INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 5, Issue 5 NOV 2016

Time Efficient and Improved Parallel Processing Method for Detecting Duplicate Data in CD-Dataset

[1]Bondugulapati Keerthana, [2]Dr. Koppula Srinivas Rao

[1]PG Scholar, Department of CSE, CMR College of Engineering & Technology, Kandlakoya, Medchal, Hyderabad, Telangana, India.

email: keerthanab1992@gmail.com

[2] Professor, Department of CSE, CMR College of Engineering & Technology, Kandlakoya, Medchal, Hyderabad, Telangana, India.

email: ksreenu2k@yahoo.com

***Abstract*—In Data mining, we are getting data from the cloud databases and large size datasets will be increased in the cloud. Hence, when we want to use that dataset, user must clean that dataset. In the processes of data cleaning, duplicate detection is one phase. At present, users wants to process the larger datasets in the less time and that not possible in the existing system. To detect the duplicates in the datasets traditionally, number of methods are there but those are not time efficient and users cannot get accurate data results. Traditionally, we used two methods namely, 1) Progressive Sorted neighborhood Method (PSNM), 2) Progressive Blocking (PB) Method. These two methods are providing the good quality in duplicate detection but those are not time efficient. To overcome this drawback, in this paper we propose a time efficient parallel processing method. This method extended by traditional progressive sorted neighborhood method only. This parallel Processing method, detect the duplicate data faster than the existing methods. In experiments, we can observe the processing time of the parallel method and normal method.**

Keywords—Data Mining, Duplicate Detection, Sorting Key, Parallel Processing.

I. INTRODUCTION

Databases play a major position in latest IT based financial system. Many industries and methods depend upon the accuracy of databases to carry out operations. For this reason (R. Ramesh Kannan, et. al, 2016), the quality of the information saved within the databases, can have huge price implications to a procedure that depends on expertise to operate and conduct trade. Whenever the duplicates need to be found from dataset we tend to select data processing. The information mining takes its 'concepts from information Discovery in info (KDD) within the field of engineering. Within the recent past, duplication is changing into a significant threat in the majority the domains. As a result of this duplication the information received is additional and therefore memory limitation becomes demanding. Therefore admin finds it troublesome to manage the information sets. The duplicate detection processes are dear. The people keep dynamic their portfolio despite retailers providing several product catalogs. In an error-free process with perfectly easy information, the development of a comprehensive view of the data contains linking --in relational terms, joining-- 2 or

more tables on their key fields. Unluckily, data commonly lack a specified, international identifier that will allow such an operation. Moreover, the data are neither cautiously managed for quality nor outlined in a consistent means across distinct data sources. Therefore, knowledge quality is as a rule compromised by way of many causes, together with knowledge entry error (e.g., studet in the place of student), missing integrity constraints (e.g., permitting entries comparable to Employee-Age=567), and a couple of conventions for recording expertise to make things worse, in independently managed databases now not handiest the values, but the structure, semantics as well as underlying assumptions about the information may just differ as good. Revolutionary duplicate detection identifies most replica pairs early in the detection approach. Instead of reducing the overall time needed to conclude the complete process (S. Ramya and Palaninehru, 2015), revolutionary procedures attempt to lessen the normal time after which a replica is determined. Duplicate detection is the method of settling on more than a few representations equal real-world purpose in a knowledge source. The quality of replica detection, i.e., its effectiveness, scalability cannot be unobserved that of the gigantic measurement of the database. The replica detection drawback has two features: First, the more than one representation are frequently no longer the equal but incorporate differences, equivalent to misspellings, converted addresses, or missing values. This makes it difficult to realize these duplicates. 2nd, reproduction detection is a very luxurious operation, because it requires the comparison of each possible pair of duplicates using the traditionally complex similarity calculate. Progressive methods make this exchange-off extra invaluable as they deliver more whole outcome in shorter quantities of time. Revolutionary Sorted nearby procedure take smooth dataset and find some replica files and progressive blocking take dirty datasets and realize significant replica files in databases. And finally, in this paper we propose parallel processing method and our work extends by these sorting methods.

