ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

Duplicate Detection by Progressive Techniques

Rohini T, Kavya D

Assistant Professor, Dept. of CSE., New Horizon College of Engineering Bengaluru, India

P.G. Student, Dept. of CSE., New Horizon College of Engineering Bengaluru, India

ABSTRACT: In today's business world the dataset plays a very important role. With the increase in the industries even there is an increase in the dataset so this will lead to the data duplication. Here we come up with solution called the data duplicate detection i.e. By using the technique called progressive duplicate detection. We present two novel, progressive duplicate detection algorithms which significantly increases the efficiency of finding the duplicate data when the execution time is limited. Here we get the quality data without any disturbance to the datasets. Duplicate detection is the process of removing replica in the repository. These algorithm dynamically adjusts behaviour by choosing parameters e.g. window size, block size etc. Comprehensive experiments show that our progressive algorithms can double the efficiency over time of traditional duplicate detection and significantly improve upon related work.

KEYWORDS: Data Duplicity Detection; PSNM; PSB; Entity resolution; Data cleaning;

I. INTRODUCTION

The data duplication is one of the critical issues in the data mining. Many industries will look for the accurate data to carry out their operations. Therefore the data quality must be significant. With the increase in the volume of data even the data quality problems arise. Multiple, yet different of the same real-world objects in data, duplicates, are one of the most intriguing data quality problems. Several representations generally are not same and have certain differences like misspelling, missing values, changed addresses, etc. which makes the detection of duplicates very difficult. The detection of duplicates is very costly because the comparison among all possible duplicate pairs is required. For example in particular Online retailers, offer huge catalogues comprising a constantly growing set of items from many different suppliers. As independent persons change the product portfolio, duplicates arise. While there is an obvious need for duplication, online shops without downtime cannot give traditional duplication.

Data has to be in integrity, if it exceeds the criteria, it is a duplicate. But due to data changes and sloppy data entry, errors such as duplicate entries might occur, making data cleaning and in particular duplicate detection indispensable A user has little knowledge about the given data but still needs to configure the cleansing process. When user has only limited, maybe unknown time for data cleansing and wants to make best possible use of it. Then, simply start the algorithm and terminate it when needed. The result size will be maximized.

In this work, however, we focus on progressive algorithms, which try to report most matches early on, while the possibility of slight increase in their overall runtime. To achieve this, they need to estimate the similarity of all comparison candidates in order to compare most promising record pairs first. Progressive duplicate detection identifies most duplicate pairs early in the detection process. 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. Then early termination, in particular, then yields more completes results on a progressive algorithm than on any traditional approach.

We propose two novel progressive duplicate detection algorithms namely:

1) Progressive sorted neighborhood method (PSNM), which performs best on small and almost clean datasets.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

2) Progressive blocking (PB), which performs best on large and very dirty datasets.

Both enhance the efficiency of duplicate detection even on very large datasets. As we propose two dynamic progressive duplicate detection algorithms, PSNM and PB, which expose different strengths and outperform current approaches.

Some of the advantages are mentioned below:

1) Improved early quality
2) Same eventual quality
3) Our algorithms PSNM and PB dynamically adjust their behavior 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.

II. RELATED WORK

There are many researches which are carried out on the duplicate detection [1],[2] also known as entity resolution. But the most prominent algorithms are the progressive blocking[3] and then the progressive Sorted Neighbourhood Method [4].

The problem of merging multiple databases of information about common entities are frequently encountered in KDD [6] and decision support applications in large commercial and government organizations. The problem we study is often called the Merge/Purge problem and is difficult to solve both in scale and accuracy. Large repositories of data typically have numerous duplicate information entries about the same entities that are difficult to cull together without an intelligent “equation theory” that identifies equivalent items by a complex, domain-dependent matching process. We have developed a system for accomplishing this Data Cleansing task and demonstrate its use for cleansing lists of names of potential customers in a direct marketing-type application. Our results for statistically generated data are shown to be accurate and effective when processing the data multiple times using different keys for sorting on each successive pass. Combing results of individual passes using transitive closure over the independent results, produces far more accurate results at lower cost. The system provides a rule programming module that is easy to program and quite good at finding duplicates especially in an environment with massive amounts of data. This paper details improvements in our system, and reports on the successful implementation for a real-world database that conclusively validates ourresults previously achieved for statistically generated data.

