Removing Fully and Partially Duplicated Records through K-Means Clustering

Bilal Khan, Azhar Rauf, Huma Javed, Shah Khusro, and Huma Javed

***Abstract*—Records duplication is one of the prominent problems in data warehouse. This problem arises when various databases are integrated. This research focuses on the identification of fully as well as partially duplicated records. In this paper we propose a de-duplicator algorithm which is based on numeric conversion of entire data. For efficiency, data mining technique k-mean clustering is applied on the numeric value that reduces the number of comparisons among records. To identify and remove the duplicated records, divide and conquer technique is used to match records within a cluster which further improves the efficiency of the algorithm.**

***Index Terms*—Data cleansing, De-Duplicator, partial duplication, K-Mean clustering.**

I. Introduction

Data warehouses store large amount of data that is used in analysis and decision making process. Data is integrated from various heterogeneous sources. In heterogeneous sources data has different formats. Data is noisy in nature [1], [2] and needs to be cleaned in data warehouse. Data cleansing is a process of detecting incorrect, redundant and missing values and then correcting them. This process also checks the format, completeness, and other business rules related errors in data.

Data cleansing process is used to improve the quality of data [3]. Some data quality problems occur because of data entry operator errors such as spellings mistakes, missing integrity constraints, missing field (e.g., date of birth used in the field of admission date), noise or contradicting entry, null values, misuse of abbreviations, and duplicated records [3]-[9]. Data quality measures the accuracy, integrity, completeness, validity, consistency and redundancy aspects of data [1], [8].

In data warehouse, data cleansing has a vital role. If the quality of data is not good, the strategic decisions taken on the basis of that data may not be good [3]. Records duplication is one of the major issues in data quality [3], [10]. It is the representation of the same real world object more than once in the same table [6], [10]-[12]. It is necessary to eliminate duplicated records in order to bring consistency and improve the quality of the data. Identification and removal of the duplicated records is an important issue in data cleansing which is the subject of this research.

This paper proposes a novel approach for detection of duplicated records by converting the field values into numeric form instead of condensing them as tokens. The proposed technique identifies and removes not only fully but also partially duplicated records. The K-mean clustering algorithm is used to reduce the number of comparisons by forming clusters and the divide and conquer approach is used to match records within the clusters.

We classified the duplicated records into three categories. a) Fully Duplicated Records, having two identical rows representing the same real world entity. b) Erroneous Duplicated Records, which in fact are duplicated records but due to the data entry operator's erroneous entry, they seem to be different. Identification of such records is a challenging process as they cannot be separated out by sorting techniques. c) Partially Duplicated Records, having partial duplication but the difference is original. Our proposed technique identifies all such kinds of records.

The rest of paper is organized as follow. We describe the related work in Section II. Section III presents the core of our approach. Experimental results are introduced in Section IV. Finally, we conclude our work in Section V.

II. Related Work

The problem of duplicated records has been extensively discussed in the literature. Bitton et al. discussed the elimination of duplicated records in large data files by sorting which brings identical records together [13]. If sorting is based on dirty fields, identical records can never get together. Sorting method is inefficient for large data files having typographical errors.

Hernandez et al. discuss the problem of merge/purge in a large database [14]. They form token keys of selected fields of the database table. Records in the table are sorted by using that key. To reduce the number of comparisons, records having same token keys are sorted and put in the same clusters [14]. The effectiveness of merge/purge approach depends on the quality of the chosen keys which may fail in bringing possible duplicated records near each other for subsequent comparison.

Character and token based techniques are used by Elmagrmid et al. for detecting record duplication [6]. Character based technique deals well with the typographical errors. But sometimes typographical conventions lead to rearrangement of words e.g. ("Bilal Khan", versus "Khan, Bilal"). Character based technique fails in order to compare such kind of strings. The token base technique is used to overcome this problem.

Token based data cleansing technique defines smart tokens that are used to identify and remove duplicated records [4], [15]. To identify the duplicated records, user selects two or three fields on the basis of unique identification of records

Manuscript received September 27, 2012; revised November 9, 2012.

