Journal of Software Engineering and Applications, 2017, 10, 529-545
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Fast Semantic Duplicate Detection Techniques in Databases

Ibrahim Moukouop Nguena[1], Amolo-Makama Ophélie Carmen Richeline[2]

[1]National Advanced School of Engineering, University of Yaounde I, Yaounde, Cameroon
[2]Computer Sciences Department, University of Yaounde I, Yaounde, Cameroon
Email: imoukouo@gmail.com, ophedinho@live.fr

How to cite this paper: Nguena, I.M. and Richeline, A.-M.O.C. (2017) Fast Semantic Duplicate Detection Techniques in Databases. *Journal of Software Engineering and Applications*, **10**, 529-545.
https://doi.org/10.4236/jsea.2017.106029

Received: January 1, 2017
Accepted: June 16, 2017
Published: June 19, 2017

Abstract

Semantic duplicates in databases represent today an important data quality challenge which leads to bad decisions. In large databases, we sometimes find ourselves with tens of thousands of duplicates, which necessitates an automatic deduplication. For this, it is necessary to detect duplicates, with a fairly reliable method to find as many duplicates as possible and powerful enough to run in a reasonable time. This paper proposes and compares on real data effective duplicates detection methods for automatic deduplication of files based on names, working with French texts or English texts, and the names of people or places, in Africa or in the West. After conducting a more complete classification of semantic duplicates than the usual classifications, we introduce several methods for detecting duplicates whose average complexity observed is less than O(2n). Through a simple model, we highlight a global efficacy rate, combining precision and recall. We propose a new metric distance between records, as well as rules for automatic duplicate detection. Analyses made on a database containing real data for an administration in Central Africa, and on a known standard database containing names of restaurants in the USA, have shown better results than those of known methods, with a lesser complexity.

Keywords

Semantic Duplicate, Detection Technique, Detection Capability, Automatic Deduplication, Detection Rates and Error Rates

1. Introduction

When two lines in a database have different identifiers while they represent the same physical reality, we call them semantic duplicates, for example, having the same employee represented twice in the company's file, with two different per-

sonnel numbers. The lines which follow give some examples of semantic duplicates:

ets youdim commerce general
youdim sarl commerce general
amougui veuve zong claudine bp douala
amougui vve zong claudine commercant
sohdidjo muifo gilles armel bp 99995
sohdjo muifo gilles armel bp 99995

Deduplication is the complete process from detection to removal of duplicates records. The duplicates treatment in a database is very important and necessary regardless of the action to be undertaken on the data. Duplicates are on average 4% of the data in the databases [1]. When the size of the database becomes larger, their retrieval becomes more expensive and difficult.

Duplicates are the cause of many problems that have a significant impact. For example, in an organization, if an employee is represented several times in the database of payroll, obviously he will have as many salaries as represented in the database, representing a loss for the organization. Duplicates also falsify the statistics because of their presence in the data used.

Duplicate detection is a very important phase in the deduplication process. To detect duplicates effectively, we need an effective technique which detects as many duplicates as possible in a reasonable time, while making the least possible mistakes. Some authors [2] [3] [4] [5] and [6] have proposed techniques based on the principles of blocking, canopy or clustering. They use these principles to reduce the complexity of the detection, for example forming blocks of similar records to limit the number of comparisons between records (principle of blocking). However, they did not work on real data for large volumes or evaluate their performance from a multicultural environment. Moreover, canopy and clustering techniques are difficult to use in transactional applications to prevent the creation of duplicates.

Peter Christen [7] realized a study comparing diverse techniques of deduplication on the following 5 criteria: recall, the number of candidate pairs, execution time, and memory space used, reduction ratio. He worked with datasets which were generated according to certain rules, and clearly establishes the necessity of comparing the methods on an empirical database with real data while looking for the best parameters of the diverse methods. In this work, we also use techniques based on the principle of blocking, but we work with real databases. One of these databases concerns the importers of Central Africa, with names based on French, English and numerous local dialects. No study to our knowledge has been done previously on real data reflecting such diversity.

Furthermore, we study the precision of the envisaged methods, as well as a new indicator of global accuracy not mentioned in Christen's work. We did not retain in our studies the execution time and the memory space. Indeed, for the studied methods, the execution time is directly proportional to the number of candidate pairs. The used algorithms are such as the requested memory space is E=O(N+D), where N is the size of the treated data and D the actual number of

duplicates found.

Metha and al. [8] evaluated the quality of a number of similarity functions on synthetic datasets using a measure used in approximate querying called discernability.

To compare the duplicate detection techniques, most commonly used metrics are the recall, the precision, the number of pairwise comparisons and the f-score. The f-score is a measure of the global efficiency, which is the value given by the harmonic average of recall and precision. We didn't see the justification for this choice. We propose a mathematical model for measuring the global efficiency, resulting in a new indicator which we call the global accuracy.

