CMC: Combining Multiple Schema-Matching Strategies based on Credibility Prediction

KeWei Tu and Yong Yu

Department of Computer Science and Engineering,
Shanghai JiaoTong University, Shanghai, 200030, P.R.China
{tukw,yyu}@sjtu.edu.cn

Abstract. Schema matching, which tries to find semantic correspondences between schema elements, is a key operation in data engineering. Combining multiple matching strategies is a very promising technique for schema matching. To overcome the limitations of existing combination systems and to achieve better performances, in this paper the CMC system is proposed, which combines multiple matchers based on credibility prediction. We first predict the accuracy of each matcher on the current matching task, and accordingly calculate each matcher's credibility. These credibilities are then used as weights in aggregating the matching results of different matchers into a combined one. The credibility prediction procedure is introduced at length, and two strategies (i.e. manual rule and machine learning) are presented. Our experiments on real world schemas validate the merits of our system.

1 Introduction

There are often multiple data sources in the same or overlapping domains. These data sources are designed and evolved independently, thus having different schemas. When data sharing is required, schema matching has to be performed to overcome the heterogeneity problem. Given two schemas, schema matching finds semantic correspondences between their elements. For example, *"Address"* in one schema matches *"Location"* in the other schema because they convey the same meaning.

With the increasing request of knowledge sharing, effective and efficient schema matching is in great demand. Since manual construction of schema matching is laborious and error-prone, numerous (semi-)automatic schema matching approaches have been developed. Among them, some utilize the element label, data type and structure information of schemas [1–3], some glean information from instances [4–7], and others reuse previous matchings [8, 5, 9, 10]. It is being increasingly accepted that, to achieve better performance in schema matching, one should make use of every possible kind of information about the schemas to be matched. Therefore, it is preferred to combine multiple matching strategies into a single system.

LSD [4] and COMA [8] are the two most well-known matcher combining systems. LSD is used in schema integration, i.e. finding mappings between various

local schemas to the same mediated schema. To combine individual matchers it performs meta-learning with the *stacking* technique [11]. COMA employs quite straightforward methods such as average and maximization for combination, so that it could avoid the burden of learning. Apart from their merits, however, both LSD and COMA suffer from some limitations:

- It could be quite difficult for LSD to collect training samples for meta-learning in generic schema matching applications other than schema integration;
- The strategy employed in LSD would incur too much training in generic schema matching applications other than schema integration;
- The meta-learning method of LSD is sensitive to base-matcher addition or removal in that the meta-learner must be re-trained;
- The meta-learner of LSD does not take into account all the information relevant to the matcher combination procedure;
- The straightforward combination methods of COMA are not sophisticated enough to work well in some complex situations;
- To achieve better combination, the users of COMA have to perform manual configuration, such as specifying the weights.

A more detailed analysis of the two systems and their limitations is presented in Sect.2.

In order to overcome the limitations discussed above as well as to further enhance the performance of matcher combining, in this paper we propose the CMC (Credibility-based Matcher Combiner) system. The matcher combination procedure of CMC is based on the observation that every base matcher has quite different performance in different matching tasks. For example, a matcher exploiting schema structure information would perform well for elements from XML schemas with rich structures, but the same matcher would become unreliable when it comes to "flat" schemas. Therefore, CMC dynamically predicts the accuracy of each matcher based on the characteristics of the current matching task, and accordingly calculates the matcher's credibility. Then, the results from various base matchers are aggregated based on their credibilities. Specifically, CMC has the following virtues:

- All the information relevant to the combination is taken into account in CMC's prediction of base matchers' credibilities, leading to better matching performances as illustrated in our experiments;
- Although one of the credibility-prediction strategies employs machine learning, training samples are easy to collect and the training cost is rather small. Besides, no re-training is needed when adding or removing base-matchers;
- The combination procedure could be carried out in a fully automatic way.

The rest of the paper is organized as follows. The next section presents related work in schema matching and analyzes the limitations of previous matcher combining systems. Sect.3 gives an overview of the CMC system. It is in Sect.4 that credibility prediction is introduced in detail. Experiment results are given out in Sect.5. Finally we conclude this paper in Sect.6.

