Intelligent Data Analysis 12 (2008) 253–270
IOS Press

Schema matching on streams with accuracy guarantees

Szymon Jaroszewicz[a], Lenka Ivantysynova[b] and Tobias Scheffer[c]
[a]*National Institute of Telecommunications, Warsaw, Poland*
E-mail: s.jaroszewicz@itl.waw.pl
[b]*Humboldt-Universität zu Berlin, Berlin, Germany*
E-mail: lenka@wiwi.hu-berlin.de
[c]*Max Planck Institute for Computer Science, Saarbrücken, Germany*
E-mail: scheffer@mpi-inf.mpg.de

Abstract. We address the problem of matching imperfectly documented schemas of data streams and large databases. Instance-level schema matching algorithms identify likely correspondences between attributes by quantifying the similarity of their corresponding values. However, exact calculation of these similarities requires processing of all database records – which is infeasible for data streams. We devise a fast matching algorithm that uses only a small sample of records, and is yet guaranteed to find a matching that is a close approximation of the matching that would be obtained if the entire stream were processed. The method can be applied to any given (combination of) similarity metrics that can be estimated from a sample with bounded error; we apply the algorithm to several metrics. We give a rigorous proof of the method's correctness and report on experiments using large databases.

1. Introduction

Schema matching is the process of identification of semantic correspondences between attributes across multiple database schemas [2]. Schema matching has recently gained in importance due to frequent multinational company mergers such as Daimler-Benz merging with Chrysler. Such endeavors require integrating both companies' databases and data warehouses. This is an enormous task: there are thousands of poorly documented database tables, hundreds of attributes each. Even partial automation of the process can be a big help.

The schema matching process constitutes a knowledge discovery challenge in its own right. Research revolves around the question how schema matching can be supported, or even automated, effectively. Schema matching algorithms generally match elements of the distinct schemas such that some similarity criterion is maximized. Schema-level matchers use a similarity criterion that refers to schema information such as attribute names, descriptions, or the schemas' structure. Unfortunately, the available schema information is often insufficient. Cases arise in which attribute names are opaque and clues on their semantics can only be found by inspecting the values. Instance-level schema matchers quantify the similarity of attributes by comparing properties of their values.

In order to guarantee that the produced mapping in fact maximizes the chosen similarity function, instance-level matchers need to exercise at least one entire pass over the databases. A complete pass is unreasonable for transactional databases that record high-speed data streams as they occur, for instance,

in retail chains, and telecommunication and banking applications and can easily grow into the order of hundreds of terabytes. An arbitrary and uninformed choice of a small sample size of transactions results in a loss of any guarantee on the optimality of the produced match. Known methods that use sampling do not guarantee any properties of the retrieved mappings.

We formalize the matching problem in a way that is both mathematically rigorous and closely tied to practical applications. We devise a schema matching algorithm, which is *guaranteed* to produce a "near-optimal" match; we detail the term "near-optimal" by means of probabilistic bounds on the difference between the retrieved matching and a matching that would result if *all* database transactions were processed. The guarantee holds even though the algorithm relies on sampling. In fact the efficiency of our algorithm does not depend on the size of the original database to which it is applied, only on the desired accuracy and error probability.

A useful similarity metric for the matching problem at hand is a base requirement for any sampling algorithm, and designing such a problem-specific metric clearly is a challenging issue in itself. Therefore, we design our solution to be generic with respect to the similarity criterion; weighted combinations of several schema-level and instance-level criteria can be applied. We review a selection of criteria, many more can be inserted into the algorithm.

The rest of this paper is organized as follows: after reviewing related work in Section 2, we formalize the problem in Section 3. We phrase the schema matching problem in a way that pays tribute to typical practical application scenarios, and is sufficiently rigid to allow us to state and rigorously prove the correctness of our solutions. We detail our solution in Section 4, discuss attribute similarity metrics in Section 5 and report on experimental results on real-world data in Section 6. Section 7 concludes.

2. Related work

Bernstein and Rahm [16] provide an overview of the schema matching problem [10,13] and a taxonomy of schema matching algorithms. Matchers can be dichotomized into schema-level and instance-level methods. Schema-level matchers maximize some similarity function that refers to attribute names and other structural information (e.g., [4,6,12,15]) whereas the similarity metrics employed by instance-level approaches (e.g., [1,10,14]) refer to the instance data; that is, to the attributes' values. The instance-level approach can even be applied in the complete absence of useful schema-level information [9] and it allows to even identify complex matching relations – *e.g.*, a factor-equivalence relation between an income attribute in Euros and a corresponding salary in Yen [3].

Most practical systems – such as Clio [19], SemInt [11], and LSD [5] – combine schema- and instance-level similarity metrics. Assembling and balancing the similarity metric requires experience and domain knowledge; machine learning [5] and experiments on synthetic matching problems can guide the parameter optimization [17]. The algorithm that we present supports the possibility of combining schema- and instance-level similarity metrics. However, accessing the database records is only required for calculation of instance- level metrics.

Our algorithm relies on sampling and on data-dependent confidence bounds. Sampling strategies play an important role for a range of data mining tasks. They have been used to find the approximately most interesting association rules [7,18], or the most significant differences between graphical models and corresponding databases [8].

3. Approximately optimal schema matching

We focus on the following *schema matching problem setting*. We are given two schemas R and S over attributes $\langle r_1, \ldots, r_n \rangle$ and $\langle s_1, \ldots, s_m \rangle$, respectively. The schemas contain imperfect information about the type and meaning of the attributes. The goal of schema matching is to identify pairs of attributes (r_i, s_j) that are semantically identical. In order to approach this problem computationally, a similarity metric $f_{R,S}(r_i, s_j)$ quantifies the similarity of attributes r_i and s_j. The similarity can exploit schema-level information such as attribute names if such information is available, and instance-level information. Instance-level similarity metrics refer to the values that occur in R and S. Naturally, instance-level information is always available – unless the databases are empty or inaccessible.

The contribution of this paper is relevant when the similarity metric $f_{R,S}(r_i, s_j)$ involves instance-level information. In order to evaluate an instance-level metric exactly, it is generally necessary to exercise a pass over the databases R and S; the main challenge that we address is that this is impossible for streams. The similarity criterion $f_{R,S}(r_i, s_j)$ is problem-specific and given in advance. We will only have to assume that it is possible to obtain an estimate of $f_{R,S}$ that lies within a bounded confidence interval.

In practice, the schema matching process is highly interactive. A user typically seeks to find out which attributes in R have counterparts in S and identify these counterparts. We phrase this problem setting as follows. *We seek to construct an algorithm that finds, for every attribute in R, a set of the k best matching candidates in S, sorted by their similarity to the attribute from R, and vice versa for all attributes in S. In addition, the algorithm has to order all attributes in R and S by their likelihood of having a counterpart in the other schema.* Our formulation of the problem setting builds upon ε-correct orderings of sequences. In an ε-correct ordering, values are sorted in decreasing order, but in case of near-ties between adjacent values (similarity differences of no more than ε), any ordering is considered correct. The special case of $\varepsilon = 0$ corresponds to a regular ordering.

Definition 1. ε-Correct Ordering. *A sequence of values $(x_1, \ldots, x_k)$ is in ε-correct ordering if, for all i between 1 and k and all j between 1 and $i-1$, the inequality $x_j \geqslant x_i - \varepsilon$ is satisfied.*

In order to formalize the *approximately optimal matching problem* we refer to stochastic approximations with confidence bounds. Intuitively, the parameters ε and δ that are involved have the following meaning. When the algorithm is restarted multiple times, then the resulting match is "ε-close" to the optimal match in a fraction of at least $1 - \delta$ of all runs.

Definition 2. Approximately Optimal Matching. *Given schemas R and S over attributes $\langle r_1, \ldots, r_n \rangle$ and $\langle s_1, \ldots, s_m \rangle$ respectively, a desired number of matches k, approximation and confidence parameters ε and δ, find a sequence of k attributes $(s_{i_1}, \ldots, s_{i_k})$ for each r_i in R, a sequence of k attributes $(r_{j_1}, \ldots, r_{j_k})$ for each s_j in S, and permutations π^r and π^s such that, with confidence $1 - \delta$,*

1. *for all attributes r_i, the retrieved sequence $(s_{i_1}, \ldots, s_{i_k})$ contains the best matching attributes (accurately up to ε) and, analogously, the sequence $(r_{j_1}, \ldots, r_{j_k})$ contains the best matching attributes for each s_j; i.e., there is no offending $s_{i'} \notin (s_{i_1}, \ldots, s_{i_k})$ such that $f_{R,S}(r_i, s_{i'}) > \min_{i'' \in \{i_1, \ldots, i_k\}} f_{R,S}(r_i, s_{i''}) + \varepsilon$ and no $r_{j'} \notin (r_{j_1}, \ldots, r_{j_k})$ such that $f_{R,S}(r_{j'}, s_j) > \min_{j'' \in \{j_1, \ldots, j_k\}} f_{R,S}(r_{j''}, s_j) + \varepsilon$;*
2. *for all attributes r_i, the corresponding $(s_{i_1}, \ldots, s_{i_k})$ are ordered by their similarity to r_i and for all s_j, the corresponding $(r_{j_1}, \ldots, r_{j_k})$ are ordered by their similarity to s_j; i.e., $(f_{R,S}(r_i, s_{i_1}), \ldots, f_{R,S}(r_i, s_{i_k}))$ and $(f_{R,S}(r_{j_1}, s_j), \ldots, f_{R,S}(r_{j_k}, s_j))$ are ε-correct orderings;*