This paper is organized as follows. We begin with an improved classification of semantic duplicates, followed by duplicate detection approaches used in our techniques. Then we propose different blocking functions associated with introduced techniques. We present the criteria used to compare these blocking functions, namely the statistical entropy, the number of pair comparison (estimation of complexity) and the average size of a block.

For the duplicate detection step, we introduce a new metric of distance calculation between the character strings, which we designated by the term *global distance*. We then use a set of built rules using the global distance to decide whether we are dealing with a duplicate or not. For each blocking technique proposed, we calculate the detection rate (recall), the error rate (precision), and global accuracy. Finally, we carried out practical experiments on two databases, one corresponding to individuals and businesses in Central Africa and the other corresponding to the "restaurant database" used by several authors [1], and containing the names of restaurants in different cities in the USA. The use of this database allows us to compare our results with those of previous work done on the same data.

2. Materials and Methods

2.1. Duplicates Global Classification Proposition

In the literature, some authors, for example [9], give a classification of variations that could lead to duplication. This classification is as follows:

- Spelling error,
- Replacing characters,
- Abbreviations and acronyms,
- Translation (linguistic Synonym),
- Missing Values (deleting characters),
- Truncation,
- Token merge,
- Token transposition.

Nevertheless, some types of variations that we met during our analysis on data have not been included in these classifications:

- Perfect inclusion (expression composed of several expressions or existing words, eg: **"thrift shop"** is perfectly included in **"thrift shop + Carpet and**

Coatings"),

- Semantic inclusion (whose expression is a general term for a set of other existing expressions, eg: **"Telephone"** is semantically included in **"appliances"**),
- Synonyms (example: **"Spare parts"** is a synonym of **"Diverse spare parts"**),
- Noise (terms that have mistakenly complement number or article, eg: **"139 diverse parcel"** is a noise of **"Diverse parcel"**).

The inclusion of these elements allows a better definition of procedures for data standardization to be used during the process of duplicate detection.

2.2. Duplicates Detection Process

Detection identifies all pairs of duplicated candidates; it is very important because it is what provides the candidates for deduplication. According to studies on the duplicates detection, we will retain that the duplicate detection process usually consists of three main phases: pretreatment, blocking and comparison (Match). Here we describe our practical approach of these steps (see Figure 1).

2.2.1. Pretreatment Step

We assume that we are working with structured data and we also assume that all fields have adequate data types. The pretreatment comprises the following steps:

- **Sampling**: This is to provide a representative sample of the data to deduplicate, or of the type of these data.
- **Visual analysis of the sample**: This analysis is performed by scanning through all the sample data in order to identify the operations of standardization and normalization to be made on the data to be normalized (date formats standardization for example), and secondly to identify "stop words". These operations are necessary for the consideration of the syntactic differences important for the real semantic duplicates. "Stop words" are words that will not be taken into account in the calculation of the blocks during the blocking phase, as it is believed that they are not meaningful for this operation. The Table 1 gives some examples of duplicates records to use for

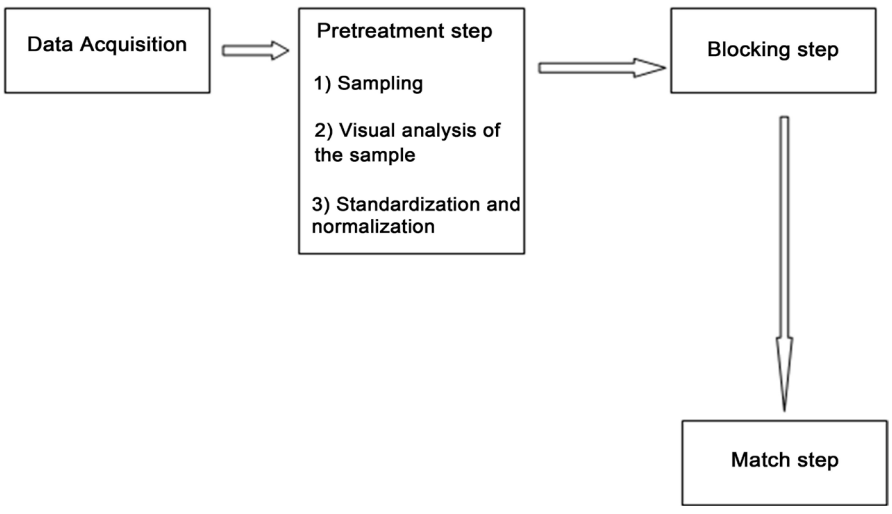

Figure 1. Duplicate detection process flow.