2 Related Work

So far, many schema matching techniques have been proposed, most of which can be adapted as base matchers in matcher combining systems. [3] proposes a matching algorithm called similarity flooding, which utilizes schema structure information by means of fix-point computation. The Naïve-Bayes matcher in LSD [4] learns from schema instances and makes prediction to match new elements. Automatch [5] constructs an attribute dictionary based on instances of previously mapped schemas and uses the dictionary to match new schemas. A similar approach is proposed in [9], with schema level information also exploited. [6] presents a novel technique which performs matching utilizing the mutual information of elements derived from instance statistical information. As existing mappings may also provide valuable information, [10] discusses how to compose these mappings for new ones. A simplified technique is proposed in COMA [8]. Cupid [2] and DIKE [1] first calculate element similarities based on element information (e.g. label, data type and constraint), then revises them based on schema structural information.

Several matcher combining systems have been proposed before CMC. LSD [4] and COMA [8] are two representative systems. LSD is the first system that broaches the idea of combining multiple matching strategies. It is used in schema integration, i.e. it aims to find out mappings between various local schemas to a same mediated schema. All individual matchers in LSD adopt machine learning techniques and are called *base learners*. For each pair of elements, each base learner predicts a similarity, then a *meta-learner* is applied to aggregate all the similarities by means of weighted average, where the weights are also obtained by machine learning. LSD has an extension called GLUE [12], which gives flexible similarity definition and a technique to take into account background knowledge. Unlike LSD, COMA uses schema-level matchers without a learning process, and it can employ hybrid matchers as its base matchers. The aggregation method in COMA is quite straightforward, such as average and maximization, with all possible parameters (e.g. weights) specified by users instead of by meta-learning.

Apart from their merits, LSD and COMA still suffer from some limitations. For the LSD system,

- There is one meta-learner for each element of the mediated schema, thus each element must have a training set. To include enough positive samples in the training set, LSD collects equivalent elements from other schemas through existing mappings. This is feasible in schema integration, since it is reasonable that many schemas have previously been mapped to the mediated schema. However, in other applications existing mappings to the target schema may be scarce, making the training set too hard to construct.
- Meta-learners are associated with particular schemas. As LSD deals with schema integration, i.e. all matching tasks aim at the same mediated schema, such association is all right. But for other applications where matching is performed on arbitrary two schemas, if the schemas are new to the system, then a new set of meta-learners must be trained from scratch. This

is time-consuming, especially considering that the number of meta-learners equals the number of schema elements. (However, it is possible to release such association at the cost of performance, i.e. to use one meta-learner for any schemas, as implemented in our comparison experiment introduced in Sect.5.)
- Meta-learner is sensitive to adding/removing base learner(s) in that the meta-learner must be re-trained. However, adding or removing base matchers may be necessary in many scenarios. For example, if two flat schemas are to be matched, then no structural matchers should be employed; and if the matching task has a time limit, then quick matchers like the label matcher are preferred, while matchers with iteration or with training phase are undesirable.
- The meta-learner used in LSD actually performs linear combination (i.e. weighted sum) of the results from base learners. The weights are obtained by training and kept unchanged thereafter. Hence, for each element of the mediated schema, its meta-learner weighs each base learner in a fixed way, regardless of what source element is being matched to this target element. This, however, is improper under certain circumstances. For example, if a schema with simple structure is being matched, then the weight of the structural matcher should become smaller. In other words, the weights should be determined by both the target element and the source element. This problem could be alleviated if non-linear combination is used by the meta-learner, such as the multi-layer perceptron, because in this way the base learners are actually weighed differently according to their outputs, and these outputs expose information of the source element to some extent. However, as these outputs can't fully represent the source element, the problem can't be entirely eliminated. Besides, using non-linear combination further adds to the computational burden of training meta-learners.

For the COMA system,

- To avoid the burden of training meta-learners, COMA employs simple methods to aggregate the results of base matchers, such as average and maximization. These methods, however, may be inadequate for complex situations, because in fact each base matcher has very different performance in different conditions (see results in [13]), and simple aggregation methods couldn't capture such performance variation.
- If better aggregation is needed, users of COMA have to manually choose and configure the combination methods, such as specifying weights for matchers.

