Efficient Dynamic Blocking Scheme for Entity Resolution Systems

Aye Chan Mon, and Mie Mie Su Thwin

***Abstract*—** Entity Resolution is the process of identifying and merging duplicate entities determined to represent the same real-world object. It involves identifying cases where multiple database entities correspond to the same real world entity. It is too expensive and time consuming to make pairwise comparisons among all records in large databases. "Blocking" is the process of grouping similar-seeming records into blocks and explores exhaustively. When dealing with very large data sources, it is nearly impossible to determine any fixed set of properties. Often, duplicate records do not share a common key and contain erroneous data that make record matching a difficult task. The quality of a record matching system highly depends on a good blocking approach that is able to accurately detect duplicates in an efficient and effective way. Despite the many techniques that have been introduced over the decades, it is unclear which technique is the current state-of-the-art. In this paper, we propose a novel "Block Based Dynamic Tree Structure" to enhance the blocking efficiency.

***Keywords*—**Entity resolution, data integration, data cleaning, blocking, pre-processing

I. Introduction

ENTITY Resolution is a process to resolve duplicated representations (references) of real objects entities. It has received considerable attention and has been addressed as record linkage, duplicate detection, hardening soft databases, the merge/purge problem and reference matching. The duplication problem arises when different references referring to the same entity exist in the same dataset and are described by similar or different attributes. For example, in personal information management systems, a person may have a tuple in one dataset with name, age , home address and other personal information, and a tuple in one dataset with name , company address, and education background. Given these datasets, the task of entity resolution is to map the references to their underlying real entities and to group them together so that all the references referring to the same entity are put in the same group[2].

A general schematic outline of the record linkage process is given in Figure 1. As most real-world data collections contain noisy, incomplete and incorrectly formatted information, data cleaning and standardisation are important pre-processing steps for successful linkage. Since potentially every record in one dataset has to be compared with every record in a second dataset, blocking or searching techniques are often used to reduce the number of comparisons. The data sets are partitioned into smaller blocks using blocking variables. Only records within the same blocks are then compared in details using the defined comparison variables. The comparison vectors generated by such detailed comparison functions are then passed to the decision model to determine the final status of the record pairs. The results of the record linkage can be assessed by the evaluation model[4].

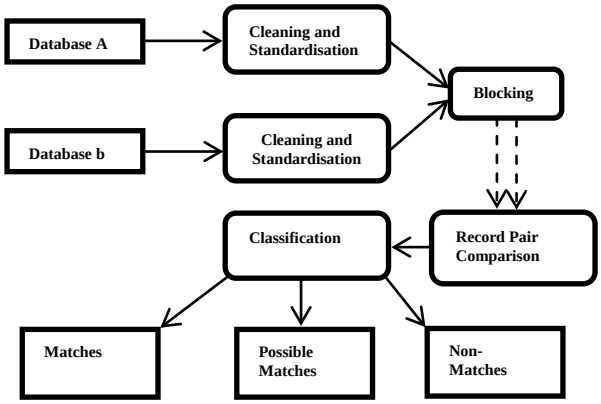

Fig 1. General Record Linkage Process.

Some of the motivating examples of entity resolution areas are Health Care System, e-Government, e-Commence, Crime Detection and National Security, Bibliographic Citations and many other areas. Without Entity Resolution system, hospital care can suffer because records created by different health care providers are not brought together into a single patient history. A hospital has a database with thousands of patient records. Every year it receives new patient data from other sources, such as government or local organizations, It is important for the hospital to link the records in its own database with the data from other sources. However usually the same information can be represented in different formats. They could be typos in the data. The main task here is to link records from different databases in the presence of mismatched information.

Aye Chan Mon is with the Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Mandalay Region, Myanmar. (achanmon@gmail.com)

Mie Mie Su Thwin is with the Ministry of Science and Technology, Nay Pyi Taw, Myanmar.(drmiemiesuthwin@mmcert.org.mm)

The main contribution of this paper is the development of partitioning in dynamic block based dynamic tree structure to reduce the number of record pairs search space with minimal accuracy loss. A key innovation is that partitioning in dynamic block can reduce search space and linking processes to cover all the duplicate records.

The rest of the paper is organized with following sections proposed system using block based dynamic tree structure for addressing limitations of current systems, analysis with previous methods, experimental evaluation and conclusions.

