Duplicates Detection Within Incomplete Data Sets Using Blocking and Dynamic Sorting Key Methods

Abdulrazzak Ali[1]
Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka
Hang Tuah Jaya, 76300, Ayer Keroh, Melaka, Malaysia

Nurul A. Emran[2]
Computational Intelligence Technologies (CIT) Research Group
Universiti Teknikal Malaysia Melaka
Hang Tuah Jaya, 76300, Ayer Keroh, Melaka, Malaysia

Siti A. Asmai[3]
Optimization, Modeling, Analysis,
Simulation and Scheduling (OptiMASS) Research Group
Universiti Teknikal Malaysia Melaka
Hang Tuah Jaya, 76300, Ayer Keroh, Melaka, Malaysia

Awsan Thabet[4]
Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka
Hang Tuah Jaya, 76300, Ayer Keroh, Melaka, Malaysia

***Abstract*—In database records duplicate detection, blocking method is commonly used to reduce the number of comparisons between the candidate record pairs. The main procedure in this method requires selecting attributes that will be used as sorting keys. Selection accuracy is essential in clustering candidates records that are likely matched in the same block. Nevertheless, the presence of missing values affects the creation of sorting keys and this is particularly undesirable if it involves the attributes that are used as the sorting keys. This is because, consequently, records that are supposed to be included in the duplicate detection procedure will be excluded from being examined. Thus, in this paper, we propose a method that can deal with the impact of missing values by using a dynamic sorting key. Dynamic sorting is an extension of blocking method that essentially works on two functions namely uniqueness calculation function (UF) (to choose unique attributes) and completeness function (CF) (to search for missing values). We experimented a particular blocking method called as sorted neighborhood with a dynamic sorting key on a restaurant data set (that consists of duplicate records) obtained from earlier research in order to evaluate the method's accuracy and speed. Hypothetical missing values were applied to testing data set used in the experiment, where we compare the results of duplicate detection with (and without) dynamic sorting key. The result shows that, even though missing values are present, there is a promising improvement in the partitioning of duplicate records in the same block.**

***Keywords*—Duplicate detection; Incomplete Data Set; Blocking Methods; Sorting key; Attribute Selection**

I. Introduction

Duplication detection is a crucial process in many data cleaning operations. Nevertheless, this process relies on time-consuming attributes comparison between the records pairs that have been reported as a common bottleneck in duplication detection [1]. To address this problem, the records are normally partitioned into small subsets so that searching for the duplicates is only performed within the small subsets. Common blocking methods that are used to partition the records into blocks (or also called as windows) are standard blocking and sorted neighborhood. These methods work well in big data sets that depend on candidate keys to sort the data set [2]. Blocking methods aims at producing a set of blocks that offers a good balance between the number of detected duplicates and the number of required comparisons. Selection of a suitable blocking method for duplication detection usually depends on domain knowledge [3]. In the context of homogeneous information spaces, these methods typically consider the frequency distribution of the values of attribute names as well as their quality (i.e., the presence of noise or missing values) to derive the most suitable blocking key(s) [4]. In order to obtain an unambiguous sorting order, it is desirable that the sorting keys are unique. In fact, within a data set, candidate key attribute (such as name) can also be used as sorting key rather than using attributes that are commonly defined to partition the data set (such as zip code) [1]. As data uniqueness is a crucial aspect in duplication detection, attributes that contain errors or missing values will hinder the creation of sorting key [5]. In addition, attributes that consists of high repetition ratio such as gender (with only two value states) are less useful as sorting key. Nevertheless, attribute such as surname can be a useful sorting key but it may frequently be reported or keyed incorrectly [4]. The sorting key with uniqueness criteria less repetition ratio will contribute in reducing the unnecessary comparisons of pairs of candidate records in duplicate detection process.

Duplicate detection is usually evaluated in terms of its accuracy. Thus, it relies on how well the blocking method deals with errors. Two kinds of error are of concerned in blocking methods which are:

1) False positive or false acceptance: This error occurs when the actual 'unmatched' records in the candidate pair space are included for comparison. This will cause unnecessary time spent for comparison.
2) (False negative or false rejection: This error occurs when the actual 'matched' records in the candidate pair space are excluded for comparison. This will cause duplicate records to remain in the data set and causes worse consequence than the false positive case.

As described by Christen (2007), the process of dividing the data set into blocks requires the following steps [6]:

1) The data set is sorted based on the blocking key value (BKV) which is created from one or more attributes.

Sorted records rely on the following underlying hypothesis [3]:
The n sorted records, $r_i \leq \ldots \leq r_{i+n}$, may satisfy: $dist(r_i, r_{i+1}) \leq dist(r_i, r_{i+2}) \leq \ldots \leq dist(r_i, r_{i+n})$, where, $dist(r_i, r_j)$, is the distance between records r_i and r_j. Most of the blocking methods use the inverted index key where all records that have the same blocking key are placed in the same indexing list.

2) The record identifiers are retrieved from the index data structure for the block and record pairs are created for comparison. Each record is associated with the rest of records in the same block to form the candidate pairs. After the comparison step is complete, a numerical value is assigned.

In the next section, we will present related works in duplicate detection where blocking methods were proposed to improve the efficiency of duplicates detection.

II. Related Work

There are at least two competing blocking methods used in duplicate detection as stated by Draisbach and Naumann (2010)[1]:

- Standard blocking method that partitions records into separate subsets, for example using "zip code" as the partitioning key.
- Sorted neighborhood method that sorts the data set according to certain key(s), such as "last name", and then slide a window of fixed size across the sorted data and compare pairs only within the window.

