MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Matcher Composition Methods for Automatic Schema Matching

Nikovski, D.; Esenther, A.; Ye, X.; Shiba, M.; Takayama, S.

TR2013-103 December 2013

Abstract

We address the problem of automating the process of deciding whether two data schema elements match (that is, refer to the same actual object or concept), and propose several methods for combining evidence computed by multiple basic matchers. One class of methods uses Bayesian networks to account for the conditional dependency between the similarity values produced by individual matchers that use the same or similar information, so as to avoid overconfidence in match probability estimates and improve the accuracy of matching. Another class of methods relies on optimization switches that mitigate this dependency in a domain-independent manner. Experimental results under several testing protocols suggest that the matching accuracy of the Bayesian composite matchers can significantly exceed that of the individual component matchers, and the careful selection of optimization switches can improve matching accuracy even further.

Springer Link

Matcher Composition Methods for Automatic Schema Matching

Daniel Nikovski[1], Alan Esenther[1], Xiang Ye[1], Mitsuteru Shiba[2], and Shigenobu Takayama[3]

[1] Mitsubishi Electric Research Laboratories,
201 Broadway, Cambridge, MA 02139, USA
[2] Mitsubishi Electric Corporation,
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan
[3] Mitsubishi Electric Information Systems Corporation,
325 Kamimachiya, Kamakura, Kanagawa 247-0065, Japan

Abstract. We address the problem of automating the process of deciding whether two data schema elements match (that is, refer to the same actual object or concept), and propose several methods for combining evidence computed by multiple basic matchers. One class of methods uses Bayesian networks to account for the conditional dependency between the similarity values produced by individual matchers that use the same or similar information, so as to avoid overconfidence in match probability estimates and improve the accuracy of matching. Another class of methods relies on optimization switches that mitigate this dependency in a domain-independent manner. Experimental results under several testing protocols suggest that the matching accuracy of the Bayesian composite matchers can significantly exceed that of the individual component matchers, and the careful selection of optimization switches can improve matching accuracy even further.

Key words: Data integration; virtual databases; uncertain schema matching

1 Introduction

It is often necessary to establish a correspondence (matching) between the schemas of two or more databases, that is, determine which schema elements refer to the same concept or physical object in the problem domain. This need arises in multiple tasks in the area of data management, such as data integration, data migration, and also the creation of virtual databases that expose the same unfiying data model while retrieving data from multiple physical databases. As a result, there has been significant research into automated and semi-automated methods for schema matching [1].

It is widely acknoledged that the automatic schema matching (ASM) problem is very difficult, because when database designers create database schemas, they rarely provide full and unambiguous information about what individual schema elements represent, and if any such information exists, it is usually not

meant for computer processing. Rather, database designers usually choose suitable words or abbreviations for the names of data elements, so as to facilitate future maintenance of the data schemas by themselves or other humans. Lexical analysis of the names of data elements, then, is an important approach to ASM. For example, the names "Street", "Str", and "StreetName" can be recognized to refer to a street, possibly in an address, and lexical analysis by string matching can reveal this similarity. A different type of information that might be useful for ASM is the structure of the data schemas, if present. In many cases, schemas are not represented by a flat list of element names, but the elements are organized in a hierarchy. For example, the element "CustomerName" might have three subelements, "FirstName", "MiddleInitial", and "FamilyName". Using such structural information is another approach to ASM. Many more approaches exist, too. For example, when the actual values of two database fields come from the same statistical distribution (e.g., over names, numbers, etc.), this can serve as evidence that the corresponding schema elements match. Dictionaries, thesauri, and other auxiliary data sources have been used for ASM purposes, too [1].

Due to the difficulty of the problem, no single method has been shown to perform best on all ASM tasks. This has led to the idea that multiple basic matchers of the types described above can be used together in a composite matcher [2, 10]. The purpose of the composite matcher is to combine the output of the individual matchers and arrive at a more accurate set of likely matches. In most cases, the output of an individual matcher k for a given pair of elements $S_1.E_i$ and $S_2.E_j$ is a similarity value v_k in the interval $[0, 1]$, where $v_k = 0$ means no similarity, and $v_k = 1$ means full confidence that the two elements match. When given a library of K different individual matchers, the objective, then, is to find a composite similarity measure v that is a function of the individual outputs v_k, $k = 1, K$.

Several methods for combining similarity values have been proposed. The LSD system [4] uses machine learning techniques to estimate weighting coefficients w_k such that the final similarity measure v is a weighted average of the individual similarity measures: $v = \sum_{k=1}^{K} w_k v_k$. The COMA system [2] extends this approach with the minimum and maximum operators: $v_{min} = \min_k w_k v_k$ and $v_{max} = \max_k w_k v_k$.

Although experimental results suggest that these methods for combining similarity values lead to matching accuracy that is higher than that of the accuracy of the individual matchers, it can be recognized that they are specific approaches to the fundamental problem of combining evidence from multiple sources (in this case, multiple individual matchers), and make very specific assumptions about the statistical structure of the evidence that might or might not be warranted in practice. In Section 2, we propose a general method for correct modeling of any kind of statistical structure in the evidence, based on Bayesian networks and probabilistic reasoning, and a statistically grounded method for composing matcher evidence using these Bayesian networks, and in Section 3, we describe the performance of the composite matcher on benchmark problems.

