International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-9, July 2020

An Efficient Multi-Phase Blocking Strategy for Entity Resolution in Big Data

Randa Mohamed Abd El-Ghafar, Ali Hamed El-Bastawissy, Eman S. Nasr, Mervat H. Gheith

Abstract: Entity Resolution (ER) is the process of identifying records that refer to the same real-world entity. It plays a key role in many applications as data warehouse, data integration, and business intelligence. Comparing every record with all corresponding records is infeasible especially for a big dataset. To overcome such a problem, blocking techniques have been implemented. In this paper, we propose a novel Efficient Multi-Phase Blocking Strategy (EMPBS) for resolving duplicates in big data. As per our knowledge, some state of art blocking techniques may result in overlapping blocks (i.e. Q-grams) which cause redundant comparisons and hence increase the time complexity. Our proposed blocking strategy has disjoint blocks and less time complexity compared to Q-grams and slandered blocking techniques. In addition, EMPBS is general and requires no restrictions on the type of blocking keys. EMPBS consists of three phases. The first one generates three single efficient blocking keys. The second phase takes the output of the first phase as an input to construct a compound key. The compound key is composed of concatenation of two single blocking keys. Three compound blocking keys are the output of this phase that will be used as an input for the last phase, which is generating the Efficient Multi-Phase Blocking Key (EMPBK). EMPBK is constructed using the union of two compound blocking keys. The implementation of EMPBS presents promising results in terms of Reduction Ratio (RR) as it achieves a higher value of RR than adopting only a single blocking key, while at the same time maintains nearly the same precision and recall. EMPBS reduced about 84% of the average number of comparisons accomplished in a single blocking key. To evaluate EMPBS, we developed a Duplicate Generation tool (DupGen) that accepts a clean semi-structured file as an input and generates labeled duplicate records according to certain criteria.

***Keywords**: Entity resolution, record linkage, big data, blocking techniques.*

Revised Manuscript Received on July 30, 2020.
* Correspondence Author

Randa M. Abd El-ghafar*, Department of Computer Science, Faculty of Graduate Studies for Statistical Research, Cairo University, Cairo, Egypt. E-mail: randa.mohamed@mof.gov.eg

Ali H. El-Bastawissy, Faculty of Computer Science, Modern Sciences and Arts University, Cairo, Egypt. E-mail: aelbastawissy@msa.eun.eg

Eman S. Nasr, Independent Researcher, Cairo, Egypt. E-mail: nasr.eman.s@gmail.com

Mervat H. Gheith, Department of Computer Science, Faculty of Graduate Studies for Statistical Research, Cairo University, Cairo, Egypt. E-mail: Mervat_gheith@yahoo.com

I. INTRODUCTION

Entity Resolution is the process of identifying the records that refer to the same real-world entity across multiple heterogeneous data sources. This is also known as record linkage, duplicate detection, entity matching, or object reconciliation. With the massive amount of data generated everywhere, from smart devices to traditional database transactions, to social media, to IOT, and others, there is an increasing number of application domains that requires ER. These applications may include e-commerce, for catalog fusion; security, to detect frauds; customer database integration, and more [1]. Data cleaning is the task of identifying incorrect, incomplete, irrelevance, inaccurate, inconsistency, and irrelevant data, fix such problems and make sure that all these issues with be automatically fixed. ER is a very important and challenging task in the data cleaning process because it highly affects the quality of the whole data being examined. The ER process includes many tasks: (1) data pre-processing, (2) indexing or blocking using one attribute or more, (3) using one or more classifier to compare records, (4) classifying the records to (duplicate, non-duplicate or possible duplicate), and (5) delivering the results [2]. Blocking is used to achieve high efficiency especially with a large dataset where data is partitioned to several chunks or blocks depending on one blocking key or more. Partitioning is done in a way that guarantees the similarities between elements of the same block and dissimilarities outside it. Traditional ER only focuses on detecting duplicates that reside on structured datasets. It involves many challenges such as different attribute names, abbreviations, and errors in data.

ER for big data is not a trivial task because it involves many challenges that arise due to the characteristics of big data itself. Those challenges include variety of data sources, different data structures, volume of data, and the rapid increasing rate of arriving, managing, and storing data.

All these reasons call for new approaches to perform ER in an effective and efficient manner.

Traditional blocking techniques that generate blocks depending on one attribute or more suffers from many issues as follows [3]:

1) Selecting which attribute to choose as a blocking key to avoid searching for all possible pairs (feature engineering).

2) How to define a good blocking key strategy is a difficult and error-prone task that generally requires a domain expert to avoid looking at all possible pairs (efficiency issue).
3) Which models to use to estimate matches and non-matches probabilities (modeling issue).
4) How to select the threshold values (evaluation issue).

To evaluate the proposed strategy, we need a labeled dataset with a pre-defined state of true and false duplicates.

In real-world applications, no labeled dataset is available for testing ER approaches because it requires manual labeling of the dataset, which is a very hard and time-consuming task. In addition, there is a clear lack of public datasets combined with a golden standard dataset that could be used for testing ER approaches. To overcome these challenges, we develop a DupGen, a Duplicate Generation tool that accepts semi-structured datasets as input and generates duplicate records based on pre-defined criteria. The generated file will be combined with a golden standard file to test our proposed strategy. The main contribution of the paper could be summarized as follow:

- It presents DupGen, a Duplicate Generation tool, which is used to generate duplicate records according to certain criteria such as (Insert/ Delete/ Swap) char(s) or word(s). The generated dataset will be used to evaluate the proposed multi-phase blocking strategy.
- It proposes an Efficient Multi-Phases Blocking strategy (EMPBS). EMPBS consists of three phases, which are used to generate an efficient blocking key. Experimental results show that the generated key achieve an obvious reduction in execution time and in number of comparisons compared to that achieved in the first phase, while at the same time maintain the same precision and recall.

The rest of this paper is organized as follows. Section II discusses the related work. Section III presents the proposed approach. Section IV discusses the experimental results. Finally, section V gives the conclusion and future work.