A. Standard Blocking Method

In this method, records are sorted using the block key value and the sorted records are grouped into separate blocks where every record has the same BKV. The key is composed of a set of data set attributes. The comparison is made for all the records within the same block. If the duplicate records detection is required in the data set which has R number of records and B number of blocks, the number of record pairs is $O(\frac{R2}{B})$ for comparison. This method increases the speed of comparison operations because of the limited comparison between records within the same block.

Basically, this method is based on the expectation-maximization (EM) algorithm as proposed by [4] to calculate the probability of matching a pair of records. The probability of matched records pair is defined as class M, $P(xi = 1|M)$, and the probability of unmatched records pair is defined as class U, $P(xi = 1|U)$ that are calculated from a number of random records. Fig. 1 illustrates a standard blocking technique which was adapted from the original source [1].

Nevertheless, as the records are distributed into equal blocks size and compared within their own block, there is a high possibility for record mismatching (false negative error) that leads to less accurate duplicate detection. Therefore, it is necessary to choose appropriate keys for sorting in order to reduce the error ratio.

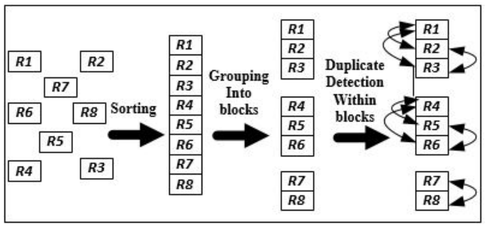


Fig. 1. Standard Blocking Technique(BKV) [1]

B. Sorted Neighborhood Method

Sorted Neighborhood (SN) (which is also called as windowing) was proposed by Hernandez and Stolfo's (1998) to reduce the time taken for comparison process [7]. As compared to the standard blocking, this method improves more on the records matching ratio. SN requires creating a string of substrings from the selected attributes of the record to form the candidate key. For example, the first three letters of the name field, the first three digits of the identification number and the first three letters of the electronic mail concatenated together are used to create candidate key as shown in Table I.

TABLE I. Example of Creating Candidates Keys

ID	Name	E-Mail	Candidate keys
12331232	Mohammed Ahmed	Moha12@yahoo.com	Moh123Moh
34241212	Saleh Noor	Salnor@yahoo.com	Sal342Sal
12123234	Noor Bint Salem	Norsal1@hotmail.com	Noo121Nor

The records are then sorted based on the candidate keys. Then the comparison of records is performed within the window. After the comparison is finished, the first record is sliding out from the window and the next record is added to the window. If the size of the fixed window is w, then any new record enter to the window is compared with the previous one $|w| - 1$ records to find the matching. The process is illustrated by Fig. 2 shows a sorted neighborhood technique adapted from the original source [8].

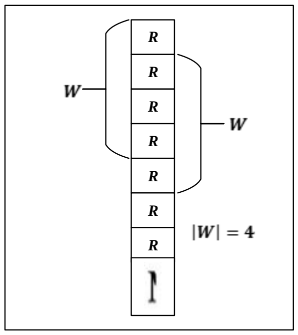


Fig. 2. Sorted Neighborhood (SN) Technique [8]

The accuracy of the SN method depends on the quality of the key and also the window size. With small window size and a high number of duplicates, the problem that can occur is overriding a number of duplicates (false negative error). However, if the size of the window is too large, unnecessary comparisons will occur. Therefore, the quality of the key in addition to the size of the window is a challenge in SN technique. This problem was resolved by independent execution several times and the records are sorted with a different key in each time with a small window size. The result is derived from the integration of results where this method is called a multi-pass SN as proposed by Hernández and Stolfo (1995) [9].

C. Duplicate Detection Within Incomplete Data Set

Many methods of duplication detection encountered a problem in compiling and comparing partially complete records. In fact, there are several types of data completeness problem can be found in data sets [10], [11] that can cause partially complete records. The presence of missing values (nulls) is one type data completeness problem that affect duplicate detection. There are several methods proposed to deal with missing values in data sets, such as to ignore records with missing entries, manual data imputation by human and the use an expectation-maximization (EM) imputation algorithm. The first two methods are impractical, while the estimation of missing values in a large number of attributes through EM becomes computationally intractable [12][13][14].

Tamilselvi and Saravanan (2009) presented a framework to detect and eliminate duplicates due to errors and missing data [15]. The proposed framework consists of six steps which are: 1) select the best attributes for duplicate identification (using attribute selection algorithm), 2) form token, 3)cluster the data (using clustering algorithm), 4) compute similarity score, 5) eliminate duplicates and 6) merge. The method of selecting attributes with complete values that do not contain duplicate values to perform tokens matching.

In [14], the authors proposed a method that extends the soft TF-IDF method to address two common cases in the detection of duplicates namely sparsity due to the number of missing entries, and the presence of more than one record duplicate. The proposed method consists of three steps: 1) create similarity scores between the records, 2) clustering the records together in independent groups, and 3)compare the different ways to create similarity scores between records (in addition to a different set of string matching, frequency-inverse methods, and n-gram techniques). The authors pointed out that the method of dealing with missing values can be replaced in the TF-IDF and Jaro-Winkler by performing data imputation with a likely candidate.