We also investigated the limits on the accuracy obtainable by means of matcher composition by analyzing the type of mistakes made by the basic matchers, and tailoring the combination methods to the kind of elements that were being matched. For example, we discovered that very different matchers were useful for matching leaf nodes and internal nodes in the schemas. Based on this analysis, we devised new composite matchers that were able to increase the matching accuracy even further, compared to the Bayesian approach, although it remains to be seen how these matchers would peform in new domains. These additional matchers are described in Section 4.

2 Bayesian Networks for Combining Outputs of Multiple Schema Matchers

When combining evidence from multiple sources, one of the major problems and causes for errors is the improper modeling of correlation and other forms of statistical dependence between variables in the problem domain. For example, when two very similar matchers k and l are applied to an ASM problem, their outputs v_k and v_l will be highly correlated — when v_k is high, then v_l will be high, too, and vice versa. For example, a lexical matcher based on edit (Levenshtein) distance would assign a medium-level similarity to the pair of element names "Street" and "State"; similarly, a lexical matcher based on the Jaccard distance between the sets of letters in the two elements would assign such similarity to the pair. For another pair of elements, for example "Street" and "Address1", both lexical matchers would compute low similarity. In either case, not only is the computed similarity misleading as regards to the correct match, but both matchers provide the same kind of evidence (both positive or both negative), so its (in this case, harmful) influence is reinforced. If a weighted sum of the two similarity values is used, the same evidence will be counted twice, in practice, which will result in a phenomenon known as over-confidence. One of the matchers is almost redundant, and including it in the composition process might actually decrease the accuracy of matching. This effect has been observed in other fields where evidence has to be combined, such as medical diagnosis, and one possible tool for handling it has been belief reasoning in Bayesian networks. Our method for combining matcher output is based on such a network.

2.1 Representation

A Bayesian network (BN) is a probabilistic graphical model that represents a set of random variables and their conditional dependencies by means of a directed acyclic graph (DAG). An edge in the DAG between two nodes signifies that the variable Y corresponding to the child node is statistically conditionally dependent on the variable X corresponding to the parent node. This dependence is expressed in a conditional probability table (CPT) stored in the child node for Y. If $X \in Par(Y)$, where $Par(Y)$ is the set of parent nodes of Y, this

table contains probability entries $Pr(Y = y|Par(Y) = z)$ for every possible combination of values x that X can take on and configurations (sets of values) z that the variables in $Par(X)$ can take on. Likewise, when there is no direct edge between two nodes, they are assumed to be conditionally independent given their parents. In particular, when two nodes have a common parent, but no edge between them, they are assumed to be conditionally independent given the value of their parent. The presence (or absence) of edges in the DAG of a Bayesian network is a way to express the statistical dependence (correlation) between variables.

A Bayesian network to be used for combining outputs of individual matchers in an ASM task is shown in Figure 1. Its DAG is a tree of depth four, with some additional edges between some of the nodes. The meaning of the nodes is as follows:

1. At the first (top) level, the root node corresponds to a Boolean variable signifying whether two schema elements match. This is the final hypothesis that has to be evaluated.
2. The nodes at the second level of the trees represent independent ways in which the two element names can match (lexical, structural, instance-based, etc.). It is expected that these variables are largely uncorrelated, because they use different information to test for possible matches. They also each correspond to clusters of individual matchers whose output is correlated. In Figure 1, one cluster represents the hypothesis that the two elements match lexically, and the other cluster represents the hypothesis that the instances (values) of the two elements in their respective databases match.
3. The nodes at the third level of the tree are also Boolean and represent the individual hypothesis that the two elements match, according to a single matcher. In Figure 1, these include two lexical matchers LM1 and LM2, one structural matcher, one synonym matcher, and two instance matchers IM1 and IM2.
4. The leaves of the tree, at the fourth level, represent the similarity values V_k, $k = 1, K$ of the individual matchers whose outputs have to be combined (in this case, for illustration, $K = 6$). These variables are continuous, and their possible values are the real numbers v_k.

The overall structure of the BN expresses the understanding that when two elements match (or don't), the outputs of the structural matcher, synonym matcher, the lexical match variable, and the instance match variable will be statistically independent. This is what is to be expected on a matching task, because these matchers all use different information from the two data schemas in order to make an estimate about whether the elements match. However, the outputs of the two lexical matchers LM1 and LM2 would be correlated, as expected if they use the same information (the names of the two elements). That is why there exists an edge between nodes LM1 and LM2. Similarly, the output of the two instance matchers would be correlated, too, because they would both use the same information to base their estimates on (namely, the contents of the two corresponding database fields). Accordingly, an edge between nodes IM1