II. RELATED WORK

ER is a very time-consuming task, especially for big datasets. Therefore, the blocking techniques have been used to divide the dataset into blocks such that the data resides in the same block is more likely to be matched. Therefore, the records reside in the same block are only compared to detect duplicate entities. Blocking techniques are used to improve the efficiency of ER algorithms.

Papadakis et al. [4] proposed a novel taxonomy of the blocking that consists of the seven dimensions, which can be summarized into the table I. Many blocking methods could be utilized such as:

Standard blocking (SB): It is the simplest form of blocking where an expert selects the most suitable attributes and a transformation function is applied to formulate the blocking key. For every distinct key, a block is created to contain all the corresponding entities. SB has two main advantages:

- It results in a disjoint block (no redundant comparisons).
- It has a linear time complexity O(|E|).

Q-gram blocking [5], [6]: It is a transformation function that converts the blocking key of SB to a sequence of Q characters (Q-grams) and defines a block for every distinct Q-gram. As an example, for Q=3, key "france" is transformed into {fra, ran, anc, nce}. This approach is more resilient to noise than SB. More advanced techniques as extended Q-grams blocking have been proposed in [5], [6], [7], which use a combination of Q-grams instead of individual ones.

Sorted Neighbourhood (SN) [8]: First, it sorts all the blocking keys to arrange the associated entities accordingly then, a window of fixed size is sliding over the entities (E) in the same windows (W). This approach has the following advantages:

- Results in a linear ER complexity (O(W.|E|)).
- It is robust to noise because it can discover errors at the end of the blocking key. On the other side, the performance of it depends heavily on the size of the window, which is difficult to configure.

Table-I: Taxonomy of the blocking dimensions

#	Dimension	Sub-type	Definition
1	Key selection	Rule-based.	Assign entities to blocks based on rule derived from expert knowledge or heuristics.
		ML-based.	Require training set and use ML to learn the best blocking key.
2	Schema awareness	Schema aware.	Presume Schema knowledge for mapping entities into blocks. Extract the blocking key from specific attributes.
		Schema agnostic.	Do not rely on schema knowledge. Extract the blocking key from all attributes.
3	Key types	Hash (equality) based methods.	Map a pair of entities to the same block if they have a common key.
		Sort or similarity-based methods.	Map a pair of entities to the same block if they have a similar key.
		Hybrid method.	Combine hash with sort-based functionality.
4	Redundancy awareness	Redundancy free method.	Assign every entity to a single block. Results in disjoint blocks.
		Redundancy positive method.	Place every entity into multiple blocks. Results in overlapping blocks. The more blocks two entities share the more similar they are.
		Redundancy neutral method.	Results in overlapping blocks. Most pairs of entities share the same number of blocks.
5	Constraint awareness	Lazy.	Impose no constraints on the blocks they create.
		Proactive.	Enforce one or more constraints on their blocks as max. block size or try to refine their comparison by discarding the unnecessary one.
6	Matching awareness	Static Methods.	It is independent of the later entity matching process. Producing a static block collection.
		Dynamic Methods.	Intertwine block building with the later entity matching process. Updating or processing the blocks dynamically as more duplicates are detected.
7	Domain	Structured.	Structures or relational datasets.

		Semi-structured.	Semi-structured datasets as XML, and RDF
		Structured and Semi-structured.	Both of Structured and Semi-structured

To improve the effectiveness of a fixed size window, many techniques have been proposed such as multi-pass SN [9]. The core of multi-pass SN is applied multiple times using different transformation functions in each iteration.

This guarantee that more duplicate could be identified even with a small window size. Another solution is the extended SN, which slides a fixed size window over a sorted list of blocking keys. The variation of the window size has a large impact on the efficiency and effectiveness of the ER. Advanced strategies adapt the window size dynamically in order to obtain better efficiency. Such advanced techniques could be grouped into three categories depending on the criterion of moving the top and bottom boundaries of the window [4]:

1) Key similarity strategy: the size of the window increases if the similarity of the blocking key exceeds a pre-determined threshold.
2) Entity similarity strategy: the size of the window relies on the similarity of the entities within the current window. In incrementally adaptive SN [10], the window size increases if the distance between the first and the last element in the current window is smaller than a pre-defined threshold. Where in the accumulative adaptive sorted neighborhood [10], a window is created with a single overlapping entity and exploits transitively to group multiple adjacent windows into small blocks if the entity of one window is a potential duplicate of the last entity in the next adjacent window.

Dynamic Strategy: the window size is changed according to the assumption that the more duplicates are found in a window; the more are expected by increasing the size of the current window. Two strategies have been proposed; duplicate count strategy (DCS) and Duplicate Count Strategy ++ (DCS++) [11]. DCS defines a fixed size window for every entity of SN's sorted list. All comparisons executed within the window; in addition, the ratio d/c is estimated where d is the newly detected duplicates and c is the executed comparisons. The window size is increased by one if $(d/c > \Phi)$ where $\Phi \in (0, 1)$ which determines the threshold that identifies the average number of duplicates per comparisons. DCS++ [11] increases the window size with the next (w-1) entities even if the new ratio becomes lower than Φ. It also uses transitive closure to skip some comparisons.

ML-based methods: there are two types of machine learning methods: supervised and unsupervised. In supervised learning approaches, the dataset contains labeled data, which indicates a positive and negative duplicate. Since it is hard to obtain a pre-defined labeled dataset, unsupervised ML has been proposed. The Blocking Schema Learner (BSL) learns the supervised blocking [12] schema that maximizes the Reduction Ratio (RR) while maintaining Pairs Completeness (PC) above the pre-determined threshold. Its output is a disjunction of the conjunction of blocking predicates. Fisher Disjunchine [13] is an unsupervised ML algorithm that uses a weak training dataset, which is generated, by using the TF-IDF similarity of two entities. Pairs with low similarity values are considered non-duplicate and vice-versa.

