A Generic Algorithm for Heterogeneous Schema Matching

You Li[1], Dongbo Liu[2,3], and Weiming Zhang[1]

[1]Department of Management Science, National University of Defense Technology, Changsha, China 410073

[2]College of Computer Science&Technology, Huazhong University of Science&Technology, Wuhan, China, 430000

[3]Institute of China Electric System Engineering, Beijing, China 100039

liyou76@yahoo.com

Abstract

Schema matching is a basic problem nowadays in many application areas, such as data integration, data warehouse and e-business. In this paper, we propose a generic schema matching method called GSM (*Generic Schema Matching*) and its optimizing approaches. GSM provides an extensible library of match algorithms to support multi-strategy matching approach. It also uses a mapping knowledge base to leverage previous matching experiences. Finally, we use GSM on the real world schemas and provide its experimental results.

Keyword: Schema, mapping, match.

I. Introduction

The progress of information and communication technologies has made accessible a large amount of information stored in different application-specific databases and web sites. The number of different information resources is rapidly increasing, and the problem of semantic heterogeneous is becoming more and more severe [1]. *Schema matching* is the task of finding semantic correspondences between elements of two schemas [2]. It plays a central role to solve the problem of *semantic heterogeneity*.

Obviously, manually specifying schema matches is a tedious, time-consuming, error-prone, and therefore expensive process. In web-based applications, such a manual approach is a major limitation due to the rapidly increasing number of data sources [3]. For example, a recent project at the GTE telecommunications company sought to integrate 40 databases that have a total of 27,000 elements (i.e. attributes of relational tables). The project planners estimated that, without the database creators, just finding and documenting the semantic mappings among the elements would take more than 12 person years [4]. Hence a faster and less labor-intensive integration approach is needed. This requires automated support for schema matching.Solutions that try to provide some

automatic support for schema matching have received steady attention over the years. The proposed techniques for automating schema matching exploit various types of schema information, e.g. element names, data types and structural properties as well as characteristics of data instances [3]. Various systems and approaches have been developed to determine schema matches semi-automatically.

Cupid [5] represents a sophisticated hybrid match approach combining a name matcher with a structural match algorithm, which derives the similarity of elements based on the similarity of their components hereby emphasizing the name and data type similarities present at the finest level of granularity (leaf level).

LSD (Learning Source Description)[6] and its extension GLUE represent powerful composite approaches to combining different matchers. Both use machine-learning techniques for individual matchers and an automatic combination of match results. Machine learning is a promising technique especially for evaluating data instances to predict element similarity. On the other hand, the accuracy of the predictions depends on a suitable training, which can incur a substantial manual effort.

COMA [2] system makes use of stored mappings. Given two schemas *S1* and *S2* that are to be matched, COMA's reuse component looks for a schema *S* in its reuse library for which it has stored matches between *S* and *S1*, and between *S* and *S2*. These stored results are combined to produce a new match.

SF (Similarity Flooding)[7] converts schemas into labeled graphs and uses fix-point computation to determine correspondences of *1*:*1* local and *m*:*n* global cardinality between corresponding nodes of the graphs. The algorithm has been employed in a hybrid combination with a simple name matcher, which suggests an initial element-level mapping to be fed to the structural SF matcher.

The SemInt match prototype [8] creates a mapping between individual attributes of two schemas. It exploits up to 15 constraint-based and 5 content-based matching criteria. SemInt uses neural networks to determine match candidates in its approach.

Perhaps the key conclusion from these research is that an effective schema matching method requires a combination of many matching techniques, such as linguistic matching of names of schema elements, comparison of their data instances, considering structural similarities between schemas, and using domain knowledge and user feedback.

In this paper, we introduce a generic schema matching method called GSM (*Generic Schema Matching*) and its optimizing approaches. GSM provides an extensible library of match algorithms to support multi-strategy matching approach. It also uses mapping knowledge base to leverage previous matching experiences. Finally, we use GSM on the real world schemas and provide its experimental results.

II. Architecture of the GSM

Figure 1 shows the architecture of GSM, which mainly includes five components: matcher library, similarity estimator, mapping selector, match quality valuator and mapping knowledge base. Match processing can take place in either interactive or automatic mode. In interactive mode, the user can interact with GSM to specify the match strategy (selection of matchers, of strategies to combine individual match results), define match or mismatch relationships, and accept or reject match candidates. In automatic mode, the match process consists of a single match iteration for which a default strategy is applied or strategy specified by input parameters.

