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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

Efficient and Effective Duplicate Detection Evaluating Multiple Data using Genetic Algorithm

Dr.M.Mayilvaganan, M.Saipriyanka

Associate Professor, Dept. of Computer Science, PSG College of Arts and Science, Coimbatore, India

Research Scholar, Dept. of Computer Science, PSG College of Arts and Science, Coimbatore, India

ABSTRACT: With the ever increasing volume of data, data quality problems abound. Multiple, yet different representations of the same real-world objects in data, duplicates, are one of the most intriguing data quality problems. The effects of such duplicates are detrimental. For instance, bank customers can obtain duplicate identities, inventory levels are monitored incorrectly, catalogs are mailed multiple times to the same household, etc. Automatically detecting duplicates is difficult. Duplicate detection is the process for identifying multiple representations of same real world entities. Nowadays, duplicate detection methods need to process ever larger datasets in ever shorter time: maintaining the quality of a dataset becomes increasingly difficult. Genetic algorithm is proposed that significantly increase the efficiency of finding duplicates if the execution time is limited. This efficiently detects the text document duplication which has same content with distinct file name or different content with same file name.

KEYWORDS: Clustering Algorithm, Genetic Algorithm, progressive SNM, progressive blocks.

I. INTRODUCTION

Data are among the most important assets of a company. But due to data changes and sloppy data entry, errors such as duplicate entries might occur, making data cleansing and in particular duplicate detection indispensable. However, the pure size of today's datasets render duplicate detection processes expensive. Online retailers, for example, offer huge catalogs comprising a constantly growing set of items from many different suppliers. As independent persons change the product portfolio, duplicates arise. Although there is an obvious need for de duplication, online shops without downtime cannot afford traditional de duplication. 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. Early termination, in particular, then yields more complete results on a progressive algorithm than on any traditional approach.

II. RELATED WORK

Databases play an important role in today's IT-based economy. Many industries and systems depend on the accuracy of databases to carry out operations. Therefore, the quality of the information (or the lack thereof) stored in the databases can have significant cost implications to a system that relies on information to function and conduct business. Much research on duplicate detection, also known as entity resolution and by many other names focuses on pair selection algorithms that try to maximize recall on the one hand and efficiency on the other hand. Adaptive techniques are capable of estimating the quality of comparison candidates. The algorithms use this information to choose the comparison candidates more carefully. In the last few years, the economic need for progressive algorithms also initiated some concrete studies in this domain. For instance, pay-as-you-go algorithms for information integration on large scale datasets have been presented. Other works introduced progressive data cleansing algorithms for the analysis of sensor data streams. However, these approaches cannot be applied to duplicate detection.

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

Disadvantages

- These adaptive techniques dynamically improve the efficiency of duplicate detection, but in contrast to our progressive techniques, they need to run for certain periods of time and cannot maximize the efficiency for any given time slot.
- Needs to process large dataset in short time
- Quality of data set becomes increasingly difficult

III. PROPOSED SYSTEM

In an error-free system with perfectly clean data, the construction of a comprehensive view of the data consists of linking—in relational terms, joining—two or more tables on their key fields. Unfortunately, data often lack a unique, global identifier that would permit such an operation. Furthermore, the data are neither carefully controlled for quality nor defined in a consistent way across different data sources. Thus, data quality is often compromised by many factors. In this proposed system two novel, progressive duplicate detection algorithms namely progressive sorted neighborhood method (PSNM), 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 project genetic programming algorithm is used to detect the duplication of text document. Text document which has same content with different name is detected and saved as duplicate. If any document has same name with different content is saved without overwrite.

Advantages

- **Improved early quality**

Let t be an arbitrary target time at which results are needed. Then the progressive algorithm discovers more duplicate pairs at t than the corresponding traditional algorithm. Typically, t is smaller than the overall runtime of the traditional algorithm.

- **Same eventual quality**

If both a traditional algorithm and its progressive version finish execution, without early termination at t, they produce the same results.

IV. PROPOSED ALGORITHM

PROGRESSIVE SNM

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 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

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

Step 12: pair←<recs[i], recs[i + dist]>
Step 13: if compare(pair) then
Step 14: emit(pair)
Step 15: lookAhead(pair)

2. 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 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

SGENETIC PROGRAMMING ALGORITHM

In this work, the GP evolutionary process is guided by a generational evolutionary algorithm. This means that there are well defined and distinct generation cycles. We adopted this approach since it captures the basic idea behind several evolutionary algorithms.

The steps of this algorithm are the following:
1. Initialize the population (with random or user provided individuals).
2. Evaluate all individuals in the present population, assigning a numeric rating or fitness value to each one.
3. If the termination criterion is fulfilled, then execute the last step. Otherwise continue.
4. Reproduce the best n individuals into the next generation population.
5. Select m individuals that will compose the next generation with the best parents.
6. Apply the genetic operations to all individuals selected. Their offspring will compose the next population. Replace the existing generation by the generated population and go back to Step 2.

