Effective Blocking for Combining Multiple Entity Resolution Systems

Aye Chan Mon
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
Ministry of Science and Technology
Nay Pyi Taw, Myanmar.
drmiemiesuthwin@mmcert.org.mm

Abstract—An important aspect of maintaining information quality in data repositories is determining which sets of records refer to the same real world entity. This so called entity resolution problem comes up frequently for data cleaning and integration. Entity Resolution (ER) is a problem that arises in many information integration applications. ER process identifies duplicated records that refer to the same real-world entity, and derives composite information about the entity. The cost of the ER process is high. In this propose paper, 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.

Keywords- entity resolution, data integration, data reduction, indexing, pre-processing

I. INTRODUCTION

Information integration is one of the most important and challenging computer science problems: Information from diverse sources must be combined, so that users can access and manipulate the information in a unified way. One of the central problems in information integration is *Entity Resolution (ER)* (sometimes referred to as deduplication). ER is the process of identifying and merging records judged to represent the same real world entity. ER is a well-known problem that arises in many applications. For example, mailing lists may contain multiple entries representing the same physical address, but each record may be slightly different, e.g., containing different spellings or missing some information. As a second example, a comparative shopping website may aggregate product catalogs from multiple merchants. A sample ER process is shown in figure 1.

Entity Resolution (ER) is a problem that arises in many information integration applications. Identifying records that *match* (e.g., records that represent the same product) is challenging because there are no unique identifiers across sources. A given record may appear in different ways in each source, and there is a fair amount of guesswork in determining which records match. Deciding if records match is often computationally expensive, e.g., may involve finding maximal common subsequences in two strings. ER uses two functions, *match* and *merge*. The match function identifies duplicated records that refer to the same real-world entity, e.g. the same person, the same product, or the same organization. The merge function derives composite information about the matched entities. For example, customer databases from different business divisions may contain multiple records representing the same person, but each record may be slightly different, e.g., with different spellings of the name or address, or missing some information [2].

With many businesses, government agencies and research projects collecting massive amounts of data, techniques that allow efficient processing, analyzing and mining of large databases have in recent years attracted interest from both academia and industry. An increasingly important task in the data preparation phase of many data mining projects is linking or matching records relating to the same entity from several databases, as often information from multiple sources needs to be integrated and combined in order to enrich data and allow more detailed data mining studies. The aim of such linkages is to match and aggregate all records relating to the same entity, such as a patient, a customer, a business, a consumer product, a bibliographic citation, or a genome sequence.

The problem of finding similar entities not only applies to records that refer to persons. In bioinformatics, record linkage can help find genome sequences in large data collections that are similar to a new, unknown sequence at hand. Increasingly important is the removal of duplicates in the results returned by Web search engines and automatic text indexing systems, where copies of documents (such as bibliographic citations) have to be identified and filtered out before being presented to the user. Finding and comparing consumer products from several online stores is another application of growing interest. As product descriptions are often slightly different, linking them becomes difficult.

If unique entity identifiers (or keys) are available in all the databases to be linked, then the problem of linking at the entity level becomes trivial: a simple database join is all that is required. However, in most cases no unique keys are shared by all databases, and more sophisticated linkage techniques need to be applied. These techniques can be broadly classified into deterministic, probabilistic, and modern, machine learning based approaches [13].

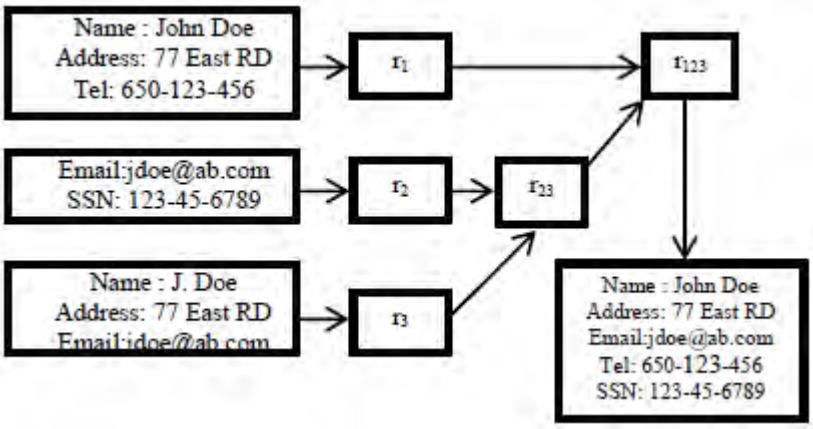

Figure 1: A Sample ER process

The main contribution of this paper is the development of partitioning in dynamic block 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 as follows. Section II is about literature review. Section III is about Background Theory and Section IV then presents our proposed system using dynamic block partitioning for addressing limitations of current systems. Section V is the analysis with previous methods. Conclusions are made in Section VI.

