Accurate Schema Matching on Streams*

Szymon Jaroszewicz[1], Lenka Ivantysynova[2], Tobias Scheffer[2]

[1]National Institute of Telecommunications, Warsaw, Poland, sj@cs.umb.edu
[2]Humboldt-Universität zu Berlin, Germany
{ivantysy,scheffer}@informatik.hu-berlin.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 match the most similar attributes with high probability. 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

In many practical situations, data mining processes are preceded by a preprocessing step in which multiple heterogeneous transaction databases are integrated into a single data warehouse. Integration requires the identification of semantic correspondences between attributes across multiple transaction database schemas [2]. In practical applications that often involve large databases, schema matching is a tedioustask, often further hindered by insufficient documentation.

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 utilize 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 attributes' 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

* Proceedings of the ECML/PKDD 2006 Workshop on Knowledge Discovery from Data Streams

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. The ad-hoc solution of processing only a small sample of transactions results in a loss of any guarantee on the optimality of the produced match. Known methods that utilize 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 attribute similarity.

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 setting in Section 3. We phrase the schema matching problem in a way that pays tribute to typical practical application scenarios, and that is sufficiently rigid to allow us to state and rigorously prove the correctness of our solutions. We detail our solution to the problem 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 [12] provide an overview of the schema matching problem 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.,* [11, 6, 4]) whereas the similarity metrics employed by instance-level approaches [9] 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 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 attribute in Yen [3].

The schema matching problem also plays a role in the exchange of messages in ecommerce applications [1], or the integration of databases due to merges or other changes of organizational structures. Most practical systems – such as Clio [15], SemInt [10], 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 [13].

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 utilized to find the approximately most interesting association rules [7, 14], 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*. Given are two schemas R and S over attributes $\langle r_1, \ldots, r_n \rangle$ and $\langle s_1, \ldots, s_m \rangle$, respectively. The goal is to identify semantically identical pairs of attributes (r_i, s_j).

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 necessary to exercise a pass over the databases R and S; the main challenge that we address is that this is computationally infeasible for large databases and impossible for streams. The similarity criterion $f_{R,S}(r_i, s_j)$ is problem-specific and given in advance. We will only 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. 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.* Based on a result of this form, a user may inspect the sorted attributes and their alleged matches and may dismiss all proposed matches occurring after some point as unlikely.

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 (differences of no more than ε), any ordering is considered correct. The $\varepsilon = 0$ case 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 $x_j \geq x_i - \varepsilon$ 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 $\langle r_1, \dots, r_n \rangle$ and $\langle s_1, \dots, s_m \rangle$ respectively, a desired number of matches k, approximation and confidence parameters ε and δ, find a sequence of k attributes $(s_{i_1}, \dots, s_{i_k})$ for each r_i in R, a sequence of k attributes $(r_{j_1}, \dots, r_{j_k})$ 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 $(s_{i_1}, \dots, s_{i_k})$ contains the best matching attributes (accurately up to ε) and, analogously, the sequence $(r_{j_1}, \dots, r_{j_k})$ contains the best matching attributes for each s_j; i.e., there is no offending $s_{i'} \notin (s_{i_1}, \dots, s_{i_k})$ such that $f_{R,S}(r_i, s_{i'}) > \min_{i'' \in \{i_1, \dots, i_k\}} f_{R,S}(r_i, s_{i''}) + \varepsilon$ and no $r_{j'} \notin (r_{j_1}, \dots, r_{j_k})$ such that $f_{R,S}(r_{j'}, s_j) > \min_{j'' \in \{j_1, \dots, j_k\}} f_{R,S}(r_{j''}, s_j) + \varepsilon$;*
2. *for all attributes r_i, the corresponding $(s_{i_1}, \dots, s_{i_k})$ are ordered by their similarity to r_i and for all s_j, the corresponding $(r_{j_1}, \dots, r_{j_k})$ are ordered by their similarity to s_j; i.e., $(f_{R,S}(r_i, s_{i_1}), \dots, f_{R,S}(r_i, s_{i_k}))$ and $(f_{R,S}(r_{j_1}, s_j), \dots, f_{R,S}(r_{j_k}, s_j))$ 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 $(\max_{j'} f_{R,S}(r_{\pi^r(1)}, s_{j'}), \dots, \max_{j'} f_{R,S}(r_{\pi^r(n)}, s_{j'}))$ and $(\max_{i'} f_{R,S}(r_{i'}, s_{\pi^s(1)}), \dots, \max_{i'} f_{R,S}(r_{i'}, s_{\pi^s(m)}))$ are ε-correct orderings.*

4 Schema Matching Algorithms

We are about to present two algorithms that estimate the similarity of attributes based on samples R' and S' of records, drawn at random from R and S. They differ from each other in the way they derive the required sample size.

The first algorithm, FSM, determines a "worst-case" sample size without accessing the database or observing the stream; it only depends on the similarity metric and parameters ε and δ. The sample size suffices to assure the quality guarantee for any possible database; it may be pessimistically large for a given, particular database. The second algorithm is called progressive FSM. Its sample size depends on the database at hand; the sample-dependent bounds used within progressive FSM are technically more involved than the worst-case bounds of regular FSM. Therefore, FSM is a natural baseline for progressive FSM.

