Approximate String Matching by End-Users using Active Learning

Lutz Büch
Institute of Computer Science
Heidelberg University, Germany
lutz.buech@informatik.uni-heidelberg.de

Artur Andrzejak
Institute of Computer Science
Heidelberg University, Germany
artur.andrzejak@informatik.uni-heidelberg.de

ABSTRACT

Identifying approximately identical strings is key for many data cleaning and data integration processes, including similarity join and record matching. The accuracy of such tasks crucially depends on appropriate choices of string similarity measures and thresholds for the particular dataset. Manual selection of similarity measures and thresholds is infeasible. Other approaches rely on the existence of adequate historic ground-truth or massive manual effort.
To address this problem, we propose an Active Learning algorithm which selects a best performing similarity measure in a given set while optimizing a decision threshold. Active Learning minimizes the number of user queries needed to arrive at an appropriate classifier. Queries require only the label *match/no match*, which end users can easily provide in their domain. Evaluation on well-known string matching benchmark data sets shows that our approach achieves highly accurate results with a small amount of manual labeling required.

Categories and Subject Descriptors

H.2.m [**Database Management**]: Miscellaneous—*Data cleaning*

Keywords

Active Learning; string similarity; similarity measures; similarity join; string matching; record matching; deduplication

1. INTRODUCTION

Data integration and -cleaning are fields with longstanding importance, that still have space for improvement [19]. Many of the tasks in those fields rely on string similarity measures, sometimes also a threshold that allows string matching. These include record matching [11], similarity join [21] and schema matching [17].

CIKM'15, October 19–23, 2015, Melbourne, Australia.

ACM 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806453.

One immediate application of approximate string matching is similarity join. It is the simple form of record matching, where the matching depends only on one key attribute. Differently from a simple join operation, equivalent entities from both databases may only have similar values in this key attribute. Specification of a similarity measure and threshold define a solution to the similarity join problem. It can then be reduced to an algorithmic problem, that can be efficiently solved, depending on the similarity measure [21].
Record matching relies on similarity measures to compute similarity values on attribute level [11]. In the Fellegi-Sunter model, binary comparison vectors are based on similarity measures and thresholds for the involved attributes, that need to be specified. In many other popular approaches similarity measures need to be specified, while thresholds for these functions themselves are not relevant. E.g., in SVM-based approaches, similarity values of several attributes are combined to a similarity vector. Then, a decision hyperplane is learned from training data. In decision tree based approaches, several thresholds are learned to constitute predicates in logical rules. These thresholds may be inconsistent for the same attribute and similarity measure across different predicates used in a tree.
Bleiholder et al. point out that the effectiveness of the record matching step in data integration is mostly affected by the quality of the similarity measure and the choice of a similarity threshold [5]. Figure 1 illustrates that the choice of a similarity measure can make a huge difference. Here, the measure TagLink equipped with a suitable threshold can match pairs of strings perfectly, while MongeElkan will produce many errors with any threshold.
In [11], a large scale comparison of frameworks for record matching is done. The authors find that supervised matching approaches require less configuration effort and knowledge than others, but aspects like the choice of similarity measures still have to be determined manually.
In this paper we study how to find a suitable similarity measure from a given pool and optimize a corresponding threshold semi-automatically. We propose an Active Learning approach to this problem, in order to minimize human input. The user is iteratively queried for the ground-truth, i.e., whether a pair of strings is equivalent or not. A stopping criterion suggests that a sufficiently accurate choice can be done and will terminate this loop. It will return the chosen similarity measure along with an appropriately tuned threshold as the final output.

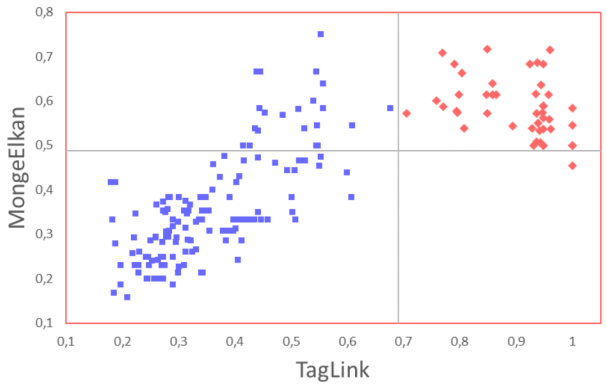

Figure 1: Similarity values and F1-optimal thresholds in data set *ucdFolks* for similarity measures MongeElkan and TagLink; blue squares represent matching pairs, red diamonds non-matching pairs w.r.t the ground-truth

Contribution. The contribution of this paper can be summarized as:

1. We propose an Active Learning algorithm to solve the string matching problem by finding the most appropriate similarity measure from a given pool, while tuning its threshold at the same time.

2. We devise a heuristic stopping criterion that measures a notion of progress of the algorithm. It is designed to stop the querying process early to prevent unnecessary labeling effort for the user.

3. We evaluate our approach on 17 benchmark string matching problems using 24 similarity measures. The results show that the stopping criterion holds after reasonably many queries and the results are close to optimal w.r.t. the hypothesis space.

4. We introduce four new string matching benchmark data sets. We propose corrections to the ground-truth of three widely used benchmark data sets.

2. BACKGROUND

Before proceeding to the introduction of Active String Matching, we provide some important notions and observations.

2.1 String similarity measures

Different fields such as databases, statistics, artificial intelligence and information retrieval have yielded a variety of similarity measures. They account for different variations in strings, like typos, swapping of words, misspellings, format inconsistencies or mishearings (i.e., phonetic variation). There are similarity measures combining others or using them on a lower level of tokens, such as MongeElkan. Others take into account information about the frequency of tokens over the whole data set. For an introduction to string similarity measures, see [7, 15]. A comprehensive account of similarity measures with special focus on name matching can be found in [6]. We regard the different similarity measures simply as black boxes that map pairs of strings on a range of real values, where higher values mean higher similarity. This is usually the interval $[0, 1]$, but this is not required.

2.2 Label distribution

String matching can be cast as a classification problem, where the instances are pairs of strings and the label reads *match* or *non-match*. However, differently from many other classification tasks, it has an inherent skewed label distribution. In a matching problem, the number of pairs will grow quadratically with the size of the involved data sets. At the same time, the number of matches can only grow linearly. In a deduplication task, this is different in theory, as the clusters of duplicates may be of any size. Yet, in practical scenarios, the clusters are limited to small sizes.
Dealing with skewed label distributions is known as the *class imbalance problem* [18, 4]. In [18], there is one short subchapter on it and [4] gives pointers for further reading.
There are two ramifications for this skewed label distribution. First, it influences the choice of quality measure and second, it requires procedures known as *indexing* and *blocking* in the record matching community.