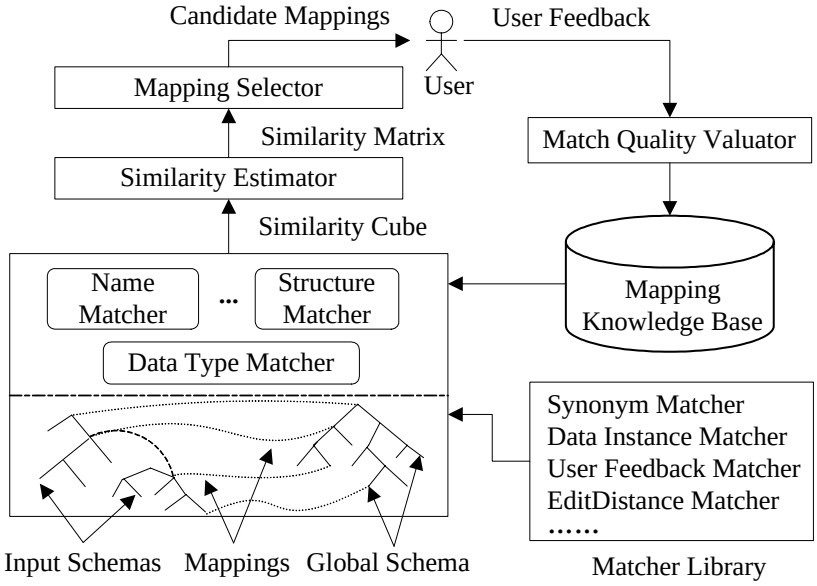

Figure 1 Architecture of the GSM

A. *Matcher Library*

GSM provides an extensible matcher library to support multi-strategy match approach. Different matchers use different match strategies and execute independently. New matchers can be easily included in the library and used.

Definition 1. Let S_1 and S_2 be two heterogeneous source schemas. For $\forall e_i \in S_1, e_j \in S_2$ the degree of similarity between e_i and e_j, which is represented as $\mu(e_i, e_j)$, can be measured by a numeric value in [0,1], i.e., $\mu(e_i, e_j): S_1 \times S_2 \to [0,1]$. $\forall e_i \in S_1, e_j \in S_2$, $\mu(e_i,e_j)= \mu(e_j, e_i)$.

During the match process, the matcher library uses the matchers to figure out the similarities of all possible matches. For the schema S_1 and S_2, let $X^k = [x_{ij}^k]_{|S_1|\times|S_2|}$ be the similarity matrix, where x_{ij}^k is used to denote the degree of similarity between $e_i \in S_1$ and $e_j \in S_2$, which is estimated by the k-th matcher. e_i and e_j are considered to be similar by the k-th matcher if x_{ij}^k is not less than the selection threshold θ.

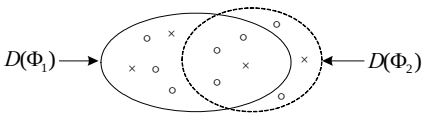

Figure 2 Overlapping of $D(\Phi_1)$ and $D(\Phi_2)$

Some matchers that use similar match strategies will cause overlapping problem. For example, considering the match results of a name matcher (including synonym match) and simply a synonym matcher will largely overlap, we can only use the name matcher in the match process. Therefore, the match library must effectively manage all the matchers and eliminate the redundant similar matchers. But with the increments of matchers and the extension of schema scales, manually deciding the overlap degree of match results is very difficult. GSM provides automatic matcher reducing mechanism to optimize the matcher library. After testing the overlap degree of match results, GSM automatically reduces the overlapping matchers. For

example, Figure 2 shows the results of matcher Φ_1 and matcher Φ_2 largely overlap. $D(\Phi_1)$ (or $D(\Phi_2)$) denotes the matches derived by the matcher Φ_1 (or Φ_2). Symbol "$\circ$" denotes the real matches, "$\times$"denotes the negative matches. We defines the parameter λ as follows, which is used to denote the overlap degree of two matchers:

$$\lambda = \max\left(\frac{|D(\Phi_1) \cap D(\Phi_2)|}{|D(\Phi_1)|}, \frac{|D(\Phi_1) \cap D(\Phi_2)|}{|D(\Phi_2)|}\right) \quad \textbf{(1)}$$

Let λ_0 be the threshold. If $\lambda \geq \lambda_0$, then we take for granted that Φ_1 and Φ_2 are similar. GSM remove the matcher with lower match quality.