Parallelization approaches: Most of the parallel ER adopts MapReduce frameworks [14]. MapReduce framework consists of two functions, map and reduce. MapReduce splits the input dataset into blocks and processes each block in parallel. Map function produces intermediate (Key, Value) for each input split, while reduce function process a list of values produced by the map function. In Schema-aware methods, hash-based methods are adopted in the MapReduce framework. The map phase implements the transformation function(s) and produces the (blocking key, entity id) pairs for each entity. The reducer phase is responsible for creating the block and aggregates all pairs within the same blocking key. Dedoop [15] provides an implementation of such methods. The sorted neighborhood is also adopted in MapReduce [16]. The map function extracts the blocking key(s) from input entities then those entities are sorted in alphabetical order by their keys.

The reducer function slides a window of fixed size within every reduce partition. Entities close to partition boundaries need to be compared across different reducer tasks. The map function is extended to replicate those entities and forwarded them to perspective reduce tasks. Regarding the schema-agnostic methods, token blocking builds an inverted index that associates every token with all entities containing it in the value of their attributes. For parallelization, a single MapReduce job is required. For entity e_i, a map function produces a (t, e_i) pairs for every token t in the values of e_i. Next, all entities that have a particular token are processed in the same reducer [17], [18].

III. THE PROPOSED EFFICIENT MULTI-PHASE BLOCKING STRATEGY (EMPBS) FOR ENTITY RESOLUTION

We proposed an <u>E</u>fficient <u>M</u>ulti-<u>P</u>hase <u>B</u>locking <u>S</u>trategy (EMPBS) to solve the problem of ER for the big data. To build the proposed EMPBS, we faced several challenges:

1) The difficulty to acquire data for testing and evaluating ER algorithms and techniques, as ER is often dealing with confidential or sensitive data such as names and addresses. It is also hard to use ER systems effectively without knowing the status of its record pairs in advance. In addition, there is a clear lack of public datasets that could be used for the ER process where it can be used as a golden dataset for testing the ER algorithms, as the only available datasets are very small size.
   However, due to the privacy and confidentiality of data, it seems that such data will never be publicly available. An alternative way is to use an artificially generated dataset where the amount of errors is introduced as well as the linkage status of the record pairs is known in advance.
   The first introduction of such dataset's generator is first introduced in [8]. To our knowledge, there is no public dataset's generator, which allows the creation of datasets with duplicate records.

All the above reasons are motivating us to develop an Artificial Duplicate Generator Tool (DupGen) which is used to artificially generate datasets with a well-known record pair status.

Duplicate records are generated upon many numbers of criteria as illustrated in the next subsection.

2) Strategy evaluation. The existence of a labeled dataset is very important for evaluating the proposed Multi-blocking strategy.

As discussed above, the existence of such labeled data for ER is very rare or does not exist.

Therefore, in order to evaluate the proposed strategy, we generate a golden dataset that contains the real true positives and true negatives matches of the synthetic dataset generated using DupGen.

3) High computational complexity. It happens due to the high number of comparisons between entities. To avoid the high computational complexity, blocking is used to distribute the big dataset into several blocks and only run the ER between elements residing in the same block, not in the whole dataset.

Dataset will be partitioned using the proposed blocking strategy EMPBS such that entities reside in the same block are more likely to be matched.

In addition, to enhance the performance, detecting similarity between blocks could be running in parallel, especially for big datasets using Apache Spark, which is a framework for big data processing that supports in-memory computations.

The next subsections will describe DupGen and EMPBS in detail.

A. The Duplicate Generator Tool (DupGen).

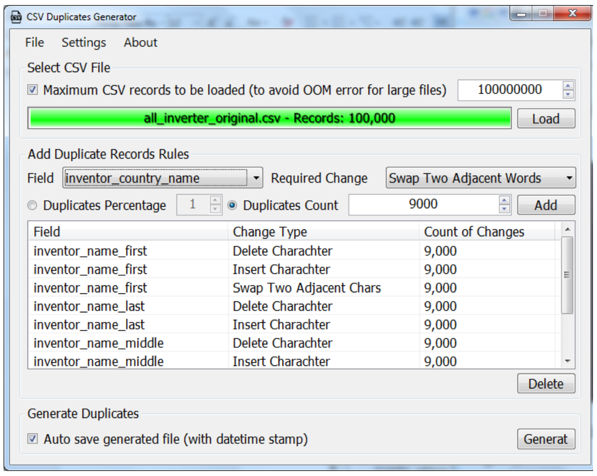
DupGen is a Duplicate Generator tool developed to generate the synthetic dataset. It accepts CSV file as an input, reads it, and generates duplicate according to certain criteria as illustrated in Fig. 1. These criteria include:

1) The required number or percentage of artificially generated records.
2) The required changes to the value of the attributes. These changes include:
   - Insert (char(s), word(s)).
   - Delete (char(s), word(s)).
   - Swap (char(s), word(s)). Which may be adjacent or not.
   - Duplicate (char(s), word(s)).

Multiple changes could be allowed or not for the same record according to user choice. Duplicate records are differentiated from original records by a flag (original/duplicate).

The original file (inventor) has 100000 records. It includes the attributes: application number, inventor name first, inventor name middle, inventor name last, inventor rank, inventor region code, inventor country code, inventor country name, and inventor address type.

We generate 9000 duplicate records from the original file according to the following criteria that are described in Table II.



