Managing Uncertainty in Schema Matcher Ensembles

Anan Marie and Avigdor Gal

Technion – Israel Institute of Technology
{sananm@cs,avigal@ie}.technion.ac.il

Abstract. Schema matching is the task of matching between concepts describing the meaning of data in various heterogeneous, distributed data sources. With many heuristics to choose from, several tools have enabled the use of schema matcher ensembles, combining principles by which different schema matchers judge the similarity between concepts. In this work, we investigate means of estimating the uncertainty involved in schema matching and harnessing it to improve an ensemble outcome. We propose a model for schema matching, based on simple probabilistic principles. We then propose the use of machine learning in determining the best mapping and discuss its pros and cons. Finally, we provide a thorough empirical analysis, using both real-world and synthetic data, to test the proposed technique. We conclude that the proposed heuristic performs well, given an accurate modeling of uncertainty in matcher decision making.

1 Introduction

Schema matching is the task of matching between concepts describing the meaning of data in various heterogeneous, distributed data sources. It is recognized to be one of the basic operations required by the process of data and schema integration [21], and thus has a great impact on its outcome.

Research into schema matching has been going on for more than 25 years now (see surveys such as [25] and various online lists, *e.g.*, OntologyMatching,[1]) first as part of a broader effort of schema integration and then as a standalone research. Due to its cognitive complexity, schema matching has been traditionally considered to be AI-complete, performed by human experts [15]. The move from manual to semi-automatic schema matching has been justified in the literature using arguments of scalability (especially for matching between large schemata [14]) and by the need to speed-up the matching process. Researchers also argue for moving to fully-automatic (that is, unsupervised) schema matching in settings where a human expert is absent from the decision process. In particular, such situations characterize numerous emerging applications triggered by the vision of the Semantic Web and machine-understandable Web resources [26]. In these applications, schema matching is no longer a preliminary task to the data integration effort, but rather ad-hoc and incremental.

[1] http://www.ontologymatching.org/

The AI-complete nature of the problem dictates that semi-automatic and automatic algorithms for schema matching will be of heuristic nature at best. Over the years, a significant body of work was devoted to the identification of *schema matchers*, heuristics for schema matching. Examples of algorithmic tools providing means for schema matching include COMA [6] and OntoBuilder [13], to name but a couple. The main objective of schema matchers is to provide schema mappings that will be effective from the user point of view, yet computationally efficient (or at least not disastrously expensive). Such research has evolved in different research communities, including databases, information retrieval, information sciences, data semantics, and others.

Although these tools comprise a significant step towards fulfilling the vision of automated schema matching, it has become obvious that the consumer of schema matchers must accept a degree of imperfection in their performance [4, 12]. A prime reason for this is the enormous ambiguity and heterogeneity of data description concepts: It is unrealistic to expect a single schema matcher to identify the correct mapping for any possible concept in a set. This argument has also been validated empirically [12]. Another (and probably no less crucial) reason is that "the syntactic representation of schemas and data do not completely convey the semantics of different databases" [22]; *i.e.*, the description of a concept in a schema can be semantically misleading. [2] went even further, arguing philosophically that even if two schemata fully agree on the semantics and the language is rich enough, schemata may still not convey the same meaning, due to some hidden semantics, beyond the scope of the schemata. Therefore, [18] argues that "[w]hen no accurate mapping exists, the issue becomes choosing the *best* mapping from the viable ones."

Choosing a heuristic that can stand up to this challenge is far from being trivial. The number of schema matchers is continuously growing, and this diversity by itself complicates the choice of the most appropriate tool for a given application domain. In fact, due to effectively unlimited heterogeneity and ambiguity of data description, it seems unavoidable that optimal mappings for many pairs of schemata will be considered as "best mappings" by none of the existing schema matchers. Striving to increase robustness in the face of the biases and shortcomings of individual matchers, several tools have enabled the use of *schema matcher ensembles*,[2] combining principles by which different schema matchers judge the similarity between concepts. The idea is appealing since an ensemble of complementary matchers can potentially compensate for the weaknesses of each other. Indeed, several studies report on encouraging results when using schema matcher ensembles (*e.g.*, see [6, 19, 8]).

In this work, we investigate means of estimating the uncertainty involved in schema matching and harnessing it to improve an ensemble outcome. We propose a model for schema matching, based on simple probabilistic principles. We then propose the use of machine learning in determining the best mapping and discuss its pros and cons. Finally, we provide a thorough empirical analysis, using both real-world and synthetic data, to test the proposed technique.

[2] The term *ensemble* is borrowed from [14, 8].

