Schema Matching across Query Interfaces on the Deep Web

Zhongtian He, Jun Hong, and David Bell

School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, Belfast, BT7 1NN, UK
{zhe01,j.hong,da.bell}@qub.ac.uk

Abstract. Schema matching is a crucial step in data integration. Many approaches to schema matching have been proposed so far. Different types of information about schemas, including structures, linguistic features and data types, etc have been used to match attributes between schemas. Relying on a single aspect of information about schemas for schema matching is not sufficient. Approaches have been proposed to combine multiple matchers taking into account different aspects of information about schemas. Weights are usually assigned to individual matchers so that their match results can be combined taking into account their different levels of importance. However, these weights have to be manually generated and are domain-dependent. We propose a new approach to combining multiple matchers using the Dempster-Shafer theory of evidence, which finds the top-k attribute correspondences of each source attribute from the target schema. We then make use of some heuristics to resolve any conflicts between the attribute correspondences of different source attributes. Our experimental results show that our approach is highly effective.

1 Introduction

There are now many searchable databases on the Web. These databases are accessed through queries formulated on their query interfaces only which are usually query forms. The query results from these databases are dynamically generated Web pages in response to form-based queries. The number of such dynamically generated Web pages is estimated around 500 times the number of static Web pages on the surface Web [1]. In many domains, users are interested in obtaining information from multiple sources. Thus, they have to access different Web databases individually via their query interfaces. For large-scale data integration over the Deep Web, it is not practical to manually model and integrate these Web databases. We aim to provide a uniform query interface that allows users to have uniform access to multiple sources [2]. Users can submit their queries to the uniform query interface and be responded with a set of combined results from multiple sources automatically.

Schema matching across query interfaces is a critical step in Web data integration, which finds attribute correspondences between the uniform query interface

A. Gray, K. Jeffery, and J. Shao (Eds.): BNCOD 2008, LNCS 5071, pp. 51–62, 2008.

and the query interface for a local database. In general, schema matching takes two schemas as input and produces a set of attribute correspondences between the two schemas [3, 4]. The problem of schema matching has been extensively studied [4,5,6,7,8,9,10,11,12,13,14,15,16]. Some of these methods [9,10,12,13,14] make use of information about schemas, including structures, linguistic features, data types, value ranges, etc to match attributes between schemas.

Match results from individual matchers are not accurate and certain, because they rely on individual aspects of information about schemas only, which are not sufficient for finding attribute correspondences between schemas. Individual matchers, however can generate some degree of belief on the validity of possible attribute correspondences.

In addition, sometimes given a source attribute, there might be two or more attribute correspondences that are not clearly distinguishable from each other by an individual matcher. For example, a data type matcher may not be able to distinguish some attribute correspondences for the same source attribute if they all have the same data type as the source attribute.

Recent research efforts have been focused on combining multiple matchers. However, how to combine different measures is a difficult issue. In the example shown in Figure 1, when we use a string similarity-based matcher, the similarity value between "Published Date" and "Publisher" is greater than the one between "Published Date" and "Release Date", while when a semantic similarity-based matcher is used, the similarity values are the other way around. Current approaches use different strategies to combine matcher-specific similarity values [12].

However, these strategies sometimes do not truly reflect how well two attributes match. Given a pair of attributes, the Max strategy selects the maximal similarity value among all the similarity values from different matchers as their similarity value. For example, if one of our matchers is data type-based matcher, and the attributes, "Publisher" and "Author", have the same data type, then the similarity value is 1 which will be chosen as their final similarity value. But obviously they do not match. On the other hand, the Min strategy selects the

