A Strategy for Allowing Meaningful and Comparable Scores in Approximate Matching

Carina F. Dorneles
Carlos A. Heuser
Viviane Moreira Orengo
UFRGS - II
Porto Alegre, Brazil
dorneles@inf.ufrgs.br
heuser@inf.ufrgs.br
vmorengo@inf.ufrgs.br

Altigran S. da Silva
Edleno S. de Moura
UFAM - ICE
Manaus, Brazil
alti@dcc.fua.br
edleno@dcc.fua.br

ABSTRACT

The goal of approximate data matching is to assess whether two distinct data instances represent the same real world object. This is usually achieved through the use of a similarity function, which returns a score that defines how similar two data instances are. If this score surpasses a given threshold, both data instances are considered as representing the same real world object. The score values returned by a similarity function depend on the algorithm that implements the function and have no meaning to the user (apart from the fact that a higher similarity value means that two data instances are more similar). In this paper, we propose that instead of defining the threshold in terms of the scores returned by a similarity function, the user specifies the precision that is expected from the matching process. Precision is a well known quality measure and has a clear interpretation from the user's point of view. Our approach relies on mapping between similarity scores and precision values based on a training data set. Experimental results show the training may be executed against a representative data set, and reused for other databases from the same domain.

Categories and Subject Descriptors

H.2.m [**Database Management**]: Miscellaneous—*Approximate matching*

General Terms

Experimentation, Measurement

Keywords

Similarity querying, data integration, data cleaning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'07, November 6–8, 2007, Lisboa, Portugal.
Copyright 2007 ACM 978-1-59593-803-9/07/0011 ...$5.00.

1. INTRODUCTION

The problem of *approximate matching* appears in several contexts, such as similarity querying, data integration and data cleaning. In *similarity querying*, the problem is to identify database records representing the same real world object as the one specified by the user in the query. In *data integration*, the problem is to determine records originating from different data sources that represent the same real world object. The problem in *data cleaning* is detecting and correcting errors in data received from external sources at a data warehouse. Such data may contain errors, e.g., spelling mistakes, inconsistent conventions across data sources, missing fields, that must be solved prior to the storage of the data into the datawarehouse. Such situations are quite common in modern databases, particularly in cases where they receive data and queries from disparate sources.

Approximate matching usually relies on the use of a *similarity function* $f(a_1, a_2) \mapsto s$ that assigns a score s to each pair of data values a_1 and a_2. Higher scores mean higher similarity. Two objects are considered to be similar, i.e., are considered to represent the same real world object, if the similarity score s surpasses a pre-defined *threshold* value.

However, the choice of a threshold value is a difficult matter. The scores returned by a function depend on the internal details of the algorithm that implements the function and usually have no meaning to the user, apart from the fact that a higher value means that two objects are more similar. As a result, one can hardly expect the threshold to be accurately provided by the user. Most likely, the threshold must be somehow predefined by an estimation or learning process. Moreover, as the score values returned by different functions have different value distributions, the quality of the result (measured in terms of recall and precision) may vary from one function to another when a specific threshold is considered. The distribution may even vary when the same function is applied to two different data sets with different distribution of values. This means that a threshold value that has been predefined for a specific function may not be adequate to another.

Our approach is based on a training phase during which precision is estimated for each function and for several different thresholds. The result of this training is a table for each function that maps each of its similarity scores into

an adjusted score. For the process of matching, the user just specifies a *precision threshold* (the precision that is expected from the matching process). The system then sets the threshold of each function to a value that corresponds to the estimated precision for this function, according to the results obtained in the training phase.

The success of our approach relies on the observation that, as shown in the experiments, it is possible to have a good estimation of precision values with a small and acceptable training effort. The experiments show that instead of executing the training process directly over the actual data set to be processed, one may train over a small set of representative data instances collected from various sources. This means that the results of the training process for a specific attribute domain (say person names) and for a specific similarity function (say Soundex) may be employed for different databases that contain attributes from the same domain without the need for retraining for each database.

The remainder of the paper is organized as follows. In Section 2, we further motivate our work through an example and in Section 3, we discuss related work that deal with the problem of approximate matching. In Section 4, we describe how the score computed by a similarity function may be mapped into a meaningful similarity score, and in Section 5 we present a data match operator that employs this meaningful score. In Section 6, we discuss a thorough empirical study on real data sets, and in Section 7 we present our conclusions and directions for future work.

2. MOTIVATING EXAMPLE

The recent literature is abundant in similarity functions for comparing textual values like person's names, institutions, dates, etc. [20, 7, 22]. Generally, such functions are imperfect, i.e., they may result in false positives and false negatives. Thus, measures for evaluating the quality of similarity functions are needed, such as the classical IR measures *recall* and *precision* [1]. The quality of a matching process depends not only on the ability of the similarity function in separating relevant from irrelevant values, but also on the threshold that is chosen.

Figure 1 presents two graphs obtained by executing several matches using Jaro-Winkler and n-grams similarity functions over a set of strings containing respectively journal titles and movie titles. Each graph plots the mean precision obtained at a specific similarity score for the several matches.

We can observe that the precision at a specific score (e.g., 0.5) may vary between both similarity functions and data sets. This example illustrates two open problems in approximate matching: (1) the values provided by a similarity function do not have a clear meaning to the users, and (2) these values are not comparable when the results of different similarity functions are combined (e.g., in tuple matching).

Understanding the meaning of the scores is essential to allow users to adjust the degree of similarity expected in each approximate matching process. In general, only users who have previous knowledge about the employed similarity function would be able to understand the meaning of the resulting scores and take advantage of this knowledge when performing queries or matches.

Furthermore, the scores of each function are not comparable. The fact that distinct similarity functions have distinct distributions of score values leads to problems in tuple matching, where the results of distinct similarity functions on distinct attributes must somehow be combined.

