Increasing the performance of an application for duplication detection

Eurorisk Systems Ltd., General Kiselov str. 31, 9002 Varna, Bulgaria

Abstract: *This paper examines performance issues of a software application that identifies duplicated records in a customer information database. It discusses approaches, logic and algorithms, analyzes essential research papers on this topic and debates problems and expected performance gains.*

Key words: *Data Cleaning, Duplicate Detection, Data Mining, Record Linkage, Edit Distance Algorithm, Artificial Intelligence, Computer Systems and Technologies*

INTRODUCTION

Problems of data cleaning oftentimes arise in large, and even middle-sized, organizations. Collecting data of objects, persons or activities, especially ones that come from different sources, can lead to this problem quite frequently. There are numerous works focusing on developing appropriate algorithms and using specially designed software for the performance of such a task.

This article [1] describes an approach for the detection of duplicate algorithms in a medium-sized international bank. This feature, built as a module within a larger project, was designed to detect potential duplicates from data obtained from different front office systems and located in one date warehouse. The analysis of data records and algorithms, that is required in order to compare and determine potential duplicates, is being realized via CLIPS rules. One important requirement for the application is to provide answers regarding potential duplicates as quickly as possible. It is to be used via the web GUI interface for online duplication checks. This article [1] describes the problems that need to be solved, the algorithm, the realization and some of the main researches conducted in this area.

Another module in the application uses the discussed approach to perform a thorough analysis, carrying out a comprehensive search through all customer rows in the database to localize duplicated rows. This article focuses on problems, solutions and optimizations that have been realized during the design and development of this project.

OFFLINE DUPLICATION SEARCH TASKS

The described approach is realized as a software module, which starts in an offline batch mode. The following tasks within this duplication search module have been defined:

- Up to 2 million records, describing customers from different institutes, have been collected into the database.
- 14 data fields are defined as essential and are used for the duplication search.
- Character strings from different character sets are enclosed.
- The goal of the search operation is to form customer groups with high degrees of similarities and assemble those groups into a single company customer.
- The complete research of the entire database is to be completed within few days.

Different approaches gave different performance results:

The comparison, according to the brute force method, is not admissible in this case because one would compare each customer to all the others. For 100.000 customers and 14 data fields, the total number of runs for the edit distance algorithm is 100.000 x 100.000 x 14 = 140.000.000.000.

The proposed solution uses flexible SQL instructions to reduce the size of the window to amount between 100 and 300 rows [1]. In the case of 100T customers, this means 100T x 200 x 14, i.e. 280.000.000 Edit Distance algorithm runs, that is 500x faster.

This article discusses an approach that improves the performance. In the proposed algorithm, clusters of duplicates are detected, mutual similarities calculated and clusters

excluded from further search operations. In this way the search area, as well as the number of the Edit Distance runs, is exponentially reduced.

CURRENT SOLUTION FOR THE DUPLICATION SEARCH

The proposed solution has been described in detail in [1]. This section will cover some basic points that are essential for the algorithm and discuss only the performance aspects.

Data Screen Size

The search of duplicates can be started for each institute customer against all other institute customers. The result per institute customer is represented in the form of a list of potential duplicates with a coefficient of similarity, expressed in %.
The data screen size defines how many data records are to be compared with the sample data record in each case. If there is one single criterion and it produces two data records that are too far apart from each other, the second one will not be included in the data screen and won't be recognized as a potential duplicate. If, in order to avoid this, the screen becomes too wide, there will be many meaningless comparisons which will affect the performance. The selected approach must use several criteria when selecting a list of potential duplicates. In this way, the size of the data screen can be kept small due to the implementation of a 'multi-pass approach' [3]. The number of data records to be compared is requested from the data base. The following conditions must be fulfilled:

- The 14 relevant data fields for the duplication search are held in the data base in both the original and the converted form (7-bit ASCII, small letters). The specially designed data table, indexed with binary (bitmap) indices, is used to keep the converted form of the strings. The data in this table is needed for all the steps required for the duplication process [1].
- Logical functions are applied using dynamic SQL instruction over the fields of the converted form table. In the first step, the size of the data screen (suitable number of rows for duplication search) is determined. Logical functions determine the size of the data screen based on the importance (priority) of data fields. If the number of selected data records is too large, the logical function changes the selection, straightening the constraints. If the number is too small, the constraints widen. The structure of logical functions is operated by the search logic. The goal is to determine the number of potential duplicates lists – using bitmap indices in the data base and flexible logical operators (and, or) – without first having to retrieve the data records themselves.

