Volume: 4 Issue: XI Month of publication: November 2016
DOI:

www.ijraset.com Volume 4 Issue XI, November 2016
IC Value: 13.98 ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Scalable approach to detect the duplicate data using Iterative parallel sorted neighbourhood method

Dr. R. Priya[1], Ms. Jiji. R[2]
[1]Head, Dept. of Computer Science, Sree Narayana Guru College, Coimbatore
[2]Research Scholar, Sree Narayana Guru College, Coimbatore

Abstract: Determining the redundant data in the data server is open research in the data intensive application. Traditional Progressive duplicate detection algorithms namely progressive sorted neighbourhood method (PSNM) with scalable approaches named as Parallel sorted neighbourhood Method, which performs best on small and almost clean datasets, and progressive blocking (PB), which performs best on large and very dirty datasets. Both enhance the efficiency of duplicate detection even on very large datasets; In this paper , we propose Iterative Progressive Sorted Neighbourhood method which is treated as progressive duplicate record detection in order to detect the duplicate records in any kind of the dataset. In comparison to traditional duplicate detection, progressive duplicate record detection satisfies two conditions through improved early quality. Iterative algorithms on PSNM and PB dynamically adjust their behaviour by automatically choosing optimal parameters, e.g., window sizes, block sizes, and sorting keys, rendering their manual specification superfluous. In this way, we significantly ease the parameterization complexity for duplicate detection in general and contribute to the development of more user interactive applications: We can offer fast feedback and alleviate the often difficult parameterization of the algorithms. The contrition of the work is as follows, we propose three dynamic progressive duplicate detection algorithms, PSNM, Iterative PSNM parallel and PB, which expose different strengths and outperform current approaches. We define a novel quality measure for progressive duplicate detection to objectively rank the performance of different approaches. The Duplicate detection algorithm is evaluated on several real-world datasets testing our own and previous algorithms. The duplicate detection workflow comprises the three steps pair-selection, pair-wise comparison, and clustering. For a progressive workflow, only the first and last step needs to be modified. The Experimental results prove that proposed system outperforms the state of arts approaches accuracy and efficiency.

Keywords – Duplicate Detection, Record linkage, Data Cleansing, Deduplication

I. INTRODUCTION

Duplicate detection is the process of identifying multiple representations of same real world entities. Progressive duplicate detection identifies most duplicate pairs early in the detection process[1]. Instead of reducing the overall time needed to finish the entire process, progressive approaches try to reduce the average time after which a duplicate is found. Early terminations, in particular, then yields more complete results on a progressive algorithm than on any traditional approach[2][3][4]. Today, duplicate detection methods need to process ever larger datasets in ever shorter time: maintaining the quality of a dataset becomes increasingly difficult. We present two novel, progressive duplicate detection algorithms that significantly increase the efficiency of finding duplicates if the execution time is limited: They maximize the gain of the overall process within the time available by reporting most results much earlier than traditional approaches. Iterative algorithms on PSNM [5] and PB [6] dynamically adjust their behaviour by automatically choosing optimal parameters, e.g., window sizes, block sizes, and sorting keys, rendering their manual specification superfluous. In this way, we significantly ease the parameterization complexity for duplicate detection in general and contribute to the development of more user interactive applications: We can offer fast feedback and alleviate the often difficult parameterization of the algorithms. The contrition of the work is as follows, we propose three dynamic progressive duplicate detection algorithms, PSNM, Iterative PSNM parallel and PB, which expose different strengths and outperform current approaches [7]. We define a novel quality measure for progressive duplicate detection to objectively rank the performance of different approaches. We exhaustively evaluate on several real-world datasets testing our own and previous algorithms. The duplicate detection workflow comprises the three steps pair-selection, pair-wise comparison, and clustering. For a progressive workflow, only the first and last step needs to be modified. The rest of paper is organized as follows, Section 2 describes the related work, section 3 describes the proposed model in detail, section 4 deals with experimental analysis and finally section 5 is concluded.

II. RELATED WORK

A. *Record linkage: Making maximum use of the discriminating power of identifying Information*

Special difficulties are encountered in devising reliable systems for searching and updating any large files of documents that must be identified primarily on the basis of names and other personal particulars. The underlying problem is that of making nearly maximum use of items of identifying information that are individually unreliable but that may collectively be of considerable discriminating power [8]. Rules that can be applied generally to name retrieval systems have been developed in a methodological study of the linkage of vital and health records into family groupings for demographic research purposes. These rules are described, and the ways in which information utilization for matching may be optimized are discussed.