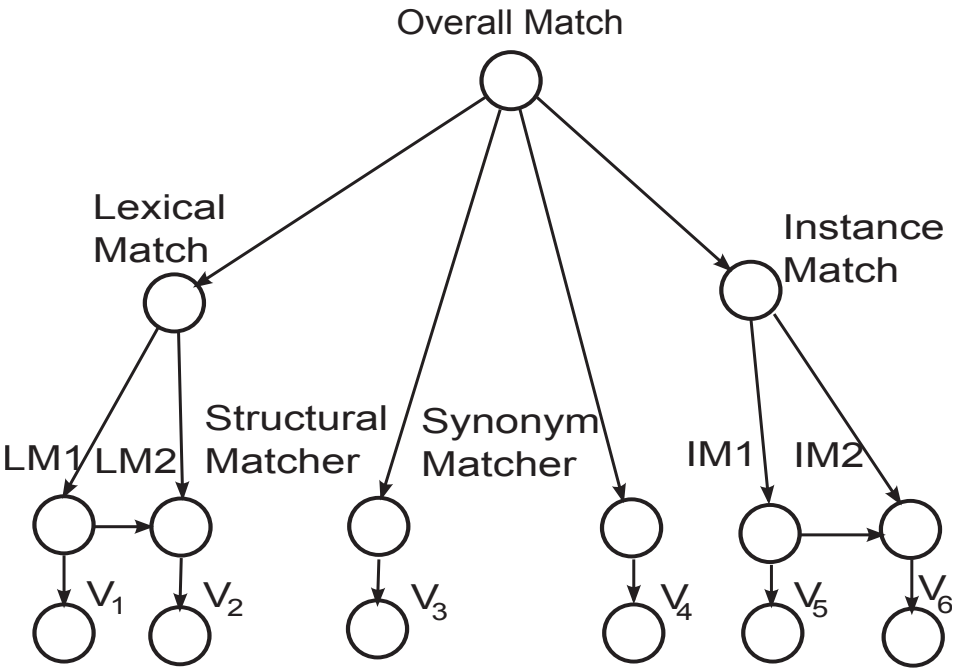

Fig. 1. A Bayesian network for combining the output of multiple individual matchers.

and IM2 reflects this dependency. This structure of the BN, then, corresponds to our understanding of which matchers produce highly correlated outputs, and which ones are statistically independent.

2.2 Parameter Estimation

In addition to the graph of the BN, if the network is to be used for inference, the parameters in its CPTs have to be specified, too. This can be done by means of labeled cases, where pairs $e_l = (S_1.E_i, S_2E_j)$ of elements $S_1.E_i$ and $S_2.E_j$, $l = 1, \ldots, N$ have been run through all K matchers, to produce the corresponding similarity values $v_{l,k}$, $l = 1, \ldots, N$, $k = 1, \ldots, K$, and the correct labeling for some or all of the remaining Boolean variables has been supplied, too.

If labels for all Boolean variables have been supplied, then the estimation of the probabilities in the CPTs of the Boolean nodes could be reduced to frequency counting. That is, the entry $Pr(Y = y|Par(Y) = z)$ is equal to the ratio of the number of cases when Y had a specific value y (either True or False) and the parents of Y were in configuration z, and the number of times the parents of Y were in configuration z (regardless of the value of Y). For the continuous nodes V_k, a suitable parametric model for the similarity values must be chosen. One possible model is a normal (Gaussian) distribution with mean $\mu$and variance σ^2. Then, two separate normal distributions $N(\mu_{k,+}, \sigma^2_{k,+})$ and $N(\mu_{k,-}, \sigma^2_{k,-})$ are estimated for positive (matching) and negative (non-matching) cases (pairs of

elements), respectively. The mean $\mu_{k,+}$ is the average of the similarity values $v_{k,i}$ of all data cases where the parent node X_k of V_k has been labeled with value True. The parameter $\sigma_{k,+}$ is the sampled standard deviation of these cases. Analogously, the parameters $\mu_{k,-}$ and $\sigma_{k,-}$ are the sample mean and standard deviation of $v_{k,i}$ over all cases when the parent node X_k has been labeled with the value False.

It is also possible to estimate the parameters in the CPTs when only some of the nodes have been labeled. A typical situation arises when a human designer has provided feedback about whether the two elements match (that is, has assigned a Boolean value to the root node of the BN), but has not explained why they match (that is, whether the match is lexical, instance-based, structural, based on a dictionary, etc.) This situation is more challenging, but as long as the graph of the network is known and fixed, it is still possible to estimate the most likely values of the parameters in its CPT. This problem is known as parameter learning with partially observed data in Bayesian networks, and can be solved by means of gradient ascent in the likelihood function or the Expectation Maximization algorithm, among other methods [9, 11].

Assuming there is a data set Σ of N independent training cases, the log-likelihood scoring function is

$$log\ L(\Theta|\Sigma) = \frac{1}{N}\sum_{i=1}^{M}\sum_{l=1}^{N} log\ P(X_{il}|Pa(X_i),\ \theta_i),$$

where Σ denotes the training data set, $Pa(X_i)$ denotes the parents of the node X_i, $i = 1, \ldots, M$, and Θ is the parameter vector $\Theta = \{\theta_1, \ldots, \theta_M\}$.

However, we only have partial observations, which means that there are several hidden nodes with no labels. For each training case, one pair of elements $S_1.E_i$ and $S_2.E_j$ is run through all K individual matchers to produce the corresponding similarity values $v_{i,j,k}$, and a true label of two elements matching or not for the root node $OverallMatch$ is provided by the human designer. With known structure and partial observation, we can use the EM (expectation maximization) algorithm to find a locally optimal maximum-likelihood estimate of the parameters. After learning parameters from training data set, each discrete node has a conditional probability table (CPT) specifying the probability of each state of the node given each possible combination of parents' states.

2.3 Inference

Given the individual similarity values $V_k = v_k$, $k = 1, K$ that have been reported by all individual matchers, and a full Bayesian network with CPTs estimated from data, we can evaluate the probability that the two elements match on the basis of all evidence, by means of a standard computational process known as belief updating. One possible method to perform belief updating is to construct the join tree of the Bayesian network, and use if for inference. This can be done by means of a number of commercial and freely available reasoning engines.