B. *Mapping Knowledge Base*

Multi-strategy approach is feasible for small schemas, but does not scale to models with tens of thousands of concepts. For example, when the matcher library uses five matchers to match the schema S_1 with 1000 elements and S_2 with 1000 elements, we will get a $10^3 \times 10^3 \times 5 = 5 \times 10^6$ similarity cube. But in the case of one to one mapping, there at most exists 1000 real matches.

Furthermore, taking into account of the variety of schema elements, different match strategies may be effective for different schema elements. For example, *name matcher* may do better to some nouns, while *data_type matcher* may be more precise for numeric elements. Therefore, selecting different matchers for different elements instead of using all matchers will improve match efficiency and accuracy.

GSM uses mapping knowledge base to learn from the previous match tasks and provide the suitable match strategies. The mapping knowledge stores the weights of the matchers for *each element*. Let the weight matrix of global schema S be $W = [w_{ki}]_{n \times |S|}$, where w_{ki} is the weight of the k-th matchers for element e_i in S, n is the number of the matchers that stored in the match library. During the match process, GSM adjusts the weight matrix according to the user's feedback to make the match strategy gradually suitable for the applications.

Let R be the real matches according to user's feedback, θ be the selection threshold. The element of input schema S_1 is denoted as e_j, while the element of global schema S is denoted as e_i. Let $w_{ki}^{(t)}$ be the weight of k-th matcher for element e_i at the time t. GSM adjusts the corresponding weight $w_{ki}^{(t+1)}$ at the time $t+1$ as follows:

1) For the real matches, i.e., $\forall (e_i, e_j) \in R$, if $x_{ij}^k \geq \theta$, then we take for granted that the k-th matcher's estimation is right, and increase the corresponding weight. Otherwise, decrease it. GSM uses the following formula to adjust the weight: $w_{ki}^{(t+1)} = w_{ki}^{(t)} e^{(x_{ij}^k - \theta)}$.
2) For the derived matches that are not real matches, i.e., $\forall (e_i, e_j) \notin R \wedge x_{ij}^k \geq \theta$, GSM consider that the k-th matcher's estimation is wrong, and decrease the corresponding weight: $w_{ki}^{(t+1)} = w_{ki}^{(t)} e^{-x_{ij}^k}$.

3) At last, GSM normalize the weights. Let n be the number of the matchers stored in the matcher library, then we get: $w_{ki}^{(t+1)} = \frac{\square w_{ki}^{(t+1)}}{\sum_{k=1}^{n} \square w_{ki}^{(t+1)}}$.

During the match tasks iteration GSM adjusts the weight matrix to make the match strategy suitable for each element in global schema. To improve the match quality, it gradually increases the weights of the matchers with right judgments, decreases the weights of the matchers with wrong judgments. For each element e_i, GSM only uses the matchers whose weights are higher than the threshold w_0. By selecting different matchers for different elements instead of using all matchers, GSM will effectively improve match efficiency and quality.

C. *Similarity Estimator and Mapping Selector*

For the element e_i in global schema S and the element e_j in input schema S_1, the similarity x_{ij}^k (k=1,2,…n) generated by n matchers can be aggregated into a single similarity, which is represented as matrix $X = [x_{ij}]_{|S|\times|S_1|}$. Here x_{ij} is used to denote the similarity aggregation of the element pair $(e_i, e_j), e_i \in S, e_j \in S_1$.

To aggregate matcher-specific similarity values for every element pair, we use the weighted sum of similarity values of the individual matchers. The weights stored in the mapping knowledge base are corresponding to the expected importance of the matchers: $x_{ij} = \sum_{k=1}^{n} w_{ki} x_{ij}^k$.

Finally, all the element pairs showing a similarity in X exceeding a given selection threshold value θ are selected to construct the candidate mapping set D.

D. *Match Quality Valuator*

Comparing the automatically derived matches with the real matches results in the sets shown in Figure 3 that can be used to define quality measure for schema matching [7]. In particular, the set of derived matches is comprised of B, the *true positives*, and C, the *false positives*. *False negatives* A are matches needed but not automatically identified, while false positives are matches falsely proposed by the automatic match operation. *True negatives*, G, are false matches, which have also been correctly discarded by the automatic match operation.