The specific contribution of this work are as follows:

- On a conceptual level, we provide a new model for schema matching, explaining the process uncertainty using simple probabilistic terms.
- We propose a new schema matching heuristic for combining existing schema matchers. The heuristic utilizes a naïve Bayes classifier, a known machine learning technique. While naïve Bayes classifiers were introduced before in the schema matching research, this technique was never applied to schema matcher ensembles.
- We present a thorough empirical analysis of the model and the heuristic, using a large data set of 230 real-world schemata as well as synthetic data. Our comparative analysis shows that the proposed heuristic performs well, given an accurate modeling of uncertainty in matcher decision making.

The rest of the paper is organized as follows. Section 2 presents the schema matching model. Section 3 introduces the new heuristic, followed by a comparative empirical analysis in Section 4. We conclude with an overview of related work (Section 5) and future research directions (Section 6).

2 Model

Let *schema* $S = \{A_1, A_2, ..., A_n\}$ be a finite set of some *attributes*. We set no particular limitations on the notion of schema attributes; attributes can be both simple and compound, compound attributes should not necessarily be disjoint, *etc.* For any schemata pair S and S', let $\mathcal{S} = S \times S'$ be the set of all possible *attribute mappings* between S and S'. Let $M(S, S')$ be an $n \times n'$ *similarity matrix* over $\mathcal{S}$, where $M_{i,j}$ represents a degree of similarity between the i-th attribute of S and the j-th attribute of S'. The majority of works in the schema matching literature define $M_{i,j}$ to be a real number in $(0, 1)$. $M(S, S')$ is a *binary similarity matrix* if for all $1 \leq i \leq n$ and $1 \leq j \leq n'$, $M_{i,j} \in \{0, 1\}$.

Schema matchers are instantiations of the schema matching process. Schema matchers differ in the way they encode the application semantics into M. Some matchers (*e.g.*, [27]) assume similar attributes are more likely to have similar names. Other matchers (*e.g.*, [19]) assume similar attributes share similar domains. Others yet (*e.g.*, [3]) take instance similarity as an indication to attribute similarity.

Let Σ be a set of possible schema mappings, where a schema mapping $\sigma \in \Sigma$ is a set of attribute mappings, $\sigma \subseteq \mathcal{S}$. Definition 1 defines the output of a schema matching process in terms of matrix satisfiability.

Definition 1 (Matrix Satisfaction). *Let $M(S, S')$ be an $n \times n'$ similarity matrix over $\mathcal{S}$ and let σ be a set of schema mappings. A schema mapping $\sigma \in \Sigma$ is said to* satisfy *$M(S, S')$ (denoted $\sigma \models M(S, S')$) if $\left(A_i, A'_j\right) \in \sigma \rightarrow M_{i,j} > 0$. $\sigma \in \Sigma$ is said to* maximally satisfy *$M(S, S')$ if $\sigma \models M(S, S')$ and for each $\sigma' \in \Sigma$ such that $\sigma' \models M(S, S')$, $\sigma' \subset \sigma$.*

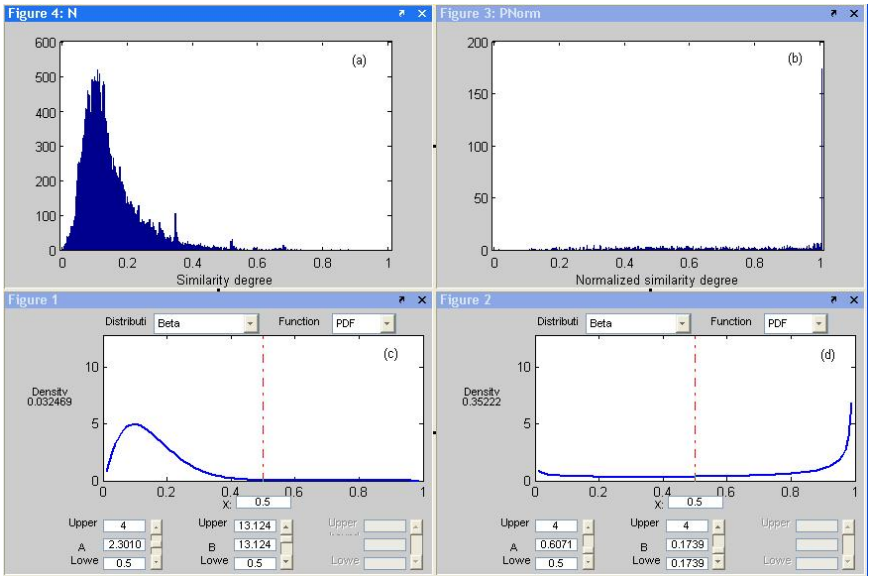

Fig. 1. Illustration of Matcher Behavior

We can therefore define the output of a schema matching process to be a similarity matrix $M(S, S')$ and derive the output schema mapping to be $\sigma \in \Sigma$ that maximally satisfies $M(S, S')$. Handling constraints such as $1:1$ cardinality constraints can be done by matrix manipulation, a topic which is beyond the scope of this paper.