In this paper the CMC system is proposed to address these problems, as well as to achieve better performance.

3 Overview of CMC

As a matcher combining system, CMC contains a set of base matchers. Most of the schema-matching techniques developed so far can be employed as base matchers. For example,

NameMatcher compares labels of schema elements to do matching.
DataTypeMatcher matches the data type of elements.
LeavesMatcher is a matcher similar to Cupid [2], which exploits the structure information of the schemas with a bias on leaf nodes.
UserFeedback is a special matcher which allow users to manually modify the results from automatic matchers.

As in LSD and COMA, all of these base matchers take two target schemas S_1 and S_2 as input, and output a similarity between 0 and 1 for each pairwise combination of S_1 elements and S_2 elements, constituting a similarity matrix of size $m \times n$, where m and n are element numbers of S_1 and S_2 respectively. Based on whether an initial similarity matrix is needed or not, base matchers can be divided into two classes, as illustrated in Fig.1. For the above examples, the former two belong to the first class and the latter two belong to the second.

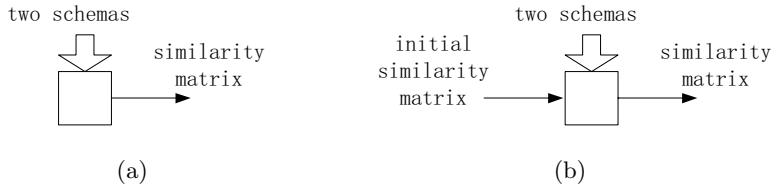
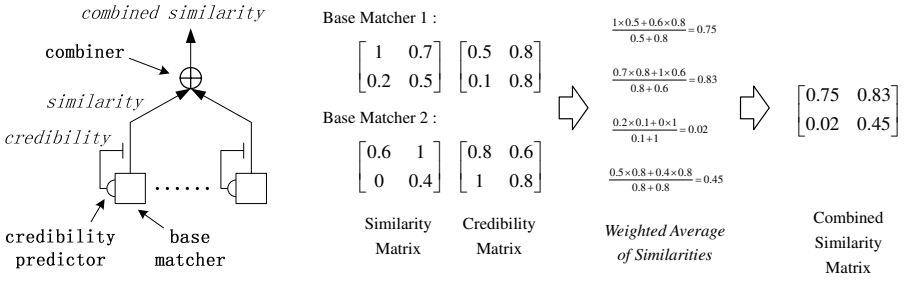


Fig. 1. Two classes of base matchers.

A key operation of CMC is the base-matcher combination process. Unlike LSD and COMA, a credibility-based approach is employed in CMC for the matcher combiner. The underlying rationale is that every base matcher performs very differently in matching different kinds of schema element pair, so the matcher combiner should take into account the anticipated performance of each base matcher for the current matching task and accordingly assign different credit on them. For instance, structure matchers are more credible if the schema elements being matched are embedded in rich structures, and DataTypeMatcher is more reliable when it says "unmatched" instead of "matched". To achieve this idea, in CMC each base matcher is attached with a credibility predictor, which dynamically predicts the matcher's credibility *for each pair of elements being matched*. In this way the combiner receives two matrices from each base matcher (i.e. the similarity matrix and the credibility matrix), then it aggregates all the similarity matrices into one matrix by weighted average, where the weights are determined by the credibility matrices. This procedure is illustrated in Fig.2. Notice that a matcher combiner itself could serve as a base matcher for another combiner.

With base matchers and combiners as modules, one could connect them freely, according to the characteristic of each base matcher and the requirement of the current task. On the other hand, CMC also provides a default connection policy for those who are not willing to do customization, which is illustrated in



(a) The Architecture (b) Sample Procedure

Fig. 2. Matcher Combiner

Fig.3. There are two layers in this default structure. The bottom layer consists of base matchers that do not require initial similarity, and a combiner aggregates their outputs. Matchers that must be initialized constitute the upper layer, with the combined similarity of the first layer serving as their initial similarity. Finally a second combiner aggregates the results from the upper layer, as well as the result of the first combiner, and output the final similarity matrix.

