similarity between them. We first classify elements into three data types: text, entity and number. Then we calculate the content based instance similarity between two leaf element a and b as follows: 1) If a and b differ in their data type, the similarity value is 0. 2) If their data types are both text, we merge their instance data into two documents, count word frequency separately, and use the vector space model to convert the two documents to two vectors to calculate cosine value as their similarity value. 3) If their data types are both entity, for every instance of a, find the instance of b that has the smallest edit distance with it. We accumulate these smallest edit distance values and average it to get the similarity value between a and b. 4) if their data types are both number, we calculate their standard deviations and means of data instance separately, and acquire the similarity value between a and b as follows:

$$Content\_Similarity = 2/\left(\frac{|std1-std2|}{std1+std2}+\frac{|avg1-avg2|}{avg1+avg2}\right) \quad (1)$$

This matcher uses element instance data and focuses on the content of instance data. Due to the impact of instance subset problem, this matcher is precise but has a low recall rate when used alone. It runs slowly because a lot of computation is needed when getting the overlapping level.

4.3.6 Matcher Combination Algorithm

So far, we have introduced five matchers with different characteristics. Element level matcher runs the fastest but is not accurate enough, ancestor path matcher cannot be used alone but is a very good supplement, tree edit distance matcher is useless for leaf elements, and statistic based instance matcher and content based instance matcher both apply to leaf elements only. How to reasonably and efficiently combine them to get a comprehensive similarity for element pairs is a crucial prob-

lem concerned in this section.

Element level matcher has the highest calculating efficiency, so we use this to help with element clustering. The other matchers are less efficient and only work on parts of the whole element pairs based on the clustering result, thus the total calculation is greatly reduced.

The two matchers that use element instance data are complementary, so we can combine them in advance. Statistical based instance matcher has high recall rate and low precision, and a high similarity calculated by this matcher is actually not so reliable. On the contrary, content based matcher has high precision rate and low recall rate, and a low similarity calculated by this matcher is actually not so reliable. Considering that statistical based matcher is much quicker than content based matcher, we can first use statistical based matcher to get a similarity, and only when this similarity is high enough will we use content based matcher to adjust the unreliable result. This combination method can guarantee both a high accuracy and a high efficiency. We call this similarity instance similarity, and it can be calculated as follows, *Sta_Sim* is the statistical based matcher, *Con_Sim* is the content based matcher and *d* is an empirical threshold.

$$Instance\_Similarity(a,b) = \begin{cases} Sta\_Sim(a,b), & Sta\_Sim(a,b) < d \\ d + (Sta\_Sim(a,b) - d) * Con\_Sim(a,b), & Sta\_Sim(a,b) \geqslant d \end{cases} \quad (2)$$

We then combine instance similarity with the result of tree edit distance matcher because the former applies to leaf elements only whereas the latter is useless for leaf elements. When two matching elements are both leaf elements, the instance similarity between them is calculated; otherwise, we use tree edit distance matcher to determine the similarity. We denote this combined similarity as subtree similarity. As for the other two matchers—element level matcher and ancestor path matcher, we call their result as element level similarity and ancestor path similarity.

At last, we get the comprehensive similarity of every element pair by calculating the weighted mean value of subtree similarity, element level similarity and ancestor path similarity.

4.3.7 Human Intervention Design

The comprehensive similarity calculated above sometimes is still not accurate enough, so we introduce human intervention to our framework. We assume that experts' advice is always right but the number of expert queries must be controlled.

For every matching element in a source schema, there are several comprehensive similarity values that belong to it and its candidates. The idea of our human intervention is that if one of these similarity values is significantly higher than others, then it is the match. However, if these similarity values are so close, we cannot determine a prominent one. In the latter case, we can form questions and send them to experts to get a real match.

We use similarity entropy to decide whether there is a need to query experts. The equation to calculate the similarity entropy is as follows:

$$Entropy(x) = -\sum_{j=1}^{k} p(sim_j) * lnp(sim_j),$$
$$p(sim_j) = \frac{sim_j}{\sum_{i=1}^{k} sim_i} \quad (3)$$

For a candidate set consisting of *K* elements, *Entropy(x)* ranges from 0 to *lnK*. We import a threshold *T* and we ask experts for advice only if *Entropy(x)* is greater than *T*lnK*, otherwise we will choose the element that has the greatest comprehensive similarity with the matching element.