When encoding the application semantic in a similarity matrix, a matcher would be inclined to put a value of 0 for each pair it conceives not to match, and a similarity measure higher than 0 (and probably closer to 1) for those attribute matches that are conceived to be correct. This tendency, however, is masked by "noise," whose sources are rooted in missing and uncertain information. Therefore, instead of expecting a binary similarity matrix, with a 0 score for all incorrect attribute mappings and a unit score for all correct attribute mappings, we would expect the values in a similarity matrix to form two probability distributions over $[0, 1]$, one for incorrect attribute mappings (with higher density around 0), and another for correct mappings.

Figure 1 provides an empirical validation for our hypothesis, based on more than 106,000 attribute mappings of 115 ontology pairs.[3] Figure 1(a) shows a distribution with a higher density around 0 that represents the similarity values that were assigned to incorrect attribute mappings by an OntoBuilder algorithm dubbed *Precedence* [13]. Figure 1(b) reflects a set of normalized similarity values of correct attribute mappings. Normalization was achieved by dividing all similarity values in a matrix by the highest value in that matrix. Figure 1 illustrates

[3] Detailed description of the data set we have used is given in Section 4.2.

that for this data set our hypothesis indeed stands and matchers indeed choose similarity values using two different distributions. Figure 1(c) and Figure 1(d) were generated using a beta distribution. According to [24]: "[t]he beta distribution can be used to model a random phenomenon whose set of possible values is in some finite interval $[c, d]$—which by letting c denote the origin and taking $d - c$ as a unit measurement can be transformed into the interval $[0, 1]$." A beta distribution has two tuning parameters, a and b. To receive a density function that is skewed to the left (as in the case of false attribute mappings, Figure 1(c)) we require that $b > a$. For right skewed density functions (as in the case of true attribute mappings, Figure 1(d)) one needs to set $a > b$. Based on the training data, and using a confidence level of 95% ($\alpha = 0.05$), the a value of the distribution of incorrect values is 2.3010 with a confidence interval of $[2.2827, 2.3192]$. We also have $b = 13.1242$ with a confidence interval $[13.0413, 13.2072]$. The confidence levels of the distribution of the correct attribute mappings are less concentrated. We have $a = 0.6071$ with a confidence interval of $[0.4871, 0.7270]$ and $b = 0.1739$ with a confidence interval of $[0.1680, 0.1798]$.

Consider a set of m schema matcher outputs $\left\{M^{(1)}, \ldots, M^{(m)}\right\}$ between two schemata S and S'. $M_{i,j}^{(l)}$ is the degree of similarity that matcher l associates with mapping the i-th attribute of S to the j-th attribute of S'. A *schema matching ensemble* is a set of schema matchers. An ensemble aggregates the similarities assigned by individual matchers to reason about the resulting aggregated ranking of alternative mappings. Such an aggregation can be modeled in various ways, one of which is presented next.

3 Naïve Bayes Heuristic

In this section we present a new heuristic for schema matching, using our schema matching model, presented in Section 2. Recall that the values in a similarity matrix are assumed to form two probability distributions over $[0, 1]$, one for incorrect attribute mappings and another for correct mappings (see Figure 1). The *naïve Bayes* heuristic attempts, given a similarity degree, to use Bayes theorem to classify an attribute mapping to one of the two groups. The naïve Bayes method is a simple probabilistic classifier that applies Bayes theorem under a strong (naïve) independence assumptions.

Given an attribute mapping (A_i, A_j) and an ensemble of matchers' output $\left\{M^{(1)}, M^{(2)}, ..., M^{(m)}\right\}$, a feature vector of (A_i, A_j) is defined to be $\left\langle M_{i,j}^{(1)}, M_{i,j}^{(2)}, ..., M_{i,j}^{(m)} \right\rangle$, where $M_{i,j}^{(l)}$ is the (i, j) similarity value of $M^{(l)}$. Let $\mathcal{F}$ be an m dimension feature space. We would like to predict the most likely target value ($v = +1$ or $v = -1$), based on the observed data sample. $+1$ stands for a correct mapping while -1 stands for an incorrect mapping. Formally, our target function is

$$f_c : \mathcal{F} \rightarrow \{+1, -1\} \tag{1}$$

The Bayesian approach to classifying a new instance (attribute mapping in our case) is to assign the most probable target value, v_{MAP}, given the attribute

values $\left\langle M_{i,j}^{(1)}, M_{i,j}^{(2)}, ..., M_{i,j}^{(m)} \right\rangle$ that describe the instance:

$$v_{MAP} = argmax_{v_j \in \{+1,-1\}} P\left\{ v_j | M_{i,j}^{(1)}, M_{i,j}^{(2)}, ..., M_{i,j}^{(m)} \right\} \quad (2)$$