3. *the permutations π^r and π^s sort the attributes of R and S, respectively, according to their likelihood of having a matching partner in the other schema; i.e., the sequences* $(\max_{j'} f_{R,S}(r_{\pi^r(1)}, s_{j'}), \ldots, \max_{j'} f_{R,S}(r_{\pi^r(n)}, s_{j'}))$ *and* $(\max_{i'} f_{R,S}(r_{i'}, s_{\pi^s(1)}), \ldots, \max_{i'} f_{R,S}(r_{i'}, s_{\pi^s(m)}))$ *are ε-correct orderings.*

Our concept of an optimal matching intrinsically refers to the similarity metric. An approximately optimal matching is one that is close, with high probability, to the optimal matching in which each attribute has a similar matching partner. Whether this approximately optimal matching is a good matching in practice additionally depends on the appropriateness of the similarity metric.

4. Schema matching algorithms

We will present two algorithms that differ from each other in the way they derive the sample size required to solve the approximately optimal matching problem. The first algorithm, called FSM, determines a "worst-case" sample size without accessing the data; the sample size only depends on the similarity metric and parameters ε and δ. The sample size determined suffices to assure the quality guarantee for any possible stream; it may be pessimistically large for a given, particular stream. The second algorithm is called progressive FSM. Its sample size depends on the database at hand. The sample-dependent bounds are technically more involved than the worst-case bounds; therefore, FSM is a natural baseline for progressive FSM and one of the questions that we will investigate empirically is whether progressive FSM outperforms the simpler FSM algorithm.

Both algorithms use *similarity confidence intervals* which bound the similarity $f_{R,S}$. These confidence intervals are of salient importance to our solution because they bound the inaccuracy incurred by the estimation process. A similarity confidence interval lays out a range of values such that, with a probability of at least $1-\delta$, the true similarity lies within that range. The benefit is that a confidence interval can be calculated based on a random sample of any size whereas a full database pass is required to determine the true similarity value. The confidence interval is wide for small samples and tightens for increasingly large samples. Similarity confidence intervals can be constructed for a wide range of instance-level similarity functions.

Definition 3. Similarity Confidence Interval. *Let $f_{R,S}$ be a similarity function. Let R', S' be random samples from R and S; Then, $\lfloor f_{R',S'} \rfloor_\delta$ and $\lceil f_{R',S'} \rceil_\delta$ form a similarity confidence interval if, with probability at least $1-\delta$, there is no pair of any $r_i \in R$ and $s_j \in S$ that violates*

$$\lfloor f_{R',S'}(r_i, s_j) \rfloor_\delta \leqslant f_{R,S}(r_i, s_j) \leqslant \lceil f_{R',S'}(r_i, s_j) \rceil_\delta. \tag{1}$$

With probability $1-\delta$, the true similarity of *all* the pairs (r_i, s_j) are within their bounds. The probability of any violation by at least one pair is bounded by δ. We will detail intervals for selected similarity functions in Section 5.

The first algorithm, "FSM", is detailed in Table 1. In Step 1, it uses a binary search to determine the smallest sample size N that suffices to guarantee that the similarity confidence intervals are tight up to ε. N is calculated without taking any samples from the database. For a given similarity metric $f_{R,S}$, the corresponding confidence interval that extends from $\lceil f_{R',S'}(\cdot,\cdot) \rceil_\delta$ to $\lfloor f_{R',S'}(\cdot,\cdot) \rfloor_\delta$ is inserted into the algorithm. (Parameterization with "$\cdot$" indicates "for any pair of attributes".)

By contrast, the progressive FSM algorithm described in Table 2 uses sample-dependent bounds. It starts to draw batches of example records from the databases and then calculates confidence intervals in

Table 1
FSM: Fast schema matching

Input: Schemas R and S, respectively; desired number of candidates k; approximation and confidence parameters ε and δ.

1. **Let** N be the smallest number such that $\lceil f_{R',S'}(\cdot,\cdot)\rceil_\delta - \lfloor f_{R',S'}(\cdot,\cdot)\rfloor_\delta \leqslant \varepsilon$ when R' and S' contain N records. (Where we parameterize the bound with "$\cdot$", this means "for any argument".)
2. Draw N records at random from R and S, let them be R' and S'.
3. **For** all (r_i, s_j), determine $\hat{f}_{R',S'}(r_i, s_j) = \frac{1}{2}(\lceil f_{R',S'}(r_i, s_j)\rceil_\delta + \lfloor f_{R',S'}(r_i, s_j)\rfloor_\delta)$ based on the samples.
4. **For** all attributes r_i: **Let** $H^*_{r_i}$ be the k elements s_j that maximize $\hat{f}_{R',S'}(r_i, s_j)$;
   For all attributes s_j: **Let** $H^*_{s_j}$ be the k elements r_i that maximize $\hat{f}_{R',S'}(r_i, s_j)$.
5. **Let** π^r sort the attributes in R by $\max_{s_{i'} \in H^*_{r_i}} \hat{f}_{R',S'}(r_i, s_{i'})$;
   Let π^s sort the attributes in S by $\max_{r_{j'} \in H^*_{s_j}} \hat{f}_{R',S'}(r_{j'}, s_j)$.
6. **Return** π^r, π^s, and for all attributes p in $R \cup S$: return the sequence of elements in H^*_p sorted by $\hat{f}_{R',S'}(p, q)$ as k best matches.

Table 2
Progressive FSM algorithm

Input: Schemas R and S, respectively; desired number of candidates k; approximation and confidence parameters ε and δ.

Let $R' = S' = \emptyset$. **Let** the batch size parameter be 10,000, by default. **Let** $t = 1$.

Let N be the smallest number such that $\lceil f_{R',S'}(\cdot,\cdot)\rceil_{\frac{\delta}{2}} - \lfloor f_{R',S'}(\cdot,\cdot)\rfloor_{\frac{\delta}{2}} \leqslant \varepsilon$ when R' and S' contain N records. **Let** $T = N/$batch size.

For $t = 1 \ldots T$ **Repeat:**

1. **Let** $\delta_t = \frac{\delta}{2t(t+1)}$.
2. Draw batches of records at random from R and S, and add them to R' and S', respectively.
3. **For** all (r_i, s_j), calculate $\lfloor f_{R',S'}(r_i, s_j)\rfloor_{\delta_t}$, $\lceil f_{R',S'}(r_i, s_j)\rceil_{\delta_t}$, and $\hat{f}_{R',S'}(r_i, s_j) = \frac{1}{2}(\lceil f_{R',S'}(r_i, s_j)\rceil_{\delta_t} + \lfloor f_{R',S'}(r_i, s_j)\rfloor_{\delta_t})$. $(\lfloor f_{R',S'}(r_i, s_j)\rfloor_\delta + \lceil f_{R',S'}(r_i, s_j)\rceil_\delta)$.
4. **If** all pairs (r_i, s_i) satisfy that $\lceil f_{R',S'}(r_i, s_j)\rceil_{\delta_t} - \lfloor f_{R',S'}(r_i, s_j)\rfloor_{\delta_t} \leqslant \varepsilon$ **then break.**

For all attributes r_i: **Let** $H^*_{r_i}$ be the k elements s_j that maximize $\hat{f}_{R',S'}(r_i, s_j)$; **For** all attributes s_j: **Let** $H^*_{s_j}$ be the k elements r_i that maximize $\hat{f}_{R',S'}(r_i, s_j)$.

Let π^r sort the attributes in R by $\max_{s^*_i \in H^*_{r_i}} \hat{f}_{R',S'}(r_i, s^*_i)$; **Let** π^s sort the attributes in S by $\max_{r^*_j \in H^*_{s_j}} \hat{f}_{R',S'}(r^*_j, s_j)$.

Return π^r, π^s, and for all attributes p in $R \cup S$: return the sequence of elements in H^*_p sorted by $\hat{f}_{R',S'}(p, q)$ as k best matches.

each step that depend on the sample processed so far. The progressive algorithm can terminate earlier than the sample-independent bound would have suggested, depending on characteristics of the database. The progressive algorithm terminates early if it is "easy" to confidently identify the matching partners for the attributes. Identifying matching partners is easy if each attribute in R is similar to one, or very few, attributes in S and dissimilar from most others. Our main result is that both, the FSM and progressive FSM algorithm solve the approximately optimal matching problem.