We explore a pay-as-you-go approach to entity resolution,[7] where we obtain partial results “gradually”as we perform resolution, so we can at least get some results faster. As we will see, the partial results may not identify all the records that correspond to the same real-world entity. Our goal will be to obtain as much of the overall result as possible, as quickly as possible. Entity resolution (ER) is the problem of identifying which records in a database refer to the same entity. In practice, many applications need to resolve large data sets efficiently, but do not require the ER result to be exact. For an example, people data from the web may simply be too large to completely resolve with a reasonable amount of work. As another example, real-time applications may not be able to tolerate any ER processing that takes longer than a certain amount of time. This paper investigates how we can maximize the progress of ER with a limited amount of work using “hints,” which give information on records that are likely to refer to the same real-world entity. A hint can be represented in various formats (e.g., a grouping of records based on their likelihood of matching), and ER can use this information as a guideline for which records to compare first. We introduce a family of techniques for constructing hints efficiently and techniques for using the hints to maximize the number of matching records identified using a limited amount of work. Using real data sets, we illustrate the potential gains of our pay-as-you-go approach compared to running ER without using hints. We have proposed a pay-as-you-go approach for ER where given a limit in resources (e.g., work, runtime) we attempt to make the maximum progress possible. We introduce the novel concept of hints, which can guide an ER algorithm to focus on resolving the more likely matching records first. Our techniques are effective when there are either too many records to resolve within a reasonable amount of time or

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

when there is a time limit (e.g., real-time systems). We proposed three types of hints that are compatible with different ER algorithms: a sorted list of record pairs, a hierarchy of record partitions, and an ordered list of records. We have also proposed various methods for ER algorithms to use these hints. Our experimental results evaluated the overhead of constructing hints as well as theruntime benefits for using hints. We considered a variety ofER algorithms and two real-world data sets. The results suggest that the benefits of using hints can be well worth the overhead required for constructing and using hints. We believe our work is one of the first to define pay-as-you-go ER and explicitly propose hints as a general technique for fast ER.

[8]Efficient duplicate detection is an important task especially in large datasets. In this paper, they have compared two important approaches, blocking and windowing, for reducing the number of comparisons. Additionally, we have introduced Sorted Blocks which is generalization of blocking and windowing. Experiments with several real-world datasets show that Sorted Blocks outperforms the two other approaches. A challenge for Sorted Blocks is finding the right configuration settings, as it has more parameters than the other two approaches. An advantage of Sorted Blocks in comparison to the Sorted Neighbourhood Method is the variable partition size instead of a fixed size window. This allows more comparisons if several records have similar values, but requires fewer comparisonsif only a few records are similar. In the future, one of our research topics will be to evaluate strategies that group records with a high chance of being duplicates in the same partitions.

Thorsten Papenbrock, ArvidHeise, and Felix Naumann[5] Both of the algorithms i.e. progressive blocking and progressive sorted neighbourhood method increase the efficiency of duplicate detection for situations with limited execution time; 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. Sorted neighbourhood method sort the data set based on some key value and compare pairs within the window size. Blockingalgorithm partition a set of record using blocking key into disjoint set. The limited records are found in same partition. By doing this the overall number of comparisons is reduced. The multi-pass method and transitive closure are used in blocking method. In windowing method, there are three phase. The first phase is to assign a sorting key to each record. Next phase is to sort the record based on key value. The final phase is to assume fixed window size and compare all pairs of records appear in the window. The multi-pass method performs the sorting and windowing approaches multiple times to avoid mis-sort due to error in the attributes. One of the advantages of using sorted block in comparing with sorted neighbourhood method is the variable partition instead of a fixed size window.

III. PROPOSED ALGORITHM

The proposed solution uses two types of novel algorithms for progressive duplicate detection, which are as follows:

- PSNM – It is known as Progressive sorted neighborhood method and it is performed over clean and small datasets.
- PB – It is known as Progressive blocking and it is performed over dirty and large datasets. Both these algorithms improve the efficiencies over huge datasets.

There are three stages in this workflow which are as follows:

a)Pair selection
b)Pair wise comparison
c) Clustering
Only the pair selection and clustering stages should be modified for a good workflow.

A. PROGRESSIVE SORTED NEIGHBORHOOD METHOD:

The Progressive Sorted Neighborhood Method (PSNM) depends on the conventional Sorted Neighborhood Method: PSNM sorts the information utilizing a predefined sorting key and just thinks about records that are inside the window of records in the sorted request. The instinct is that records that are close in the sorted request will probably be copies than records that are far separated, on the grounds that they are as of now comparative as for their sorting key. All the more specifically, the separation of two records in their short positions (rank-separation) gives PSNM an

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

evaluation of their coordinating probability. The PSNM calculation utilizes this instinct to iteratively change the window size, beginning with a little window of size two that rapidly finds the most encouraging records. This static methodology has, as of now been proposed as the Sorted List of Record Pairs clue , The PSNM calculation varies by powerfully changing the execution request of the correlations taking into account middle results (Look-Ahead).Furthermore, PSNM integrates progressive sorting stage (Magpie Sort) and can logically prepare significantly bigger datasets.

