Training Selection for Tuning Entity Matching

Hanna Köpcke
University of Leipzig, Germany
koepcke@informatik.uni-leipzig.de

Erhard Rahm
University of Leipzig, Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

Entity matching is a crucial and difficult task for data integration. An effective solution strategy typically has to combine several techniques and to find suitable settings for critical configuration parameters such as similarity thresholds. Supervised (training-based) approaches promise to reduce the manual work for determining (learning) effective strategies for entity matching. However, they critically depend on training data selection which is a difficult problem that has so far mostly been addressed manually by human experts. In this paper we propose a training-based framework called STEM for entity matching and present different generic methods for automatically selecting training data to combine and configure several matching techniques. We evaluate the proposed methods for different match tasks and small- and medium-sized training sets.

1. INTRODUCTION

Entity matching (also known as entity resolution, deduplication, record linkage, object matching, fuzzy matching, similarity join processing, reference reconciliation) is the task of identifying object instances or entities referring to the same real-world object. It is a crucial step in data cleaning and data integration. Entities to be resolved may reside in distributed, typically heterogeneous data sources or may be stored in a single data source, e.g., in a database or search engine store. They may be physically materialized or dynamically be requested from sources, e.g., by database queries or keyword searches.

Entities from web data sources are particularly challenging to match as they are often highly heterogeneous and of limited data quality, e.g., regarding completeness and consistency of their descriptions. Fig. 1 illustrates some of the problems for the popular web entity search engine Google Scholar and five duplicate entries for the same paper. The bibliographic entities have automatically been extracted from web pages or PDF documents and contain numerous quality problems such as misspelled author names, different ordering of authors, heterogeneous venue denominations etc.. Google Scholar performs already an entity matching by clustering references to the same publication to aggregate their citations and fulltext sources. However as the duplicates in the example show, the obtained results are far from perfect influenced by the mentioned quality and heterogeneity problems. This illustrates that there is a big potential for improving data quality by better entity matching techniques. This would be critical for tasks requiring the examination of all duplicates, e.g., to collect all citations of publications for a citation analysis. Similar match problems appear in many application domains, e.g., to compare prices for equivalent products (DVDs, cameras, hotel rooms, etc.) offered in the same or different web sites.

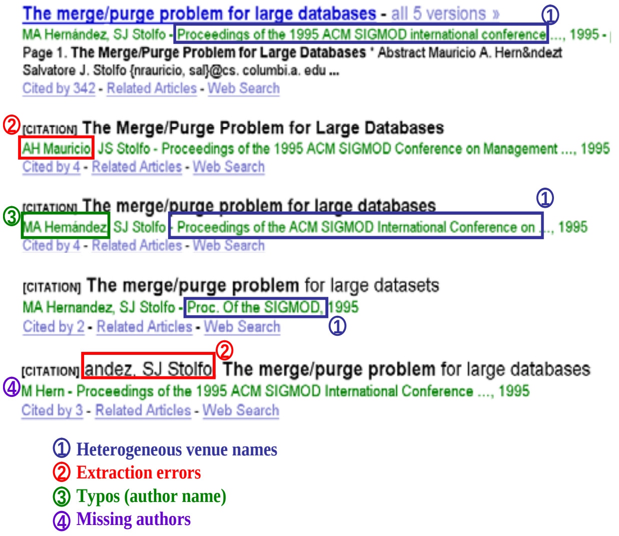
Numerous approaches have been proposed for entity matching (EM) especially for structured data, e.g., in relational databases [13]. Due to the large variety of data sources and entities to match there is no single “best” solution. Instead it is often beneficial to combine several algorithms for improved EM quality, e.g., to consider the similarity of several attributes or related entities. The need to support multiple approaches has led to the development of several entity matching frameworks, e.g., [12], [27]. Such frameworks allow the flexible combination and customization of different methods to achieve good quality results for a given EM task. Unfortunately, the flexibility comes at the price of requiring the user to determine a suitable match strategy for a particular problem. Key decisions to be made include

- Selecting the attributes to be used for matching,
- Choosing a similarity function to be applied, and
- Selecting a threshold for the similarity values above which entities are considered to match.

The chosen configuration can have a large impact on the overall quality but even experts will find it difficult and time-consuming to determine a good selection. This is because there are typically numerous possibilities for each decision resulting in a huge number of possible combinations from which to choose for configuring an entity matching strategy. We thus see a strong need for self-tuning entity matching to largely automatically determine an effective EM strategy for a given task.

While a number of previous studies has investigated training-based (supervised) machine learning for entity matching (see related work), the challenge of self-tuning entity matching is far from solved. An important issue is how training data is selected. Most previous studies on training-based entity matching have provided little details on the selection and size of training data and have not studied the influence of different training sets [5], [9]. In other cases, the authors used a relatively large amount of training data thus favoring good match quality however at the expense of a high manual overhead for labeling [8], [21], [22]. In this study we propose different generic methods for automatically selecting training data to be labeled. The training selection methods are part of a new generic framework that supports the automatic construction of entity matching strategies. In this study the proposed framework serves as an evaluation platform to compare

VLDB '08, August 24-30, 2008, Auckland, New Zealand.



The merge/purge problem for large databases - all 5 versions »
MA Hernández, SJ Stolfo - Proceedings of the 1995 ACM SIGMOD international conference ..., 1995 -
Page 1. The Merge/Purge Problem for Large Databases ' Abstract Mauricio A. Hern&ndezt Salvatore J. Stolfo {nrauricio, sal}@cs. columbi.a. edu ...
Cited by 342 - Related Articles - Web Search

[CITATION] The Merge/Purge Problem for Large Databases
AH Mauricio JS Stolfo - Proceedings of the 1995 ACM SIGMOD Conference on Management ...., 1995
Cited by 4 - Related Articles - Web Search

[CITATION] The merge/purge problem for large databases
MA Hemández, SJ Stolfo - Proceedings of the ACM SIGMOD International Conference on ..., 1995
Cited by 4 - Related Articles - Web Search

[CITATION] The merge/purge problem for large datasets
MA Hernandez, SJ Stolfo - Proc. Of the SIGMOD, 1995
Cited by 2 - Related Articles - Web Search

[CITATION] andez, SJ Stolfo The merge/purge problem for large databases
M Hern - Proceedings of the 1995 ACM SIGMOD International Conference ..., 1995
Cited by 3 - Related Articles - Web Search

Figure 1: Duplicate paper entities in Google Scholar

different training selection methods and determine the impact of different training sizes on EM accuracy.