Quality Metric

We measure the quality of a string matching process by *F-measure* (or F_1*-score*). It combines both precision and recall by calculating the harmonic mean of both values. It is widely used in record matching and in information retrieval.
See [13] for a discussion of different quality measures for record matching and especially deduplication. The arguments translate directly to string matching, since it can be seen as a special case, where records have exactly one data field. Most importantly, neither optimizing for precision nor for recall alone will yield useful results in the context of record matching.

Indexing

A skewed label distribution makes learning hard. One idea to mitigate this is to undersample the larger class (i.e., non-matches). The process of undersampling non-matches is called *indexing* (see, e.g. [6, 7]). Indexing can be viewed as a classification of matches aiming for high precision under the constraint of perfect recall. That is, it should eliminate many non-matches without dismissing any matches.
The indexing technique *blocking* eliminates non-matches by requiring the agreement of a *blocking key* that captures a broad measure of similarity. A blocking key can be, e.g., the phonetic code of a name attribute.
We use the *n-gram* blocking implemented in SecondString [2] (with parameter $n = 4$). N-gram blocking is a type of blocking that uses many blocking keys. It generates all n-grams of both strings of a pair and when they agree on at least one n-gram, the pair will be considered for matching. E.g., with $n = 4$, the pair "Franklin Delano Roosevelt", "Grover Cleveland" will be considered, as they share the 4-gram "evel".

2.3 Active Learning

Active Learning is defined as the problem of learning a hidden variable (the *label*), starting from unsupervised data and given the ability to inductively reveal the label of single instances. It is hence a form of semi-supervised learning. The number of labels required to learn a model is part of the performance objective. The model is learned by induc-

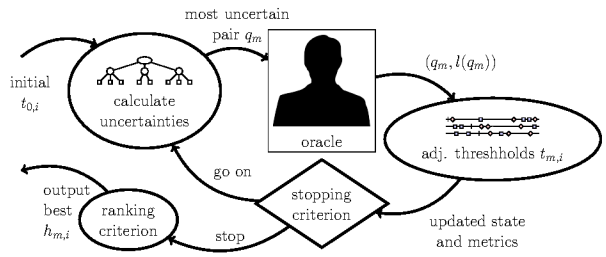

Figure 2: The process of Active String Matching

tively asking for single instances to be labeled, in a way that promises to help best in learning. In each iteration, the label for the new training instance is queried from an oracle such as a domain expert or a physical experiment.
The *query strategy* defines which instance to query next for a label, and is the core of any Active Learning algorithm. It depends on the classification model chosen and the characteristics of the classification problem. It may base its decision on unlabeled data and the few labeled data so far. Some general paradigms have proven very useful in designing such strategies. They are guided by principles that can shortly be described as follows: maximizing uncertainty of the label, minimizing the space of consistent (or likely) hypotheses, minimizing agreement among a committee of hypotheses or maximizing the gained information. See [18] for a general overview over Active Learning and a detailed account of different frameworks and paradigms.

3. ACTIVE STRING MATCHING

This section describes the *Active String Matching* approach proposed in this work. Its purpose is to learn with a least number of user queries a classification function for labeling pairs of strings as matches or non-matches.
We assume a scenario where initially no labels exist. To label the pairs of strings automatically we need a classification model. For this purpose, a user is successively queried to label a pair of strings whether they match or not. Each answer enlarges the training set for learning the model. Based on properties of this interim model, a next pair of strings is selected for labeling, or our algorithm decides to terminate this process. In that case, the algorithm outputs a prediction function (classifier) as its main result. This function can be subsequently used for tasks like similarity join or deduplication. Figure 2 shows the overall process.
The key components of our approach are the following. First, the *learning classifiers* component (Section 3.1) specifies how to optimize the thresholds of individual similarity measures w.r.t. *F_1-score* based on the set of currently known labels. From these, an aggregated prediction function is defined.
Secondly, the *query strategy* (Section 3.2 and Section 3.3) decides which string pair to query next, based on the current knowledge. Finally, the *stopping criterion* decides whether the quality of the learned model is sufficient, in which case it terminates the labeling process (Section 3.4).

We will use the following terms and notations (see Table 1). Let Q be the set of all string pairs that are considered for matching. We denote by m the number of finished iterations (which equals the number of answered queries). The symbol Q_m denotes the set of string pairs $q_1, \ldots, q_m$ which

Symbol	Definition
$m \in \mathbb{N}$	iteration number
Q	set of string pairs
$Q_m = \{q_1, .., q_m\}$	set of m queried string pairs
$l(q_1) \ldots, l(q_m)$	labels provided by the user
$F_1(T, p)$	F_1-score w.r.t. a predictor p and ground-truth of a set $T \subseteq Q$
$S = \{s_1, \ldots, s_n\}$	set of n similarity measures
$t_{m,i} \in \mathbb{R}$	threshold assigned to s_i in iteration m
$p_{m,i}$	predictor based on s_i and threshold $t_{m,i}$ (in iteration m)
p^*_m	best predictor found in iteration m

Table 1: Essential symbols used in Section 3.1

were labeled in iterations $1, \ldots, m$. We write $l(q_1), \ldots, l(q_m)$ for the labels assigned by the user to $q_1, \ldots, q_m$, respectively. In general, the function $l : Q \to \{0, 1\}$ represents the *ground-truth*, i.e. the labels assigned by the oracle. Here we identify the label *match* with 1 (or *positive*), and the label *non-match* with 0 (or *negative*).
Let $T \subseteq Q$ be a set of string pairs with known ground-truth and function p which predicts labels for elements in T. Then we denote by $F_1(T, p)$ the F_1-score computed over the ground-truth for T and labels assigned by p for all elements in T. In other words, $F_1(T, p)$ indicates the quality of predictions provided by p over string pairs in T.

3.1 Learning Classifiers

We first describe how, at the end of each iteration, a classification model for string matching is computed. Informally, our classifier is constructed as a string similarity measure $s : Q \to \mathbb{R}$ and a threshold $t \in \mathbb{R}$: if for a pair q of strings we have $s(q) \geq t$, then we predict the label *match*, otherwise *non-match*. We consider a collection of n such similarity measures and corresponding thresholds. Learning is performed after each new query by adjusting thresholds of all measures (Section 3.1.1), and identifying a best performing similarity measure (Section 3.1.2).
In detail, let S be a set of n similarity measures $s_1, \ldots, s_n$. For a fixed iteration number m let $t_{m,1}, \ldots, t_{m,n}$ denote real values used as thresholds. Then (for a specific m) we define n *prediction functions* or *predictors* $p_{m,1}, \ldots, p_{m,n}$ via:

$$p_{m,i}(q) = \begin{cases} 1 & \text{if } s_i(q) \geq t_{m,i} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We denote as p^*_m the best predictor found in iteration m (characterized by a measure and its threshold). If our algorithm terminates at m, p^*_m becomes the overall output of the learning process.

