Record linkage approaches in big data: A state of art study

Conference Paper · December 2017

DOI: 10.1109/ICENCO.2017.8289792

CITATIONS
2

READS
647

4 authors:

Randa M Abd El-Ghafar
Institute of Statistical Studies and Research
5 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Mervat H. Gheith
58 PUBLICATIONS **307** CITATIONS
SEE PROFILE

Ali El-Bastawissy
Modern Sciences and Arts University
66 PUBLICATIONS **254** CITATIONS
SEE PROFILE

Eman Nasr
36 PUBLICATIONS **135** CITATIONS
SEE PROFILE

Record Linkage Approaches in Big Data: A State Of Art Study

Randa M. Abd El-Ghafar, Mervat H.Gheith
Computer Science Department
Institute of Statistical Studies and Research, Cairo University, Cairo, Egypt
randa_mohamed_cs@yahoo.com,
mervat_gheith@yahoo.com

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

Eman S. Nasr
Independent Researcher
Cairo, Egypt
nasr.eman.s@gmail.com

***Abstract*—Record Linkage aims to find records in a dataset that represent the same real-world entity across many different data sources. It is a crucial task for data quality. With the evolution of Big Data, new difficulties appeared to deal mainly with the 5Vs of Big Data properties; i.e. Volume, Variety, Velocity, Value, and Veracity. Therefore Record Linkage in Big Data is more challenging. This paper investigates ways to apply Record Linkage algorithms that handle the Volume property of Big Data. Our investigation revealed four major issues. First, the techniques used to resolve the Volume property of Big Data mainly depend on partitioning the data into a number of blocks. The processing of those blocks is parallelly distributed among many executers. Second, MapReduce is the most famous programming model that is designed for parallel processing of Big Data. Third, a blocking key is usually used for partitioning the big dataset into smaller blocks; it is often created by the concatenation of the prefixes of chosen attributes. Partitioning using a blocking key may lead to unbalancing blocks, which is known as data skew, where data is not evenly distributed among blocks. An uneven distribution of data degrades the performance of the overall execution of the MapReduce model. Fourth, to the best of our knowledge, a small number of studies has been done so far to balance the load between data blocks in a MapReduce framework. Hence more work should be dedicated to balancing the load between the distributed blocks.**

Keywords*—*Big Data; Big Data Integration; blocking; entity matching; entity resolution; Hadoop; machine learning; MapReduce; Record Linkage.

I. INTRODUCTION

The term Big in Big Data does not only refer to the size of the data, but it also indicates many characteristics known as the five dimensions of Big Data, or 5Vs of Big Data, as shown in Fig. 1. Those 5Vs are; Volume which refers to "Big" in the term Big Data; Variety which refers to Data that come from a variety of sources and has many formats (structured as traditional databases, semi-structured as XML files or non-structured as Images); Velocity which refers to how quickly the data could arrive, be stored, and retrieved; Veracity which represents the reliability of data; and Value which is the high value of data that could be obtained by analyzing a huge amount of data. Actually, the majority of the data (about 80%) is unstructured data and this explains why a business is mainly concerned with managing unstructured data [1].

Data Integration is the process of combining data from many different sources and providing a user with one unified view of these data. Big Data Integration differs from traditional Data Integration in the three basic dimensions of Big Data known as 3Vs (Volume, Variety, Velocity), as they make the process of Big Data Integration more challenging and complicated. For the Volume, the number of data sources has grown to be millions for a single domain [2]. For the Velocity, many of the data sources are very dynamic and rapidly exploding as a direct result of the rate at which data is moving and being collected from heterogeneous data sources [2]. For Variety, Big Data projects include data sources from different domains, which are naturally diverse as they refer to different types of entities and relationships; these different types of sources are in need to be integrated into a unified view of data [2].

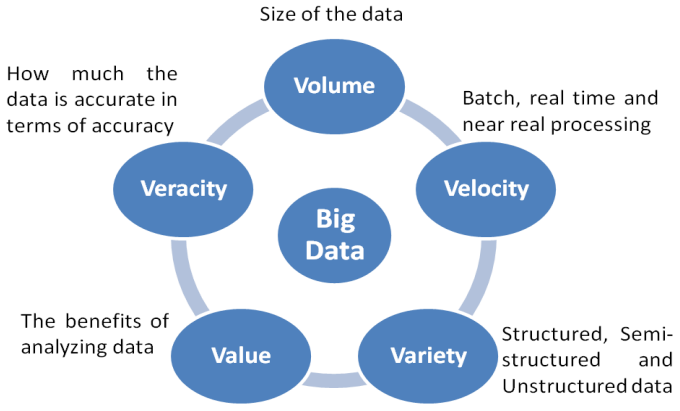

Fig. 1. Five dimensions of Big Data.

