Identification and Removal of Duplicated Records

Bilal Khan, Azhar Rauf, Sajid H. Shah and Shah Khusro

Department of Computer Science, University of Peshawar, Peshawar, Pakistan

Abstract: Various data quality problems arise when data is integrated from different heterogeneous sources into a data warehouse. Records duplication is one of the prominent problems in data warehouse. 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. This algorithm provides precision to identifying duplicated records efficiently. Experimental results show that this research improves Recall and F-Score of duplicated records.

Key words: Recall, Data cleansing, De-Duplicator, Partial duplication, F-Score

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 data warehouse. Data cleansing is a process of detecting and correcting incorrect, redundant and missing values. This process also checks the format, completeness and violation of business rules 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, mismatch 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 [4-7]. Data quality measures the accuracy, integrity, completeness, validity, consistency and redundancy aspects of data [8, 9].

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 [4]. Records duplication is one of the major issues in data quality [4, 10, 11]. It is the representation of the same real world object more than once in the same table [5, 10]. It is necessary to eliminate duplicated records in order to bring consistency and improve the quality of the data. To identify duplicated records and remove them efficiently, the researchers proposed different techniques in the area of data mining and data warehousing [12-19]. Records duplication is also known as entity resolution, record linkage or merge purge.

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 both fully and 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 seem to be different due to erroneous entry by data entry operator. 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.

Corresponding Author: Bilal Khan, Department of Computer Science, University of Peshawar, Peshawar Pakistan. Cell. +92-312-9576357, E-mail: smbilal_84@yahoo.com.
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.

Related Work: The problem of duplicated records has been extensively discussed in the literature. Bitton and Dewitt [20] discuss the elimination of duplicated records in large data files by sorting which brings identical records together. 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 and Stolfo [21] discuss the problem of merge/purge in a large database. 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 [21]. 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 Elmargmrid et al. [6] for detecting record duplication. 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 [1]. To identify the duplicated records, the user selects two or three fields on the basis of unique identification of records and produces tables of sorted tokens 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 records, 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 ‘Atif Khan’ will be ‘AK’. Although actual values are different but tokens created for these values are same. Thus non duplicated records are considered as duplicated records.

Beskales et al. [22] introduce new probabilistic ETL tool for identification and elimination of duplicated records. The function of this ETL tool is data transformation. Specific threshold is used to identify either duplicated or non-duplicated records.

Ahmed and Aziz. [23] discuss different techniques to improve accuracy rate of data quality. They introduce data cleansing framework which consists of attribute selection, token formation, clustering and eliminator functions. The drawback of this technique is that it based on token based technique, so this technique introduces the large number of false positive values.

Panse et al. [24] propose the method for duplicate detection in probabilistic data. But probabilistic data does not provide accurate results.

Proposed De-duplicator Algorithm: Data cleansing process is used to identify and remove duplicated records. The problem of record duplication can be explained by an example in the healthcare business. If a customer’s record is stored more than once, 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.

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 the same phone number with missing 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 that stores all the calculated values separated with comma (,) corresponding to relevant row.

In Table 1, Equation (1) converts the column Name, Fname and Sal values into numeric form and store in appended column i.e. "Numeric Conversion". As Table 1 contains three fields; first, all the field values of the table are converted into numeric form and stored in the appended column.

\[
\sum[(\text{radix}^{\text{position}} \times \text{alphabetvalue}) \mod \text{radix}]
\]  

(1)

In Equation (1) alphabetvalue is used to marked from 0-9 and aA=10, bB=11, ..., zZ=35 and \( \Delta \) 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 than 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 alphav. The value of position is marked from right to left starting with 0. Figure 1 describes the process of conversion.

Figure 2 describes the complete algorithm of data cleansing and numeric conversion. All the attribute values (from first attribute i.e. \( j=1 \) to last attribute i.e. \( n \)) of rows (from first row \( i=1 \) to last row \( m \)) are taken as an input. In step 1 the algorithm checks the properties of attribute and brings them into uniform format. To bring consistency and improve quality of data, few defined special characters and variations of attribute value are removed in step 2 and 3. In step 4, abbreviations are expanded, for example if UOP value is stored in column "University Name" then algorithm expands it to University of Peshawar. Using Equation (1) step 5 converts the attribute value into numeric form and the numeric value is stored in the extra appended column.