3.1.1 Adjusting thresholds

The following procedure is outlined (for a generic quality metric) as the *straight-forward empirical method* to optimize thresholds in [3]. The values $l(q_1), \ldots, l(q_m)$ of labels queried in m first iterations are used as follows to find the thresholds $t_{m,1}, \ldots, t_{m,n}$. For a fixed similarity measure s_i we find an optimized threshold $t_{m,i}$ by evaluating the respective F_1-score for candidate threshold values t. How we find candidate values for t is described below. Let p be a predic-

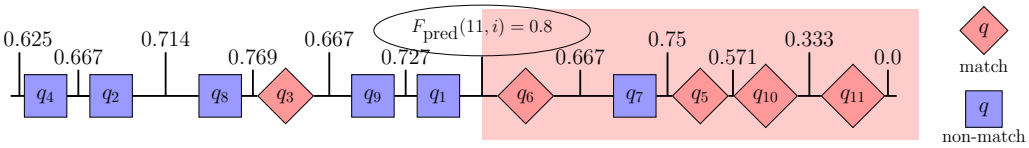

Figure 3: String pairs q_j are aligned according to their similarity value w.r.t to some similarity measure s_i (similarity values are not shown). Vertical lines indicate possible thresholds, labeled with their respective empirical F-measure. The right side of the optimal threshold is highlighted to indicate pairs that are predicted to be matches, the white area (on the left side) non-matches. Diamonds (red) indicate actual matches and squares (blue) indicate actual non-matches. Pairs without revealed labels are not shown for clarity.

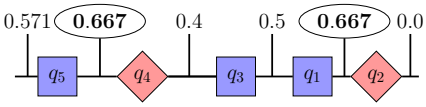

Figure 4: Maximum F_1-score values for a measure s_i may be achieved by multiple candidate thresholds

tor defined by Equation (1) for s_i and a fixed t. We compute the *empirical F-measure* of p, denoted as $F_1(Q_m, p)$, based on the known ground-truth for $Q_m = \{q_1, \ldots, q_m\}$. Among all candidate thresholds for s_i, we consider only those with the highest empirical F-measure. From these best threshold candidates we select only one as a threshold $t_{m,i}$ as described in Section 3.1.2.

The candidate thresholds for s_i are found as follows. We first compute the values $s_i(q_1), \ldots, s_i(q_m)$, that intersect the real number line in at most $m + 1$ intervals. The candidate thresholds are then the arithmetic means of each (closed) interval (and additionally the minimal value s_i and the maximal value plus a small constant $s_i + \epsilon$). Note that any point within one interval yields the same empirical F-measure for the corresponding candidate predictor.

Figure 3 illustrates this. There are 11 labeled pairs $q_1, \ldots, q_{11}$ that are aligned in a linear ordering, according to their similarity value as measured by s_i (increasing from left to right, not shown). The candidates for a new threshold $t_{11,i}$ for similarity measure s_i are the "middles" of each of the 12 intervals defined by $s_i(q_1), \ldots, s_i(q_{11})$. The vertical lines indicate these candidate thresholds; the corresponding empirical F-measures are shown in the text labels.

The interval defined by $s_i(q_1)$ and $s_i(q_6)$ gives rise to thresholds with empirical F-measure of 0.8. This is the best value, and so we use as a threshold $t_{m,i}$ for s_i the middle of this interval. All string pairs q with $s_i(q) \geq t_{m,i}$ are now classified by the resulting prediction function $p_{11,i}$ as matches (shaded or red area). Obviously, this prediction function errs on some of the queried pairs, namely q_3 (false negative) and q_7 (false positive).

3.1.2 Ranking of candidate predictors

There are two types of ambiguity when selecting p_m^*:
(i) for each measure s_i several candidate thresholds might lead to the same empirical F-measure, and (ii) after all n thresholds are fixed, several predictors $p_{m,i}(q)$ $(i = 1, \ldots, n)$ might achieve the highest value $F^*_{\text{pred}}(m)$ of the empirical F-measure.

Ambiguities of type (i) are illustrated in Figures 3 (thresholds of the same interval are equivalent) and 4 (there may even be different equivalent intervals). We resolve those by selecting as the threshold $t_{m,i}$ for s_i the middle of the "rightmost" interval (i.e., containing the highest similarity values) among all these empirically optimal choices (e.g., between q_1 and q_2 in Figure 4). This choice has little impact on performance of the algorithm.

Ambiguity in case (ii) occurs since many predictors can have the same F-measure in a given iteration. As shown in the evaluation (Section 4), this becomes significantly less pronounced in later iterations. We devise here a simple secondary heuristic ranking to pick the best from these highest scoring predictors.

For each candidate predictor $p_{m,i}$ we inspect the interval containing the selected threshold for $p_{m,i}$: the lower the number of pairs with *unrevealed* labels in that interval, the higher the secondary ranking for the predictor. Our heuristic intuition is that the spread of these noisy pairs are expected to be more concentrated in good performing hypotheses, which leave few intermediate pairs unqueried.

3.2 Aggregated Prediction Function

In the following, we introduce an *aggregated prediction function* p_m that is computed in each iteration m. This function p_m is required by the query strategy described in Section 3.3. We describe how this is done in Sections 3.2.1 and 3.2.2. In Section 3.3, we explain how we define the most uncertain pair based on p.

In this paper, we use the term *hypothesis* as defined in [18]. A hypothesis is a classifier (or a configuration that is sufficient to identify one), that *explains* the data by generalizing the ground-truth. In our case, a hypothesis $h = (s, t)$ is determined by a specific similarity measure s from $S = \{s_1, \ldots, s_n\}$ and a corresponding threshold $t \in \mathbb{R}$. Consequently, the hypothesis space is $H = S \times \mathbb{R}$. Restating Equation (1), each hypothesis $h = (s, t)$ is assigned a prediction function defined by $p_h(q) = 1 :\Leftrightarrow s(q) \geq t$. We will write $h_{m,i}$ as shorthand for $(s_i, t_{m,i})$ and $p_{m,i}$ for $p_{h_{m,i}}$. We also define as $F_{\text{pred}}(m, i) := F_1\{Q_m, p_{m,i}\}$ the *empirical* F_1-score for hypothesis $h_{m,i}$ w.r.t. observed labels only (recall that Q_m is the set of already labeled string pairs).