☑ Maximum CSV records to be loaded (to avoid OOM error for large files) 100000000

all_inverter_original.csv - Records: 100,000

Field: inventor_country_name — Required Change: Swap Two Adjacent Words

Duplicates Percentage: 1 — [✓ Duplicates Count]: 9000

Field	Change Type	Count of Changes
inventor_name_first	Delete Charachter	9,000
inventor_name_first	Insert Charachter	9,000
inventor_name_first	Swap Two Adjacent Chars	9,000
inventor_name_last	Delete Charachter	9,000
inventor_name_last	Insert Charachter	9,000
inventor_name_middle	Delete Charachter	9,000
inventor_name_middle	Insert Charachter	9,000

☑ Auto save generated file (with datetime stamp)

Fig. 1.Duplicate generation tool

The duplicate dataset has 109000 records and has an additional four attributes that will be used for testing. The additional attributes are OriginalID, IsOriginal, number of changes, and changes description. The attribute "Original ID" indicates the ID of the original record while the attribute "IsOriginal" specifies if the record is original or not. It takes two values true or false. The attribute "number of changes" counted the changes made to the original file. Finally, the attribute "changes description" illustrated the changes made to the attributes of the original record where the original record may have more than one change. Table III illustrates a toy example of duplicate records generated by DupGen tool where four records have been generated from the original record that has application number 5923961. The synthetic records have IsOriginal flag with value ="False" and no. of changes which indicate the number of changes has been made to the original record. For example, number 3 illustrates that three changes have been made to the original record and these changes are described in the "change description" attribute.

The generated synthetic dataset will be investigated in order to generate blocks and resolve duplicates using the proposed strategy. The next subsection describes the proposed EMPBS in detail.

B. The proposed Efficient Multi-Blocking Phase Blocking Strategy (EMPBS).

As mentioned before, traditional ER approaches apply matching techniques that result in O(n2) comparisons, which causes massive execution time for a big dataset. Blocking techniques have been produced to reduce such execution time by distributing the dataset to several blocks or chunks and only compare records in this block. Blocks are processing in parallel to solve the problem of the enormous size of the dataset and enhance the performance of ER approaches. Blocks could be constructed using many techniques such as using the initial three characters of a certain field. In addition, many blocking methods could be used such as Sorted Neighborhood, Standard Blocking, Q-gram Indexing, and Canopy Clustering with TF-IDF [14].

In order to reduce the time for similarity computation, we propose an Efficient Multi-Blocking Phase Strategy (EMPBS). EMPBS consists of three subsequent phases that begins with the single blocking key generation, which is created from a pre-defined bag of blocking methods, and ends with the generation of the efficient blocking key. The algorithm of the proposed EMBPS is illustrated in figure 2. The three phases of algorithm are discussed in detail in the next subsections.

Table- II: Criteria used to generate duplicate datasets.

Attribute Name	Insert Char(s)	Delete Char(s)	Swap two adjacent char(s)	Swap two adjacent words(s)
Attribute_name_first	√	√	√	
Attribute_name_last	√	√		
Attribute_name_middle	√	√	√	
Inventory_country_name				√

Table-III: Synthetic records generated by DupGen tool

Attribute name	Record 1	Record 2	Record 3	Record 4	Record 5
Application_number	5923961	5923961	5923961	5923961	5923961
Inventor_name_first	CLAUDINE	CLUDINE	CLAUDIN!E	CLAUDIN!E	CLAUDIN!E
Inventor_name_middle				4	4
Inventor_name_last	MAURIC	MAURIC	MAURIC	MAURIC	lMAURIC
Inventor_rank	1	1	1	1	1
inventor_region_code					
Inventor_country_code	CH	CH	CH	CH	CH
Inventor_country_name	SWITZERLAND	SWITZERLAND	SWITZERLAND	SWITZERLAND	SWITZERLAND
Inventor_address_type	residence	residence	residence	residence	residence
Original ID	24318	24318	24318	24318	24318
Is Original	TRUE	FALSE	FALSE	FALSE	FALSE
No. Of Changes	0	1	1	2	3
Changes Description		[inventor_name_first] Delete Charachter	[inventor_name_first] Insert Charachter	[inventor_name_first] Insert Charachter & [inventor_name_middle] Insert Charachter	[inventor_name_first] Insert Charachter & [inventor_name_middle] Insert Charachter & [inventor_name_last] Insert Charachter

1) The Single Blocking Key (SBK) generation

This is the first step of generating an efficient multi-phase blocking key. This phase begins by blocking the sample dataset by selecting one of the blocking keys from a bag of blocking methods and attributes (BBMA) such as first name, Soundex of the first two characters of first name, last name, and middle name. After constructing the SBKs, then the candidate pairs are generated using the blocking keys, then the similarity between each candidate pairs is calculated.to construct the similarity vector, Jaro-winkler and exact match methods are used for the attributes first name, middle name, last name, region, and country name. In the last step in this phase, we used Golden DataSet (GDS) to calculate the effectiveness of the SBK.

```
Inputs: Big dataset, Golden DataSet (GDS), a Bag of Blocking
Methods and Attributes (BBMA).
Output: Single Blocking Key (SBK), Compound Blocking Key
(CBK), Efficient Blocking Key (EBK).
Phase 1: The Single Blocking Key (SBK) generation
//Select a sample of the generated dataset such that it contains
true duplicates and false duplicate.
For (iter =0; iter < count(BBMA); iter ++) {
    Generate (SBK using (BBMA [iter]))
    Count (candidate pairs) using BBMA [iter]
    Construct the similarity vector for the candidate pairs
    Calculate effectiveness using GDS
    Calculate efficiency ( )
    SBK.add (BBMA[iter], F-measure) }
Phase 2: Construction of the Compound Blocking Key
(CBK)
//Select the three Single Blocking Keys (SBK) that achieve the
highest F-measure.
k=0;
For (i= 0; i<3; i++) {
    For (j= i; j<2; j++){
        CBK[k] = SBK[j]+ SBK[j+1];
        K++; }
}
    Foreach cbk in CBK{
        Block the dataset using cbk
        Generate the candidate pairs using cbk
        Count (candidate pairs)}
    Foreach candidate pair in candidate pairs{
            Calculate the similarity vector
            Calculate using GDS
            Calculate efficiency ( ) }
Phase 3: Construction of the Efficient Multi-Phase Blocking
Key (EMPBK)
K=0;
For (i=0; i<2; i++){
    //Construct EBPBK such that it is composed of the union of 2
    two cbk
    For (j=i; i<3;i++) {
            EBPBK [k] = cbk[j] U cbk[j+1] }
            foreach ebk in EBPBK {
              Block the dataset using ebk
              Generate the candidate pairs using ebk
              Count (candidate pairs)}
            foreach candidate pair in candidate pairs{
              Calculate the similarity vector
              Calculate using GDS
              Calculate the efficiency ( )
              }}
```

Fig. 2.The proposed efficient multi-phase blocking algorithm for Big Dataset

2) The Compound Blocking Key (CBK) generation

This phase takes the generated SBKs as an input where the Compound Blocking Key (CBK) is composed of a concatenation of more than a SBK. Many compound keys could be constructed, but we are concerned of the ones that achieve the best performance.