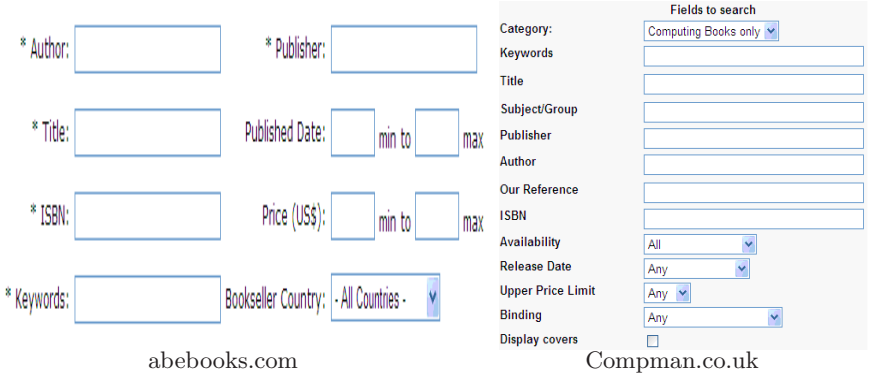

Fig. 1. Two real-world query interfaces on the Web

lowest similarity value. For example, if a string similarity-based matcher is one of the matchers, the similarity value of "Published Date" and "Release Date" is very low, but actually they are a right match. The third strategy, Average, treats all the matchers equally. For instance, if two matchers are string similarity-based matcher and semantic similarity-based matcher respectively, the average similarity value between "Published Date" and "Publisher" is higher than the one between "Published Date" and "Release Date", but we all know that "Published Date" and "Release Date" is the correct match. It appears that the semantic similarity-based matcher should have a higher level of importance. The Weighted strategy is the most popular strategy that calculates a weighted sum of the similarity values of all the individual matchers, where weights correspond to different levels of importance of the individual matchers. However, assigning weights to different matchers now becomes an issue. Weights have to be manually generated and are domain dependent.

To address these issues, we propose a new strategy for combining multiple matchers. We use four individual matchers to measure the similarity between attributes, and make use of Dempster-Shafer (DS) theory of evidence to combine the results from these matchers.

Finally, sometimes two or more different source attributes may have the same attribute correspondence. In our approach, we keep the top-k matches of each source attribute. We then use some heuristics to resolve any conflicts between the matches of different source attributes.

The rest of this paper is organized as follow. Section 2 introduces the Dempster-Shafer (DS) theory of evidence. Section 3 describes how to use DS theory in our approach. Section 4 describes how to resolve conflicts of different source attributes. In Section 5, we report our experiments on our prototype using a dataset which contains the schemas of real-world query interfaces. Section 6 compares our work with related work. Section 7 concludes the paper.

2 Dempster-Shafer Theory of Evidence (DS)

The DS theory of evidence, sometimes called evidential reasoning [17] or belief function theory, is a mechanism formalized by Shafer [18] for representing and reasoning with uncertain, imprecise and incomplete information. It is based on the modeling of uncertainty in terms of upper and lower probabilities that are induced by a multi-valued mapping rather than as a single probability value.

Definition 1. ***Frames of Discernment.*** *A frame of discernment (or simply a frame), usually denoted as Θ, contains mutually exclusive and exhaustive possible answers to a question, one and only one of which is true.*

In DS theory, a frame of discernment is used to represent a set of possible answers to a question. For example, a patient has been observed having two symptoms: "coughing" and "sniveling" and only three types of illness could have caused these symptoms: "flu"(F),"cold"(C) and "pneumonia"(P). We use a frame $\Theta = \{F, C, P\}$ to represent these types of illness.

Definition 2. ***Mass functions.*** *A function, m:* $2^{\Theta} \rightarrow [0,1]$*, is called a mass function on a frame* Θ *if it satisfies the following two conditions:*

$$m(\phi) = 0 \tag{1}$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{2}$$

where ϕ *is an empty set and* A *is any subset of* Θ*.*

Given a frame of discernment, Θ, for each source of evidence, a mass function assigns a mass to every subset of Θ, which represents the degree of belief that one of the answers in the subset is true, given the source of evidence. For example, when the patient has been observed having the symptom "coughing", the degree of belief that the patient has "flu" or "cold" is 0.6 and the degree of belief that the patient has "pneumonia" is 0.4, that is $m_1(\{C, F\}) = 0.6$ and $m_1(\{P\}) = 0.4$. Similarly, with the symptom of "sniveling", we have another mass function: $m_2(\{F\}) = 0.7$, $m_2(\{C\}) = 0.2$ and $m_2(\{P\}) = 0.1$.