The architecture of Data Integration is composed of three steps, Schema alignment, Record Linkage and data fusion. The three steps aim to address the challenges of integrating data from multiple sources as semantic ambiguity, instance representation ambiguity, and data inconsistency [2] as shown in Fig. 2.

Schema alignment addresses the semantic ambiguity challenges and aims to recognize which attributes have the same meaning and which does not [2]. Record Linkage addresses the challenges of the ambiguity of instance representation and aims to recognize which elements represent the same real-world entity and which are not [2]. Data fusion addresses the challenges of data quality and aims to know which values to use in case of conflict attributes value arises from many sources [2].

This paper presents a survey of Record Linkage techniques found in the literature to handle the increasing volume of data. Adding the volume dimension to the traditional Record Linkage makes the Record Linkage more complicating and challenging as it compares a pair of records using one or more similarity techniques which make Record Linkage more expensive task that could take days to be accomplished

The rest of this paper is organized as follows. Section II presents the state of art of Big Data Integration techniques. Section III explains blocking techniques and MapReduce programming model. Finally, the conclusion and future work are given in section IV.

II. THE STATE OF ART OF BIG DATA INTEGRATION TECHNIQUES

The importance of Big Data Integration resulted in increasing amounts of researches in the fields of schema mapping, Record Linkage and data fusion of Big Data Integration over the past few years to deal with the challenges associated with them. Table I shows a summary of these techniques [3]. This section addresses Record Linkage techniques that handle the Volume property of Big Data.

A. Record Linkage Definition and Importance

Preparing and cleaning any dataset for analysis is very important step because of the concept of garbage in garbage out. It is a costly process, error pruning and time-consuming. Actually, it takes about 80% of the whole time spent on Preparing and cleaning any dataset for analysis is very important step because of the concept of garbage in garbage out. It is a costly process, error pruning and time-consuming. Actually, it takes about 80% of the whole time spent on analytics. According to Gartner, $44 billion total in 2014 alone was invested for successfully preparing data obtained from many sources for use in data analysis [4].

Record Linkage, Entity Resolution or Entity Matching is a crucial step in data cleaning which aims to find records (i.e., database tuples) that refer to the same real-world entity [5], [6]. Record Linkage is an expensive process that could take many hours or even days. The situation become more complicated especially for large datasets as it compares a pair of records using one or more similarity measures [7].

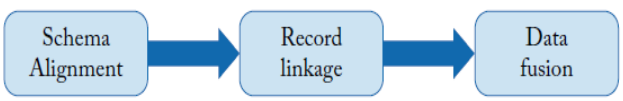

Fig. 2. Traditional Data Integration architecture [2].

B. Challenges Involved in Record Linkage of Big Data

Traditional Record Linkage depends on measuring the similarity between a static set of structured tuples that have the same schema. Record Linkage is not a simple and trivial task due to many reasons. First, typographical errors which prevents records from being associated with the same individual. Second, choosing the fields that will be used in the detecting similarity process between records. Thirds, determining the threshold that will decide which records are considered to be duplicate and which are not (i.e. If the similarity between records is 80% or above they will consider duplicate). In Big Data Integration the situation is more complicated and challenging because data has very heterogeneous structure and coming from many sources (structured, unstructured, and semi-structured), and data sources are continuously evolving and dynamic. This makes the Record Linkage more challenging and non-trivial in Big Data Integration [3].

C. Record Linkage Techniques in Big Data

Traditional Record Linkage approaches become inefficient and ineffective when we are examining large datasets. New techniques have been proposed to address the challenges of volume dimension by using MapReduce to efficiently and effectively parallelize the process of Record Linkage. Parallelize Record Linkage depends on effectively distribute the workload between many nodes exploiting the techniques of adaptive blocking [2].

Applying Record Linkage with each update of large datasets from scratch is unaffordable especially when the datasets are dynamic and continuously evolving. Incremental Record Linkage has been proposed to address the challenges of velocity aspects. It allows efficient incremental Record Linkage in cases of any updates, inserts or deletes of any record in the dataset [2]. To address the variety challenge of Big Data Integration, new techniques have been proposed to link or match a structured text with unstructured or free text. Matching structured data with unstructured text could happen in many cases as trying to match unstructured offers to structured products description of people with tweets or blog posts about their shopping experience [8]. Finally, to address the veracity aspects, a variety of clustering and linking techniques have been proposed.