II. Proposed System of Block Based Dynamic Tree Structure For Entity Resolution Systems

Entity Matching is an important and difficult step for integrating data. A key distinction between prior works and our approach is that previously described methods focus on improving blocking efficiency while assuming that an accurate blocking function is known and its parameters have been tuned manually. They focused on fixed size blocking for matching process and it consumes so much time of matching processes.

In **Key Creation Process**, a key for each record in the list is computed by extracting relevant fields or portions of fields. In **Sorting**, sort the records by using the created key in lexicographical order. The system uses a Dynamic Partitioning on blocks before generating the match tasks. In Reference List, we keep record of which blocks maintain which ranges. The system use flexible "**Adaptive Block**" to obtain high performance, to reduce search space and to cover the entire same entities block size. All records are divided into blocks according to Lexicographical Order. Match tasks are done within the blocks which are Partitioning Dynamically.

In **Mixed Block**, incomplete information will be kept in that block. Data inserted in blocks are sorted in order to be efficient for insertion, retrieving and removal of records and each of which is identified by key. It uses dynamic tree structure and Rotation which means same entities are split between child trees then merge to same child tree, will use if needed. The Proposed System is efficient due to less search space and avoiding unnecessary links to other blocks. According to the above Proposed Step, records can be retrieved efficiently. The proposed system is shown in figure 2. The System is a multistep process:

- Data Standardization
- Key Creation
- Sorting
- Dynamic Block Creation
- Dynamic Tree Structure
- Record Matching
- Record Classification

A. Standardization

Most real world data collections contain noisy, incomplete, incorrectly formatted, or even out-of-date data. This standardization process is employed before performing entity resolution in order to increase the probability of finding matches. The main task of data standardization is the conversion of the raw input data into well-defined consistent forms.

In name standardization, all letters are converted into upper case and remove certain characters (like punctuations) as shown in Table I.

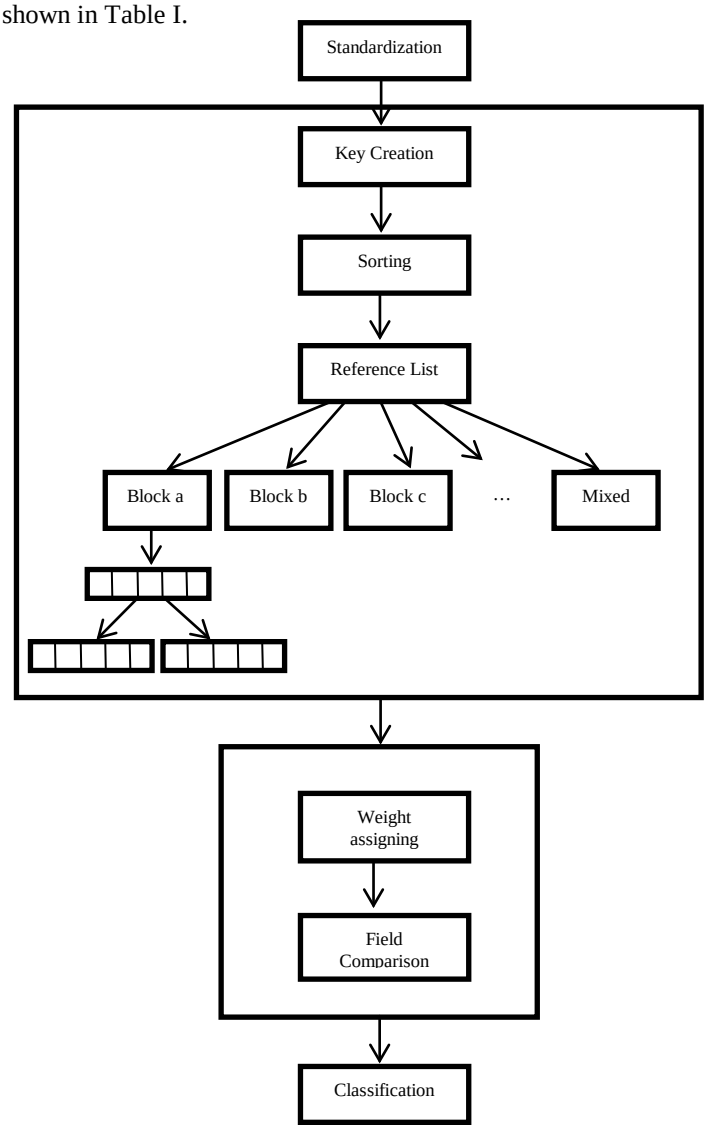

Fig. 2 Proposed System

TABLE I
NAME STANDARDIZATION

	Name	Replacement
1	Bela Williams	bela williams
2	Kennith Showers	kennith showers
3	Shane Garcia	shane garcia
4	Bela Moore	bela moore
5	Bela Willam	bela willam
…	…	…