II. LITERATURE REVIEW

Entity Resolution is a very active research topic. Entity Resolution is the process of determining whether two references to real world objects are referring to the same or to different, objects. Entity means real world objects and Resolution means entity poses a question.

In order to reduce the search space (i.e. the number of record pairs to be compared), many blocking methods have been recently proposed in the literature and some of them are reviewed in this section. The main idea in blocking is to group similar records together called blocks using available information form the records.

The entity matching problem was originally defined in 1959 by Newcombe et al. [4] and was formalized by Felligi and Sunter [5] 10 years later. Since then it has been considered under various facets and from different communities, including the AI research community, the DB research community, and industry. Numerous approaches have been proposed for entity matching especially for structured data. Many Entity Resolution approaches have been proposed and evaluated as described in recent survey [3, 1].

D.g. Brizan and A.U.Tansel,[3] surveyed the literature for the methodologies proposed or developed for entity resolution and record linkage in 2006. It provides a foundation for solving many problems in data warehousing. They also provide a base for further research in solving entity resolution and record linkage problems. Little or no research has been directed at the problem of maintenance of clean and linked relations and provides a base for further research in solving entity resolution and Record Linkage.

M.A. Hern' andez and S.J.Stolfo [8] implemented that performs a generic Merge/Purge process that includes a declarative rule language for specifying an equational theory making it easier to experiment and modify the criteria for equivalence. P.Christen [12] surveyed the twelve variations of six indexing techniques in 2011. Their complexities are analyzed, and their performance and scalability is evaluated within an experimental framework using both synthetic and real data sets. These experiments highlight that one of the most important factors for efficient and accurate indexing for record linkage and deduplication is the proper definition of blocking keys. First ideas for parallel matching were described in [11]. P.Christen [11] showed how the record linkage processes can be partition and parallelized.

M.Mchelson and S.A.Macskassy [9] proposed a measurement technique for Entity Resolution. The authors propose to focus on measuring the entities themselves rather than the matching records. T.Kirsten and L.Kolb, et al. [14] proposed a generic model for parallel processing of complex match strategies that may contain several matchers in 2010. The parallel processing is based on general partitioning strategies that take memory and load balancing requirements into account. Compared to this work [14] allows the execution of a match workflow on the Cartesian product of input entities. This is done by partitioning the set of input entities and generating match

tasks for each pair of partitions. The disadvantage is that only the matching itself is executed in parallel. Blocking is done upfront on the process.

L.Kolb, A.Thor and E.Rahm [6] did not investigate Sorted Neighborhood blocking but show how Sorted Neighborhood can combine with MapReduce. The approach is based on a complex workflow consisting of several MapReduce jobs. The disadvantage is that we have to give the number of Map Tasks and Reduce Tasks. In real world, predefined values for number of map task and reduce task is not convenient. L.Klob, A.Thor and E.Rahm [7] proposed that how Block based Techniques can be combine with Map Reduce tasks. The approach is based on blocking techniques and MapReduce jobs. The disadvantage is that we have to do redundant MapReduce task because all the duplicates records are not cover with one step.

In contrast to our proposed system, blocks are partitioned into dynamic size so that linking blocks between matching process can be reduced. We can reduce the search space for entity matching and can apply a complex match strategy for each pair of entities within a block. In previous works, they only mainly focus on build step or retrieve step. In this paper, we will mainly focus in the build step in creation of dynamic blocks and retrieving step from created dynamic block.