TABLE I. SUMMARY OF STATE OF BIG DATA INTEGRATION TECHNIQUES [3]

Big Data Property	Schema mapping	Record linkage	Data fusion
Volume	Integrating Deep Web, Web tables/lists	Adaptive blocking, MapReduce-based linkage	Online fusion
Velocity		Incremental linkage	Fusion in a dynamic world
Variety	Dataspace systems	Linking text to structured data	Combining fusion with linkage
Veracity		Value-variety tolerant linkage	Truth discovery

Those techniques focus on out of date values and how to deal with erroneous values effectively. Dealing with erroneous and out of date values is very important because it may prevent correct Record Linkage [2].

The techniques that handle the increasing volume of data depend on parallel processing where the input data is partitioned into a number of blocks to distribute the workload between them. Then, these blocks will be processed by many numbers of reduce tasks in the MapReduce programming model described below.

III. Blocking Techniques and MapReduce Programming Model.

This paragraph discusses the blocking techniques used to partition the Big Dataset into a number of chunks where the expected similar records are more likely to be placed in the same chunk. After distributing the dataset into chunks, a similarity function is executing in parallel between those chunks utilizing the MapReduce programming model. The main idea of parallelism is to speed up the Record Linkage runtime and the exploit the ability to scale up.

A. Blocking Techniques

The full detection of Record Linkage process requires the complete scan of all records to compute the similarity between them. Performing a complete scan of all records require executing a Cartesian product of similarity checks between n input dataset which will require a complexity of O(n2). This makes the process of Record Linkage very exhaustive, expensive and tidy process and prone to errors, especially with large datasets. Partitioning the data into blocks where each block contains only records that are more likely to be similar solves this problem by only comparing the records within the same block [9].

Blocking Key is used to partition the Big Datasets into smaller blocks depending on a chosen entity attributes' values. The blocking key is often created by the concatenation of the prefixes of the chosen attributes [10]. Standard blocking, Sorted neighborhood, Q-gram Indexing, and Canopy Clustering with TF-IDF are the different Blocking methods for Record Linkage listed by [11], [12] as illustrated in Table II. The sorted neighborhood method is one of the best-used blocking methods implied by many authors in [13], [14], [10], .[15], [16], [17].

B. MapReduce Programming Model

MapReduce is a shared nothing programming model. It is specially designed to handle the exhaustive processing by paralleling distribution of the workload between multiple clusters or multiple nodes. The design of nodes or clusters could be easily scaled out if needed [18]. In MapReduce paradigm, data is partitioned as described before and placed in blocks then, data in each block is represented by key-value pairs. The processing is done to where the data is located using two user-defined functions, Map and Reduce. The map function is used to partition the input dataset into chunks to be processed separately according to the required function. The reduce function is used to sort and collect the final output from each map function and generate one final output.

TABLE II. Summary of The Blocking Techniques

Standard Blocking	Sorted Neighborhood	Q-Gram Indexing	Canopy Clustering with TF-IDF
- All records with the same blocking key value (BKV) will be on the same block. - Only records within the same block will be compared to check the duplication between them.	- The database is sorted according to (BKV). - A window of a fixed sized is moved on the sorted records. -Candidate records are generated only from records of the current window.	- All database records that have not only the same BKV but also a similar BKV are inserted into the same block. - Each (Blocking Key Value) is transformed to a list of Q-Grams. - Combinations of these q-gram lists are then generated down to a minimum length, determined by a user threshold. - Then a record identifier is inserted into more than one block.	The clustering is done by calculating Similarity between the Blocking Key Values using measures such as TF-IDF or Jaccard.

Fig. 3 illustrates a simple example of MapReduce program. Suppose we want to calculate the frequency of each char from the input string (3 lines in this case for simplicity) by utilizing MapReduce Programming model. First, the input data is split or partitioned into 3 chunks. Second, each chunk goes to a mapper which is responsible to generate a key-value pair (<char>, <char frequency>). Third, the reducer task starts with shuffle and sort step. It sorts the keys generated from the map phase into a larger data list to be easily grouped and integrated into the reduce phase.