Theorem 1. *Let $\lceil f_{R',S'}(\cdot,\cdot)\rceil_\delta$ and $\lfloor f_{R',S'}(\cdot,\cdot)\rfloor_\delta$ be similarity confidence bounds that satisfy Definition 3. Then both, the FSM* (*Table 1*) *and progressive FSM algorithm* (*Table 2*) *find, for any pair of schemas R and S and any input parameters $0 < \delta \leqslant 1$, $0 < \varepsilon$, and k, an approximately optimal matching as detailed in Definition 2.*

The proof of Theorem 1 is given in Appendix A. Copies of the implementation can be obtained for research purposes from the authors.

5. Attribute similarity metrics

We will now discuss three exemplary similarity measures for attributes and show how confidence bounds can be derived. We will sketch the individual bounds as lemmas; the main result of this section is the bound for the weighted sum of all presented similarity measures. Any similarity metric can be integrated into the algorithm, if it can be estimated with bounded error based on a sample.

Various similarity measures require measuring the similarity of two probability distributions. For this purpose we will use a measure E based on Euclidean distance. Let $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ be probability distributions. E is defined as $E(P,Q) = 2 - \sum_{i=1}^{n}(p_i - q_i)^2$. We subtract the sum of squares from 2 in order to guarantee that E has high values for similar distributions, while remaining nonnegative.

Similarity measures will take two attributes as arguments and will be denoted with lowercase letter f with different superscripts. When deriving the bounds for our similarity metrics, the following (well-known) observations will be helpful. The following are upper and lower bounds on an underlying probability p, given a corresponding observed frequency (a count of events divided by the total number of observations) $\hat{p}$.

$$\lfloor p \rfloor_\delta = \max(\hat{p} - e, 0), \ \lceil p \rceil_\delta = \min(\hat{p} + e, 1),$$

where $\hat{p}$ is the value of p estimated from data. The term e can be expressed in terms of the well-known Hoeffding inequality or in terms of the inverse normal distribution. Based on Hoeffding's inequality, $e = \sqrt{\frac{1}{2N}\log\frac{2}{\delta}}$; and using the Normal distribution, $e = z_{1-\frac{\delta}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$, where N denotes the sample size and $z_{1-\delta}$ is the $1-\delta$ quantile of the normal distribution that can be looked up in a table.

5.1. Euclidean distance

Given two attributes r and s with identical domains, a natural measure of their similarity is given by

$$f^E_{R,S}(r,s) = E(P_r, Q_s), \tag{2}$$

where P_r and Q_s are probability distributions of r and s respectively.

Note that the requirement of equal domains is very strict. It can however easily be removed by replacing the domain of each attribute with the union of both original domains.

Lemma 1. *Let $f_{R,S}(r_i, s_j)$ be defined as in Eq. (2). Then for given schemas R, S and all pairs of attributes $r \in R, s \in S$, the following similarity confidence intervals satisfy Definition 3.*

$$\lfloor f^E_{R',S'}(r,s) \rfloor_\delta = \left\{ 2 - \sum_{i=1}^{n_r} \left((\lceil p_{r,i} \rceil_{\frac{\delta}{M}})^2 + (\lceil q_{s,i} \rceil_{\frac{\delta}{M}})^2 - 2 \lfloor p_{r,i} \rfloor_{\frac{\delta}{M}} \lfloor q_{s,i} \rfloor_{\frac{\delta}{M}} \right) \right\}, \tag{3}$$

$$\lceil f^E_{R',S'}(r,s) \rceil_\delta = \left\{ 2 - \sum_{i=1}^{n_r} \left((\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}})^2 + (\lfloor q_{s,i} \rfloor_{\frac{\delta}{M}})^2 - 2 \lceil p_{r,i} \rceil_{\frac{\delta}{M}} \lceil q_{s,i} \rceil_{\frac{\delta}{M}} \right) \right\}, \tag{4}$$

where R', S' are samples taken from R and S respectively, n_r and n_s are the size of the domains of r and s, $(p_{r,1}, \ldots, p_{r,n_r})$ and $(q_{s,1}, \ldots, q_{s,n_s})$ are probability distributions of r and s respectively, and $M = \sum_{r \in R} n_r + \sum_{s \in S} n_s$, over categorical attributes.

The proof of lemma 1 is given in Appendix B.

5.2. Permuted Euclidean distance

When comparing two categorical attributes, it is not clear which values correspond to each other. Therefore, all possible permutations of values in the domains of attributes have to be considered.

Let π be a permutation of $\{1, \ldots, n\}$ and let r and s be categorical attributes such that $n_r = n_s = n$. Denote by πQ_s the result of applying permutation π to the distribution $Q_s = (q_{s,1}, \ldots, q_{s,n})$ of s, i.e., $(q_{s,\pi(1)}, \ldots, q_{s,\pi(n)})$. The similarity measure is now defined as

$$f^{\pi E}_{R,S}(r, s) = \max_{\pi \in \Pi(n)} E\left(P_r, \pi Q_s\right), \tag{5}$$

where $\Pi(n)$ is the set of all permutations of $\{1, \ldots, n\}$.

Lemma 2. *Let $f_{R,S}(r_i, s_j)$ be defined as in Eq. (5). Then for given schemas R, S and all pairs of attributes $r \in R, s \in S$, the following similarity confidence intervals satisfy Definition 3.*

$$\lfloor f^{\pi E}_{R',S'}(r, s) \rfloor_\delta = \max_{\pi \in \Pi} \lfloor f^{E}_{R',S'}(r, \pi s) \rfloor_\delta, \tag{6}$$

$$\lceil f^{\pi E}_{R',S'}(r, s) \rceil_\delta = \max_{\pi \in \Pi} \lceil f^{E}_{R',S'}(r, \pi s) \rceil_\delta, \tag{7}$$

where R', S' are samples taken from R and S respectively, $(p_{r,1}, \ldots, p_{r,n_r})$ and $(q_{s,1}, \ldots, q_{s,n_s})$ are probability distributions of r and s respectively, and $M = \sum_{r \in R} n_r + \sum_{s \in S} n_s$, over categorical attributes.

The proof of lemma 2 is given in Appendix B.

It turns out that it is possible to compute the similarity without checking all possible permutations. It is sufficient to sort P_r and Q_s in order of decreasing probabilities. To see why this produces an optimal ordering, notice that if $p_1 > p_2$ and $q_1 > q_2$, then $2 - (p_1 - q_1)^2 - (p_2 - q_2)^2 > 2 - (p_1 - q_2)^2 - (p_2 - q_1)^2$. Thus transposing q_1 and q_2 increases the similarity. The result follows by applying the transposes repeatedly to first put both largest probabilities in the same positions, then both second largest probabilites in the same positions, etc.

5.3. Bigrams

This measure is based on the distribution of bigrams (i.e., two character subsequences) in attribute values. Such an n-gram matcher is also used in [4].

Let B be a small integer (10 in our case). For an attribute r create a synthetic attribute r' computed as follows: for a given record t, pick a random bigram from the value $t[r]$ and convert it to an integer based on ASCII values of the two characters. Take this number modulo B. The obtained number is the value of r' in the record t. It is easy to see that r' is a categorical attribute with B values. Such attributes can easily be compared using $f^E_{R,S}$ giving rise to the following similarity measure

$$f^{Bg}_{R,S}(r, s) = f^{E}_{R,S}(r', s'). \tag{8}$$

Drawing a random bigram assures that r' and s' are sampled iid (independently and from identical distributions). Using the hashing scheme with B buckets reduces the domain of the random variables compared to using the set of all bigrams which, in turn, narrows the confidence bounds. Empirically, we find that matchings obtained using this method are similar those found without hashing; however, hashing allows for tight confidence intervals.

Lemma 3. *Let $f_{R,S}(r_i, s_j)$ be defined as in Eq. (8). Then for given schemas R, S, samples R', S' taken from those schemas and all pairs of attributes $r \in R, s \in S$, the following similarity confidence intervals satisfy Definition 3.*

$$\lfloor f^{Bg}_{R',S'}(r,s)\rfloor_\delta = 2 - \sum_{i=1}^{n_{r'}}\Big[(\lceil p_{r',i}\rceil_{\frac{\delta}{M}})^2 + (\lceil q_{s',i}\rceil_{\frac{\delta}{M}})^2 - 2\lfloor p_{r',i}\rfloor_{\frac{\delta}{M}}\lfloor q_{s',i}\rfloor_{\frac{\delta}{M}}\Big] \tag{9}$$

$$\lceil f^{Bg}_{R',S'}(r,s)\rceil_\delta = 2 - \sum_{i=1}^{n_{r'}}\Big[(\lfloor p_{r',i}\rfloor_{\frac{\delta}{M}})^2 + (\lfloor q_{s',i}\rfloor_{\frac{\delta}{M}})^2 - 2\lceil p_{r',i}\rceil_{\frac{\delta}{M}}\lceil q_{s',i}\rceil_{\frac{\delta}{M}}\Big] \tag{10}$$

The proof of lemma 3 is given in Appendix B.

