International Journal of Advanced Engineering Research and Science (IJAERS) [Vol-3, Issue-11, Nov- 2016]
https://dx.doi.org/10.22161/ijaers/3.11.37 *ISSN: 2349-6495(P) | 2456-1908(O)*

Novel and Efficient approach for Duplicate Record Detection

Mrs. D. V. LalitaParameswari.[1], K. Mounika[2]

[1]Sr. Asst. Professor, Dept. of CSE, GNITS, Hyderabad, India
[2] M. Tech Student, Dept. of CSE, GNITS, Hyderabad, India

Abstract*—Similarity check of real world entities is a necessary factor in these days which is named as Data Replica Detection. Time is an critical factor today in tracking Data Replica Detection for large data sets, without having impact over quality of Dataset. In this system primarily introduce two Data Replica Detection algorithms, where in these contribute enhanced procedural standards in finding Data Replication at limited execution periods. This system contribute better improvised state of time than conventional techniques. We propose two Data duplicate record detection algorithms namely progressive sorted neighbourhood method (PSNM), which performs best on small and almost clean datasets, progressive blocking (PB), and parallel sorted neighbourhood method which performs best on large and very grimy datasets. Both enhance the efficiency of duplicate detection even on very large datasets.*

Keywords*— *Data cleaning, Duplicate detection, Entity Resolution, Progressiveness.

I. INTRODUCTION

Now-a-days, Databases play a primary role in IT situated economy. Many industries as well as systems rely on the accuracy of databases to carry out operations. As a result, the worth of the data will be saved in the databases; can have significant price suggestions to a system that relies on data to operate and perform business. In an error-free system with exactly clean data, the construction of a comprehensive view of the information contains linking -- in relational phrases, joining-- two or more tables on their key fields. Unfortunately, information most commonly needs a unique, world identifier that may permit such an Operation. Furthermore, the information is neither cautiously controlled for outstanding nor defined in a consistent means throughout distinctive data sources. [2]Accordingly, information quality is frequently compromised by using many causes, together with knowledge entry errors (e.g., studet as an alternative of student), missing integrity constraints (e.g., enabling entries), and more than one conventions for recording information To make things poorer, in independently managed databases not most effective the values, but the constitution, semantics and underlying assumptions about the data could vary as well. The Progressive techniques may method larger dataset in brief span of time and also the quality of knowledge is additionally smart relatively. The Progressive duplicate detection makes it totally different from the normal approach by yielding additional advanced results throughout the first termination; the algorithms of duplicate detection additionally compute the duplicates at a virtually constant frequency however the progressive algorithms increase the time because it finds out the duplicates at the first stage itself. The proposed system enhances the strength of duplicate detection even on very massive datasets. The parameterization complexness for duplicate detection is created comfortable generally and contributes to the event of additional user interactive applications.

II. LITERATURE SURVEY

The sorted Neighborhood process depends on the assumption that replica records can be close in the sorted record, and accordingly shall be when compared for the duration of the merge step. The effectiveness of the sorted neighborhood strategy is totally dependent upon the contrast key that's selected to sort the records. Typically, no single key shall be plenty to sort the documents in this sort of approach that all the matching files may also be detected. If the error in a file occurs within the unique discipline or element of the subject that's the fundamental a part of the sorting key, there's a very small probability that the file will turn out to be practically an identical record after sorting. [4]To expand the quantity of identical records merged, Herna´ndez and Stolfo carried out a approach for executing a couple of independent runs of the Sorted-Neighborhood Method by means of using yet another sorting key and a slightly small window every time. This process is known as the multi-pass technique. This method is similar in spirit to the multiple-run blocking approach described above. Each impartial run produces a collection of pairs of documents that can be merged. The final outcomes, including the transitive closure of the files matched in extraordinary passes, are due to this fact computed.

https://dx.doi.org/10.22161/ijaers/3.11.37 *ISSN: 2349-6495(P) | 2456-1908(O)*

A. Map-Reduce Algorithm

A map reduced algorithm was introduced which has high affability for scheduling about responsibilities for dynamic load balancing. [6]The author Oktie, presents the Stringer framework that gives an evaluation arrangement to understanding what hindrances remain towards the objective of really flexible as well as broadly useful duplication recognition calculations. Few unrestrained bunch algorithms are assessed for copy discovery by broad examinations over totally different arrangements of string information with numerous attributes. A theme was introduced to combine multisource data. The results from the preliminary examinations are according that was taken from four card inventory databases that rescale to over ten million records are according within the paper.