The authors are with the Department of Computer Science University of Peshawar, Pakistan (e-mail: smbilal_84@yahoo.com).

and then produces sorted token tables over these fields. Tokens are created on the basis of initials of letters. An obvious drawback of this technique is that in many cases it considers non duplicated records as duplicated recorders, causing an increase in the value of false positive (non duplicated records considered as duplicated records). For example, token created for the name column containing values 'Aslam Khan' and 'Asim Khan' will be 'AK'. Although actual values are different but token created for these values is same. Thus non duplicated records are considered as duplicated records.

III. Proposed De-Duplicator Algorithm

Data cleansing process is used to identify and remove duplicated records. This problem can be explained as an example in the healthcare business. If a customer's record is stored more than one times, the company will send him mails more than once as he is considered another individual but in fact he is the same person. Similarly in data warehousing where analysts make decisions, such redundancy can cause the analysis to produce the wrong result that leads to wrong decisions and thus the business will suffer.

There is a need to detect and remove duplicated records from the data warehouse. Duplication affects the overall performance of data warehouse and also slows down the knowledge extraction process by data mining.

The proposed de-duplicator algorithm is primarily used to identify and remove duplicated records in data warehousing. This algorithm not only improves the data quality but also the performance of the data warehouse. The algorithm uses three steps to identify fully and partially duplicated records. These steps are conversion, clustering and matching.

A. Conversion

De-duplicator algorithm first brings the data into a uniform format. As the data fed to the data warehouse comes from different operational systems, there could be numerous formatting issues in data. One of them is data type format mismatch. For example, a date may be in the formats of dd-mm-yyyy, mm-dd-yyyy, or yyyy-mm-dd. Similarly, a phone number having country code and city code in one record but in another record same phone number without city and country code. Such formatting and missing values issues are resolved and data is brought to a uniform format. Similarly, abbreviations are expanded.

To standardize and remove inconsistency in the data, our approach brings the data into a uniform format and then converts all field values (whether string, numeric or date) into numeric form by applying the radix formula on data. After conversion of the field values into numeric form, an extra column is appended storing all the calculated values into that column corresponding to relevant row separated with comma (,).

Table I contains three fields; first of all field values of the table are converted into numeric form and then stored in the appended column.

$$\sum[((radix)^{position} \times alphvalue) \bmod m] \quad (1)$$

where alphvalue is marked from 0-9 and aA=10, bB=11,……, zZ=35 and m is any large prime number.

The value of radix is greater than or equal to 36 because it consists of 36 characters (10 digits i.e. 0-9 + 26 alphabets + special characters). As the value of radix depends on digits, alphabets and special characters that is why its value is greater then or equal to 36 (e.g. radix>=36). The use of special characters may increase the value of radix because special character values are also used in alphval. The value of position is marked from right to left starting with 0. Fig. 1 describes the process of conversion. Fig. 2 describes the complete algorithm of data cleansing and numeric conversion.

TABLE I: Numeric Value Conversion

Name	Fname	Sal	Numeric Conversion
Asim	Asghar	14000	1321, 1487, 728

ASIM

Let m = 731 (large prime number)

$$\sum\{((36)^3 \times 10) \bmod 731) + ((36)^2 \times 28) \bmod 731) + ((36)^1 \times 18) \bmod 731) + ((36)^0 \times 22) \bmod 731)\}$$
$$= 182 + 469 + 648 + 22$$
$$= 1321$$

Fig. 1. Numeric value conversion formula

Input: Table with different data format, and abbreviations
Output: Uniform format table with extra appended attribute having numeric value

Algorithm
Begin
For attribute *j* = 1 to last attribute, n

For row *i* = 1 to last row, m

1. Bring attribute values into uniform format
2. Remove the special character
3. Remove the variation of attribute values
4. Expand abbreviations
5. Convert all the values into numeric form
6. Put the numeric value into appended attribute separated with comma (,)

end

Fig. 2. Algorithm for numeric conversion

B. Clustering

After storing the values in the Numeric Conversion column, again radix formula given in equation (1) is applied on values of the Numeric Conversion column and output is stored in the Final Output column as shown in Table II. Then K-means clustering algorithm [16], [17], [18] is applied on the data stored in Final Output column and results are stored in the Clustered column. In this manner matching records are stored in one cluster. But clustering reduces the number of

comparisons and ultimately improves the performance.