The continuous variables V_k, under the chosen Gaussian parametrization, can be incorporated into the process of belief updating in the form of virtual (uncertain) evidence [12]. To supply virtual evidence to a belief updating engine, all that is needed is the likelihood ratio of the observed values v_k for the similarity value variables V_k:

$$L(V_k = v_k|X_k) \doteq \frac{Pr(V_k = v_k|X_k = T)}{Pr(V_k = v_k|X_k = F)} = \frac{N(v_k|\mu_{k,+}, \sigma^2_{k,+})}{N(v_k|\mu_{k,-}, \sigma^2_{k,-})},$$

where $N(v|\mu, \sigma^2)$ is the probability that measurement v comes from normal distribution with mean μ and variance σ^2, and X_k is the parent node of V_k in the BN.

After the process of belief updating concludes, all Boolean nodes in the network will be assigned probability values according to the observed evidence (values) v_k for the similarity value variables V_k. The probability of the root node is the final estimate that the two elements match, given the combined evidence of the individual matchers.

3 Experimental Results

In order to evaluate the match accuracy of any matcher described below, we used five XML schemas for purchase orders, CIDX, Excel, Noris, Paragon and Apertum, kindly provided to us by the University of Leipzig. The figure of merit for evaluation of the accuracy of matching was the popular f-measure, defined as the harmonic mean of precision and recall, as used in the information retrieval community. If the number of true matches identified by the matching system as such (hits) is A, the number of true matches not identified as such (misses) is B, and the number of cases when two elements do not match, but the matcher incorrectly declares a match (false positives) is C, the f-measure F can be computed as $F = 2A/(2A + B + C)$.

We developed 13 basic schema matchers and evaluated the ability of the proposed Bayesian method to combine their outputs so as to improve the accuracy of matching. Of these, 11 were lexical matchers: CosineSimilarity, HammingDistance, JaroMeasure, LevenshteinString, BigramDistance, TrigramDistance, QuadgramDistance, PrefixName, SuffixName, AffixName, SubstringDistance. One matcher, PathName, was structural, comparing the entire paths of the two elements in their respective XML schemas. The last basic matcher was neither lexical nor structural: the Synonym matcher declared a match if and only if the two tested elements were found in a list of synonyms relevant to the domain of purchase orders. Based on their method of operation, the similarity values computed by the 11 lexical matchers can be expected to be highly correlated and statistically dependent; in contrast, the synonym matcher could be expected to produce output that is largely independent of the lexical matchers. Experimental evaluation of their pairwise dependence confirms this intuition:

Figure 2 shows the pairwise correlation between all 13 pairs of matchers, evaluated from all pairs of elements in all ten pairs of schemas. Clearly, all 11 lexical matchers are highly correlated, whereas their correlation with the Synonym matcher is minimal. Somewhat surprisingly, the structural matcher, PathName, is the least correlated with any other matcher.

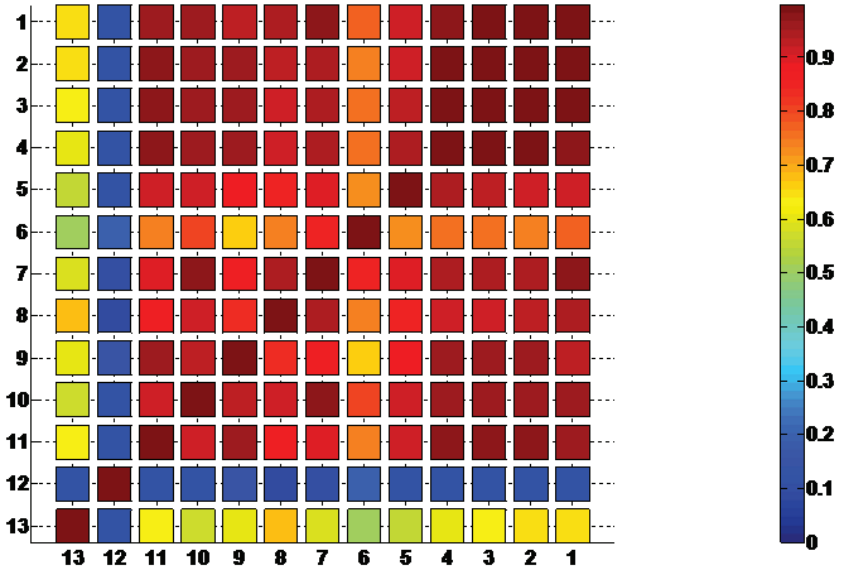

Fig. 2. Pair-wise correlations between all pairs of basic matchers, numbered as follows: 1: Edit Distance; 2: Sub-string Distance; 3: Bi-Gram Distance; 4: Tri-Gram Distance; 5: Quad-Gram Distance; 6: Cosine Similarity; 7: Hamming Distance; 8: Jaro Measure; 9: Affix Name; 10: Prefix Name; 11: Suffix Name; 12: Path Name; 13: Synonym.

The kind of major correlation that exists between lexical matchers is illustrated in Figure 3 that shows a scatter plot of the similarity values computed by the Edit (Levenshtein) Distance matcher and the Sub-string Distance matcher. Their high correlation (0.9892) makes one of them almost redundant, if the other one is present.

Regarding the experimental evaluation of matching accuracy, as with any machine learning method, care should be given to the training and testing evaluation protocol, that is, which data are used for training and which data are used for testing. We used three evaluation protocols, as described below.