B. *Parallel sorted neighbourhood blocking with MapReduce*

It enables the efficient parallel execution of data-intensive tasks such as entity resolution on large datasets. We investigate challenges and possible solutions of using the MapReduce programming model for parallel entity resolution. In particular, we extract and evaluate two MapReduce-based implementations for Sorted Neighborhood blocking that either use multiple MapReduce jobs or apply tailored data replication [9].

III. PROPOSED MODEL

A. *Data partitioning*

In this Module, we partition the large record into different random cluster in order to achieve the reduced time taken for the data processing and early detection of the duplicates inside the cluster. Sampling Method with the threshold limit (size) is specified in accessing the data with the data type from the database.

1) *Progressive sorted neighbourhood method:* PSNM sorts the input data using a predefined sorting key and only compares records that are within a window of records in the sorted order. The intuition is that records that are close in the sorted order are more likely to be duplicates than records that are far apart, because they are already similar with respect to their sorting key. More specifically, the distance of two records in their sort ranks (rank-distance) gives PSNM an estimate of their matching likelihood. The PSNM algorithm uses this intuition to iteratively vary the window size [10], starting with a small window of size two that quickly finds the most promising records. The PSNM algorithm differs by dynamically changing the execution order of the comparisons based on intermediate results (Look-Ahead). Furthermore, PSNM integrates a progressive sorting phase (Magpie Sort) and can progressively process significantly larger datasets.

Algorithm
Parameters considered in the Analysing multiple representation for data

a) Reference Key D is a reference to the data, which has not been loaded from disk yet.
b) The sorting key K defines the attribute or attributes combination that should be used in the sorting step.
c) Window key W specifies the maximum window size, when using early termination, this parameter can be set to an optimistically high default value.
d) Key I define the enlargement interval for the progressive iterations.
e) Partitioning Condition
f) The number of necessary partitions pNum, while considering a partition overlap records to slide the window across their boundaries.
g) Record should be ordered based on the sorting key using progressive Sorting function**.**

B. *Duplicates Detection using Iterative Parallel sorted neighbourhood method*

Duplicates are detected based on the window size by gradually increasing to estimate the close neighbours and less far way neighbours later on For each of these progressive iterations[11], PSNM reads the entire dataset once. Since the load process is done partition-wise, PSNM sequentially iterates and loads all partitions.

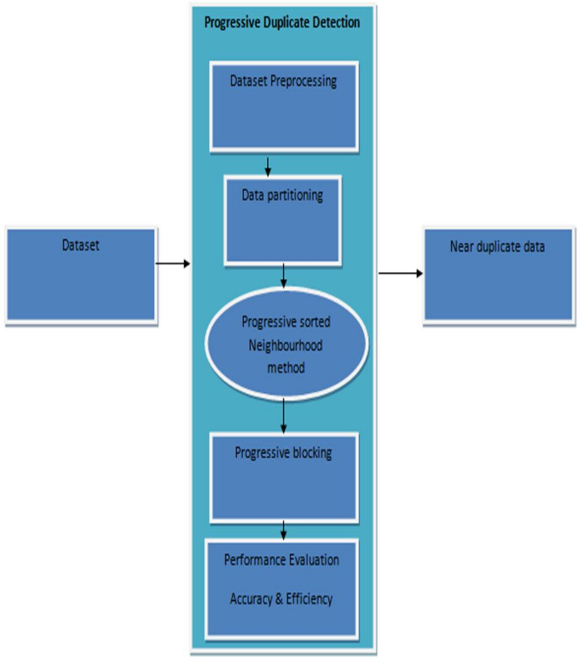

Figure 1. Architecture Diagram of the proposed model

To process a loaded partition, PSNM first iterates overall record rank-distances dist that are within the current window interval , PSNM then iterates all records in the current partition to compare them to their dist-neighbor. The comparison is executed using the compare (pair) function.