The main contributions of this paper are:

- Presentation of a comprehensive, training-based framework for automatically constructing (learning) entity matching strategies called STEM for ***s***elf-***t***uning ***e***ntity ***m***atching
- Description of methods to automatically generate training data for entity matching.
- Comparative evaluation of the training selection methods for different training sizes and four match tasks of different application domains. We specifically analyze the effectiveness for small training sizes which incur only a modest effort for labeling.

In the next section we outline our framework for tuning entity matching. We then propose two methods for selection of training data. Section 4 provides a comprehensive evaluation of the effectiveness of the methods for four match tasks. Related work is reviewed in Section 5 before we conclude.

2. STEM: A TRAINING-BASED ENTITY MATCHING FRAMEWORK

2.1 Problem definition

We assume the sets of entities to be resolved come from two sources S_A and S_B. Entities are of a particular semantic entity type (e.g., publication or product). Each entity is represented by a set of attribute values. The entity matching problem is to find all pairs of equivalent entities which refer to same real-world object.

Definition 1 (Entity match problem / match task): Entity matching is a process which takes as input two sets of entities $A \subseteq S_A$ and $B \subseteq S_B$ of a particular semantic entity type from data sources S_A and S_B. The output is a subset of A x B containing all correspondences between entities representing the same real-world object.

Note that the inputs need not be entire sources but can be subsets of them, e.g., the results of queries. Resolving entities within a single source is supported as a special case ($A=B$, $S_A = S_B$).

To solve an entity matching problem we want to utilize several matchers in combination. A *matcher* is defined as follows:

Definition 2 (Matcher): A matcher m takes as input two sets of entities $A \subseteq S_A$ and $B \subseteq S_B$ of a particular semantic entity type from data sources S_A and S_B. It produces as output a similarity table $T = \{(a_, b, s) \mid a \in A, b \in B, s \in [0,1]\}$. The similarity value s indicates the strength of the similarity between two entities $a \in A$ and $b \in B$.

Given a match task and a set of matchers the general objective of our framework is to automatically find an optimal entity matching strategy. An EM strategy defines a selection, configuration and combination of matchers to solve an entity matching problem. It can be defined as follows:

Definition 3 (EM strategy): An EM strategy ($S_{A,}$ $S_{B,}$ M, f) for two data sources S_A and S_B of a particular semantic entity type consists of a set M of matchers and a decision function f. For any entity pair (a, b| $a \in S_A$, $b \in S_B$), f applies the matchers M and assigns a label $l \in$ {match, non-match}. For any subsets $A \subseteq S_A$ and $B \subseteq S_{B,}$ the EM strategy returns as output a set of correspondences $C = \{(a, b| a \in A, b \in B, f(a, b) = match)\}$ consisting of all entity pairs labeled with l = match by f.

The general tuning problem can be defined as follows:

Definition 4 (General tuning problem): Given two sets of entities $A \subseteq S_A$ and $B \subseteq S_B$ of a particular semantic entity type from data sources S_A and S_B, the general tuning problem is to find an optimal EM strategy s which maximizes a utility function U.*

The utility function U may take diverse factors into account, in particular matching accuracy and execution time. For the initial evaluation of our framework we focus on the matching accuracy measured in terms of precision, recall and F-measure (Section 4.1). We plan to consider more general utility functions including execution time as future work.

2.2 Framework architecture

Fig. 2 illustrates the architecture and use of our generic entity matching framework STEM. With generic we mean that the framework should be applicable to different application domains, to different data sources and to different subsets of a data source.

We distinguish two main processing phases or modes: 1) the specification and 2) the application of an EM strategy. The *specification phase* determines suitable strategies to solve a new EM problem. This phase is typically determined offline, either completely manually or largely automatically by machine learning techniques based on training data. EM strategies determined in the specification phase are stored in a *repository* together with describing properties (input sources, entity type, output characteristics, observed execution times, etc.).

In the *application phase* an EM strategy is selected from the repository and applied to the input entities to be resolved. This phase may be performed offline or online, e.g., for input data that is derived at runtime by queries or other application processing (e.g., mashups). The result of applying the EM strategy is a set of