B. Sorted Neighborhood Method with Map-Reduce

This method introduces new approaches reduce time.

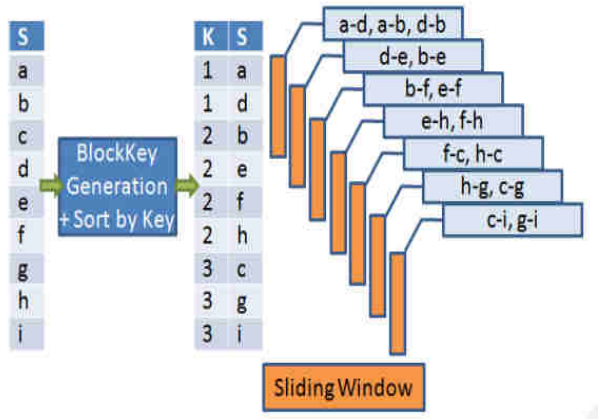


Fig.1: Example Execution of SNM

Sorted Neighborhood Method (SNM) is a popular blocking approach that works as follows;

A blocking key K is determined for each of n entities. Generally the concatenated prefixes of a few attributes form the blocking key. Afterwards the entities are sorted with the aid of this blocking key. A window of a fixed size w is then forward over the sorted records & in each step all entities within the window, i.e., entities inside a distance of w−1, are when put next. Above figure shows a SNM example execution for a window size of w = 3. This is the time consuming process.

C. Incremental Adaptive SNM

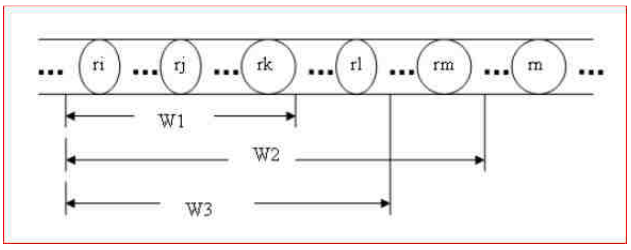


Fig.2: Incremental Adaptive SNM

The fundamental thought is to measure whether records in a small neighborhood are close or sparse and if there are rooms to develop/shrink within the window, and then the window size is extended or decreased dynamically. In order to measure the record distribution within a window, it appears as if we need to measure the distances between all of the records within the window. If the distance between the primary and final record satisfies dist(r1,rw1), the place φ is the distance threshold. This distance indicates that files within the present window are virtually every other, so there's still room to enlarge the window size to find more abilities duplicate records. In any other case the window must be retrenched.

III. FRAMEWORK

A. Duplicate Detection Architecture

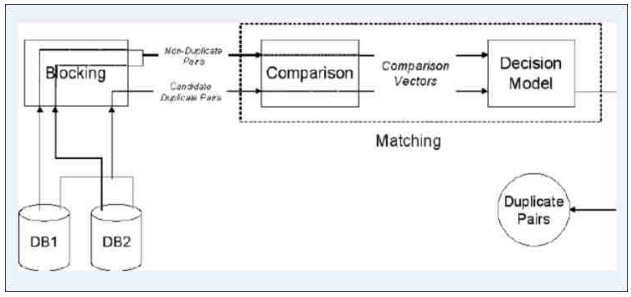


Fig.3: Duplicate Detection System

For instance, if we take an online shopping database, in that numbers of catalogues are there and number of employees is enter the data into the database. So, there is possible to enter the same data number of times. That is referred as duplicate data. If this duplicate data is increased in the database then there is no space for other information means here reduces the storage space of the database. This is the major problem of duplicate data. To overcome this problem, there are various approaches but those are not efficient as well as they are time consuming approaches. In fig3, first the complete data has to be collected from databases. After that, our system need to select pairs of data and compare those pairs. Which pairs are duplicates those duplicates are clustered into a group. Like this system can detect and remove the duplicate data. The main objective of this paper is to detect duplicate data and count the duplicates in the large datasets within the less time. For that in this paper, we propose two new methods to detect the duplicate data as well as count the duplicates in the complete dataset as a parallel. Those two algorithms are;