III. BACKGROUND THEORY

Before *Blocking* methods efficiently select a subset of record pairs for subsequent similarity computation, ignoring the remaining pairs as highly dissimilar and therefore irrelevant. Blocking techniques typically form groups (blocks) observations using indexing or sorting. This allows efficient selection of instance pairs from each block for subsequent similarity computations. Some blocking methods are based on the assumption that there exists a pre-defined similarity metric that correctly captures similar pairs for the domain and task at hand, while others assume that forming blocks based on lexicographic sorting of records on some field(s) yields an optimal strategy.

When linking large databases, blocking is essential in order to make record linkage possible at all, and to improve the efficiency of the linkage process. Blocking, however, will reduce the linkage quality, as it is very likely that some true matched record pairs will be removed by the blocking process, if records are not being inserted into the correct block (or blocks) due to variations and errors in their blocking key values. The blocking step can be split into the following two sub-steps:

a) **Build**: All records in the databases are read, the blocking key values are created, and the records are inserted into a suitable index data structure. For most blocking techniques, an *inverted index* can be used. The blocking key values will become the keys of the inverted index, and the record identifiers of all records that have the same blocking key value will be inserted into the same inverted index list. The record attribute values required in the comparison step will be inserted into another data structure, for example a hash-table with record identifiers as keys (which can be main memory or disk based).

b) **Retrieve**: Record identifiers are retrieved from the index data structure block by block, and the corresponding record attribute values required for the comparisons are retrieved. Candidate record pairs are then generated from these records, by pairing all records in a block from one database with all records in the same block from the other database. The generated candidate record pairs are then compared and the resulting vectors containing numerical similarity values are given to a classifier [12].

A. Standard Blocking

The *Standard Blocking (SB)* method partitions records into mutually exclusive blocks where records in each block share an identical blocking key value. A blocking key is a variable composed from the record attributes in a data set. There is a cost-benefit trade-off to be considered in choosing the blocking keys. A further consideration in the selection of blocking keys is the error characteristics of the record attributes used. To achieve good linkage accuracy, it is preferable to use the least error-prone attributes available as the blocking key. Multiple blocking keys are also used as a way to mitigate the effects of errors in blocking keys. Another strategy of reducing the effects of spelling and transcription errors in record attributes used as blocking keys (typically name and address attributes) is to use phonetic encodings, such as Soundex or NYSIIS, of these attributes.

IV. The number of generated record pairs depends on the number of blocks and their sizes. 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*. Thus, the number of record pairs to be compared can be dominated by large blocks [12].

B. Sorted Neighbourhood

The Sorted Neighbourhood (SN) method sorts the records based on a sorting key and then moves a window of fixed size *w* sequentially over the sorted records. Records within the window are then paired with each other and included in the candidate record pair list. The use of the window limits the number of possible record pair

comparisons for each record to 2w − 1. The resulting number of record pair comparisons (assuming two data sets with *n* records each) of the SN method is O(*wn*).

One problem with the SN method arises if the number of records in a block is larger than the window size. The problem of SN can be avoided by using the transitivity property. Two approaches of this techniques can be implemented are Sorted Array Based Approach and Invented Index Based Approach. Another recently developed approach was Adaptive Sorted Neighborhood Method [13].

C. Bigram Indexing

The Bigram Indexing (BI) Method allows for fuzzy blocking. The basic idea is to convert the blocking key values into a list of bigrams (sub-strings containing two characters) and build sublists of all possible permutations using a threshold. The resulting bigram lists are sorted and inserted into an inverted index, which will be used to retrieve the corresponding record numbers in a block.

Like standard blocking, the number of record pair comparisons with two data sets with *n* records each and *b* blocks all containing the same number of records is $O(\frac{n^2}{b})$. However, the number of blocks b will be much larger in bigram indexing [12].

D. Canopy Clustering

Canopy Clustering is a two-step efficient clustering method for high dimensional data sets. The key idea involves using a cheap, approximate distance measure to efficiently divide the data into overlapping subsets (called canopies). Clustering is then performed by measuring exact distances only between points that occur in a common canopy. In the case of blocking, only the first step is performed to form canopies using TF/IDF (Term Frequency/Inverse Document Frequency) as the approximate distance measure. Only records in the same canopy will be compared in detail. The number of record pair comparisons resulting from canopy clustering is $O(\frac{fn^2}{c})$ where c is the number of canopies and f is the average number of canopies a record belongs to. The parameters should be set so that f is small and c is large, in order to reduce the amount of computation [13].