When more than one mass function is given on the same frame of discernment, the theory also provides us with Dempster's combination rule. If m_1 and m_2 are two mass functions on frame Θ, then $m = m_1 \bigoplus m_2$ is the combined mass function, where $\bigoplus$ means using Dempster's combination rule, defined as follows:

$$m(C) = \frac{\sum_{A \bigcap B=C} m_1(A) m_2(B)}{1 - \sum_{A \bigcap B=\phi} m_1(A) m_2(B)} \tag{3}$$

In the above example, we combine two mass functions, m_1 and m_2, to get m($\{C\}$)=0.207, m($\{F\}$)=0.724 and m($\{P\}$)=0.069. Therefore given the two symptoms the patient has, it is more likely that he is having "flu".

3 Combining Multiple Matchers Using DS Theory

Given a source schema and a target schema, our approach combines a set of individual matchers using the Dempster-Shafter theory of evidence to produce a set of attribute correspondences between the two schemas. Our approach consists of a number of steps: 1. Applies each of the individual matchers to the two given schemas; 2. Interprets the results from the individual matchers using the DS theory; 3. Combines the results from the individual matchers using the Dempster's combination rule to produce the top k attribute correspondences of each source attribute; 4. Decides on the attribute correspondence of each source attribute and resolves conflicts between attribute correspondences of two or more source attributes.

3.1 Individual Matchers

We use four individual matchers, the first three matchers are based on the linguistic features of attribute names and the last matcher uses the data types of attributes.

Semantic Similarity: We use WordNet[1], an ontology database to compute the semantic similarity between two words. We use the traditional edge counting approach to measuring word similarity. We define similarity between two words as $S(w_1, w_2) = 1/L$, where L is the shortest path in WordNet between these two words. Suppose that two attribute names have two sets of words $S_1 = \{w_1, w_2, ..., w_m\}$ and $S_2 = \{w'_1, w'_2, ..., w'_n\}$. We compare the similarity values between each word in S_1 with every word in S_2 and find the highest semantic similarity value. We then get a similarity value set for S_1: $Sim_1 = \{s_1, s_2, ..., s_m\}$. Using the same method we get a similarity value set for S_2: $Sim_2 = \{s'_1, s'_2, ..., s'_n\}$. From these two similarity value sets we calculate the similarity value between two attribute names S_1 and S_2 as:

$$Sim(S_1, S_2) = \frac{\sum_{i=1}^{m} s_i + \sum_{i=1}^{n} s'_i}{m + n} \tag{4}$$

where m is the number of the words in S_1, n is the number of the words in S_2.

Edit Distance-Based Matcher: Edit distance is the number of edit operations necessary to transform one string to another [19]. We define the edit distance-based string similarity as follows:

$$sim_{ed}(w_1, w_2) = \frac{1}{1 + ed(w_1, w_2)} \tag{5}$$

where w_1 and w_2 are two words, $ed(w_1, w_2)$ is the edit distance between these two words. Similar to the semantic similarity matcher, we get two similarity value sets for two attribute names first and then calculate the similarity value between two attribute names based on the two similarity value sets using the formula defined in (4).

Jaro Distance: Similar to the edit distance matcher, we use the formula in (4) to calculate the similarity value between two attribute names, where the similarity value between two words is calculated using the Jaro distance instead. The Jaro distance measures the similarity of two strings based on the number and order of the common characters between them. Given two strings $s = a_1 \cdots a_k$ and $t = b_1 \cdots b_l$, a character a_i in s is in common with t, if there is a $b_j = a_i$ in t such that $i - H \leq j \leq i + H$, where $H = \frac{min(|s|,|t|)}{2}$. Let $s' = a'_1 \cdots a_{k'}$, be the characters in s which are in common with t (in the same order they appear in s) and $t' = b'_1 \cdots b_{l'}$ similarly. We define a *transposition* for $s' and t'$ to be a position i such that $a'_i \neq b'_i$. Let $T_{s,t}$ be half the number of transpositions of s' and t'. The Jaro similarity metric for s and t is defined as follows [21]:

$$Jaro(s, t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right) \tag{6}$$

Data Types: As discussed in [8], we define that two data types are compatible if they are the same or one subsumes another (is-a relationship). Currently we focus