1. Progressive Sorted Neighborhood Method (PSNM)
2. Progressive Blocking (PB)

And these two are generalized by the existing sorted neighborhood method. In existing method we got the good quality duplicate data but it is very time consuming.

https://dx.doi.org/10.22161/ijaers/3.11.37 *ISSN: 2349-6495(P) | 2456-1908(O)*

Hence, this paper introduce the two progressive and parallel methods. These two are detect the duplicates within in the milliseconds.

B. Sorting Key Selection

In this project we are sorting the dataset by using the magpie sorting. In these sorting methods, we need to select the sorting key to sort the dataset through that key. Importance of this sorting key is, we are mostly applying these two algorithms on the large datasets means those are in thousands and lakh of records are stored in the dataset. But, sometimes user needs deduplication and detect the duplicate count on only particular data. This type of situations, we need a sorting. Without sorting key it is difficult to sort the data from dataset.

For selecting the sorting we propose an Attribute Concurrency method. Through this method we can select the best key for sorting. An attribute concurrency method works based on the multi-pass execution method. This multi-pass method executes the multiple keys in each pass. Attribute Concurrency method we apply to the progressive sorted neighborhood method as well as progressive blocking.

C. Progressive Sorted Neighborhood Method (PSNM)

The algorithm takes five input parameters: D is a reference to the data, which has not been loaded from disk yet. The sorting key K defines the attribute or attributes combination that should be used in the sorting step. W specifies the maximum window size, which corresponds to the window size of the traditional sorted neighborhood method. When using early termination, this parameter can be set to an optimistically high default value. Parameter I defines the enlargement interval for the progressive iterations. For now, assume it has the default value 1. The last parameter N specifies the number of records in the dataset. This number can be gleaned in the sorting step, but we list it as a parameter for presentation purposes. Progressive Sorted Neighborhood Require: dataset reference D, sorting key K, window size W, enlargement interval size I, number of records N

Step 1: procedure PSNM(D, K, W, I, N)
Step 2: pSize← calcPartitionSize(D)
Step 3: pNum←[N/pSize-W + 1)]
Step 4: array order size N as Integer
Step 5: array recs size pSize as Record
Step 6: order ←sortProgressive(D, K, I, pSize, pNum)
Step 7: for currentI← 2 to[W/I]do
Step 8: for currentP ←1 to pNum do
Step 9: recs← loadPartition(D, currentP)
Step 10: for dist belongs to range(currentI, I, W) do
Step 11: for i ←0 to |recs|_ dist do
Step 12: pair←<recs[i], recs[i + dist]>
Step 13: if compare(pair) then
Step 14: emit(pair)
Step 15: lookAhead(pair)

D.Progressive Blocking

The algorithm accepts five input parameters: The dataset reference D specifies the dataset to be cleaned and the key attribute or key attribute combination K defines the sorting. The parameter R limits the maximum block range, which is the maximum rank-distance of two blocks in a block pair, and S specifies the size of the blocks. Finally, N is the size of the input dataset.

Progressive Blocking Require: dataset reference D, key attribute K, maximum block range R, block size S and record number N.

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

E.Parallel Sorted Neighborhood Method

In particular, introduced a two phase parallel SNM, which executes a traditional SNM on balanced, overlapping partitions. Here, we can instead use our PSNM to progressively find duplicates in parallel. By using this method duplicate detection to deliver results even faster compare to progressive sorted neighborhood method and progressive blocking.