Both algorithms use *similarity confidence intervals* which bound the similarity $f_{R,S}$. 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 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 an arbitrary 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*

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

Table 1. FSM: Fast Schema Matching Algorithm.

```
Input: Schemas $R$ and $S$, respectively; desired number of candidates $k$; approximation and confidence parameters $\varepsilon$ and $\delta$.

1. Let $N$ be the smallest number such that $\lceil f_{R',S'}(\cdot,\cdot)\rceil_\delta - \lfloor f_{R',S'}(\cdot,\cdot)\rfloor_\delta \leq \varepsilon$ 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 $\pi^r$ sort the attributes in $R$ by $\max_{s_{i'} \in H^*_{r_i}} \hat{f}_{R',S'}(r_i, s_{i'})$; let $\pi^s$ sort the attributes in $S$ by $\max_{r_{j'} \in H^*_{s_j}} \hat{f}_{R',S'}(r_{j'}, s_j)$.
6. Return $\pi^r$, $\pi^s$, 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.
```

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. For a given similarity metric $f_{R,S}$, the corresponding confidence interval is inserted into the algorithm. (Parameterization with "$\cdot$" indicates "for any pair of attributes".) By contrast, the progressive FSM algorithm described in Table 2 starts to draw batches of records from the databases and then calculates confidence intervals in 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.

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 arbitrary 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 $0 < \delta \leq 1$, $0 < \varepsilon$, and k, an approximately optimal matching as detailed in Definition 2.*

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

5 Attribute Similarity Metrics

We will now briefly sketch three exemplary similarity measures for attributes; we refer the reader to the corresponding technical report [**?**] for the full treatment. Various similarity measures require measuring the similarity of two probability distributions. For this purpose we will use a measure E based on Euclidean

Table 2. Progressive FSM Algorithm.

```
Input: Schemas $R$ and $S$, respectively; desired number of candidates $k$; approximation and confidence parameters $\varepsilon$ and $\delta$.
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 $\lceil f_{R',S'}(\cdot,\cdot)\rceil_{\frac{\delta}{2}} - \lfloor f_{R',S'}(\cdot,\cdot)\rfloor_{\frac{\delta}{2}} \leq \varepsilon$ when $R'$ and $S'$ contain $N$ records. Let $T = N/\text{batch size}$.
For $t = 1 \dots T$ Repeat:
    1. Let $\delta_t = \frac{\delta}{2t(t+1)}$.
    2. Draw batches of records from $R$ and $S$, 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 $\lceil f_{R',S'}(r_i,s_j)\rceil_{\delta_t} - \lfloor f_{R',S'}(r_i,s_j)\rfloor_{\delta_t} \leq \varepsilon$ 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 $\pi^r$ sort the attributes in $R$ by $\max_{s_i^* \in H^*_{r_i}} \hat{f}_{R',S'}(r_i,s_i^*)$; let $\pi^s$ sort the attributes in $S$ by $\max_{r_j^* \in H^*_{s_j}} \hat{f}_{R',S'}(r_j^*,s_j)$.
Return $\pi^r$, $\pi^s$, 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.
```

distance. Let $P = (p_1, \dots, p_n)$ and $Q = (q_1, \dots, q_n)$ 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.

Euclidean Distance. Given two attributes r and s with identical domains, a natural measure of their similarity is given by $f^E_{R,S}(r,s) = E(P_r, Q_s)$, where P_r and Q_s are probability distributions of r and s respectively. The upper and lower bounds are then

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

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

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. The similarity measure is now defined as

$$f^{\pi E}_{R,S}(r,s) = \max_{\pi \in \Pi(n)} E\left(P_r, \pi Q_s\right), \tag{2}$$

where $\Pi(n)$ is the set of all permutations of $\{1, \ldots, n\}$. The following similarity confidence intervals satisfy Definition 3.

$$\lfloor f^{\pi E}_{R',S'}(r,s)\rfloor_\delta = \max_{\pi \in \Pi} \lfloor f^{E}_{R',S'}(r,\pi s)\rfloor_\delta; \qquad \lceil f^{\pi E}_{R',S'}(r,s)\rceil_\delta = \max_{\pi \in \Pi} \lceil f^{E}_{R',S'}(r,\pi s)\rceil_\delta.$$

Bigrams. This measure is based on the distribution of bigrams (*i.e.,* two character subsequences) in attribute values. We hash all possible bigrams into B buckets; the resulting bounds are

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

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

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. Vector $P^r = (P^r_l, P^r_u, P^r_d, P^r_p)$ contains the averaged proportions of records belonging too these classes. 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 the above case.

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.

$$f_{R,S}(r,s) = \frac{w_{\pi E} f^{\pi E}_{R,S}(r,s) + w_{Bg} f^{Bg}_{R,S}(r,s) + w_c f^c_{R,S}(r,s)}{w_{\pi E} + w_{Bg} + w_c} \tag{3}$$

The resulting bound for the weighted sum is simply the weighted sum of the corresponding bounds.