3.1 Testing on Training Data set

This is the simplest evaluation protocol, where we use the same data set for testing and training. Its purpose is to evaluate how well we can fit the training data. Under this protocol, we define ten matching tasks that correspond to all

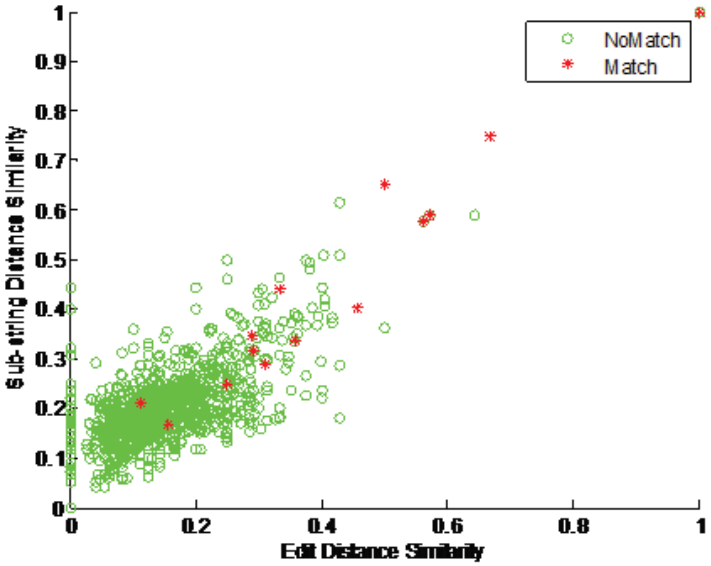

Fig. 3. Scatter plot of similarity values computed by the Edit Distance and Substring Distance matchers. Their output is clearly correlated, resulting in a correlation coefficient of 0.9892.

possible pairs of the five XML schemas. For each matching task (pair of schemas), we build a dedicated Bayesian composite matcher that is specific for this task. The same data set, then, is used as evidence to predict the belief for every pair of elements. This is the most lenient evaluation protocol, since the learning algorithm has seen during training the data that will be used for testing.

After a similarity matrix is computed for all pairs of elements of two database schemas, an additional global matching step called Max1/Delta is performed to produce the final match decisions, based on the understanding that most often (but not always) mappings between database elements are one-to-one [2]. Since this procedure is sensitive to the exact value of the Delta parameter, we present below results as a function of it. After global match decisions have been obtained, they are compared with the ground truth, and the f-measure for this pair of schemas is computed. These f-measures are averaged over all pairs of tasks in the testing data set (in this case, ten pairs of tasks), in order to arrive at the final overall f-measure.

Figure 4 shows a comparison between all 13 basic matchers and the Bayesian Composite Matcher (BCM). The accuracy of the BCM reaches 0.819 and is significantly higher than that of any other matcher. It is also practically constant for a wide range of the parameter Delta. The performance of Path Name matcher is better than other individual matchers, because it is a hybrid matcher combining two basic match techniques.

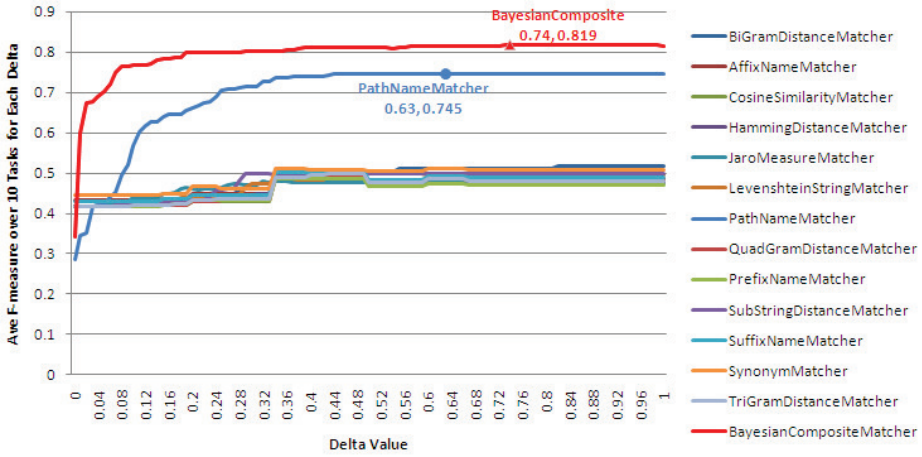

Fig. 4. Comparison of average f-measure between the Bayesian Composite Matcher and all other matchers.

3.2 Leave-One-Out Cross Validation (LOOCV)

A more realistic testing protocol is under the leave-one-out cross validation (LOOCV) method, where training and testing data are clearly separated. Each of the ten pairs of schemas is used for testing, using a BCM that was learned using the other nine pairs of schemas. The results are averaged over the ten pairs, as follows:

1. Build training and testing data sets for 10 test tasks. For instance, if the similarity matrix of $Excel \leftrightarrow Noris$ is used as testing set, the training data set for this test task is a collection of similarity matrices of the remaining 9 schema pairs.
2. Learn one Bayesian composite matcher for each task based on its training data.
3. Implement $Max1/Delta$ selection approach on the composite similarity matrix generated by each Bayesian Composite Matcher.

3.3 Exclusive Leave-One-Out Cross Validation (ExclLOOCV)

The second protocol described above still allowed the training algorithm to see data from the pair of schemas that would be used for testing, but not the ground truth for their direct match. To eliminate any exposure of the training algorithm to data that would be used for testing, we modified the LOOCV procedure as follows. For each task, if the test pair is $A \leftrightarrow B$, the training examples only come from the three remaining schemas not involving either A nor B. For example, if one test set is $Excel \leftrightarrow Noris$, it will be tested with the Bayesian composite matcher that has used only the following three pairs of schemas for training:

$CIDX \leftrightarrow Apertum$, $CIDX \leftrightarrow Paragon$, and $Apertum \leftrightarrow Paragon$. This is the maximally realistic testing protocol.

