Filters for Eliminating Duplicate Records

Otilia PÎRLOG
National Defense Ministry

High quality data is essential for gaining the confidence of users of most current decision support applications. Data cleansing is critical for a wide variety of applications. The duplicate elimination problem of detecting multiple records, which describe the same real world entity, is an important data cleaning problem. Large proportion of time in data cleansing is spent on the comparisons of records. First, it is presented a simple and fast comparison method based on textual similarity. The second method determines the potential duplicate records based on the level of similarity between the two records and a third record. The construction is derived from the triangular inequalities applied to the records of databases. The last approach uses a dimensional hierarchy and adopts a grouping strategy. Since such groups are often much smaller than the entire relation, this strategy allows us to compare pairs of tuples in each group.
Keywords: *data quality, duplicate elimination, field similarity, record similarity.*

Data warehouse construction

Data warehouses are repositories of data collected from several data sources. For decision support, a data warehouse must provide high quality data and services. Therefore, significant amount of time and money are spent on the process of detecting and correcting errors and inconsistencies. Errors in databases have been reported to be of up to ten percent range and even higher in a variety of applications. In [Wang95] it is reported that more than $2 billions of U.S. federal loan money had been lost because of poor data quality at a single agency, manufacturing companies spend over 25% of their sales on wasted practices, service companies up to 40%.

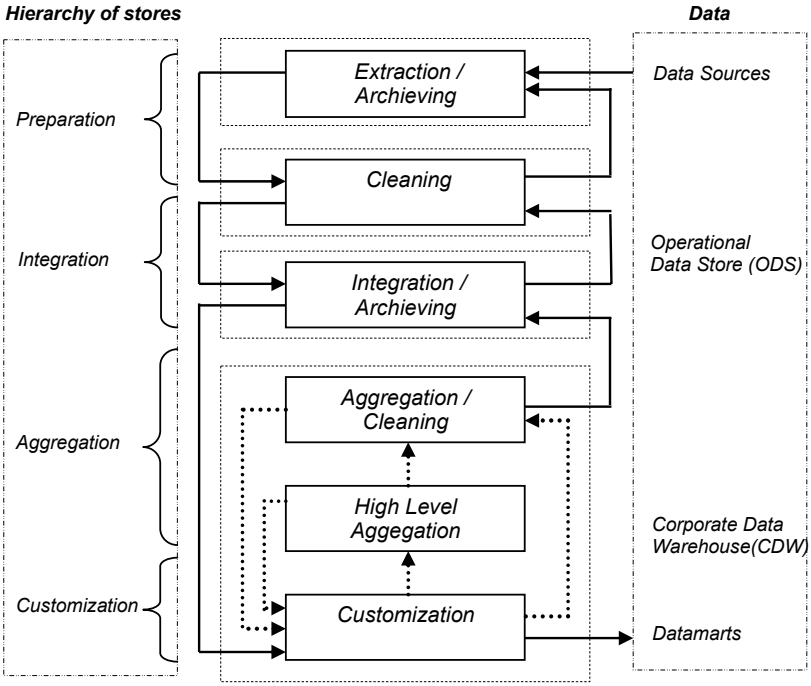

Fig.1. The hierarchical construction of a data warehouse

The data warehouse can be defined as a hierarchy of data stores which goes from source data to highly aggregated data (often called datamarts). Between these two extremes there can be other data stores depending on the requirements of applications. One of these stores is the corporate data warehouse store (CDW) which groups all aggregated views that serve to generate the datamarts. Since data sources are independent, they may adopt independent and potentially inconsistent conventions. The corporate data store can be complemented by an operational data store (ODS) which groups the data collected and integrated from the sources. The hierarchy of data stores is a logical way to represent the data flow between the sources and the datamarts.

There are four levels in the construction of the hierarchy of stores (figure 1). The first level includes the extraction of data from the operational data sources, their cleaning with respect to the common rules defined for the data warehouse store and their possible archiving in the case when integration needs some synchronization between extractors. The next level is the integration of data originated from heterogeneous sources. This level is often coupled with rich data transformation capabilities. The process continues with the data aggregation for the purpose of datamarts construction, which is the last level.

Similarity of records

The integrated data frequently contains approximately duplicate field-values and records that refer to the same entity but are not identical. Variations in representation can arise from typographical errors, misspellings, abbreviations, as well as other sources. The problem of identifying approximately duplicate records in databases has been studied as record linkage, the merge/purge problem, hardening soft databases and field matching.