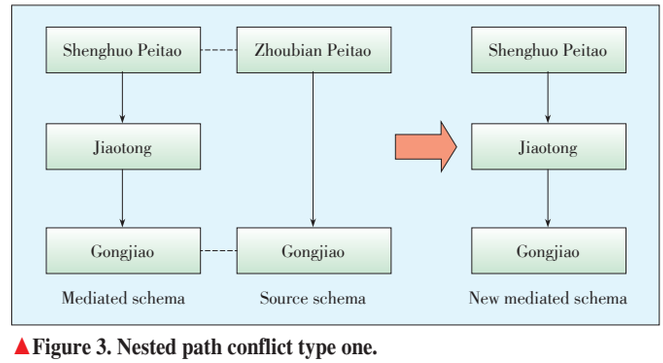
4.4 Conflict Resolution

Conflicts may arise when elements get matched. Conflicts and our solution strategies are as follows:

1) Synonym conflict. Our resolution is to reserve old element's label to be the one in the mediated schema but also record the new matching element's label for later use.
2) Data type conflict. Our resolution is to reserve the one of higher precision type or stronger expression. Data type conversion is needed.
3) Element substructure conflict. This conflict occurs when a leaf element is matched to an internal element. Our resolution is to reserve the internal element's substructure. In fact, this kind of conflict can be automatically resolved during the operation of algorithm in section 4.2.
4) Nested path structure conflict. This kind of conflict is the most complicated one. It arises from the inconsistency of two paths from currently matched elements and already matched ancestors elements in two schemas.

The nested path structure conflict can be subdivided to three types of conflicts:

1) Path in mediated schema contains nested element. See **Fig. 3**, there is an intermediated element labeled 'Jiaotong' between 'Shenghuo Peitao' and 'Gongjiao' the in mediated schema. This kind of conflict can be automatically resolved



▲**Figure 3. Nested path conflict type one.**

during the operation of algorithm in section 4.2.

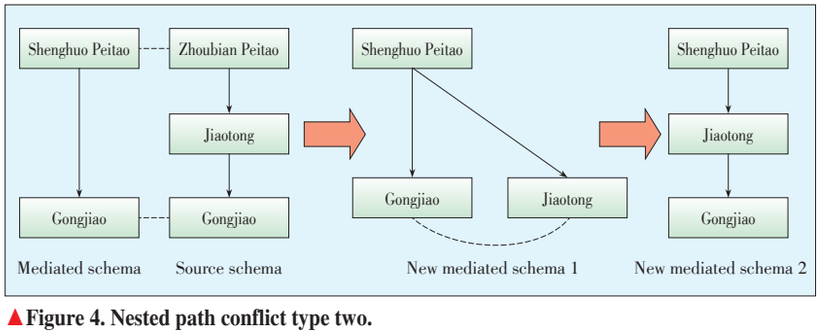
2) Path in source schema contains nested element. See **Fig. 4**, there is an intermediated element labeled 'Jiaotong' between 'Shenghuo Peitao' and 'Gongjiao' the in mediated schema. If we do not take any measures, the integrated mediated schema will look like new mediated schema 1 because we cannot find a matching object for 'Jiaotong', it will be added to the mediated schema as the right-most child element of 'Shenghuo Peitao', which is a mistake. However, this kind of mistake can be detected later when it comes to 'Gongjiao', its original father element 'Jiaotong' in source schema now becomes its brother. Our solution is to cut down the connection between 'Gongjiao' and 'Shenghuo Peitao' in the mediated schema and add a connection between 'Gongjiao' and its new father element 'Jiaotong'. Finally, we get the correct mediated schema, the new mediated schema 2.

3) There are nested elements in two schemas and the two nested elements are falsely not matched. See Fig. 2, 'Goufang Jingjiren' and 'Zheye jingli' are dissimilar in string, so they are falsely not matched and 'Zheye jingli' is placed into the mediated schema as brother element of 'Goufang Jingjiren'. This forms a new mediated schema 1, which is a mistake. However, this kind of mistake can be detected later when it comes to element 'Mingzi'; its original father element in source schema now becomes its father's brother. Our solution is to cancel the previous matching decision of element 'Zheye jingli', and match it to the element 'Goufang Jingjiren' according to the matching result of their child elements. Finally, we will get the correct mediated schema, the new mediated schema 2.

5 Evaluation

5.1 Datasets

Experimental data is collected from multi-source heterogeneous data sets in second-hand housing domain, including second-hand housing information published by anjuke (http://www.anjuke.com), 5i5j (http://bj.5i5j.com), and lianjia (http://www.lianjia.com) in January 2015. Their XML schemas contain 51, 50 and 46 elements, respectively. We first do data integration on these schemas manually to get a standard result. Basic information about experimental data set is shown in **Table 1**.