Table 1. Some examples of duplicates records.

ets youdim commerce general	youdim sarl commerce general
concept sarl bp 01456	ste concept sarl imprimerie edition
socafi sarl commerce general	ste socafis sarl commerce general
calorix sarl bp 142 yaounde	ets calorix bp yaounde
njiya ousmane chirurgien dentiste	njiya ousmano chirurgien dentist
amougui veuve zong claudine bp douala	amougui vve zong claudine commercant
sohdidjo muifo gilles armel bp 99995	sohdjo muifo gilles armel bp 99995
ngagni kinetcheu melanie lafortune commerce bonanjo	ngagni kwetcheu melanie lafortune bp douala
societe soeurs cameroun import-export	societe soeurs cameroun sarl bp 11380
valittu serge gaetan bp douala	valittu serges gaetan bp 00222
ets tchami and sons bp 1119	ets tchami et son bp 1119
ocean agri-tech industrie de transf.	ocean agri-tech sa bp 038
ombe bp 015	ombe sarl commerce general
veto prestation sarl distrib.pdts pharma.	veto prestations bp 082
la globale des travaux bp 02374	ste globale des travaux bp douala
ets global trading consulting prestation de sces	global trading consulting bp douala
ets kouontche jean marie bp 081	kouontche jean marie bp 081
ets tiokour ousminau bp 999	tiokour ousminau bp 999

visual analysis.

- **Standardization and normalization**

It is used to normalize data before blocking and comparison. The main works carried out at this stage are:

- The sorting of the words, which consists of putting in alphabetical order the words of a given name, in order to recognize the occurrences of this name, even if the words that compose it are written in a different order.
- Replacement of multiple spaces between words by a single space.
- The conversion of the whole of each name in uppercase.
- Replacement of reference strings (city names for example) by normalized values. For example, for the restaurant database, the term "city" after the name of a city is removed. Thus, "New York City" will be replaced with "New York".
- Any other normalization operation deemed necessary after visual analysis phase (date normalization for example).

Analyzing these real cases examples helps to understand the need for this step, and to deduce normalization rules.

2.2.2. Blocking Step

The blocking step consists of dividing the data set in small blocks of similar data that could be duplicates, in order to reduce the pair comparison number. In

practice, several approaches can be used for blocking. In this work, we are only interested in the approaches based on blocking functions which take only the used record as input to determine its block. We will call them blocking functions by hashing. Nevertheless, we compare their performance with the results in the literature relating to other blocking functions.

These approaches are of particular interest, in that the computing time of a block does not depend on the size of the dataset. Therefore, they can be efficiently implemented in management software, at the transaction level, to automatically detect the potential duplicates when creating records. They consist of a set of operators which are applied to each record. Each operator applied to a record produces a key which is the code of the record. A function (blocking function) groups the keys produced by the used operators to give a unique key considered as block identifier of processed record. This block identifier is added to the record as a new field and is kept in the database for later use.

Assuming that for the detection of duplicates only the elements of the same block are compared, the choice of the blocking function is critical. With a bad function, the rate of duplicates ending up in a different block can be significant, resulting in bad recall during deduplication. Similarly, a function which discriminates few will make very large blocks, increasing the number of comparisons, and therefore the complexity of the algorithm. That's why in this work we study several blocking functions by hashing, in order to deduct recommendations on their strengths and their weaknesses.

Let us note that all block hashing techniques performance depends on the distribution of errors in real data. Therefore, the methods must be compared to a sample of actual data representative of the usually treated data. This is what we are doing here through the choice of experimental data. This article does not merely propose methods but studies their behavior on real data in order to derive recommendations.

2.2.3. Match Step

The third step, the Match step, consists of comparing pairs of data to say whether they form a duplicate or not. Here we compare the pairs belonging to the same block. This step, in general, uses some metrics of similarity distance calculation between the pairs of records. A similarity distance metric is a function which takes as input two records and returns a value considered as their similarity distance. In our work, this phase uses rules of detection in addition to the metric of similarity distance calculation. This phase is considered as a bottleneck because it becomes difficult and very expensive when you have very large blocks.

At the end, the phase of comparison produces the equivalence classes of duplicates. An equivalence class is a set of equivalent records relative to an equivalence relation. The set of criteria used to decide that two records are duplicates constitutes the equivalence relation of our classification. The transitivity of the relation is imposed by the fact that if we have three rows A, B and C, if A is equivalent to B and B is equivalent to C, we immediately deduce that A and C are equivalent and we place them in the same class, without attempting to calcu-

late the similarity distance metrics for the couple (A, C).

Each equivalence class is given by an identifier, and to every element of this class, we associate the value of this identifier. This identifier can be constructed by selecting an item in each class which will be designated as the canonical representative of the class, and by taking its ID. The value of this identifier is such that in a single query we can find all the elements of every class, which is very useful for all treatments that may follow, such as eliminating duplicates.