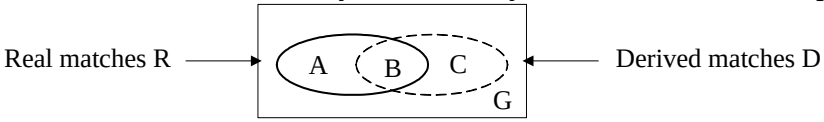

Figure 3 Comparing real matches and automatically derived matches

Based on the cardinality of these sets, two common measures, *Precision* and *Recall*, which actually originate from the information retrieval field, can be computed. However, neither *Precision* nor *Recall* alone can accurately assess the match quality. In particular, *Recall* can easily be maximized at the expense of a poor *Precision* by returning all possible mappings. On the other side, a high *Precision* can be achieved at the expense of a poor *Recall* by returning only few (correct) mappings.

$$Precision = \frac{|B|}{|B|+|C|} \quad (2)$$

$$Recall = \frac{|B|}{|A|+|B|} \quad (3)$$

Hence it is necessary to consider both measures. GSM uses the *F-measure* to measure the match quality, which also stems from the information retrieval field. The intuition behind this parametrized measure ($0 \le \alpha \le 1$) is to allow different relative importance to be attached to *Precision* and *Recall*.

$$F-measure(\alpha) = \frac{|B|}{(1-\alpha)*|A|+|B|+\alpha*|C|} = \frac{Precision*Recall}{(1-\alpha)*Precision+\alpha*Recall} \quad (4)$$

III. Schema Mapping Using GSM

GSM flexibly combines several matchers. It can be used not only in data integration application, which has global schema, but also in generic schema mapping tasks.

A *data integration system I* is a triple (S, $\{S_i\}$, $\{M_i\}$), where S is a target schema, $\{S_i\}$ is a set of source schemas, and $\{M_i\}$ is a set of source-to-target mappings, such that for each source schema S_i there is a mapping set M_i from S_i to S.

In the data integration application, GSM is used to fulfill the task of schema matching between the global schema S and several local data source schema $\{S_i\}$. The mapping knowledge base constructs the weight matrix W for every element in global schema S. By learning the experiences from the previous mapping tasks, GSM adjusts the marchers' weights to make the match strategies applicable to each element.

For the generic match task, which matches two given schemas S_1 and S_2, GSM assumes a transitive nature of the similarity relation between elements [2], i.e. if a is similar to b and b to c, then a is (very likely) also similar to c. Of course wrong match candidates may be determined in cases where the transitivity property does not hold.

A common approach to determine the transitive similarity is to multiply the individual similarity values. This approach, however, may lead to rapidly degrading similarity values. For instance, for $contactFirstName \xleftrightarrow{0.5} Name \xleftrightarrow{0.7} firstName$, the similarity between *concactFirstName* and *firstName* would become 0.5*0.7=0.35, which is unlikely to reflect the similarity, which we would expect for the two names. To calculating transitive similarities, we thus prefer the alternatives for combining the results, which will return the average similarity. In the example the approach will result in similarity value 0.6. Given two schemas *S1* and *S2* that are to be matched, GSM uses the same match strategy stored in the mapping knowledge base. It first matches the input schema *S1* and *S*, and then *S2* and *S* to get the similarity matrix respectively. We get the aggregated similarity matrix by using the following formula:

$$\forall e_i \in S_1, e_j \in S_2,\ \ \mu(e_i, e_j) = \frac{1}{2}\max\left\{\mu(e_i, e_k') + \mu(e_i, e_k') \middle|\ \ e_k' \in S\right\} \quad (5)$$

IV. Evaluation on Real World Schemas

To evaluate the performance of the algorithm for schema matching tasks, we use GSM to find mappings in the travel domain. In our experiment, we first build a XML Schema as the global schema *S* and use GSM to match *S* with the relational schemas, which are extracted from six travel agencies and constructed independently. For short, we refer to them as 1,2,3,4,5,6. In the task 7 and task 8, GSM respectively matches *S* with the schemas of directory categories Directory>Recreation>Travel in *Yahoo* and *Google*. Currently, we have developed four matchers in the matcher library, a *name matcher*, a *synonym matcher*, a *dataType matcher* and a *dataInstance matcher* [3]. We use GSM and the SMC (*Simply Matcher Combination*) method that simply uses several matchers without mapping knowledge base. Table 1 shows the characteristics of two methods.

Table 1 The characteristics of two methods

Method	Matcher Selection	Using Knowledge Base	Combination Strategy
GSM	Select matchers for each element	Yes	Weighted Sum
SMC	Use all matchers for each schema	No	Average

We show the match performance in Figure 4 and Figure 5 respectively. The two methods use the same matcher library with the same initial state.