III. Problem Definition

Several important issues need to be considered when record attributes are selected to be used as blocking keys. The first issue is that the quality of the values in these attributes will influence the quality of the generated candidate record pairs.

Ideally, attributes containing the fewest errors, variations, or missing values should be chosen. Any error in an attribute value used to generate a BKV will potentially result in records being inserted into the wrong block, thus leading to missing true matches [12][16].

In SN method, the selected attributes will be used as the blocking key. If the window size is small, then the number of comparisons will be reduced but the false negative value will be increased. If the window size is large then the number of comparisons will be increased with reducing the number of false negatives. For this kind of problem, the calculation of distinct and missing values is very important in the selection of blocking key. A numeric blocking key will not be effective in blocking the record for comparison. Therefore, the identification of measurement type and the type of attributes are important in the selection of blocking key [5]. To avoid miss-sorting due to errors in the attributes that are used to generate the key, again, multi-pass variants of SN produce multiple keys and perform the sorting and windowing multiple times. Like the standard blocking method, the transitive closure is finally calculated. Research has produced many variants of SN method including the one that avoids the choice of keys [15] and a variant for nested XML data [8].

Bigram indexing method is less sensitive to typo errors or some missing information as compared to the previous blocking methods. The features of this method are, first, this method allows each record to be in multiple blocks if needed, and it uses an inverted index [8]. There are three criteria which are very important in attribute selection for data cleaning which are identifying key attributes, classifying attributes (with high distinct value and low missing value) and measurement types of the attributes [17]).

In this section, the impact of the missing values on the blocking key will be elaborated through the following example. Suppose that we have a complete data set that contains employee personal information such as (name, address, city, phone, and sex) as shown in Table II. In this example data set, duplicate records are present in pairs, which are records (1,2), (3,4) and (8,9). In discussing the creation of the sorting key and the effect of missing values on duplicate detection, we consider the following cases:

TABLE II. Personal Information

no	name	address	city	phone	sex
1	Ahmad	435 s. la cienegablv.	losangeles	310/246-1501	M
2	Ahmed	435 s. la cienegablvd.	losangeles	310-246-1501	M
3	Bel	701 stone canyon rd.	bel air	310/472-1211	M
4	Bell	701 stone canyon rd.	bel air	310-472-1211	M
5	Brit	12224 venturablvd.	studio city	818/762-1221	M
6	Brown	23725 w. malibu rd.	malibu	310-456-0488	F
7	Jim	9560 dayton way	losangeles	310/276-0615	M
8	Johne	14016 venturablvd.	sherman oaks	818/788-3536	M
9	Johne	14016 venturablvd.	sherman oaks	818-788-3536	M
10	Navy	2709 main st.	losangeles	310-352-8035	F
11	Nicol	624 s. la breaave.	losangeles	213-938-1447	F
12	Nicolas	2600 main st.	santamonica	310/392-9025	M

A) **Single sorting key:** In this case, the creation of sorting key depends on one of the record attributes as shown in Table III where the sorting key is the attribute name. The problem, in this case, is sorting key will not appear when the attribute "name" value is missing in a record.

Missing sorting key causes the record to be placed out of the appropriate records block. For example, as "name" value for record 9 is missing, the record becomes the last record after the sorting and it will be excluded from the block.

TABLE III. SINGLE SORTING KEY

no	name (sortkey)	address	city	phone	sex
1	Ahmad	435 s. la cienegablv.	losangeles	310/246-1501	M
2	Ahmed	435 s. la cienegablvd.	losangeles	310-246-1501	M
3	Bel	701 stone canyon rd.	bel air	310/472-1211	M
4	Bell	701 stone canyon rd.	bel air	310-472-1211	M
5	Brit	12224 venturablvd.	studio city	818/762-1221	M
6	Brown	23725 w. malibu rd.	malibu	310-456-0488	F
7	Jim	9560 dayton way	losangeles	310/276-0615	M
8	Johne	14016 venturablvd.	sherman oaks	818/788-3536	M
10	Navy	2709 main st.	losangeles	310-352-8035	F
11	Nicol	624 s. la breaave.	losangeles	213-938-1447	F
12	Nicolas	2600 main st.	santamonica	310/392-9025	M
9		14016 venturablvd.	sherman oaks	818-788-3536	M

B) **Multiple sorting key:** In this case, the sorting key consists of the strings of partial attribute values as shown in Table IV. In this table, the sorting key value consists of the first three characters of the attribute "name" and the first three characters of the "city". The sorting key is totally missing when all attribute values used as sorting key are missing; partially missing if at least one attributes value is missing. In both cases, the records with a missing sorting key will be outside of the records block.