[1] http://wordnet.princeton.edu/

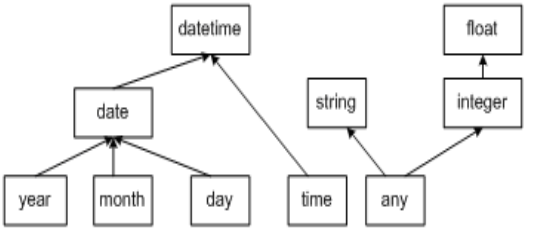

Fig. 2. Data types and their relationships

only on the data types that are shown in Figure 2. For example, in Figure 2, an "integer" is a "float". So we say that their data types are the same or compatible (incidentally, "null" is compatible to any data type). The similarity value between two attribute names is 1, if their data types are the same. Otherwise it is 0.

3.2 Interpreting Results from Individual Matchers

Assume that we have a source schema, $S = \{a_1, a_2, ..., a_m\}$, where a_i, for $i = 1, 2, ..., m$, is a source attribute, and a target schema, $T = \{b_1, b_2, ..., b_n\}$, where b_j, for $j = 1, 2, ..., n$, is a target attribute. For each source attribute, a_i, we have a set of possible correspondences in the target schema $\{< a_i, b_1 >, < a_i, b_2 >, ..., < a_i, b_l >\}$. It is also possible that a_i, may have no correspondence in the target schema at all. We therefore have a frame of discernment for a_i, $\Theta = \{< a_i, b_1 > , < a_i, b_2 >, ..., < a_i, b_l >, < a_i, null >\}$, where $< a_i, null >$ represents that there is no correspondence for a_i in the target schema. This frame of discernment contains an exclusive, exhaustive set of answers (see Definition 1) to the question of finding an attribute correspondence for a_i, in the target schema. One and only one of these answers is true.

Generating Indistinguishable Subsets of Attribute Correspondences. For some matchers we cluster Θ into a set of indistinguishable subsets, because some attribute correspondences may not be distinguishable. For example, if a source attribute has the same data type with two target attributes, then the two correspondences cannot be distinguished from each other, so we cluster these indistinguishable correspondences into a subset.

Generating Mass Distributions on Indistinguishable Subsets. We now describe how to generate a mass distribution that assigns a mass to an indistinguishable subset of Θ, on the basis of the similarity measures on the attribute correspondences in the subset.

Given an indistinguishable subset of attribute correspondences, we have a similarity value for each correspondence, which represents how well the two attributes in the correspondence match according to the criterion used by the matcher. Suppose the subset is $\{< a_i, b_{i1} >, < a_i, b_{i2} >, ..., < a_i, b_{il} >\}$, a mass

assigned to the subset is calculated based on the similarity values for all the attribute correspondences in the subset as follows:

$$m'(A) = 1 - \Pi_{j=1}^{l}(1 - Sim(a_i, b_{ij})) \tag{7}$$

where $Sim(a_i, b_{ij})$ is the similarity value for one of the correspondences in the subset. For the special singleton subset, $\{< a_i, null >\}$, since we do not have a similarity value for it by any matcher, the mass assigned to the subset is calculated as follows:

$$m'(\{< a_i, null >\}) = \Pi_{j=1}^{l}(1 - Sim(a_i, b_{ij})) \tag{8}$$

The mass distributed to $\{< a_i, null >\}$, therefore, represents the degree of belief that none of the target attributes is the attribute correspondence of source attribute, a_i.

DS theory requires that the sum of all masses assigned to every indistinguishable subset equals to 1. We scale the mass distribution, m', by the following formula:

$$m(A) = \frac{m'(A)}{\sum_{B \subseteq \Theta} m'(B)} \tag{9}$$

where A and B are subsets of Θ. The mass distribution produced by (9) assigns a mass to every indistinguishable subset of Θ, which represents the degree of belief by the matcher that the attribute correspondence of the source attribute, a_i, belongs to the subset.