IV . PROPOSED SYSTEM OF EFFECTIVE BLOCKING FOR COMBINING MULTIPLE ENTITY RESOLUTION SYSTEMS

Entity Matching is an important and difficult step for integrating data. To reduce the typically large space for doing entity matching is time consuming. Blocking entails a logical partitioning of the input entities such that all matching entities should be assigned to the same output partition called block. By restricting the match comparisons to the entities of the same block the match overhead can often drastically be reduced.

A general schematic outline of the record linkage process is given in Figure 2. 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].

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 contrast, our approach attempts to construct an optimal blocking function automatically and adaptive blocking to speed up matching process and searching process. Because blocking functions can be learned using any combination of similarity predicates on different record fields, and no assumptions are made about the number of record fields or their type. In this paper we focus on the blocking/searching component in the information flow diagram.

In this paper, we propose a dynamic block based searching method which consists of following strategies:

- Standardization
- Key Creation
- Sorting
- Dynamic Block Creation for Data Partitioning
- Record Comparison
- Record Classification

The data sources are from multiple databases and we need to combine same entities into one record. The proposed system is shown in figure 3.

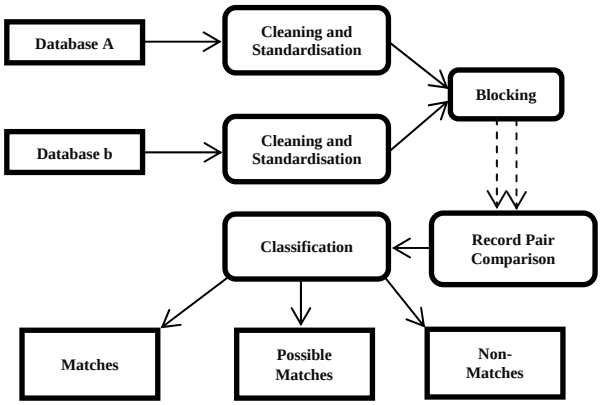

Figure 2. General Record Linkage Process.