Eq. 2, together with Bayes theorem and under the simplifying assumption that the similarity values are conditionally independent given the target value, is used to specify the target value output of the naïve Bayes classifier v_{NB} to be:

$$v_{NB} = argmax_{v_j \in \{+1,-1\}} P\{v_j\} \prod_{l=1}^{m} P\left\{ M_{i,j}^{(l)} | v_j \right\} \quad (3)$$

$P\{v_j\}$ is estimated by counting the frequency with which each target value $v_j \in \{+1, -1\}$ occurs in the training dataset. $P\left\{ M_{i,j}^{(l)} | v_j \right\}$, the probability to observe a mapping with similarity degree equal to $M_{i,j}^{(l)}$ given that the mapping is correct/incorrect is taken from the estimated distribution of correct and incorrect mappings, as suggested in Section 2.

Example 1. To illustrate the naïve Bayes heuristic, consider a naïve Bayes classifier with two matchers (a bivariate instance space). Each mapping is represented by a vector of length 2, consisting of the similarity values of the Precedence and Graph matchers. Figure 1 provides an illustration of the two Precedence matcher distributions used by the classifier and Table 1 (Section 4.2) provides the tuning parameters for the distributions. The number of training negative mappings is $N_{neg} = 104387$ and the number of positive training mappings is $N_{pos} = 1706$. Consider a new mapping pair with a similarity value vector $\overrightarrow{\mu} = \langle \mu_{prec}, \mu_{graph} \rangle = \langle 0.5, 0.6 \rangle$ and assume that the maximum values in the *Precedence* and *Graph* similarity matrices are $\max_{\mu}^{prec} = 0.6$ and $\max_{\mu}^{graph} = 0.8$, respectively. The probability of the mapping to be negative, given the vector of similarity measures $\overrightarrow{\mu} = \langle 0.5, 0.6 \rangle$ is

$$P(neg|\overrightarrow{\mu}) = \frac{N_{neg}}{N_{neg} + N_{pos}} \cdot P_{\alpha_{neg}^{prec}, \beta_{neg}^{prec}}(\mu_{prec}) \cdot P_{\alpha_{neg}^{graph}, \beta_{neg}^{graph}}(\mu_{graph}) \quad (4)$$

$$= \frac{104387}{104387 + 1706} \cdot 0.0097 \cdot 0.0034 = 3.2449e-005 \quad (5)$$

where $P_{\alpha_{neg}^{prec}, \beta_{neg}^{prec}}$ and $P_{\alpha_{neg}^{graph}, \beta_{neg}^{graph}}$ are the density functions of the beta distributions of the *Precedence* and *Graph* matchers, respectively. To evaluate the probability of the given mapping to be positive, one needs to first normalize the values in $\overrightarrow{\mu}$ yielding a vector $\overrightarrow{\mu}' = \langle \mu'_{prec}, \mu'_{graph} \rangle = \langle \frac{0.5}{0.6}, \frac{0.6}{0.8} \rangle$, followed by calculating $\frac{N_{pos}}{N_{neg}+N_{pos}} \cdot P_{\alpha_{pos}^{prec}, \beta_{pos}^{prec}}(\mu'_{prec}) \cdot P_{\alpha_{pos}^{graph}, \beta_{pos}^{graph}}(\mu'_{graph})$, yielding $P(pos|\langle 0.83, 0.75 \rangle) = 0.0057$. Therefore, the naïve Bayes heuristic will determine this mapping to be positive.

The time complexity of the *naïve Bayes* heuristic is $O(n^2)$, since each entry in the matrix requires a constant number of computations. As a final comment, it is worth noting that the naïve Bayes heuristic no longer guarantees, in an by itself, a 1 : 1 cardinality constraint. To enforce this requirements, a constraint enforcer [16], such as an algorithm for solving Maximum Weight Bipartite Graph problem, should be applied to the resulting binary matrix of the heuristic.

4 Experiments

We now present an empirical evaluation of our heuristic. We report in details on our experimental setup (Section 4.1), the data that was used (Section 4.2), and the evaluation methodology (Section 4.3). We then present in Section 4.4 the experiment results and provide an empirical analysis of these results.

4.1 Experiment setup

In our experiments we have used three matchers, briefly discussed below. Detailed description of these matchers can be found in [13]:

Term: Term matching compare attribute names to identify syntactically similar terms. To achieve better performance, terms are preprocessed using several techniques originating in IR research. Term matching is based on either complete word or string comparison.