3.3 Combining Mass Distributions from Multiple Matchers

We now have a mass distribution by each of the individual matchers, which assigns a mass to every indistinguishable subset of Θ. A mass distribution can be seen as an opinion expressed by a matcher on the degree of belief that the attribute correspondence of the source attribute belongs to an indistinguishable subset. Using Dempster's combination rule, we can take into account different opinions by different matchers by combining the mass distributions by these matchers. The mass distribution produced after this is used to select the top k attribute correspondences of each source attribute.

4 Resolving Conflicts between Attribute Correspondences

We have now the top k attribute correspondences of each source attribute. However, these attribute correspondences have so far been selected for an individual source attribute only. There might be conflicts between attribute correspondences of two or more source attributes (ie. the best correspondences of two different source attributes are the same target attribute). To resolve any conflicts that may arise between attribute correspondences, the attribute correspondences of source attributes are collectively selected to maximize the sum of all the

masses on the attribute correspondence of every source attribute. The algorithm is given in Algorithm 1.

For example, suppose that both source and target schemas have three attributes, the source attributes are $\{Author, Publisher, Published\ Date\}$, and the target attributes are $\{Author, Keywords, Release\ Date\}$. We have the top k ($k = 3$) correspondences of each source attribute: { m(<*Author,Author*>) = 0.88, m(<*Author,null*>) = 0.11, m(<*Author,Keywords*>) = 0.01 }, { m(<*Publisher,Author*>) = 0.47, m(<*Publisher,null*>) = 0.40, m(<*Publisher, Keywords*>) = 0.13}, { m(<*Published Date,Release Date*>) = 0.87, m(<*Published Date,null*>) = 0.13, m(<*Published Date, Author*>) = 0.0 }. The top attribute correspondence of "Author", <*Author,Author*>, is in conflict with the top correspondence of "Publisher", <*Publisher,Author*>. By using our algorithm, { <*Author,Author*>,<*Publisher,null*>,<*Published Date,Release Date*> } has the maximum sum of mass function values.

Algorithm 1. Resolving Conflicts

Input: A set of all the possible combinations of attribute correspondences $\Omega = \{C|C = \{< a_1, b'_1 >, < a_2, b'_2 > ... < a_m, b'_m >\}\}$,where $< a_i, b'_i > \in \{< a_i, b_{i1} >, < a_i, b_{i2} >, ..., < a_i, b_{ik} >\}$ (the top $k$only correspondences of a_i)

Output: A collection of correspondences with the maximum sum of the mass values of the correspondences for every source attribute

1: $Max \leftarrow 0; Best \leftarrow null.$
2: **for** each C$\in \Omega$ **do**
3: $Sum = \Sigma_{i=1}^{m} m(< a_i, b'_i >)$, where $m(< a_i, b'_i >)$ is the mass function value of $< a_i, b'_i >$
4: **if** $Sum > Max$ **then**
5: $Max \leftarrow Sum; Best \leftarrow C;$
6: **end if**
7: **end for**
8: **return** Best

5 Experimental Results

5.1 Dataset

To evaluate our approach, we have selected a set of query interfaces on the real-world websites from the ICQ Query Interfaces dataset at UIUC, which contains manually extracted schemas of 100 interfaces in 5 different domains: Airfares, Automobiles, Books, Jobs, and Real Estates. In this paper we have focused on 1:1 matching only, so only 88 interfaces are chosen from the dataset. In each domain we choose an interface as the source interface, and use others as the target ones.

5.2 Performance Metrics

We use three metrics: precision, recall, and F-measure [20, 10, 22]. Precision is the percentage of correct matches over all matches identified by a matcher.

Table 1. The precisions of different matchers

	Edit distance	Jaro distance	Semantic similarity	Our matcher
Airfares	83.3%	56.8%	86.4%	92.0%
Automobiles	84.4%	48.1%	93.1%	96.3%
Books	87.0%	48.8%	92.0%	94.4%
Jobs	68.5%	50.0%	71.0%	91.9%
Real Estates	86.8%	52.9%	81.6%	93.8%
Average	82.1%	51.3%	84.8%	93.7%

Recall is the percentage of correct matches by a matcher over all the matches by domain experts. F-measure is the incorporation of precision and recall. In our approach the "no match" ("null") is also an answer to the source attribute. So it is always possible to find a correspondence to each source attribute, that is, the number of matches by our approach equals to the number of matches identified by experts. So in our approach, precision and recall are both the same, and we use precision only.