5.4. Character proportions

This measure compares the proportions of characters that fall into predefined subsets. We use four subsets: lowercase letters, uppercase letters, digits, and an additional set for all remaining printable characters. For instance, Clio uses this idea [19]. Let r be an attribute. For each record t we compute the proportions p_l, p_u, p_d, p_p of characters of $t[r]$ belonging to each subset. After taking the averages of p_l, p_u, p_d, p_p over all records we obtain a vector $P^r = (P^r_l, P^r_u, P^r_d, P^r_p)$. Elements of this vector lie in the interval $[0, 1]$ and sum up to 1; hence, such vectors can be compared like probability distributions. The similarity measure can now be defined as

$$f^c_{R,S}(r,s) = E(P^r, P^s).$$

Bounds for $f^c_{R,S}$ are defined analogously to Lemma 3 but in terms of vectors P^r and P^s defined above.

5.5. Weighted sum

A weighted sum of all similarity measures forms the most general case. Any weighted subset of measures can be obtained by setting weights to either zero or desired nonzero values.

$$f_{R,S}(r,s) = \frac{w_{\pi E} f^{\pi E}_{R,S}(r,s) + w_{Bg} f^{Bg}_{R,S}(r,s) + w_c f^c_{R,S}(r,s)}{w_{\pi E} + w_{Bg} + w_c} \tag{11}$$

Theorem 2. *Let $f_{R,S}(r_i, s_j)$ be defined as in Eq. (11). Then for given schemas R, S, samples R', S' taken from those schemas, and all pairs of attributes $r \in R, s \in S$, the following similarity confidence intervals satisfy Definition 3.*

$$\lfloor f_{R',S'}(r,s)\rfloor_\delta = \frac{w_{\pi E}\lfloor f^{\pi E}_{R',S'}(r,s)\rfloor_\delta + w_{Bg}\lfloor f^{Bg}_{R',S'}(r,s)\rfloor_\delta + w_c\lfloor f^c_{R',S'}(r,s)\rfloor_\delta}{w_{\pi E} + w_{Bg} + w_c} \tag{12}$$

$$\lceil f_{R',S'}(r,s)\rceil_\delta = \frac{w_{\pi E}\lceil f^{\pi E}_{R',S'}(r,s)\rceil_\delta + w_{Bg}\lceil f^{Bg}_{R',S'}(r,s)\rceil_\delta + w_c\lceil f^c_{R',S'}(r,s)\rceil_\delta}{w_{\pi E} + w_{Bg} + w_c} \tag{13}$$

The proof of Theorem 2 is given in Appendix B.

5.6. Combining schema- and instance-level

When schema-level information is available – for instance in the form of attribute names or descriptions – it is advisable to use this information. The similarity function can combine schema- and instance-level components:

$$f_{R,S}(r,s) = w_S f^{\text{Schema}}(r,s) + w_I f^{Instance}_{R,S}(r,s).$$

For instance, f^{Schema} can quantify the similarity of attributes' names and descriptions. Since f^{Schema} is independent of the database, it follows immediately that the bounds are

$$\begin{aligned}\lfloor f_{R',S'}(r,s)\rfloor &= w_S f^{Schema} + w_I \lfloor f^{Instance}_{R',S'}(r,s)\rfloor, \text{ and} \lceil f_{R',S'}(r,s)\rceil \\ &= w_S f^{Schema} + w_I \lceil f^{Instance}_{R',S'}(r,s)\rceil.\end{aligned}$$

6. Experiments

We want to investigate empirically (a) whether the FSM and progressive FSM algorithms are practically applicable for large databases. We want to (b) compare the performances of FSM (using Hoeffding's bounds and normal bounds), progressive FSM (using normal bounds), and a baseline instance-based matcher that uses the same similarity function but processes the entire database. The baseline is not practical for streams and large databases, it always retrieves the matches that maximize the similarity on the database. Theorem 1 *guarantees* that FSM and progressive FSM return approximately optimal matches; nevertheless, we will (c) empirically study the chance of the algorithms finding the correct matches for attributes. The similarity function used for all experiments is the weighted average of all similarity functions studied in Section 5, with all weights fixed to 1.

6.1. Benchmark problems

In order to assure the reproducibility of our results, we use three publicly available databases. The KDDCup 1998 customer relationship management database contains 481 attributes and over 190,000 records; the majority of attributes are numerical but some categorical attributes are included. The KDDCup 1999 database is another standard benchmark to be audited; it includes a wide variety of intrusions simulated in a military network environment. The database has in total 42 attributes, 17 of which are almost always zero; another 14 almost always assume the value "0.00". This database contains nearly 5 million records. The census database is an extract from the 1994 and 1995 current population surveys conducted by the U.S. Census Bureau. The database has 42 attributes in total, the majority of them contain text. There are close to 300,000 records. The execution time of both FSM and progressive FSM is constant in the desired number of solutions k (except for the final step of printing the output); the experimental results are identical for any possible value of k.

In order to conduct controlled experiments, we randomly split each of the databases into two parts containing half of the records – for the KDDCup 1998 database, we use the split that is provided with the data. We then use the schema matching algorithms to match the two halves of the databases. When the algorithm matches an attribute with itself, this is counted as a true positive. Based on the number of attributes that are matched with itself, we determine precision and recall, and F-measure.

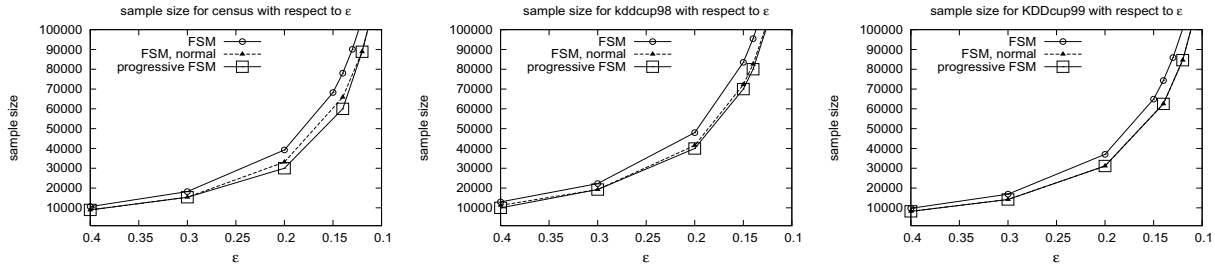


Fig. 1. Sample size against ε for schema matching algorithms.

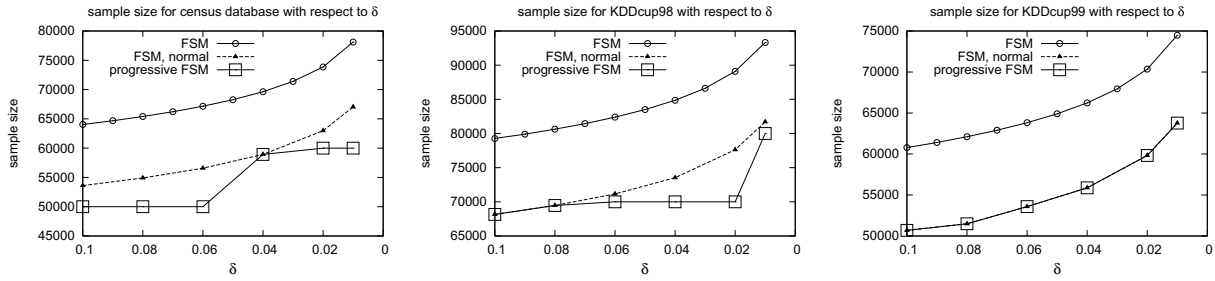


Fig. 2. Sample size against δ for schema matching algorithms.

Figure 1 shows the number of database records that FSM and progressive FSM draw from the database and process before finding an approximately optimal match (using $\delta = 0.05$). For the census and KDD Cup 1998 data, the early stopping criterion (Step 4) of progressive FSM occasionally kicks in, reducing the number of samples on average compared to FSM. For the KDD Cup 1999 database, early stopping in Step 4 is never exercised. The reason is that due to the sparseness of the data the matching problem is very hard. Many attributes assume zeros most of the time which makes many potential matching partners appear equally good. For all problems, we observe that the normal bounds reduce the required sample size over the Hoeffding bounds.

Figure 2 compares the sample size required by FSM and progressive FSM for varying values of δ (using $\varepsilon = 0.15$). For small values of δ, we can observe the benefit of the progressive version of the algorithm more clearly. In particular, while the sample-independent bounds of FSM and FSM with normal bounds follow a smooth curve, we observe that the point of termination of progressive FSM depends on the data and progresses in steps of 10,000; progressive FSM can terminate significantly earlier, depending on the data and on the value chosen for δ.

Figure 3 compares the execution time of FSM, progressive FSM, and the baseline algorithm that executes a pass over the entire database for varying values of ε (using $\delta = 0.1$). Again, progressive FSM is the fastest method, followed by FSM. The baseline algorithm that calculates the similarities based on the entire database is substantially slower, depending on the database size. For the KDD Cup 1999 database, we are unable to obtain results for the baseline algorithm which here exceeds our patience.