In Table II, the total numbers of groups are two and total records are four. If there is a single group in table then the numbers of comparisons will be 6. Because row 1 is compared with 3 rows i.e. row 2nd, 3rd and 4th, row 2 compared with 2 different rows i.e. 3rd and 4th rows and row 3rd row compared with only one row i.e. row 4. But with two groups, it reduces the number of comparisons. For example in Table II, we have two groups. To find the duplicated records, we compare the records within a cluster which reduces the number of comparison and our de-duplicator algorithm works faster.

TABLE II: CLUSTERING

Name	Fname	Sal	Numeric Conversion	Final Output	Clustered
Amjad	Qasim	8000	1686, 1397, 438	2559	G1
Zaker	Qasim	8000	1319, 1397, 438	2794	G1
Zaheer	Tanveer	14321	2105, 2767, 1034	3482	G2
Daud	Thalet	16233	1424, 2186, 1713	3261	G2

C. Matching

After conversion step, the divide and conquer approach is applied on each row of the cluster. This approach divides the values recursively into smaller pieces and continues the process until certain smallest size is reached. Then compares the single value of one record is with the single value of other record. If match is found between values of records then the percentage duplication of records is calculated.

In Table III, three attribute values of both records are match and total attribute values are four. That's why both of these records are 75% duplicated ($3/4 \times 100 = 75\%$).

The difference between these records is due to data entry operator error. In Table III, data entry operator types 'Dajid' instead of typing 'Sajid' in the name field value in second row. Such a difference is called erroneous difference and corrected by the domain expert.

TABLE III: PARTIALLY DUPLICATED RECORDS

Name	Fname	Job	Salary	Appended Column
Sajid	Asif	Accountant	10500	1707,1314,2535, 1141
Dajid	Asif	Accountant	10500	1382,1314, 2535, 1141

In Table IV, two records are 75% duplicated. There is an original difference between these records. For example, difference occurs in the name field of row 1 and 2. These two records are for two different individuals. Only name field values are different i.e. 'Imad' and 'Iman' and other attributes value of both the records are same. When domain expert analyzes that the difference is original then he/she keeps the records.

If both records are fully duplicated as in Table V, then the duplicate records are discarded and the original row is kept. For example, in Table V both the records are 100% duplicated, when the divide and conquer approach is applied on these records, the system identifies that these records are fully duplicated.

TABLE IV: ORIGINAL DIFFERENCE

Name	Fname	Job	Salary	Appended Column
Imad	Irfan	Prof	18950	996, 1406, 1238, 1826
Iman	Irfan	Prof	18950	1006, 1406, 1238, 1826

TABLE V: FULLY DUPLICATED RECORDS

Name	Fname	Job	Salary	Appended Column
Ifnan	Saif	Clerk	6500	1614, 1267, 1357, 1326
Ifnan	Saif	Clerk	6500	1614, 1267, 1357, 1326

If records are partially duplicated then the threshold value is checked. If the percentage of duplication crosses the threshold, the program displays those records and mentions clearly those attribute values having difference among them as an output to analyze whether this was an actual difference or erroneous difference among those records.

For example, the difference in Table III is an erroneous entry which is corrected and stored. In this manner, both records become identical or fully duplicated. The duplicated records are then discarded and single row is kept. But in case of actual difference between the records as in Table IV, both rows are kept.

Sometimes column values don't look fully matched but they are actually matched. For example: column name having two values one is "Asim Ali Asghar" and other is "Asim Asghar" are actually matched but it does not seem that these values are matched.

To identify such matching records, domain expert defines threshold for column value and when threshold value is crossed, the algorithm considers that the values of column are matched.