By applying the MapReduce model in the Record Linkage problem for Big Data, The Map function will be used to read the input datasets in parallel from the connected nodes and partition them into small chunks and redistribute them based on their blocking key to the reduce functions/tasks that are responsible for Record Linkage process.

C. Blocking based Record Linkage using MapReduce programming model

By utilizing MapReduce model to solve the Record Linkage problem with Big Data, The Map function or mapper will read the input datasets in parallel and partition them into small chunks in the form of a key-value pair (blocking key, entity). The default hash partition function will use this key-value pairs to redistribute the blocks based on their blocking key to the reduce functions/tasks. The reducer function is

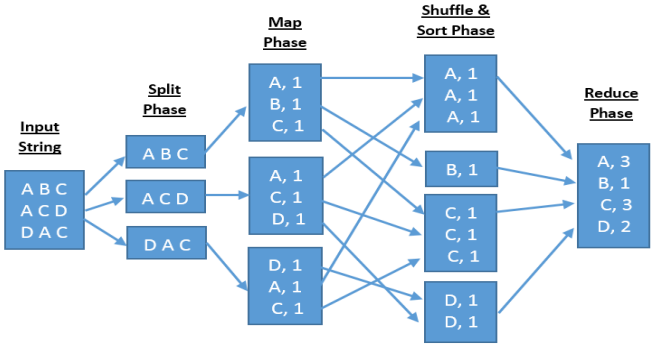

Fig. 3. Calculating the frequency of characters using MapReduce paradigm.

responsible for Record Linkage process using one or combination of the similarity techniques. Entities will whereas the entity has the same blocking key will be processed in the same reducer. This guarantees that all input entities having the same blocking key and reside in the same block will be processed using the same reducer as illustrated in Pseudocode depicted in Fig. 4. So, MapReduce offers a convenient programming model for scaling out record Linkage process by running Record Linkage techniques in parallelized blocking using MapReduce model [19]. The use of basic MapReduce programming model has the following Limitations:

- It is susceptible to data Skew because of unequal distributions of blocks.
- Single or few number of reducers tasks may be the dominant factor of the execution time of the whole Record Linkage process.
- Some nodes of MapReduce model may be ideal waiting for others to complete their tasks.

Therefore, the basic MapReduce model with Big Data is not scalable if we don't solve the bottleneck that is happening during the execution of large blocks, not effective because of the time-consuming cost of running large blocks, and finally has difficulty of handling fault tolerance.

Partitioning Big Datasets into smaller blocks using blocking key techniques may lead to unequal distributed blocks which cause a problem known as data skew. Data skew is happening when the data is not evenly distributed across data partition. An uneven distribution of data degrades the performance of the overall execution as some nodes will sitting idle waiting for some other to finish their job with larger sizes. Data skew can occur on both map and reduce phases. Solving data skew that happens in map phase is easy whereas it is very complex and demanding issue in the reduce phase because the majority of the execution is happening the reduce phase.

A small number of studies consider the use of MapReduce paradigm to solve the load-balancing problem, especially in the Record Linkage. Hadoop is one of the most famous frameworks that implement MapReduce programming model. An extremely improvement in the efficiency of the whole process of the Record Linkage is noticed by balancing the workload of the reducers in the matching step [19].

Unequal distributed of the blocks will cause many problems as Load balancing and fault tolerance between the Reducers in MapReduce programming Model. Many authors tried to solve the load balancing problem by utilizing the idea of blocking key distribution as Yipeng [20] introduced a partitioning solution that dynamically balances the workload of records' comparison to solve the problem of uneven workload that happens due to unequal partitions blocks. Their evaluation achieves a significant improvement that outperforms the default partition of Hadoop for Record Linkage problems involving data skew.

Kolb, et al. [21] addressed the skew problems by proposing a general balancing approach named BlockSplit. The aim of BlockSplit is to reduce the search space of Record Linkage and evenly distribute the workload between the reducers in

```
Map()
{
    Foreach input entity
    {
        Determine the blocking key
    }
    Return Key-value pairs (blocking key, entity)
}
Use the default hash partition to assign each block to the Reduce()
Foreach Block
{
    Reduce ()
    {
        Calculate Similarity between entities()
        Return matching entity pairs
    }
}
```

Fig. 4. Pseudocode of Record Linkage of Big Data using basic MapReduce model.