Finally, Fig. 4 compares the F-measure of FSM, progressive FSM and the baseline algorithm that processes the entire database. For the census and KDD Cup 1998 databases, the F-measures are above 0.9 and 0.8, respectively, even for $\varepsilon = 1$! For decreasing values of ε, the F-measure grows further. The KDD Cup 1999 database, by contrast, is very difficult. Since many attributes almost always assume a value of zero, most similarity values are very close and it is difficult to identify the best match; the F-measure is lower, accordingly.

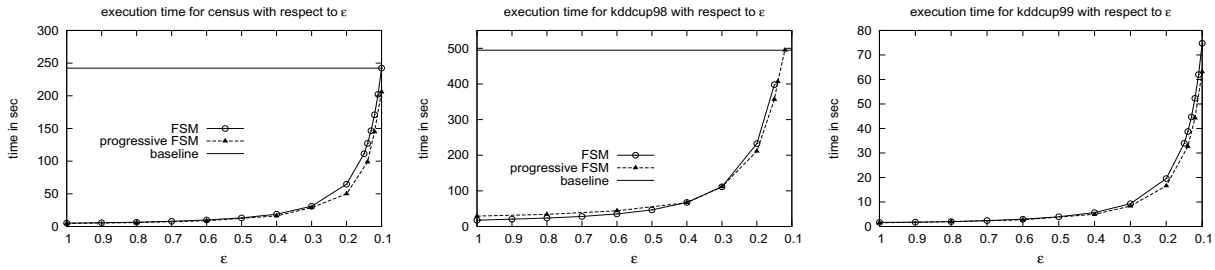


Fig. 3. Execution time against ε for schema matching algorithms.

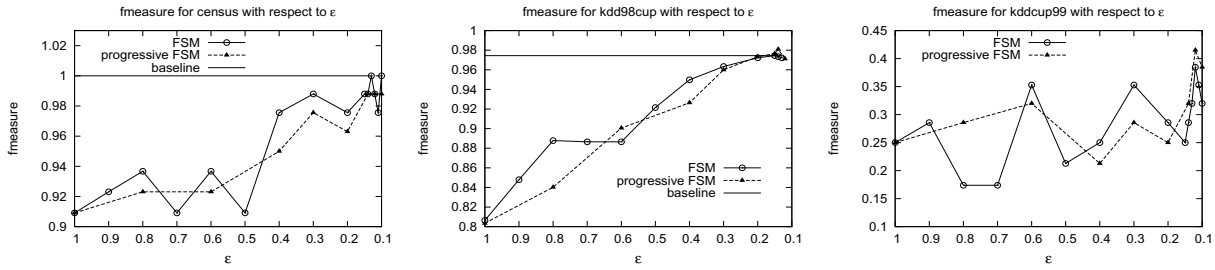


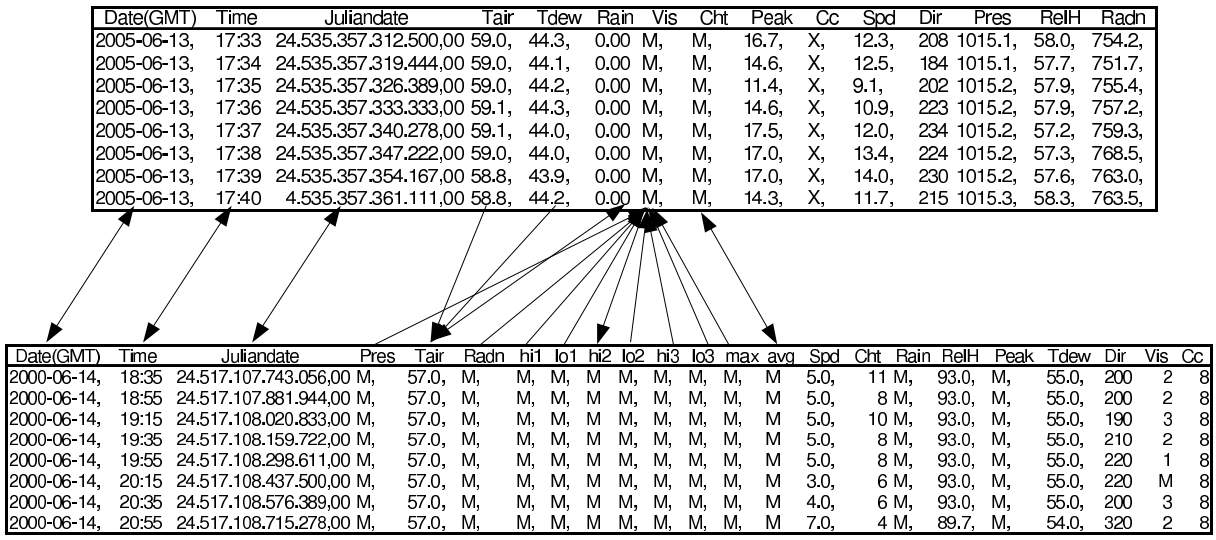
Fig. 4. F-measure against ε for schema matching algorithms.

Note that the low F-measure for the KDD Cup 1999 database is not in contradiction to the guarantee provided by FSM: The F-measure quantifies how frequently attributes are matched to their actually semantically equivalent partner (in our experimental setting, to themselves). By contrast, Theorem 1 guarantees that the retrieved match for each attribute is "nearly as good as the optimal match". The baseline algorithm processes the entire database in the first place and therefore obtains a constantly high F- measure for the census and KDD Cup 1998 databases. For the KDD Cup 1999 problem, the baseline algorithm exceeds reasonable execution time, no result can be provided. For the FSM and progressive FSM algorithms, execution time and sample size as well as the quality of the solution depend on the parameters ε and δ rather than on the size of the database.

6.2. Weather data case study

In this set of experiments, we explore the behavior of the progressive FSM algorithm on schemas of distinct weather databases that share only a common subset of attributes. Semantically identical attributes of the two distinct databases vary greatly, and the data contain many missing values which makes this problem a challenging benchmark. We choose a snapshot of 275,000 records in 2005 from the University of Washington and Arlington Airport weather streams. Excerpts are visualized in Figure 5. We use the progressive FSM with normal bounds. For $\delta = 0.1$ and $\epsilon = 0.2$, progressive FSM terminates after seeing $N = 20,000$ records; for $\delta = 0.1$ and $\epsilon = 0.15$ after 40,000 transactions. The retrieved match does not improve after processing the entire 275,000 records. Therefore, the early stopping does not come at the price of a poorer match.

Figure 5 shows the attributes of both schemas sorted according to the ordering determined by progressive FSM; i.e., the algorithm considers the leftmost attributes most likely (the rightmost attributes least likely) to have corresponding attributes in the other database. The arrows indicate the single best



Date(GMT)	Time	Juliandate	Tair	Tdew	Rain	Vis	Cht	Peak	Cc	Spd	Dir	Pres	RelH	Radn
2005-06-13,	17:33	24.535.357.312.500,00	59.0,	44.3,	0.00	M,	M,	16.7,	X,	12.3,	208	1015.1,	58.0,	754.2,
2005-06-13,	17:34	24.535.357.319.444,00	59.0,	44.1,	0.00	M,	M,	14.6,	X,	12.5,	184	1015.1,	57.7,	751.7,
2005-06-13,	17:35	24.535.357.326.389,00	59.0,	44.2,	0.00	M,	M,	11.4,	X,	9.1,	202	1015.2,	57.9,	755.4,
2005-06-13,	17:36	24.535.357.333.333,00	59.1,	44.3,	0.00	M,	M,	14.6,	X,	10.9,	223	1015.2,	57.9,	757.2,
2005-06-13,	17:37	24.535.357.340.278,00	59.1,	44.0,	0.00	M,	M,	17.5,	X,	12.0,	234	1015.2,	57.2,	759.3,
2005-06-13,	17:38	24.535.357.347.222,00	59.0,	44.0,	0.00	M,	M,	17.0,	X,	13.4,	224	1015.2,	57.3,	768.5,
2005-06-13,	17:39	24.535.357.354.167,00	58.8,	43.9,	0.00	M,	M,	17.0,	X,	14.0,	230	1015.2,	57.6,	763.0,
2005-06-13,	17:40	4.535.357.361.111,00	58.8,	44.2,	0.00	M,	M,	14.3,	X,	11.7,	215	1015.3,	58.3,	763.5,

Date(GMT)	Time	Juliandate	Pres	Tair	Radn	hi1	lo1	hi2	lo2	hi3	lo3	max	avg	Spd	Cht	Rain	RelH	Peak	Tdew	Dir	Vis	Cc
2000-06-14,	18:35	24.517.107.743.056,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	5.0,	11	M,	93.0,	M,	55.0,	200	2	8
2000-06-14,	18:55	24.517.107.881.944,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	5.0,	8	M,	93.0,	M,	55.0,	200	2	8
2000-06-14,	19:15	24.517.108.020.833,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	5.0,	10	M,	93.0,	M,	55.0,	190	3	8
2000-06-14,	19:35	24.517.108.159.722,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	5.0,	8	M,	93.0,	M,	55.0,	210	2	8
2000-06-14,	19:55	24.517.108.298.611,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	5.0,	8	M,	93.0,	M,	55.0,	220	1	8
2000-06-14,	20:15	24.517.108.437.500,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	3.0,	6	M,	93.0,	M,	55.0,	220	M	8
2000-06-14,	20:35	24.517.108.576.389,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	4.0,	6	M,	93.0,	M,	55.0,	200	3	8
2000-06-14,	20:55	24.517.108.715.278,00	M,	57.0,	M,	M,	M,	M	M,	M,	M,	M,	M	7.0,	4	M,	89.7,	M,	54.0,	320	2	8

Fig. 5. Uni Washington (top) matched against Arlington Airport (bottom) weather stream.

match for each attribute; to avoid cluttering the diagram, we draw only arrows between attributes with a particularly high similarity value ($\geqslant 2$). The rightmost attributes therefore have no displayed arrows.