In order to carry out this phase quickly, two algorithms are possible. The first merely sorts the data into a list using the codes of blocks. Then for the comparison, we take every record and we compare it with the records which follow and which are in the same block. The advantage is that there is no need for additional memory space, and the inconvenience is that there is a time used for the sorting. The second algorithm browses the list of data and places elements in a hash table having as key the code of block and for value the list of the elements of the same block. The advantage is a smaller time than sorting, and the inconvenience is a slightly greater memory occupancy (total size of the codes of blocks + (B + N) P, where B is the number of blocks, N the number of Lines and P the size of a pointer). Nevertheless, this extra space remains small enough not to hold our attention for the rest of the work.

2.3. Blocking Functions Used

2.3.1. Soundex Technique

Soundex is a phonetic algorithm for indexing words by their pronunciation in British English [10]. The basic principle is to encode with the same chain the names with the same pronunciation, so as to find a similarity between them despite minor differences in writing. It is based on codes assigned to each string (four letters). These codes are based on pronunciation; the consonants having same pronunciation have the same code.

2.3.2. Metaphone Technique

Metaphone algorithm is similar to the Soundex algorithm because it codifies the words according to their pronunciation in English [11]. For words whose pronunciation is near, he created a similar key. Metaphone keys generated have a variable size, unlike the soundex which is limited to four letter size. It builds keys as soundex does and keeps itself (not limited to 4 characters or do not add 0 to obtain 4 characters).

2.3.3. Homonym Technique

Here we introduce the algorithm of the homonym; we are building it with some modifications on existing algorithms. Each name is associated with a code that is obtained in the following manner:

- Change the name in capital letters (uppercase);
- Replace the white characters (multiple or not) by a space character;
- Eliminate all “connectors” in the word (e.g.: and, the, of, etc.);

- Sort the words of the name in alphabetical order;
- Delete some word ends, corresponding to a predefined set of the word ends to be deleted (not pronounced, often forgotten);
- Change some word ends, by replacing them by their normalized equivalents, stemming from a predefined set;
- For each word remove all the vowels and replace all double consonants by one;
- Concatenate all codes obtained from each word.

The first twelve characters of concatenated codes constitute the homonym code of the record. In practice, the homonym being calculated on the name can be enriched by adding the year of birth, the city, and the homonym of the name of the father or the mother. This additional information helps to better differentiate between the persons with the same name.

Example :

The syllabes and ballons ⇒ THE SYLLABES AND BALLONS ⇒ THESYLLABESANDBALLONS ⇒ SYLLABES BALLONS ⇒ BALLONS SYLLABES ⇒ BALLONSYLLABE ⇒ BLN SLB ⇒ BLNSLB

2.3.4. Initials Technique

We introduce the algorithm of initials; it's an algorithm which allows to index words by their initials. For a string, this algorithm returns a code which is the concatenation of the initials of each word of the string. The code is computed as follows:

- If the string is only one word, smaller or equal to three characters, the code is represented by this word.
- If the string is a string of less than three words, the code is equal to the combination of the first two and last two characters of each word. If any of these words has a single character, this unique character will be used like its code.
- If the string is a string of more than two words, the code is the combination of each first character of each word.

2.3.5. Init Final Technique

This algorithm uses for each word the beginnings and ends of the word to build the code. Its principle is as follows:

- If the string is only one word, smaller or equal to three characters, the code is represented by this word.
- If the string is only one word of more than three characters, the code is equal to the combination of the first two and last two characters of the word.
- If the string has more words, the code is the combination of each first and last character of each word.

2.3.6. Dual Loop Algorithm

This algorithm is the naive algorithm comparing every record with all the others. It is an algorithm which consumes much time for its execution and thus has a very high complexity.

2.4. Match Step

2.4.1. The Metric of Similarity Distance Calculation

A similarity distance metric can be defined as a positive function which takes two records as input and returns a number which is the said distance of similarity between them. In the literature, we have as similarity distance metrics between two words the **Levenshtein distance**, the **Hamming difference**, the **n-gram algorithm**, the **Jacquard index**, etc. Here we introduce another distance similarity metric between two strings used in the remainder, the general distance.

2.4.2. General Distance between Strings

The principle of "General distance" algorithm is as follows:

For two given names (may consist of several words) if they are null then their distance is equal to 0. If one of them is null or empty then the distance is equal to the size of the word not null. If both are not null, a matrix of size (n, 2) with n equal to the number of words of the first name is established. The matrix in the first column contains the shortest distance of each first name word in relation to all available words of the second name, in the second column the size of the aforementioned word.