TABLE IV. MULTIPLE SORTING KEY

no	sortkey	name	address	city	phone	sex
1	Ahmlos	Ahmad	435 s. la cienegablv.	losangeles	310/ 246-1501	M
4	Belbel	Bell	701 stone canyon rd.	bel air	310- 472-1211	M
5	Bristu	Brit	12224 venturablvd.	studio city	818/ 762-1221	M
6	Bromal	Brown	23725 w. malibu rd.	malibu	310- 456-0488	F
7	Jimlos	Jim	9560 dayton way	losangeles	310/ 276-0615	M
9	Johshe	Johne	14016 venturablvd.	sherman oaks	818- 788-3536	M
2	los		435 s. la cienegablvd.	losangeles	310- 246-1501	M
10	Navlos	Navy	2709 main st.	losangeles	310- 352-8035	F
11	Niclos	Nicol	624 s. la breaave.	losangeles	213- 938-1447	F
12	Nicsan	Nicolas	2600 main st.	santamonica	310/ 392-9025	M
8	she		14016 venturablvd.	sherman oaks	818/ 788-3536	M
3			701 stone canyon rd.		310/ 472-1211	M

The purpose of sorting key creation is to partition the data set into smaller subsets so that the number of comparisons can be reduced. To compare the records in the data set, in the traditional way of duplicates detection each record in the data set will be compared with the rest of the records in the data set. If N is the number of records in the data set, the number of comparisons is a matrix $N \times N$. Thus, as there are 12 records in the example, there are $12 \times 12 = 144$ comparisons will be required. To solve this problem, the data set is partitioned by a sorting key into small blocks with a certain length. For example, suppose that n is the length of the block (which also represents the number of records in the block), thus the numbers of comparisons for each block is $\frac{n(n-1)}{2}$. Suppose that n is 4, with 12 records, the data set will be divided into three blocks. The number of comparisons in each block is equal to $\frac{4(4-1)}{2} = 6$. Thus, the total number of comparisons for all blocks is 6 × 3 = 18, which is less than the number of comparisons in case if we deal the data set as a single block (where the number of comparisons equals to $\frac{12(12-1)}{2} = 66$).

Missing values can affect the distribution of records on blocks. In our previous example, if the data set is partitioned into blocks of length 4, then in the data set shown in Table III record 9 with a missing value will be is excluded from the comparison process. Similarly, for data set in Table IV, records 2, 3 and 8, will be excluded from the block under measure.

Typically, multiple passes method is used in order to overcome the false negative error as shown in the example. In this way, the records that are not blocked together in one pass will have the potential to be blocked and compared in another pass to avoid from being misclassified. Since two records cannot be matched due to missing values, the variables chosen for the blocking phase should be relatively complete, with only a few missing values. Such blocking strategy will reduce the set of potential matches to a more manageable number. Christen and Goiser (2007) recommend researchers to report the specific steps of their blocking strategy in order to ensure blocking success where in their work, blocking method is used to deal with record linkage problem [18]. To deal with the missing values and their effect in the creation of the sorting key, in the next section, we describe our proposal of blocking with a dynamic sorting key.

IV. THE PROPOSED METHOD

In this section, we will describe the extension of the traditional method we propose in determining the attributes that are selected in the composition of the sorting key through a dynamic sorting key. In particular, we focus on the problem caused by the presence of missing values in the sorting key attributes. The proposed dynamic sorting key aims to avoid false negative errors by reducing the number of records that are excluded from the blocks due to the missing values. An experiment was conducted in order to evaluate the proposed method.

A. Attribute Selection

Attributes selection is a pre-requisite process for the dynamic key generation. Attributes selection stage is one important stage upon which duplicates detection relies. In this stage, "appropriate" attributes are selected to create sorting keys in the classification stage. To choose the appropriate attribute there are three criteria to meet: (a) identifying key attributes,

(b) attributes with high distinct value and low missing value and (c) attribute measurement types. In the end, attributes with the highest priority are selected for the further process [19], [15]. Attributes selection process relies on the rate of missing values and repetition values in each attribute and also the threshold value for the acceptance rate.

The attributes selection process passes through three main sequential stages as shown in Algorithm 1. The aim of the attributes selection algorithm is to reduce the time and increase speed in later stages. The algorithm begins by setting the threshold value T, where (0<T<1). The higher the value of T means the higher the acceptability on the amount of repetition and missing values in the attribute. A good selection of threshold value is very important since the choice of a very low threshold value affects the creation of the sorting key, which in turn affects the detection of duplicates in the data set.

Algorithm 1 Attributes selection algorithm

```
Input: $N$ Attributes, $n$ number of rows, $T$
Output: $S$ Subset of attributes
Var: $A$ Attribute set, $i, j$
begin
  1) $threshold \leftarrow T$
  2) Unique $(U)$ value of the attribute $Ai$ if row $i_1^n = Ai + 1$
  3) Missing $(C)$ value of the attribute $Ai$ if row $i_1^n = Null$
  4) Calculate $AVG$=$avg(U \wedge C)$
  5) Compare $(T, AVG)$
  6) Rank $(N, AVG)$
end
Compute:
  1) $U_j = n-$ Count(Distinct $A_j$)
  2) $C_j = \frac{\sum_{i=0}^{N} completeness_{i,j}}{n}$
```

- **Uniqueness:** the uniqueness coefficient is calculated to measure the repetition ratio in the attributes values. As it is not possible to rely on attributes that contain duplicate values such as a gender attribute (that contains values only namely male and female) this function is crucial in attribute selection. To calculate the uniqueness coefficient, the uniqueness function (UF) is applied to each attribute in the data set. Figure 3 shows the flow of UF.
- **Completeness:** Figure 4 shows the flow of the algorithm that is called as completeness function (CF) to calculate the number of missing values for the attributes that have a high proportion of uniqueness (based on the threshold value) in the previous stage. To describe CF, suppose that:
  A is a data set.
  M is an array where each element in M is the number of missing values in the attribute in A.
  A_{attr} is the attribute identifier in A.
  M_{Key} is the attribute identifier in array M. The steps taken for CF to measure the missing values are as follow:
  1) Create one dimension array M with a number of elements equal to the number of attributes in A. Each element in M consist of key and values, where the key represent the attribute name of A and the value represents the ratio of missing values in the attribute A_{attr} and $M_{Key} = A_{attr}$.
  2) Check if the attribute which is represented in $A_{j,i}$ contains missing values by comparing the value of $A_{j,i}$ with the missing value representations such as null, blank space or "?" character. Thus, if the value of $A_{j,i}$ equal any representation of missing values, the value of M_j increases by one.

