To Study Of Progressive Techniques For Efficient Duplicate Deduction Mechanism

R.Hamsaveni 1 M.Vinay Kumar 2, N.Nirosh Kumar 3

ABSTRACT: Databases contains very large datasets, where various duplicate records are present. The duplicate records occur when data entries are stored in a uniform manner in the database, resolving the structural heterogeneity problem. Detection of duplicate records are difficult to find and it take more execution time. In this literature survey papers various techniques used to find duplicate records in database but there are some issues in this techniques. To address these Progressive algorithms has been proposed for that significantly increases the efficiency of finding duplicates if the execution time is limited and improve the quality of records.

Keywords: Duplicate record detection, Sorted Neighborhood method, Blocking, Entity resolution.

1. 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.

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 to W+1 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)

IV. CONCLUSION

The progressive sorted neighbourhood method and progressive blocking algorithms 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 later. This paper surveys different research papers that proposed various algorithms for detection of duplicate records. The progressive algorithms of duplicate detection are used to overcome disadvantages in various research papers.

REFERENCES