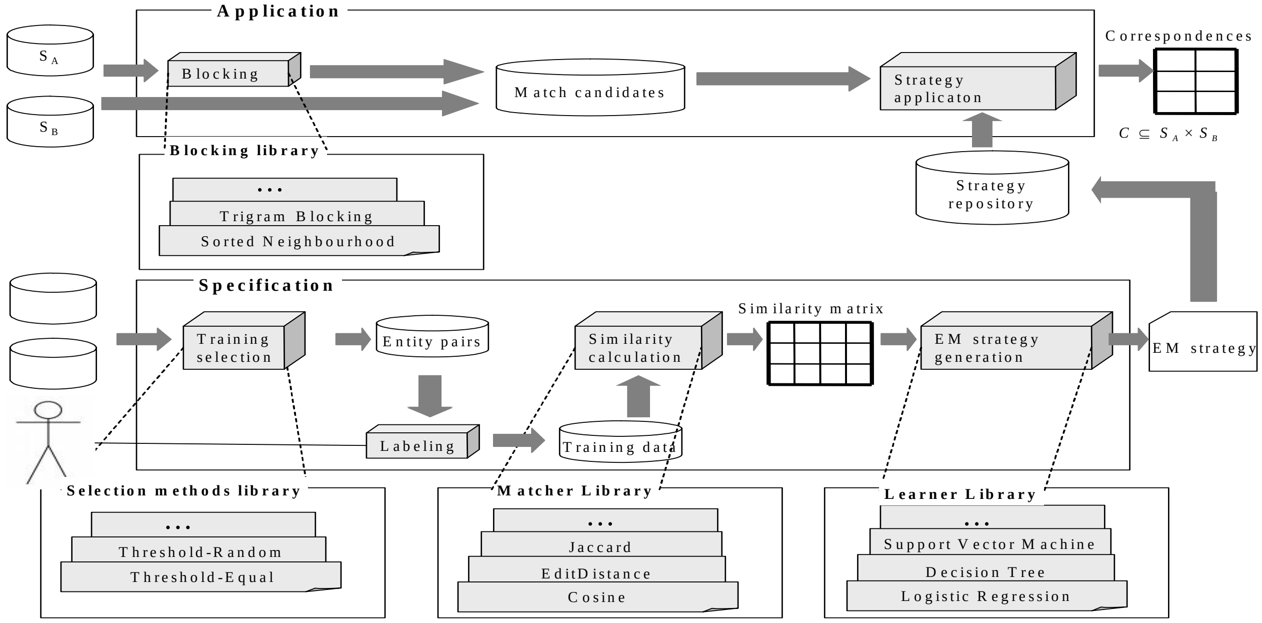


Figure 2: Overview of STEM

correspondences indicating matching entities, i.e. entities which are considered to represent the same real-world object. The EM strategy may be applied on the whole input data or on the result of a blocking step. Blocking is needed for large inputs and reduces the search space for entity matching from the Cartesian product S_A x S_B to a small subset of the most likely correspondences. There are many known algorithms for blocking [3] and we provide several of them, e.g., to exclude all pairs with very low string similarity.

The selection of an EM strategy is mainly determined by the given input sources; in case of multiple choices the selection may be controlled manually or based on statistical data from previous use cases such as execution time or result quality. In this paper, we manually specify the EM strategy to be applied since we want to comparatively evaluate different strategies. In future work we plan to automatically select from predefined EM strategies as an additional kind of tuning. For example, this feature can be useful to consider runtime constraints to select a fast EM strategy with sufficient recall/precision.

In this paper we focus on the training-based generation of EM strategies in the specification phase. This is performed offline because it requires some manual guidance for labeling training entities resulting in a semi-automatic generation of EM strategies. The automatically executed steps can also be time-consuming especially when we generate several strategies to choose from in the application phase. The workflow of determining an EM strategy is illustrated in the lower part of Fig. 2. It consists of three main steps:

- *Generation of training data.* We do not require the manual specification of training data but allow that training data is automatically selected from the entity sets to be matched or other data sets from the same application domain. The result of this step is a set of entity pairs T and a labeling. The labeling is determined manually and indicates for each pair whether the two entities match or not. The pairs to be labeled are determined based on a training selection method.
- *Similarity computation*. Multiple matcher algorithms are applied to determine the similarity for entity pairs in the training data T. Either all supported matchers or a specified subset of them is applied to the training input. The result of this step is a similarity matrix indicating a similarity value for each training pair and matcher.
- *Learning.* The final step is the adoption of a supervised learner algorithm to determine the EM strategy. The learner uses the similarity matrix and the manual labeling as input. It determines the execution order and weighting of the matchers and specifies how matchers are combined.

For each of the three steps our framework provides several methods to choose from as well as additional configuration options. In particular we support several approaches for training selection, for matching, and for learning. The approaches used in our evaluation are (partially) indicated in Fig. 2. The modularity of our framework allows adding further methods if necessary or promising.

The methods for training selection are presented in the next section. The matcher library provides different similarity functions, in particular generic string similarity measures (edit distance, q-grams, cosine, TF/IDF, etc), for defining attribute matchers. An attribute matcher uses one of the similarity functions and applies them to a pair of corresponding attributes of the input entity sets for which the similarity is to be computed. Additional matchers utilizing context information or auxiliary information such as dictionaries can be added as needed.

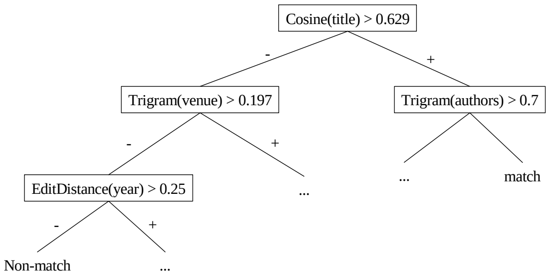


Figure 3: Sample EM strategy for a bibliographic match task generated by a decision tree learner

When all (attribute) matchers should be considered for determining an EM strategy all applicable similarity functions are employed for each attribute pair. We may also use a predefined subset of the matchers to speed up the execution time for matching both during the specification and application phase. Moreover, a smaller number of matchers may require fewer training data and still achieve sufficient EM quality.

The learner library provides several supervised machine learning algorithms or learners, in particular decision tree, SVM (support vector machine), logistic regression and their combination. The learners treat the task of determining an EM strategy as a two-class (match or non-match) classification problem. On the basis of the similarity values for the training examples the learners automatically determine the order and weighting of the matchers and specify how they are combined. Fig. 3 shows a sample EM strategy for a bibliographic match task as determined by the decision tree learner. Each node of the decision tree contains a test whether or not a certain similarity threshold is exceeded for a selected matcher. The match decision is reached by starting with the test of the root node and continuing with the further tests until a leaf node is reached.

While the flexibility of the architecture allows the generation of many strategies for different training selection methods, matcher configurations, and learners, we will here use this flexibility mainly for the comparative evaluation of different approaches. The goal is to identify effective default methods so that only a few EM strategies need to be generated for selection in the application phase.

3. TRAINING DATA SELECTION

The effectiveness of supervised machine learning methods critically depends on the size and quality of the available training data. For entity matching it is important that the training data is representative for the entities to be matched and exhibit the variety and distribution of errors observed in practice. Furthermore, the training data should allow the observation of differences between the available matcher algorithms so that an effective combination of different algorithms can be learned. This requires that the training data contain a sufficient number of both matching as well as non-matching entity pairs. On the other hand, for tuning entity matching it is important to limit the manual overhead for labeling. Hence, we wish to keep the number of training entity pairs to be labeled manually as low as possible.