3.2.1 Clustering of Similarity Measures

We account for redundancy among similarity measures by clustering measures that are mutually similar. It is important to notice about the hypotheses that they are not independent of one another. Some similarity measures may not even be defined similarly, but produce a very highly correlated ordering of pairs. In order to combine their predictions in a meaningful way, it is important to account for this redundancy. Otherwise some similarity measures might

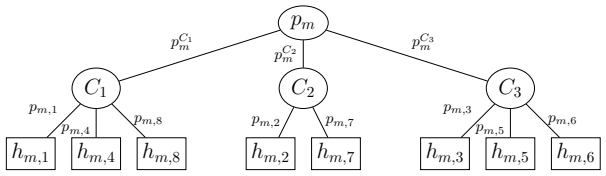

Figure 5: Structure of the aggregated prediction function p_m (a fictional example with $n = 8$)

Symbol	Definition
$F_{\text{pred}}(m,i) :=$ $F_1\{Q_m, p_{m,i}\}$	**empirical** F_1; the F_1 score for $h_{m,i}$ w.r.t. observed labels only
$C_1, \ldots, C_k \subseteq S$	k **clusters** of similarity measures
$p_{m,i} : Q \to \{0,1\}$	**prediction function** of hypothesis $h_{m,i}$
$p_m^{C_j} : Q \to [0,1]$	(intermediate) prediction function associated to cluster C_j
$p_m : Q \to [0,1]$	**aggregated prediction function**
$w_{m,i}, w_m^{C_j}$	**weights** used in the aggregation
$p_m^{\text{target}} := 1 - r_m$	**target prediction** value

Table 2: Symbols used for the query strategy

be overrepresented and dominate the voting.
We represent similarity measures by the vector of similarity values, indexed by all considered string pairs. The redundancy among two similarity measures is then computed by their mutual Pearson correlation coefficient. Calculating the matrix of correlations between all pairs of similarity measures yields a weighted undirected graph. The nodes correspond to similarity measures and the weighted edges to the pairwise Pearson correlation. Clusters $C_1, \ldots, C_k$ in this graph correspond to sets of mutually similar similarity measures. We apply a graph clustering algorithm [16] that maximizes the so-called *modularity* of a graph clustering to achieve this. We used the implementation Linloglayout [1]. The clustering is computed once in the beginning and remains the same over the course of all iterations. In our experiments with a set of 24 similarity measures, k ranged from 2 to 5 and the cluster sizes from 1 to 12.

3.2.2 Aggregating

In the following, we explain how we combine in each iteration m the individual prediction functions $p_{m,i}$ to an overall prediction function p_m.
We do this in two steps, which we illustrate in Figure 5. The first step combines the prediction functions of hypotheses whose similarity measures are in the same cluster C to an intermediate prediction function p_m^C. The second step combines all predictions functions p_m^C to the overall prediction function p_m.
We estimate the predictive power of a prediction function $p_{m,i}$ with the F-measure on the seen labels. We assign weights $w_{m,i} = F_{\text{pred}}(m,i)$ to favour promising predictions over poor predictions. The prediction functions p_m^C are defined as follows:

$$p_m^C(q) \coloneqq \left(\sum_{s_i \in C} w_{m,i}\right)^{-1} \cdot \left(\sum_{s_i \in C} p_{m,i}(q) \cdot w_{m,i}\right)$$

These prediction functions are then combined to the overall prediction function p_m:

$$p_m(q) \coloneqq \left(\sum_{j=1}^{k} w_m^{C_j}\right)^{-1} \cdot \left(\sum_{j=1}^{k} p_m^{C_j}(q) \cdot w_m^{C_j}\right),$$

where $w_m^{C_j} = \frac{1}{|C|} \sum_{s_i \in C} F_{\text{pred}}(m,i)$.
The weights w_m^C for the functions p_m^C reflect how promising the hypotheses whose similarity measures belong to cluster C are on average.
The expression for $p_m(q)$ can be expanded and then simplified to the following:

$$p_m(q) = \left(\sum_{j=1}^{k} \sum_{s_i \in C_j} \frac{1}{|C_j|} w_{m,i}\right)^{-1} \cdot \left(\sum_{i=1}^{n} p_{m,i}(q) \cdot w_{m,i}\right)$$

3.3 Query Strategy

In terms of [18], the strategy can be described as an instance of uncertainty sampling. That is, the paradigm that is based on a prediction function for unseen labels and assigns a measure of certainty to each prediction. The basic insight is that there is little value in querying the instances whose labels can be predicted with high certainty. Conversely, the most gain for the learning progress can be expected from querying the most uncertain instances.
The prediction function $p_m : Q \to [0,1]$ calculates a real value for each string pair. We now define our notion of uncertainty based on this function.
A clear prediction of $p_m(q) = 1.0$ implies that each hypothesis predicts the query to be a match. Conversely, $p_m(q) = 0.0$ means that each hypothesis predicts a non-match.
When the overall prediction $p_m(q)$ is equal to 0.5, the label of q is highly uncertain. This prediction can only occur if sufficiently highly-weighted or at least many hypotheses predict either label. This means that no matter which label will turn out to be true, a significant part of the hypotheses will see its prediction disproved. I.e., significant in the sense of a combination of the individual confidences. The first idea is hence to define the next query as

$$\text{argmin}_{q \in Q \setminus Q_{m-1}} |0.5 - p_m(q)|$$

Yet, to achieve fast convergence, we have to correct this notion of uncertainty to account for the overall prediction $p_m(q)$ which can be biased in early iterations. We will therefore replace the value 0.5 by a *target prediction*, denoted by $p_m^{\text{target}} \in [0,1]$.
Since the thresholds have to be set to some initial setting, they are bound to be inappropriate in the beginning. Assume that the thresholds in iteration m are such that the functions $p_{m,i}$ are collectively biased towards predicting matches. That is, each function $p_{m,i}$ has a high false positive rate w.r.t. the whole ground-truth. Then the set of pairs that get a committee prediction close to 0.5 will consist of more non-matches than matches. So the label of such a pair is actually not most uncertain, when the hypotheses are collectively biased. The target prediction p_m^{target} for iteration m must be higher than 0.5 if the hypotheses are generally biased towards predicting matches. An analogous argument applies in the reverse situation.

We measure the current bias of the committee by calculating the ratio of match labels among the revealed labels. We define the target prediction p_m^{target} as the complement, or ratio of non-match labels among the seen labels:

$$p_m^{\text{target}} := 1 - r_m = \frac{1}{m-1} \sum_{n=1}^{m-1} 1 - l(q_n)$$

It will yield a value higher than 0.5, if the committee was biased towards matches in the past, as desired. This way the target prediction self-regulates until the thresholds adjust to the respective regions where the labels are truly uncertain. The next query is hence defined as follows:

$$q_m := \text{argmin}_{q \in Q \setminus Q_{m-1}} \left| p_m^{\text{target}} - p_m(q) \right|.$$

Note that the query q_m minimizing the expression above may not be uniquely defined. Therefore, we devised a secondary ranking of the pairs to further distinguish the pairs. This ensures that informative queries are done even when the prediction p_m is not (yet) well prepared to decide this. We measure the variance of ranks w.r.t. the different linear orderings that the similarity measures give. It can be calculated by unsupervised data. This is defined as follows.
For each pair, calculate the rank that it is assigned by each similarity measure s. That is, the pair q with the highest similarity value $s(q)$ will get rank 0 w.r.t. s, the pair with the second highest will get rank 1, and so on. Then calculate for each pair the variance with respect to its similarity rank across all similarity measures. The higher this variance is, the higher the pair be will appear in the secondary ranking for querying.