5.3 Discussion on Experimental Results

First, for each domain we perform four experiments, we use three individual matchers: edit distance, Jaro distance and semantic similarity (the data type matcher cannot be used alone) separately to find matches between the source and target schemas, and compare their results with our new approach. In Table 1, we can see that the precisions of individual matchers and our approach. Our matcher gets an average precision of "93.7%" which is much higher than individual matchers.

Second, we compare our results with the work in [9], which also uses the same dataset for their experiments. However, in [9], they not only focus on 1:1 matching but also handle 1:m matching. In their experiments, a 1:m match is counted as m 1:1 matches. So we can only roughly compare our approach with their work.

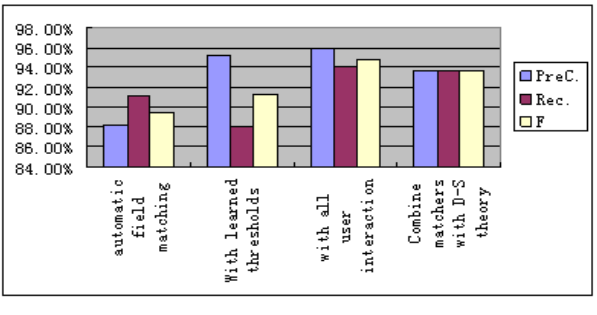

Fig. 3. Precision, recall and K measure of different matchers

As we discussed in section 5.2, precision is the same as recall in our experiments. According to the definition of F-measure in [9], $F = \frac{2PR}{P+R}$, where P is precision and R is Recall, the F-measure is also the same as precision. In [9] they did three experiments, the first one is on automatic matching which used a weighted strategy to combine multiple matchers and a 0 threshold is used to select the combined match results. The second experiment is almost the same as the first one but the threshold is obtained by training. The last one allowed user interaction. As shown in Figure 3, we can see that without training (learned threshold), the results of our approach are better. When they use the learned threshold, their precision is better than ours, but we have higher recall and F-measure. Finally, when user interactions are allowed in their approach, their results are better than ours. So we can see, our approach is effective and accurate for an automatic schema matching across query interfaces without training and user interaction.

6 Related Work

Many approaches have been proposed for automatic schema matching [9, 10, 12, 13, 14, 15, 16]. We relate our work to the existing works in two kinds of matching methods.

First, like in our approach, strategies have also been proposed in some approaches [10, 12] to combine multiple matchers. Cupid [10] considers linguistic similarity and structural similarity between elements and uses a weighted formula to combine these two similarities together. The weighted strategy is the most popular strategy used in combining individual matchers. However, weights have to be manually generated and are domain dependent.

COMA [12] also does 1:1 matching, and combines individual matchers in a flexible way. It allows users to tailor match strategies by selecting the match algorithms and their combination for a given match problem. It also allows users to provide feedback which can help improve match results. In this system, several aggregation strategies have been provided for users to choose. They are Max, Min, Average and Weighted strategies. As we discussed in Section 1,these strategies are effective in some situations while sometimes they cannot combine results well, and choosing strategies by users involves human efforts.

In [9], they also used weights to combine multiple matchers. However, they used clustering to find attribute correspondences across multiple interfaces. A threshold is required for the combined match results, which needs to be manually generated and is domain dependent. So this approach also involves human effort.

Second, some approaches [6, 8] do not combine multiple matches. MGS [6] and DCM [8] depend on the distribution of attributes rather than linguistic or domain information. Superior to other schema matching approaches, these approaches can discover synonyms by analyzing the distribution of attributes in the given schemas. However, they work well only when a large training dataset is available, and this is not always the case.