Most previous studies on training-based entity matching have provided little details on the selection and size of training data and have not studied the influence of different training sets [5], [9]. In other cases, the authors used a relatively large amount of training data thus favoring good match quality however at the expense of a high manual overhead for labeling [8], [21], [22]. In this study we propose different generic methods for automatically selecting training data to be labeled. Furthermore, we want to use the flexibility of our framework to evaluate these methods and determine the impact of different training sizes on EM accuracy.

A naïve method for training selection is to randomly choose entities from the sets of entities to be resolved for labeling. However, this way it cannot be guaranteed that we obtain representative training data. This is because for a given entity from the first dataset, most entities from the second dataset are most likely non-matches while only few will match. Hence, a random selection of entity pairs will result in an imbalanced training set where the non-matching entity pairs will heavily outnumber the matching pairs. This phenomenon is known as the class imbalance problem and has been reported as an obstacle to the generation of good classifiers by supervised machine learning algorithms [17].

A better training strategy is the so-called static-active selection of entity pairs proposed in [6]. This method compares the entities to be resolved with some state-of-the-art string similarity measure and selects only pairs that are fairly similar according to this measure, i.e. their similarity value meets a certain threshold. By asking the user to label those entity pairs a training sample with a high proportion of matching pairs can be obtained. At the same time, non-matching entity pairs selected using this method are likely to be “near-miss” negative examples that are more informative for training than randomly selected pairs most of which tend to be “easy” non-matches. The idea is easy to implement and seems a reasonable way to obtain representative training data. We thus want to further evaluate it as a strategy for training selection. In particular, we want to empirically investigate the following questions not considered in [6]:

- To what degree can we solve the class imbalance problem, i.e. provide sufficient matching and non-matching entity pairs?
- How influencing is the choice of the threshold for the similarity measure above which pairs of entities are selected for labeling? Is there a good default threshold?
- What is the influence of the training size on EM accuracy, i.e. how few training pairs would be sufficient to obtain good results?

For this study we propose and evaluate two strategies for selecting entity pairs:

- *Threshold-Random (n,m,t)*: n object pairs are randomly selected among the ones satisfying a given minimal threshold t applying a similarity measure m. This approach is very simple but may be prone to the class imbalance problem, e.g., for lower (higher) thresholds the number of non-matching (matching) pairs may dominate the training.
- *Threshold-Equal (n,m,t)*: This strategy selects an equal number (n/2) of matching and non-matching pairs from the ones satisfying a given minimal threshold t applying

a similarity measure m. This approach may require a higher manual overhead for labeling than Threshold-Random because after n selections one of the two classes (matching / non-matching) will most likely not yet have n/2 training examples. Hence, we have to select and label further pairs until n/2 examples are available for both classes.

Alternatively to these two selection methods, entity pairs could be selected using active learning methods [23], [26]. Active learning methods attempt to iteratively identify those pairs leading to maximal performance improvements when added to the training set. These pairs are then presented to the user for labeling. The system is re-trained on the training set including the newly added labeled example. We plan to incorporate selection methods based on active learning into our framework in the future and compare it with the two selection methods proposed in this paper.

4. EXPERIMENTAL EVALUATION

We present a comprehensive evaluation of the proposed training selection methods within the presented framework for real-life datasets, including bibliographic references from Google Scholar. We also investigate the impact of training size on the effectiveness of different learners for automatically constructing EM strategies.

We first outline the evaluation setting, in particular the chosen datasets, match algorithms and learners. Section 4.2 evaluates the Threshold-Random and Threshold-Equal approaches for training selection. We then comparatively evaluate the effectiveness of the learners (Section 4.3) and different match configurations (Section 4.4) for different training sizes.

4.1 Evaluation Setting

4.1.1 Datasets

We evaluate our approach for four match tasks for which the perfect match result is known. Table 1 indicates for each task the number of involved objects, the number of available attributes and the number of correspondences in the perfect match result. The two largest match tasks come from the bibliographic domain and involve subsets of three real-life data sources: Google Scholar (Scholar), DBLP and ACM Digital Library (ACM). We will provide more details on the associated match tasks below. The other (smaller) match tasks cover different domains and have been taken from the RIDDLE repository[1]. The Parks task consists of two data sources with names of parks; only one attribute is provided per source. The Restaurant problem compares restaurant listings from two restaurant guides. Both sources have four attributes: name, address, city and cuisine. We applied our framework to the remaining RIDDLE match tasks as well but observed similar results than for two selected tasks.

The bibliographic tasks are significantly larger than the RIDDLE tests and match publication sets between Scholar and DBLP (Scholar-DBLP) and between ACM and DBLP (ACM-DBLP). Scholar maintains a huge collection of publication entries automatically extracted from documents. As illustrated in the introduction this source has many data quality problems and duplicates making it very challenging to perform entity matching. DBLP and ACM focus on computer science publications and are manually maintained. Compared to Scholar they are of higher quality, especially DBLP. Since DBLP has almost no duplicates the EM result between Scholar and DBLP can also be used for determining the duplicates in Scholar. This is because all Scholar entries matching the same DBLP publication can be considered duplicates.

Table 1: Overview of evaluation match tasks

match task	# entities		# attr.	# corresp.
	source1	source2		
Scholar-DBLP	64,263	2,616	4	5,347
ACM-DBLP	2,294	2,616	4	2,224
Parks	258	396	1	250
Restaurant	533	331	4	112

As shown in Table 1, our evaluation datasets cover 2,616 publications from DBLP, 2,294 publications from ACM and 64,263 publications from Google Scholar. We thus have up to 169 million entity pairs (Scholar-DBLP) in the Cartesian product between these datasets. We applied a simple blocking strategy based on a trigram similarity check with a low threshold to eliminate most pairs which are clearly non-matches. The reduced datasets contain about 607,000 (Scholar-DBLP) and 494,000 (ACM-DBLP) entity pairs. To determine the quality of the entity matching strategies we manually determined the "perfect" EM results mappings with the cardinalities shown in Table 1.