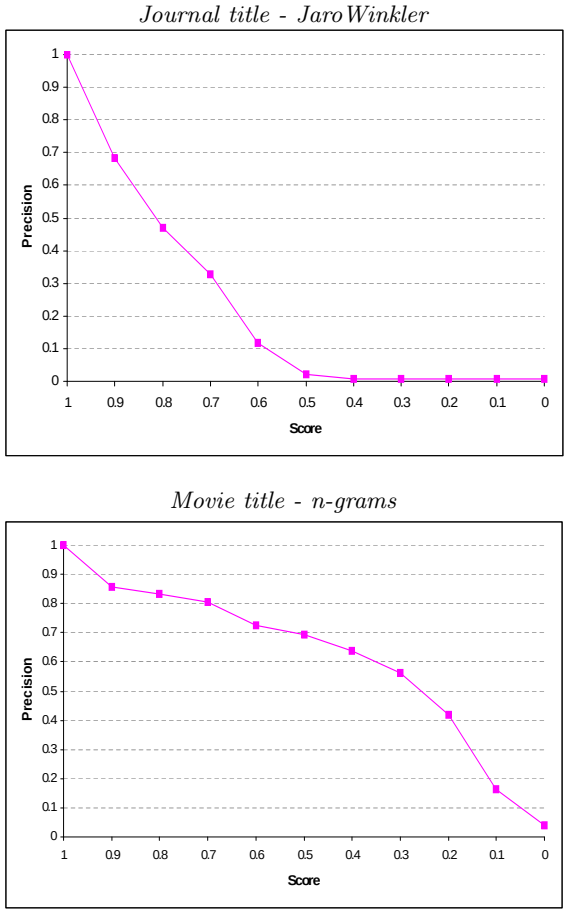

Figure 1: Precision at various similarity scores

3. RELATED WORK

In the context of databases, approximate matching constitutes a core operation to address problems related to flexible query processing, record linkage and data cleaning.

In flexible query processing the goal is to provide suitable answers to ill-specified queries in which users have a limited understanding of the data or of the schema, and when specific personal preferences have to be fulfilled. In this case, instead of requiring values in tuples to match exactly the arguments in the queries, approximate matching is used [18, 9, 3, 16]. For this, similarity functions based on characters (e.g., Levenshtein distance, n-gram distances), tokens (e.g., Jaccard distance, TF/ID), phonetics (e.g., soundex distance) or semantics (e.g., ontological distance) are used to match values to arguments and to give a similarity score to evaluate the match. In particular, approximate matching has been applied in the context of the join operator [13, 15]. The degree of similarity between query arguments and database values is usually bound by a user specified threshold. Since a number of different similarity functions might be used in the same query, e.g., when several query arguments are used, in general specifying a proper threshold for each argument is not an easy task.

Approximate matching techniques have been extensively used for integrating data from distinct data sources for many years, since the seminal work on the so called *record linkage* problem [12]. In this work, the similarity between records of two distinct files is given by the sum of the similarities scores between pairs of common attributes (fields) of these files. Lately, as distinct similarity functions were considered for distinct attribute types and domains, this simple strategy had to be revised. Therefore, recent works have proposed distinct strategies, most of them based on machine learning techniques [22, 2, 10, 19].

In [2] the authors experiment on three distinct strategies for evaluating the similarity between records: (1) comparing the two records as a whole, as if they were a single field/attribute; (2) comparing each field and averaging the resulting similarity scores, which is essentially what is proposed in [12]; and (3) using a feature vector to represent the fields and train a binary support vector machine classifier using this vector. Distinctly from what we propose in the present paper, each experiment with the three strategies involves a single, fixed similarity function. Recognizing that using a single fixed function for all the fields is to restrictive, the authors also propose an adaptive edit distance function which can be learned to particular textual data types. This adaptive edit distance function is similar to the one proposed in [19]. In our work, we generalize this approach by allowing distinct similarity functions of any kind to be used, and not only edit distance functions over textual attributes. Notice, however, that such an approach can be combined with the approach we here propose. Learning is also extensively used in *Active Atlas* [22], a system whose main goal is to learn rules for mapping tuples (referred to as objects) from two distinct relations (referred to as sources) in order to establish an identity between them.

Data received at a data warehouse from external sources usually contain errors. For dealing with this, several work in literature have proposed the use of *data cleaning* techniques to detect and eliminate these errors, thus improving the quality of the data warehouse [5, 14]. The existing techniques may also include detecting potentially duplicated data. In general, a common technique for accomplishing data cleaning is to validate the incoming tuples against reference relations. These relations can be internal to the data warehouse, e.g., one of the relations that already exists, or external, e.g., a zip code table from the postal department. In [5] the authors propose a fuzzy match algorithm for data cleaning. Given a similarity function and an input tuple, the goal is to return the reference tuple which is closest to the input tuple. In [14], the approach is to generate individual rankings for each attribute present in a query statement according to a specific similarity metric and then to combine the individual rankings in order to obtain a global one. This global ranking is such that the distance among the individual rankings is minimized.

The fact that different similarity functions have different distributions of score values leads to problems in tuple matching, where the results of different similarity functions on different attributes must somehow be combined. This problem has been investigated in previous work [22, 14]. Solutions such as normalizing the similarity scores to values in a predefined range have been proposed. These normalizations may help, but still, the score results of each function may have different meanings.

4. PRECISION AS A MEANINGFUL SIMILARITY SCORE

In this section we present a detailed discussion of our proposal for mapping disparate scores generated by distinct similarity functions into a single meaningful score. The main idea consists in obtaining, for the score values provided by a specific similarity function, corresponding score values that are meaningful to users and that can be compared with values provided by another function. This correspondence is drawn from an estimation of the quality of the scores provided by each original function. For this estimation, we use the *precision* measure, a well known IR measure for evaluating the quality of similarity functions [1].

Consider a function f that calculates the similarity between two values a_1 and a_2 of a given attribute A, such that $f(a_1, a_2) \mapsto s$ and $0 \leq s \leq 1$. We call any value returned by f a *ground score*. This notion is formalized in Definition 1.