MapReduce programming model. BlockSplit consists of 2 main jobs. Each of them is done by a number of maps and reduces tasks. First one is analysis job. This job receives the input partitions and performs blocking using blocking key. The output of this job is the number of entities per each block, known as Block Distribution Matrix (BDM). BDM is the input of the second job known as matching phase. Record Linkage is done between elements of the same block in this phase. The approach of BlockSplit took the size of blocks into account and assigned each block to a number of reduce tasks after passing the checking of load balancing constraints. Small blocks will be processed in one reducer. Large Blocks are divided into smaller numbers of sub-blocks based on the input partitions for better performance of parallel matching occurred within multiple reduce tasks. But this approach as mentioned by Kolb, et al. is [22] may still lead to unbalanced reduce load balance because of the different size of sub-blocks. Also, they used round robin as a balancing technique in the first job (analysis) in order to evenly distribute the blocks between the available reducers. Using round-robin technique did not lead to a balanced number of elements between reducers, although it could be ignored because time in this job is not the dominant factor.

Kolb, et al. [22] proposed and evaluated PairRange, a load balancing for MapReduce based Entity resolution approach. PairRange redistributes the entities between reducers such that each reducer computes nearly the same numbers of entities comparisons. PairRange consists of two main jobs, first one is calculating the BDM as described in [21]. The second job calculates the total number of comparisons from all blocks and redistributes them to all available reducers such that each reducer takes almost an equal range of entities comparisons. The approaches proposed by Kolb, et al. in [21] and [22] are scalable as they scale for increasing the number of working nodes comparing to the basic MapReduce model, Robustness against data skew and finally achieve a decrease in execution time comparing to Basic MapReduce model as many nodes are added. BlockSplit and PairRange approaches are only applied to a large scale of datasets and not yet explore in Big Datasets. No filtering or pruning strategy where implemented in both approaches to limit the number of comparison between elements of the same block. In pairRange approach, Elements are not equally distributed in the reducers although the resulting comparisons are balanced between reducers. It is not

clear in PairRange approach how it ignores many elements comparisons as it occurs in other reducers.

Chen, et al. [23] introduced a methodology to carry out Record Linkage without having to perform pair-wise matching. They used matching keys based on MapReduce framework. This methodology consisted of three main parts, Standardization, match key generation, and transitive closure. They overcome the limitation of transitive closure by designing a new iterative transitive closure that applies the method on multiple match keys in MapReduce scenario.

Papadakis, et al. [24] proposed a method to remove redundant comparisons. They also introduced a measurement for quantifying the redundancy entailed by blocking method and explain how to use it to tune the process of comparison pruning. They applied the proposed blocking techniques on two large datasets. The results showed a remarkable increase in the efficiency of the comparison and blocking process.

While Jin, et al. [25] proposed DISC, a distributed algorithm used for single Record Linkage hierarchical clustering based on MapReduce programming model. They introduced the analysis of the algorithm, including an upper bound on the computation cost. The Algorithm was flexible to run in a big dataset by configuring some parameters and showed a scalable speedup of up to 160 on 190 computer cores.

Hsueh, et al. [19] applied multiple key distributions. They tried to solve the Record Linkage problem for a large dataset by proposing a MapReduce algorithm that has multiple keys. The proposed algorithm consists of two phases, first is the combination-based blocking and second is load-balanced matching. They used the two keys in the combination-based blocking to filter out unnecessary entity pairs. In the matching step, the load balancing is done by obtaining statistics of the total number of computation required for each block than evenly distribute the number of comparisons between all the reducers. They only used one input partition. Solving the load balancing problem in this case is much easier than having more than one input partition as in [21] and [22].

Current state of art employs the supervised or unsupervised learning based approaches in the Record Linkage problem where supervised learning aims to classify a pair of records as matched or non-matched. The classifier is learned or trained depending on sets of records labeled as matched or non-matched then it is tested until having an acceptable accuracy. The goal of learning based Record Linkage is to classify new pair records as matched or non-matched depending on the trained data.

Many authors considered the use of machine learning in the Record Linkage problem as Kolb, et al. [17] discussed the use of MapReduce programming model to parallel the workload distribution between many reducers using variations of the Sorted Neighborhood Method (SNM) with a varying size window. They proposed two different strategies for calculating similarity in the case of the Cartesian product of two input sources. The first one named MapSide depended only on computing similarity in the Map phase. While the second one name ReduceSplit evenly distributes entity pairs of the two input sources across all available reduce tasks that finally apply similarity computation for each reducer. Each pair-wise similarity between the common properties in the two data sources serves as a feature of the classification model. Computing similarities are the dominant factor in the classification model because it consumes about 95% of the time. No blocking techniques or load balancing strategies were employed to handle the computational skew results because of unbalanced blocks. No pruning similarity computation was developed to reduce the overall number of comparisons of entities that are likely to be non-matched.