To accomplish that, we made different CBKs using the heuristic "The compound key is constructed using the SBKs that achieve the highest efficiency". For example, constructing the compound key as the Soundex of the first two characters of the last name concatenated with the Soundex of the first character of the middle name as they achieve the best f-measures when taking them as a SBK. The next step is the generation of the candidate pairs using the constructed compound keys and calculate the similarity between them using the same methods in the SBK generation. Then, the effectiveness of the CBK is calculated using the GDS.

3) The Efficient Multi-phase Blocking Key (EMPBK) generation

Generating the EMPBK is the last phase of EMPBS. It takes the constructed CBKs as input to generate the EMPBKs. EMPBK is constructed by the union of two CBKs. Numerous EMPBKs could be generated, but we are interested in the one that achieves the best performance. To achieve this, we construct it by using the heuristic "The CBKs that achieve the best performance will in return achieve highest efficiency with a reduction in the number of candidate pairs. This will in turn reduce the execution time". For example, using the union of (Soundex of the first two characters of the last name concatenated with the Soundex of the first character of the middle name) and (region code concatenated with the Soundex of the first two characters of the last name) as they achieve the best performance when adopting them as a compound key. Then, the candidate pairs are generated using the EMPBKs. After generating the candidate pairs using the blocking key of each of the three phases, similarities between the attributes are calculated. There are different methods that could be adapted to calculate the similarity; these methods include character distance similarity-based, token similarity-based, Phonetic similarity-based, and numeric similarity-based [19], [20]. On each candidate pairs, we run the similarity functions in order to produce a comparison vector for each pair. To computer the comparison vectors, a similarity function is defined using the columns: first name, middle name, last name, region code, and country-code as illustrated in Table IV. Exact match is used for attributes region and country code, while Jaro-Winkler is used for first, middle, and last name. Jaro-Winkler is used a lot in record linkage projects because it gives priority to the leading characters of the string and usually the errors exist in the trailing characters. To classify a comparison vectors as matched or non-matched, we used a weighted average over the computed vectors to get the final score of similarity. We tried many weights and run them to see the performance of the ER classifier. We choose weight [30, 10, 40, 5, 15] based on heuristic "more frequency items are given higher weights". Experiments show that this heuristic gives a better recall and precision. We classified records as matched if they exceed a threshold of 0.8.

Table-IV: Similarity matrix

Attribute	Similarity Method
'inventor_name_first'	'jarowinkler' method
'inventor_name_middle'	
'inventor_name_last'	
'inventor_region_code'	Exact match
'inventor_country_code'	

IV. EXPERIMENTAL RESULTS.

EMPBS is evaluated using the synthetic data generated by our DupGen tool. In this section, we evaluate EMBPS in terms of effectiveness and efficiency. We run our experiments on local PC that has intel ® core ™ i7.35174 CPU @ 1.90 GHZ 2.4 GHZ , and 16 GB RAM. For developing our strategy, we used Apache Spark 2.4.3 installed on Windows 7 operating system. We used the inventor dataset, a synthetic dataset generated by DupGen tool developed by us. Inventor dataset has 109000 entities, nine original attributes, and four non-original attributes generated for testing our proposed strategy. To measure the effectiveness of EMPBS, four matrices has been applied:

1) Recall, which estimates the portion of the matches that were discovered by the classifier. Recall is calculated by

$$\frac{True\ positives}{True\ positives + False\ Negatives} \tag{1}$$

2) Precision, which estimates the portion of the executed comparisons that results in matches. It is calculated by

$$\frac{True\ positives}{True\ positives + False\ positives} \tag{2}$$

3) F-measure, which is the harmonic mean of precision and recall, and it is calculated by

$$\frac{2 \times precison \times Recall}{precision + Recall} \tag{3}$$

4) Reduction Ratio (RR) that estimates the portion of comparisons, which are avoided with respect to the comparison achieved by the Cartesian product.

In terms of efficiency, we measure the whole execution time of the whole phases involved in the ER process. Next, are the results of the efficiency and effectiveness of EMPBS.

In this section, we evaluate the efficiency and effectiveness of EMPBS. First, we conduct experiments using many single blocking keys as last name, first name, and region code. Then the resulted number of candidate pairs, performance time, precision, recall, F-measure, and RR are calculated. Table V demonstrates the obtained results of each phase of the proposed EMPBS. The SBKs that achieve the highest three precision, recall, and f-measure are keys having numbers seven, six, and three.

Those blocking keys represent the Soundex of the first two characters of the last name, the Soundex of the first character of the middle name, and region code respectively. As illustrated in Fig. 3, the preceding blocking keys archive 83%, 81%, and 80.8% F-measure respectively.

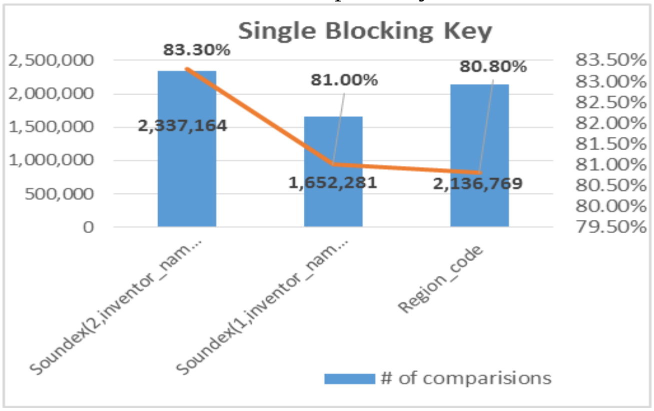


Fig. 3. The number of comparisons and f-measure achieved in best three single blocking key phase

Therefore, these three single blocking keys will be nominated for the next phase of the blocking strategy as they achieve the heights f-measure.