Definition 1. (Ground Score) *Let D_A be a domain. Let $f : D_A \times D_A \longrightarrow \mathcal{R}_S \subseteq [0, 1]$ be a function that evaluates the similarity between a pair of D_A values and returns a value between $[0, 1]$. The function f is called a* ground similarity function *or simply* ground function *and any value $g \in \mathcal{R}_S$ is called a* ground score.

Now, consider a value q of a given attribute. Using a ground function, it is possible to rank a set of values of this attribute according to the similarity between each value from the set and q.

Definition 2. (Ground Ranking) *Let $A \subseteq D_A$ be an attribute whose domain is D_A and let f be a ground function (Definition 1). A permutation $R(q) = \{a_1, a_2, \ldots, a_n\}$ of A is called a* ground ranking according to $q \in A$ *iff $f(q, a_k) \geq f(q, a_{k+1})$ for $k = 1, \ldots, n-1$.*

Intuitively, we may say that function f estimates the belief that the two values being compared are semantically equal or, in other terms, the belief that the two values being compared are distinct representations of the same object. Assuming that f is consistent, we know that the closer the ground score is to 1, the greater is the chance that the two values represent the same object. On the other hand, the closer the ground score is to 0, the smaller is the chance. Ideally, ground scores greater than zero would only be associated with values that represent the same object. However, since similarity functions are in general imprecise, there is no way of telling how close to 1 the ground score has to be for two attribute values to be considered as representing the same real world object. This issue has to do with the *suitability* or the *quality* of the function in identifying distinct representations of an object in the domain of the attribute.

The accuracy of each function in identifying whether or not two values represent the same real world object can be estimated by observing the scores given by such function to an attribute for a small set of training values. Following a fairly standard procedure adopted in the IR area, it is possible to use this training to estimate the precision achieved by each function for each attribute and each score level. This procedure consists in, for each value q of the training set, labeling each a_i in the ranking $R(q) = \{a_1, a_2, \ldots, a_n\}$ as "relevant" (i.e., "representing the same real world object as the one represented by q") or as "irrelevant" (i.e., "representing a real world object that is distinct from the one represented

Pos.	Title	Ground Score	Precision
1	The Green Mile	1	1
2	Green Mile, The	0.606060624	1
3	The Good Thief	0.3125	0.666666667
4	The Hulk	0.307692319	0.5
5	The Santa Clause	0.294117659	0.4
6	The Matrix	0.285714298	0.333333333
7	The Pianist	0.275862068	0.285714286
8	The Patriot	0.275862068	0.25
9	The New Guy	0.275862068	0.222222222
10	The Terminal	0.266666681	0.2
11	The Animatrix	0.258064508	0.181818182
12	The Terminator	0.25	0.166666667
13	The Last Dragon	0.24242425	0.153846154
	...	...	...
18	The Matrix Reloaded	0.216216221	0.111111111
19	The Fabulous Baker Boys	0.195121944	0.105263158
20	Terminal, The	0.193548381	0.1
	...	...	...

Table 1: Ranking of movie titles values according to the value $q =$"The Green Mile"

by q"). Next, the *precision* of the similarity function at each score level may be calculated as follows.

Let $R(q)_i$ be the number of items in the ranking $R(q)$ that are labeled as "relevant" up to position i. The precision is calculated for each position i in $R(q)$ by the equation:

$$p_i = R(q)_i / i \tag{1}$$

Precision is used as a measure of quality for similarity functions. The precision values computed at the training phase are adopted as an estimation of the precision achieved by the function for any new comparison of pairs of values of the same attribute.

Consider as an example an attribute Title that contains movie titles. Table 1 presents a list of Title values ranked according to the similarity with the string $q =$"The Green Mile" by the n-grams similarity function (any other string similarity function could have been used for this example). The first column contains the position (i) of the value a_i in the ranking; the second column contains the attribute value a_i; the third column contains the ground score $f(q, a_i)$ as given by the similarity function; and the fourth column contains the precision measure p_i (Equation 1). The values at positions $i = 1$ and $i = 2$ (marked in gray) are considered as relevant, i.e. these values are considered to represent the same real world object as q. Thus, we have a precision $p_i = 1$ for these two positions. From position $i = 3$ on, precision decreases, indicating a decrease in the quality of the results obtained with the similarity function.

The precision values in Table 1 can also be interpreted as a measurement of the belief in the scores generated by the q-grams similarity function for evaluating the equivalence between the elements in this data set and a given value q. Specifically, if a value p_i appears associated to a ground score g_i in position i in this table, p_i can be interpreted as being the belief that relevant values are matched to q if ground scores greater or equal to g_i are considered. Take for example, position $i = 3$ in Table 1. The ground score in this line is $g_i = 0.3125$ and the precision is $p_i = 2/3$. An interpretation for this is that, if a ground score $g \geq 0.3125$ is taken, 66.6% of the matches will be relevant.

These observations can be generalized for any other similarity function that consistently generates score values. That is, given a similarity function f, a value q, an attribute A and a ranking $R(q)$ over A according to q, we can build a table similar to Table 1 that maps ground score values to precision values.

Table 1 was built using a single value q. However, to have a better understanding of the behavior of the similarity function when applied to a given attribute, we must use a set of representative values $Q = \{q_1, q_2, \ldots, q_m\}$ to generate a ranking $R(q_i)$ for each value $q_i \in Q$. In this case, instead of building a distinct table for each ranking, we would like to have a single table that summarizes the behavior of the function for all q_i. Then, a single mapping would be generated to express the quality of the ground similarity function considering all q_i in Q.