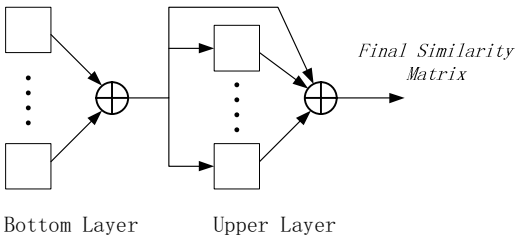


Fig. 3. The default structure.

To convert the final similarity matrix to the matching result, i.e. correspondences between schema elements, CMC adopts the method introduced in COMA. Specifically, for each element we select the element of the other schema with the best similarity value exceeding a threshold. For details about this process please refer to [8].

4 Credibility Prediction

In CMC, the credibility of a matcher indicates how much the combiner should trust the matcher. As one can see from the previous section, credibility prediction is a quite crucial operation for the CMC system. In this section, we will present the two steps in predicting the credibility of a base matcher, i.e. *accuracy*

predicting and *converting accuracy to credibility*. One important feature of this mechanism is that the prediction procedures of base matchers are independent with each other, thus the predictor of one matcher will not be affected by the addition, removal or relocation of other matchers.

4.1 Accuracy Predicting

As the first step of credibility prediction, we predict a matcher's accuracy *for each inputted pair of schema elements*.

For a specific matcher, its matching accuracy in a matching task is correlated with several features of the task (here a *matching task* means the estimating of a pair of schema elements' similarity). For example, for some structure matchers, the number of edges connected to the element to be matched can serve as a feature, because with more edges there is usually more structural information that can be utilized, leading to higher matching accuracy. With this knowledge, we predict a matcher's accuracy in the current matching task as *the mean accuracy of the set of tasks bearing the same features as the current task*. Given that the output of a matcher is a numeric similarity, the mean accuracy is defined in terms of the *mean square error* (MSE) of that set of tasks:

$$MSE = E_{\mathcal{F}}[(sim - sim_{actual})^2]$$

$\mathcal{F}$ is that set of matching tasks bearing the same features as the current task, and $E_{\mathcal{F}}$ represents the mathematical expectation on the set $\mathcal{F}$. sim_{actual} is 1 for matched element pairs and 0 otherwise. Obviously the less MSE is, the higher the accuracy is. Two strategies are presented here to estimate MSE for different kinds of matchers.

Manual Rule. For some matchers, the MSE estimation is intuitive enough to be formulated manually. For our sample matchers in Sect.3, this strategy could be applied to DataTypeMatcher and UserFeedback.

For DataTypeMatcher, the outputted similarity is the only feature correlated to the matching accuracy, and it indicates the probability that the data types of the two elements are matched. If the data types are unmatched, then these two elements can't be matched at all. If the data types match, then the elements' being matched or not could be equally possible. Therefore, $MSE = (1 - sim) \times (sim - 0)^2 + sim \times \frac{(sim-1)^2+(sim-0)^2}{2} = \frac{1}{2}sim$.

For UserFeedback, things become even simpler. For the similarities that are confirmed or modified by the user, the highest confidence should be assigned, so $MSE \to 0$. For the others, we just keep their credibility unchanged and skip this prediction procedure.

Notice that the matcher combiner of CMC could also be regarded as a base matcher, so its accuracy must be calculated as well. With the formulas from [14], we could formulate the MSE of a matcher combiner as follows, under the

assumption that base matchers are uncorrelated.

$$MSE = \sum_{i,j} w_i w_j C_{ij} = \sum_i w_i^2 C_{ii} = \sum_i w_i^2 MSE_i$$

Here w_i is the weight of the i-th matcher, and C_{ij} is the correlation between the i-th and the j-th matcher, which is zero under our assumption if $i \neq j$, and equals MSE_i otherwise.

Learning to Predict. For most schema matchers, it is difficult, if not impossible, to manually formulate the MSE calculation. Examples include two of our sample matchers, NameMatcher and LeavesMatcher. Therefore, we use machine-learning techniques for the estimation, i.e. learning to predict MSE from the features of the current matching task.