Composition: A composite term is composed of other terms (either atomic or composite). Composition can be translated into a hierarchy. Similarity is determined based on the similarity of their neighbors.

Precedence: In any interactive process, the order in which data are provided may be important. In particular, data given at an earlier stage may restrict the availability of options for a later entry. When matching two terms, we consider each of them to be a pivot within its own schema, thus partitioning the graph into subgraphs of all preceding terms and all succeeding terms. By comparing preceding subgraphs and succeeding subgraphs, we determine the confidence strength of the pivot terms.

The *naïve Bayes* heuristic uses each of the three matchers as input to its feature vector. The *Naïve Bayes* heuristic was implemented using Java 2 JDK version 1.5.0_09 environment, using an API to access OntoBuilder's matchers and get the output matrices.

We have also experimented with each of the three matchers and a weighted linear combination of them into a combined matcher. This combination also included a fourth matcher, called Value, which uses the domain of attributes as evidence to their similarity. The combined matcher is clearly dependent on the other matchers and therefore violates the *naïve Bayes* heuristic assumption. The experiments were run on a laptop with Intel Centrino Pentium m, 1.50GHz CPU, 760MB of RAM Windows XP Home edition OS.

4.2 Data

For our experiments, we have selected 230 Web forms from different domains, such as dating and matchmaking, job hunting, Web mail, hotel reservation, news, and cosmetics. We extracted each Web form ontology (containing the schema and composition and precedence ontological relationships) using OntoBuilder. We have matched the Web forms in pairs (115 pairs), where pairs were taken

from the same domain, and generated the exact mapping for each pair.[4] The ontologies vary in size and the proportion of number of attribute pairs in the exact mapping relative to the target ontology. Another dimension is the size difference between matched ontologies.

We ran the four matchers and generated 460 matrices. For each such matrix, we have applied an algorithm for solving Maximum Weight Bipartite Graph problem, to generate a $1:1$ schema mapping, as a baseline comparison.

Matcher	α_{pos}	β_{pos}	α_{neg}	β_{neg}
Term	0.2430	0.0831	0.2951	4.6765
Graph	0.4655	0.1466	0.8360	9.1653
Precedence	0.6071	0.1739	2.3010	13.1242
Combined	0.6406	0.2040	2.6452	16.3139

Table 1. Beta parameters

In addition, we have generated 100 synthetic schema pairs. For each pair S and S' we have uniformly selected its schema sizes from the range $[30, 60]$. As an exact mapping we selected a set of n mapping pairs, where n takes one of three possible values, $n_1 = \min(|S|, |S'|)$, $n_2 = 0.5n_1$, and $n_3 = 2n_1$. For n_1 we have enforced a $1:1$ cardinality constraint. n_2 represents a situation in which not all attributes can be mapped and n_3 represents a matching that is not of $1:1$ cardinality. Then, using the beta distributions learned from the training data for each of the four matchers we have created, for each schema pair, four synthetic matrices, one per a matcher using a beta generator class *cern.jet.random.Beta* distributed with *colt.jar* jar file. The entries of a matrix use $(\alpha_{pos}, \beta_{pos})$ and $(\alpha_{neg}, \beta_{neg})$ parameters (See Table 1) for the beta distribution of the positive mapping measures and the negative mapping measures, respectively.

4.3 Evaluation methodology

We use two main evaluation metrics, namely Precision and Recall. Precision is computed as the ratio of correct element mappings, with respect to some exact mapping, out of the total number of element mappings suggested by the heuristic. Recall is computed as the ratio of correct element mappings, out of the total number of element mappings in the exact mapping. Both Recall and Precision are measured on a $[0, 1]$ scale. An optimal schema matching results in both Precision and Recall equal to 1. Lower precision means more false positives, while lower recall suggests more false negatives. To extend Precision and Recall to the case of non $1:1$ mappings, we have adopted a correctness criteria according to which any attribute pair that belongs to the exact mapping is considered to be correct, even if the complex mapping is not fully captured. This method aims at compensating the matchers for the $1:1$ cardinality enforcement.

[4] All ontologies and exact mappings are available for download from the OntoBuilder Web site, http://ie.technion.ac.il/OntoBuilder).

It is worth noting that our test was conducted on a wide range of real-world schemata. Such a real world challenge was tried at the 2006 OAEI ontology matching evaluation [10] with average performance of 39.25% Precision, 40.40% Recall, and 39.82% F-Measure.[5]

4.4 Results and analysis

We are now ready to present our results and empirical analysis. We present a comparative analysis of the proposed heuristic with existing heuristics, using the full data set. We then analyze two obstacles in successfully using the heuristic and describe two additional experiments, aimed at evaluating the impact of each such obstacle on the heuristic performance.