7 Conclusions and Future Work

In this paper we proposed a new approach to combining multiple matchers by using the Dempster-Shafer theory of evidence and presented an algorithm for resolving the conflicts among the correspondences of different source attributes. In our approach, different matches are viewed as different sources of evidence, and mass distributions are defined on the basis of the match results from these matchers. We use Dempster's combination rule to combine these mass dustributions, and choose the top k correspondences of each source attribute. Conflicts between the correspondences of different source attributes are finally resolved. We have implemented a prototype and tested it using a large dataset that contains real-world query interfaces in five different domains. The experimental results demonstrate the feasibility and accuracy of our approach.

We have focused on one-to-one matching between schemas in this paper. In the near future we will extend our approach to complex matching. There are more issues on uncertainty in complex matching, such as how many groups the attributes in a schema should be divided into, and which group should contain a specific attribute. Using uncertainty theory to address these issues could be feasible and effective.

References

1. Bergman, M.K.: The deep web: Surfacing hidden value. BrightPlanet (2001)
2. Dragut, E.C., Yu, C.T., Meng, W.: Meaningful labeling of integrated query interfaces. In: Proceedings of the 32th International Conference on Very Large Data Bases (VLDB 2006), pp. 679–690 (2006)
3. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal of Data Semantics, 146–171 (2005)
4. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 334–350 (2001)
5. He, B., Tao, T., Chang, K.C.C.: Clustering structured web sources: A schema-based, model-differentiation approach. In: Proceedings of the joint of the 20th International Conference on Data Engineering and 9th International Conference on Extending Database Technology (ICDE/EDBT) Ph.D. Workshop, pp. 536–546 (2004)
6. He, B., Chang, K.C.C.: Statistical schema matching across web query interfaces. In: Proceedings of the 22th ACM International Conference on Management of Data (SIGMOD 2003), pp. 217–228 (2003)
7. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering (ICDE 2002), pp. 117–128 (2002)
8. He, B., Chang, K.C.C., Han, J.: Discovering complex matchings across web query interfaces: a correlation mining approach. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), pp. 148–157 (2004)

9. Wu, W., Yu, C.T., Doan, A., Meng, W.: An interactive clustering-based approach to integrating source query interfaces on the deep web. In: Proceedings of the 23th ACM International Conference on Management of Data (SIGMOD 2004), pp. 95–106 (2004)
10. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001), pp. 49–58 (2001)
11. Wang, J., Wen, J.R., Lochovsky, F.H., Ma, W.Y.: Instance-based schema matching for web databases by domain-specific query probing. In: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004), pp. 408–419 (2004)
12. Do, H.H., Rahm, E.: Coma - a system for flexible combination of schema matching approaches. In: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002), pp. 610–621 (2002)
13. Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M., Vincini, M.: Information integration: The momis project demonstration. In: Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000), pp. 611–614 (2000)
14. Castano, S., Antonellis, V.D., di Vimercati, S.D.C.: Global viewing of heterogeneous data sources. IEEE Transactions on Knowledge and Data Engineering 13(2), 277–297 (2001)
15. Doan, A., Domingos, P., Levy, A.Y.: Learning source description for data integration. In: Proceedings of the 3rd International Workshop on the Web and Databases (WebDB 2000) (Informal Proceedings), pp. 81–86 (2000)
16. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: A machine-learning approach. In: Proceedings of the 20th ACM International Conference on Management of Data (SIGMOD 2001), pp. 509–520 (2001)
17. Lowrance, J.D., Garvey, T.D.: Evidential reasoning: An developing concept. In: Proceedings of the IEEE International Conference on Cybernetics and Society (ICCS 1981), pp. 6–9 (1981)
18. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
19. Hall, P., Dowling, G.: Approximate string matching. Computing Surveys, 381–402 (1980)
20. Halevy, A.Y., Madhavan, J.: Corpus-Based Knowledge Representation. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 1567–1572 (2003)
21. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence Workshop on Information Integration on the Web (IIWeb 2003), pp. 73–78 (2003)
22. van Rijsbergen, C.J.: Information Retrival. Butterworths (1979)
23. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.Y.: Learning to match ontologies on the semantic web. VLDB Journal 12(4), 303–319 (2003)