Kolb, et al. [26] devolved a tool called Dedoop (Deduplicate with Hadoop) which supports MapReduce-based Entity Resolution for large datasets. Dedoop had a GUI for its workflow as shown in Fig. 5 that consisted of three steps; blocking, similarity computation and matching decision of the input blocks depending on their similarity value. The final step (match classification) could depend on a trained classifier using a Machine Learning algorithm on the training dataset. Several blocking techniques could be used in the blocking, similarity computation, and match classification. Dedoop did not offer any pruning steps to reduce the number of unnecessary comparisons.

Cao, et al. [9] proposed a new algorithm for the blocking of the Record Linkage problem. The proposed algorithm learned the blocking schema for both labeled data and unlabeled data. Experiments showed that using unlabeled data in the learning process could decrease the number of candidate matches while at the same time maintaining the same level of true matches [9].

Moir, et al [27] introduced the Generic Record Linkage (GRL) framework to classify pairs of entities as matching pairs or non-matching pairs based on their features and the semantic relationship between them. The proposed approach applied supervised Machine Learning to determine the features of the attributes and their semantic relationships used in the classification process. The applying of the proposed approach results in increasing the accuracy of the classification.

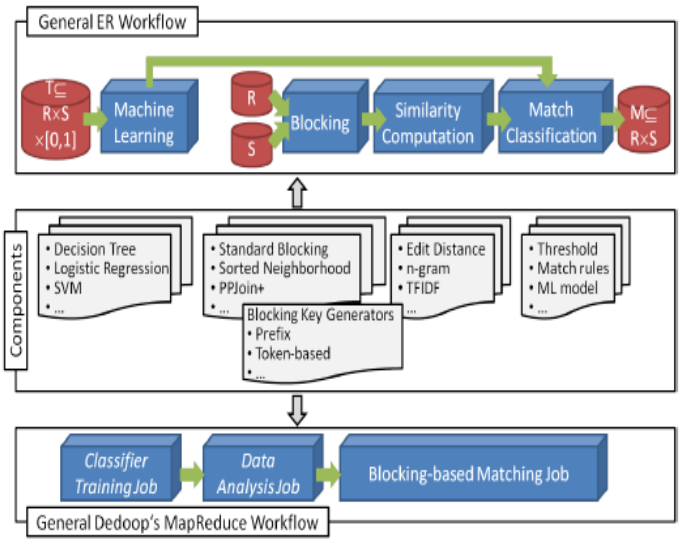

Fig. 5. Overview of Dedoop [26].

IV. Conclusion and Future Work

Record Linkage techniques have many challenges involved in Big Data due to the high resolution of it. This paper surveys the Record Linkage techniques used to solve the volume property of Big Data. Most techniques used MapReduce model for distributed computing and parallel processing. All solutions depend on partitioning the data into blocks using a chosen key by map function, then resolve the duplicates using one of the techniques of duplicate resolution in reduce function. Partitioning input data into blocks may lead to data skew, which results in unbalanced workload that can be solved by additional MapReduce Job to determine the blocking key distribution.

Yet there is not much effort done in load balancing of MapReduce techniques so more work should be dedicated to it because it has not been actively explored. More attention should be pointed to pruning an unnecessary number of comparisons of entities that are likely to be non-matched during the similarity computations. In addition, using ML in the field of Record Linkage is still in its infant stage and needs more efforts to be exerted on it. In future work, we plan to find an adaptive method to balance the load between the data blocks in MapReduce programming model. We will try to execute the mentioned load balancing approaches (BlockSplit and PairRange) on Big Data as they only explored on large data scale. We will exploit ML to formalize the best attributes that could be used for blocking. In addition, trying to apply some pruning methods to avoid the unnecessary comparisons between the records located in the same blocks.

References

[1] M. Dhavapriya and N. Yasodha, "Big Data Analytics Challenges and Solutions Using Hadoop, Map Reduce and Big Table," International Journal of Computer Science Trends and Technology (IJCST), vol. 4, no. 1, 2016.

[2] X. L. Dong and D. Srivastava, Big Data Integration, Morgan & Claypool Publishers, 2015.