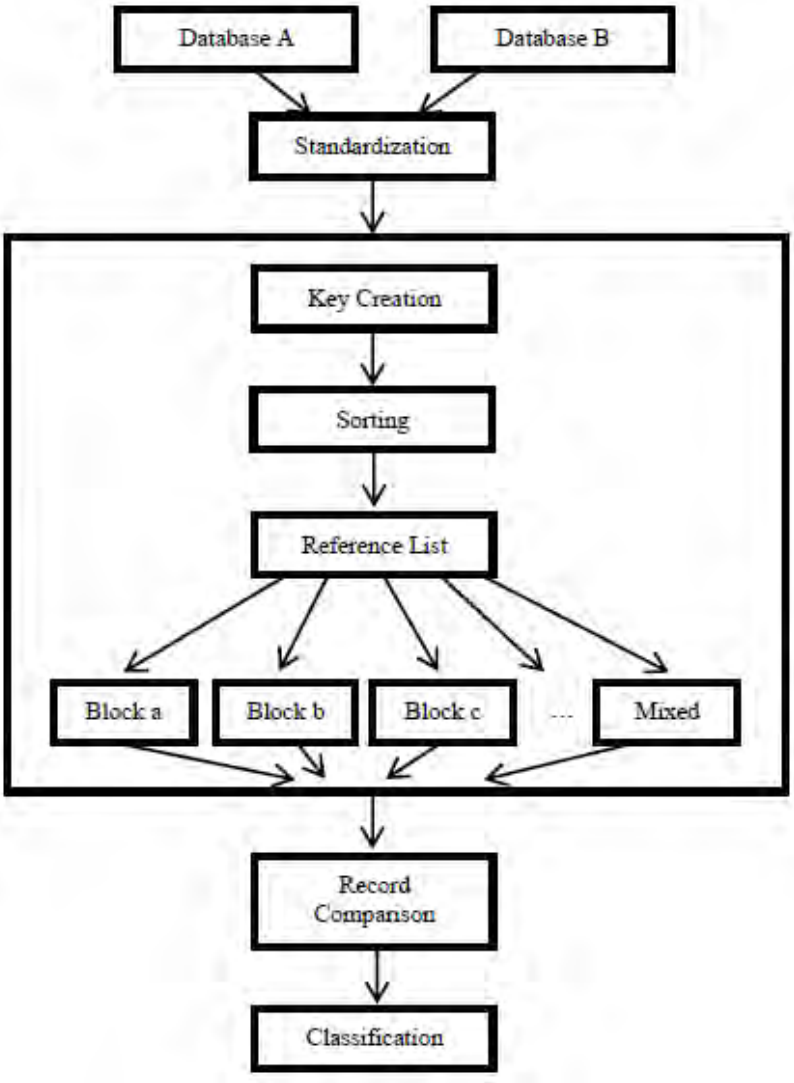

Figure 3. Proposed System

A. Standardization

Data standardization is important preprocessing steps for successful entity resolution and before such data can be loaded into data warehouses or used for further analysis. The first step in parsing a component consists in cleaning the input string by converting all letters into lowercase, by removing unwanted characters and by replacing certain characters by others. For example, all forms of brackets are replaced by a vertical bar. The standardisation of each record in the input data sets can be done independently of all records.

This standardization process is employed before performing record linkage in order to increase the probability of finding matches. Without standardization, many true matches could be wrongly designated as non-matches because the common identifying attributes do not have sufficient similarity.

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 2.

In gender attribute, it is replaced by one of the standard format such as Male is replaced by 'm' and Female is replaced by 'f' as shown in Table 3.

For date of birth it is converted into standardize format (day-month-year) as shown in Table 4.

After standardization the records from various database, it is easy to search for the duplicate records.

B. Key Creation

Data The idea is to choose a high quality key in terms of accuracy, completeness and consistency. The choice of a key with a low completeness value; after a sorting on the basis of such a key, the potential matching records can be not close, due to null values. The choice of the key is a fundamental step of the matching process, as the results of the matching are directly influenced by it. There is also a cost trade-off to be considered in choosing the blocking keys. A further consideration in the selection of blocking keys is the error characteristics of the record attributes used. To achieve good linkage accuracy, it is preferable to use the least error-prone attributes available as the blocking key.

A key is defined to be a sequence of a subset of attributes or substrings within the attributes, chosen from the record. Keys must provide sufficient discriminating power. The name, date of birth, sex, and all concern data are stored in database as shown in Table 1. 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 5.

TABLE I. RECORD STRUCTURE OF CLIENT DATABASE FOR BOOKSTROE

ID	1		4	5
Name	Bela Williams		Bela Moore	Bela William
Mid Name	M		M	
SSN	934545761		981234935	915211198
DOB	10/01/57		08/2/89	10/1/57
Gender	Male		Male	Male
Race	Caucasian		African	Caucasian
NickName	Bela		Bela	Bela
City	Evererr		Sedrowool	Everett
Date_txt3	10/01/1957	…	05/12/1969	11/11/1994
Date_txt4	1957/10/01		1969/05/12	1994/11/11
…	…		…	…

TABLE II. RECORD NAME ATTRIBUTE STANDARDIZATION

ID	**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
…	…	…

TABLE III. GENDER ATTRIBUTE STANDARDIZATION

Original	**Replacement**
Male	m
Female	f

TABLE IV. DATE ATTRIBUTE STANDARDIZATION

Original	**Repalcement**
10/01/57	10.1.1957
01.01.57	1.1.1957

TABLE V. 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
..	…	..	…		…

TABLE VI. KEY SORTING

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

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.

```
Input  : Created Key
Output : Corresponding Block

Begin
Search the Key initial in Block List
If Key is found in Block List
        Then go to the corresponding block
Else
        Go to the Mixed block
End
```

Algorithm 1. Reference List Process

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. We therefore assign such entities to a dedicated *mixed* block. Entities of this block have to be matched against the entities of all blocks.

F. Dynamic Block Creation for Data Partitioning

Data Partitioning can be of great help in facilitating the efficient and effective management of big data. But data partitioning could be a complex process which has several factors that can affect partitioning strategies and design, implementation, and management considerations in an entity resolution.

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 4.