F. Magpie Sorting:

The sorting of records may be a block preprocessing step that we are able to already use to (progressively) execute

https://dx.doi.org/10.22161/ijaers/3.11.37 *ISSN: 2349-6495(P) | 2456-1908(O)*

some initial comparisons. Magpie Sort may be a naive algorithm that works the same as Selection Sort. The name of this algorithmic rule is impressed by the larcenous bird that collects beautiful things whereas only being able to hold a few of them directly. Magpie Sort repeatedly iterates overall records to search out the presently top-x smallest ones. Thereby, it inserts each record into a sorted buffer of length x. whether the buffer is full; every new inserted record displaces the biggest record from the list. Each iteration the final order are often supplemented by following top x records from the buffer. A record that has been emitted once won't be emitted once more. In fact, Magpie Sort integrates the complete first progressive iteration of PSNM. Overall, this sorting method generates only a small overhead, as a result of the algorithmic rule needs to repeat over the complete dataset anyway whenever a partition has to be read from disk.

G. Attribute Concurrency Method:

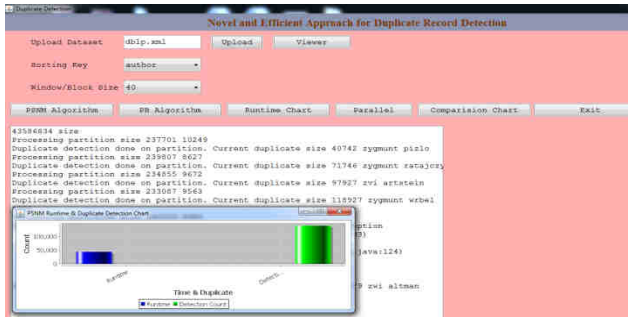
The best key for locating the duplicate is usually hard to identify. Selecting good keys can increase the progressiveness. Multi-pass execution will be applied for progressive SNM. Key separation isn't required in PB algorithmic rule. Here all the records are taken and checked as a parallel processes 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.

IV. EXPERIMENTAL RESULTS

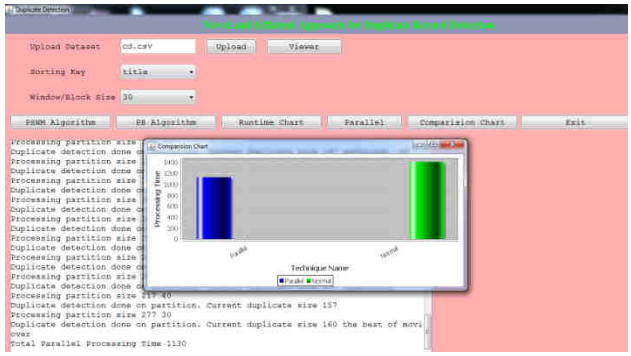
In this system consider some datasets (DBLP,CD DATASETS) which is in different formats like xml ,csv etc. To detect the duplicates and duplicate count in the dataset, first select the sorting key and then give the 'title', 'author', and some attributes in the dataset are the sorting keys. After that choose window size or block size. Finally implements PSNM, PB& PARALLEL SNM.

The below screen shows that the duplicate count and duplicate data in the dataset;

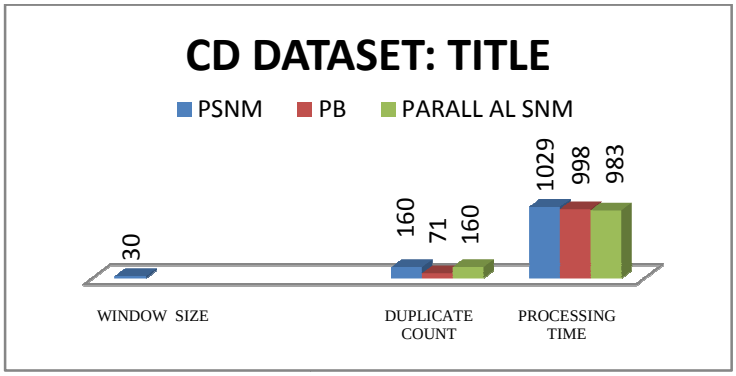
The below screen shows that the normal execution or processing time of the algorithms;