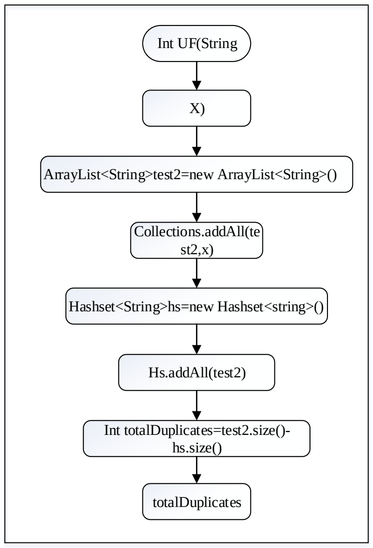

Int UF(String
X)
ArrayList<String>test2=new ArrayList<String>()
Collections.addAll(test2,x)
Hashset<String>hs=new Hashset<string>()
Hs.addAll(test2)
Int totalDuplicates=test2.size()-hs.size()
totalDuplicates

Fig. 3. Uniqueness Calculation Function

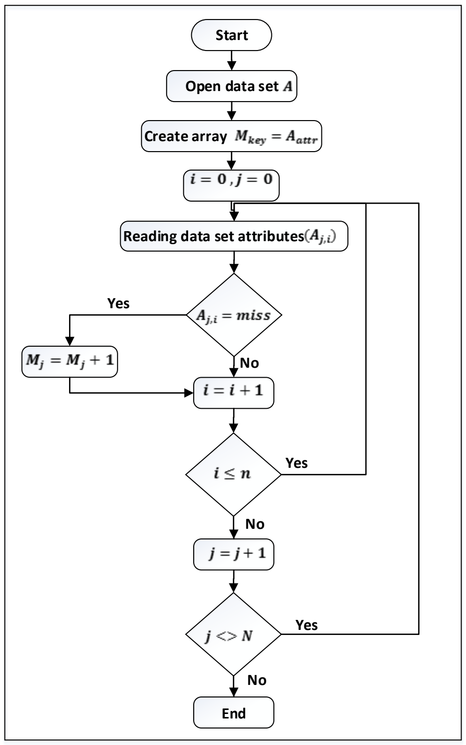


Fig. 4. Completeness Function

- **Aggregation and ranking:** In the final step, we calculate the average of the uniqueness and the completeness scores for each selected attribute. The result (AVG) is compared with the threshold value (T) that was set at the beginning. Attributes with a high average of duplicates values and missing values are excluded from the candidate list. This means the result fails to fulfill T. Attributes that are in the candidate list are ranked according to their average scores.

B. Dynamic Sorting Key

Dynamic sorting key creation depends on the attribute selection stage, that determines the high-rank attributes. In this method, three high-ranking attributes are selected to form a sorting key. In addition, the sub-string size is determined from the first three characters of each selected attribute where the sub-strings are grouped together to form a single string representing the sorting key after removing the blanks in the values of the attributes (if any). Figure 5 shows the steps to create the dynamic sorting key. A new attribute (called as

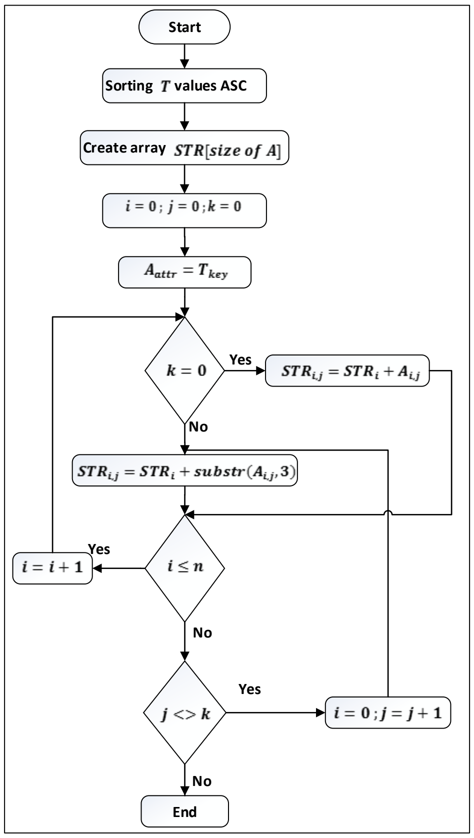

Start
Sorting T values ASC
Create array $STR[size\ of\ A]$
$i = 0\ ;\ j = 0\ ; k = 0$
$A_{attr} = T_{key}$
$k = 0$
Yes
$STR_{i,j} = STR_i + A_{i,j}$
No
$STR_{i,j} = STR_i + substr(A_{i,j}, 3)$
$i \leq n$
Yes
$i = i + 1$
No
$j <> k$
Yes
$i = 0\ ; j = j + 1$
No
End

Fig. 5. Dynamic Sorting Key

ksort) is added to the data set to store the values of sorting keys that were created by dynamic sorting key function. After the dynamic sorting key is generated, the data set is sorted based on its values. The sorting key is used to partition the data set into small blocks to reduce the comparisons between the records pairs. In our method, we use DuDe toolkit (as proposed by Draisbach and Naumann (2010) [20]) to test the performance of duplicates records detection within incomplete data set with dynamic sorting key. In Section V, we will present the results of duplicate detection with the proposed method.

