ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Continuous Duplicate Detection

B.Guna Harika
M. Tech Student,
Sreenidhi Institute of Science and Technology, Hyderabad.

Dr.V.V.S.S.S.Balaram
Professor,
Sreenidhi Institute of Science and Technology, Hyderabad.

Sunil Bhutada
Associate Professor,
Sreenidhi Institute of Science and Technology, Hyderabad.

Abstract:

As the usage of data increasing abundantly, the problem of data quality flourishes. In today's world objects the same data is representing in different customs, duplication is one among data quality problems. Duplication of data causes unfavorable effects. For instance, bank trader obtains duplicate identities, catalog levels are monitored inaccurately, etc. Detecting duplicates automatically is difficult task. Process of identifying various representations of same real world data entity is Duplicate detection. In today's world detecting a large quantity of duplicate entity in short period of time and maintaining data quality is difficult. Using Genetic Algorithm, we can solve the problem of duplicate detection and increase efficiency of detecting duplicate data in short period. This technique provides a competent way to detect replicated content in different location switch different file name or dissimilar content with matching file name.

Key words:

Duplicate detection, entity resolution, progressive, and data cleaning.

1. INTRODUCTION:

Knowledge discovery or Data mining is the logical process of extracting and analyzing massive set of data and extort the implication of the data. To make proactive and knowledge driven decisions data mining tools are used. These tools predict behaviors in less time whereas traditional tools consume a lot of time to resolve. These tools clean data bases for hidden patterns, finding analytical information. Although data mining is still in its infancy, wide range of industries including trade, funding, medical field, manufacturing shipping, and aerospace uses data mining techniques and tools to take benefit of chronological data. Data mining provides analysts to identify significant Data used to find relationships and patterns in turn to create enhanced business decisions. Using pattern recognition, statistical and mathematical techniques we can filter information in warehouse. In today's IT based economy databases play a vital role. Industries depend on the accurateness of databases to accomplish operations. The quality and cost implication are stored in the database should be more significant to the data which relies on operating and carry out businesses.

In an ideal system with perfect data, the structure of a complete analysis of the data consists of relational terms combination of two or more tables on key fields. Data lacks to provide unique identifier to per such operation. The data suspiciously neither controls the quality of data nor a consistent approach among various data sources.

Earlier systems such as continuous duplicate detection algorithms-(PSNM) Progressive Sorted Neighborhood Method and (PB) Progressive Blocking, produces large amount of replicated data sets. To detect the replicate data sets we apply genetic programming algorithm (GP). GP algorithm is good at detecting similarities and finding duplicate records.

In the below, section 2 discusses about literature survey, section 3 talks about progressive Continuous Duplicate Detection, section 4 discusses about PSNM, section 5 discusses about PB, section 6 discusses about attribute parallel method, section 7 discusses about genetic programming algorithm, section 8 discusses about architecture, section 9 discusses about performance evaluation and results.

2. LITERATURE SURVEY:

L. Kolb et.al [1] discussed about Cloud based infrastructures that supports the effective parallel execution of data-intensive tasks such as entity resolution on large datasets. The work investigated about the challenges and feasible solutions of using the model of Map Reduce programming for parallel entity resolution. In precise, two Map-based implementations are proposed and evaluated for Sorted Neighborhood blocking which may use multiple Map Reduce jobs or apply a optimized data replication among the two proposals considered.

P. Christen [2] discussed about Record linkage process that matches records referred to the same entities but collected from many different databases. If the same is applied on a single database, then the process is termed as deduplication. Surprisingly, in many applications, matched data is crucial either due to cost or they can have information which may be unavailable elsewhere. In data cleansing process, the vital step is to eliminate the duplicate records from a single database which may further influence the possible outcomes of the subsequent data mining or data processing. As the size of the single database is advancing today, the complexity of the matching process also increased and became one of the major challenges for record linking and deduplication.

Therefore, in the matching process, it is aimed to reduce the comparison among the recorded pairs by eliminating the unmatched pairs, at the same time, there should be no compromise on the quality of matching. In this work, a survey of 12 variations of 6 indexing techniques were presented besides with analysis of their complexity and evaluation of performance and scalability within a frame work which use both real and synthetic datasets. U. Draisbach and F.Naumann [3] suggested that to find multiple dataset records which represent similar real-world entities, one can employ duplicate detection process.