3.4 Stopping Criterion

In an approach using Active Learning it is useful to have a stopping criterion, i.e., a function indicating when the querying process can be terminated. It is important not to stop too early, when the solution is still improving. On the other hand, the algorithm should not issue additional queries after the solution is stable.
The stopping criterion can be designed independently of the query strategy and the ranking of candidate predictors. We tried a lot of other intuitive heuristics, but none seemed to be more successful than the one we will introduce now. The main problem in devising a stopping criterion is the scarcity of information that is inherent to Active Learning.
A trivial stopping criterion is a threshold on the number of queried labels. However, we noticed that the algorithm "progresses" at different speeds for different data sets. We measured the progress of the algorithm in terms of the *true F-measure* $F_{\text{true}}(m) := F(Q, p_m^*)$ of the currently best ranking prediction function p_m^* (for the ranking criterion, refer to Section 3.1.2). The true F-measure uses all hidden labels and can hence not be used by the algorithm.
However, (at least for the considered data sets) another measure seems to capture this progress well and does not require the hidden ground-truth. This measure is the total number of changes in thresholds that have occurred since the beginning of the algorithm:

$$TC_m := \sum_{i=1}^{n} \sum_{j=0}^{m-1} 1 - \delta(t_{j,i}, t_{j+1,i}),$$

where δ is equal to 1 if its arguments are equal and 0 otherwise. A stopping criterion can be devised as a condition that TC_m has exceeded a certain threshold. In Section 4.3 we compare different thresholds for TC_m in context of our data sets.

4. EVALUATION

In this section we evaluate our approach under several aspects. These include user labeling effort vs. accuracy (Section 4.3), convergence behavior (Section 4.4), impact of the stopping criterion (Section 4.5), and other aspects, such as impact of the predictor ranking (Section 4.6).

4.1 Experimental Setup

Data Sets. We use for the evaluation 17 data sets summarized in Table 3. Four data sets are created by us (marked with $^+$ in Table 3). To obtain the ground-truth we pursued a similar approach as [8], with a final manual review. We have also found and corrected errors in the ground-truth in 3 existing data sets (*business*, *animal*, and *bird2*). Details of these errors as well as all data sets can be downloaded from `http://pvs.ifi.uni-heidelberg.de/team/lb/`.

Similarity measures. In our experiments we use $n = 24$ similarity measures $s_1, \ldots, s_{24}$ taken from the library SecondString [2]. Some of these measures depend only on the two input strings, while others take into account frequencies of substrings across the whole data set. Our approach treats the similarity measures as black boxes.

Other parameters. Our experiments have shown that several parameters described in Section 3 do not have a notable impact on the behaviour of the algorithm. These include initial values of thresholds, and selecting the actual threshold value among several candidates with identical F_1-scores. Therefore, we report only the results obtained via implementation choices described above.

4.2 Evaluation metrics

We use in the following some symbols defined in Table 1. Recall that for a fixed iteration number m, the (intermediate) result of the learning process is the best predictor p_m^* (if m is the final iteration, p_m^* is the final result of the process). The following two metrics evaluate the convergence and accuracy of p_m^*. The *true* F_1, denoted as $F_{\text{true}}(m)$, is the F_1-score of the best predictor p_m^* taking into account the labels of *all* string pairs in Q (i.e., the complete ground-truth):

$$F_{\text{true}}(m) = F_1(Q, p_m^*).$$

This metric estimates the generalization capability of p_m^* since it takes into account all labels of the data set, not only the queried ones.
The *maximal* F_1 (symbol $F_{\max}$) is the F_1-score of the best possible prediction function across the hypothesis space $H = S \times \mathbb{R}$ w.r.t. the whole ground-truth, i.e.,

$$F_{\max} = \max_{h \in H} F_1(Q, p_h).$$

The *match ratio* in iteration m (denoted r_m) is the ratio of match labels versus non-match labels among the observed pairs. Finally, TC_m is the *total number of threshold changes*, which is the cumulative number of times when a threshold has changed for any similarity measure in any iteration.

Dataset name	domain	src 1	src 2	Original		Reduced		
				pairs	match	pairs	match	matches / pairs
string matching problems								
bird3[8]	common+scientific animal names	23	15	345	15	25	*14*	56.00%
USPresidents$^+$	personal names	43	43	1,849	43	173	43	24.86%
ucdFolks[14]	personal names	45	45	2,025	45	184	45	24.46%
DBconferences$^+$	names of database conferences	54	54	2,963	54	2441	54	2.21%
bird1[8]	common+scientific animal names	317	20	6,340	19	672	19	2.83%
faoMembers$^+$	country names	194	194	37,636	194	2,633	*192*	7.29%
bird2[8]*	common animal names	914	68	62,152	64	4,089	64	1.57%
game[8]	names of computer games	798	105	83,790	41	4,276	41	0.96%
bird4[8]	common+scientific animal names	564	155	87,420	155	11,297	155	1.37%
park[8]	names of (national) parks	393	258	101,394	252	6,767	*250*	3.69%
census[9]	synthetic peronal names+addresses	449	392	176,008	329	18,438	*326*	1.77%
fodorZagrat[20]	restaurant names+addr+phone+style	532	331	176,092	114	73,657	*112*	0.15%
nobelLaureates$^+$	personal names	839	839	703,921	839	27,011	*831*	3.08%
business[8]*	company names	1162	962	1,117,844	310	502,316	*309*	0.06%
animal[8]*	common animal names	4719	817	3,855,423	178	93,661	*175*	0.19%
string deduplication problems (single source)								
UVA[14]	institute names	116		6,670	280	2,932	*272*	9.28%
coraATDV[12]	publication references	956		456,490	7,766	453,987	7,766	1.71%

Table 3: Benchmark data sets (ordered by number of pairs) for string matching (upper part) and deduplication problems (lower part). Column "Reduced" shows effects of the indexing step. * indicates that ground-truth has been corrected; italic number of matches means false negatives due to indexing. $^+$indicates new data sets. Summed entries in column "type" indicate concatenated strings of different types.