4.1.2 Match algorithms

For the evaluations we consider attribute matchers based on generic string similarity measures. Each matcher thus specifies one of the available string similarity functions as well as the pair of attributes of the input datasets for which the similarity is computed. The corresponding attributes from the input datasets have been manually determined beforehand. Table 1 indicates that between one and four attribute pairs could be used for matching (e.g., the bibliographic attributes title, authors, venue, and year). We use the following seven string similarity measures (for detailed descriptions see, e.g., [10], [13]): EditDistance, Cosine, Jaccard, Jaro-Winkler, Monge-Elkan, Trigram and TF/IDF. Applying the seven similarity functions to the available attributes gives a total of 28 attribute matchers for the bibliographic and Restaurant test cases and 7 matchers for the Parks problem.

4.1.3 Learners

We evaluate three supervised learners which have been utilized previously for entity matching, namely decision trees, logistic regression and Support Vector Machine (SVM). In addition, we evaluate a *multiple learning approach* combining the three single learners through voting. Two entities are considered a match if at least two of the three learners classify it as a match (majority consensus). The motivation is that the combined approach may compensate weaknesses of individual learners. For our experiments we use the learner implementations provided by RapidMiner, formerly Yale [20], a free open-source environment for machine learning algorithms.

[1] http://www.cs.utexas.edu/users/ml/riddle/

4.1.4 Evaluation Measures

As in many other EM evaluations, we measure the quality of the match strategies with the standard measures precision and recall, and F-measure with respect to the manually determined "perfect" mappings. The three measures are formally defined as follows: Let M_p be the set of correspondences that an EM strategy identifies. Let M_a be the set of all correct correspondences from the set of entities to be resolved. Precision is defined as $P = |M_p \cap M_a| /| M_p|$ recall $R = | M_p \cap M_a |/| M_a|$, and F-measure = $(2P \cdot R)/(P + R)$.

4.1.5 Manual baseline strategies

For comparison with the automatically generated match strategies we manually configured a baseline strategy for each match task.

For the *bibliographic match tasks* we evaluated 14 configurations (thresholds 0.5 and 0.8) for the seven similarity measures and 18 configurations using the trigram similarity measure for two threshold values (0.5 and 0.8) either on one attribute (title or authors) or using two attributes. The EM strategy using trigram similarity on both title and authors with a threshold of 0.5 performed reasonably well on both EM tasks (F-measure 91.4% for the ACM-DBLP task and 82.3% for Scholar-DBLP). We therefore choose this strategy as a baseline strategy.

For each of the two simpler RIDDLE match tasks we evaluated 14 attribute configurations (thresholds 0.5 and 0.8 for each of the seven similarity measures) and use the best configuration as the baseline strategy. For the Parks problem MongeElkan with threshold 0.8 performs best (92.3% F-measure) while on the Restaurant task the best configuration is Trigram with threshold 0.8 (88.1% F-measure).

4.2 Evaluation of training selection methods

We evaluate the two methods proposed for selecting entity pairs for labeling (Section 3.1), *Threshold-Random (n,m,t)* and *Threshold-Equal (n,m,t)*. Since we want to minimize the manual labeling work as much as possible we focus on the selection of only few training pairs (n = 20, 50, 100, and 500). The threshold t is varied from 0.4 to 0.8.

Fig. 4 displays the F-measure results for the four match tasks Scholar-DBLP, ACM-DBLP, Parks and Restaurant. The results for *Threshold-Random* (*Threshold-Equal*) are shown in the top (lower) four diagrams. The results are achieved with Trigram attribute matchers and SVM as the learner. Other match configurations and learners gave similar relative results. For comparison the F-measure results for the manually determined baseline match configurations are also shown.

We first observe that despite the use of very small training sets the constructed EM strategies outperform the baseline strategy in many cases. The simple *Threshold-Random* approach is somewhat more dependent on the training size n and the choice of the threshold than *Threshold-Equal,* especially for the ACM-DBLP and Restaurant problems. As indicated in the top diagrams of Fig. 4, the achieved F-measure results and thus the training quality drop sharply for threshold values of 0.7 or higher for the ACM-DBLP and the Restaurant problems. This is because the ACM-DBLP and Restaurant problems all involve relatively clean datasets for matching, thus high threshold values mainly select matching entity pairs. Hence *Threshold-Random* runs into a class imbalance problem with many matching and few non-matching pairs. Additionally, the non-matching entity pairs selected with a high threshold may be rare outliers and we risk that the learner is