However, due to the bulk cost involved in exhaustive comparison, promising record pairs are selected for comparison by typical algorithms. Blocking and windowing are the two challenging approaches. In blocking, record partitions into disjoint subsets, whereas in windowing method, especially the Sorted Neighborhood method, records are compared within a window by selecting over the sorted records. In this work, a new algorithm in several forms called Sorted Blocks, which could generalize the above two approaches was presented. Sorted Blocks were evaluated here by conducting extensive experiments with different datasets. For that, this new algorithm requires fewer comparison to find the similar duplicates.

O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller[4] discussed that the formation of duplicate records in large databases leads to a major data quality concern. Entity resolution called as record linkage or duplicate detection is used to identify duplicates that refer to the same-world entity and to assist the data cleansing process. A Stringer system is presented in this work to understand the barriers that remain in the duplication algorithms provided within an evaluation framework. Also this work used Stringer to calculate the clusters (a group of potential duplicates) quality obtained from various unconstrained clustering algorithms used along with approximate join techniques.

The work is highly encouraged by applying recent and significant advancements which have made high efficient approximate join algorithms. Interestingly, the work results have proved that some clustering algorithms performed efficiently with accuracy and scalability which were not used regularly before.

3. PROGRESSIVE CDD:

Continuous duplicate detection (CDD) is combination of Progressive sorted neighborhood method (PSNM) and progressive blocking (PB),PSNM performs best on fewer amounts of data and clean datasets.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

PB performs best on large and unclean datasets by using these two algorithms to identify the duplicate data, now we implement the generic programming algorithm (GP)used to detect duplicate data. Duplications are saved when data with same content and different name is detected.

4. PROGRESSIVE SORTED NEIGHBORHOOD METHOD

Progressive sorted neighborhood method (PSNM) [6] sorts given data by using predefined sorting key and compares records which are in the sorted order. The perception record closes in the sorted order and likely duplicates the records, because they are similar with their sorting key. In particular, distance of two records sorttheir ranks (rank-distance) and gives PSNM to estimate matching likelihood. The PSNM algorithm uses perception to iteratively differ in window size, begin with a small window size that quickly finds the most capable records. This algorithm is used for small datasets and it calculates as suitable partition size p Size, i.e., the utmost number of records that fit in memory.

If the data is taken from a database, it calculates the size of data and matches this to the available main memory. Otherwise, a sample of records are taken and for each field it estimates the size of a record with the largest values. The algorithm calculates the number of necessary partitions spNum, while considering a partition overlap of W – 1 record to slide the window across their boundaries. The order-array keeps the order of records with respect to the given key K. By storing only record IDs in this array; we assume that it can be kept in memory. To hold the actual records of a current partition, PSNM declares the record array.

```
ProgressiveSortedNeighborhoodRequire:
datasetreferenceD, sortingkeyK,
windowsizeW, enlargementintervalsizeI,
numberofrecordsN
Step 1: procedurePSNM(D, K, W, I, N)
Step 2: pSize ← calcPartitionSize(D)
Step 3: pNum ← [N/pSize − W + 1)]
Step 4: arrayordersizeNasInteger
Step 5: arrayrecssizepSizeasRecord
Step 6: order ← sortProgressive
(D, K, I, pSize, pNum)
Step 7: forcurrentI ← 2 todW = Iedo
Step 8: forcurrentP ← 1 topNumdo
Step 9: recs ← loadPartition(D, currentP)
Step 10: fordistbelongstorange
(currentI, I, W) do
Step 11: fori ← 0 to |recs|_ distdo
Step 12: pair ←< $recs[i], recs[i + dist]$ >
Step 13: ifcompare(pair) then
Step 14: emit(pair)
Step 15: lookAhead(pair)
```

PSNM sorts the dataset D by key K and the sorting is done by progressive sorting algorithm. Later, PSNM linearly increases the window size from 2 to the maximum window size W in steps of one. In this manner, promising close neighbors are selected first and later on less promising far-away neighbors are selected. For every successive progressive iteration, PSNM reads the entire dataset once. As the loading process is in partition-wise, PSNM iterates and loads all partitions in sequence. In order to process a loaded partition, the PSNM first iterates overall record rank-distances dist that are within the current window interval current. For I= 1, means the distance is only one, which specifies the record rank-distance of the current main-iteration. PSNM then iterates all records in the current partition to compare them to their dist-neighbor. The compare (pair) function is employed for comparison. Ifthis function returns “true”, a duplicate has been found and can be give out.