It is important to select appropriate features of matching tasks for each matcher, and the feature set should include all the possible factors that may affect the matching accuracy.

- For NameMatcher, longer labels often convey more information, so the label lengths of the two elements to be matched are two features. Likewise, the number of words in the label could also be regarded as a feature.
- For LeavesMatcher, the number of leaves of an element node would influence the effectiveness of this matcher, as well as the depth of that node. Another important feature is the credibility of the initial similarity, which will doubtless affect this matcher's performance.
- The outputted similarity is also a useful feature for most matchers, because matchers often have different reliability on different outputs.

With features selected, we train a learner which takes the values of the features as the input and output the estimated MSE. Any existing schema matchings can be used to construct the training set.

A straightforward method to construct the training set is, for each existing schema matching: (1) Run the base matcher on it to get a similarity matrix, so that the squared error of each similarity can be computed. (2) Construct a training sample for each pair of elements $< e_1, e_2 >$ where e_1 and e_2 come respectively from the two schemas of the existing matching; let the input of the training sample be the feature values of $< e_1, e_2 >$, and the target output be the squared error of their similarity outputted by the base matcher.

However, since in actual schema matching the matched element pairs are far less than unmatched ones, the training set constructed in this way is highly unbalanced, leading to a predictor with an overwhelming bias. To overcome this problem, before training we duplicate those samples that are constructed by matched element pairs, so as to make the numbers of the two kinds equivalent.

Notice that although machine learning is used here, for a particular matcher once the predictor is trained, it could be applied for any matching task and no retraining is mandatory. Moreover, while all kinds of supervised learning methods can be used here, the online learning techniques [15] are preferred as the learner

could improve itself in operation, thus further eliminating the worry of having insufficient existing schema matchings for training.

4.2 From Accuracy to Credibility

With accuracy (i.e. MSE) estimated, the credibility of each outputted similarity can be calculated as follows:

$$cred = e^{-C \times MSE}$$

Here C is a non-negative constant, determining how fast the credibility falls with the increase of MSE. When C is positive, higher credibility is assigned to matchers with higher accuracy (i.e. with lower MSE); but if C is zero, the system simply averages the results from matchers, as in COMA. The empirical value of C is 1.0.

5 Evaluation

We evaluated CMC on several real world schemas, i.e. five XML schemas for purchase orders, which were first used in [8]. These schemas have 55 elements on average, and their XML-tree depths are at least 4. The matches between them have been manually established, which serve as either the benchmark or the training set.

Four measures are adopted to evaluate matching results. Among them, *precision* and *recall* are highly coupled measures that are widely used. *Overall* has been used in several schema matching literatures like [3, 8], taking into account the manual effort needed to complete the matching task after automatic matching. *F-measure* is the harmonic mean of precision and recall and has been used in [5].

The CMC system used in the experiments consists of four base matchers, i.e. NameMatcher, DataTypeMatcher, PathNameMatcher and LeavesMatcher. PathNameMatcher compares the names of the paths from the XML root to the elements being matched. The other three matchers are introduced in Sect.3. UserFeedBack is not employed so as to exclude subjective factors. The default structure of CMC is used, i.e. the former three matchers are in the bottom layer and the last one is in the upper layer. The machine learning technique used in credibility prediction is the multilayer perceptron [15].

For comparison, another two combination methods are also tested. The first one, the average-combination method, simply averages the results from base matchers. This is also the default combination method of COMA. The second uses the meta learning method of LSD, i.e. stacking [11], for combination. As discussed in Sect.2, the meta learner in LSD is not applicable for generic schema matching, so we make a slight modification on it, i.e. training one meta-learner for all elements, instead of training multiple ones for different elements in the mediate schema. Notice that when testing these two methods, only the combiners

in the system are substituted while the base matchers and their connection remain unchanged. In addition, all the three methods employ the same converter, which is discussed in Sect.3, to convert the similarity matrix to matches. For each method the converter's parameters are tuned to achieve the best performance, and it happens that the three sets of parameters are the same, i.e. $threshold = 0.5$ and $delta = 0.02$, as in the COMA system.