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

7. Present the best individual(s) in the population as the output of the evolutionary process.

The evaluation at Step 2 is done by assigning to an individual a value that measures how suitable that individual is to the proposed problem. In our GP experimental environment, individuals are evaluated on how well they learn to predict good answers to a given problem, using the set of functions and terminals available.

The resulting value is also called raw fitness and the evaluation functions are called fitness functions.
Notice that after the evaluation step, each solution has a fitness value that measures how good or bad it is to the given problem. Thus, by using this value, at Step 5 it is possible to select which individuals should be in the next generation. Strategies for this selection may involve very simple or complex techniques, varying from just selecting the best n individuals to randomly selecting the individuals proportionally to their fitness.

Reference Reconciliation in Complex Information Spaces

Reference reconciliation is the problem of identifying when di®erent references (i.e., sets of attribute values) in a dataset correspond to the same real-world entity. Most previous literature assumed references to a single class that had a fair number of attributes (e.g., research publications). We consider complex information spaces: our references belong to multiple related classes and each reference may have very few attribute values. A prime example of such a space is Personal Information Management, where the goal is to provide a coherent view of all the information on one's desktop. Our reconciliation algorithm has three principal features. First, we exploit the associations between references to design new methods for reference comparison. Second, we propagate information between reconciliation decisions to accumulate positive and negative evidences. Third, we gradually enrich references by merging attribute values. Our experiments show that (1) we considerably improve precision and recall over standard methods on a diverse set of personal information datasets, and (2) there are advantages to using our algorithm even on a standard citation dataset benchmark.

V. LITERATURE SURVEY

Duplicate Record Detection: A Survey

Often, in the real world, entities have two or more representations in databases. Duplicate records do not share a common key and/or they contain errors that make duplicate matching a difficult task. Errors are introduced as the result of transcription errors, incomplete information, lack of standard formats, or any combination of these factors. In this paper, we present a thorough analysis of the literature on duplicate record detection. We cover similarity metrics that are commonly used to detect similar field entries, and we present an extensive set of duplicate detection algorithms that can detect approximately duplicate records in a database. We also cover multiple techniques for improving the efficiency and scalability of approximate duplicate detection algorithms. We conclude with coverage of existing tools and with a brief discussion of the big open problems in the area. Index Terms—Duplicate detection, data cleaning, data integration, record linkage, data deduplication, instance identification, database hardening, name matching, identity uncertainty, entity resolution, fuzzy duplicate detection, entity matching.

Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem

The problem of merging multiple databases of information about common entities is frequently encountered in KDD 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 “equational 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

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

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 our results previously achieved for statistically generated data.

Pay-As-You-Go Entity Resolution

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 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.

Framework for Evaluating Clustering Algorithms in Duplicate Detection

The presence of duplicate records is a major data quality concern in large databases. To detect duplicates, entity resolution also known as duplication detection or record linkage is used as a part of the data cleaning process to identify records that potentially refer to the same real-world entity. We present the Stringer system that provides an evaluation framework for understanding what barriers remain towards the goal of truly scalable and general purpose duplication detection algorithms. In this paper, we use Stringer to evaluate the quality of the clusters (groups of otential duplicates) obtained from several unconstrained clustering algorithms used in concert with approximate join techniques. Our work is motivated by the recent significant advancements that have made approximate join algorithms highly scalable. Our extensive evaluation reveals that some clustering algorithms that have never been considered for duplicate detection, perform extremely well in terms of both accuracy and scalability.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a GP-based approach to record duplication. Our approach is able to automatically suggest duplication functions based on evidence present in the data repositories. The suggested functions properly combine the best evidence available in order to identify whether two or more distinct record entries are replicas (i.e., represent the same real-world entity) or not. This is extremely useful for the no specialized user, who does not have to worry about setting up the best set of evidence for the replica identification task.

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.
    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. F. Naumann and M. Herschel, An Introduction to Duplicate Detection. San Rafael, CA, USA: Morgan & Claypool, 2010.
3. 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.
4. 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.
5. X. Dong, A. Halevy, and J. Madhavan, "Reference reconciliation in complex information spaces," in Proc. Int. Conf. Manage. Data, 2005, pp. 85–96.
6. 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.
7. O. Hassanzadeh and R. J. Miller, "Creating probabilistic databases from duplicated data," VLDB J., vol. 18, no. 5, pp. 1141–1166, 2009.
8. 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.

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

9. 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.

BIOGRAPHY

Dr. M.Mayilvaganan is a Research Associate in the computer Science Department, PSG College of Arts and Sciences, Coimbatore, India. He received Master of Computer Applications (MCA) degree in 1990 and also completed in PhD doctorate in 2009. His research interests are Data Mining, Big Data Analysis, networking etc....

.