II. RELATED wORK

Dong et al. Perform reproduction detection within the PIM area by way of using relationships to propagate similarities from one duplicate classification to yet another. The important focus of their process is developing of effectiveness with the aid of using relationships. In contrast, we are aware of increasing effectively via using relationships. Before describing our method in detail we supply some definitions and gift an illustration of our technique. (Ahmed K. Elmagarmid, Panagiotis G.Ipeirotis, Vassilios, S. Verykios, 2007) mostly, in the actual world, entities have two or extra representations in databases. Duplicate records don't share a fashioned key and/or they incorporate blunders that make duplicate matching a complex challenge. Error is offered as the outcome of transcription error, incomplete expertise, lack of usual formats, or any blend of those reasons. On this paper, we gift a thorough evaluation of the literature on replica report detection. We cover similarity metrics which can be most of the time used to notice identical area entries, and we proposed an extensive set of duplicate detection algorithms that can notice approximately reproduction records in a database. We additionally cover a couple of tactics for improving the efficiency and scalability of approximate duplicate detection algorithms. We conclude with protection of existing tools and with a quick dialogue of the significant open problems in

the discipline(S. Ramya and Palaninehru, 2015). The trouble that we learn has been identified for more than five many years because the file linkage or the file matching trouble in the records community. The purpose of document matching is to identify records in the equal or unique databases that confer with the same real-world entity, even if the files should not identical.

Pay as you go method investigates how we are able to maximize the development of ER with a restricted amount of work making use of "hints," (S. E. Whang, D. Marmaros, and H. Garcia-Molina,2012) which presents expertise on documents which are prone to point out to the same object. A hint can be represented in one of a kind formats (e.g., a clustering of records based on their possibility of matching), and ER can use this information as a guiding principle for which files to evaluate first. A pay-as-you-go technique to entity resolution, the place we acquire fractional results regularly" as a way to as a minimum get some outcome faster. An ER approach is very luxurious due to very large data sets and compute-intensive report comparisons.

A. Adaptive Approaches

Earlier publications on replica detection traditionally focus on decreasing the overall runtime (X. Dong, A. Halevy, and J. Madhavan, 2005). Thereby, one of the crucial proposed algorithms is already in a position of estimating the high-quality of assessment candidates. The algorithms use this information to prefer the assessment candidates extra cautiously. For the equal rationale, other systems utilize adaptive windowing methods, which dynamically modify the window measurement relying on the quantity of not too long ago located duplicates. These adaptive techniques dynamically fortify the efficiency of reproduction detection, but run for unique durations of time and might- not maximize the affectivity for any given time slot.

B. Drawbacks of Traditional Methods

1) These adaptive approaches dynamically give a boost to the efficiency of duplicate detection, however unlike our revolutionary procedures, they have got to run for unique durations of time and can't maximize the affectivity for any given time slot.

2) Wants to method giant dataset in less time

3) Quality of dataset turns into increasingly complex

III. FRAMEWORK

The main aim of this paper is to detect the duplicate data in the different large and small datasets as a parallel. In this paper, we are detecting the duplicates on CD dataset. To detect duplicate data in the dataset, we follow the three steps,

- Pair selection
- Pair wise comparison
- Clustering

A. System Architecture

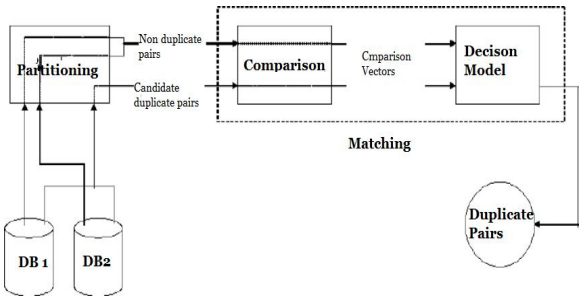


Fig.1. System Architecture

In the above architecture, we take some datasets and the first step, we are partition our complete dataset. Partition nothing but if we give a partition

size=30 then this means, we are keeping 30 records in every partition. After partition the dataset, we can perform the sorting algorithm on the dataset. In that sorting, it will compare the duplicates as pair wise comparison. After comparison it will display the duplicate pairs to us.

B. Dataset Overview

In this paper, we are detecting the duplicates on CD dataset. It contains 9763 records and these records related to the music and audio CDs. This dataset contain some attributes like, ID, artist, category, genre, cdextra, and year. From these attributes we can get some attributes as sorting keys by using attribute concurrency method. If we select "artist" as a sorting key then the processing done only based on the artist related data only and after completion of processing it display the duplicate text of artist attribute from dataset.