The algorithm finds exact matches for time, date, Julian date, and T_{air}. These attributes can be matched based on character bigrams that occur in their values Attribute T_{dew} is erroneously matched to T_{air}. Attribute Cht (all missing values) is matched to avg (also all missing values). The Arlington stream contains attributes with almost only missing values that are not present in the Washington stream; they are matched to vis (also almost only missing values). It is inevitable that an instance-based schema matcher will map attributes with all missing values to other attributes with all missing values, when no additional schema-level clues are available. For T_{dew}, the diverging ranges of values prevent the involved similarity metrics from identifying these attributes as equivalent.

These results underscore that (a) the sampling-based approximation and stopping criterion are feasible, scalable, and do not deteriorate the quality of the results compared to processing all data. (b) The results emphasize that instance-information can in many cases be used to match equivalent attributes; however, the similarity metric has to be designed to be appropriate for the problem at hand, and attributes with all missing values and attributes whose values diverge too strongly between databases limit the applicability of instance-level metrics.

7. Conclusion

Our formulation of the approximately optimal schema matching problem is closely tied to practical applications and at the same time mathematically rigorous. The FSM and progressive FSM algorithm provably solve this problem; that is, for each attribute of either schema, they find the k approximately best matching partners, and approximately order them according to their similarity. Finally, the attributes in either schema are approximately sorted according to their likelihood of having a partner in the other schema. The term "approximately" here means "up to user-specified approximation parameters ε and δ".

The required sample size and execution time of the algorithms depends on these parameters, but are independent of the amount of data available; therefore, the algorithms apply to infinite streams. For

the progressive FSM algorithm, sample size and execution time depend on the stream at hand. If there are some similar but many dissimilar potential matches, then progressive FSM can identify the similar matches faster and terminate early.

Our experiments lead to a number of conclusions. (a) FSM and progressive FSM are feasible and practically applicable for streams and very large databases. They can be applied to a range of similarity metrics; we specified bounds for a selection of measures that can easily be extended. (b) FSM with normal bounds and progressive FSM are equally fast for difficult matching problems with many very similar attributes. Progressive FSM is faster than FSM otherwise. The normal bounds outperform the Hoeffding bounds. (c) While theoretical results guarantee an approximately optimal match, we observe empirically that attributes are often actually matched to semantically equivalent attributes. For the census and KDD Cup 1998 databases, high F-measures are observed even for large values of ε.

Acknowledgment

Tobias Scheffer is supported by the German Science Foundation DFG under Grant SCHE540/10-2.

The authors would like to thank the Reviewers for the thorough reviews and helpful remarks, especially for pointing out the quick way of maximizing the E measure over all permutations.

Appendix A. Proof of Theorem 1

We will prove Theorem 1 separately for FSM and progressive FSM.

Theorem 1 for FSM

From Definition 3 (definition of a similarity confidence interval) it follows that, after Step 1 has been executed, with probability $1-\delta$ there is no pair of attributes r_i in R and s_j in S such that $\lfloor f_{R',S'}(r_i,s_j)\rfloor_\delta \leqslant f_{R,S}(r_i,s_j) \leqslant \lceil f_{R',S'}(r_i,s_j)\rceil_\delta$. We will now show that the nonexistence of such an offending pair (r_i,s_j) implies that the returned solution satisfies the three conditions of Definition 2. Since the risk of existence of an offending pair is bounded by δ, this suffices to prove the theorem.

Before proving the three properties that characterize an approximately optimal match, let us show a useful property of the algorithm. Let (r,s) and (r',s') be two pairs of attributes whose similarity confidence bounds have been determined in Step 3 of the algorithm. Then,

$$\hat{f}_{R',S'}(r,s) \geqslant \hat{f}_{R',S'}(r',s') \Rightarrow f_{R,S}(r,s) \geqslant f_{R,S}(r',s') - \varepsilon \tag{14}$$

Implication (14) is proven as follows. Equation (15) follows from the definition of $\hat{f}_{R',S'}$ in Step 3 of the algorithm. This implies Eq. (16) because, in Step 1, N is chosen such that for all pairs $\lceil f_{R',S'}(\cdot,\cdot)\rceil - \lfloor f_{R',S'}(\cdot,\cdot)\rfloor \leqslant \varepsilon$. In addition, from Definition 3 and the first sentence of this proof we know that, for any pair, $\lceil f_{R',S'}(\cdot,\cdot)\rceil \geqslant f_{R,S}(\cdot,\cdot) \geqslant \lfloor f_{R',S'}(\cdot,\cdot)\rfloor$.

$$\hat{f}_{R',S'}(r,s) \geqslant \hat{f}_{R',S'}(r's') \Leftrightarrow \frac{1}{2}(\lceil f_{R',S'}(r,s)\rceil + \lfloor f_{R',S'}(r,s)\rfloor)$$

$$\geqslant \frac{1}{2}(\lceil f_{R',S'}(r',s')\rceil + \lfloor f_{R',S'}(r',s')\rfloor) \tag{15}$$

$$\Rightarrow f_{R,S}(r,s) + \frac{\varepsilon}{2} \geqslant f_{R,S}(r',s') - \frac{\varepsilon}{2} \tag{16}$$

$$\Leftrightarrow f_{R,S}(r,s) \geqslant f_{R,S}(r',s') - \varepsilon \tag{17}$$

1. In step 4, the algorithm selects $H^*_{r_i}$ such that $\hat{f}_{R',S'}(r_i, s_j)$ is maximized. That is, for every element s_j inside $H^*_{r_i}$, including $s_{i_{min}} = \text{argmin}_{s \in H^*_{r_i}} \hat{f}_{R',S'}(r_i, s)$ and every element s' outside $H^*_{r_i}$, we have that $\hat{f}_{R',S'}(r_i, s_{i_{min}}) \geqslant \hat{f}_{R',S'}(r_i, s')$. Then Eq. (14) implies $f_{R,S}(r_i, s_{i_{min}}) \geqslant f_{R,S}(r_i, s') - \varepsilon$, and so the first condition of an approximately optimal match is satisfied.
2. For all attributes r_i, the retrieved sequence $(s_{i_1}, \ldots, s_{i_k})$ is sorted according to $\hat{f}_{R',S'}(r_i, s_{i'})$. Therefore, if some element $s_{i'}$ occurs prior to $s_{i''}$ it must be true that $\hat{f}_{R',S'}(r_i, s_{i'}) \geqslant \hat{f}_{R',S'}(r_i, s_{i''})$. Equation (14) then implies that $f_{R,S}(r_i, s_{i'}) \geqslant f_{R,S}(r_i, s_{i''}) - \varepsilon$ which satisfies the definition of an ε-correct ordering (Definition 1). The same argument applies to the ordering of $(r_{j_1}, \ldots, r_{j_k})$.
3. In Step 5, the permutation π_r sorts the attributes in R according to $\max_{s_{i'} \in H^*_{r_i}} \hat{f}_{R',S'}(r_i, s_{i'})$. When π_r sorts some attribute r prior to r', then $\max_{s^*_i \in H^*_r} \hat{f}_{R',S'}(r, s_{i'}) \geqslant \max_{s_{i'} \in H^*_{r'}} \hat{f}_{R',S'}(r', s_{i'})$. Equation (14) now implies $\max_{s_{i'} \in H^*_r} f_{R,S}(r, s_{i'}) \geqslant \max_{s_{i'} \in H^*_{r'}} f_{R,S}(r', s_{i'}) - \varepsilon$; that is, $(\max_{j'} f_{R,S}(r_{\pi^r(1)}, s_{j'}), \ldots, \max_{j'} f_{R,S}(r_{\pi^r(n)}, s_{j'}))$ is an ε-correct ordering. The same argument applies to the ordering $(\max_{i'} f_{R,S}(r_{i'}, s_{\pi^s(1)}), \ldots, \max_{i'} f_{R,S}(r_{i'}, s_{\pi^s(m)}))$.