▲**Figure 4. Nested path conflict type two.**

5.2 Metrics and Baseline Method

We compare the matching result of our approach and the manual result, and primary evaluating metrics are precision rate (precision), recall rate (recall), and F-measure (F_measure). The equations used to calculate these are as follows. TP is the number of matching element pairs that exist in both results. FP is the number of matching element pairs that exists only in results of our approach. FN is the number of matching element pairs that exists only in the manual result.

▼**Table 1. Basic information about data set**

Number of source schemas	Total number of elements	Total number of instance data items	Total number of element pairs	Number of matched element pairs	Number of elements in manual mediated schema
3	147	30,121	7196	108	73

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (4)$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (5)$$

$$F\_measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

Performance oriented schema mediation (POSCHE) is the most similar related work with us, and we both do data integration in an incremental manner. Therefore, we choose POSCHE for comparison. In the meantime, whether there exists manual intervention in our framework imposes great impact on experimental result. Our framework makes decision on its own if there is no human intervention. Also, we conduct experiments on the effect of human intervention extent on the accuracy of our approach.

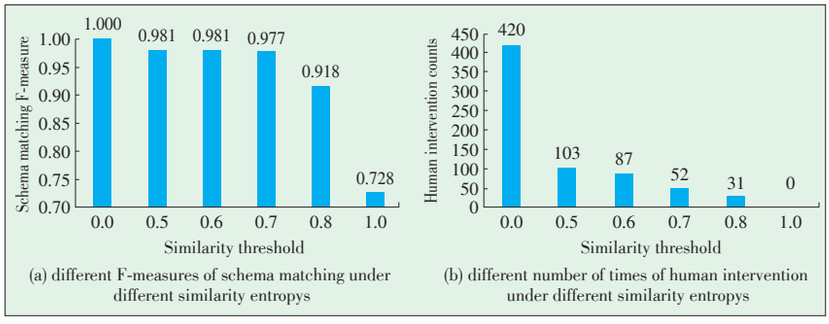
5.3 Experimental Result

5.3.1 Influence of Manual Intervention

We set the threshold of similarity entropy, denoted T, to 0.0, 0.5, 0.6, 0.7, 0.8 and 1.0 to conduct six experiments, respectively. We record the number of expert queries and F-measures for every experiment. T=0.0 represents that decision is made all by human while T=1.0 represents decision is made all by machine. Experimental result is shown in **Fig. 5**. We can see that more human intervention corresponds to better F-measure. In overall consideration of human degree of effort and approach performance, we set T to 0.7, under which little human effort is paid but performance is much improved.

5.3.2 Performance of Schema Matching

We compare the performance of POSCHE, our framework without human intervention (T=1.0),



(a) different F-measures of schema matching under different similarity entropys

(b) different number of times of human intervention under different similarity entropys

▲**Figure 5. Human intervention experiment.**

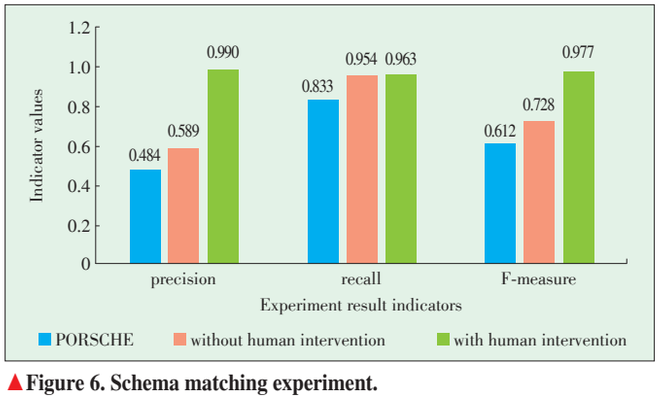
our framework with human intervention (T=0.7) in three experiments. Evaluating indicators are precision rate, recall rate and F-measure (**Fig. 6**).

Our framework with human intervention is better than our framework without human intervention in terms of all three indicators. Our framework without human intervention is better than POSCHE. In the experiment using our framework with human intervention, only 52 questions were posed to experts, which account for only 0.7% of the whole search space (totally 7196 pairs) and 12.4% of all questions (totally 420 questions are needed if all handled by human), but the F-measure reached up to 97.7%.