C. Data Set

To test our method, we used the restaurant data set that provided by Hasso-Plattner-Institute (HPI)[1] which has been frequently used in duplicate detection research. The restaurant data were extracted from the RIDDLE repository. This data set consists of real 864 restaurant records, which was taken from the Fodor's and Zagat's restaurant guides that contains 112 duplicates. The data set comprises the names and addresses of restaurants. We have used restaurant data set that are configured by Duplicate detection toolkit (DuDe). Restaurant data set underwent a series of changes as reported in [20]. These changes are:

- changed file format from "arff" to "csv".
- removed header information.
- inserted row with column names.
- added a unique identifier for each record.
- deduplicated information in data sets.

Several additional changes have been made to the restaurant data set to prepare it for the experiment. The changes in the so-called data preparation step are:

- Partitioned the comma-separated value (CSV) file into attribute headings. The data set was originally separated by a semicolon.
- Eliminated blank spaces between the values for all attributes.
- Applied conceptual of arbitrary pattern to add missing values to the data set by deleting values from the data set attributes randomly (following the method in [21]).

D. Experiment Configuration

To show the effect the missing values on the blocking method, we apply our method by using DuDe toolkit that has been used to detect duplicate records. DuDe is an open source tool that is opened for modifications and additions on the functions code for research purposes. We add attributes selection functions to select the proper attributes before the dynamic sorting key can be created. Figure 6 shows the experiment flow. In the experiment, a java class was created to read restaurants data sets that were stored in CSV format. Data preparation step as described earlier was performed before the attribute selection algorithm can be applied. To generate the dynamic sorting key, the UF was applied on the restaurant data set to get the uniqueness factor for each attribute. The CF was applied against the attributes with the highest uniqueness score (low repeated values) before the dynamic sorting key can be created. Figure 7 shows the values of dynamic keys (Ksort) created from the first three characters of the attributes values "Phone", "Address" and "Class" (that exhibit low repeated values and low missing values).

In the proposed method, we modified the configuration of SN algorithm that is used in DuDe toolkit. The modification

[1]https://hpi.de/naumann/projects/data-quality-and-cleansing/dude-duplicate-detection.html#c114678

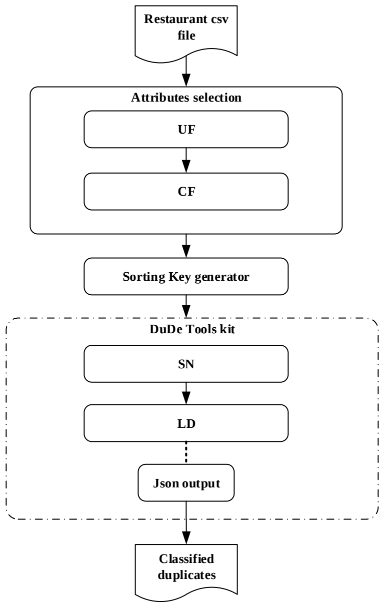
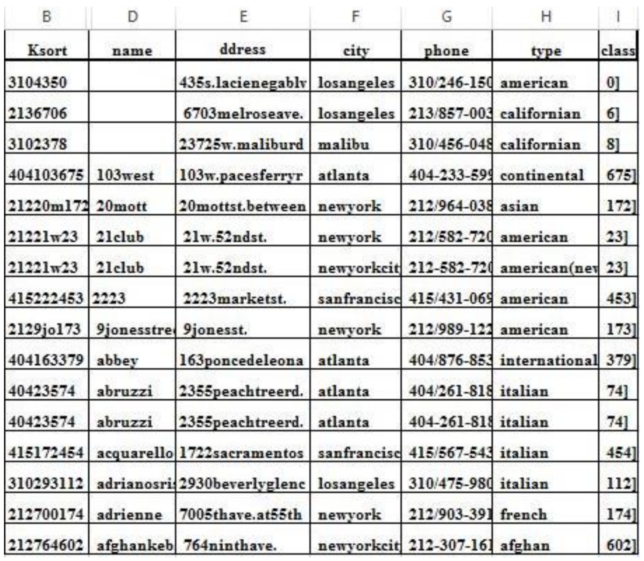


Fig. 6. Experiment Flowchart



B	D	E	F	G	H	I
Ksort	name	ddress	city	phone	type	class
3104350		435s.lacienegablv	losangeles	310/246-150	american	0]
2136706		6703melroseave.	losangeles	213/857-003	californian	6]
3102378		23725w.maliburd	malibu	310/456-048	californian	8]
404103675	103west	103w.pacesferryr	atlanta	404-233-599	continental	675]
21220m172	20mott	20mottst.between	newyork	212/964-038	asian	172]
21221w23	21club	21w.52ndst.	newyork	212/582-720	american	23]
21221w23	21club	21w.52ndst.	newyorkcit	212-582-720	american(nev	23]
415222453	2223	2223marketst.	sanfrancisc	415/431-069	american	453]
2129jo173	9jonesstre	9jonesst.	newyork	212/989-122	american	173]
404163379	abbey	163poncedeleona	atlanta	404/876-853	international	379]
40423574	abruzzi	2355peachtreerd.	atlanta	404/261-818	italian	74]
40423574	abruzzi	2355peachtreerd.	atlanta	404-261-818	italian	74]
415172454	acquarello	1722sacramentos	sanfrancisc	415/567-543	italian	454]
310293112	adrianosri	2930beverlyglenc	losangeles	310/475-980	italian	112]
212700174	adrienne	7005thave.at55th	newyork	212/903-391	french	174]
212764602	afghankeb	764ninthave.	newyorkcit	212-307-161	afghan	602]

Fig. 7. Sorting Key

allows SN algorithm to pair and to sort the records based on dynamic sorting key where window size was set to 10. Listing 1 shows the configuration of the SN algorithm.