Firstly, we build the sets of words in each name; we respectively denote s1 and s2. For each word of s1 if the same word is present in s2 then the matrix is filled with a distance equal to 0 and both words are removed from the two sets. If the two sets are not empty, the distances of each word of s1 with all words of s2 are calculated (using, for example, Levenshtein distance), and then the matrix is filled with the smallest distance and the size of the corresponding word. The word of s2 used for the smallest distance and the corresponding s1 word are deleted. The action is repeated until one of the sets is empty. If it is s1 which is empty we stop, otherwise (s2 empty) the other words of the s1 matrix are filled with the distance equal to the size of every word. The general distance between the two strings is given by the sum of the distances in the first column of the matrix.

It may be noted that the distance thus calculated is not symmetrical $D(m1,m2)!=D(m2,m1)$ when the names have a different words number.

Example:

n_1 = small solid house $\Rightarrow S_1$ = {small, solid, house}

n_2 = house solid small $\Rightarrow S_2$ = {house, solid, small}

n_3 = liquid house

Let us calculate $D(S_1,S_2)$

$M_0=0$ using the word small, $M_1=(0\quad 5)$ using the word solid $M_2=\begin{pmatrix}0 & 5\\ 0 & 5\end{pmatrix}$ using the word house $M_3=\begin{pmatrix}0 & 5\\ 0 & 5\\ 0 & 5\end{pmatrix}$

$$D(S_1,S_2)=0+0+0=0$$

Let us calculate $D(S_1,S_2)$

$M_0=0$ using the word house, $M_1=(0\quad 5)$ using the word small

$$M_2 = \begin{pmatrix} 0 & 5 \\ l(\text{small}, \text{liquid}) & 5 \end{pmatrix} \text{ using the word solid } M_3 = \begin{pmatrix} 0 & 5 \\ l(\text{small}, \text{liquid}) & 5 \\ 5 & 5 \end{pmatrix}$$

$$D(S_1, S_3) = 0 + l(\text{small}, \text{liquid}) + 5$$

$l(\text{small}, \text{liquid})$ is a distance between the words small and liquid. In this paper, we use Levenshtein distance.

Note: the second column of the matrix is not used here, but is kept for future analysis.

2.4.3. Some Detection Rules Used

We give here the main rules used for the detection of duplicates. Let us recall that according to our approach, we always begin with a sampling of the set to be used and a visual analysis of the sample thus constituted. This analysis can lead to the production of new rules specific to the dataset used. These rules will come to complete the below-expressed rules.

Rule 1: Input errors

For two specific records (with the size of each record greater than or equal to three words), having one different word. The two records are equivalent if the similarity distance (e.g. Levenshtein distance) between the distinct words is less than $n/3$ (with n the size of the longest of both different chains).

This rule was obtained empirically, after a statistical analysis on a set of data representative of people in Central Africa. Considering the obtained results by maintaining this rule on data of restaurants in the USA, we maintain it as being a general empirical rule.

Rule 2: Perfect inclusion

For two given records possessing at least two words each, if the words set of the first contains all words of the second (or vice versa) then we shall say that both records are equivalent.

2.5. Comparison Criterion of Techniques for Duplicates Detection

Prior to the duplicate detection, we build the blocks of records using techniques described above. Blocking techniques will be compared on their algorithmic quality using three criterion:

1) Statistical entropy, which is given by:

$$E = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (1)$$

With n the total number of formed blocks and p_i the probability of belonging to a block i.

2) The average size of formed blocks,
3) The algorithmic complexity of de-duplication which is estimated by the average of the square of the sizes of blocks formed on, which is:

$$C = \sum_{i=1}^{n} \frac{sizei^2}{2n} \quad (2)$$

Duplicate detection techniques are then compared with regard to their effi-

ciency to detect duplicates, based on rules and metrics which we developed.

We do not display calculations time here, as they can vary greatly from one computer to another. On the other hand, this time can be written as $T = O(n \times C)$, where C is the algorithmic complexity defined above.

Finally, we use the detection rate and the error rate of every technique, to determine its efficiency and its reliability. The detection rate will be assessed through what is called here the recall (proportion of the total number of pairs of correct candidates duplicates found compared with the total number of pairs of true duplicates in the studied set). The error rates will be assessed by the precision of each technique. For a technique, the precision is the proportion of the total number of true duplicates found by this technique, compared to the total number of pairs of duplicates proposed by the same technique. The global effectiveness will be assessed through the global accuracy defined in the sequel. The f-score will be also estimated, in order to make comparisons with other works of the literature.

2.5.1. Evaluation of the Global Accuracy of a Detection Technique

For the evaluation of the global accuracy, we assume that losses when detecting a false duplicate and losses when not detecting true duplicates have the same cost. In our design, the global cost of the error is estimated by the recall, the precision and the total number of duplicates.

Let:

- Ntd: total number of true duplicates,
- Nsd_i: total number of duplicates suggested by the detection technique,
- $Ntsd_i$: total number of true duplicates suggested by the detection technique,
- $Nfsd_i$: total number of false duplicates suggested by the detection technique,
- E the total error.