The next phase is the compound blocking key generation where the compound blocking key is constructed using the concatenation of a two nominated keys resulted from the previous phase. For example, the compound-blocking key that achieves the best F-measure is composed of the two single blocking keys (Soundex of the first 2 letters of the last name & Soundex of the first letter of middle name) which achieves also the best f-measure (75.3%) as illustrated in figure 4. The last phase of the proposed strategy is the construction of the efficient multi-phase blocking key where it is the union of two compound blocking keys. For example, (Soundex of the first character of middle name and Soundex of the two characters of the last name) union (the region code and Soundex of the two characters of the last name) are constructed as a multi-phase blocking key. This key achieves 82.2% f-measure, which is better than the average of the f-measure of three single blocking keys that construct it (81.7 %) as illustrated in Fig. 5.

Table V illustrates the precision, recall, and f-measure of the proposed strategy. The effectiveness results are shown for different weights for the single blocking key, as the weights [30, 10, 40, 5, 15] accomplish the highest f-measure, we utilized it for the preceding two phases of EMPBS.

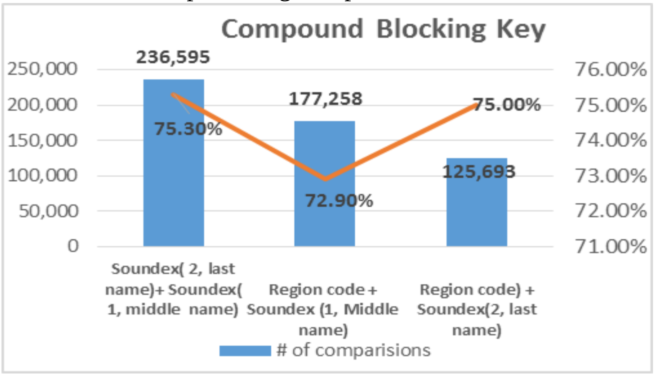


Fig. 4. The number of comparisons and f-measure achieved in the best three compound key

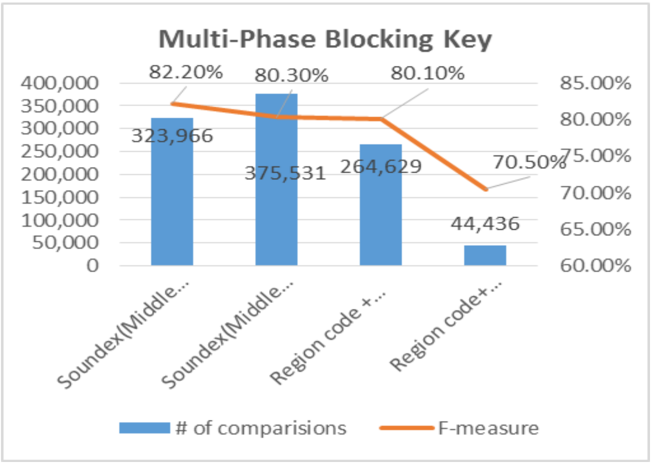


Fig. 5. The number of comparisons and f-measure achieved in the best three multi-phase blocking key

Reduction Ratio (RR) estimates the relative decrease in the number of comparisons conveyed by the blocking techniques. RR is very important for measuring the effectiveness of the ER because it calculates the percentage of the comparisons that are avoided after applying the blocking technique.

EMPBS presents promising results in terms of RR, where it achieves RR value higher than adopting only a single blocking key. Applying EMPBS fulfills an enhancement of the efficiency of ER since it reduces up to 84% of the average total number of comparisons accomplished in a single blocking phase. Table VI shows the average execution time, the number of comparisons, F-measure for the first two phases of EMPBS, which will be used in table VII to calculate accomplished improvement by EMPBS. Table VII illustrates the archived enhancement regarding the following three measures:

1) F-measure: Using the proposed EMPBS achieves 0.5 % enhancement for the best efficient blocking key (key 1) compared to the average of the best three single blocking keys, which are used to construct the final efficient blocking key. While comparing to the 2nd phase of the proposed strategy, it achieves an enhancement in all the proposed blocking keys.
2) The number of comparisons: EMPBS achieves a remarkable reduction in the number of comparisons regarding both the first and the second phases of the key generation. For example, the first efficient key (Key (1)) achieves 84% reduction in the number of comparisons compared to the first phase (single blocking key) and 79% reduction compared the second phase (compound key generation).
3) Execution Time: EMPBS achieves a remarkable decrease in execution time compared to the first phase. For example, the first proposed efficient key (key 1) accomplished a 75% reduction in execution time compared to the average execution time of the single blocking keys that construct it.