C. progressive blocking

Blocking algorithms assign each record to a fixed group of similar records (the blocks) and then compare all pairs of records within these groups. Progressive blocking is a novel approach that builds upon an equidistant blocking technique and the successive enlargement of blocks. Like PSNM, it also pre-sorts the records to use their rank-distance in this sorting for similarity estimation. Based on the sorting, PB first creates and then progressively extends a fine-grained blocking. These block extensions are specifically executed on neighbourhoods around already identified duplicates, which enables PB to expose clusters earlier than PSNM.

Block Comparison matrix is created to sort the records that form a similarity index is considered to be duplicate. In case of ties, the algorithm prefers block pairs with a smaller rank-distance[12], because the distance in the sort rank still defines the expected similarity of the records. The extensions continue until all blocks have been compared or a distance threshold for all remaining block pairs has been reached.

We modelled a priority queue to frequently read the top elements from this list to estimate the density of duplicate items which exceeds the maximum block range. The identified duplicate later rank the duplicate density of this block pair with the density in other block pairs. Thereby, the amount of duplicates is normalized by the number of comparisons, because the last block is usually smaller than all other blocks. If the PB algorithm is not terminated prematurely, it automatically finishes when the list of similar Pairs is empty,

D. Blocking Techniques

A block pair consisting of two small blocks defines only few comparisons. Using such small blocks, the PB algorithm carefully selects the most promising comparisons and avoids many less promising comparisons from a wider neighbourhood[13]. However, block pairs based on small blocks cannot characterize the duplicate density in their neighbourhood well, because they represent a too small sample. A block pair consisting of large blocks, in contrast, may define too many, less promising comparisons, but produce better samples for the extension step. The block size parameter S, therefore, trades off the execution of non-promising comparisons and the extension quality.

1) *Attribute Concurrency:* The best sorting or blocking key for a duplicate detection algorithm is generally unknown or hard to find. Most duplicate detection frameworks tackle this key selection problem by applying the multi-pass execution method. This method executes the duplicate detection algorithm multiple times using different keys in each pass. However, the execution order among the different keys is arbitrary[14]. Therefore, favouring good keys over poorer keys already increases the progressiveness of the multi-pass method. In this section, we present two multi-pass algorithms that dynamically interleave the different passes based on intermediate results to execute promising iterations earlier. The first algorithm is the attribute concurrent Iterative PSNM (AC-IPSNM), which is the progressive implementation of the multi-pass method for the IPSNM algorithm, and the second algorithm is the attribute concurrent PB (AC-PB), which is the corresponding implementation for the PB algorithm

IV. EXPERIMENTAL RESULTS

A. *Dataset Collection and Data Parsing*

The DBLP dataset is used in this work. The DBLP-dataset2 is a bibliographic index on computer science journals and proceedings which is represented as XML File. Data parsing is applied for breaking them up into parts of data (for example, the nouns (objects), verbs (methods), and their attributes or options) which can then be managed into features and clusters.

B. *Performance Evaluation*

The proposed algorithms increase the efficiency of duplicate detection for any size of records in dataset with limited execution time;

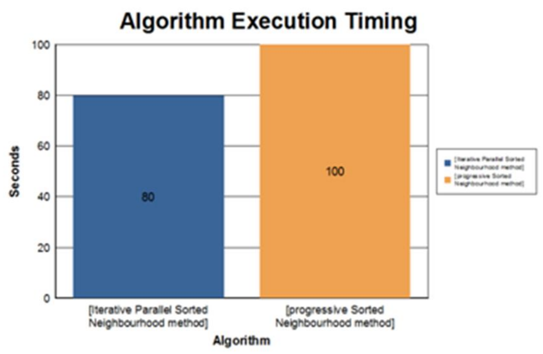

Figure 2- Performance Evaluation on Execution Time of the Progressive Duplicate Detection Methods

The system is capable for dynamic change and the ranking of duplicates estimated using the accuracy parameters such as precision, recall and f-measure with the existing solution treated as progressive sorted neighbourhood method. The comparison based on intermediate results to execute different passes of any order.