Precision is the fraction of correct predictions among all pairs of citations predicted to fall in the same cluster:

$$P_i = \frac{Ntsd_i}{Ntsd_i + Nfsd_i} \tag{3}$$

The recall is the fraction of correct predictions among all pairs of citations that truly fall in the same cluster:

$$R_i = \frac{Ntsd_i}{Ntd} \tag{4}$$

We have:

$$E = (1 - R_i)\, Ntd + Nfsd_i \tag{5}$$

Using Equations (3)-(5), one deduces:

$$E = (1 - R_i)\, Ntd + \frac{(1 - P_i) R_i Ntd}{P_i} \tag{6}$$

The first component of the total error represents the cost of the error when a true duplicate was not detected and the second component represents the cost

error when a fake duplicate was detected. Using Equation (6), one deduces:

$$E = \left(1 - \frac{(2P_i - 1)R_i}{P_i}\right) Ntd \tag{7}$$

Since the global rate of errors represented by $\frac{E}{Ntd}$ equals $1 - \frac{(2P_i - 1)R_i}{P_i}$ we deduce that the global accuracy equals:

$$PG_i = \frac{(2P_i - 1)R_i}{P_i} \tag{8}$$

One can observe that if precision is less than 0.5, the global accuracy is negative. It means that the number of errors is greater than the number of true duplicates.

3. Results

In this section, we offer two comparisons: The first one uses confidential real data from a database in Central Africa, and the second uses a database of restaurants in the USA, which is available on the Internet for the purpose of comparison of deduplication techniques. This second comparison allows us to assess the performance and efficiency of proposed techniques compared to the best results in the literature using the same data.

As criteria for comparison, we note the statistical entropy, the complexity, the detection rate, the error rate and global accuracy.

For this first set of comparisons, we use a database of names in Central Africa, with a table containing 65,000 records.

We note that each of the proposed techniques has two versions: the simple version and the enriched version. The simple version involves the use of a single field of the table to form the blocks (name field), while the enriched version involves the use of two fields to blocking (the name and address). In this case, name and address are concatenated together into one attribute after data normalization. This new attribute is used during the blocking step.

This comparison can help to classify different techniques based on the statistical entropy, the average block size, and computational complexity. The Table 2 gives the results of this comparison.

The technique with the highest entropy is the enriched metaphone with 11.0795 and that having the smallest entropy is the simple initial with 8.1031. Knowing that the more the entropy is big, the more the blocks are homogeneous. We can conclude that from this point of view of statistical entropy the best technique is the enriched metaphone. For us, it translates more homogeneous blocks into size.

The average gives the average size of the groups formed from the blocking phase. The more the average size is small, the more records in the group are close. From the perspective of average size, the technique with the smallest average size is the enriched metaphone with 1.0075 and the one with the highest

Table 2. Comparison of techniques based on Entropy, Average of block size formed by every technique and Complexity (Algorithmic).

Blocking techniques	Entropy	Average	Complexity
Simple initial	8.1031	4.9849	316.0652
Enriched Initial	9.2924	2.4886	54.9253
Simple soundex	11.0113	1.0512	2.2648
Enriched Soundex	11.0794	1.0077	2.0315
Simple Metaphone	10.9987	1.0578	2.3191
Enriched Metaphone	11.0795	1.0075	2.0315
Simple Homonym	10.5533	1.0499	2.261
Enriched Homonym	10.6074	1.0114	2.0494
Simple InitFinal	10.6938	1.2401	4.0686
Enriched InitFinal	11.0368	1.0347	2.1704

Table 3. Comparison of techniques based on Recall, Precision and Global accuracy.

Technique	Detected total number	Total Number of true duplicates detected	Total Number error	Recall	Precision	Global accuracy
Simple Homonym	1140	1129	11	75.47%	99.04%	74.74%
Simple Metaphone	1102	1085	17	72.53%	98.46%	71.41%
Simple Initfinal	1176	1120	56	74.87%	95.24%	71.31%
Simple Soundex	1116	1090	26	72.86%	97.67%	71.16%
Enriched Homonym	383	383	00	25.60%	100%	25.60%
Enriched Soundex	299	298	01	19.92%	99.67%	19.85%
Enriched Initfinal	292	289	03	19.32%	98.97%	19.12%
Enriched Metaphone	229	229	00	15.31%	100%	15.31%

average is the "simple initial" with 4.9849.

Techniques with the best algorithmic complexity are enriched metaphone and enriched soundex with 2.0315 and the one with the largest algorithmic complexity is the initial simple with 316.0652.

It may be noted here that the techniques based on the homonym have very good values for all the criteria, usually not far from the best values.

The Table 3 shows the result of the comparison of many techniques based on Recall, Precision and Global accuracy.