Listing 1. Configuration of the SN algorithm

```
//Attributes Selection
//Missing values compensate
//Dynamic sorting key(Ksort) generator
new SN algorithm with window size 10 and
in-memory processing enabled
algorithm=new SortedNeighborhood
Method(Ksort,10);
algorithm.enableInMemoryProcessing();
algorithm.addDataExtractor(extractor);
```

The SN algorithm returns the pairs to be classified by the comparator that uses Levenshtein Distance (LD) for the comparison string $ksort$ that was created in the previous step with similarity measure equals to 0.9. Listing 2 shows the configuration of the comparator.

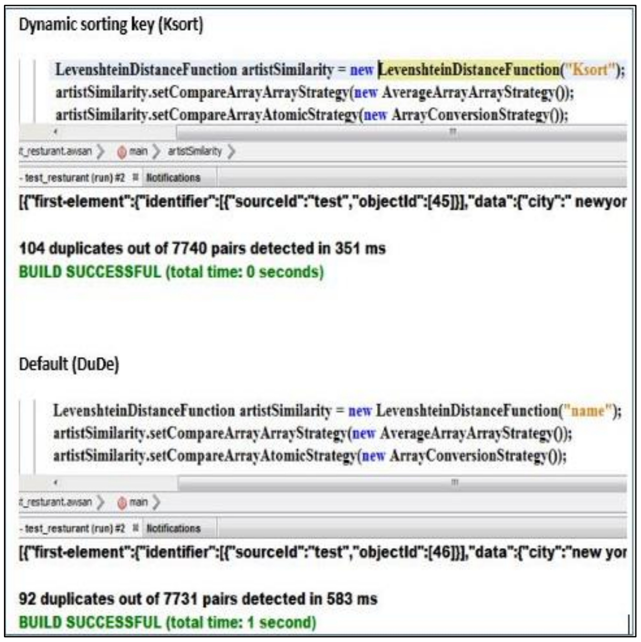
Listing 2. Configuration of the comparator

```
levComp = new LevenshteinDistance
Comparator("ksort");
```

As DuDe is used to classifying duplicate pairs, the post processor is configured to compute the transitive closure and to write the final output of the pairs that are classified as a duplicate in Json file.

V. Results, Analysis and Discussions

As described in the previous section, the experiment produces two sets of result namely DuDe with dynamic sorting key's (denoted as Dude') and the original Dude's (denoted as Dude). The results show that using Dude', the number of comparison pairs for the restaurant data set was 7,740 pairs and the number of duplicate records detected was 104. With Dude, lower number of pairs of comparisons and duplicate records was yielded, with 7,731 and 92 respectively. Figure 8 shows the results of the experiments in Dude's toolkit interface.



Dynamic sorting key (Ksort)

```
LevenshteinDistanceFunction artistSimilarity = new LevenshteinDistanceFunction("Ksort");
artistSimilarity.setCompareArrayArrayStrategy(new AverageArrayArrayStrategy());
artistSimilarity.setCompareArrayAtomicStrategy(new ArrayConversionStrategy());
```

```
[{"first-element":{"identifier":[{"sourceId":"test","objectId":[45]}],"data":{"city":" newyor

104 duplicates out of 7740 pairs detected in 351 ms
BUILD SUCCESSFUL (total time: 0 seconds)
```

Default (DuDe)

```
LevenshteinDistanceFunction artistSimilarity = new LevenshteinDistanceFunction("name");
artistSimilarity.setCompareArrayArrayStrategy(new AverageArrayArrayStrategy());
artistSimilarity.setCompareArrayAtomicStrategy(new ArrayConversionStrategy());
```

```
[{"first-element":{"identifier":[{"sourceId":"test","objectId":[46]}],"data":{"city":"new yor

92 duplicates out of 7731 pairs detected in 583 ms
BUILD SUCCESSFUL (total time: 1 second)
```

Fig. 8. Experiment Results

According to the results of the experiments, out of 112 actual duplicates, Dude' managed to detect 92% of it, with the ratio of false negative equals to 7.14%. This error rate is low and it suggests that that Dude' offers more accuracy than Dude with 82% of duplicates detected and 17.86% of false negatives. In terms of detection speed, Dude' has demonstrated a lower run time of 351 ms as compared to DuDe with 583 ms. Figure 9 illustrates the results.