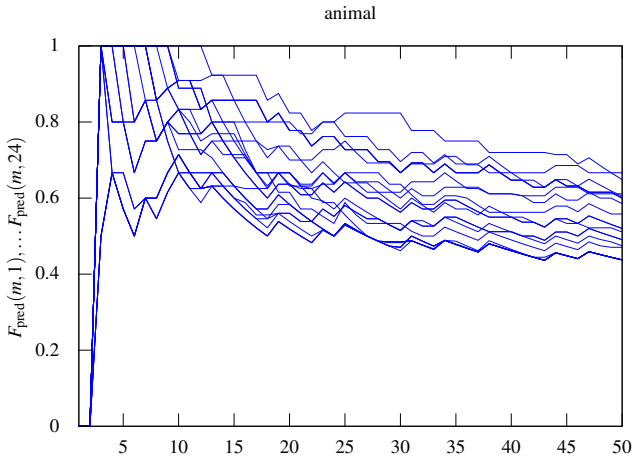

Figure 6: The empirical F_1-scores decrease with higher number of iterations.

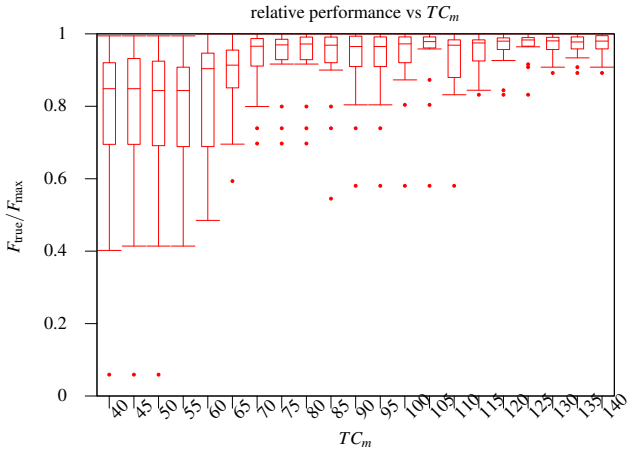

Figure 7: The ratio of true F_1-scores of the best predictor ($F_{\mathbf{true}}(m)$) to $F_{\mathbf{max}}$ vs. total number of threshold changes TC_m (each box/whisker plot shows distribution over all data sets).

4.3 User effort and predictor accuracy

Table 4 reports results for all data sets after learning prediction models with two different stopping criteria A: $TC_m \geq 75$ and B: $TC_m \geq 130$. For criterion A, the average number of queries (or iterations) is 7 and never exceeds 10, which shows that the labeling effort for the user is very low. At the same time, the ratios $F_{\mathbf{true}}/F_{\mathbf{max}}$ of solution quality (i.e. F_{true}) to maximum achievable quality (F_{max}, specific to a data set) are high. This indicates that the final matching prediction functions have been learned well.

In case of stopping criterion B, the ratios $F_{\mathbf{true}}/F_{\mathbf{max}}$ are in general higher, but not to a large degree. Also here the average number of iterations until stop is relatively low (around 15) which implies an acceptable labeling effort. Summarizing, we conclude that Active Learning performs very well, and allows minimizing user effort while achieving accurate prediction models.

4.4 Convergence behavior

Figure 8 gives more insight into convergence behavior of some selected representative data sets. While *animal* and *business* (top row) illustrate benign changes of $F_{\text{true}}(m)$ (the

		A: stop at $TC_m \geq 75$					B: stop at $TC_m \geq 130$				
Data set	$F_{\max}$	final m	s^*	t_m^*	F_{true}	$\frac{F_{\mathbf{true}}}{F_{\mathbf{max}}}$	final m	s^*	t_m^*	F_{true}	$\frac{F_{\mathbf{true}}}{F_{\mathbf{max}}}$
bird3	1.000	5	L2_JW	0.923	1.000	100.00%	15	TL	0.598	1.000	100.00%
USPres.	0.953	6	ST	0.403	0.925	96.98%	14	TL	0.778	0.943	98.85%
ucdFolks	1.000	5	ASD	0.342	0.917	91.67%	12	TL	0.643	0.989	98.90%
DBconf.	0.874	8	ME	0.779	0.851	97.45%	14	ME	0.840	0.845	96.75%
bird1	0.947	7	TFIDF	0.513	0.947	100.00%	16	TL	0.766	0.944	99.69%
faoMembers	0.961	9	L2_L	-0.300	0.958	99.70%	21	L2_ME	0.949	0.897	93.37%
bird2	0.944	7	JWT	0.982	0.918	97.25%	12	JC	0.633	0.929	98.42%
game	0.846	10	L2_JO	0.886	0.805	95.16%	20	L2_JO	0.927	0.827	97.70%
bird4	0.980	6	TL	0.615	0.951	96.96%	12	L2_JW	0.900	0.955	97.45%
park	0.970	7	SL	0.675	0.913	94.08%	15	ST	0.571	0.921	94.94%
census	0.899	6	A	0.609	0.665	73.94%	12	TL	0.816	0.817	90.80%
fodorZagrat	0.978	8	TL	0.751	0.959	98.09%	17	JWT	0.644	0.968	99.05%
nobelLaur.	0.989	7	JWT	0.399	0.790	79.93%	15	ST	0.443	0.882	89.21%
business	0.971	8	ASD	0.685	0.960	98.93%	16	ASD	0.663	0.964	99.30%
animal	0.926	8	M	0.495	0.891	96.24%	18	M	0.444	0.893	96.43%
UVA	0.907	7	JO	0.858	0.633	69.72%	13	TL	0.721	0.899	99.04%
coraATDV	0.800	6	ME	0.551	0.648	81.07%	15	L2_L	-0.675	0.736	92.07%
average		**7.06**				**92.29%**	**15.12**				**96.59%**

Table 4: Results after model learning with stopping criteria A: $TC_m \geq 75$ and B: $TC_m \geq 130$. Final m is the number of iterations, s^* is the best final similarity measure, and t_m^* the corresponding final threshold. $F_{\mathbf{true}}/F_{\mathbf{max}}$ gives the ratio of solution quality (i.e. $F_{\mathbf{true}}$) to maximum achievable quality ($F_{\mathbf{max}}$). Similarity measures: L2_*=*Level2, JW=*JaroWinkler*, ST=*SourcedTFIDF*, ASD=*AveragedStringDistanceLearner*, ME=*MongeElkan*, L=*Levenstein*, JWT=*JaroWinklerTFIDF*, JO=*Jaro*, TL=*TagLink*, SL=*ScaledLevenstein*, M=*Mixture*, JC=*Jaccard*. Note that thresholds may be negative.**

F_1-score of the best predictor, specific to iteration number m), *UVA* and *nobelLaureates* (bottom row) show less desirable behavior of $F_{\text{true}}(m)$. We also note that in the first iterations highest-ranked similarity measure changes frequently (indicated by vertical lines), while after 7 iterations the measure stabilizes. This is less pronounced for the "bad" cases *UVA* and *nobelLaureates*.