Our algorithm not only specifies the threshold for the single column value matched but it also specifies the threshold for all columns to identify the partial duplicated records. For example, we have 10 columns and two records having 8 columns matched values and values of two columns are not matched. In such a situation domain expert needs to specify the threshold. If the matching values of records cross the threshold then it display those records and mention the non matching values. The domain expert can correct if there is any erroneous difference, discard the record and keep the original entity otherwise leaves the records.

In Fig. 3, De-Duplicator algorithm describes the complete procedure of comparison between records and identifying duplicated records. Symbols used in De-Duplicator algorithm are explained in Table VI.

TABLE VI: FULLY DUPLICATED RECORDS

Symbol	Description
Δ	Percentage duplications between records
Ξ	Threshold value specified by the domain expert
P	First position of record i.e. 1
Q	Last position of record i.e. n
V	Number of values after dividing the row into two portion

Input: Table with appended column and duplicated records,
Output: Cleaned table

Algorithm
Begin
For row i = 1 to last row, n

1. for v, p =1 to q
2. if v>1 then go to step 7
3. else compare all the value with the corresponding value of other row
4. if match found b/w values then
5. calculate the δ and go to step 8
6. else go to step 13
7. divide(p+q)/2 and go to step 2
8. if δ = 100% then discarded the duplicated record and go to step 13
9. else if $\delta >= \xi$ then
10. display the records and mention the attributes values having difference b/w values
11. if difference is due to data quality then correct the entities and go to step 8
12. else go to 13
13. exit

Fig. 3. De-Duplicator Algorithm

IV. Experimental Result

The dataset has been taken from [19] for designing an experiment on the de-duplicator algorithm. The data set named "Restaurant" contains 864 records where 112 records are duplicated.

Our de-duplicator algorithm identifies and removes all fully duplicated records with or without clustering. The use of clustering reduces the number of comparisons. Thus for fully duplicated records, it provides 100% accuracy. In Fig. 4, the graph shows that when the number of clusters increases, the elapsed time decreases. For example, when we have one group then it takes more time to identify and remove duplicated records. But when the number of clusters increases then elapsed time decreases. Time elapsed is used for fully duplicated records as well as partially duplicated records.

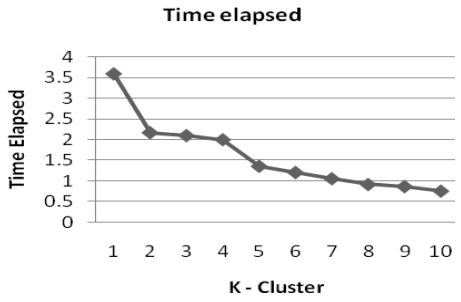

Fig. 4. Graph between time elapsed and Number of cluster

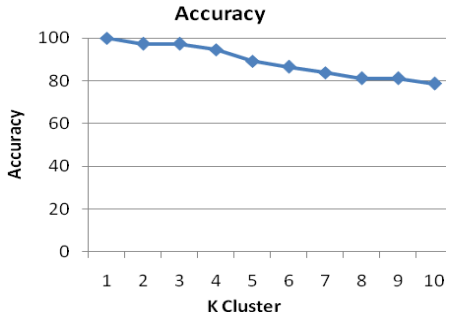

Fig. 5. Graph between k-cluster and accuracy

We need to identify partially duplicated records which may occur in different groups. We can not compare partially duplicated records which are present in different groups. In Fig. 5, the graph represents the accuracy of partially duplicated records, which decreases by increasing the number of clusters.

V. Conclusion

A numeric conversion and matching technique of record de-duplication is explained in this article. An algorithm for numeric conversion is used which converts all attributes data either string, date, or numeric into a standard numeric form. Numeric form of attribute value is used to create clusters which are helpful in reducing the number of comparisons. On the basis of these clusters, divide and conquer technique is used parallel in all these clusters to identify and remove the duplicated records.

In the proposed technique, only single table is used instead of multiple sorted tables. Our technique not only detects fully duplicated records but also partially duplicated records.

References

[1] P. Poonniah, *Data Warehousing Fundamentals- A comprehensive guide for IT professionals*, 1st ed., 81-265-0919- 8, Glorious Printers: New Delhi, India, 2006.