B. Key Creation

The idea is to choose a high quality key in terms of accuracy, completeness and consistency. A key is defined to be a sequence of a subset of attributes or substrings within the attributes, chosen from the record. The name, date of birth, sex, and all concern data are stored in database as shown in Table I. It is created by the first three consonants of a last name are concatenated with the first letter of the first name field, followed by a gender field and day, month, last three numbers of the year of birth . If the name is only one word then it will take consonants of three words of the name. The basic key creation for this system is shown in Table II.

TABLE II
KEY CREATION EXAMPLE

ID	Name	DOB	Sex		Key
1	bela williams	10.1.1957	m		wllbm101957
2	kennith showers	9.10.1946	m		shwkm910946
3	shane garcia	3.5.1948	m		grcsm35948
4	bela moore	8.2.1989	m	→	mrbm821989
5	bela willam	10.1.1957	m		wllbm101957
. .	…	..	..		…

C. Sorting

The keys are now used for sorting the entire dataset with the intention that all equivalent or matching data will appear close to each other in the final sorted list. The keys are now used for sorting the entire dataset. Sort the records in the data list using the created list. All records are sorted according to the alphabetic manner of created key. Therefore, all equivalent or matching records will appear close to each other. If the data was not sorted, a record may be near the beginning of the array of records and a duplicate record may be near the end of the array of records.

TABLE III
SORTING

ID	Key
1	grcsm35948
2	mrbm821989
3	shwkm910946
4	wllbm101957
5	wllbm101957

D. Reference List

In **Reference List**, we keep record of which blocks maintain which ranges. The Reference List is the central access point of the infrastructure for managing the execution of spilt blocks and maintains the block data. The system use flexible "**Adaptive Block**" to obtain high performance, to reduce search space and to cover the entire same entities block size. All records are divided into blocks according to Lexicographical Order. Match tasks are done within the blocks which are Partitioning Dynamically. Reference List will perform the preprocessing and post-processing of partitioning data as well as matching the final results. It is also responsible for assigning the input records into corresponding blocks and retrieving the data from corresponding block.

E. Mixed Block

In **Mixed Block**, incomplete information will be kept in that block. Information may be omitted because user is unwilling or unable to supply it. Due to missing data values or other data quality issues in real-world data, it may not always be possible to assign entities to a unique block.

F. Dynamic Block Creation for Data Partitioning

Data Partitioning can be of great help in facilitating the efficient and effective management of big data. In this paper, the system first partition the dynamic blocks of the input records and record searching in that dynamic blocks. Data partitioning also results in faster data addressing and efficient data retrieval.

After sorting the all records in the database, it was partitioned the collection of records into set of blocks in the alphabetic manner. Firstly, the created key is search in the reference list to know which blocks it should be kept. If the initial of the created key is the same as the reference list's key then it will save in the corresponding block. Otherwise, the record must be added to the mixed block. This is done until no record remains in the data source. The same records are in the same group as shown in figure 3.

Blocking is used to reduce the number of candidate record comparison pairs to a feasible number whilst still maintaining linkage accuracy. Blocking entails a logical partitioning of the input entities such that all matching entities should be assigned to the same output partition, also called block. By restricting the match comparisons to the entities of the same block the match overhead can drastically be reduced.