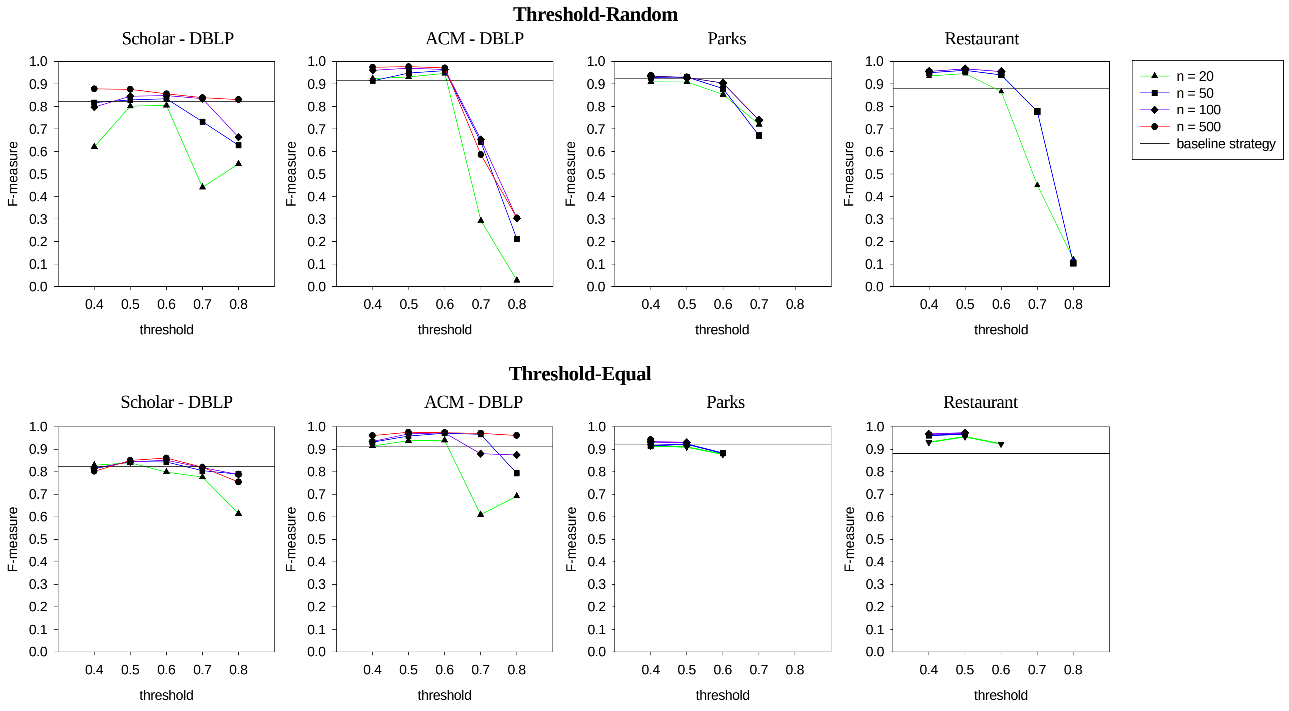


Figure 4: Comparison of Threshold-Random and Threshold-Equal training selection

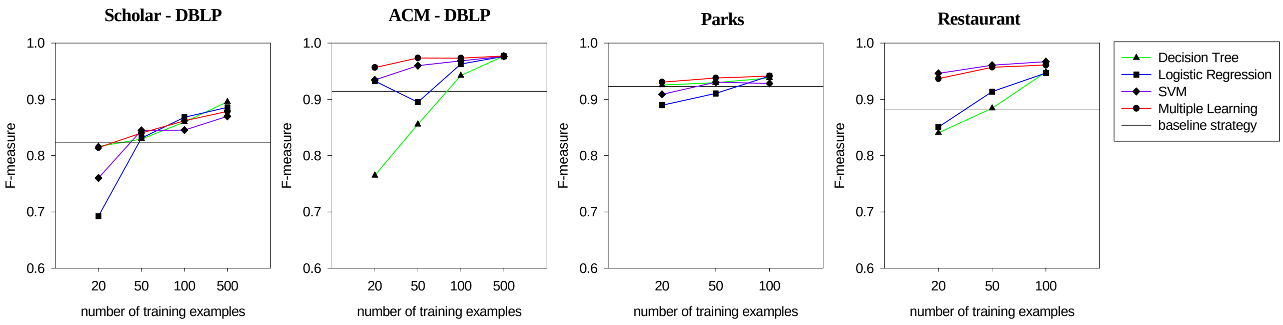


Figure 5: Comparison of different learners

overfitted to those special cases preventing to classify other entity pairs correctly. For threshold values 0.4 to 0.6, *Threshold-Random* achieves very good results for ACM-DBLP and Restaurant. For Scholar-DBLP the training size n seems more significant than the choice of threshold, but the best results are again achieved for t between 0.4 and 0.6.

For all four match tasks, *Threshold-Equal* also performs best in the threshold range 0.4 to 0.6. However, it is much more stable than *Threshold-Random* with respect to different training sizes and threshold values because it avoids the class imbalance problem. This comes at the prize of a higher manual labeling effort to guarantee equal numbers of matching and non-matching entity pairs in the resulting training set. For example, for obtaining 20 entity pairs with equal numbers of matching and non-matching pairs for the Scholar-DBLP match problem using a threshold of 0.4 we had to label 400 pairs, that is a 20 times higher effort than with *Threshold-Random*. For the two smaller problems, Parks and Restaurant, we could not even find the required number of non-matches for Threshold-Equal and threshold values of 0.7 or higher, indicating a limited applicability of this method for small match tasks.

We also observe that *Threshold-Equal* and *Threshold-Random* achieve similar maximal F-measure values of about 86% for Scholar-DBLP, 96% for ACM-DBLP, 93% for Parks and 97% for Restaurant (n=500). We therefore propose the cheap and more general *Threshold-Random* approach with a threshold of 0.5 as the default method for training selection and use it in our further evaluation. We repeated the experiments for other similarity measures than Trigram to select training samples. We found out that TF/IDF performs similarly well and exhibits a comparable behavior w.r.t. the training sizes and thresholds. Hence this measure would be an alternative choice for use with Threshold-Random.

4.3 Learner evaluation

Next, we want to evaluate the influence of the training size on the relative quality of the four supervised learners: decision tree, logistic regression, SVM and the multiple learning approach. We used the same match configurations as for the evaluation of the training selection methods in the previous section.

Fig. 5 shows the F-measure results for the four match tasks achieved with the automatically generated EM strategies and different training sizes (x-axis). Training sizes vary between 20 and 500 entity pairs. For the two smaller match problems the near-optimal results are already achieved for 100 or fewer pairs. We observe that all learners benefit from increasing the training size especially for the more difficult Scholar-DBLP problem. For this task choosing only 20 training entities is clearly insufficient but all four learners match or exceed the baseline performance already for 50 training pairs. For the ACM-DBLP and Restaurant tasks only 20 training pairs were sufficient for SVM and the multiple learner approach to outperform the manually determined baseline approach. The Parks task offered little optimization potential since its data sources provide only one attribute. Furthermore, the chosen baseline configuration performed already very well and could not be much improved by the consideration of additional matchers on the attribute.

Altogether, we see that the multiple learner approach is especially stable and among the top-performing approach for all four match tasks. This shows that it is able to combine the advantages of the different approaches. From the remaining learners, SVM performed best in most cases.

Compared to the multiple learning and SVM approaches the decision tree and logistic regression learners perform worse and rather unstable. On the ACM-DBLP and the Restaurant match tasks they both need a higher number of training examples to construct an effective EM strategy. For less than 100 training pairs the decision tree learner is clearly inferior to the other three learners and even the baseline strategy on the ACM-DBLP match task. The decision tree learner is among the best approaches for two tasks (Scholar-DBLP, Parks) but performs worst for the two other problems.

We therefore conclude that for a small amount of training data the multiple learning approach or SVM can already be very effective and outperform (many) manually configured EM strategies. Furthermore, they are to be preferred to using the decision tree or logistic regression learners.