C. Sorting Key

1) Need of Sorting Key

Importance of this sorting key is, generally large dataset contains lakhs and thousands of records. For every time read complete dataset and detect the all duplicates in the dataset is not possible. In sometimes user needs detect the duplicate data and detect the duplicate count on only particular data. In this type of situations, we need a sorting key. Without sorting key it is difficult to sort the data from dataset.

To sorting the dataset, we are using magpie sorting. In this sorting we are select one sorting key. To select the best key for sorting we are using attribute concurrency method.

a) Sorting Key Selection

The best key for locating the duplicate is usually hard to identify. Selecting good keys can increase the progressiveness. Here all the records are taken and checked as a parallel processes so as to reduce average execution time. The records are kept in multiple resources when splitting. The intermediate duplication results are intimated instantly when found in any resources and came back to the most application. Therefore the time consumption is reduced. Resource consumption is same as existing system however the information is kept in multiple resource memories.

D. Algorithms

1) Progressive Sorted Neighborhood Method(PSNM) Algorithm

Require: dataset reference D, sorting key K, window size W, enlargement interval size I, number of records N

```
1:  procedurePSNM(D, K, W, I, N)
2:  pSize← calcPartitionSize(D)
3:  pNum← [N/(pSize – W + 1)]
4:      array order size N as Integer
5:      array recs size pSizeas Record
6:  order ← sortProgessive(D, K, I, pSize,    pNum)
7:  forcurrentI← 2 to [W/I] do
8:  for current ← 1 topNumdo
9:  recs ← loadPartition(D, currentP)
10:             for dist∈ range (current, I, W) do
11: for i← 0 to|recs| –distdo
12: pair ← <recs[i], recs[i + dist]>
13: if compare(pair) then
14: emit(pair)
```

15: lookAhead(pair)

2) Progressive Blocking (PB) Algorithm

Require: dataset reference D, key Attribute K, maximum block range R, block size S and record number N

1: **procedure PB(D, K, R, S, N)**
2: pSize← calcPartitionSize(D)
3: bPerP← [pSize/S]
4: bNum← [N/S]
5: pNum← [bNum/bPerP]
6: **array** order **size** N **as** Integer
7: **array** blocks **size** bPerP**as** <Integer, Record[]>
8: **priority queue** bPairs**as** <Integer, Integer, Integer>
9: bPairs← {<1,1,_>,… ,<bNum, bNum, _>}
10: order ← sortProgressive(D, K, S, bPerP, bPairs)
11: **for** i← 0 **to** pNum – 1 **do**
12: pBPs← get(bPairs, i.bPerP, (i+1).bPairs)
13: blocks ← loadBlocks(pBPs, S, order)
14: compare(blocks, pBPs, order)
15: **while** bPairs is not empty **do**
16: pBPs← {}
17: bestBPs← takeBest([bPerP/4], bPairs, R)
18: **for** bestBP∈bestBPs**do**
19: **if** bestBP[1] – bestBP[0] < R **then**
20: pBPs← pBPs∪ extend(bestBP)
21: blocks ← loadBlocks(pBPs, S, order)
22: compare(blocks, pBPs, order)
23: bPairs← bPairs∪pBPs
24: **procedure** compare(blocks, pBPs, order)
25: **for** pBP∈pBPs **do**
26: <dPairs, cNum>– comp(pBP, blocks, order)
27: emit(dPairs)
28: pBP[2] – |dPairs| / cNum

E. Parallel Processing Method

Parallel processing means we execute the number of processes at a time that means parallel this is caused by using some concurrency methods. In this method first we are partition the dataset complete dataset. These concurrency methods execute the all partitions of the dataset at a time to reduce the execution time of the process. This proposed method selects the sorting key from dataset by using attribute concurrency method. And it also takes the window/block size to partition the complete dataset. Basically, our proposed system extended by traditional Progressive Sorted Neighborhood Method (PSNM) and Progressive Block (PB) for that reason we need to give the window size as partition size. Based on these sorting key and window size, the parallel processing method executes the all partitions of the dataset and it also display the parallel processing time of the proposed method.