Table-V: Performance of Multi-Phase Blocking Strategy

ID	Blocking attribute	Number of pairs	Duration	Precision	Recall	F-Measure
	The Single Blocking Key (SBK)					
1	inventor_name_last	35,297	0:0:01	99.61	32.38	48.88
2	inventor_name_first	98,902	0:0:04	98.71	23.65	38.16
3	Region_code	2,136,769	0:0:34	97.92	68.84	80.84
4	inventor_name_middle	1,652,281	0:0:28	96.64	51.25	66.98
5	Soundex(2,inventor_name_first)	1,652,281	0:0:21	97.22	58.45	73.01
6	Soundex(1,inventor_name_middle)	1,652,281	0:3:38	97.37	69.23	80.92
7	Soundex(2,inventor_name_last)	2,337,164	0:1:28	98.80	72.03	83.32
	The Compound Blocking Key(CBK)					
1	Composite Blocking key of: Region code + (middle name + last name)	11,980	0:0:02	99.41	17.67	30.00
2	Composite Blocking key of: Soundex(2, last name)+ Soundex(1, middle name)	236,595	0:0:18	98.97	60.67	75.23
3	Composite Blocking key of: Region code + Soundex(2, first name)	119,917	0:0:11	97.78	48.58	64.91
4	Composite Blocking key of: Region code + Soundex (1, Middle name)	177,258	0:16	97.85	58.06	72.88
5	Composite Blocking key of: Region code) + Soundex(2, last name)	125,693	0:0:08	98.94	60.33	74.96
6	Composite Blocking key of: Region code + Soundex (1, Middle name) + Soundex (2, last name)	38,322	0:0:02	99.03	50.70	67.07
7	Composite Blocking key of: Region code + Soundex(2, first name) + Soundex(1, middle name) + Soundex(2, last name)	24,036	0:0:07	99.11	35.41	52.18
	The Multi-phase Blocking Key (MPBK)					
1	(Soundex(1, Middle name) + Soundex(2, last name)) U (Region code + Soundex(2, last name))	236,595 U 125,693 = 323,966 pairs	0:0:28	98.90	70.30	82.19
2	(Soundex(1, Middle name) + Soundex(2, last name)) U (Region code + Soundex(1, middle name))	236,595 U 177,258 = 375,531 pairs	0:0:20	97.97	68.04	80.30
3	(Region code + Soundex(1, last name)) U (Region code + Soundex(1, middle name))	125,693 U 177,258 = 264,629 pairs	0:0:38	97.94	67.69	80.05

Table-VI: The average execution time, number of comparisons, and f-measure of EMPBS

ID of SBK	# of comparisons	Precision	Recall	F-measure	Processing Time	# of comparisons AVG	F-measure AVG	Execution time AVG
7	2,337,164	98.8	72.03	83.30%	0:01:28	2,042,071	81.70%	00:1:53
6	1,652,281	97.37	69.23	81.00%	0:03:38			
3	2,136,769	97.92	68.84	80.80%	0:00:34			

ID of CBK	# of comparisons	Precision	Recall	F-measure	Processing Time	AVG			
						IDS of CBKs	Comparisons	F-measure	Execution time
2	236,595	98.97	60.67	75.30%	0:00:18	(2 U 5)	181,144	75%	0:00:13
4	177,258	97.85	58.06	72.90%	0:00:16	(2 U 4)	206,927	74%	0:00:17
5	125,693	98.94	60.33	74.90%	0:00:08	(5 U 4)	151,476	74%	0:00:12

ID of MPBK	IDS of CBKs	# of comparisons	Precision	Recall	F-measure	Execution Time
1	(2 U 5)	323,966	98.9	70.3	82.20%	0:00:28
2	(2 U 4)	375,531	97.97	68.04	80.30%	0:00:32
3	(5 U 4)	264,629	97.94	67.69	80.10%	0:00:20

Table-VII: The achieved reduction of the number of comparisons and execution time of EMPBS.

MPBKs	F-Measure		# of Comparisons		AVG Execution time	
	Compared to the 1st phase	Compared to the 2nd phase	Compared to the 1st phase	Compared to the 2nd phase	Compared to the 1st phase	Compared to the 2nd phase
1	0.5 %↑	7.1 % ↑	84% ↓	79% ↑	75% ↓	115% ↑
2	1.4 % ↓	6.2 % ↑	82% ↓	81% ↑	82% ↓	88% ↑
3	1.6 % ↓	6.2 % ↑	87% ↓	75% ↑	94% ↓	67% ↑

Fig. 6, Fig.7, and Fig. 8 conclude the results achieved by adopting proposed the MPBS, which could be summarized as:

1. It achieves a remarkable reduction in the number of comparisons compared to the average number of comparisons resulted in the single blocking keys that compose the efficient multi-phase blocking key.

2. It maintains almost the same f-measure with a huge decrease in execution time therefore adopting this strategy in big datasets will contribute in achieving a high efficiency due to the huge reduction of execution time.