Comparative Performance Analysis In our first experiment we provide a comparative analysis of the performance of the naïve Bayes heuristic with four heuristics that enforce a mapping cardinality of 1 : 1. Figure 2 illustrates the results. The x axis represents the four different data sets, with Precision on the y axis in Figure 2(left) and Recall in Figure 2(right).

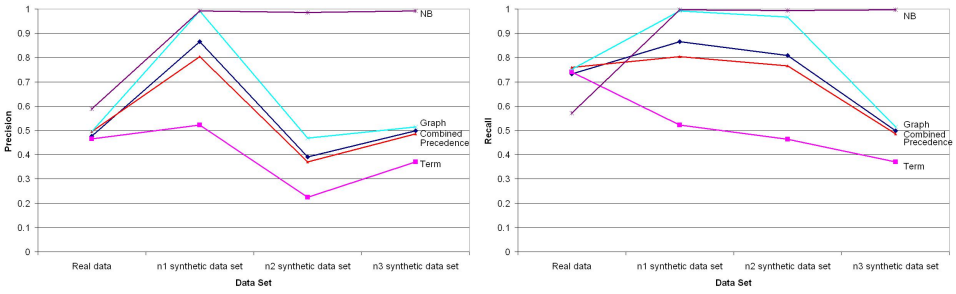

Fig. 2. Comparative Performance Analysis

In terms of Precision, the naïve Bayes heuristic outperforms all other heuristics. For the real data set, this improvement in Precision comes at the cost of Recall. This disadvantage disappears in the simulated data, where the naïve Bayes heuristic dominates other heuristics, even for the simulated data with n_1, where the 1 : 1 cardinality constraint holds (although not enforced for the proposed heuristic). For this case, the Graph heuristic comes in very close behind.

Two main reasons may explain this behavior. First, the naïve assumption of independence does not hold in our case, since OntoBuilder heuristics are all heavily based on syntactic comparisons. Second, it is possible that the training data set, based on which the beta distributions are determined, does not serve

[5] See http://keg.cs.tsinghua.edu.cn/project/RiMOM/oaei2006/main.html for details.

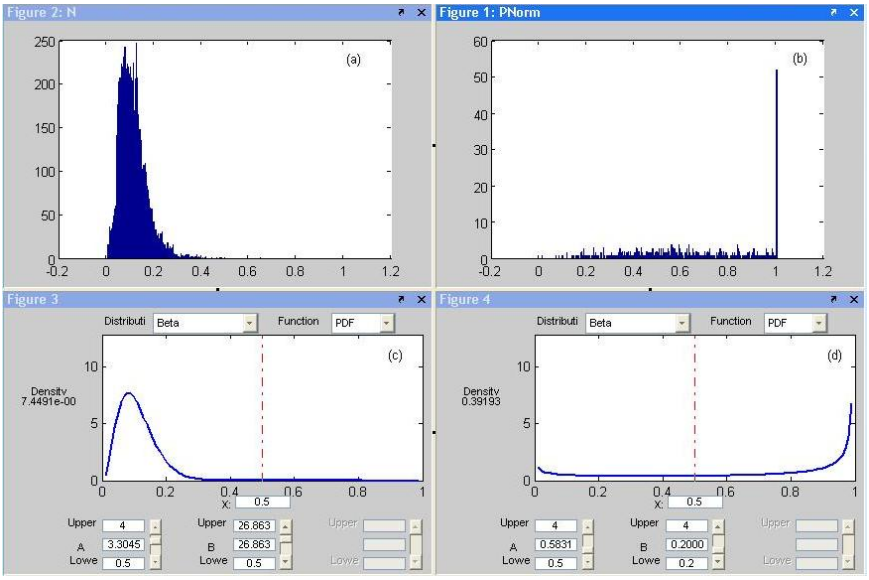

Fig. 3. Illustration of Matcher Behavior after Outlier Elimination

as a good estimator for the matchers decision making. We shall now investigate these hypotheses in more depth.

Cleaning the Data In this experiment we have eliminated from the real-world data set some of the matrices that have a high percentage of outliers. In statistics, an outlier is an observation that is numerically distant from the rest of the data. Outliers in our matrices involve correct attribute mappings with very low similarity measures and incorrect attribute mappings with relatively high similarity measures. To compute (either positive or negative) outliers, we define $Q1$ and $Q3$ to be first and third quartiles, respectively, over all the similarity measures of the positive (negative) mappings, and IQR to be the interquartile range $Q3-Q1$. An outlier is a mapping that its similarity measure $\mu < Q1-1.5\cdot IQR$ or $\mu > Q3+1.5\cdot IQR$. We have ranked all the Combined heuristic matrices in a decreasing order of outlier numbers and chose the top 51 schema pairs (153 matrices), based on which we have regenerated the beta distributions for all heuristics. Figure 3 presents the beta distribution of the Preference heuristic for the new data set. Compared with Figure 1, we see that the distribution of normalized values of correct attribute mappings remain stable**.** The distribution of incorrect attribute mappings is tighter here, yielding lower variance. As a result, we expect less false negatives and increasing Recall. The confidence levels of the new distributions reveal a slightly different story. Again, we are looking at a confidence level of $\alpha = 0.05$. For the incorrect attribute mappings we have $a = 3.3045$ with a confidence interval of $[3.2680, 3.3409]$ and $b = 26.8633$ with a confidence interval of $[26.5499, 27.1768]$. For correct attribute mappings, $a =$