This proves Theorem 1. □

Theorem 1 for Progressive FSM

The main loop of the algorithm in Table 2 can terminate for two reasons. Either N examples have been processed and from the calculation of N and the definition of the similarity confidence interval (Definition 3) we know that in this case, with probability 1-$\frac{\delta}{2}$, there is no pair (r_i, s_j) that violates $\lfloor f_{R',S'}(r_i, s_j) \rfloor_{\frac{\delta}{2}} \leqslant f_{R,S}(r_i, s_j) \leqslant \lceil f_{R',S'}(r_i, s_j) \rceil_{\frac{\delta}{2}}$. Alternatively, the algorithm can terminate in step 4 when all pairs (r_i, s_j) satisfy $\lceil f_{R',S'}(r_i, s_j) \rceil_{\delta_t} - \lfloor f_{R',S'}(r_i, s_j) \rfloor_{\delta_t} \leqslant \varepsilon$. In this case, Definition 3 says that with probability $1 - \delta_t$, there is no pair (r_i, s_j) that violates $\lfloor f_{R',S'}(r_i, s_j) \rfloor_{\delta_t} \leqslant f_{R,S}(r_i, s_j) \leqslant \lceil f_{R',S'}(r_i, s_j) \rceil_{\delta_t}$. After terminating in either of these steps, the algorithm returns results analogously to the algorithm in Table 1.

Assuming that all similarities lie within their corresponding similarity confidence intervals, Eq. (14) is satisfied again and the proof in Section 7 shows that the returned solution satisfies the three conditions of an approximately optimal matching characterized in Definition 2. However, we have assumed that all the similarity values throughout the execution of the algorithm lie within their corresponding similarity confidence interval. We now have to bound the risk that during the course of execution of the algorithm there is any pair (r_i, s_j) whose true similarity lies outside its similarity confidence interval.

When N examples have been processed, then the risk that any pair (r_i, s_j) violates

$$\lfloor f_{R',S'}(r_i, s_j) \rfloor_{\frac{\delta}{2}} \leqslant f_{R,S}(r_i, s_j) \leqslant \lceil f_{R',S'}(r_i, s_j) \rceil_{\frac{\delta}{2}} \tag{18}$$

is at most $\frac{\delta}{2}$; this follows immediately from Definition 2. Furthermore, during the t-th iteration of the main loop the algorithm incurs an additional risk of δ_t that there is at least one pair whose similarity lies outside its confidence interval. We use the union bound to assess this risk as follows.

$$\begin{aligned}
&\text{Pr}[\text{During execution, there is a } (r_i, s_j) \text{ that violates } (18)] \\
&\leqslant \frac{\delta}{2} + Pr[\text{Step 4, iteration } t \text{ some } (r_i, s_j) \text{ violates } (18)] \\
&\leqslant \frac{\delta}{2} + \sum_{t=1}^{\infty} \delta_t = \frac{\delta}{2} + \sum_{t=1}^{\infty} \frac{\delta}{2t(t+1)}
\end{aligned} \tag{19}$$

$$\leqslant \frac{\delta}{2} + \frac{\delta}{2}\sum_{t=1}^{\infty}\frac{1}{t(t+1)} = \delta \tag{20}$$

This completes the proof. □

Appendix B. Proof of correctness of confidence intervals

We will now prove Theorem 2; during the course of the proof, we will also prove Lemmas 1 and 2. Recall that $f_{R,S}$ is a weighted sum of three other measures. The following Lemma makes it possible to derive intervals for $f_{R,S}$ from the intervals for its three constituents.

Lemma 4. *If $[\lfloor x_i \rfloor_\delta, \lceil x_i \rceil_\delta]$ for $i = 1, \ldots, m$ are simultaneous confidence intervals for $x_1, \ldots, x_m$ respectively, with significance level δ, and $w_i \geqslant 0$ for $i = 1, \ldots, m$, then*

$$\lfloor w_1 x_1 + \ldots + w_m x_m \rfloor_\delta = \sum_{i=1}^{m} w_i \lfloor x_i \rfloor_\delta \tag{21}$$

$$\lceil w_1 x_1 + \ldots + w_m x_m \rceil_\delta = \sum_{i=1}^{m} w_i \lceil x_i \rceil_\delta \tag{22}$$

is a confidence interval for $w_1 x_1 + \ldots + w_m x_m$ with significance level δ.

Proof.

$$\Pr\left\{w_1 x_1 + \ldots + w_m x_m \notin \left[\sum_{i=1}^{m} w_i \lfloor x_i \rfloor_\delta, \sum_{i=1}^{m} w_i \lceil x_i \rceil_\delta\right]\right\} \leqslant \Pr\{\exists i : w_i x_i \notin [w_i \lfloor x_i \rfloor_\delta, w_i \lceil x_i \rceil_\delta]\}$$
$$= \Pr\{\exists i : x_i \notin [\lfloor x_i \rfloor_\delta, \lceil x_i \rceil_\delta]\} \leqslant \delta. \qquad \square$$

It is easy to see from the definitions in Section 5 that we only need to prove confidence intervals for $f^E_{R,S}$ and $f^{\pi E}_{R,S}$. We will therefore now prove Lemmas 1 and 2.

Proof. Using basic algebraic transformations it is easy to see that

$$\left(\forall r \in R, \forall i \in \{1 \ldots n_r\} : p_{r,i} \in [\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}}, \lceil p_{r,i} \rceil_{\frac{\delta}{M}}]\right)$$
$$\text{and } \left(\forall s \in S, \forall j \in \{1 \ldots n_s\} : q_{s,j} \in [\lfloor q_{s,j} \rfloor_{\frac{\delta}{M}}, \lceil q_{s,j} \rceil_{\frac{\delta}{M}}]\right)$$

implies

$$\forall r \in R, \forall i \in \{1 \ldots n_r\}, \forall \{s \in S : n_s = n_r\}, \forall j \in \{1 \ldots n_s\} : (p_{r,i} - q_{s,j})^2 \in$$
$$\left[(\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}})^2 + (\lfloor q_{s,j} \rfloor_{\frac{\delta}{M}})^2 - 2\lceil p_{r,i} \rceil_{\frac{\delta}{M}} \lceil q_{s,j} \rceil_{\frac{\delta}{M}}, (\lceil p_{r,i} \rceil_{\frac{\delta}{M}})^2 + (\lceil q_{s,j} \rceil_{\frac{\delta}{M}})^2 - 2\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}} \lfloor q_{s,j} \rfloor_{\frac{\delta}{M}}\right],$$

and after replacing j with an i-th index of a permutation of $\{1, \ldots, n\}$:

$$\forall r \in R, \forall \{s \in S : n_s = n_r\}, \forall i \in \{1 \ldots n_r\}, \forall \pi \in \Pi(n_r) : (p_{r,i} - q_{s,\pi_{(i)}})^2 \in$$
$$\left[(\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}})^2 + (\lfloor q_{s,\pi_{(i)}} \rfloor_{\frac{\delta}{M}})^2 - 2\lceil p_{r,i} \rceil_{\frac{\delta}{M}} \lceil q_{s,\pi_{(i)}} \rceil_{\frac{\delta}{M}}, (\lceil p_{r,i} \rceil_{\frac{\delta}{M}})^2\right.$$
$$\left. + (\lceil q_{s,\pi_{(i)}} \rceil_{\frac{\delta}{M}})^2 - 2\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}} \lfloor q_{s,\pi_{(i)}} \rfloor_{\frac{\delta}{M}}\right].$$

Notice that we are just permuting the confidence intervals for single probability estimates, and not introducing any new ones. This allows us to test all permutations without increasing M.

After summing over all $i = 1, \ldots, n_r$ we get

$$\forall r \in R, \forall \{s \in S : n_s = n_r\}, \forall \pi \in \Pi(n_r) : \sum_{i=1}^{n_r} (p_{r,i} - q_{s,\pi_{(i)}})^2 \in$$

$$\left[\sum_{i=1}^{n_r} \left((\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}})^2 + (\lfloor q_{s,\pi_{(i)}} \rfloor_{\frac{\delta}{M}})^2 - 2 \lceil p_{r,i} \rceil_{\frac{\delta}{M}} \lceil q_{s,\pi_{(i)}} \rceil_{\frac{\delta}{M}} \right), \right.$$

$$\left. \sum_{i=1}^{n_r} \left((\lceil p_{r,i} \rceil_{\frac{\delta}{M}})^2 + (\lceil q_{s,\pi_{(i)}} \rceil_{\frac{\delta}{M}})^2 - 2 \lfloor p_{r,i} \rfloor_{\frac{\delta}{M}} \lfloor q_{s,\pi_{(i)}} \rfloor_{\frac{\delta}{M}} \right) \right].$$