It is worth mentioning that our experiments were designed to compare different combination strategies, instead of measuring the absolute matching performance. Therefore some powerful but very complicated techniques were not used, and the system configuration was not specially optimized for the matching tasks in hand.

5.1 Single Matcher vs. Combination

We first compared the performance of base matchers against the combination methods. The test was made on the first four of the ten matches between the five testing schemas, with the rest of the matches constituting the training set for both CMC and the meta-learning method.

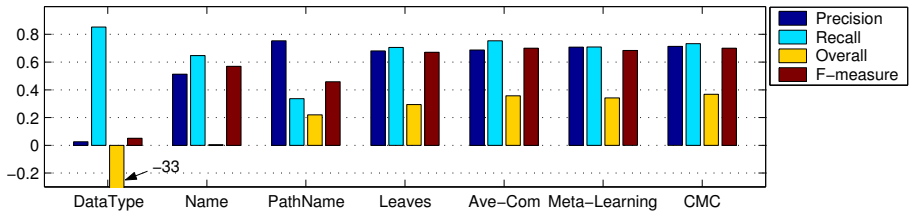


Fig. 4. The performance of single base matchers and combination methods.

The advantage of combination methods can be easily seen in Fig.4. The integrated measures (i.e. Overall and F-measure) are all improved significantly by combination. LeavesMatcher performs significantly better than the other three base matchers, but this is also the contribution of combination: LeavesMatcher takes as the initial similarity the combined results of the other three matchers.

5.2 Comparing Combination Methods

To compare the performance of the three combination methods, we made a comprehensive test on all the ten matches between the five testing schemas. Considering that machine learning is used in CMC and the meta-learning method, we adopted a cross-validation strategy [15]. The ten matches were divided into five groups, and each time two successive groups were used for testing and the rest were used for training. Thus the testing was conducted for five times altogether. Comparisons between CMC and the other two methods are respectively

illustrated in Fig.5 and Fig.6, where the data is computed by subtracting the results of the contrast method from the results of CMC.

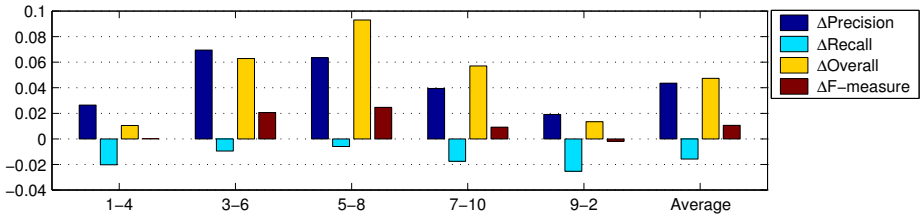


Fig. 5. Average-Combination vs. CMC

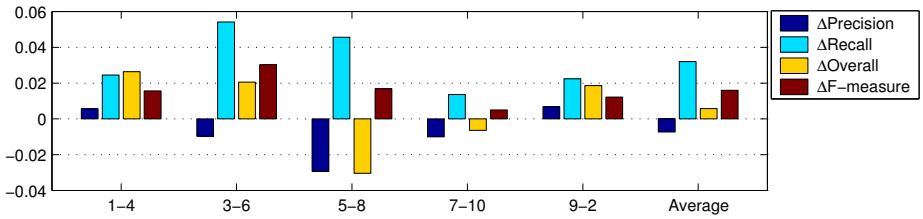


Fig. 6. Meta-Learning vs. CMC

On average the two integrated measures (i.e. Overall and F-measure) are increased by CMC in both comparisons. For individual tests, the Overall of CMC is only reduced in two out of the ten comparison tests, while is increased in all the others with the highest increase near 0.1, which means nearly 10% more manual effort is saved; the F-measure of CMC is increased in nine out of the ten comparison tests and only decreases very slightly in the fifth test compared to the average method.

It is interesting to see that, CMC outperforms the average method by higher precision while outperforms the meta-learning method mainly by higher recall. We suppose this somewhat exposes the characteristic of the three methods, and obviously CMC has a more balanced performance on precision and recall.