This could be achieved, for instance, simply by calculating the average of the precision values for a specific ground score value in each ranking. However, as ground score values are likely to be distinct in each ranking, another procedure is required. To address this issue, we resort to another common Information Retrieval procedure, which consists in interpolating the precision results to achieve an average of all results [1]. First, we define a set of arbitrary ground score values and dispose them as points in a scale within a range. For the experiments we have taken the 11 values $0.0, 0.1, \ldots, 0.9, 1.0$, but a larger number of values could be employed in specific cases, when the scores that result of some similarity function are concentrated on a very small range of values. Then, for each ranking $R(q_i)$, we compute an *interpolated precision* value in each of these points based on the real precision values obtained for this ranking. This process is formally defined below:

Definition 3. (Interpolated precision in a ground score) *Let $R(q)$ be a ground ranking according to a value q (as in Definition 2) and consider that for each position i in $R(q)$ there is pair $\langle g_i, p_i \rangle$, where g_i represents the ground score in i and p_i represents the precision at that position. Also, consider that g_{MIN} and g_{MAX} are respectively the minimum and the maximum ground score values in $R(q)$. Let $0 \leq \hat{g} \leq 1$ be an arbitrarily predefined value. The* interpolated precision *for $\hat{g}$ is given by the formula:*

$$\dot{p}(\hat{g}, R(q)) = \frac{p_{low} + p_{high}}{2} \tag{2}$$

where:

- *p_{low} is the smallest value occurring with g_{low} in $R(q)$ and g_{low} is the highest ground score value in $R(q)$ such that $\hat{g} \geq g_{low} \geq g_{MIN}$;*
- *p_{high} is the smallest value occurring with g_{high} in $R(q)$ and g_{high} is the lowest ground score value in $R(q)$ such that $\hat{g} \leq g_{high} \leq g_{MAX}$.*

Definition 3 shows how to calculate the interpolated precision for an arbitrary value $0 \leq \hat{g} \leq 1$. The intuition behind it is the idea that if $\hat{g}$ is a ground score value that already exists in the ranking, the interpolated precision is the same as the known precision value for this score value. Otherwise, the interpolated precision is the average between the known precision values of the highest and the lowest ground score values near $\hat{g}$. For example, in the ranking in Table 1, the interpolated precision for $\hat{g} = 0.5$ is $\dot{p} = 0.833333333$,

the average between the precision values at positions $i = 2$ and $i = 3$. Notice that there may be two or more distinct known precision values for the same ground score value in the ranking. In this case, we choose to always take the lowest precision value when calculating the interpolated precision.

Finally, we compute the average of all interpolated precision values for each predefined point over all rankings $R(q_i), q_i \in Q$. In our work we use the term *adjusted score* to refer to these averages. At this step, we also handle the problem of non-monotonicity. Interpolated precision values may be non-monotonic, i.e., for a specific query it may happen that precision increases as ground scores decrease, or more formally it may happen that for some $0.1 \leq \hat{g} \leq 1$ we have $\dot{p}(\hat{g}, R(q)) < \dot{p}(\hat{g} - 0.1, R(q))$. As we require the adjusted scores to be monotonic, instead of taking directly the interpolated precision values for a specific ground score $\hat{g}$, we take the maximum average interpolated precision over all ground scores that are equal or lower than $\hat{g}$. More precisely, these averages are computed as stated in Definition 4.

Definition 4. (Score Mapping Table) *Let* $Q = \{q_1, \ldots, q_m\}, Q \subseteq A$ *be a set of values of a given attribute* A. *Let* $R(q_k)$ *be the ground ranking according to* $q_k \in Q$ *(Definition 2). Consider that, for each position* i *of* $R(q_k)$, *there is pair* $\langle g_{i,k}, p_{i,k} \rangle$ *representing the ground score and the precision in that position.*

A Score Mapping Table *is a list of pairs* $\langle \hat{g}_i, \hat{p}_i \rangle$ *where* $\hat{g}_i \in \{0.0, 0.1, \ldots, 0.9, 1.0\}$ *is an arbitrarily predefined representative score value and* $\hat{p}_i$ *is the maximum average interpolated precision in* $\hat{g}_i$ *for each ranking* $R(q_k)$, *that is,*

$$\hat{p}_i = \max_{j=0}^{i} \left(\frac{\sum_{k=1}^{|Q|} \dot{p}(\hat{g}_j, R(q_k))}{|Q|} \right) \tag{3}$$

where $\dot{p}(\hat{g}_j, R(q_k))$ *is the interpolated precision for* $\hat{g}_j$ *as in Definition 3.*

We call each $\hat{p}_i$ *an* adjusted score *value.*

Ground Score	*Adjusted Score*
1.0	1.0
0.9	1.0
0.8	0.75
0.7	0.5
0.6	0.176470
...	..
0.0	0.0

Table 2: An example of a score mapping table

Table 2 illustrates an example of a score mapping table generated after building several rankings over a dataset.

The score mapping table allows us to define two functions, one mapping ground scores into adjusted scores and the other one mapping adjusted scores into ground scores. These functions are defined below and will be used in the next sections.

Definition 5. (Ground score to adjusted score mapping) *Let* $M = \{\langle 0.0, p_0 \rangle, \ldots, \langle 1.0, p_{10} \rangle\}$ *be a score mapping table for* f *as in Definition 4. Consider a* ground score *value* $0 \leq g \leq 1$. *The function* $\alpha_M(g) \mapsto [0, 1]$ *that maps ground scores into adjusted scores is defined as:*

$$\alpha_M(g) = \begin{cases} p_i & \text{if } \langle g, p_i \rangle \text{ occurs in } M \\ \frac{p_i + p_{i+1}}{2}, \text{ where } g_i < g < g_{i+1} & \text{otherwise} \end{cases} \tag{4}$$

For example, consider the score mapping table in Table 2 and two ground scores to be mapped, 0.6 and 0.62. The ground score 0.6 appears in the table and maps directly to the adjusted score 0.176470. The ground score 0.62 does not appear in the table, and thus maps to 0.338235, the average between the adjusted scores for ground score 0.6 and ground score 0.7.