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. Algorithm 1 describes the creation of the dynamic block.

```
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
```

Algorithm 2. Dynamic Block Creation

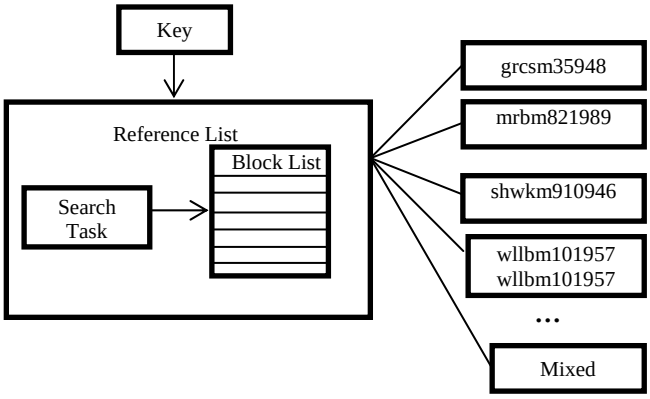

Figure 4. Dynamic block creation

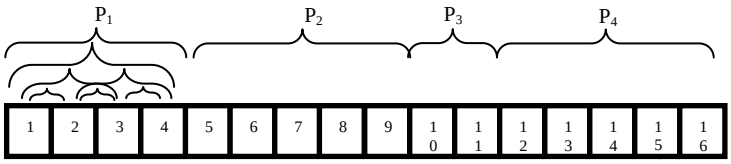

Figure 5. Dynamic Record search in previous methods

G. Record Searching in Dynamic Block

Several techniques have been developed in the past searching the space of record pairs. In previous proposed method, if user searches for a record than it need to search all the entire dataset and consume so much time and cost. When searching a record with two datasets which have (2600 datasets and 64000 objects) that took up approximately 75h for single serial searching. The execution times increase even more if matching on multiple attributes is applied. It needs to search all the data one by one and need to compare all the data to the end even it found the desire record in the middle of the datasets.

The previous methods use fixed size sliding window or blocking method. When fixed size sliding window is applied, and records within a window are checked to determine if they are duplicates using a merging rule for determining similarity between records. The window size is fixed and there may be same records with different window sizes. They have to do redundant comparison between each record and consume time. A disadvantage of previous approach is that the window size is fixed and difficult to configure. If it is selected too small, some duplicates might be missed. On the other hand, a too large window results in many unnecessary comparisons. The

overlap between the partitions results in several windows which have to be checked for duplicates. As shown in figure 3, there are 4 partitions and all the records in the partition in P_1 needs to compare each other's.

In our proposed system, the processing time of record searching can save time consuming and cost. Because the system will search only in the appropriate blocks and can give back results to the user as soon as possible. When a user wants to search a record, the system creates a key for this record. Created key initial is compared in Reference List and search only in destination block. One of the advantages of the partitioning the dynamic blocks is that it can search data as fast as possible. Data are saved in the dynamic block and the data are positioned in efficient way. The proposed method process explores the space of record pairs in order to identify records that are prospective matches. The solution of the problem of previous methods is to identity only those record pairs which have high probability of matching, leaving uninspected those pairs that look very different.

```
Input
-User Search Query
Output
-Complete Information about User or Partial  Information about User

Begin
        Accept User Search Data
        Create Key
Search in Reference List the same initial as Search Key
        Search in the Corresponding Block
        If user created key== Existing key
          Then exit searching && display "result to   the user"
        Else if  Compare three letters of both keys
            Then compare two records
        Else  display "no such record"
End
```

Algorithm 3. Record searching in Dynamic Block

H. 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 record linkage system is usually detailed comparison of record pairs.

The basic idea of string comparison is to be able to compare pairs of strings such as 'Mon, Min' that contain minor typographical error. String Comparison in record linkage can be difficult because lexicographically "near by" record look like "matches" when they are in fact not. Possible errors range from small typographical slips to complete name and address changes.

Human expert needs to specify the conditions for *Equational Theory*. Such conditions might involve multiple string similarity metrics and inferred knowledge. An example of Equational Theory is shown below:

Given two records, r1 and r2.