Example: The permitted size of the data screen is defined as [100 .. 300] data records.
The following „where "- clause of an SQL instruction for COUNTS determines two potential duplicates (too few for the data screen size):

```
NAME='Wex & Wex Gruen GmbH' or BRIEF_NAME='Wex & Wex Gruen GmbH' or
FULL_NAME='Wex & Wex Gruen GmbH' or FOUNDING_BIRTH_DATE={d'1900-12-24'}
and STREET='Fortnerstr 193' and( CITY='Graz-Strassgang' or POSTCODE='8054') and
REGISTER_NO='58449F' andIDENT_NUMBER='88429' and        SEX='0'
```

After replacing the first „and "(see above) with „or " in the search logic, the "where "- clause extends the constraints and 147 potential duplicates are determined:

```
NAME='Wex & Wex Gruen GmbH' or BRIEF_NAME='Wex & Wex Gruen GmbH' or
FULL_NAME='Wex & Wex Gruen GmbH' or FOUNDING_BIRTH_DATE={d'1900-12-24'}
or STREET='Fortnerstr 193' and  ( CITY='Graz-Strassgang' or POSTCODE='8054') and
REGISTER_NO='58449F' and IDENT_NUMBER='88429' and SEX='0'
```

The size of the potential duplicates list (147) ranges between 100 and 300. The data screen size is determined and data records are taken from the data base.

Comparison of data records

Data for the selected customer is compared to records from the data screen window. For each data record in the data screen, an evaluation of similarities is computed. The evaluated data records are presented to the user to help him make a final decision. During the duplication evaluation process [1], individual data fields are compared using the edit distance algorithm [7,8]. The results for the separate fields are weighted and combined together in order to generate the final coefficient of similarity:

$$\text{Result} = \sum (\text{Weight}(\text{Feld i}) * \text{Similarity}(\text{Field i})) / \sum (\text{Weight (Field i)}),$$

where

$$\text{Weight}(\text{Field i}) = \text{Max}(\text{Priorities}) + 1 - \text{Priority}(\text{Field i}),$$

$$\text{Total Similarity} = 1 - \prod (1 - \text{SimilarityCoefficient}(\text{Field i}))$$

The maximum similarity result in the case of equality is 100%. If a previously defined minimal value is reached, the data record can be rated as a potential duplicate. If values for a field with high priority have small or no similarities (if data for a field is equal to 0, it is treated as unknown and not equal), a high negative evaluation is assigned, which can't be compensated by adding the weighted evaluations of the remaining comparison elements. The result in this case is lower than 100%, independent of the fact that the remaining fields could potentially reach the similarity of 100%.

Performance of the proposed solution

On the test Sun server, the proposed solution reaches a performance of 3.500 to 3.900 customers per hour. The goal is to increase this performance significantly.

INCREASING OF PERFORMANCE USING CLUSTERING

Principles

The idea is to locate clusters of similar customer rows, group them and exclude members from later searches and comparisons. The following fundamental aspects are important:

- The similarity from A to B is commutative, i.e. similarity (A, B) = similarity (B, A). This is connected to the meaning of the Levenshtein distance [4], used for the measurement of the Edit Distance algorithm [5,6], which is defined as the minimum number of insertions, deletions or substitutions necessary to transform one string into another.
- Customers with levels of similarity higher than previously defined (for example 50%) must be grouped together in order to create a cluster and be capsulated. This is based on the assumption that customers in a cluster have smaller similarities compared to all other customers, thus it doesn't make any sense to calculate the similarity coefficients. The group can be considered as complete and excluded from further searches. In this way, the number of search operations in the search area will be reduced.
- It must be possible to calculate internal similarities between cluster member rows, based on the similarities between one member and all others. For example, after calculating the similarity coefficients between (A,B), (A,C), (A,D) one must be able to determine the similarity between (B,C),(B,D),(C,) without a re-run of the Edit Distance algorithm, thus reducing the number of customer comparisons. Table 1 illustrates one such example of this principle:

		Sim	Dev	Var		Sim			Sim			Sim
a	a	1,00	0,00	0,00	b	a			c	a			d	a	
a	b	0,80	0,20	0,04	b	b	1,00		c	b			d	b	
a	c	0,60	0,40	0,16	b	c			c	c	1,00		d	c	
a	d	0,70	0,30	0,09	b	d			c	d			d	d	1,00

Table 1.: Cluster of customers with higher degrees of similarity

where

Sim = Similarity
Dev = Deviation = 1 – Similarity
Var = Variance = Deviation ^2

In the example above four customers (A, B, C, D) are given. The standard algorithm for the duplication search [1] produces the following similarities: (A,B) = 0,8; (A,C) = 0,6; (A,D) = 0.7, and (A,A) = 1, which is trivial and obvious. The level for clustering is defined to 50%. All the calculated similarities are greater than the selected level. This means that the 4 rows will form a cluster, while all remaining rows will have similarities to the cluster members, which are below this border. Several mutual similarities are missing however (see empty fields in Table 1. (B, C), (C, D)). The computing of those missing similarities will save 9 comparison cycles, i.e. with only 3 comparisons (customer A to all others) 12 comparisons will be covered. The performance gain depends on the size of the cluster – the higher the better.