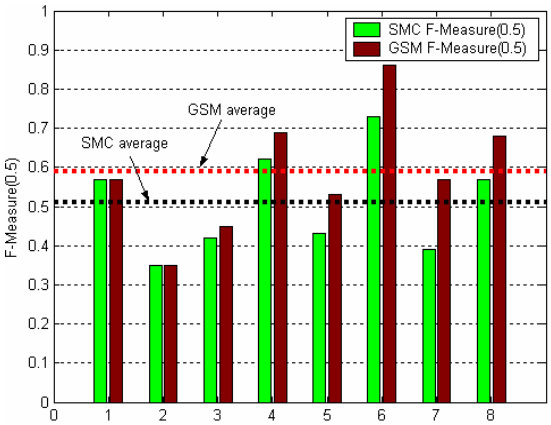

Figure 4 Match quality of GSM and SMC

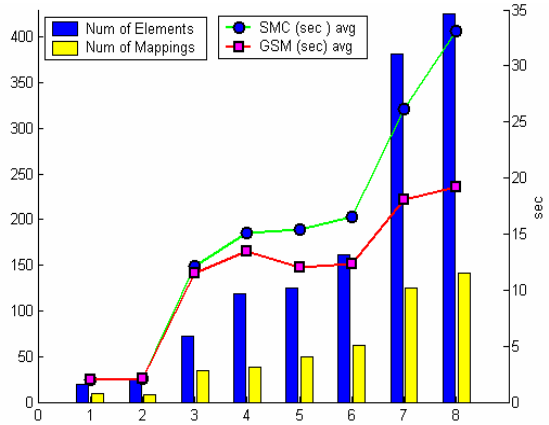

Figure 5 Time GSM and SMC needed to perform matching on different schemas

We can learn from the experiment that in the first three match tasks, the performance of SMC and GSM are similar. But with the increasing number of match tasks (task 4,5,6), the performance of GSM is better than SMC for it chooses the matchers for each element instead of for each schema. Moreover, GSM shows better performance than SMC when the scale of data source schema enlarged.

V. Conclusions

GSM is a generic schema matching method, which including two main components *matcher library* and *mapping knowledge base*. It uses the extensible matcher library to support an efficient multi-strategy match approach. By using the mapping knowledge base, GSM learns from the previous match experiences that make its match strategy gradually suitable for each schema element.

Future wok will focus on extending the match library and improving the learning capability. We also try to provide a friendly interface to the users.

References

[1] Fausto Giunchiglia and Pavel Shvaiko: Semantic Matching. In the Knowledge Review journal, 18(3):265-280,2004.

[2] Hong-Hai Do, Erhan Rahm, COMA-A system for flexible combination of schema matching approaches. In Proc. 28th VLDB Conference.

[3] Erhard Rahm, Philip A. Bernstein. A survey of approaches to automatic schema matching. VLDB Journal 10, 2001.

[4] AnHai Doan:Learning to Map between Structured Representations of Data. PhD thesis, University of Washington.

[5] Jayant Madhavan, Philip A. Bernstein, Erhard Rahm, Generic Schema Matching with Cupid, VLDB 2001.

[6] AnHai Doan, P. Domingos, A. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In SIGMOD Record, 2001.

[7] Sergey Melink, Hector Garcia-Molina, Erhard Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In Proc. 18th Intl. Conf. On Data Engineering, San Jose CA, 2002.

[8] Li W, Clifton C, Liu S (2000) Database integration using neural network: implementation and experiences. Knowl Inf Syst 2(1): 73-96.

[9] M. Dell'Erba, O. Fodor, F. Ricci, H. Werthner:. Harmonise: A Solution for Data Interoperability. IFIP I3E 2002.

[10] Do H.-H., Melnik S., and Rahm E.: Comparison of Schema Matching Evaluations, Proc. GIWorkshop "Web and Databases", Erfurt, Oct. 2002.

You Li received her B.S. and an M.S. from National University of Defence Technology in 1999 and 2002 respectively. She is currently a Ph.D. candidate at National University of Defence Technology. Her research interests include data integration and intelligent decision-support systems.

Dongbo Liu is a professor of computer software at Institute of China Electronic System Engineering. He received his B.Sc. and M.Sc. degree in Computer Science from National University of Defence Technology. He is currently a Ph.D. candidate at College of Computer Science&Technology, Huazhong University of Science and Technology. His research interests include fuzzy logic, fuzzy database, data mining and data integration. He is a co-author of A Fuzzy PROLOG Database published by Research studies Press and John Wiley & Sons.

Zhang Weiming is a professor of Department of Management Science and Engineering at National University of Defense Technology, Changsha, P.R. China. His research mainly focuses on information system engineering and intelligent decision-support systems.