Figure 5 shows a comparison between the two variants of the LOOCV evaluation protocol for the Bayesian Composite Matcher. It can be seen that the accuracy drops to 0.76 under usual LOOCV and 0.73 under exclusive LOOCV.

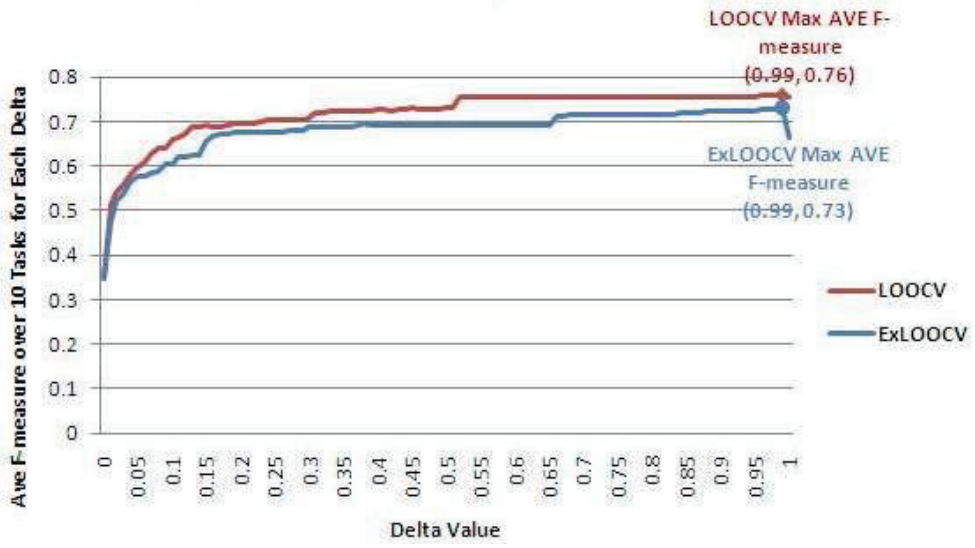

Fig. 5. Comparison of Bayesian composite matcher performance under LOOCV and exclusive LOOCV testing protocols.

4 Non-Bayesian Matcher Composition Optimizations

Based on familiarity with the domain acquired during the Bayesian Networks approach, we further explored non-Bayesian matcher composition approaches. A series of optimizations proved promising and are described here.

Typically, datasets have element names with strong lexical components, so it was critical to develop a sound fundamental lexical (name) matcher that could be heavily used as a component in higher-level composite matchers. The resulting Name matcher was a combination of fundamental matchers described in Section 3, using token processing and largely COMA-style techniques [2] on the element name tokens. Incorporation of abbreviation and synonym information was an integral part of this process, too. As an example, averaged across all 10 schema pairs in the COMA dataset, we only achieve an f-measure of 0.433 using a simple prefix name matcher, which looks for identical prefixes in column names. A synonym matcher, utilizing a lookup table is also an important component, but only achieved an f-measure of 0.461 by itself. Through experimental analysis, we found that combining lexical matchers into a Name Composite

matcher achieved a marginal increase to 0.494, but that the resulting matcher was a critical component of higher-level matchers. A PathName matcher, which compares the full paths to each element in their XML schemas, based on this Name matcher boosted the f-measure to 0.820. Note that these matchers incorporated additional optimizations that were applied in parallel, some of which are described below.

The next step was to develop a Structure matcher which uses different strategies based on the type of nodes (root, interior or leaf node) being compared [19]. The matchers that we found to produce the best overall f-measures are shown in the table.

Table 1. Best matchers for various combinations of node types.

XML Schema node type	Matcher applied
LEAF-LEAF	PathName
ROOT-ROOT	LeafPath
INTERIOR-INTERIOR	ChildPath, SiblingPath
INTERIOR-ROOT	ChildPath
INTERIOR-LEAF	LeafPath
ROOT-LEAF	LeafPath

Here LeafPath is an application of the Name matcher to the paths to all leaves of the element nodes being compared. Similarly, ChildPath is an application of the Name matcher to the paths to all of the children of the nodes begin compared, and SiblingPath is an application of the Name matcher to all of the paths to the siblings of the nodes being compared. For the ten schema pairs in the COMA dataset, the Structure matcher achieved an average f-measure of 0.871. In addition to combining matchers, it was often found to be useful to "pre-filter" candidate element pairs by simply eliminating those for which the Name similarity was less than 50% This REQUIRE_ELEMENT_NAMES_TO_BE_SIMILAR optimization was originally created as an optimization to improve compute time, but actually had a noticeable positive effect on overall results.

We next developed a LinearComposite matcher which allows one to manually specify weights for component matchers. While this technique itself is by no means automatic, there were some interesting findings in this work. The matcher was initially developed as a mechanism to strengthen effects of the Name and SiblingPath matchers, which the StructureMatcher seemed to diminish in some cases. The LinearComposite matcher was enhanced to consider the types of the XML Schema nodes being compared, as well. For example, for INTERIOR-INTERIOR node comparisons, it just uses StructureMatcher, but for LEAF-LEAF comparisons it uses additional information such as SiblingName and data type. The improved results suggest, perhaps somewhat intuitively, that interior nodes are more sensitive to where they appear in the XML hierarchy, but that

leaf nodes are more sensitive to their siblings. The LinearComposite matcher, including various optimizations, achieved an average f-measure of 0.901 across all 10 schema pairs.