Volatility Algebra

Volatility Algebra is used to determine the mutual similarity of coefficients in terms of risk calculation [2] (e.g. in VaR/CoVaR). The RiskMetrics volatility and correlation data are related to price changes of underlying market variables, not to the values of variables, i.e. the volatility of price changes and correlations reflects the correlation between price changes of two market variables that are being dealt with. It doesn't matter whether the variables correlate to each other; important is the correlation between their price changes. In this case, the standard deviation of price changes that are assumed to be normally distributed could be interpreted as an accidental unsystematic error in the time series. This requires the usage of algebra for the calculation of the compound error, which arises in complex formulas of stochastic variables. If two stochastic variables are arguments of a arithmetic function:

$$z = f(x, y),$$

then the compound error obtained from the unsystematic error of the two stochastic variables is given as an approximate estimation (ignoring higher order products):

$$V_z^2 = \left(\frac{\partial z}{\partial x}\right)^2 * V_x^2 + \left(\frac{\partial z}{\partial y}\right)^2 * V_y^2 + 2 * \frac{\partial z}{\partial x} * V_x * \frac{\partial z}{\partial y} * V_y * Corr_{x,y}$$

From the formula above one can see that, for the given data of two normally distributed simple or compound objects:

First object:	X	Second object:	Y
Absolute Volatility:	V_x	Absolute Volatility:	V_y
Current Value:	x	Current Value:	y

Absolute Covariance: $Covar_{xy} = V_x * V_y * Corr_{xy}$

the following compound objects (Z: Absolute Volatility V_x, Current Value z) are derived using stochastic arithmetic operators (Table 2):

Sum operator: **Z = X + Y**	**Multiplication** operator: **Z = X * Y**
$z = x + y, \quad V_z = \sqrt{V_x^2 + V_y^2 + 2 * Covar_{xy}}$	$z = x * y, \quad V_z = z * \sqrt{\frac{V_x^2}{x^2} + \frac{V_y^2}{y^2} + 2 * \frac{Covar_{xy}}{x * y}}$
Difference operator: **Z = X - Y**	**Division** operator: **Z = X / Y**
$z = x - y, \quad V_z = \sqrt{V_x^2 + V_y^2 - 2 * Covar_{xy}}$	$z = x / y, \quad V_z = z * \sqrt{\frac{V_x^2}{x^2} + \frac{V_y^2}{y^2} - 2 * \frac{Covar_{xy}}{x * y}}$

Table 2: Compound operators

Sum and Difference operators use the absolute volatility of two objects to calculate the volatility of the resulting object. On the other hand, Multiplication and Division operators involve relative volatilities, obtained by dividing the absolute volatility on the current values.

Computation of similarities

This section presents a method for the computation of mutual similarities from determined similarities. Based on the sample above, similarity (A, B) = 0.8 and similarity (A, C) = 0.6 will be used to determine similarity (C, B). The steps for the computation are shown in Fig 1.

$$Deviation_{(C,B)}) = \sqrt{Dev_{(A,B)}^2 + Dev_{(A,C)}^2 - 2 * Dev_{(A,B)} * Dev_{(A,C)} * Correlation_{(A,C;A,B)}}$$

The minus sign (-) in the formula is due to the negative behavior correlation. The computation of the similarity (C, B) with the sample data and a correlation coefficient of 0,5 leads to the following result:

Deviation(C, B) = SQRT(0,04+0,16-0,08) = SQRT(0,12) = 0,35
Similarity(C, B) = 1 – 0,35 = 0,65

If the correlation coefficient C = 0 is selected,

Deviation(C, B) = SQRT(0,04+0,16) = SQRT(0,20) = 0,45
Similarity(C, B) = 1 – 0,45 = 0,55

If the correlation coefficient C = 1 is selected,
Deviation(C, B) = SQRT(0,04+0,16-0,16) = SQRT(0,04) = 0,2
Similarity(C, B) = 1 – 0,2 = 0,8