Definition 6. (Adjusted score to ground score mapping) *Let* $M = \{\langle 0.0, p_0 \rangle, \ldots, \langle 1.0, p_{10} \rangle\}$ *be a score mapping table for* f *as in Definition 4. Consider an* adjusted score *value* $0 < p \leq 1$. *The function* $\gamma_M(p) \mapsto [0, 1]$ *that maps adjusted scores into ground scores is defined as:*

$$\langle \gamma_M(p), p_i \rangle \in M, p_{i-1} < p \leq p_i \tag{5}$$

For example, consider again the score mapping table that appears in Table 2. By this definition the adjusted score 0.75 would map to the ground score 0.8, as would do all adjusted scores greater than 0.5 and less than or equal to 0.75.

Notice that the values are mapped in distinct ways according to the direction of mapping. When mapping from ground scores to adjusted scores, we take the average of the adjusted scores, whereas when mapping from adjusted scores to ground scores, we take the next higher corresponding ground score. The reasons for this asymmetry of mappings become clear in the next section.

5. ALLOWING MEANINGFUL QUALITY ESTIMATIONS IN APPROXIMATE MATCHING

As stated in the introduction, we propose that the threshold used in match operations is specified in terms of the precision that the user expects from the match operation, since this metric can be used to quantify the quality of the results. It is easy to interpret an adjusted score value x as an indication that the chance of each answer retrieved to be correct is $x\%$.

We believe that providing the adjusted score, as a value that establishes a quality requirement when specifying instance matching, is very intuitive for users. Furthermore, it not only shields the users from knowing how each individual function behaves, but it also creates an opportunity for a single estimation to be provided for the whole match, regardless of the similarity functions used.

We begin by formalizing a match operator that is based just on the ground score and then extend this notion to a match operator that is based on the adjusted score.

Definition 7. (Ground match operator) *Let A be an attribute and $a_1, a_2 \in A$ be values in A. Consider a ground similarity function f as in Definition 1. Consider a* ground threshold *value $0 \leq gt \leq 1$.*

The ground match operator $GM(a_1, a_2, f, gt) \mapsto \{\mathsf{YES}, \mathsf{NO}\}$ *returns* YES*, meaning that the objects match, if $f(a_1, a_2) \geq gt$, and* NO *otherwise.*

Now, following these ideas, we replace the ground threshold (gt) in the match operator by another parameter pt that we call *precision threshold*. Hence, we define a new match operator, called *precision match operator*.

Definition 8. (Precision match operator) *Let A be an attribute and $a_1, a_2 \in A$ be values in A. Let f be a ground similarity function as in Definition 1 and let $M = \{\langle 0.0, p_0\rangle, \ldots, \langle 1.0, p_{10}\rangle\}$ be a score mapping table for f as in Definition 4. Let γ_M be a function that maps adjusted scores into ground scores for table M as in Definition 6. Consider a* precision threshold *value $0 < pt \leq 1$.*

The precision match operator
$PM(a_1, a_2, f, pt) \mapsto \{\mathsf{YES}, \mathsf{NO}\}$
is defined by
$PM(a_1, a_2, f, pt) = GM(a_1, a_2, f, \gamma_M(pt))$

The parameter pt, the precision threshold, is first mapped to the corresponding ground score by function $\gamma_M(pt)$. According to Definition 6, $\gamma_M(pt)$ is the ground score that corresponds to the highest adjusted score value in the mapping score table M that is close to pt. We call this value *estimated ground threshold*. Then, the estimated ground threshold is used as a parameter for the ground match operator to actually "execute" the match. Notice that we do not allow the precision threshold to assume the value zero.

We are now able to explain why, when mapping adjusted scores to ground scores by the function γ_M, we took the highest corresponding ground score (as opposed to taking the average of two values as we have done in the mapping from ground to adjusted scores). The underlying idea is that, in mapping a precision threshold pt to a ground threshold gt, we want to assure that the matching process leads to at least pt% correct results. To assure this percentage of correct matches, we must take the next highest ground score that corresponds to pt in the score mapping table M. If we take any value lower than $\gamma_M(pt)$ we are at risk of violating the requirement of providing at least pt% correct results for the matches with this function.

In the next section we present the results of experiments we have carried out to corroborate the accuracy of the mapping between precision thresholds and ground thresholds.

6. EXPERIMENTAL EVALUATION

In this section we describe the experiments we have performed in order to empirically validate our approach. The goal of these experiments is to demonstrate that the adjusted score, i.e., the precision estimated by training, may be used instead of the original similarity score calculated by a similarity function, allowing the user to specify the quality of the matching process in meaningful terms.

The adjusted scores are estimated by a training process that is executed over some data set. In the experiments, we evaluate two alternatives of data sets for the training process: (1) to use the same database that contains the instances to be matched (the *actual* data set), and (2) to use a larger set of values representing the domain of the values to be matched (a *representative* data set). Using the actual data set for training potentially leads to more accurate results, since the same data set is used during training and matching. On the other hand, training must be repeated for each new database to be matched. Furthermore, if a database has gone through a major update, a new training may be needed. As for the second option, the representative data set, a single training would be adopted for any set of instances of the attribute, but the adjusted scores may not be as accurate as in the case of the actual database.

In both cases, the set of queries Q used for the training consists of 40 distinct values corresponding to real world objects occurring in the data set. These values were used to calculate the score mapping table for every similarity function at every attribute on the database. This means we estimated the behavior of the studied similarity function based on a very limited set of queries, which turns the training costs acceptable. As shown in our experiments, this small number of training queries is enough to achieve good estimation results in all domains and data sets experimented, indicating that our method can be easily applied in practice.

6.1 Experimental setup

The experiments were performed on three distinct domains:

- **Citation domain**. Each article has three attributes: the Title of the article, the name of the Journal in which the article was published and the Publisher of the journal. Values in this domain are in English and come from *The Collection of Computer Science Bibliographies*[1] and from individual BibTeX files collected among the members of the *UFRGS Database Group*[2], totalizing 16,519 citation entries.

- **Movie domain**. This domain comprises data about movies. Each movie has three attributes: Title, Director and Genre. Values in this domain are in English and originate from multiple video rental data sources, such as *Blockbuster US*, *Blockbuster UK* and *Blockbuster CA*. The instances were manually extracted from the Web totalizing 353 movie entries.