Duplicate elimination is hard because it is caused by several types of errors, like typographical errors and equivalence errors – different representations of the same logical value. For most of the occurrences, these representations are not non-unique and non-standard. It is important to detect and clean equivalence errors because an equivalence error may result in several duplicate tuples.

In order to detect inexact duplicates, the most reliable way is to compare every record with every other record, which takes $N(N-1)/2$ comparisons, where N is the number of records in the database. To cleanse large databases this way would take very long time.

Most existing approaches have focused on efficient algorithms for locating potential duplicates rather than precise similarity metrics for comparing records. In order to reduce the cleansing time, in this paper there are presented two methods for detecting potential duplicates relied on threshold-based textual similarity functions. If the degree of similarity for two records exceeds a certain threshold, σ, they are treated as a potential duplicate pair.

Similarity is used to describe the degree of similarity of records. Similar records should have large similarity and dissimilar records should have small similarity. The class of equivalence errors can be addresses by building sets of rules.

Suppose that we determined the similarities between corresponding fields of records R_1 and R_2, $Sim_{Fj}(R_1, R_2)$, for all fields F_j, where $j=1 \div n_f$. Based on the field similarity, we can compute the similarity for records, $Sim(R_1, R_2)$. Suppose that a database has fields F_1, F_2 ... F_{nf} with field weights w_1, w_2 ... w_{nf}, respectively, where $\sum_{j=1}^{nf} w_j = 1$.

The similarity of records is given by the expression:

$$Sim(R_1, R_2) = \sum_{j=1}^{nf} (Sim_{Fj}(R_1, R_2)) \times w_j)$$

A Textual Similarity Filter

Most of the domain-independent methods for duplicate elimination rely on textual similarity functions. The fields are treated as sets of characters, so called ‘bag of words’.

Before cleansing there is a pre-processing on records, which deals with data type checks, format standardization and inconsistent abbreviations. After this process, for any two duplicate records, the corresponding fields in

them should have almost the same characters. Suppose a field F in record A has the character set $A_F=\{x_1, x_2 \dots x_n\}$ and the corresponding field in record B has the character set $B_F=\{y_1, y_2 \dots y_m\}$, where n and m are the numbers of characters of these two fields.

We define the field similarity, $Sim: D \times D \mapsto [0,1]$, the number of characters in the intersection of these fields divided by the larger number of characters of them:

$$Sim_F(A,B) = \frac{|A_F \cap B_F|}{max(n,m)}$$

The intersection of fields A_F and B_F represents the vector $z\ (max\ (n, m))$, which elements z_k, $1 \leq k \leq max\ (n, m)$ are computed following the next rule:

$$\begin{cases} z_k = 1, \text{ if } x_k = y_k \\ z_k = o, \text{ if } x_k \neq y_{ks} \end{cases}$$

Using the vector $z(max(n,m))$ we compute the intersection $|A_F \cap B_F|$: $|A_F \cap B_F| = \sum_{k=1}^{\max(n,m)} z_k$

Even if this is a positional expression, we have to observe that two similar fields have approximately the same characters. If we compute the number of apparitions for every discrete character, each character will have the same number of apparitions in those fields. Opposite, if the lengths $n=m$, the probability for having the same number of every character in two fields and the fields to be different tends to zero.

Let define the vectors $CarA_F(n_1)$ and $CarB_F(n_2)$, that represent the number of apparitions of every discrete character in the fields A_F and B_F. The dimensions of these vectors are less than the dimensions of A_F and B_F, $n_1 \leq n$ and $n2 \leq m$. Now, the intersection of fields A_F and B_F will be computed by using of the vector $h(max\ (n_1, n_2))$, which elements h_l, $1 \leq l \leq max\ (n_1, n_2)$ are decided following the next rule:

$$\begin{cases} h_l = 1, \text{ if } CarA_F(l) = CarB_F(l) \\ h_l = 0, \text{ if } CarA_F(l) \neq CarB_F(l) \end{cases}$$

The intersection $|A_F \cap B_F|$ will be $|A_F \cap B_F| = \sum_{l=1}^{\max(n1,m2)} h_l$ and the level of similarity between fields:

$$Sim_F(A,B) = \frac{|A_F \cap B_F|}{max(n_1,n_2)}$$

A Triangle Inequality Filter

The complementary value for similarity is the distance between fields, $d: D \times D \mapsto [0,1]$. For the same records, A and B, $d(A,B)=1 - Sim(A,B)$.