B. PROGRESSIVE BLOCKING:

Dynamic Blocking (PB) is a novel approach that expands upon an equidistant blocking system and the progressive growth of pieces. Like PSNM, it likewise pre-sorts the records to utilize their rank-separation in this sorting for likeness estimation. In light of the sorting, PB first makes and after that dynamically expands a fine-grained blocking. These square expansions are specifically executed on neighborhoods around as of now identified copies, which empower PB to uncover, bunch sooner than PSNM.

IV. PSEUDO CODE

A. PROGRESSIVE SORTED NEIGHBORHOOD METHOD

The algorithm takes five input parameters: D is a reference to the data, which has not been loaded from disk yet. Thesorting 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 neighbourhood 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 Neighbourhood 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 ←sort Progressive (D, K, I, pSize, pNum)
Step 7: for currentI← 2 todW=Iedo
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: look Ahead (pair)

B. PROGRESSIVE BLOCKING

The algorithm accepts five input parameters: The dataset reference D specifies the dataset to be cleaned and the keyattribute 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 ofthe input dataset. Progressive Blocking Require: dataset reference D, key attribute K, maximum block range R, block size S and record number N.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

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 belongs to 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 belongs to pBPs do
Step 26: <dPairs, cNum> comp(pBP, blocks, order)
Step 27: emit(dPairs)
Step 28: pBP[2] ←|dPairs|/ cNum

V. SIMULATION RESULTS

PSNM executes the same comparisons because the natural SNM procedure, the algorithm takes longer to finish. The rationale for this commentary is the increased number of totally pricey load strategies. To cut down their complexity, PSNM implements partition caching. We now evaluate the average SNM algorithm, a PSNM algorithm without partition caching and a PSNM algorithm with partition caching on the DBLP-dataset. The results of this experiment are shown in determine four in the left graph. The scan suggests that the advantage of partition caching is big: The runtime of PSNM decreases by using 42% minimizing the runtime change between PSNM and SNM to just 2%.

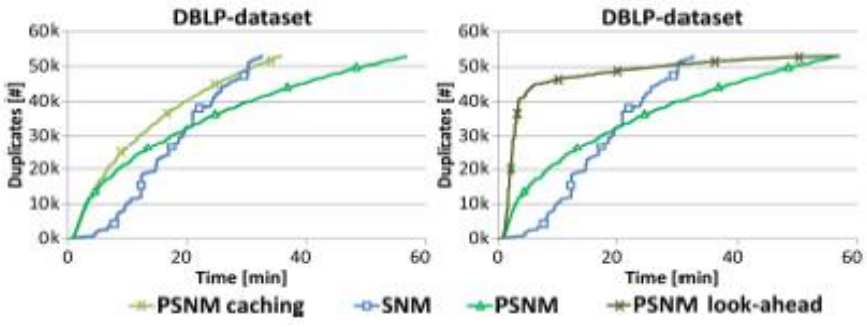

Fig 1. Effect of partition caching and look-ahead

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

VI. CONCLUSION AND FUTURE WORK

In this paper we have gone through the progressive sorted neighbourhood method and then the progressive blocking. Both the algorithms will adjust automatically based on the parameter .These both algorithms increase the efficiency of duplicate detection for situations with limited execution time and high accuracy. In future work, we want to combine these progressive approaches with scalable approaches for the duplicate detection in order to deliver the result even faster. The parallel sorted neighbourhood can be executed to find in parallel.

REFERENCES

[1] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 19, page no. 1, 2007
[2] F. Naumann and M. Herschel, An Introduction to Duplicate Detection. Morgan & Claypool, 2010.
[3] H. B. Newcombe and J. M. Kennedy, "Record linkage: making maximum use of the discriminating power of identifying information," Communications of the ACM, vol. 5, page no. 11, 1962.
[4] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," Data Mining and Knowledge Discovery, vol. 2,page no. 1, 1998.
[5] Thorsten Papenbrock, Arvid Heise, and Felix Naumann "Progressive Duplicate Detection" IEEE Transactions on Knowledge and Data Engineering DOI 10.1109/TKDE.2014.2359666.
[6]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.
[7] U. Draisbach and F. Naumann, "A generalization of blocking and windowing algorithms for duplicate detection," in Proc. Int. Conf. Data Knowl. Eng., pp. 18–24, 2011
[8] Steven Euijong Whang "Pay-As-You-Go Entity Resolution" IEEE transactions on knowledge and data engineering, vol. 25,page no. 5, may 2011

BIOGRAPHY

Ms Rohini T. Assistant professor, Dept. of Computer Science and Engineering in New Horizon College of Engineering, which is located in Outer Ring Road, Panathur Post, Kadubisanahalli,Bangalore – 560087.

Ms Kavya D. Pursuing M Tech. Computer Science and Engineering in New Horizon College of Engineering, which is located in Outer Ring Road, Panathur Post, Kadubisanahalli, Bangalore – 560087.