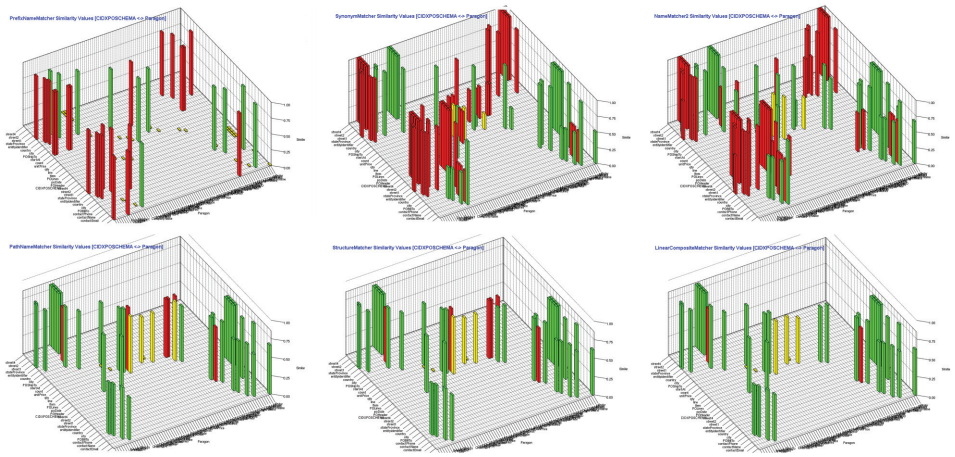

Fig. 6. CIDXPOSCHEMA and Paragon element similarity scatterplots. The matchers and f-measures achieved were (top row, left-to-right) PrefixName 0.268, Synonym 0.548, Name 0.557, (bottom row, left-to-right) PathName 0.854, Structure 0.866, LinearComposite 0.903. Green=true positive, Red=false positive, Yellow=false negative.

Two other optimizations involved re-evaluating neighbor similarities. In the first, MULTIMATCH_OPTIMIZATION, if a schema element had multiple matches, we would re-consider all of its similarities with a slightly less stringent threshold. The other, SIBLING_MULTIMATCH_OPTIMIZATION, was that if a leaf element pair had multiple siblings that were matches, then we would re-compare all siblings of each, primarily with a less stringent threshold. This could catch cases where the leaf nodes were representing the parts of an address, for example. In this case one would expect multiple siblings to match their counterparts in the set of address leaf nodes in the other schema. Finally, there are simple heuristics which can be applied such as eliminating unlikely node combinations with the ELIMINATE_BAD_DATATYPE_MATCHES optimization. For example, ROOT nodes never matched LEAF nodes.

In general, our approach when investigating such optimizations and matcher combination alternatives was to expose the enhancements as parameters so that a human operator could experiment with them via a GUI. As can be seen in the table, individual optimizations can have different effects on matchers, though we highlight those that help most with the higher level combination matchers (Structure and LinearComposite).

Table 2. Effect of turning off some of the individual optimizations on average f-measures for various matchers.

Optimization Turned OFF:	Matcher:					
	PrefixName	Synonym	Name	PathName	Structure	LinComp
NONE	0.433	0.461	0.494	0.820	0.871	0.901
REQUIRE_ELEMENT_NAMES_TO_BE_SIMILAR	0.433	0.461	0.496	0.748	0.715	0.792
MULTIMATCH_OPTIMIZATION	0.449	0.544	0.583	0.831	0.861	0.874
SIBLING_MULTIMATCH_OPTIMIZATION	0.433	0.459	0.492	0.821	0.869	0.898
ELIMINATE_BAD_DATATYPE_MATCHES	0.405	0.456	0.490	0.786	0.868	0.897

5 Related Work

As mentioned in the first section, many methods for creating composite matchers have been tried, and this section explains the difference between them and the proposed approach. One major distinction between these methods is whether they rely on manual tuning of the composition sttructure and parameters, or such parameters are estimated from a training set and verified on an independent test set. The composition methods developed in the COMA [2, 6] and GLUE [14] systems are based on manual tuning of the composition parameters, so comparison with learning methods for tuning parameters is not entirely correct; a composite matcher that is manually tuned with a specific set of schemas in mind can certainly be expected to be more accurate than a learning matcher that is tested under a cross-validation protocol.

Among the learning methods for composing matchers, our approach is most similar to the one proposed by Marie and Gal [13], who have approached the problem from a Bayesian network perspective, too, arguing that a disciplined approach to handling match uncertainty has to be applied. However, their approach is based on Naive Bayes networks, that is, two-level Bayesian networks with one root node that corresponds to the matching event, and many leaf nodes that are directly children to the root node. It can be shown that such a Naive Bayes network has the same classification properties as a logistic regression model, and the decision surface is linear, similar to the one used in the LSD and GLUE systems [4, 14]. In contrast, a full (non-naive) Bayesian network like the one proposed in this paper can model arbitrary correlations and decision surfaces.