- **Name domain.** This is an artificial data set containing person names generated by the Febrl (Freely Extensible Biomedical Record Linkage) tool [6]. The data set contains 5,000 strings representing 500 distinct person names.

In addition to the actual data sets, a representative data set was also built for the Citation and Movie domains. We stress that representative data sets are used only for the training process. The values for these data sets come mainly from Web sites from several distinct sources. By doing so, we were aiming at removing any source-related bias these data sets might have.

For the Citation domain, the values for title, journal and publisher attributes have been extracted from DBLP, Citeseer, Journals and Conference Web sites[3]. For the Movie

[1] http://liinwww.ira.uka.de/bibliography/Database/index.html
[2] http://metropole.inf.ufrgs.br/DBGroup/
[3] The values for the Citation domain were extracted from Web sites of publishers such as Elsevier, ACM Press, IEEE pub., and so on

domain, the values for the title, director and genre attributes have been extracted from several different lists of English-titled movies on the Web.

Table 3 shows for each attribute the total number of values and the number of distinct real world objects represented by the attribute values in the actual and representative data sets. The numbers of distinct real world objects were obtained by manual inspection in each data set.

Data Set	Domain	Attribute	Total	Distinct Objects
Actual	Movies			
		Title	147	75
		Director	92	78
		Genre	35	29
	Citation			
		Title	10910	8416
		Journal	519	254
		Publisher	469	147
	Name			
		Names	5000	500
Representative	Movies			
		Title	2626	1890
		Director	10438	6754
		Genre	412	228
	Citation			
		Title	13204	5281
		Journal	1197	536
		Publisher	585	161

Table 3: Total number of values and number of distinct real world objects represented by the attribute values in the actual and representative data sets

As previously mentioned, we have used a specific similarity function for each attribute depending on the attribute domain. Table 4 lists the similarity functions that have been applied to each attribute. The functions we have used are available at the SecondString[4] or SimMetrics[5] libraries. Based on experiments previously executed [11, 21], we have tried to apply functions that are appropriate for the specific attribute. A discussion and evaluation of several similarity functions may be found in literature [7, 17, 8].

Domain	Attribute/Function
Movie	Title - n-grams
	Director - JaroWinklerTFIDF
	Genre - Levenstein
Citation	Title - n-grams
	Journal - JaroWinkler
	Publisher - MongeElkan
Names	Name - Soundex

Table 4: Similarity functions applied to each attribute

6.2 The training process

The result of the training process is a score mapping table (Definition 4) that represents the relation between ground scores and adjusted scores (estimated precision) for a specific similarity function applied to a specific attribute. We have followed the procedure that is described in detail in Section 4.

1. **Training set**. A sample consisting of 40 distinct values $Q = \{q_1, q_2, \ldots, q_{40}\}$, randomly chosen amongst the attribute values, was built. Identical duplicate values were eliminated, since our aim is to evaluate the ability of our approach in finding distinct representations of a real world object.
2. **Ground ranking computation**. Next, for each value $q \in Q$ a *ground ranking* $R(q) = \{a_1, a_2, \ldots, a_n\}$ was built based on Definition 2.
3. **Interpolated precision computation**. In this step, a human expert labeled each value a_i in each ranking as "relevant" (i.e., "representing the same real world object as q") or as "irrelevant" otherwise. Then, for each position i in the ranking, the *precision* at this position was computed by Equation 1. Finally, the *interpolated precision* (Definition 3) at each one of the eleven ground scores $[0.0, 0.1, \ldots, 0.9, 1.0]$ was computed.
4. **Score mapping table computation**. Having as input the 40 rankings generated in the previous step, the *score mapping table* (Definition 4) for this attribute was computed.

The training set size was established by observing how adjusted scores vary when the size of the training set increases. For instance, Figure 2 shows that behavior of attribute Title in the Citation representative data set. This figure depicts plots of adjusted scores for the ground scores 0.4, 0.5 and 0.8. As illustrated, the values of the adjusted score tend to stabilize, in this specific case, at 30 values. Experiments with the remaining attributes show that with training sets of around 40 values stability is reached. This training set size is in accordance to a rule-of-thumb that is often applied in the information retrieval field [4]. By this rule, a set of queries of size between 25 and 50 is usually employed in the evaluation of search engines.

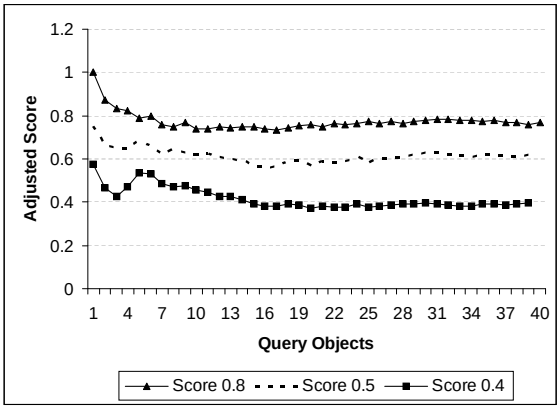

Figure 2: Evolution of adjusted score values wrt. training set size

The learning process resulted in two score mapping tables for each attribute in the Citation and Movie domains:

[4]secondstring.sourceforge.net/
[5]sourceforge.net/projects/simmetrics/

one table (M_a) corresponding to the actual data set and the other table (M_r) corresponding to the representative data set. For the Name domain a single mapping table was obtained, as this domain contains a single attribute and no representative data set was built for it.

6.3 Verifying the accuracy of the adjusted score

The experiments in this section aim at comparing the precision results obtained when using ground scores with those obtained using adjusted scores.

For each attribute we repeated the steps described in the previous section, but employed a different set of 40 randomly chosen values and instead of applying the *ground match operator* (Definition 7) we applied the *precision match operator* (Definition 8), i.e., instead of taking directly the ground scores that result from the similarity function, we took the corresponding adjusted scores as given by the score mapping table. So as to demonstrate what happens when our approach is not used, we have also computed the precision when applying the ground match operator (exactly the same procedure as in training but with different query objects).