5. PROGRESSIVE BLOCKING

Progressive blocking (PB) [6] is a novel approach that builds upon a technique called equidistant blocking and the successive enlargement of blocks. Like PSNM, it also presorts the records to use their rank-distance in this sorting for similarity estimation. This algorithm is used for large and very dirty datasets.

Itaccepts five input parameters such as dataset references D, key attribute K, maximum block range R, block size S and record number N. Initially, PB estimates the number of records per partition p Size by using a pessimistic sampling function. The algorithm also analyses the number of loadable blocks per partition b Per P, the total number of blocks bNum, and the total number of partitions pNum. PB then defines the three main data structures such as the order-array which stores the ordered list of record IDs, the blocks-array which holds the current partition of blocked records, and the b Pairs-list which stores all recently evaluated block pairs.

Therefore, a block pair is represented as a triple of (blockNr1, blockNr2, duplicates Per Comparison).The b Pairs-list is implemented as a priority queue, because the algorithm frequently reads the top elements from this list. The PB algorithm sorts the dataset using the progressive Magpie Sort Algorithm.Felix Naumann [5] deals with finding multiple records in a dataset which represent the real world entity. In this work, the author introduced an algorithm called sorted blocks which generalizes both blocking and windowing approaches. Sorted Neighbourhood method sort the data set based on some key value and compare pairs to the window size. Afterwards, then it loads all blocks partition-wise from disk to execute the comparisons within each block. After the pre-processing, the PB algorithm starts progressively extending the most promising block pairs.

```
ProgressiveBlockingRequire: datasetreferenceD,
keyattributeK, maximumblockrangeR,
blocksizeSandrecord
numberN
Step 1: procedurePB(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: arrayordersizeNasInteger
Step 7: arrayblockssizebPerPas
            $< Integer; Record[] >$
Step 8: priorityqueuebPairsas
            $< Integer; Integer; Integer >$
Step 9: bPairs ← {< 1,1, −>, . . . , $< bNum, bNum, −$
            >}
Step 10: order
← sortProgressive(D, K, S, bPerP, bPairs)
Step 11: fori ← 0 topNum − 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: whilebPairsisnotemptydo
Step 16: pBPs ← {}
Step 17: bestBPs ← takeBest([bPerP/4], bPairs, R)
Step 18: forbestBPbelongstobestBPsdo
Step 19: ifbestBP[1] _ bestBP[0] $< Rthen$
Step 20: pBPs ← pBPsUextend(bestBP)
Step 21: blocks ← loadBlocks(pBPs, S, order)
Step 22: compare(blocks, pBPs, order)
Step 23: bPairs ← bPairsUpBPs
Step 24: procedurecompare(blocks, pBPs, order)
Step 25: forpBPbelongstopBPsdo
Step 26: $< dPairs, cNum$
            $> comp(pBP, blocks, order)$
Step 27: emit(dPairs)
Step 28: pBP[2] ← |dPairs|/ cNum
```

6. ATTRIBUTE PARALLEL METHOD:

The best key for finding the duplicate is generally hard to identify. Selecting good keys will increase the progressiveness. Multi-pass execution can be applied for progressive SNM. Key separation is not needed in PB algorithm. Here all the records are taken and checked as parallel processes in order to reduce average execution time. After splitting, the entire records are kept in multiple resources. If any intermediate duplicate results are found in any resources, they are intimated immediately and returned to the main application. So the time consumption is reduced. The consumption of resource is similar to the existing system, however, the data is kept in multiple resource memories.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

7. GENETIC PROGRAMMING ALGORITHM:

In this effort, the GP evolutionary process is guided by a generational evolutionary algorithm which contributes well defined and distinct generation cycles. This approach is adopted as it captures the basic idea behind several evolutionary algorithms. The steps of this algorithm are the following:

1. To initialize the number of people (with random or user provided individuals).
2. Calculate all individuals in the present population, assigning a numeric rating or fitness value to each one.
3. If the criteria of terminationfulfilled, then execute the last step. Otherwise continue.
4. Repeat the best n individuals into the next generation population.
5. Select m individuals that will compose the next generation with the best parents.

The below architecture shows the overall performance of all the above algorithm.