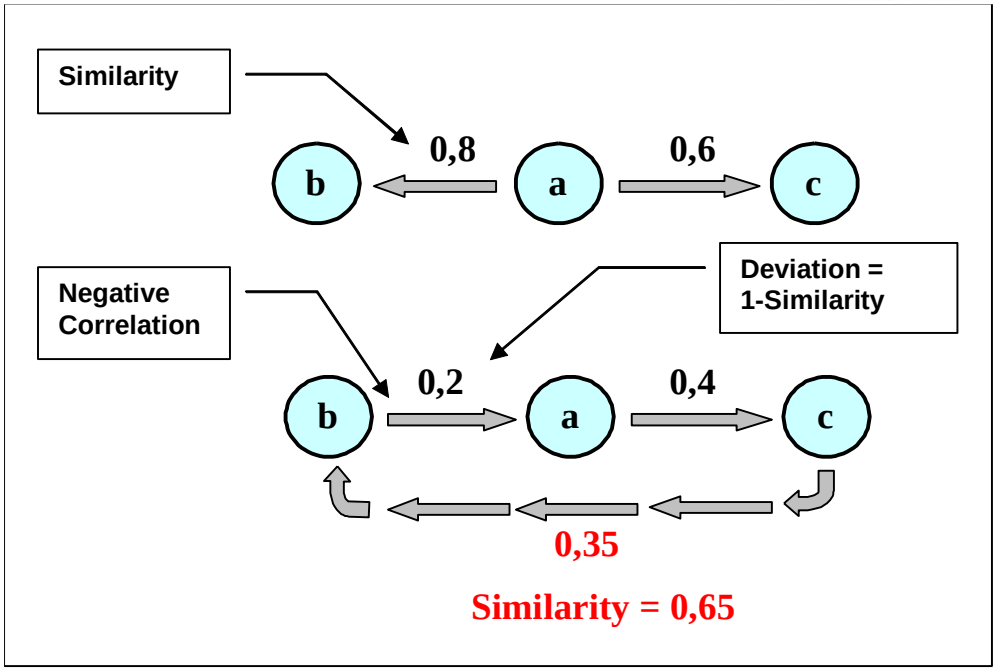

Figure 1

The results of all calculations, based on the discussed principles and correlation coefficient = 0.5 are given in Table 3.

	Sim	Dev	Var		Sim		Sim		Sim
A a	1,00	0,00	0,00	b a	0,80	c a	0,60	d a	0,70
b	0,80	0,20	0,04	b	1,00	b	0,65	b	0,74
c	0,60	0,40	0,16	c	0,65	c	1,00	c	0,64
d	0,70	0,30	0,09	d	0,74	d	0,64	d	1,00

Table 3: Cluster of customers with a higher degree of similarity

The following considerations and conclusions can be defined for the selected approach:

- The correlation cannot be negative, it has values between 0 and 1.

	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95	1,00
0,50	0,50	0,52	0,54	0,56	0,56	0,57	0,56	0,56	0,54	0,52	0,50
0,55	0,52	0,55	0,57	0,59	0,60	0,61	0,61	0,60	0,59	0,57	0,55
0,60	0,54	0,57	0,60	0,62	0,64	0,65	0,65	0,65	0,64	0,62	0,60
0,65	0,56	0,59	0,62	0,65	0,67	0,69	0,70	0,70	0,69	0,67	0,65
0,70	0,56	0,60	0,64	0,67	0,70	0,72	0,74	0,74	0,74	0,72	0,70
0,75	0,57	0,61	0,65	0,69	0,72	0,75	0,77	0,78	0,78	0,77	0,75
0,80	0,56	0,61	0,65	0,70	0,74	0,77	0,80	0,82	0,83	0,82	0,80
0,85	0,56	0,60	0,65	0,70	0,74	0,78	0,82	0,85	0,87	0,87	0,85
0,90	0,54	0,59	0,64	0,69	0,74	0,78	0,83	0,87	0,90	0,91	0,90
0,95	0,52	0,57	0,62	0,67	0,72	0,77	0,82	0,87	0,91	0,95	0,95
1,00	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95	1,00

Table 4: Similarity Coefficients for correlation coefficient C = 0.5

- Investigations using sample data have shown that a correlation of 0,5 produces a good approximation of real results. The computation function for the similarity between B and C with correlation = 0.5 and different similarities for (A, B) (perpendicular) and (A, C) horizontal) is shown in Table 4 and Fig. 2.