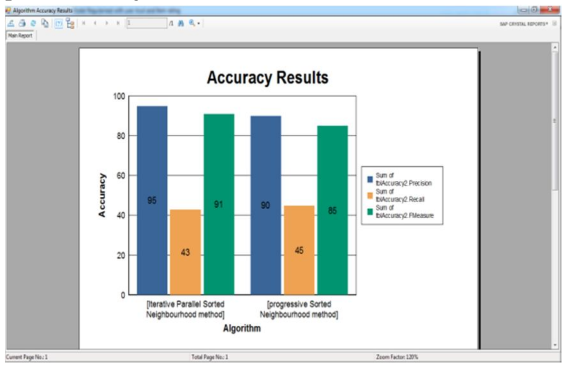
Accuracy Results
Accuracy
100
80
60
40
20
0
95
43
91
90
45
85
[Iterative Parallel Sorted Neighbourhood method]
[progressive Sorted Neighbourhood method]
Algorithm
Sum of tblAccuracy2.Precision
Sum of tblAccuracy2.Recall
Sum of tblAccuracy2.FMeasure

Figure 3- Performance Evaluation on accuracy parameter of the Progressive duplicate Detection Methods

Using this measure, experiments showed that proposed iterative approaches outperform the traditional SNM by up to 100 percent and related work by up to 30 percent.

Table 1: Performance Evaluation of progressive duplicate detection algorithms

Technique	Execution Time	Precision	Recall	F-Measure	Duplicates detected
Progressive Sorted neighbourhood method	100ms per dataset	90	45	85	120 Records per milli seconds
Iterative parallel Sorted Neighbourhood Method	140ms per dataset	95	43	91	

For the construction of a fully progressive duplicate detection workflow, we proposed an Iterative parallel Sorted neighbourhood method and results is depicted
in the figure 2 and table 1.
The adaptations AC-Iterative PSNM and AC-PB use multiple sort keys concurrently to interleave their progressive iterations. By analyzing intermediate results, both approaches dynamically rank the different sort keys at runtime, drastically easing the key selection problem.

V. CONCLUSION

Iterative Parallel sorted neighbourhood method and progressive blocking has been designed and implemented. Both algorithms increase the efficiency of duplicate detection for situations with limited execution time and high accuracy. They dynamically change the ranking of comparison candidates based on intermediate results to execute promising comparisons first and less promising comparisons later. To determine the performance gain of our algorithms, we proposed a novel quality measure for progressiveness that integrates seamlessly with existing measures. Using this measure, experiments showed that our approaches outperform the traditional SNM by up to 100 percent and related work by up to 30 percent. For the construction of a fully progressive duplicate detection workflow, we proposed a progressive sorting method, Magpie, a progressive multi-pass execution model, Attribute Concurrency, and an incremental transitive closure algorithm. The adaptations AC-Iterative PSNM and AC-PB use multiple sort keys concurrently to interleave their progressive iterations. By analyzing intermediate results, both approaches dynamically rank the different sort keys at runtime, drastically easing the key selection problem.

REFERENCES

[1] 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.
[2] 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.
[3] F. Naumann and M. Herschel, An Introduction to Duplicate Detection. San Rafael, CA, USA: Morgan & Claypool, 2010.
[4] 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.
[5] M. A. Hern_andez 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.
[6] X. Dong, A. Halevy, and J. Madhavan, “Reference reconciliation in complex information spaces,” in Proc. Int. Conf. Manage. Data, 2005, pp. 85–96.
[7] 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.
[8] O. Hassanzadeh and R. J. Miller, “Creating probabilistic databases from duplicated data,” VLDB J., vol. 18, no. 5, pp. 1141–1166, 2009.
[9] U. Draisbach, F. Naumann, S. Szott, and O. Wonneberg, “Adaptive windows for duplicate detection,” in Proc. IEEE 28th Int. Conf. Data Eng., 2012, pp. 1073–1083.
[10] S. Yan, D. Lee, M.-Y. Kan, and L. C. Giles, “Adaptive sorted neighborhood methods for efficient record linkage,” in Proc. 7th ACM/ IEEE Joint Int. Conf. Digit. Libraries, 2007, pp. 185–194.
[11] J. Madhavan, S. R. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, and A. Halevy, “Web-scale data integration: You can only afford to pay as you go,” in Proc. Conf. Innovative Data Syst. Res., 2007.
[12] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy, “Pay-as-you-go user feedback for dataspace systems,” in Proc. Int. Conf. Manage. Data, 2008, pp. 847–860.
[13] C. Xiao, W. Wang, X. Lin, and H. Shang, “Top-k set similarity joins,” in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 916–927.
[14] P. Indyk, “A small approximately min-wise independent family of hash functions,” in Proc. 10th Annu. ACM-SIAM Symp. Discrete Algorithms, 1999, pp. 454–456.