[3] X. L. Dong and D. Srivastava, "Big data integration," in ICDE, 2013.

[4] F. Castanedo, Data Preparation in the Big Data Era, USA: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA, 2015.

[5] L. Getoor and A. Machanavajjhala, "Entity Resolution: Theory, Practice & Open Challenges," in VLDB Endowment 5(12), 2012.

[6] C. Kong, M. Gao, C. Xu, W. Qian and A. Zhou, "Entity Matching Across Multiple Heterogeneous Data Sources," in International Conference on Database Systems for Advanced Applications, Cham, 2016.

[7] H. Köpcke, A. Thor and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," in Proceedings of the VLDB Endowment, 2010.

[8] A. Kannan, A. Kannan, R. Agrawal and A. Fuxman, "Matching unstructured product offers to structured product specifications," in 17th ACM SIGKDD international conference on Knowledge discovery and data mining., 2011.

[9] Y. Cao, Z. Chen, J. Zhu, P. Yue, C.-Y. Lin and Y. Yu, "Leveraging Unlabeled Data to Scale Blocking for Record Linkage," in Proceedings of the 22nd International JointConference on Artificial Intelligence (IJCAI), 2011.

[10] L. Kolb, A. Thor and E. Rahm, "Parallel Sorted Neighborhood Blocking with MapReduce," in Proc. Conf. Datenbanksysteme in Buro, Technik und Wissenschaft, 2011.

[11] 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.

[12] P. Christen, "A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication," in IEEE Transactions on Knowledge and Data Engineering X(Y), 2011.

[13] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee and N. Wolfgang, "A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces," in IEEE Transactions on Knowledge and Data Engineering, 2013.

[14] G. Papadakis, E. Ioannou, C. Niedere, T. Palpanas and N. , "Eliminating the Redundancy in Blocking-based Entity Resolution Methods," in Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries, 2011.

[15] L. Kolb, T. and E. Rahm, "Multi-pass Sorted Neighborhood Blocking with MapReduce," in Computer Science-Research and Development, 2012.

[16] D. G. Mestre and C. E. Pires, "An Adaptive Blocking Approach for Entity Matching with MapReduce," in SBBD, 2013.

[17] L. Kolb, H. Köpcke, A. Thor and E. Rahm, "Learning-based Entity Resolution with MapReduce," in CloudDB, 2011.

[18] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in the 6th conference on Symposium on Operarting Systems Design & Implementation, Berkeley, CA, USA, 2004.

[19] S.-C. Hsueh, M.-Y. Lin and Y.-C. Chiu, "A Load-Balanced MapReduce Algorithm for Blocking-based Entity-resolution with Multiple Keys," in Parallel and Distributed Computing, 2014.

[20] Y. Huang, "Record linkage in an Hadoop environment," School of Computing, National University of Singapore., 2011.

[21] L. Kolb, A. Thor and E. Rahm, "Block-based Load Balancing for Entity Resolution with MapReduce," in Proceedings of the 20th ACM international conference on Information and knowledge management, 2011.

[22] L. Kolb, A. Thor and E. Rahm, "Load Balancing for MapReduce-based Entity Resolution," in International Conference on Data Engineering (ICDE), IEEE, Leipzing,German, 2012.

[23] C. Chen, D. Pullen, R. H. Petty and J. R. Talburt, "Methodology for Large-Scale Entity Resolution Without Pairwise Matching," in IEEE International Conference on Data Mining Workshop (ICDMW), 2015.

[24] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas and W. Nejdl, "To Compare or Not to Compare:Making Entity Resolution more Efficient," in Proceedings of the international workshop on semantic web information management, 2011.

[25] C. Jin, M. M. A. Patwary, A. Agrawal, W. Hendrix, W.-k. Liao and A. Choudhary, "DiSC: A Distributed Single-Linkage Hierarchical Clustering Algorithm using MapReduce," in 4th International SC Workshop on Data Intensive Computing in the Clouds (DataCloud), 2013.

[26] L. Kolb, A. Thor and E. Rahm, "Dedoop: Efficient Deduplication with Hadoop," VLDB Endow, vol. 12, no. 5, p. 1878–188, 2012.

[27] C. Moir and J. Dean, "A Machine Learning approach to Generic Entity Resolution in support of Cyber Situation Awareness," in Proceedings of the 38th Australasian Computer Science Conference (ACSC 2015), 2015.