Subtracting from 2 we get l

$$\forall r \in R, \forall \{s \in S : n_s = n_r\}, \forall \pi \in \Pi(n_r) : 2 - \sum_{i=1}^{n_r} (p_{r,i} - q_{s,\pi_{(i)}})^2 \in$$

$$\left[2 - \sum_{i=1}^{n_r} \left((\lceil p_{r,i} \rceil_{\frac{\delta}{M}})^2 + (\lceil q_{s,\pi_{(i)}} \rceil_{\frac{\delta}{M}})^2 - 2 \lfloor p_{r,i} \rfloor_{\frac{\delta}{M}} \lfloor q_{s,\pi_{(i)}} \rfloor_{\frac{\delta}{M}} \right), \right.$$

$$\left. 2 - \sum_{i=1}^{n_r} \left((\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}})^2 + (\lfloor q_{s,\pi_{(i)}} \rfloor_{\frac{\delta}{M}})^2 - 2 \lceil p_{r,i} \rceil_{\frac{\delta}{M}} \lceil q_{s,\pi_{(i)}} \rceil_{\frac{\delta}{M}} \right) \right].$$

After substituting confidence interval for $f^E_{R,S}$

$$\forall r \in R, \forall \{s \in S : n_s = n_r\}, \forall \pi \in \Pi(n_r) : f^E_{R,S}(r, \pi s) \in [\lfloor f^E_{R',S'}(r, \pi s) \rfloor_\delta, \lceil f^E_{R',S'}(r, \pi s) \rceil_\delta],$$

and finally

$$\forall r \in R, \forall \{s \in S : n_s = n_r\} : \max_{\pi \in \Pi(n_r)} f^E_{R,S}(r, \pi s) \in [\max_{\pi \in \Pi(n_r)} \lfloor f^E_{R',S'}(r, \pi s) \rfloor_\delta,$$

$$\max_{\pi \in \Pi(n_r)} \lceil f^E_{R',S'}(r, \pi s) \rceil_\delta].$$

Below '$\Rightarrow$' denotes logical implication. By a similar (and simpler) argument we can derive

$$(\forall r \in R, \forall i \in \{1 \ldots n_r\} : p_{r,i} \in [\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}}, \lceil p_{r,i} \rceil_{\frac{\delta}{M}}])$$

$$\text{and } (\forall s \in S, \forall j \in \{1 \ldots n_s\} : q_{s,j} \in [\lfloor q_{s,j} \rfloor_{\frac{\delta}{M}}, \lceil q_{s,j} \rceil_{\frac{\delta}{M}}])$$

$$\Rightarrow \forall r \in R, \forall \{s \in S : n_s = n_r\} : f^E_{R,S}(r, s) \in [\lfloor f^E_{R',S'}(r, s) \rfloor_\delta, \lceil f^E_{R',S'}(r, s) \rceil_\delta].$$

Combining the two above implications and using modus tollens we get

$$\exists r \in R, \exists s \in S : n_s = n_r, f^E_{R',S'}(r, s) \notin [\lfloor f^E_{R',S'}(r, s) \rfloor_\delta, \lceil f^E_{R',S'}(r, s) \rceil_\delta]$$

$$\vee \exists r \in R, \exists s \in S : n_s = n_r, f^{\pi E}_{R,S}(r, s) \notin [\lfloor f^{\pi E}_{R',S'}(r, \pi s) \rfloor_\delta, \lceil f^{\pi E}_{R',S'}(r, \pi s) \rceil_\delta] \quad (23)$$

$$\Rightarrow \exists r \in R, i \in \{1 \ldots n_r\} : p_{r,i} \notin [\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}}, \lceil p_{r,i} \rceil_{\frac{\delta}{M}}] \vee \exists s \in S, j \in \{1 \ldots n_s\}$$
$$: q_{s,i} \notin [\lfloor q_{s,j} \rfloor_{\frac{\delta}{M}}, \lceil q_{s,j} \rceil_{\frac{\delta}{M}}].$$

Let A denote an event that one of the similarity measures considered is outside of its confidence interval, i.e. Eq. (23). Since if for two events A and B we have: if $A \Rightarrow B$ then $\Pr\{A\} \leqslant \Pr\{B\}$, it follows that

$$\begin{aligned}
\Pr\{A\} &\leqslant \Pr\Big\{\exists r \in R, i \in \{1 \ldots n_r\} : p_{r,i} \notin [\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}}, \lceil p_{r,i} \rceil_{\frac{\delta}{M}}] \vee \exists s \in S, j \in \{1 \ldots n_s\} : \\
&\qquad q_{s,j} \notin [\lfloor q_{s,j} \rfloor_{\frac{\delta}{M}}, \lceil q_{s,j} \rceil_{\frac{\delta}{M}}]\Big\} \\
&\leqslant \sum_{r \in R} \sum_{i=1}^{n_r} \Pr\big\{p_{r,i} \notin [\lfloor p_{r,i} \rfloor_{\frac{\delta}{M}}, \lceil p_{r,i} \rceil_{\frac{\delta}{M}}]\big\} + \sum_{s \in S} \sum_{j=1}^{n_s} \Pr\big\{q_{s,j} \notin [\lfloor q_{s,j} \rfloor_{\frac{\delta}{M}}, \lceil q_{s,j} \rceil_{\frac{\delta}{M}}]\big\} \\
&\leqslant \sum_{r \in R} \sum_{i=1}^{n_r} \frac{\delta}{M} + \sum_{s \in S} \sum_{j=1}^{n_s} \frac{\delta}{M} = \delta. \quad \square
\end{aligned}$$

Proof of Theorem 2. Bounds for single probability estimates, together with Lemmas 2 and 3 prove bounds for $f^{\pi E}_{R,S}$ and $f^{Bg}_{R,S}$. For the case of $f^{c}_{R,S}(r,s)$, note that the elements of vectors P^r and P^s are averages of numbers from the interval [0,1], so they are, in fact, random variables on that interval, and Hoeffding and normal bounds (through Central Limit Theorem) apply to them. Now, Lemma 1 applies giving confidence intervals for $f^{c}_{R,S}$. Applying Lemma 4 to confidence intervals for $f^{\pi E}_{R,S}$, $f^{Bg}_{R,S}$ and $f^{c}_{R,S}$ completes the proof. □

References

[1] J. Berlin and M. Motro, Autoplex: automated discovery of content for virtual databases. In *Proceedings of the International Conference on Cooperative Information Systems*, 2001.

[2] P. Bernstein and E. Rahm, Data warehouse scenarios for model management. In *Proceedings of the International Conference on Entity Relationship Modeling*, 2000.

[3] R. Dhamankar, Y. Lee, A. Doan, A. Halevy and P. Domingos, imap: Discovering complex semantic matches between database schemas. In *Proceedings of the ACM SIGMOD Conference*, 2004.

[4] H.-H. Do and E. Rahm, Coma – a system for flexible combination of schema matching approaches. In *Proceedings of the International Conference on Very Large Databases*, 2002.

[5] A. Doan, P. Domingos and A. Halevy, Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the ACM SIGMOD Conference*, 2001.

[6] A. Doan, P. Domingos and A. Levy, Learning source descriptions for data integration. In *Proceedings of te WebDB Workshop*, 2000.

[7] C. Domingo, R. Gavalda and O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, *Data Mining and Knowledge Discovery* **6**(2) (2002), 131–152.

[8] S. Jaroszewicz and T. Scheffer, Fast discovery of unexpected patterns in data, relative to a bayesian network. In *Proceedings of the ACM SIGKDD Conference*, 2005.

[9] J. Kang and J. Naughton, On schema matching with opaque column names and data values. In *Proceedings of the ACM SIGMOD Conference*, 2003.

[10] W. Li and C. Clifton, Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the International Conference on Very Large Databases*, 1994.

[11] W. Li and C. Clifton, Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural network, *Data and Knowledge Engineering* **33**(1) (2000), 49–84.

[12] J. Madhavan, P. Bernstein and E. Rahm, Generic schema matching with Cupid. In *Proceedings of the International Conference on Very Large Databases*, 2001.

[13] R. Miller, Y. Ioannides and R. Ramakrishnan, Schema equivalence in heterogeneous systems: bridging theory and practice, *Information Systems* **19**(1) (1994), 3–31.
[14] F. Naumann, C. Ho, X. Tian, L. Haas and N. Megiddo, Attribute classification using feature analysis. In *Proceedings of the International Conference on Data Engineering*, 2002, 271.
[15] L. Palopoli, D. Sacca and D. Ursino, An automatic technique for detecting type conflicts in database schemas. In *Proceedings of the International Conference On Information and Knowledge Management*, 1998.
[16] E. Rahm and P. Bernstein, A survey of approaches to automatic schema mapping, *VLDB Journal* **10** (2001), 334–350.
[17] M. Sayyadian, Y. Lee, A. Doan and A. Rosenthal, Tuning schema matching software using synthetic scenarios. In *Proceedings of the VLDB Conference*, 2005.
[18] T. Scheffer and S. Wrobel, Finding the most interesting patterns in a database quickly by using sequential sampling, *Journal of Machine Learning Research* **3** (2002), 833–862.
[19] L. Yan, R. Miller, L. Haas and R. Fagin, Data driven understanding and refinement of schema mappings. In *Proceedings of the ACM SIGMOD Conference*, 2001.