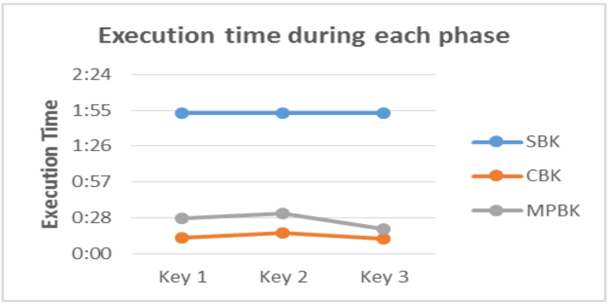


Fig. 6.The execution time during each phase of MPBS

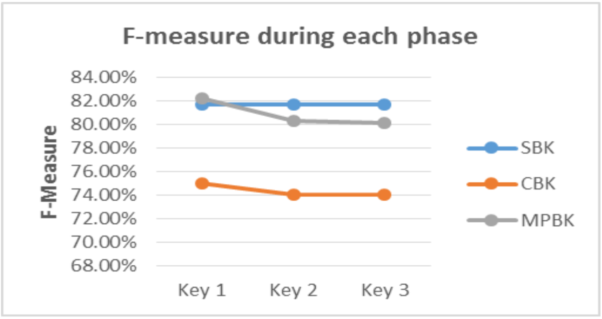


Fig. 7.The achieved F-measure during each phase of MPBS

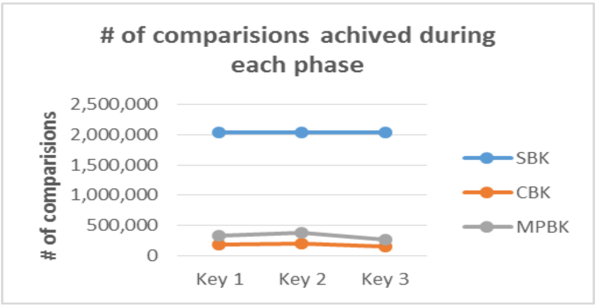


Fig. 8.The performed number of comparisons during each phase of MPBS

V. CONCLUSION AND FUTURE WORK

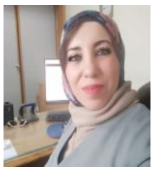
In this paper, we proposed EMPBS, an Efficient Multi-Phase Blocking Strategy for resolving duplicates in big data. EMPBS consists of three subsequent phases. In the first phase, three efficient blocking keys are generated. These keys are the input for the second phase, which constructs compound blocking keys from two single blocking keys generated in phase 1. The lase phase takes the output of the second phase to construct an Efficient Multi-Phase Blocking Key (EMPBK). EMPBK is constructed using the union of two compound blocking keys generated in the second phase. To implement and test our proposed strategy, we searched for a golden dataset but to the best of our knowledge, there is no publically available big dataset due to its sensitivity. Therefore, we developed DupGen, a duplicate generation tool that generates a duplicate dataset according to certain criteria. From the experimental results, we can highlight that EMPBS presents a better results regarding the efficiency of the single blocking key techniques. EMPBS achieved a remarkable increase in the Reduction Ratio (RR). In addition, it reduced about 84% of the average number of comparisons compared to only adopting a single blocking key without a negative impact of effectiveness. As future work, we intend to utilize a distributed parallel processing framework like Apache Spark to implement the proposed strategy with a big dataset in a real cluster to enhance blocking efficiency.

REFERENCES

1. L. GAGLIARDELLI, S. ZHU, G. SIMONINI and S. BERGAMASCHI, "Bigdedup: A Big Data integration toolkit for duplicate detection in industrial scenarios," in Proc. Int. Conf. on Transdisciplinary Engineering (TE2018), 2018.
2. P. A. Albanese and J. M. Ale, "Data matching and deduplication over big data using Hadoop framework," in CACIC, 2016.
3. G. Simonini, S. Bergamaschi and H. V. Jagadish, "BLAST: a Loosely Schema-aware Meta-blocking Approach for Entity Resolution," in PVLDB, 2016.
4. G. Papadakis, D. Skoutas, E. Thanos and T. Palpanas, "A servay of blocking and filtering techniques for entity resolution," Association for Computing Machinery (ACM), 2019.
5. P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, vol. 24(9), p. 1537–1555, 2012.
6. G. Papadakis, G. Alexiou, G. Papastefanatos and G. Koutrika, "Schema-agnostic vs Schema-based Configurations for Blocking Methods on Homogeneous Data," PVLDB, vol. 9(4), p. 312–323, 2015.
7. R. Baxter, P. Christen and T. Churches, "Comparison of fast blocking methods for record linkage," in ACM SIGKDD '03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, 2003.
8. M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases.," SIGMOD,, p. 1995, 127–138.
9. M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem.," in Data Min. Knowl. Discov, 1998.
10. P. Wang, C. Xiao, J. Qin, W. Wang, X. Zhang and Y. Ishikawa, "Local similarity search for unstructured text," in SIGMOD,, 2016.
11. U. Draisbach, F. Naumann, S. Szott and O. Wonneberg, "Adaptive windows for duplicate detection," in ICDE, 2012.
12. M. Michelson and C. A. Knoblock, "Learning blocking schemes for record linkage," AAAI, p. 440–445, 2006.
13. M. Kejriwal and D. P. Miranker, "An unsupervised algorithm for learning blocking schemes," ICDM, p. 340–349, 2013.

14. R. M. Abd El-Ghafar, M. H. Gheith, A. H. El-Bastawissy and E. S. Nasr, "Record Linkage Approaches in Big Data: A State Of," in 13th International Computer Engineering Conference (ICENCO), Cairo, 2017.
15. L. Kolb, A. Thor and E. Rahm, "Dedoop: Efficient Deduplication with Hadoop," VLDB Endow, vol. 12, no. 5, p. 1878–188, 2012.
16. L. Kolb, T. and E. Rahm, "Multi-pass Sorted Neighborhood Blocking with MapReduce," in Computer Science-Research and Development, 2012.
17. V. Efthymiou, K. Stefanidis, M. Hersche and V. Christophide, "Entity Resolution in the Web of Data.," in nternational World Wide Web Conference (WWW), Seoul, South Korea, 2014.
18. V. Efthymiou, K. Stefanidis and V. Christophides, "Big Data Entity Resolution: From Highly to Somehow Similar Entity Descriptions in the Web," IEEE Big Data, 2015.
19. W. W. Cohen, P. Ravikumar and S. E. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks," International Joint Conference on Artificial Intelligence, p. 73–78, 2003.
20. J. Mielke, "A phonetically-based phonetic similarity metric," in NELS, 2009.

AUTHOR PROFILE



Randa Mohamed is a master holder in computer science & information system. Currently, she is a PDH researcher in faculty of graduate studies for statistical research at Cairo University, Egypt. She is working as a Project manager at ministry of finance (IT Department). Her research interests include Big data, record linkage, natural language processing, machine learning, and data integration.

Ali H. El-Bastawissy is recently the Dean of faculty of Computer Science in Modern Science & Art (MSA) University. He has written and published more than 50 articles and studies in Data Modeling and storage, Data Integration, Business Intelligence, and Corporate Information Strategies. He has over 35 years of experience designing and implementing business intelligence and enterprise integration solutions for dozens of Middle East Ministries and Governmental Associations to help them accelerate decision-making and improve corporate performance. Prof. El-Bastawissy supervised more than 35 master studies and about 10 PhD theses during his academic research life.

Eman S. Nasr is a PHD degree holder in computer science. She is currently an independent Scholar. Her research interests include requirements engineering, embedded software systems, crowdsourcing, systems analysis, Web services, computer aided instruction, computer based training, decision making, feature extraction, formal specification, formal verification, genetic algorithms, human computer interaction, artificial intelligence, natural language processing, quality of service, software and engineering.

Mervat H. Gheith is currently an associate professor in the department of computer science in faculty of graduate studies for statistical at Cairo university, Egypt. Her research interests includes crowdsourcing, Turing machines, computer aided instruction, computer based training, feature extraction, formal verification, genetic algorithms, human computer interaction, artificial intelligence, natural language processing, quality of service, emotion recognition, and pattern classification.