4.4 Influence of match configuration

To examine the usefulness of using a greater selection of matchers we now compare the performance of the four learners for two match configurations. For this comparison we focus on the more challenging match task Scholar-DBLP. In addition to the previously considered configurations with the four trigram matchers we now consider all (7) similarity measures resulting in a total of 28 matchers for the four attributes.

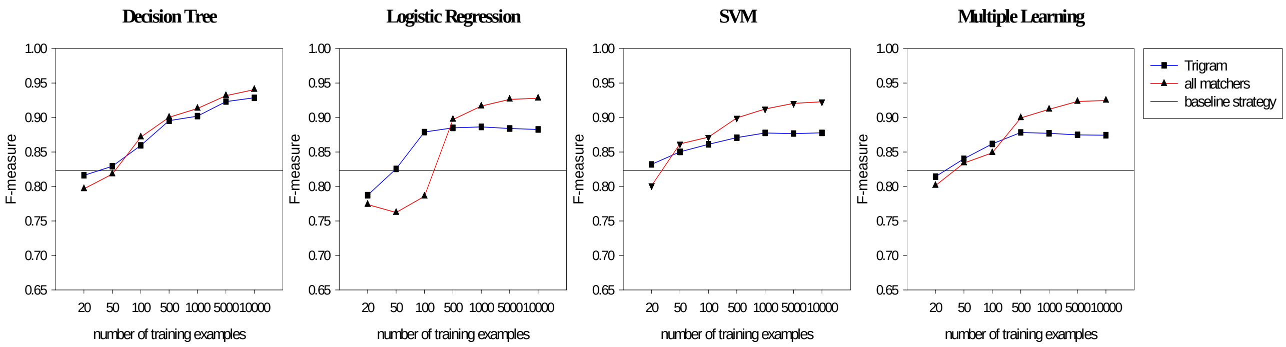


Figure 6: Evaluation of different matcher selections

Fig. 6 contains a separate diagram for each of the four learners comparing the F-measure results for the two match configurations and different training sizes. The curves indicate that for small amounts of training examples the increased choice of matchers does not significantly improve performance but mostly decreases performance. This indicates that the learners need more training to effectively deal with the much increased solution space for selecting, ordering and weighting the applicable matchers. We therefore studied additional training sizes of up to 10,000 training pairs for this experiment. With more training, all learners eventually benefit from the increased number of matchers and outperform the EM strategies based on only four matchers. The improvements are especially significant for the three learners SVM, logistic regression and multiple learners for 500 and more training samples. SVM achieves this already for 50 training examples. The decision tree learner exhibits similar performance for both match configurations so that it cannot effectively utilize the additional matchers. On the other hand, it benefits the most from increased training sizes, especially for the trigram matcher, and achieves the absolute best F-measure (94.1% for n=10,000).

To summarize we observe that our tuning framework can utilize a large choice of matchers to improve performance albeit at the expense of more training. SVM needs the least training data to utilize the increased optimization potential. For large training sizes decision tree performed best.

4.5 Concluding remarks

Our findings so far suggest several default configurations and alternatives for our framework to generate promising EM strategies for a new match problem. For training selection, the threshold-random method should be used to generate n training samples. A good default seems n=100; fewer training samples may suffice for small-sized match problems or comparatively clean data sources (e.g., ACM-DBLP, Restaurant). The matcher configuration may just apply all available matchers (default) or a restricted set of matchers, e.g., with only a subset of string similarity measures such as Trigram and TFIDF. As we have seen, restricting the number of matchers reduces the dependency on larger training sizes; it will also reduce the computational overhead compared to the execution of all matchers. For the learners, both SVM and multiple learning could be applied to generate EM strategies. In future work we will study the tradeoffs between match accuracy and execution time and how suitable heuristics can consider these tradeoffs in choosing from several EM strategies. We will also study methods to automatically propose suitable match configurations, e.g., by using the training samples to determine ineffective matchers beforehand and eliminate them for learning EM strategies

5. Related work

There has been a large body of research on entity matching and its variations. For a recent survey and tutorial see [13] and [18], respectively. Most previous studies used a single match approach like threshold-based attribute matching (similarity join [7]), clustering [19], or context-based matching [1], [11], [29].

A single match approach typically performs very differently for different domains and match problems. For example, it has been shown that there is no universally best string similarity measure [16], [24]. Hence, one needs to customize individual matchers (e.g., selection of attributes and similarity functions) and/or combine several matchers for improved accuracy. Several toolboxes and frameworks support such customization to specify an entity matching strategy, e.g., by a workflow or operator tree of several matchers [14], [12], [25], [27]. However, these specifications typically have to be performed manually which is a very difficult and time-consuming administration task even for tuning experts.

The main approach to automating the generation of entity matching strategies is the use of supervised (training-based) approaches or learners. They aim at adapting the matcher configuration and combination to specific match problems thereby supporting tuning. The learners supported in our framework have already been investigated for entity matching in several previous studies, namely decision trees [15], [23], [26], [28], logistic regression [22], and Support Vector Machine [5], [21], [23]. In addition, maximum entropy [9], and Naïve Bayes [23] approaches have been considered.

Bilenko and Mooney have shown in an empirical evaluation [6] that the Support Vector Machine is more accurate than a decision tree learner. They used SVM at two levels. At the first level they tune four matchers, namely edit distance applied to four attributes of the evaluation dataset. At the second level they apply the Support Vector Machine on the tuned matchers from the previous step.

While the previous studies showed already the potential of supervised methods for tuning, no comprehensive framework such as ours has been studied so far. Our framework allows us to utilize multiple training selection methods, matcher configurations and learners and can easily add additional methods for a comparative evaluation, e.g., like the proposed multiple learner approach. As already discussed in Section 3, most previous evaluations have provided only limited information on how training examples where acquired and how many were necessary to achieve the stated results making it difficult to judge whether good results were due to the approach or clever (time-consuming) manual training data selection.