Further, we have to consider the triangle inequality. Let x and y be vectors. Then the triangle inequality is given by

$$|x| - |y| \leq |x + y| \leq |x| + |y|$$

If real numbers (x, y, z) are the sides of a triangle, then $z < x + y$, the triangular inequalities are the inequalities $|x - y| \leq z \leq x + y$

For any three records, A, B and C, from this inequality we derive the following two properties:

- $d(A,C) \leq d(A,B) + d(B,C)$
- $d(A,C) \geq d(A,B) - d(B,C)$

By substitution in these expressions with the equivalent values of similarities, we have:

$$Sim(A,C) \geq Sim(A,B) + Sim((B,C) - 1$$
$$Sim(A,C) \leq 1 - Sim(A,B) - Sim((B,C)$$

In this way, we are able to have information about the similarity between two records, A and C, without making a direct comparison of them. We call the right side of expressions, respectively, $LL = Sim(A,B) + Sim((B,C) - 1$, lower limit and $UL = 1 - Sim(A,B) - Sim((B,C)$, upper limit similarity.

Suppose that the similarity threshold is σ, $0 \leq \sigma \geq 1$. For any three records A, B and C, the records A and C are duplicate if $LL(A,C) \geq \sigma$ and they are non-duplicate if $UL(A,C) < \sigma$. The last situation is when the records do not satisfy any of these rules. This means that we have to decide the relationship between them by applying more exactly method of comparison.

By reducing the number of comparisons, the time for computation for large databases is shorter. From this point of view, the method represent a filter for settle the potential duplicate records. Like all detection methods, we first have to sort the records using a settled key.

A Grouping Strategy Filter

The windowing strategies sort a relation on a key and compare all records within a skid-

ding window on the sorted order. In this way, the equivalence errors may be not adjacent to each other in lexicographical sort orders.

Using textual similarity functions to detect duplicates, due to equivalence errors, requires that the threshold be dropped low enough. The result of it is a large number of false positive pairs of tuples incorrectly detected to be duplicates. A very efficient method for reducing the number of false duplicates is to use the dimensional hierarchies typically associated with dimensional tables in data warehouses.

In figure 2 it is presented a hypothetical schema for database information.

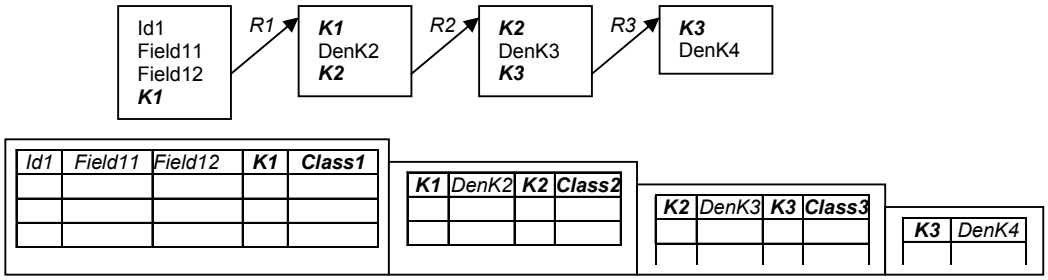

Fig.2. The Hypothetical Schema of a Database

The dimensional hierarchy consists of three relations, connected by referential links that realize the joins between key and foreign key. They are used in order to avoid comparing all pairs of tuples in each relation. There are a top and bottom relations within this hierarchy.Relations $R_1, ...R_m$ with keys $K_1, ... K_m$ constitute a dimensional hierarchy if and only if there is a key – foreign key relationship between R_{i-1} and R_i , where $2 \leq i \leq m$. R_i is the i^{th} relations in the hierarchy. R_1 and R_m are the bottom and the top relations, and R_i the *child* of R_{i+1}.

A tuple v_i in R_i joins with a tuple v_j in R_j if there exists a tuple v in R such that the projection of v on R_i and R_j equal v_i and v_j respectively. Specifically, we say that v_i in R_i is a child of v_{i+1} in R_{i+1} if v_i joins with v_{i+1}. The notations for $Class_1, Class_2$ and $Class_3$, from the hypothetical example, mark the possibility of grouping those tuples into children sets.

In typical dimensional tables of data warehouses, the values of key attributes $K_1, ... K_m$ are artificially generated by the loading process before a tuple v_i is inserted into R_i. In this approach, a tuple in the parent relation R_i joins with a set, which we call its child set, of tuples in the children relation.