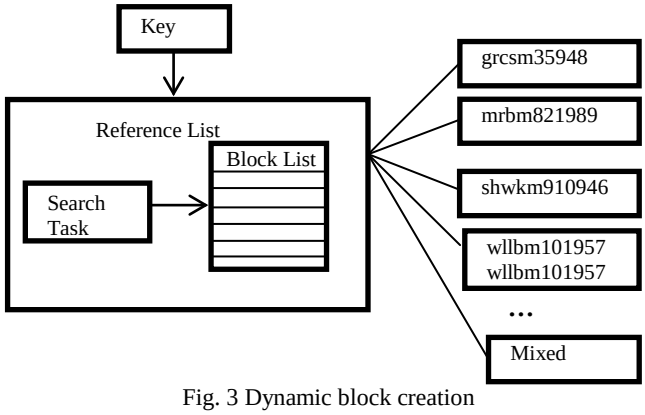

Fig. 3 Dynamic block creation

Algorithm 1: Dynamic Block Creation

```
Input:
-Number of records n
Output:
-Block []

Begin
  n=Number of records
  Initialize Block[]={}
    For i=1..n:
       Create Key
       Sort records on key
       Look in the Reference List to add into the
       corresponding block
        If created key initial key==Reference List initial key
            Then add to the corresponding Block[]
       Else
         Add to the Mixed Block
       Until no records
       Return dynamic block
End
```

G. Record Matching

A good blocking method can greatly reduce the number of record pair comparisons and achieve significant performance speed up. The performance bottleneck in entity resolution system is usually detailed comparison of record pairs. Human expert needs to specify the conditions for *Equational Theory*. Such conditions might involve multiple string similarity metrics and inferred knowledge.

The equational theory does not detect the second pair as duplicates. So, a *distance function* applied to the fields of the records. A string comparator function returns a value depending on the degree of the match of the two strings. Because pairs of strings often exhibit typographical variation, the entity resolution needs effective string comparison functions that deal with typographical variations. The proposed system use weight assigning and field comparison in Field Matching stage. In weight assigning method, it consider a match only if it has a similarity greater than or equal to threshold. In real world, using only weight assigning method cannot gain high accuracy. So, the propose system use field comparison step as an additional step in record matching stage. Example of record matching is shown in figure 4.

H. Record Classification

In the case of user query, the entries in the query fields are compared to an existing database and potential matches will be provided to the user of the system. Firstly, we have to compare selected field comparison. If the result is greater than a fixed threshold, the two values are considered equal. After that we have to compare overall fields' comparison. The number of equal pairs of values in the two records is greater than a threshold, and then the two records are considered as duplicate. Both thresholds are set to fixed value.

According to the above Proposed Step, records can be retrieved efficiently. The output blocks may largely differ in their size depending on the entity value. The Proposed System is efficient due to less search space and avoiding unnecessary links to other blocks. The proposed system will reduce the search space for the entity matching and linkage process between different blocks. The complexity of the preprocessing steps will increase because it needs to check before splitting the entities into different blocks.

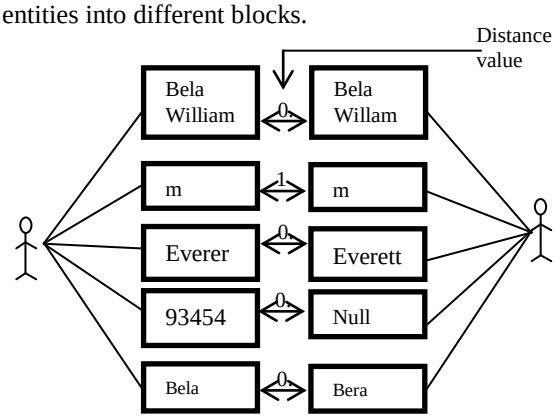

Fig 4. Record Matching process

III. Evaluation Analysis of The System

After In traditional method, assuming two data sets with n records each are to be linked, the blocking key results in *b* blocks, and each block contains n/b records, the resulting number of record pairs is $O(\frac{n^2}{b})$.This is of course the ideal case, hardly ever achievable with real data. In general, the number of record pairs is $O(\sum_{i=1}^{b} n_i^2)$ where n_i is the number of records in block *i*.

In the above proposed system, create key phase is an O (n) operation; the sorting phase is O (n log n). The resulting number of record pair comparisons is O(w n), where n is the number of records in the database, w is the dynamic block size. Record Searching using the Dynamic tree is O(log n) complexity.