6 Conclusion

Aiming at the problems in domain of human-centric services, we propose a novel approach of schema integration with data from domain of human-centric services. In our approach, we use a mediated schema to help quickly integrate multiple schemas. Every schema is matched and integrated to the mediated schema only once (i.e., one iteration) and the mediated schema is updated and extended after each iteration. During each iteration, a depth-first search algorithm is used to control element matching and integration order. Five matchers which utilize both schema metadata and instance data are combined to complete schema matching. We introduced a similarity entropy based interactive method of human intervention controlling to make matching results more precise. After schema matching, a set of conflict resolution strategy is used to solve all kinds of complex conflicts and then form a better and more complete new mediated schema. We finally use real second-hand housing data from the internet to design and conduct experiments. The results show that our approach performs very well and requires very little human intervention.

A limitation of our approach is that although the degree of human intervention can be minimized, performance is not satisfactory if there is no human intervention. In the future, we will further study automatic schema matching and integration algorithms in order to improve matching performance with no human intervention. Another future work is to study entity matching work which is another important technique to fully accomplish the task of forming a complete data set in smart city.



▲**Figure 6. Schema matching experiment.**

References

[1] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic schema matching with cupid," Microsoft Research, Microsoft Corporation, Tech. Rep. MSR-TR-2001-58, Aug. 2001.
[2] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with COMA++," in *ACM SIGMOD International Conference on Management of Data*, Baltimore, USA, Jun. 2005. doi: 10.1145/1066157.1066283.
[3] N. F. Noy and M. A. Musen, "Anchor-PROMPT: using non-local context for semantic matching," in *Workshop on Ontologies and Information Sharing at IJCAI*, Aug. 2001, Seattle, USA.
[4] S. Madria, K. Passi, and S. Bhowmick, "An XML schema integration and query mechanism system," *Data & Knowledge Engineering*, vol. 65, no. 2, pp. 266–303, May 2008.
[5] R. A. Pottinger and P. A. Bernstein, "Merging models based on given correspondences," in *29th International Conference on Very Large Data Bases*, Berlin, Germany, Sept. 2003, pp. 862–873.
[6] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," *Journal on Data Semantics IV*, vo. 3730, pp. 146–171, 2005. doi: 10.1007/11603412_5.
[7] P. Bouquet, L. Serafini, and S. Zanobini, "Semantic coordination: a new approach and an application," in *Second International Semantic Web Conference*, Sanibel Island, USA, Oct. 2003, pp. 130–145. doi: 10.1007/978-3-540-39718-2_9.
[8] F. Giunchiglia and M. Yatskevich, "Element level semantic matching," in *Meaning Coordination and Negotiation Workshop at ISWC*, Hiroshima, Japan, Nov. 2004.
[9] B. He, K. C.-C. Chang, and J. Han, "Discovering complex matchings across web query interfaces: a correlation mining approach," in *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, USA, Aug. 2004. doi: 10.1145/1014052.1014071.
[10] W. Su, J. Wang, and F. Lochovsky, "Holistic query interface matching using parallel schema matching," in *22nd International Conference on Data Engineering*, Atlanta, USA, Apr. 2006, pp. 122–125. doi: 10.1109/ICDE.2006.77.
[11] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM Journal on Computing*, vol. 18, no. 6, pp. 1245–1262, Dec. 1989. doi: 10.1137/0218082.
[12] D. Shasha, J. T. L. Wang, and R. Giugno, "Algorithmics and applications of tree and graph searching," in *Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Madison, USA, 2002, pp. 39–52. doi: 10.1145/543613.543620.
[13] C. Liu, J. Wang, and Y. Han, "Mashroom+: an interactive data mashup approach with uncertainty handling," *Journal of Grid Computing*, vol. 12, no. 2, pp. 221–244, Jun. 2014. doi: 10.1007/s10723-013-9280-5.
[14] W.-S. Li, C. Clifton, and S.-Y. Liu, "Database integration using neural networks: implementation and experiences," *Knowledge and Information Systems*, vol. 2, no. 1, pp. 73–96, Mar. 2000. doi: 10.1007/s101150050004.
[15] S. Massmann and E. Rahm, "Evaluating instance-based matching of web directories," in *11th International Workshop on the Web and Databases*, Vancouver,