The below screens shows that, the comparison between the normal processing time and parallel processing time of the algorithms;

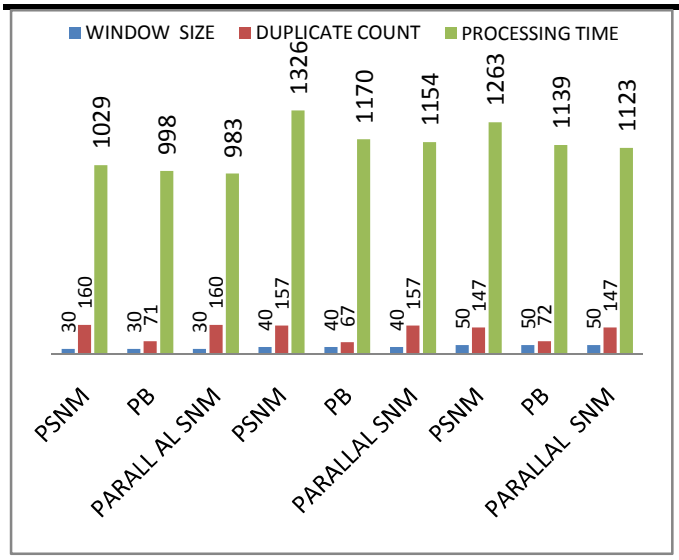


The below graph shows that,the no of duplicates in CD dataset the keyword "Title", It shows the no of duplicates and processing time with in a window in all the methods.



Three approaches will give the results varying the window sizes will get the duplicate count and processing time as shown below,

https://dx.doi.org/10.22161/ijaers/3.11.37 *ISSN: 2349-6495(P) | 2456-1908(O)*



From above results it can prove that proposed algorithms are time efficient and scalable approaches. Parallel SNM will take less time to compare with PSNM and PB.

V. CONCLUSION

In this paper, introduced two methods named, Progressive SNM, progressive blocking and parallel SNM which improve the efficiency of duplicate detection. By this efficiency that time will be reduced for duplicate detection. These two algorithms are generalized by the traditional sorted neighborhood method only. Using these two algorithms reduced the processing time of duplicate detection as well as increased performance. Parallel SNM achieves the better processing time and duplicate count accurately compare to PSNM and PB. In future work, implementing of all the time factors and methods to improve the performance in parallel approach.

REFERENCES

[1] Papenbrock T., Heise A. and Naumann F. (2015), „Progressive duplicate detection", Proc. IEEE Trans. Know. Data Eng., vol. 27, No. 5, pp. 1316-1329.

[2] Whang S. E., Marmaros D., Molina H. (2012),„Pay-as-you-go entity resoln", IEEE Trans. Know. Data Eng., vol. No 25.5, pp. 1111–1124.

[3] Draisbach U, Naumann F, Szott S, Wonneberg O. (2012),„Adaptive windows for duplicate detection", Proc. IEEE 28th Int. Conf. Data Eng., pp. 1073-1083.

[4] Draisbach U. and Naumann F. (2011), „A generalization of blocking and windowingalgorithms for duplicate detection",Proc. Int. Conf. Data Knowl. Eng., pp. 18-24.

[5] Hassanzadeh O., Chiang F., Lee H.C., Miller R. J. (2009), „Framework for Evaluating Clustering Algorithms in Duplicate Detection",Proc. Very Large Databases Endowment, Vol. 2, pp.1282-1293.

[6] 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.

[7] H. B. Newcombe and J. M. Kennedy, "Record linkage: Making maximum use of the discriminating power of identifying information," Commun. ACM, vol. 5, no. 11, pp. 563–566, 1962.

[8] 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.

[9] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, ―Swoosh: a generic approach to entity resolution,‖ VLDB Journal.

[10] R. Baxter, P. Christen, and T. Churches, ―A comparison of fast blocking methods for record linkage,‖ in Proceedings of the ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation, 2003, pp. 25–27.