The Object Identification Framework[*]

Mattis Neiling
Free University of Berlin, Dept. of Economics
Institute for Information Systems
mneiling@wiwiss.fu-berlin.de

Steffen Jurk
BTU Cottbus
Dept. of Database and Information Systems
sj@informatik.tu-cottbus.de

ABSTRACT
The object identification problem in databases has been tackled in many different ways, e.g. *Record Linkage*, or the *Sorted Neighborhood Method.* We present a framework, that allows the evaluation of the competing approaches. Appropriate experiments on a real-world database has been made.

1. MOTIVATION
Object Identification is essential if a real-world objects data is distributed over multiple databases. The object identification task becomes complicated, if no global and consistent identifier is shared all over the databases. This situation occurs often for real-world applications, such as *information integration*, *data warehousing* or the *administrative record census.* Then, object identification can only be performed through an utilization of the *identifying information* provided by the object's data. Unfortunately real-world data is dirty, hence identification mechanisms like *natural keys* fail — we have to take care of the variations and errors of the data. Consequently, object identification can not be guaranteed to be fault-free.
Different methods tackle the object identification problem, e.g. There are several algorithms developed and successful applied to large databases like *Record Linkage* [11], or the *Sorted Neighborhood Method* [4]. But until now no comprising methodology for object identification exist that allows an evaluation of competing solutions.
In this paper we recommend a generic framework that forms a base for such an universal methodology. We do not discuss new algorithms, but we provide a generic approach to object identification problems. Our framework for object identification requires no global identifiers. Within the framework the object identification task is interpreted as a specific classification problem. It consists basically of the three succeeding steps: (1) *Conversion*, (2) *Comparison* and (3) *Classification.* If identifying attributes are derivable from all the data sources (step 1), these attributes can be compared for pairs of elements (step 2), and finally the pairs can be classified as (*not*) *matched*, or *matched to some degree*, according to their comparison values (step 3). To be efficient for large databases, we apply a preselection technique for reducing the number of pairs of elements to compare by incorporating suitable index structures.
We illustrate our framework with a case study of a customer address database.

[*]This work was supported by the Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316) and summarizes parts of the dissertation of the first author [6].

2. THE FRAMEWORK IN A NUTSHELL
A database schema S consists of classes with types. Types are recursively built from a type system τ with basic types ν, labels ℓ and references to labels $\ell : \tau$ by use of constructors like $\times$ (tuple), $\{\ \}$ (set), and $[\ ]$ (list). Then a database D is an instance of S, whereby the elements are instances of classes belonging to S.

2.1 Conversion Step
Identification of objects requires an intensional overlap of the data sources, e.g. the existence of common attributes. At least a set of attributes with identifying information has to be *derivable* from both data sources.
For two databases (sources) D_1, D_2 with schemas S_1, S_2 we introduce a conversion procedure. First we determine these classes C, that are usable for identification, e.g. books within a library but without information of lends to users. For each class C we require two partial schema mappings (conversion functions) $h_1^C : S_1 \to C$ and $h_2^C : S_2 \to C$ that extract common structures of C from S_1 and S_2, respectively. Formally a common schema S is derivable from S_1 and S_2, such that a real-world object represented in S_1 and S_2 became the same representation in S.[1] The schema S should contain all the identifying information provided by both S_1 and S_2.

2.2 Comparison Step
Since the conversion yields a common schema, comparison of elements (instances of classes) becomes more easier. The comparison vector of two elements e_1 and e_2 is defined by a (finite) set of user-defined functions, $f(e_1, e_2) = (f_1(e_1, e_2),$

[1]Since references to other classes might contain identifying information, the references should be mapped into S adequately: References between instances of classes of S_i should be mapped into S, if representable therein.

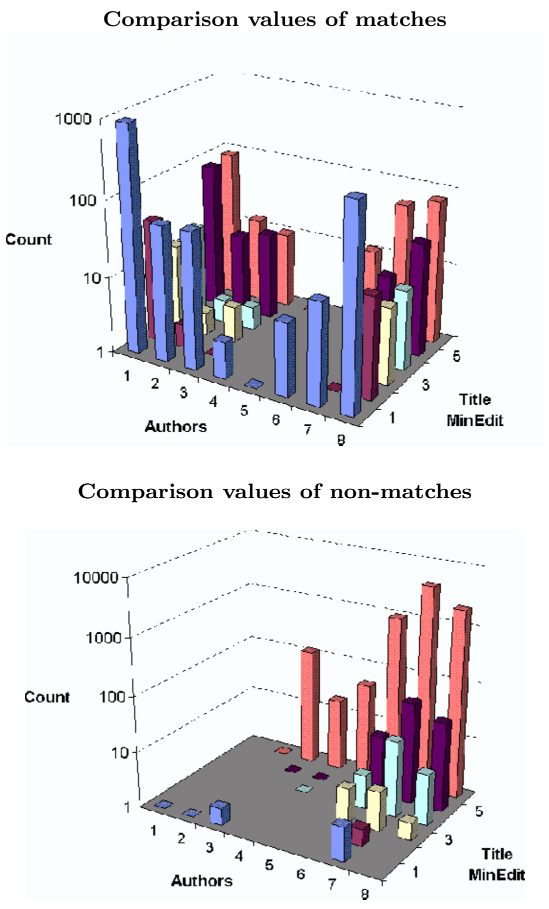

Figure 1: Frequency of comparison values taken from a sample of pairs library records (Note: the z–axis is logarithmic scaled).