IF the last name of r1 equals the last name of r2,

AND the first names differ slightly,

AND the address of r1 equals the address of r2

THEN

r1 is equivalent to r2.

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 record linkage needs effective string comparison functions that deal with typographical variations.

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.

Each selected field will have a quality of match value. The individual field quality of matches will be combined into a composite score for each record. This value will be used to determine the overall probability of record matching. There are several methods available for determination of record matching from the quality of match of the individual fields. The individual field quality of matches will be combined into a composite score for each record. This value will be used to determine the overall probability of record matching. 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. Example of record matching is shown in figure 6.

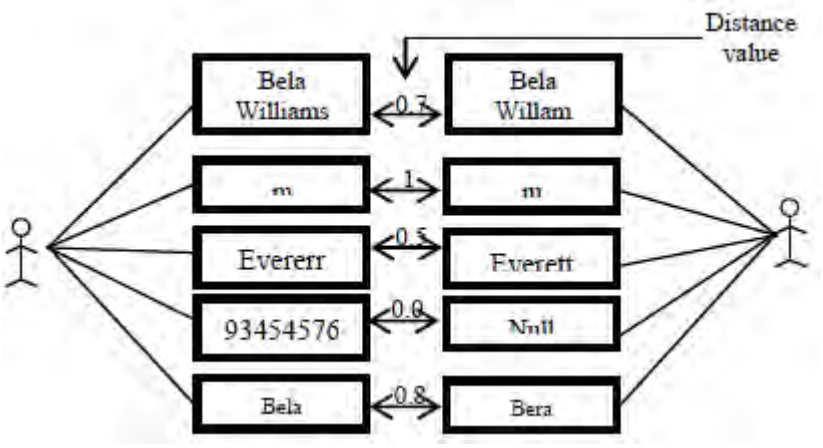

Figure 6. Record Matching process

I. 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. The list will be limited by a threshold value and the user will have the ultimate decision regarding a match. In the case of a new record entry, the record will be compared to the database for near identical matches. The threshold for allowing a match may be set high to prevent incorrectly matching two different records.

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.

V. EVALUATION 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. But one thing to consider is that the complexity may be a little high when dividing blocks.

VI. CONCLUSION

Many organizations rely on a variety of data in their daily operations and needs to integrate the systems to make better use of the information scattered in different systems. Entity matching is a critical, yet difficult and time-consuming, problem in integrating heterogeneous data sources and demands automated support. In this paper, we have proposed the partition the block into dynamic block to reduce search space and reduce linkage process when comparing records pairs. Dynamic Blocking is needed to efficiently eliminate linkage process between different blocks. Our partitioning strategy 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. Future directions of this work include tree like 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.

[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.

[5] I.P. Fellegi, A.B. Sunter, A theory for record linkage, J. Am. Stat. Assoc. 64 (328) (1969) 1183–1210.

[6] L.Kolb, A.Thor, E.Rahm, "Parallel Sorted Neighborhood Blocking with Map Reduce", In:BTW, pp.45-64 (2011).

[7] L.Kolb, A.Thor, E.Rahm, "Block-based Load Balancing for Entity Resolution with Map Reduce", CIKM'11, October 24-28,2011, Glasgow, Scotland, UK.

[8] M. A. Hern´andez and S. J. Stolfo, "The merge/purge problem for large database", In Proceedings of SIGMOD-95.

[9] M. Mchelson, S.A. Macskassy, "Record Linkage Measures in an Entity Centric World"

[10] P.Christen, "Towards Parameter-free Blocking for Scalable Record Linkae", TR-CS-07-03, Technical Reports

[11] P. Christen, "Parallel Computing Techniques for High-Performance Probabilistic Record Linkage" Department of Computer Science, The Australian National University, ACT 0200.

[12] P.Christen, "A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication," IEEE Transactions on Knowledge and Data Engineering, vol. Z, no. Y, ZZZZ 2011.

[13] R. Baxter, P. Christen, and T. Churches, "A comparison of fast blocking methods for record linkage", In Proceedings of KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation.

[14] T. Kirsten, L. Kolb, M. Hartung, A. Grob, H. Kopcke, E. Rahm, "Data Partitioning for Distributed Entity Matching", in Preceedings of the VLDB Endowment, Vol.3, No.2 , 2010.