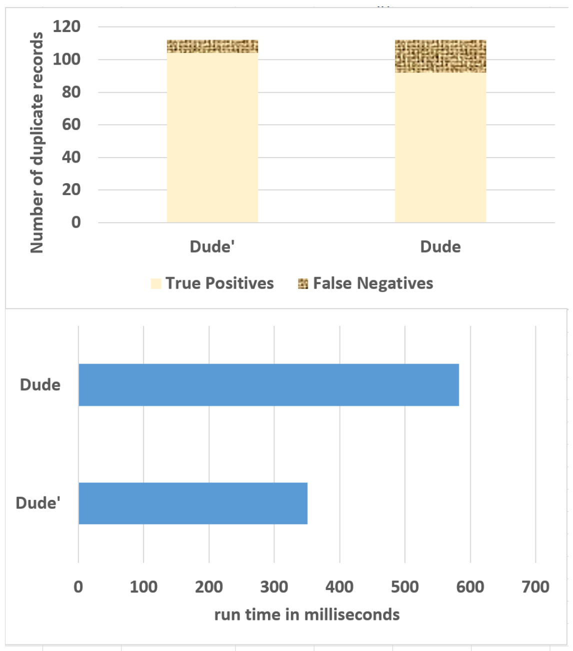


Fig. 9. Comparison of Accuracy between DuDe' and DuDe

VI. Conclusion

As a conclusion, detecting duplicates within a data set is a challenge especially when missing values are present. In this paper, we focus to answer the question of how duplicate records can be accurately detected within incomplete data sets. In search of the answer, we proposed an extension to SN blocking method that adds dynamic sorting key algorithm within the attribute selection step. To evaluate the proposed method, we conducted an experiment where a real restaurant data set and DuDe toolkit were used. The results show that not only the number of false negatives duplicates can be reduced, the proposed method is also faster than the duplicate detection that was performed without the proposed extension. Nevertheless, whether the proposed method behaves well or not in duplicate detection using other blocking methods such as bigram indexing and canopy clustering with TFIDF (Term Frequency/Inverse Document Frequency) is an open problem for future work.

Acknowledgment

We would like to thank Hasso-Plattner-Institute (HPI) for the data set and duplicate detection tool (DuDe) used this research. We also acknowledge financial assistance received from the Universiti Teknikal Malaysia Melaka (UTeM) in publishing this work.

References

[1] U. Draisbach and F. Naumann, "A generalization of blocking and windowing algorithms for duplicate detection," *Data and Knowledge Engineering (ICDKE), 2011 International Conference on. IEEE*, pp. 18–24, 2011.

[2] B. Carlo and M. Scannapieca, *Data Quality*. ACM Computing Classification, 2006.

[3] S. Yan, D. Lee, M.-Y. Kan, and L. C. Giles, "Adaptive Sorted Neighborhood Methods for Efficient Record Linkage," *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 185–194, 2007. [Online]. Available: http://doi.acm.org/10.1145/1255175.1255213

[4] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989.

[5] J. T. J and V. Saravanan, "Token-based method of blocking records for large data warehouse," *Advances in Information Mining*, vol. 2, no. 2, pp. 5–10, 2010.

[6] P. Christen, "Improving data linkage and deduplication quality through nearest-neighbour based blocking," *Discovery*, 2007.

[7] M. A. Hernández and S. J. Stolfo, "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem," *Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 9–37, 1998.

[8] J. Shin, "Comparative Study on Blocking Methods in Record Linkage," Ph.D. dissertation, Oklahoma State University, 2009.

[9] M. A. Hernández and S. J. Stolfo, "The Merge / Purge Problem for Large Databases," *ACM SIGMOD Record*, pp. 127–138, 1995.

[10] N. Emran, S. Embury, and P. Missier, "Model-driven component generation for families of completeness," in *6th International Workshop on Quality in Databases and Management of Uncertain Data, Very Large Databases (VLDB)*, 2008.

[11] N. A. Emran, "Data completeness measures," in *Advances in Intelligent Systems and Computing*, vol. 355, 2015, pp. 117–130.

[12] T. D. Pigott, "A Review of Methods for Missing Data," *Educational Research and Evaluation*, vol. 7, no. 4, pp. 353–383, 2001.

[13] N. J. Horton and K. P. Kleinman, "Much Ado About Nothing: A Comparison of Missing Data Methods and Software to Fit Incomplete Data Regression Models," *The American Statistician*, 2007.

[14] Y. Van Gennip, B. Hunter, A. Ma, D. Moyer, R. de Vera, and A. L. Bertozzi, "Unsupervised record matching with noisy and incomplete data," 2018.

[15] J. Tamilselvi and V. Saravanan, "Detection and elimination of duplicate data using token-based method for a data warehouse: A clustering based approach," *International Journal of Dynamics of Fluids*, vol. 5, no. 2, pp. 145–164, 2009.

[16] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederé, and W. Nejdl, "A blocking framework for entity resolution in highly heterogeneous information spaces," *IEEE Transactions On Knowledge and Data Engineering*, 2013.

[17] J. Jebamalar Tamilselvi and C. Brilly Gifta, "Handling Duplicate Data in Data Warehouse for Data Mining," *International Journal of Computer Applications*, vol. 15, no. 4, pp. 7–15, 2011.

[18] P. Christen and K. Goiser, "Quality and Complexity Measures for Data Linkage and Deduplication," *Springer*, pp. 127–151, 2007.

[19] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Technique (The Morgan Kaufmann Series in Data Management Systems)*. Elsevier, 2011.

[20] U. Draisbach and F. Naumann, "DuDe: The Duplicate Detection Toolkit," *Proceedings of the International Workshop on Quality in Databases (QDB)*, 2010.

[21] C. K. Enders, *Applied Missing Data Analysis*. The Guildford Press, 2010.