Furthermore, the Bayesian network proposed in this paper is also different from the Bayesian network classifiers used in the YAM system [16] in that our network includes unobservable nodes corresponding to types of matchers; in contrast, YAM employs the BayesNet classifier from the WEKA library that can learn the structure of a fully observable network by adding and removing edges, but cannot add unobservable nodes [17]. Unobservable nodes corresponding to a type of matcher (e.g. lexical, dictionary-based, structural, etc.) present a natural way of representing the conditional dependency between multiple matchers of the same type, because they restrict the edges of the graph only to the nodes of the same type. In contrast, a fully-connected BN without hidden nodes would require an exponential number of CPT parameters to be estimated, which would make

it practically impossible to collect the data necessary for estimating them. This problem is further compounded by the continuous values of the similarity values produced by basic matchers — in fact, it is not immediately clear how YAM would have been able to learn a fully connected BN with 13 continuous nodes representing the similarity values of each basic matcher, from the few thousand examples available from the PO dataset under the two LOOCV protocols.

On the other hand, non-linear classifiers such as decision trees [15] can indeed represent non-linear decision surfaces from a limited number of training examples, but are not inherently probabilistic, and the binary decisions output by them are not easy to use in the global assignment process that determines the entire mapping between two schemas from the pair-wise matches between their individual elements. Other probabilistic approaches to the automatic schema matching problem include the use of an attribute dictionary in the AUTOMATCH system, where training examples of matching schemas are used to compile the dictionary, and candidate elements from new schemas are compared probabilistically to the dictionary. Although this approach does result in probabilistic estimates of matches, the compilation of the dictionary requires many training examples, and is best suited to domains where many pairs of schemas have to be matched repeatedly.

6 Conclusions and Future Work

We have proposed a novel method for creating composite matchers for the purpose of automatic schema matching. Its main advantage is the explicit modeling of the conditional statistical dependence between the similarity values computed by individual basic matchers. Experiments suggest that it combines successfully the outputs of such matchers, and achieves matching accuracy significantly exceeding that of the individual matchers. Furthermore, its outputs are estimates of the genuine probabilities of match, which allows the application of decision-theoretic methods for optimal judgment whether elements match, or not. Further work will focus on leveraging the clear semantics of the computed probabilities for improving the accuracy of the global matching algorithm, as well as on improving the computational properties of the proposed Bayesian method.

As a means of comparison and investigation into the limits on the accuracy obtainable by means of matcher composition, we analyzed and identified several typical matching mistakes made by the basic matchers, and devising composition methods that could avoid them, without designing domain-specific matchers. These matchers increased the matching accuracy even further, compared to the Bayesian approach, although it remains to be seen in practice how these matchers would peform in new domains.

References

1. E. Rahm, P. A. Bernstein, A Survey of Approaches to Automatic Schema Matching, VLDB Journal, 10:4 2001.

2. H.H. Do, E. Rahm, COMA - A System for Flexible Combination of Schema Matching Approaches, in Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), 2002.
3. W. Li, C. Clifton, A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Network, Journal of Data and Knowledge Engineering 33: 1, 49-84, 2000.
4. A. Doan, P. Domingos, and A. Halevy., Learning to Match the Schemas of Databases: A Multistrategy Approach, Machine Learning Journal, no. 50, pp. 279–301, 2003.
5. S. Bergamaschi, S. Castano, M. Vincini, D. Beneventano, Semantic Integration of Heterogeneous Information Sources, Journal of Data and Knowledge Engineering 36: 3, 215-249, 2001.
6. H. H. Do, E. Rahm, Matching Large Schemas: Approaches and Evaluation, Journal of Information Systems, Vol. 32, Issue 6, Sep. 2007.
7. A.H. Doan, P. Domingos, A. Halevy, Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach, SIGMOD 2001.
8. D.W. Embley, Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. WIIW 2001.
9. D. Heckerman, A Tutorial on Learning Bayesian Networks, Journal of Learning in Graphical Models, pp. 301-354, 2001.
10. J. Tang, J. Z. Li, Using Bayesian Decision for Ontology Mapping, Journal of Web Semantics, Vol. 4, Issue 4, Dec. 2006.
11. Thiesson, B., Accelerated Quantification of Bayesian Networks with Incomplete Data, Proceedings of the Conference on Knowledge Discovery in Data, 1995, pp. 306-311.
12. Rong Pan, Yun Peng, Zhongli Ding, Belief Update in Bayesian Networks Using Uncertain Evidence, 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), 2006, pp.441-444.
13. A. Marie and A. Gal. Managing Uncertainty in Schema Matcher Ensembles. Proceedings of the 1st International Conference on Scalable Uncertainty Management. Washington, DC, October 2007, pp. 60-73.
14. A.H. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A. Halevy, Learning to Match Ontologies on the Semantic Web, The VLDB Journal 12 (4), 2003, pp. 303-319.
15. F. Duchateau, Z. Bellahsene and R. Coletta, A Flexible Approach for Planning Schema Matching Algorithms, OTM Conferences (CooPIS), 2008, pp. 249-264.
16. F. Duchateau, R. Coletta, Z. Bellahsene, R. J. Miller, Not Yet Another Matcher, Proceedings of CIKM'09, Hong-Kong, China, November 2009, pp. 2079-2080.
17. Ian H. Witten, Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques, Second Edition, Morgan Kaufmann, 2005.
18. Berlin, J. and A. Motro: Database Schema Matching Using Machine Learning with Feature Selection. CAiSE 2002, pp.452-466.
19. A. Rajesh and S.K. Srivatsa, XML Schema Matching — Using Structural Information. International Journal of Computer Applications, Vol. 8, No. 2, 34-41, 2010.