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 utilize this information. The similarity function can combine schema- and instance-level components: $f_{R,S}(r,s) = w_S f^{Schema}(r,s) + w_I f^{Instance}_{R,S}(r,s)$. 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

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

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

6 Experiments

In our experiments we want to investigate (a) whether the FSM and progressive FSM algorithms are practically applicable for large databases; we want to (b)

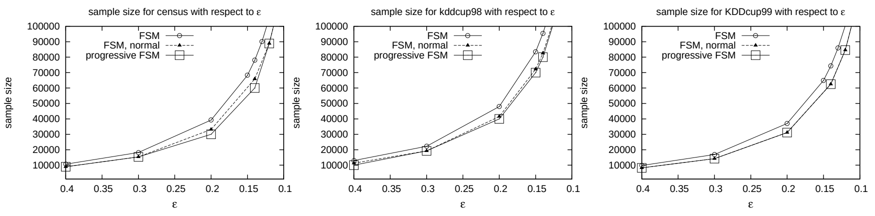

Fig. 1. Sample size against ε for schema matching algorithms.

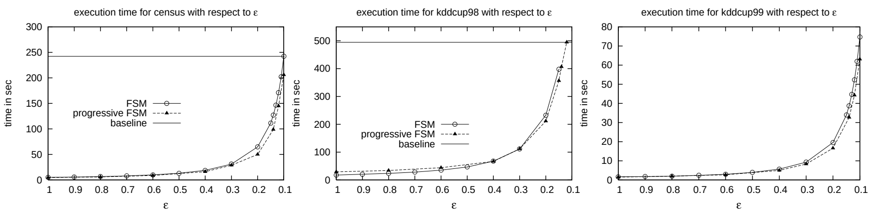

Fig. 2. Execution time against ε for schema matching algorithms.

compare the performances of FSM (using Hoeffding's and Normal bounds), progressive FSM (using Normal bounds), and a baseline instance-based matcher. The baseline matcher uses the same similarity function but processes the entire database, thus always retrieving 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. We use three publicly available databases: the KDDCup 1998 customer relationship management database [1] contains with 481 attributes and over 190,000 records; the KDDCup 1999 database [2] with nearly 5 million records and 42 attributes, 17 of which are almost always zero; and the census database [3] with 42 attributes (the majority of them contain text) and close to 300,000 records.

In order to conduct controlled experiments, we randomly split each of the databases into two parts containing half of the records. 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, recall, and F-measure.

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. 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

[1] http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html
[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[3] http://kdd.ics.uci.edu/databases/census-income/census-income.html

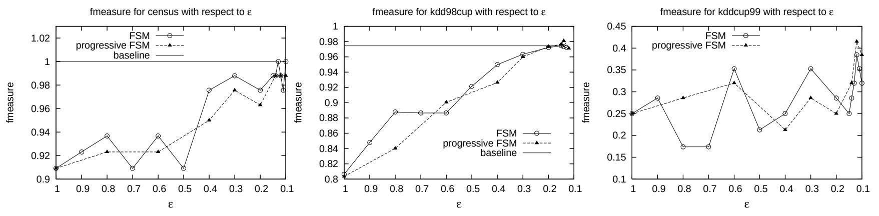

Fig. 3. F-measure against ε for schema matching algorithms.

stopping in Step 4 is never exercised. The reason is that due to the sparseness of the data the matching problem is very hard. For all databases, we observe that the Normal bounds reduce the required sample size over the Hoeffding bounds used by the FSM algorithm.

Figure 2 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 (with $\delta = 0.1$). Again, we observe that progressive FSM is the fastest method, followed by FSM. For the KDD Cup 1999 database, the baseline algorithm exceeds our patience.

Finally, Figure 3 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.

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 our patience, 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.

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 algorithms 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 required sample size of the algorithms depends on parameters ε and δ, but is independent of the database size. The database can even be an infinite stream. For the progressive FSM algorithm, sample size depends on the actual databases, 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 the theoretical results guarantee an approximately optimal match, we observe empirically that attributes are often actually matched to semantically equivalent attributes.

References

1. D. Aumueller, H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *Proceedings of the ACM SIGMOD Conference*, 2005.
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 Osamu Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. *Data Mining and Knowledge Discovery*, 6(2):131–152, 2002.
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. W. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the International Conference on Very Large Databases*, 1994.
10. W. Li and C. Clifton. Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural network. *Data and Knowledge Engineering*, 33(1):49–84, 2000.
11. J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of the International Conference on Very Large Databases*, 2001.
12. E. Rahm and P. Bernstein. A survey of approaches to automatic schema mapping. *VLDB Journal*, 10:334–350, 2001.

13. M. Sayyadian, Y. Lee, A. Doan, and Arnon Rosenthal. Tuning schema matching software using synthetic scenarios. In *Proceedings of the VLDB Conference*, 2005.
14. T. Scheffer and S. Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *Journal of Machine Learning Research*, 3:833–862, 2002.
15. 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.