8. Architecture

The below Architecture represents the flow of duplicate detection in the give data.First we will select the dataset and load all the data, then we need to preprocess all the data ie we have to change the data into sequential order and remove all the noisy content. Then in next step we will cluster all the data items with respect to their similarity, after clustering we will separate the data with respect to the values. Then after we will select the subset of the data then we start detecting all the duplicates present in the parent data and all the subsets. Finally we will get the accurate results.

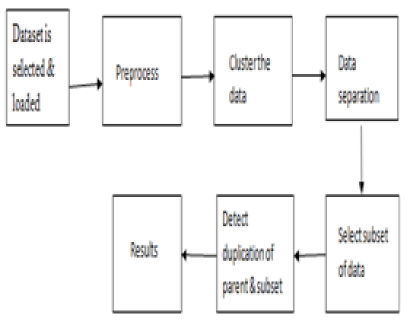

Fig 5.1 Architecture

9. PERFORMANCE EVALUATION AND RESULTS:

To evaluate the performance of this work, the errorsintroduced in duplicate records range from small type of graphical changes to large changes of some fields.Generally, the ratio of duplicate records number to the database records number is termed as the database duplicate ratio. To analyze the efficiency of this work,proposed approach is applied on a selected data. In this research work, the time taken for detecting duplicate data along with analysis of elimination process is carried out to find the efficiency while saving time.

Experiment:

In this process, sample data is selected and loaded into to perform duplicate detection. Preprocessing is applied to the selected data to remove any missing values and clean the data. We should form clusters of data depending on the categories. Data is separated in subsets and a subset of data is selected to detect replica of data. We here compare the results of Efficiency and earlier and proposed algorithms.

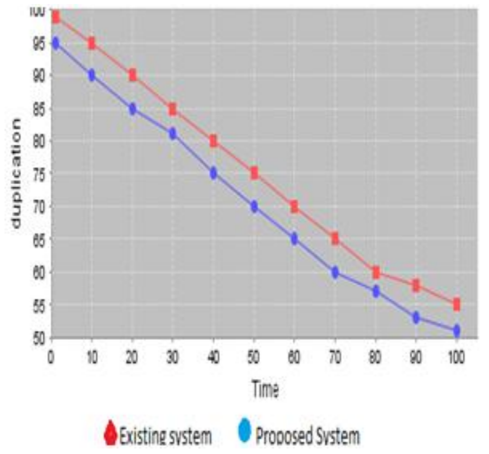

Fig 6.1 Comparing Duplication & Time

10. CONCLUSION:

In this paper, both progressive sorted neighborhood method and progressive blocking were discussed. Both algorithms can handle the situations with limited execution time and increase the efficiency of duplicate detection.

Also, they dynamically change the ranking of comparison based on intermediate results to perform high promising comparisons first and less promising comparisons later. Further, to estimate the performance gain of our algorithms, a quality measure of novel is proposed for progressiveness that integrates effortlessly with existing measures.

11. REFERENCES

[1]L. Kolb, A. Thor, and E. Rahm, “Parallel sorted neighbourhood blocking with mapreduce,” in Proceedings of the Conference Daten bank systeme in Büro, Technik und Wissenschaft (BTW), 2011

[2]P. Christen, “A survey of indexing techniques for scalable record linkage and deduplication,” IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 24, no. 9, 2012.

[3]U. Draisbach and F. Naumann,“A generalization of blocking and windowing algorithms for duplicate detection,”in Proc. Int. Conf.DataKnowl. Eng., 2011, pp. 18-24.

[4]O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller, “Framework for evaluating clustering algorithms in duplicate detection,” in Proceedings of the International Conference on Very Large Databases (VLDB), 2009

[5]Papenbrock T., Heise A. and Naumann F. (2015),’’Progressive duplicate detection‟, Proc. IEEE Trans. Know. Data Eng., vol. 27, No. 5, pp. 1316-1329.

[6]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.

[5]U. Draisbach and F. Naumann, “A generalization of blocking and windowing algorithms for duplicate detection,” in Proc. Int. Conf.DataKnowl. Eng., 2011, pp. 18–24.

[6]R. Jeffery, M. J. Franklin, and A. Y. Halevy, “Pay-as-you-go user feedback for dataspace systems,” in Proceedings of the International Conference on Management of Data (SIGMOD), 2008.