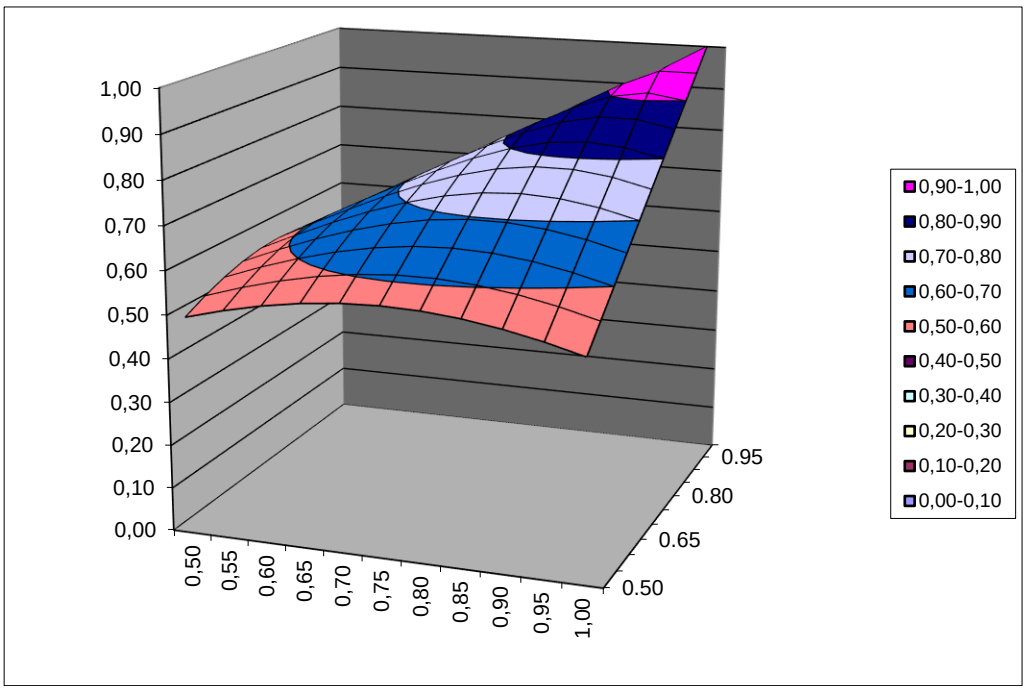

Figure 2

By selecting the correlation of 0,5 some plausible characteristics are brough about:

- The computed similarity is not lower than the similarity of the two input similarities (see Table 4).
- If the two input similarities are equal, the computed similarity too is equal to them (Table 4, the diagonal).

Estimation of the performance increase

The reduction of the number of reading and comparison operations depends on the number and size of clusters. The more duplication clusters are detected and the bigger they are, the greater the performance gain.

If there is no possibility of forming clusters, there is still performance gain, due to the usage of Similarity(A,B) = Similarity(B,A).

CONCLUSIONS AND FUTURE WORK

The discussed approach was realized as a small test module. The following conclusions were drawn:

- The preparation tasks, involving data modification, must be performed on the database server. This approach enables a better performance.
- To attain the suggested performance results in real-time, additional databases and binary indices must be prepared during the previous night batch mode.
- Data for already calculated similarity coefficients is being kept in hash tables in the memory. This is a quick way to determine whether a particular pair of rows has

already been compared.

REFERENCES

[1] P.Paskalev, A.Antonov. Intelligent Application for Duplication Detection, In Proceedings of the International Conference on Computer Systems and Technologies CompSysTech 2006, IIIA.27.1 – IIIA.27.8, 2006.
[2] J.P.Morgan/Reuters RiskMetrics Technical Document, J.P.Morgan/Reuters Ltd. 1996.
[3] M. A. Hernandez and S. J. Stolfo. The merge/purge problem for large databases. In Proceedings of the 1995 ACM SIGMOD, pages 127–138, San Jose, CA, May 1995.
[4] Bilenko M., Mooney R. Adaptive Duplicate Detection Using Learnable String Similarity Measures In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), Washington DC, pp.39-48, August, 2003
[5] J.J. Zhu and L.H. Ungar. String Edit Analysis for Merging Databases. In proceedings of International Conference on Knowledge Discovery and Data Mining (KDD) 2000.
[6] I. P. Fellegi and A. B. Sunter. A theory for record linkage. Journal of the American Statistical Association, 64:1183–1210, 1969.
[7] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. Journal of Molecular Biology, 48:443–453, 1970.
[8] C. Sander and R. Schneider, "Database of homology-derived protein structures and the structural meaning of sequence alignment," Proteins, vol. 9, no. 1, pp. 56--58, 1991.

ABOUT THE AUTHORS

Dr. Eng. Anatoliy Antonov
Eurorisk Systems Ltd.
E-mail: Antonov at eurorisksystems dot com

Eng. Plamen Paskalev
Eurorisk Systems Ltd.
E-mail: ppaskalev at eurorisksystems dot com