0.5831 with a confidence interval of $[0.4076, 0.7586]$ and $b = 0.2$ with a confidence interval of $[0.191, 0.209]$. For all parameters, we observe an increased confidence interval, suggesting a possibly higher rate in probability estimation.

Matcher	Change in Precision	Change in Recall
Term	−8.13%	−5.97%
Graph	−8.97%	−8.88%
Precedence	−7.15%	−6.37%
Combined	−8.76%	−9.06%
NB	−4.28%	1.93%

Table 2. Change of Precision and Recall between Data Sets

Table 2 summarizes the changes in Precision and Recall between the full data set and the reduced one. The results show that indeed Recall was increased for the naïve Bayes heuristic. It comes at the cost of Precision, indicating an increase in the number of false positives in parallel. A somewhat surprising result was that all other matchers performed worse on the reduced set. In any event, these changes were not extremely big, which leads us to hypothesize that the naïve Bayes heuristic performance in both data sets was impaired by the invalid assumption of matcher independence.

Simulating Matcher Independence To test the performance of the naïve Bayes heuristic in a setting of matcher independence, we have used the synthetic matrices. In this synthetic data sets, while all values in each matrix were generated using the same distribution, a specific attribute pair is assigned a value by a matcher independently of other matchers.

A comparison of the performance of the naïve Bayes heuristic with the same three heuristics we have used before are given in Figure 2 above. We observe that Precision improves for all matchers, when using the synthetic data and keeping the $1:1$ cardinality constraints. This is most likely due to the way the matrices are generated. The amount of improvement depends on the beta distribution parameters of each matcher. For example, the Term matcher has a weaker distinction between correct and incorrect mappings, yielding less accurate prediction. This may also explain the reduced Recall for the Term matcher, while all other matchers increase their Recall measure.

The naïve Bayes heuristic dominates the other matchers for all synthetic data, indicating that indeed the matcher independence assumption serves as an obstacle to better performance. Another interesting observation involves the ability of the naïve Bayes heuristic to manage non $1:1$ mappings. The other four matchers show a sharp decline in Precision for n_2, since about half of their attribute mappings are bound to be incorrect. For n_3 we see an increase in Precision, since the range of possibilities of mapping correctly has significantly increased. For Recall, we see deterioration with the n_3 data set, due to the

inability of these matchers to choose attributes that violate the $1:1$ cardinality constraint. We note that the naïve Bayes heuristic maintains an almost perfect Precision and Recall for all three synthetic data sets, which means, among other things, that the specific method for measuring Precision and Recall for the n_2 and n_3 data sets could not affect the true effectiveness of the heuristic.

5 Related Work

In this section we focus on two specific aspects that are most relevant to this work, namely uncertainty management in schema matching and the use of machine learning in schema matching.

5.1 Uncertainty Management

Attempts to provide clear semantics to schema matching involves model theory for schema mappings [1, 18, 2]. In [1] mappings were represented using schema morphisms in categories. Roughly speaking, a category is a collection of schemata and their inter-schema mappings, represented as a morphisms. A morphism can be composed in an associative manner. Morphisms are designed so that they preserve integrity constraints among schemata. The work in [18] provides explicit semantics to mappings, using models and satisfiability. [2] provides a formal model of schema matching for topic hierarchies, rooted directed trees, where a node has a "meaning," generated using some ontology. A schema matcher (schema matching method in the authors own terminology) is a function from a mapping to a boolean variable. The limitations in this line of models, with respect to uncertainty modeling was presented in [18, 2] and discussed in Section 1.

The research described in [12] proposes a model that represents uncertainty (as a measure of imprecision) in the matching process outcome. In [11], building on the results of [12], the current "best mapping" approach was extended into one that considers top-K mappings as an uncertainty management tool. In this work, we propose a model for estimating the level of uncertainty in matcher decision making and offer a heuristic to harness uncertainty and improve on existing matching methods.