..., $f_k(e_1, e_2)$). Each f_i represents a numeric value with respect to a nominal, ordinal or cardinal scale. To compare single attribute values suitable binary functions can be applied, e.g. (discrete) metrics. If relevant for identification, nominal scaled comparison functions can manage structural differences, missing values, and other special cases for pairs of elements. The choice of comparison functions can be guided by a *discernibility principle: Matches and non-matches should have different comparison vectors.*

Example 1. Consider a sample of 10,000 pairs of book objects and two comparison functions. Function f_1 reflects the degree of equal authors of two books within a range of 1 to 8. Where 8 reflects a missing author list and 1 a fit of all authors. Function f_2 reflects the edit distance of the book titles ranging from 0 to 10 edits (cases 0–4) or more edits (case 5). Figure 1 depicts the outcomes of the comparison vectors (f_1, f_2) with respect to the number of matching and non-matching pairs in the chosen sample.

2.3 Classification Step

The comparison functions span a multidimensional comparison space V, where certain regions indicate for comparison values of matches or non-matches, respectively. Unfortunately, there are often regions, where a separation becomes more incorrect, t.i. certain comparison vectors indicate neither matches nor non-matches (e.g. the right upper edge in figure 1). Sometimes a refinement of the comparison functions for this regions leads to better results.
The object identification can be perceived as a *classifier* $\delta : V \to [0, 1]$ for comparison vectors $v = f(e_1, e_2)$ of pairs of elements e_1, e_2, whereby 0 denotes matches, 1 non-matches and $(0, 1)$ a certain *degree of match.*
The *classifier* δ can be defined manually (e.g. the decision rules of the *Sorted Neighborhood Method*) Alternatively it can be learned from given example data, t.i. a set of matches and non-matches. For example, within the *Record Linkage* method, the likelihood ratios of comparison vectors for matches and non-matches (the so-called *odds*) are estimated and used as classifier.
There exist many suitable classification methods in literature, e.g. *Decision Tree Induction*, *k-Nearest Neighbor Classification*, *Support Vector Machines*, *Neural Networks*, *Bayes Classifier*, etc. Also a combination of different classifiers has been studied under the term *Boosting*, e.g. applied for object identification in [10]. The interested reader may consult textbooks about *Machine Learning* (e.g. [5, 1]), or existing *Data Mining Software.*

2.4 Preselection Of Pairs

To be efficient for large databases, preprocessing is applied. Let δ' be a classifier for pairs of elements from two databases D_1, D_2. Within the preprocessing we avoid pairs of elements that are likely not matched. T.i. we use a combination $\sigma = \bigcup_j (\bigcap_i \sigma_{ij})$ of *selectors* σ_{ij}, where σ_{ij} filters pairs from the cross product space $D_1 \times D_2$. Then we can apply the classifier $\delta = \delta' \circ \sigma$ for object identification, reducing the number of pairs to check.
Each selector has a *selection rate*, quantifying the percentage of the selected pairs from $D_1 \times D_2$, an *error rate*, estimating the portion of the not selected matches, and *preprocessing costs.* Typically, the lower the selection rate the better the performance of the whole identification task, since the main cost of object identification is determined by loading and processing of pairs. But obviously there is a trade off between the error rate and the selection rate. Thus choosing a selector among a set of possible selectors (including combinations of them) becomes an optimization problem, c.f. [8], [6, Ch.5].

Example 2. A *relational selector* σ poses conditions on attribute values, e.g. requiring equality (this is sometimes called *blocking*) or containment of a value in a list, or limiting the variation of cardinal scaled attributes by some $\Delta > 0$. Index structures, such as bitmaps or tree-based structures, are available in database management systems and can be used to achieve efficient data access.
A *metrical selector* σ poses conditions on attributes in terms of a given (multidimensional) metric, e.g. the *Minimum–Edit–Distance* for strings. A metrical selector allows (1) the selection of the k nearest neighbors of an element, or (2) the selection of all elements within a Δ–environment for $\Delta > 0$. Metrical index structures can be employed, e.g. the M-tree [3] or the MVD-tree [2].

Remark 1. Since the sort key of the *Sorted Neighborhood Method* induces a total order of the records, the scan window of size $2k$ can be perceived as a metrical selector σ filtering the k nearest neighbors of an record.