IV. Experimental Evaluation

The proposed system can solve different match tasks on heterogeneous Web data entities in comparison to manually tuned strategies with a state-of-the-art match approach. The approach has several parameters to configure. We use the Bibliographic DBLP-ACM, which is one of the standard dataset in the entity resolution community. The most important is the overall similarity threshold. The proposed system considers an entity pairs a match only if it has a similarity greater than or equal to this threshold. For these tests, we used the first or first two attributes listed in table4 (publication titles and authors for the bibliographic tasks). However, finding suitable parameter settings is very challenging, even for

TABLE IV
OVERVIEW OF STANDARD DATASET

	Match Task	Source Size(number of entities)		Mapping Size (number of correspondences)	
Domain	Attributes	Source 1	Source 2	Input mapping (blocking result)	Perfect result
Bibliographic DBLP-ACM	Title Authors Venue Year	2616	2294	494000	2224

domain experts, due to the large number of possible parameter combinations. To find better baseline results than we do using the default parameters, to find the best settings for the similarity thresholds. Appropriate parameters for the similarity thresholds are very important to the accuracy and efficiency of the system. The system set the similarity threshold 0.8 for field comparison. As for entity resolution, the threshold varies depending on different datasets since the ambiguity of these datasets is different. We manually tune the similarity threshold and choose the value that can help to obtain the best accuracy and efficiency.

As many other entity resolution approaches do, we access the accuracy of our entity resolution system with precision, recall and F-measure, in which the F-measure is defined as the harmonic mean of precision and recall. After the implementation of the proposed system, the system outputs total of 1987 records. The number of correct matches made by the proposed system is 1927. The system falsely matches 60 records pair.

TABLE V
ACCURACY OF THE PROPOSED SYSTEM

	Precision	Recall	F-measure
Proposed System	96.98%	86.64%	91.72%

To access the performance, we measure the execution time. For execution time, we use the CPU time for the time spent on a certain process or task. The unit is ms(milliseconds) in CPU time . The processing time for the proposed system is approximately 59 ms.

The evaluation has shown that the proposed system proved to be very effective in that we could achieve very high match accuracy and fast execution time.

V. CONCLUSION

Entity Resolution has been recognized as a key process in support of data cleaning for removing duplicate records and in data integration as a way to aggregate information for the same entity across different information sources. However, the goal of most ER systems has been narrowly defined as simply executing a set of matching rules on a given set of records linking together those records determined to be equivalent. The propose system is independent on the applied area chosen.

The proposed system enables matching entities assigned to the same block. It uses Dynamic Blocking to obtain high performance, to reduce search space and to cover the entire same entity within block step. In the proposed system, input data is split according to the blocking variables. As no comparisons are conducted between different blocks, each block can be processed independently form all others. Blocks can contain different numbers of records which results in varying processing times. We propose an effective blocking for combining multiple entity resolution systems. In this paper, we have proposed "Dynamic Blocking" to obtain high performance, to reduce search space and to cover the entire same entity within block. It also applies "Dynamic Tree" as an additional steps to get efficient searching. Future directions of this work will include applying the proposed system in ecommerce application and cloud based structure as an additional step to do efficient record searching in entity resolution systems.

REFERENCES

[1] S. A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios, "Duplicate Record Detection: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1,pp. 1-26,2007.
http://dx.doi.org/10.1109/TKDE.2007.250581

[2] Bilenko, M., and Mooney, R. J. (2003a), "Adaptive Duplicate Detection Using Learnable String Similarity Metrics," Proceedings of ACM Conference on Knowledge Discovery and Data Mining, Washington, DC, August 2003, 39-48.

[3] D.G. Brizan, A.U. Tansel, "A Survey of Entity Resolution and Record Linkage Methodologies", Communications of the IIMA, Issues 3, Volume 6,2006.

[4] H.B. Newcombe, J.M. Kennedy, S.J. Axford, A.P. James, "Automatic linkage of vital records", Science 130 (1959) 954–959.
http://dx.doi.org/10.1126/science.130.3381.954

[5] I.P. Fellegi, A.B. Sunter, A theory for record linkage, J. Am. Stat. Assoc. 64 (328) (1969) 1183–1210.
http://dx.doi.org/10.1080/01621459.1969.10501049

Aye Chan Mon was born in Taunggyi,Myanmar. She received her Bachalor Degree in Computer Science (B.C.Sc) in 2006 from Computer University(Taunggyi),Myanmar. She got her Master Degree in Computer Science (M.C.Sc) in 2010 from Computer University (Pin Lon),Myanmar. She is now attending Ph.D(IT) at University of Technology(Yatanarpon Cyber City),Myanmar. Her research interests are entity resolution and data mining.