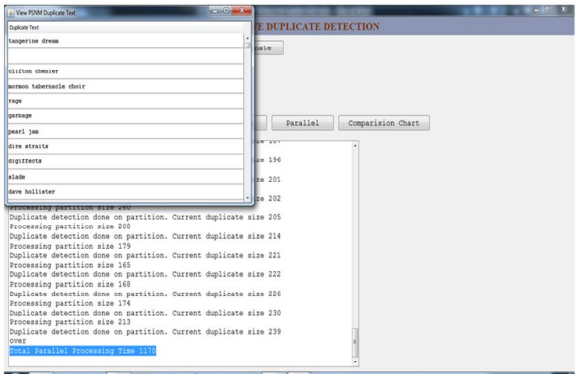
IV. EXPERIMENTAL RESULTS

In our experiments, we are detecting the duplicates on the CD-Dataset by using parallel processing method. The first step is in our experiment are we needs to upload the CD-dataset into the system. After upload dataset, we must select the sorting key and the window/block size. This window/block size used to partition the complete dataset and it is calculated by this formula:

Partitions of Complete Dataset=Dataset Size/ (Window/block size)

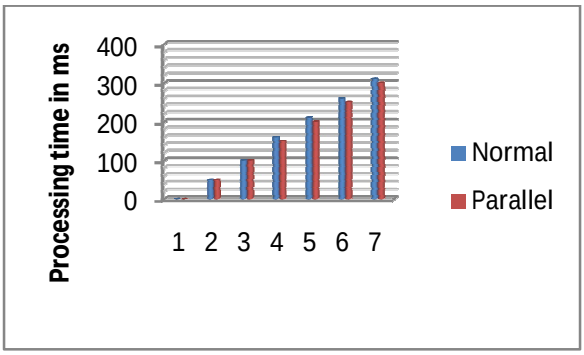
By using this formula, partitions size will be displayed and every partition size along with it duplicates size we can view in the system. And finally we get the processing time of the algorithms also it displays.

Here, we perform the traditional PSNM algorithm as well as traditional PB algorithm to verify the processing time our proposed parallel processing method.



The above screen shows that the processing time of the parallel method.

The below screen shows that the comparisons chart for normal processing time and parallel processing time:



From our experiments, we can say that our proposed parallel processing method is a time efficient method to detect the duplicates.

V. CONCLUSION

Finally, we conclude that in this paper we proposed a time efficient and parallel processing method. This proposed method inspired by the traditional PSNM and PB algorithms. In our proposed method we can get the duplicate detection time, duplicate count and duplicate text. In this experiment we used CD-Dataset and from this dataset we detect the duplicate count and duplicate text within the milliseconds of time. Eventually, we proved that our proposed method is time efficient than the traditional algorithms.

References

[1] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," IEEE Trans. Knowl. Data Eng., vol. 19,no. 1, pp. 1–16, Jan. 2007.

[2] S. E. Whang, D. Marmaros, and H. Garcia-Molina, "Pay-as-you-go entity resolution," IEEE Trans. Knowl. Data Eng., vol. 25, no. 5, pp. 1111–1124, May 2012.

[3] M. Wallace and S. Kollias, "Computationally efficient incremental transitive closure of sparse fuzzy binary relations," in Proc. IEEE Int. Conf. Fuzzy Syst., 2004, pp. 1561–1565.

[4] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," IEEE Trans. Knowl. Data Eng., vol. 24, no. 9, pp. 1537–1555, Sep. 2012.

[5] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz, "The Plista dataset," in Proc. Int. Workshop Challenge News Recommender Syst., 2013, pp. 16–23.

[6] M. A. Hernandez and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," Data Mining Knowl. Discovery, vol. 2, no. 1, pp. 9–37, 1998.

[7] X. Dong, A. Halevy, and J. Madhavan, "Reference reconciliation in complex information spaces," 2005, pp. 85–96.

[8] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller, "Framework for evaluating clustering algorithms in duplicate detection," Proc. Very Large Databases Endowment, vol. 2, pp. 1282–1293, 2009.

[9] S. Ramya and PalaninehruA, "Study of Progressive Techniques for Efficient Duplicate Detection", 2015.

[10] R. Ramesh Kannan, D. R. Abarna, G. Aswini, P. Hemavathy, "Effective Progressive Algorithm for Duplicate Detection on Large Dataset", 2016.