We mention only the semantic constrains introduced in [7]. Attributes fulfilling the constraints of a *semantic* or *approximative key* are good candidates for attributes to compare, since these attributes are almost as good for identification as natural keys in databases. Further *(approximative) differentiating keys* can be used for the definition of relational selectors, since the unequalness of their attribute values indicates for non-matches.

3. CASE STUDY

We tested three classification methods on a database including 250,000 German private address records.[2] Additionally a list of over 55,000 duplicates was provided by the *Double Clean* service of the *Deutsche Post*[3], that employs the regularly updated register of German postal delivery addresses. We used this list as reference for the evaluation of the correctness of a classification.

For object identification we used the attributes Firstname, Lastname, Zip, City, Street, and Birthdate. We derived further attributes like Fullname, Housenumber, and Sex.

As comparison functions we used the *Minimum-Edit-Distance* for strings, *equal/not equal* patterns for other attributes, or additionally we took care of the occurrence of missing values, and therefore we added the third case *missing*.

As preselection σ we applied the matching of at least one phonetic code of the Fullname according to the Kölner Phonetic [9]. There were only 34 duplicates without any matching, thus the error rate of this preselection was below 0.1%. Thereby we yield a selection rate only about 1%, t.i. each record must be compared with 100 up to 20,000 records.

3.1 Evaluation

We tested the classification methods *Record Linkage* and *Decision Tree* on several samples of pairs, each method with different parameterizations.

The parameterizations of *Record Linkage* differ (1) in the selection of attributes (ranging from 4 to 9, always two name comparisons and further comparison of attributes like Zip, Street, Birthdate and Sex) and (2) the allowed factor interactions between them.[4]

For *Decision Tree Induction* we varied (1) the measure used for partitioning, e.g. *information gain*, *information gain ratio*, or *gini-index* and (2) whether pruning was applied or not, while always all attributes were used from the DT learner.

We applied each parameterization on samples of increasing size (5,000, 10,000 and 20,000 pairs), and we generated for each size 12 random samples. We report the results of the correctness test, where a sample was split up into a learn and a test sample. We calculated the α– and β–error rates on the test sample, defined by

$$\alpha = \frac{\#\text{misclassified matches}}{\#\text{matches in the sample}},\ \beta = \frac{\#\text{misclassified non-matches}}{\#\text{non-matches in the sample}}.$$

While all of the Decision Tree parameterizations behave similar with α below 1% and β below .5% (for the largest sample), only the α–error of Record Linkage gets a little bit poorer in all settings (slightly above 1% at best). But there were some false-specified parameterizations: Record Linkage models with more than 5 of all 12 attributes leads to α–errors above .25%, and the smaller models without two-factor interactions produce similar results. If the number of involved attributes increases, the multinomial distribution can not be estimated adequately. Furthermore, the cardinal scales of distance measures are reduced to nominal scales for Record Linkage.

We conclude, that if many attributes with identifying information are given, the divide-and-conquer approach immanent to Decision Trees performs best.

[2]These tests are part of the *Benchmarking*-Chapter in [6], where further tests were applied to library data and online apartment announcement data.

[3]C.f. `http://www.addressfactory.de`

[4]The restriction of the factor interactions are fundamental for the estimation of the multinomial distribution.

4. OUTLOOK

The progress of computer technology and the exponentially increasing amount of data requires methods, that are capable to integrate data from different sources. Therefore the solution of object identification problems is essential for nowadays and future information systems. Based on our framework object identification methods can be specified, implemented and evaluated. Furthermore it is possible to get the best method fulfilling the requirements of an specific application in terms of correctness, performance, costs, or other quality criteria.

5. REFERENCES

[1] M. Berthold and D. Hand, editors. *Intelligent Data Analysis: An Introduction.* Springer, 1999.

[2] T. Bozkaya and Z. Özsoyoglu. Indexing large metric spaces for similarity search queries. *TODS*, 24(3):361–404, 1999.

[3] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *VLDB'97*, 1997.

[4] M. Hernandez and S. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[5] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification.* Horwood, New York, 1994.

[6] M. Neiling. *Identifizierung von Real-Welt Objekten in Multiplen Datenbanken.* Dissertation (draft), BTU Cottbus, Cottbus, Germany, 2003. in German.

[7] M. Neiling, S. Jurk, H.-J. Lenz, and F. Naumann. Object identification quality. In *Intl. Workshop on Data Quality in Cooperative Information Systems (DQCIS2003), Siena, Italy*, 2003.

[8] M. Neiling and R. Müller. The good into the pot, the bad into the crop. preselection of record pairs for database integration. In *1st Workshop DBFusion 2001.*

[9] H.J. Postel. Die Kölner Phonetik - ein Verfahren zur Identifizierung von Personennamen auf Grundlage der Gestaltanalyse. *IBM-Nachrichten*, 19:925–931, 1969.

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

[11] W. Winkler. Matching and record linkage. In Cox, ed., *Business Survey Methods*, J. Wiley, New York, 1995.