From the two figures one can also find that the meta-learning combination method is better than the average method in the overall performance. This is not surprising as meta-learning is much more complicated than the average method.

6 Conclusion and Future Work

Matcher combination is a promising technique for schema matching. Against the limitations of existing combination systems, in this paper we propose the CMC system, which combines multiple matchers based on credibility prediction. For an individual matcher, its accuracy on each input is dynamically predicted,

by either manual rule or automatic learner. Based on the predicted accuracy, its credibility is calculated. Then a combiner aggregates results from its base matchers by weighted average, with each matcher's weight specified by its current credibility. Our experiments demonstrate the advantage of this approach.

The following is a list reviewing how the limitations of previous systems discussed in Sect.1 and Sect.2 are overcome by CMC.

- When machine learning may be used in accuracy prediction, arbitrary existing matches can be used for training, and online learning is also applicable, so collecting training set will not be a problem. Once trained, the predictor can work for arbitrary matching task and no retraining is obligatory, so the time for training is neglectable. Actually, as training can be accomplished by developers, the end users may even be unaware of it.
- The credibility prediction for each base matcher is independent, so adding or removing matchers won't affect the combination.
- Our combination method could take into account any available information of the current matching, which is specified as input features of credibility prediction.
- The combination procedure is fully automatic unless the users do not want to use the default connection policy of base matchers.

In future work, we plan to add more powerful base matchers to CMC for better performance. Furthermore, it would be interesting to study other applications of credibility prediction, such as for base-matcher selection.

References

1. Palopoli, L., Saccà, D., Terracina, G., Ursino, D.: Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. IEEE Transactions on Knowledge and Data Engineering **15** (2003) 271–294
2. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In Apers, P.M.G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., Snodgrass, R.T., eds.: Proceedings of the Twenty-seventh International Conference on Very Large Data Bases: Roma, Italy, 11–14th September, 2001, Los Altos, CA 94022, USA, Morgan Kaufmann Publishers (2001) 49–58
3. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and ist Application to Schema Matching. In: Proc. 18th ICDE, San Jose, CA (2002)
4. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: a machine-learning approach. SIGMOD Record (ACM Special Interest Group on Management of Data) **30** (2001) 509–520
5. Berlin, J., Motro, A.: Database schema matching using machine learning with feature selection. In: Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE). (2002)
6. Kang, J., Naughton, J.F.: On schema matching with opaque column names and data values. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data 2003, San Diego, California, June 09–12, 2003, New York, NY 10036, USA, ACM Press (2003) 205–216

7. Guo, M., Yu, Y.: Mutual enhancement of schema mapping and data mapping. In: ACM SIGKDD 2004 Workshop on Mining For and From the Semantic Web. (2004)
8. Do, H.H., Rahm, E.: COMA — A system for flexible combination of schema matching approaches. In: VLDB 2002: proceedings of the Twenty-Eighth International Conference on Very Large Data Bases, Morgan Kaufmann Publishers (2002) 610–621
9. Madhavan, J., Bernstein, P., Chen, K., Halevy, A., Shenoy, P.P.: Corpus-based schema matching. In: Information Integration on the Web Workshop at IJCAI03. (2003)
10. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In Freytag, J.C., Lockemann, P.C., Abiteboul, S., Carey, M.J., Selinger, P.G., Heuer, A., eds.: VLDB 2003: Proceedings of 29th International Conference on Very Large Data Bases, September 9–12, 2003, Berlin, Germany, Los Altos, CA 94022, USA, Morgan Kaufmann Publishers (2003) 572–583
11. Wolpert, D.H.: Stacked generalization. Neural Networks **5** (1992) 241–259
12. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to Map between Ontologies on the Semantic Web. In: Proceedings of the World-Wide Web Conference (WWW-2002), ACM Press (2002)
13. Yatskevich, M.: Preliminary evaluation of schema matching systems. Technical Report # DIT-03-028, Department of Information and Communication Technology, University Of Trento (Italy) (2003)
14. Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble method for neural networks. In Mammone, R.J., ed.: Neural Networks for Speech and Image processing, Chapman-Hall (1993)
15. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)