In terms of precision, we discover that the technique of enriched homonym and that of enriched metaphone are better with 100% precision. In terms of the global accuracy, which includes the recall and precision we discover that the simple homonym is the best technique because it has the best global accuracy. In addition, it has the best recall (75.47%), and a precision of 99%.

- *Results and interpretations of the second comparison*

For the second comparison, we use the classic restaurant's database in the USA, containing a table of 864 records with 112 pairs of known duplicates. The blocks are calculated on the basis of the name. The enrichment is made by using

the city, normalized to avoid among other errors the term polluting as "city".

The Table 4 allows us to compare the different techniques based on results obtained with these methods on standard data used in the literature by several authors. It is found that the techniques with the best recall are the technique of the enriched homonym and the technique of simple homonym with 89% of recall. In terms of Precision, we realize that the technique of the enriched homonym and that of enriched metaphone are better with 100% of precision. In terms of the global accuracy which includes the recall and precision, we realize that the enriched homonym is the best technique. It is surprising to discover that techniques as simple as the initfinal have an overall accuracy that is not 5% worse than that of the best techniques.

The authors *Bhagyashri* and *et al.* have worked on the same database of restaurants and produced satisfactory results by varying the thresholds of their rules application, as well as the fields used to form blocks. Table 5 shows the best results in their work. We can see that the technique of enriched homonym gets better results in all major criteria: recall precision, global accuracy and f-score.

Below is the best result obtained using the same data by certain authors [1].

This table helps to compare our results with those obtained by Bhagyashri, A. Kelkare and Prof. KB Manwade using the same database.

This report is more interesting as there is, in this case, no search for optimal parameters for application of rules to have the best score. Basic rules, developed during the work on a central Africa database, were used without modification for the restaurant's database, and without a parameter. We can note the very low

Table 4. Comparison of techniques based on Recall, Precision and Global accuracy using restaurant database.

Technique	Detected total number	Total number errors	Total number correct	Recall	Precision	Global accuracy	F-measure	Pair comparisons	Complexity
Simple Homonym	218	18	200	89.29%	92.67%	82.23%	90%	1372	1.59
Enriched Homonym	200	0	200	89%	100%	89.00%	95%	1247	1.44
Simple Metaphone	212	18	194	87%	92%	79.43%	89%	1380	1.60
Enriched Metaphone	184	0	184	82%	100%	82.00%	91%	1048	1.21
Simple Soundex	212	18	194	87%	92%	79.43%	89%	1323	1.53
Enriched Soundex	184	0	184	82%	100%	82.00%	91%	1048	1.21
Simple InitFinal	210	18	192	86%	91%	77.49%	89%	1401	1.62
Enriched InitFinal	182	0	182	81.30%	100%	81.30%	90%	1093	1.26

Table 5. Some results obtained by Bhagyashri, A. Kelkare and Prof. KB Manwade using Restaurant database.

Recall	Precision	Global accuracy	F measure	Pair comparisons	complexity
87%	79%	63.87%	83%	4 258	4.9282
88%	85%	72.47%	86.5%	4 307	4.985
87.40%	98%	85.62%	92.7%	4 544	5.2593

number of pair comparisons (with 1800 comparisons) compared to what was best in this work (4200 comparisons), a constant exceptional precision of the technique of enriched homonym (100%), and a very good recall.

Knowing that we are working on a hashing method, its direct application in software for quick-duplication can be considered. The high precision of the enriched versions could be considered for automatic deduplication.

We applied the techniques of homonyms to a database of Cameroonian workers and their parents, containing 1,600,000 lines. The Algorithmic complexity remained less than 2, what allows us to conclude that even for very big databases these techniques can apply at time T=O(2n), which demonstrates a good scalability. Let us note that in passing from the restaurant's database to that of the importers, algorithmic complexity remained less than 2, while the size increased from 864 to 65,000.

4. Discussions

According to the results obtained, we conclude that the simple homonym blocking technique is the best techniques in all cases for the recall criteria. Therefore, this technique is the best candidate for duplicate prevention on databases application. It can be used to prevent duplicate creation by showing to the user the "duplicates candidates" while creating a record, and by asking the user to confirm record creation or not. Due to the fact that it is a block-hashing technique, one can just create a field homonym in each record, and uses this field to store the value of the simple homonym of this record. During a new record creation, a simple request can be made in order to obtain and show all records with the same value of homonym than the record in creation. This strategy was implemented in various software and gives very good results.

For the precision criteria, the enriched homonym is the best technique, with a precision of 100% in all studied cases. It can be considered as a good technique in a process of automatic deduplication if the precision is very important. The simple homonym technique is a good candidate for automatic deduplication due to its good global accuracy, and the enriched homonym is the best candidate when the field used to improve precision doesn't lead to a low recall, due to its good global accuracy.