An example of the result of this experiment for attribute Journal in Citation database is shown in Table 5. The first column contains the eleven threshold values for which the precision was computed. The second column (*apg*) contains the values of precision that were computed using the *ground* match operator and the third column (*apa*) contains the values of precision with *adjusted* match operator.

Threshold	apg	apa
1	1	1
0.9	0.395210825	0.825
0.8	0.173330854	0.825
0.7	0.096967027	0.825
0.6	0.037191466	0.423593677
0.5	0.005228376	0.423593677
0.4	0.00216138	0.423593677
0.3	0.001792878	0.423593677
0.2	0.001764729	0.199612164
0.1	0.001739088	0.105933207
0.0	0.001739088	0.047110765

Table 5: Precision values at the eleven points for the attribute Journal in the Citation database

Remember that the central aspect of our approach is to allow the user to specify the quality of the matching process in terms of a threshold value that expresses the precision he/she expects. If our approach works correctly, the average precision obtained with the adjusted score (*apa*) should be equal to the given threshold. However, as the estimation process is not likely to be perfect, generally there will be differences between these two values. For example, in Table 5 the average precision computed for the threshold value 0.5 is 0.423593677.

In order to validate our approach, we must evaluate the difference between the average precision obtained in the experiments and the expected precision given by the user as a threshold.

Figure 3 shows the values for average precision for the attributes in the Citation domain. Plots for Journal attribute appear in the two graphs on the left, for Publisher attribute in the two graphs in the middle and for Title attribute in the two graphs on the right. The graphs at the top contain the results using the adjusted score estimated over the actual data sets, whereas the graphs at the bottom reflect training over the representative data sets. In each graph, the x-axis contains the threshold values and the y-axis contains the average precision obtained for both, ground and adjusted scores. Each graph contains three lines. The straight line (y=x) represents the ideal case, when the average precision is exactly the expected precision provided by the user by means of the threshold value. The dotted line corresponds to the values apa_i of the average precision achieved when using adjusted scores, whereas the dashed line shows the values of average precision achieved when using the ground score. In other words, the dotted line reflects the application of our approach, whereas the dashed line reflects the results when our approach is not applied. This last line is given as a baseline for the evaluation of our approach.

As the graphs visually show, our approach leads to values that are clearly correlated to the precision expected by the user (straight line), which is closer to the ideal behavior. We stress that these results were obtained with low training effort, involving samples of just 40 values.

In order to quantify the differences between the expected precision given by the user by means of a threshold value and the average precision obtained by our approach, we have calculated the deviation of the squares of the differences (as in the least squares fitting method) between the lines. This deviation is given by the equation:

$$d_a = \sum_{i=0}^{10} [apa_i - t_i]^2 \quad (6)$$

where t_i is one of the eleven threshold values (the precision expected by the user) and apa_i is the average precision achieved for each threshold when using adjusted scores.

We have also computed the deviation between the results obtained when our approach is not applied (dashed line), i.e., when the results of the similarity function are directly applied, with the ideal case (straight line). This deviation is computed by the formula:

$$d_g = \sum_{i=0}^{10} [apg_i - t_i]^2 \quad (7)$$

where t_i is one of the eleven threshold values and apg_i is the average precision computed using directly ground scores.

Domain	Attribute	dg	da (Representative Dataset)	da (Actual Dataset)
Movies				
	Title	0.245324404	0.035447116	0.018727643
	Director	0.548393691	0.070206345	0.191055858
	Genre	0.180179338	0.0351317	0.025824542
Citation				
	Title	0.148328543	0.047467229	0.051953852
	Journal	1.042834481	0.076918953	0.071390621
	Publisher	1.269746357	0.19185443	0.150769219
Name				
	Names	2.188456337		0.167900628

d_g – Deviation between the ideal case and the results obtained when the similarity function is directly applied.
d_a – Deviation between the ideal case and the results when the adjusted score is applied with training on the actual data sets and training on the representative data sets

Table 6: Deviation values

Table 6 shows the deviation values for each attribute. Column d_g contains the deviation between the ideal case and