Figure 6 shows that the empirical F_1-scores $F_{\text{pred}}(m, i)$ decrease with growing iteration number m for all measures. This can be attributed to the fact that the uncovered labels come only from the "noisy" area that contains all hard-to-separate pairs. This also explains why $F^*_{\text{pred}}(m)$ decreases below the actual performance $F_{\text{true}}(m)$. So, the empirical F_1-scores do not approximate the true performance, but are sufficient to compare the quality of current predictors.

Note that the output prediction function p^*_m is chosen from the p_h (with $h \in H$). The aggregated prediction function p_m will not be output. It is only used to define the next query to the user. In fact, p_m performs significantly worse than the individual predictor p^*_m chosen in each time step by the algorithm. Our experiments showed that it is very unstable for most data sets (it does not converge) and shows worse F_1-scores most of the time; often significantly. A reason is that even a bad similarity measure s_i can achieve precision of r_m and recall of 1.0 by a small threshold (for $r_m = 0.3$ this equals $w_{m,i} = F_{\text{pred}}(m, i) = 0.462$). So p_m may always be somewhat influenced by inappropriate measures.

4.5 Stopping criterion

The choice of the stopping criterion determines the trade-off between user labeling effort and accuracy. Figure 8 shows that for *animal* and *business* the matching quality $F^*_{\text{pred}}(m)$ increases rapidly in the first few iterations and remains virtually unchanged after that. For *UVA* and *nobelLaureates* more than 10 iterations are needed in order to achieve a stable quality level (even not so in the case of UVA). This refines the findings from Section 4.3: for some data sets, less then 10 iterations are sufficient (i.e., stopping criterion A), while others require more labeling effort. Thus, the choice of a *universal* stopping criterion is hardly possible. Depending on the application, either low user effort (criterion A) or potentially higher accuracy (criterion B) must be preferred. Figure 9 shows on the right y-axis the values of TC_m used in our stopping criterion (i.e. cumulative number of threshold changes over all measures in S). More conclusive is Figure 7: we see that the variance of the true F_1-scores over all data sets decrease for TC_m values above 75, suggesting this as a potential threshold.

4.6 Other aspects

Figure 9 shows (on the left axis) the F_1-score of the best ranked predictor (line with crosses) among all predictors with maximal empirical F_1-scores in iteration m (shaded area). It illustrates the use of a secondary heuristic ranking explained in Section 3.1.2, especially when a lot of competing predictors (with the same best empirical F_1-scores) exist in the early iterations.

Figure 8 also shows that the match ratio r_m is very balanced, while tending towards lower values. Considering the skewed label distribution (see Table 3), this constitutes a very balanced sample. This indicates that the query strategy works well in terms of selecting string pairs with informative labels.

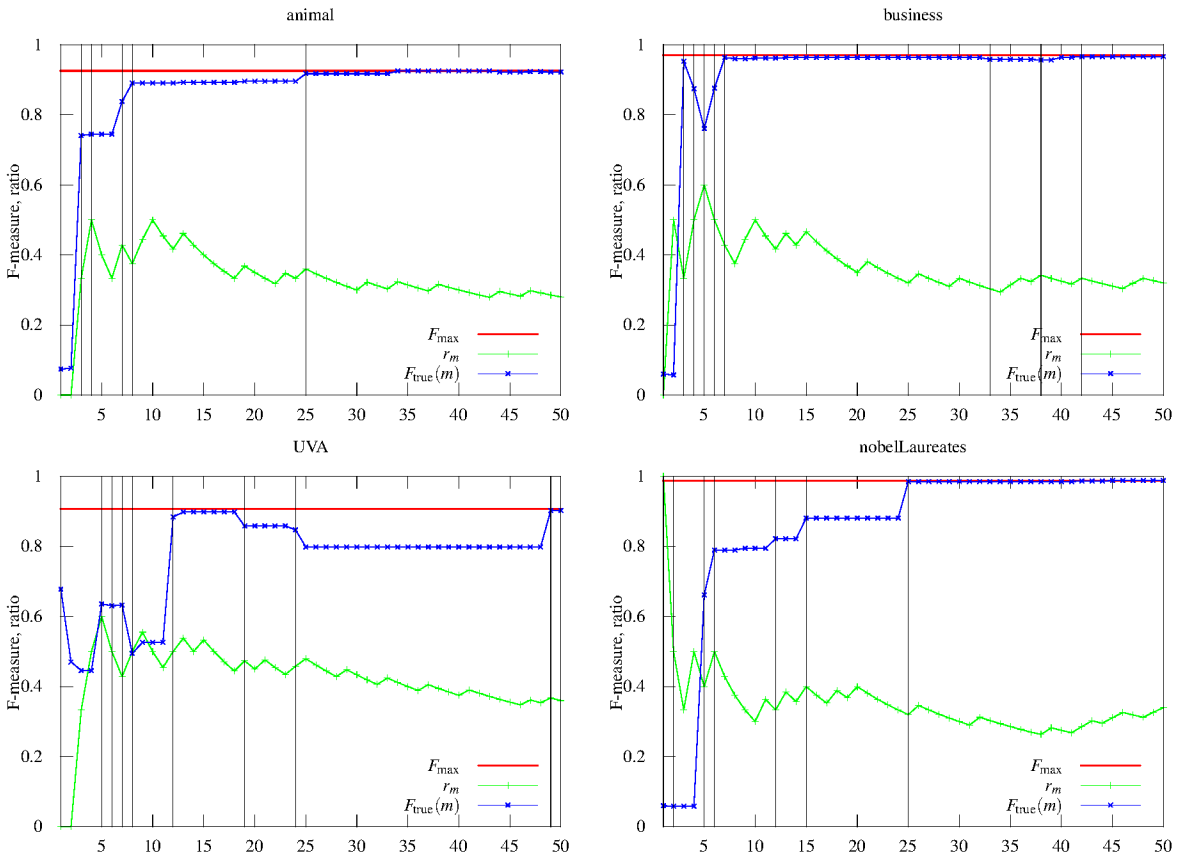

Figure 8: Convergence behavior for exemplary Active Learning runs. The x-axis shows the iteration number m. The values of $F_{\max}$, r_m, and $F^*_{\text{pred}}(m)$ are plotted on the y-axis. Changes in $F^*_{\text{pred}}(m)$ result from adjusting thresholds and switching the best ranked similarity metric (latter changes are indicated by vertical bars).