5.2 Machine Learning and Schema Matching

Machine learning has been used for schema matching in several works. Autoplex [3], LSD [7], and iMAP [5] use a naïve Bayes classifier to learn attribute mappings probabilities using instance training set. SEMINT [17] use neural networks to identify attribute mappings. APFEL [9] determine heuristic weights in an ensemble and threshold levels using various machine learning techniques, namely decision trees in general and C4.5 in particular, neural networks, and support vector machines. C4.5 was also used in [28], using WordNet relationships as features. sPLMap [23] use naïve Bayes, kNN, and KL-distance as content-based classifiers. All these works applied machine learning directly to the schemata, while our approach is to apply it to the similarity matrix outcome.

6 Conclusions

In this work we have presented a heuristic for schema matching, based on a probabilistic model of matchers and a well-known machine learning classifier. We have empirically analyzed the properties of the naïve Bayes heuristic using both real world and synthetic data. Our empirical analysis shows that the proposed heuristic performs well, given an accurate modeling of uncertainty in matcher decision making. We have also discussed the current limitations of the heuristic, and in particular its naïve assumption regarding matcher independence. Therefore, future research involves fine tuning the similarity measure distribution estimation. We will also look into more advanced methods (*e.g.*, discriminant analysis [20]) that do away with the independence assumption of the naïve Bayes classifier.

References

[1] S. Alagic and P. Bernstein. A model theory for generic schema management. In *Database Programming Languages, 8th International Workshop, DBPL 2001, Frascati, Italy, September 8-10, 2001*, pages 228–246, 2001.

[2] M. Benerecetti, P. Bouquet, and S. Zanobini. Soundness of schema matching methods. In *Proceedings of ESWC 2005*, pages 211–225, 2005.

[3] J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2001.

[4] P. Cudré-Mauroux et al. Viewpoints on emergent semantics. *Journal on Data Semantics*, 6:1–27, 2006.

[5] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: Discovering complex mappings between database schemas. In *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 383–394, 2004.

[6] H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the International conference on Very Large Data Bases (VLDB)*, pages 610–621, 2002.

[7] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In W. G. Aref, editor, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 509–520, Santa Barbara, California, May 2001. ACM Press.

[8] C. Domshlak, A. Gal, and H. Roitman. Rank aggregation for automatic schema matching. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(4):538–553, 2007.

[9] M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with apfel. In *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, pages 186–200, 2005.

[10] J. Euzenat, M. Mochol, O. Svab, V. Svatek, P. Shvaiko, H. Stuckenschmidt, W. van Hage, and M. Yatskevich. Introduction to the ontology alignment evaluation 2006. In *Proceedings of Ontology Matching 2006 Workshop at ISWC'06*, 2006.

[11] A. Gal. Managing uncertainty in schema matching with top-k schema mappings. *Journal of Data Semantics*, 6:90–114, 2006.

[12] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB Journal*, 14(1):50–67, 2005.

[13] A. Gal, G. Modica, H. Jamil, and A. Eyal. Automatic ontology matching using application semantics. *AI Magazine*, 26(1):21–32, 2005.

[14] B. He and K.-C. Chang. Making holistic schema matching robust: an ensemble approach. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 429–438, 2005.

[15] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 51–61. ACM Press, 1997.

[16] Y. Lee, M. Sayyadian, A. Doan, and A. Rosenthal. eTuner: tuning schema matching software using synthetic scenarios. *VLDB Journal*, 16(1):97–122, 2007.

[17] W.-S. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, 2000.

[18] J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 80–86, 2002.

[19] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of the International conference on Very Large Data Bases (VLDB)*, pages 49–58, Rome, Italy, Sept. 2001.

[20] G. Marcoulides and S. Hershberger. *Multivariate Statistical Methods*. Lawrence Erlbaum Associates, 1997.

[21] S. Melnik. *Generic Model Management: Concepts and Algorithms*. Springer-Verlag, 2004.

[22] R. Miller, L. Haas, and M. Hernández. Schema mapping as query discovery. In A. E. Abbadi, M. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the International conference on Very Large Data Bases (VLDB)*, pages 77–88. Morgan Kaufmann, 2000.

[23] H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Information Processing and Management*, 43(3):552–576, 2007.

[24] S. Ross. *A First Course in Probability*. Prentice Hall, 5 edition, 1997.

[25] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal of Data Semantics*, 4:146 – 171, Dec. 2005.

[26] B. Srivastava and J. Koehler. Web service composition - Current solutions and open problems. In *Workshop on Planning for Web Services (ICAPS-03)*, Trento, Italy, 2003.

[27] W. Su, J. Wang, and F. Lochovsky. Aholistic schema matching for web query interfaces. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, pages 77–94, 2006.

[28] L. Xu and D. Embley. A composite approach to automating direct and indirect schema mappings. *Information Systems*, 31(8):697–886, Dec. 2006.