One limitation of this work is that the global accuracy is calculated assuming that the cost of not detecting a duplicate is the same as the cost of a false duplicate detection. It is not true in some cases. In these cases, we need to have another model in order to have a global quality indicator of duplicate detection techniques.

5. Conclusions and Perspectives

This work had for objective to propose duplicate detection techniques, and compare their performance with real data in order to make recommendations on their use, improve the efficiency of detection and reduce the algorithm com-

plexity. There was also a concern to propose algorithms of blocking by hashing, having performances good enough to be effectively used in the transactional systems. Several techniques of blockings were introduced, as well as metrics of distances and rules of comparisons. The technique of homonym has been proved particularly effective compared with all the criteria of comparison, and seems to us particularly indicated in its simple or enriched version, for the detection of duplicates in the transactional or decision-making systems.

Nevertheless, we see that all techniques studied generally have a recall rate of less than 90%. One of the main challenges is the improvement of the recall, keeping a low complexity.

Very good precision (100%) obtained by the enriched homonym paves the way for automatic deduplication opportunities. Doing it implies the introduction of a risk function to more effectively measure the impact of an error in the automatic deduplication and integrate the results within deduplication algorithms. This is also an interesting perspective to analyze, for automatic deduplication of large databases.

Acknowledgements

We thank MrTanyi Ekokobe for his help in correcting this document.

Funding

This work was supported by Megasoft Cameroon and by the CETIC project of the National Advanced School of Engineering of Yaounde.

Contribution

We introduce several methods for detecting duplicates whose average complexity observed is less than O(2n), after conducting a more comprehensive classification of semantic duplicates than the usual classifications. Through a simple model, we highlight a global efficacy rate, combining precision and recall. We propose new metrics distance between records, as well as rules for automatic duplicate detection. We study the performances of the proposed methods on real data in Central Africa.

References

[1] Bhagyashri, Kelkar, A. and Manwade, K.B. (2012) Identifying Nearly Duplicate Records in Relational Database, *International Journal of Computer Science and Information Technology & Security*, **2**, 514-517.

[2] Baxter, R., Christen, P. and Epidemiology, C.F. (2003) A Comparison of Fast Blocking Methods for Record Linkage. *Proceedings of Workshop Data Cleaning, Record Linkage, and Object Consolidation*, Washington DC, August 24-27 2003, 25-27.

[3] Aizawa and Oyama, K. (2005) A Fast Linkage Detection Scheme for Multi-Source Information Integration. *Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration*, Tokyo, 8-9 April 2005, 30-39. https://doi.org/10.1109/WIRI.2005.2

[4] Yan, S., Lee, D.W., Kan, M.-Y. and Lee, C.G. (2007) Adaptive Sorted Neighbor-

hood Methods for Efficient Record Linkage. *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, Vancouver, 18-23 June 2007, 185-194.

[5] McCallum, Nigam, K. and Ungar, L.H. (2000) Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, Boston, 20-23 August 2000, 169-178.

[6] Ramos, J. (2003) Using Tf-idf to Determine Word Relevance in Document Queries. https://pdfs.semanticscholar.org/b3bf/6373ff41a115197cb5b30e57830c16130c2c.pdf?_ga=1.183821606.1606390940.1482600858

[7] Christen, P. (2007) Performance and Scalability of Fast Blocking Techniques for Deduplication and Data Linkage. *Proceedings of the VLDB Endowment*, Vienna, 23-28 September, **1**, 1253–1264.

[8] Metha, Kadhum, Alnoory, Musbah and Aqel, M. (2011) Performance Evaluation of Similarity Functions for Duplicate Record Detection. Master's Thesis, Middle East University, Beirut. http://elibrary.mediu.edu.my/books/2014/MEDIU11238.pdf

[9] Bianco, G.D., Galante, R. and Heuser, C.A. (2011) A Fast Approach for Parallel Deduplication on Multicore Processors. 26*th Symposium on Applied Computing SAC*'11, TaiChung, 21-25 March 2011, 1027-1032. https://pdfs.semanticscholar.org/0261/d528ec0e1a31e8a40053acd60513beebef2d.pdf https://doi.org/10.1145/1982185.1982411

[10] Raghavan, H. and Allan, J. (2004) Using Soundex Codes for Indexing Names in ASR Documents. *Proceeding SpeechIR* '04 *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL* 2004, Boston, 6 May 2004, 22-27. https://doi.org/10.3115/1626307.1626312

[11] Christen, P. (2006) A Comparison of Personal Name Matching: Techniques and Practical Issues. 6*th IEEE International Conference on Data Mining-Workshops*, Hong Kong, 18-22 December 2006, 290-294. https://pdfs.semanticscholar.org/654d/51abeb59861dde5f8097127a5b5a12147f9f.pdf https://doi.org/10.1109/icdmw.2006.2