[2] R. Arora, P. Pahwa, and S. Bansal, "Alliance Rules for Data Warehouse Cleansing,"in *Proceeding of International Conference of Signal Processing Systems*, IEEE, 2009.

[3] M. Rehman and V. Esichaikul, "Duplicated Record Detection for Database Cleansing," in *Proceeding of Second International Conference on Machine Vision,* 2009.

[4] T. E. Ohanekwu and C. I. Ezeife, "A Token-Based Data Cleaning Technique for Data Warehouse System," University of Windsor, in *Proceeding of 9th International Conference on Database Theory ICDT*, Siena, Italy 2003.

[5] A. D. Chapman, "Principals and methods of data cleaning: Primary Species and Species-Occurrence Data," version 1.0. Report for the Global Biodiversity Information Facility, Copenhagen Australia 2005.

[6] A. K. Elmagrmid, P. G. Ipeirotis, and V. S. Verykois, "Duplicated Record Detection: A Survey," *IEEE Transaction on Knowledge and Data Engineering*, vol. 19, no. 1, IEEE, 2007.

[7] C. I. Ezeife and A. O. Udechukwu, "Data position and profiling in domain independent warehouse cleaning," Canada: University of Windsor, in ICEIS. 2003.

[8] H. Muller and J. C. Freytag, "Problems, methods, and challenges in comprehensive data cleansing," Humboldt-Universität zu Berlin, Institut fur Informatik 2003.

[9] J. J. Tamilselvia and V. Saravanan, "Handling noisy data using attribute selection and smart tokens," in *Proceeding of International Conference on Computer Science and Information Technology*, IEEE, 2008.

[10] S. O. A. Pei, "A comparative study of record matching algorithms," RWTH Aachen, Thesis, Germany, University of Edinburgh, Scotland 2008.

[11] X. Zhu, P. Zhang, X. Wu, D. He, C. Zhang, and Y. Shi, "Cleansing noisy data streams," in *Proceeding of Eighth IEEE International Conference on Data Mining*, 2008.
[12] J. Jebamalar, Tamilselvi, and V. Saravanan, "Detection and elimination of duplicate data using token-base method for data warehouse: a clustering based approach," in *Proceeding of International Journal of Dynamics of Fluids*, vol. 5, pp. 145–164, 2009.
[13] D. Bitton and D. J. Dewtt, "Duplicate record elimination in large data files," *ACM Transactions on Database Systems*, vol. 8, pp. 255-265, 1983.
[14] M. A. Hernandez and S. J. Stolfo, "The merge/purge problem for large databases," *Proceedings of the ACM SIGMOD International Conference on Management of Data,* pp. 127-138, 1995.
[15] E. Rahm and H. I. Do, "Data cleaning: Problem and current approaches," In *proceedings of IEEE Bulletin of Technical Committee on Data Engineering*, vol. 23, no.4, pp. 3-13. Germany: IEEE 2000.
[16] W. Su, J. Wang, and F H. Lochovsky, "Record matching over query results from multi web databases," *IEEE Trabsactions on Knowledge and Data Engineering Manuscript TKDE*- 2008- 12-0639, 2009.
[17] O. Benjelloun, H G. Molina, D Menestrina, Q. Su, S. E. Wang, and J. widom, "Swoosh: a generic approach to entity resoulation," *The VLDB Journal,* 2009, vol. 18, pp.255-276.
[18] C. Batini, C. Cappliello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Computing Survey*, vol. 41, no. 3, Article 16. July 2009.
[19] M. Bilenko, RIDDLE, Repository of information on duplicate detection, record linkage, and identity uncertainty (2002). [Online] Available: http://www.cs.utexas.edu/users/ml/riddle/index.html.

Bilal Khan did his MS / M.Phil in computer science in 2012 from Department of Computer Science, University of Peshawar, Pakistan. He has published different papers in International and National Conferences and Journals. Currently he is working as Programmer of the Finance Department, Khyber Pakhtunkhwa, Pakistan. His research interests are Data Mining, Data warehousing, Data Quality and Clustring.