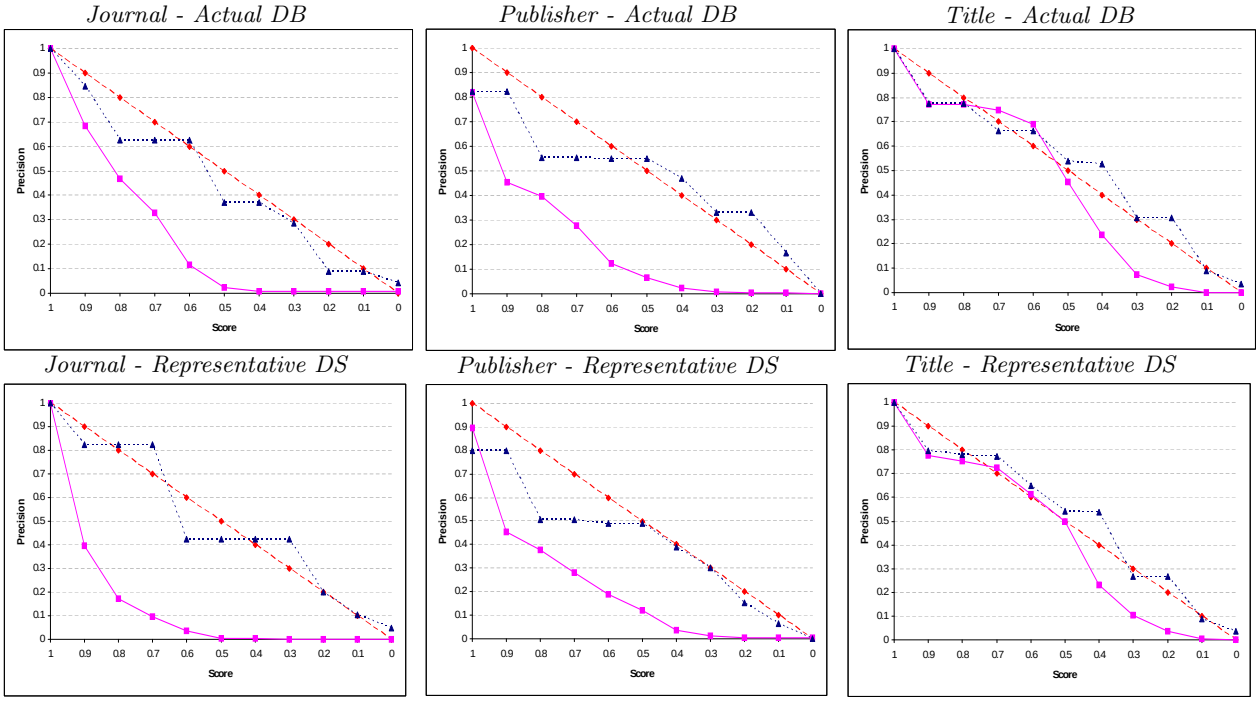

Figure 3: Average precision computed using adjusted and ground scores for the Citation domain

the results obtained when the ground similarity function is directly applied. Columns labeled with d_a contain the deviation values obtained by using the score mapping table that results from training on the *actual databases*, as well as deviation values obtained from training on the *representative data sets*.

The results obtained in these experiments show that using our approach with the precision match operator we obtain a behavior that is closer to the ideal case. This fact can be visually observed on the graphs in Figure 3 and is confirmed by the small deviation values shown in Table 6. Notice that, for the Movies domain, the results in Table 6 are in fact a compact representation of results presented in the graphs in Figure 3.

In addition, the results show that training on representative data sets leads to results that are close to those obtained when training on actual databases.

7. CONCLUSIONS

In this paper we have presented an approach for solving a central problem in approximate matching, namely that of defining a threshold for similarity functions. On the basis of a training process, we define a mapping between scores computed by a similarity function and the precision that is to be expected if several matches are executed using the corresponding score as a threshold. This mapping allows us to introduce a match operator that receives as a parameter the precision that is expected from the matching process. This expected precision is a value that is meaningful to the users, as opposed to the threshold specified in terms of the scores returned by a similarity function.

The experiments performed for evaluating our approach show the accuracy of the mapping between scores obtained by a similarity function and the corresponding expected precision. Furthermore, these experiments indicate that the training process executed on a representative data set (set of values of an attribute collected from several sources emulating the domain of an attribute) leads to values that are close to those obtained when the training process is executed over the actual database. This is a promising result, meaning that the training may be performed once for several databases that include data from the same domain.

Human intervention to label each value in each ranking as "relevant" or "irrelevant" is required during the training process. An alternative to reduce this human intervention could be the approach presented in [21], in which this labeling is performed semi-automatically. In that approach, instead of having to label each value in the ranking, the user just informs the number of distinct real world objects represented in several samples of the data set.

In this paper we have used precision as an adjusted score for similarity functions. The underlying idea is to allow users to specify the results they expect from the matching process by stating the percentage of correct results required. However, other quality measures could be used as well. For example, another intuitive measure could be *recall*. If estimated recall is used instead of estimated precision, users would be able to specify the results they expect by stating the percentage of the total relevant objects in the database they expect to be matched. Some preliminary results we have obtained show that this is feasible.

Acknowledgments

We would like to thank Marcos Freitas Nunes for the help with the experiments. This work has been partially supported by CAPES, CNPq and FAPERGS (Projects Gerindo, XML/Broker and PER/XML; Carina F. Dorneles scholarship).

8. REFERENCES

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, September/October 2003.

[3] N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. Database Syst.*, 27(2):153–187, 2002.

[4] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *ACM SIGIR 2000*, pages 33–40, New York, NY, USA, 2000. ACM Press.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD 2003*, pages 313–324, New York, NY, USA, 2003. ACM Press.

[6] P. Christen, T. Churches, and M. Hegland. Febrl - a parallel open source data linkage system. In *PAKDD 2004 (LNAI 3056)*, pages 638–647. Springer, 2004.

[7] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*, pages 73–78, 2003.

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

[9] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB 2004*, pages 864–875, 2004.

[10] A. Doan, Y. Lu, Y. Lee, and J. Han. Profile-based object matching for information integration. *IEEE Intelligent Systems*, 18(5):54–59, 2003.

[11] C. F. Dorneles, C. A. Heuser, A. E. N. Lima, A. S. da Silva, and E. S. de Moura. Measuring similarity between collection of values. In *WIDM 2004: 6th ACM Intl. Workshop on Web Information and Data Management*, pages 56–63, New York, NY, USA, 2004. ACM Press.

[12] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Society*, 64:1183 – 1210, 1969.

[13] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an rdbms for web data integration. In *WWW 2003*, pages 90–101, New York, NY, USA, 2003. ACM Press.

[14] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the results of approximate match operations. In *VLDB 2004*, pages 636–647, 2004.

[15] S. Guha, N. Koudas, D. Srivastava, and X. Yu. Reasoning about approximate match query results. In *ICDE 2006*, page 8, Atlanta, GA, USA, 2006.

[16] N. Koudas, A. Marathe, and D. Srivastava. Flexible string matching against large databases in practice. In *VLDB 2004*, pages 1078–1086, Toronto, Canada, 2004.

[17] L. Lee. On the effectiveness of the skew divergence of statistical language analysis. *Artificial Intelligence and Statistics*, pages 65–72, 2001.

[18] A. Motro. Vague: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, July 1988.

[19] E. S. Ristad and P. N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May 1998.

[20] SecondString. Carnegie Mellon University. Project Page, http://secondstring.sourceforge.net/.

[21] R. K. Stasiu, C. A. Heuser, and R. Silva. Estimating recall and precision for vague queries in databases. In *CAISE 2005*, Lecture Notes in Computer Science, pages 187–200. Springer Verlag, 2005.

[22] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.