Let $f_1, ... f_m$ be binary functions called duplicate detection functions, where each f_i takes a pair of tuples in R_i and returns *1* if they are duplicates, and *0* otherwise. Let $r=[r_1, ...r_m]$ and $s=[s_1, ...s_m]$ be two entities. We say that r is a duplicate of s if and only if $f_i(r_i, s_i)=1$ for all $i=1 \div m$. Therefore, a straightforward duplicate detection algorithm would be to independently determine sets of duplicate tuples at each level of the hierarchy and then to determine duplicate entities over the entire hierarchy. As we move down the hierarchy, the reduction in the number of comparisons is significant.

For the hypothetical example, we can process each of the R_1, R_2 and R_3 relations independently, to determine duplicate pairs of tuples in these relations. We may then identify pairs of duplicate entities if their corresponding tuples at each level in the hierarchy (K1, K2, K3 and K4) are either equal or duplicates.

As we mentioned, the difference between this filter and the former consist in detection of equivalence errors. So, the level of similarity between two fields, F_1 and F_2, noted *Sim(F1,F2)*, is given by: $Sim(F1,F2)=val[val(Sim_{cont} - \sigma_{cont}) + val(Sim_{occ} - \sigma_{occ})]$, where:

- $val(x):R \rightarrow \{0,1\}$ is a function defined as follows: $val(x) = \begin{cases} 1, \text{if } x > 0 \\ 0, \text{otherwise} \end{cases}$
- Sim_{cont} represents the containment similarity

- σ_{cont} is the containment threshold
- Sim_{occ} represents the co-occurrence similarity
- σ_{occ} is the co-occurrence threshold

This expression cumulates two kinds of similarities, textual and containment, and two kinds of threshold, also textual and containment.

We assume that each tuple v can be split into a set of tokens (words) using a tokenization function (say, based on white spaces). The *containment similarity metric* is given by the fraction of v_1 tokens that v_2 contains. Depending on the domain characteristics, this similarity may be substitute with textual similarity.

If $i > 1$, we say that two tuples, v_1 and v_2, in R_i *co-occur* through a tuple v in R_{i-1} if they both join with v. Note that while measuring co-occurrence between two tuples in R_i we only use R_{i-1} . The restriction improves efficiency because the number of distinct combinations joining with a tuple in R_i increases as we go further down the hierarchy. The *co-occurrence similarity metric* is given by the containment similarity metric between the children sets.

A more efficient technique is to process a parent relation in the hierarchy before processing its child. After we process the topmost relation, we group the child relation below into relatively smaller groups and compare pairs of tuples within each group. In this way, the comparison will be finished at that level of hierarchy where a non-duplicate pair of tuples occurs. This is a top-down traversal of the hierarchy. Otherwise, same sets of tuples in R_i may be processed in multiple groups causing repeated comparisons between the same pairs of R_i tuples.

Conclusions

Time is critical in cleansing large databases. In this paper we first present a simple comparison method, based on textual similarity. The other detection method is based on triangle inequality property. So, we filter out a lot of unnecessary comparisons. The last filter exploits the dimensional hierarchies in data warehouses in order to detect duplicates in dimensional tables. Depending on the characteristics of the content of databases, we can choose between these methods or develop a mixed technique.

References

[Anan02], Ananthakrishna, R., Chaudhuri, S., Ganti, V.: *Eliminating Fuzzy Duplicates in Data Warehouses,* In Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, China, 2002.

[Han02], Han, J., Kamber, M.: *Data Mining: Concepts and Techniques,* Publisher: Wiley, John & Sons, Incorporated, March 2002.

[Mong00], Monge, A.E.,: *Matching Algorithm within a Duplicate Detection System,* In IEEE Data Engineering Bulletin, volume 23(4), December 2000.

[Nati02], National Technical University of Athens: *Modeling Data Warehouse Refreshment Process As a Workflow Application*, White Paper, 02 APR 2002.

[Rama01], Raman, V., Hellerstein, J.M.: *Potter's wheel: An Interactive Data Cleaning System,* In Proceedings of the 27th International Conference on Very Large Databases, Rome, 2001.

[Redm01], Redman, T.C., Daugherty, M., Daugherty, M.: *Data Quality: The Field Guide,* Publisher: Elsevier Science & Technology Books, January 2001.

[Sung02], Sung, S., Li, Z., Sun, P.: *A Fast Filtering Scheme for Large Database Cleansing,* In Proceedings of the eleventh international conference on Information and knowledge management, McLean, Virginia, USA, pages: 76 – 83, 2002.

[Wang95], Wang, R.Y., Storey, V.C., Firth, C.P.: *A Framework for Analysis of Data Quality Research,* In IEEE Transactions on Knowledge and Data Engineering, 7(4), pages: 623-640, 1995.