Recently, Chaudhuri et. al. [8] presented a fast training-based tuning approach for the combined use of several matchers. The idea is to construct (match) operator trees, corresponding to the Or-connection of several And-connected similarity predicates on attributes. These predicates are automatically determined by a recursive divide and conquer strategy on the labeled training data. An implementation based on SQL-Server2005 achieved significantly better execution times than for a standard SVM implementation at a comparable accuracy. The authors do not discuss how they selected the training data; the number of training examples varied for different datasets. The study is complementary to ours since their approach could be plugged into our framework as a promising learner to generate EM strategies.

Most previous work focuses on maximizing the quality of entity matching, in particular precision and recall. Supporting fast entity matching is similarly important especially for large match problems. Several approaches have been developed for blocking (see [3] for an overview) to reduce the search space for entity matching from the Cartesian product to a small subset of the most likely correspondences. Such methods can be plugged into our framework. Similarly we can incorporate proposed enhancements to speed-up individual matchers and learners, e.g., by utilizing set similarity join techniques [2] or the optimizations proposed in [4].

6. SUMMARY AND OUTLOOK

We proposed and evaluated two generic methods for automatically selecting training data to be labeled. The training selection methods are part of a powerful generic framework called STEM that supports the automatic construction of entity matching strategies. The framework addresses the problem of how to automatically configure and combine several matchers within an entity matching strategy.

Our evaluation on diverse datasets showed that automatically constructed EM strategies using the proposed training selection methods can significantly outperform manually configured combined strategies. This is especially true for difficult match tasks and often possible for small training sizes incurring a low labeling effort. The evaluation of several learners revealed that the SVM and the multiple learning approach produce the most stable results even for small training sizes. By contrast decision trees perform well only for large amounts of training data.

In future work, we will further investigate the new framework and focus on both EM quality and runtime performance.

7. REFERENCES

[1] Ananthakrishna, R., Chaudhuri, S., and Ganti, V.: Eliminating Fuzzy Duplicates in Data Warehouses. In *Proc. of VLDB*, 2002.

[2] Arasu, A., Ganti, V. and Kaushik, R..: Efficient exact set-similarity joins. In *Proc. of VLDB*, 2006.

[3] Baxter, R., Christen, P, and Churches, T.: A comparison of fast blocking methods for record linkage. In *Proc. of ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[4] Benjelloun, O, Gracia-Molina H., Kawai, H., Larson, T. E., Minestrina, D., Su, Q., Thavisomboon, S., and Widom, J.: Generic Entity Resolution in the SERF project. *IEEE Data Engineering Bulletin* , Vol. 29, Number 2, 2006.

[5] Bilenko, M. and Mooney, R. J.: Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD*, 2003.

[6] Bilenko, M. and Mooney, R. J.: On Evaluation and Training-Set Construction for Duplicate Detection. In *Proc. of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003..

[7] Chaudhuri, D., Ganti, V., and Kaushik, R.: A Primitive Operator for Similarity Joins in Data Cleaning. In *Proc. of. ICDE*, 2006.

[8] Chaudhuri, S., Chen, B.-C., Ganti,V., and Kaushik, R.: Example-driven design of efficient record machting queries. In *Proc. of VLDB*, 2007.

[9] Cohen, W. W. and Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In *Proc. of ACM SIGKDD*, 2002.

[10] Cohen, W. W., Ravikumar, P., and Fienberg, S. E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proc. of Workshop on Information Integration on the Web (IIWeb)*, 2003.

[11] Dong, X., Halevy, A., and Madhavan, J.: Reference reconciliation in complex information spaces. In *Proc. of ACM SIGMOD*, 2005.

[12] Elfeky, M. G., Elmagarmid, A.K., and Verykios, V.S.: TAILOR: A Record Linkage Tool Box. In *Proc. of ICDE*, 2002.

[13] Elmagarmid, A.K., Ipeirotis, P.G., and Verykios, V.S.: Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19(1), 2007.

[14] Galhardas, H., Florescu, D., Shash, D., and Simon, E.: AJAX: An Extensible Data Cleaning Tool. In *Proc. of ACM SIGMOD*, 2000.

[15] Ganesh, M., Srivastava, J., and Richardson, T.: Mining entity-identification rules for database integration. In *Proc. of SIGKDD*, 1996.

[16] Guha, S., Koudas, N., Marathe, A., and Srivastava, D.: Merging the results of approximate match operations. In *Proc. of VLDB*, 2004.

[17] Japkowicz, N. and Stephen, S.: The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis Journal* 6(5), 2002.

[18] Koudas, N., Sarawagi, S., and Srivastava, D.: Record linkage: Similarity measures and algorithms. In *Proc. of ACM SIGMOD*, 2006.

[19] McCallum, A., Nigam, K., and Ungar, L. H.: Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proc. of ACM SIGKDD*, 2000.

[20] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M, and Euler, T.: Yale: Rapid Prototyping for Complex Data Mining Tasks. In *Proc. of ACM SIGKDD*, 2006.

[21] Minton, S. N, Nanjo, C., Knobloch, C. A., Michalowski, M., and Michelson, M.: A Heterogeneous Field Matching Method for Record Linkage. In *Proc. IEEE International Conference on Data Mining*, 2005.

[22] Pinheiro, J. C., and Sun, D. X.: Methods for linking and mining massive heterogeneous datasets. In *Proc. of ACM SIGKDD*, 1998.

[23] Sarawagi, S., and Bhamidipaty, A.: Interactive deduplication using active learning. In *Proc. of ACM SIGKDD*, 2002.

[24] Sarawagi, S. and Kirpal, A.: Efficient set joins on similarity predicates. In *Proc. of ACM SIGMOD*, 2004.

[25] Shen, W., DeRose, R., Vu, L., Doan, A., and Ramakrishnan, R.: Source-aware Entity Matching: A Compositional Approach. In *Proc. of ICDE*, 2007.

[26] Tejada, S., Knoblock, C. A., and Minton, S.: Learning object identification rules for information integration. *Information Systems Journal*, 26(8), 2001.

[27] Thor, A., and Rahm, E.: MOMA - A Mapping-based Object Matching System. In *Proc. of CIDR*, 2007.

[28] Verykios, V. S., Moustakides, G.V., and Elfeky, M. G.: A Bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12(1), 2003.

[29] Weis, N., and Naumann, F.: DogmatiX tracks down Duplicated in XML. In *Proc. of ACM SIGMOD*, 2005.