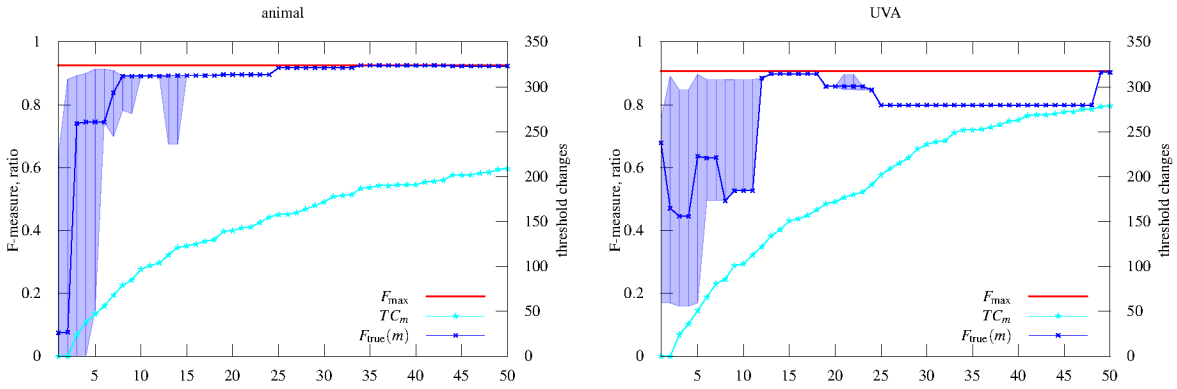

Figure 9: The x-axis shows the iteration number m. The left y-axis shows the F_1-scores, while the right y-axis reports the value of TC_m. The line labeled with $F_{\text{true}}(m)$ shows the true F_1-score of the best ranked predictor according to our ranking. The highlighted interval indicates the minimal and maximal true F_1-score among all predictors with maximal empirical F_1-scores.

5. RELATED WORK

We are aware of two works on optimizing threholds of similarity measures [3, 10]. Both approach this problem in the context of information retrieval. They consider query strings that are directed to a database of documents. The goal is to retrieve a ranked list of documents that contains relevant documents to a given query. A similarity measure is used to compare query strings to documents. A threshold decides, whether a document will be deemed relevant to the query. There can be many relevant documents to a query and the quality is measured in terms of the contingency table, e.g. in accuracy or F_1. Documents are usually very long strings. In [3], the authors describe a system that was implemented

to take part in a benchmark competition for text retrieval. It is trained online on a stream of queries. They introduce *score-distributional threshold optimization*, which uses a statistical model to estimate a threshold. The considered similarity measure is TFIDF with some preprocessing (like stemming and stop word removal). The approach works with any quality metric that is a function of the contingency table, like the F-measure. The paper outlines the *straightforward empirical method* to optimize thresholds w.r.t. a given quality metric, that exhaustively considers all thresholds on the present training data. The authors dismiss this simple approach, because of its drawbacks in their online problem setting. Most computations involved in their method can be updated incrementally. Also, it is able to produce a broad prediction for the threshold with only sparse supervised data. It is unclear, however, how score-distributional threshold optimization performs on only a few samples, since it is only evaluated in a very specific online setting. The approach is only evaluated on TFIDF and does not point out any method to compare several similarity measures.
The authors of [10] use the *straightforward empirical method* w.r.t. optimizing accuracy. Additionally, they formulate a statistical model with a bivariate Gaussian distribution. They show that this model captures the notion of an accuracy-optimal threshold well. Unfortunately, the evaluation of the optimization is not conclusive. Only one data set with artificial edit variations is used, and one similarity measure (Levenstein edit distance). The result suggests, that eventually the threshold arrives in the optimal interval. In the one experiment this happens close to 40 used samples. One sample corresponds to a ranking list which requires 8 labels (*relevant* or *irrelevant* to the query). The authors suggest to use clustering as an unsupervised method to approximate ground-truth to mitigate the labeling effort. The evaluation does not measure how good the intermediately approximated thresholds are, in terms of any quality metric. The sampling method for documents and queries is not described. Finally, a quality metric for similarity measures is introduced (arithmetic mean of accuracy and the size of the output interval of optimized thresholds). This metric is evaluated on eight similarity measures. The results show that this measure preserves the ranking by accuracy. Hence the empirical method for threshold optimization can also be used to compare similarity measures in terms of accuracy.

6. CONCLUSION

We have developed a novel method for finding a similarity measure and an appropriate threshold that work well specifically for given data, in order to solve the string matching problem. It requires no existing ground-truth and can be used by end-users. The experimental evaluation shows that good results can be achieved with only very few iterations. We propose two stopping criteria based on a notion of progress. Their thresholds have been determined empirically. The result of our proposed algorithm can directly be used in important applications like similarity join, record matching and schema matching.

7. REFERENCES

[1] https://code.google.com/p/linloglayout/.

[2] http://secondstring.sourceforge.net/.

[3] A. Arampatzis and A. van Hameran. The score-distributional threshold optimization for adaptive binary classification tasks. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 285–293, 2001.

[4] J. Attenberg and F. Provost. Inactive learning?: difficulties employing active learning in practice. *ACM SIGKDD Explorations Newsletter*, 12(2):36–41, 2011.

[5] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys (CSUR)*, 41(1):1, 2008.

[6] P. Christen. A comparison of personal name matching: Techniques and practical issues. In *ICDM Workshops*, 2006.

[7] P. Christen. *Data Matching*. 2012.

[8] W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, pages 288–321, July 2000.

[9] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In S. Kambhampati and C. A. Knoblock, editors, *Proceedings of IJCAI Workshop on Information Integration on the Web*, 2003.

[10] R. Da Silva, R. Stasiu, V. M. Orengo, and C. A. Heuser. Measuring quality of similarity functions in approximate data matching. *Journal of Informetrics*, 1(1):35–46, 2007.

[11] H. Koepcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, pages 197–210, Feb. 2010.

[12] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000.

[13] D. Menestrina, S. E. Whang, and H. Garcia-Molina. Evaluating entity resolution results. *Proceedings of the VLDB Endowment*, 3(1-2):208–219, 2010.

[14] A. E. Monge, C. Elkan, and others. The Field Matching Problem: Algorithms and Applications. In *KDD*, pages 267–270, 1996.

[15] F. Naumann and M. Herschel. An introduction to duplicate detection. *Synthesis Lectures on Data Management*, 2(1):1–87, 2010.

[16] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 79(2), 2009.

[17] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[18] B. Settles. *Active Learning*. 2012.

[19] M. Stonebraker, I. F. Ilyas, S. Zdonik, G. Beskales, and A. Pagan. Data Curation at Scale: The Data Tamer System. *6th Biennial Conference on Innovative Data Systems Research*, 2013.

[20] S. Tejada, C. A. Knoblock, and S. Minton. Learning Object Identification Rules for Information Integration. *Information Systems*, 2001.

[21] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems (TODS)*, 36(3):15, 2011.