Canada, Jun. 2008.
[16] J. Fan, M. Lu, B. C. Ooi, *et al.*, "A hybrid machine-crowdsourcing system for matching web tables," in *IEEE 30th International Conference on Data Engineering*, Chicago, USA, 2014, pp. 976–987. doi: 10.1109/ICDE.2014.6816716.
[17] H.-Q. Nguyen, D. Taniar, W. Rahayu, and K. Nguyen, "Double-layered schema integration of heterogeneous XML sources," *Journal of Systems and Software*, vol. 84, no. 1, pp. 63–76, Jan. 2011. doi: 10.1016/j.jss.2010.07.055.
[18] A. Aboulnaga and K. el Gebaly, "μBE: user guided source selection and schema mediation for internet scale data integration," in *IEEE 23rd International Conference on Data Engineering*, Istanbul, Turkey, Apr. 2007, pp. 186–195. doi: 10.1109/ICDE.2007.367864.
[19] K. Saleem, Z. Bellahsene, and E. Hunt, "PORSCHE: performance oriented schema mediation," *Information Systems*, vol. 33, no. 7, pp. 637–657, Nov. 2008. doi: 10.1016/j.is.2008.01.010.
[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference on Learning Representations*, Scottsdale, USA, May 2013.

Manuscript received: 2015-08-26

Biographies

Ding Xia (847525974@qq.com) is a postgraduate of Department of Information Science and Technology, Peking University, China. His research interests including ubiquitous computing.

Da Cui (443021181@qq.com) is a postgraduate of Department of Information Science and Technology, Peking University, China. His research interests including ubiquitous computing.

Jiangtao Wang (jiangtaowang@pku.edu.cn) is a postdoc researcher of Department of Information Science and Technology, Peking University, China. His research interests including mobile crowdsensing and ubiquitous computing.

Yasha Wang (wangyasha@pku.edu.cn) , PhD, is a professor of National Engineering and Research Center of Software Engineering, Peking University, China. His research interests including software reuse, data analytics, ubiquitous computing.

Call for Papers

ZTE Communications Special Issue on
Multiple Access Techniques for 5G

5G mobile cellular networks are required to provide the significant increase in network throughput, cell-edge data rate, massive connectivity, superior spectrum efficiency, high energy efficiency and low latency, compared with the currently deploying long-term evolution (LTE) and LTE-advanced networks. To meet these challenges of 5G networks, innovative technologies on radio air-interface and radio access network (RAN) are important in PHY design. Recently, non-orthogonal multiple access has attracted the interest of both academia and industry as a potential radio access technique. The upcoming special issue of *ZTE Communications* will focus on the cutting-edge research and application on non-orthogonal multiple access and related signal processing methods for the 5G air-interface. The expected publication date is July 2016. Topics related to this issue include, but are not limited to:

- Non-orthogonal multiple access (NOMA)
- Filter bank multicarrier (FBMC)
- Generalized frequency division multiplexing (GFDM)
- Faster than Nyquist (FTN) transmissions
- Signal detection and estimation in NOMA
- Resource allocations for 5G multiple access
- Cross-layer optimizations of NOMA
- Design and implementation on the transceiver architecture.

ZTE Communications (http://www.zte.com.cn/magazine/English) is a peer-reviewed international technical journal ISSN (1673-5188) and CODEN (ZCTOAK). It is edited, published and distributed by ZTE Corporation (http://www.zte.com.cn), a major international provider of telecommunications, enterprise and consumer technology solutions for the Mobile Internet. The journal focuses on hot topics and cutting-edge technologies in ICT. It has been listed in Inspec, Cambridge Scientific Abstracts (CSA), and Ulrich's Periodicals Directory. *ZTE Communications* was founded in 2003 and has a readership of 5500. It is distributed to telecom operators, science and technology research institutes, and colleges and universities in more than 140 countries.

Paper Submission:

Please directly send to j.yuan@unsw.edu.au and copy to all guest editors, with the subject "ZTE-MAC-Paper-Submission".

Tentative Schedule:

Paper submission due: March 31, 2016;
Review complete: June 15, 2016;
Final manuscript due: July 31, 2016.

Guest Editors:

Prof. Jinhong Yuan, University of New South Wales, Australia (j.yuan@unsw.edu.au)

Dr. Jiying Xiang, ZTE Corporation, China (xiang.jiying@zte.edu.cn)

Prof. Zhiguo Ding, Lancaster University, UK (z.ding@lancaster.ac.uk).

Dr. Liujun Hu, ZTE Corporation, China (hu.liujun@zte.com.cn)

Dr. Zhifeng Yuan, ZTE Corporation, China (yuan.zhifeng@zte.com.cn)