<table>
<thead>
<tr>
<th>Name</th>
<th>Fname</th>
<th>Sal</th>
<th>Numeric Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asim</td>
<td>Asghar</td>
<td>14000</td>
<td>1321, 1487, 728</td>
</tr>
</tbody>
</table>

Table 2: Clustering

<table>
<thead>
<tr>
<th>Name</th>
<th>Fname</th>
<th>Sal</th>
<th>Numeric Conversion</th>
<th>Final Output</th>
<th>Clustered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 2. Then K-mean clustering algorithm 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.

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

In table 2, the total numbers of groups are two and total records are four. If there exists 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 compared with only one row i.e. 4th row. But with two groups, the numbers of comparisons are reduced. For example in table 2, 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.

In Table 2, Equation (1) converts the values of Numeric Conversion column again into numeric form and stores the value in Final Output column and then applies the K-mean clustering algorithm on Final Output column.
C. Matching: After clustering 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 the single value of one record is compared with the single value of other record. If match is found between values of records then the percentage duplication of records is calculated. As dividing and matching the row values take constant time. Therefore running time of dividing and matching among rows is $O(1)$ which improve the efficiency.

\begin{verbatim}
match_count = 0;
IF utl.match.edit_distance_similarity(tmp_student(i).
  name, tmp_student(j).name) > \delta
  THEN match_count = 1/n * 100;
  --n is the total number of column
END IF;
\end{verbatim}

For matching process we use the edit distance similarity formula which calculates the similarities or percentage matching between values. If matching values is greater than threshold $\delta$, then values are considered as duplicated values. Thus the algorithm updates the match counter if the value exceed the threshold and sum all the calculated values of match counter.

\begin{verbatim}
IF match_count = X
  -- X = 100% which means that records are fully duplicated
THEN tmp_student.DELETE(j); END IF;
\end{verbatim}

If value of match counter is equal to 100 percent then the records are fully duplicated otherwise check the value of match counter if it is greater than the value of $\xi$ then algorithm will display the records.

\begin{verbatim}
IF match_count >= \xi
THEN
dbms_output.put_line('Display: ||tmp_student(j).name');
END IF;
\end{verbatim}

In table 3, there are four attributes and two records. Values of three attributes in both records get matched and value of one attribute remains unmatched. That’s why both of these records are 75% duplicated (3/4 * 100 = 75%).

The difference between these records is due to data entry operator error. In table 3, data entry operator types ‘Dajid’ instead of typing ‘Sajid’ in the name field value in second row. Such difference is called erroneous difference and corrected by the domain expert.

<table>
<thead>
<tr>
<th>Name</th>
<th>Fname</th>
<th>Job</th>
<th>Salary</th>
<th>Append Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sajid</td>
<td>Asif</td>
<td>Accountant</td>
<td>10500</td>
<td>1707,1134,2535,1141</td>
</tr>
<tr>
<td>Dajid</td>
<td>Asif</td>
<td>Accountant</td>
<td>10500</td>
<td>1382,1134,2535,1141</td>
</tr>
</tbody>
</table>

In Table 3 records are partially duplicated and duplication is due to data entry operator errors, in the name field values. Domain expert correct one entry and make the 100% duplicated record.

In table 4, 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 5, then the duplicate records are discarded and the original row is kept. For example, in table 5 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.

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 3 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 4, 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.
Table 6: Symbol Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta )</td>
<td>Percentage duplications between records</td>
</tr>
<tr>
<td>( \xi )</td>
<td>Threshold value specified by the domain expert</td>
</tr>
<tr>
<td>P</td>
<td>First position of record i.e. 1</td>
</tr>
<tr>
<td>Q</td>
<td>Last position of record i.e. n</td>
</tr>
<tr>
<td>V</td>
<td>Number of values after dividing the row into two portion</td>
</tr>
</tbody>
</table>

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 matching values in single column 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 matched values in 8 columns and unmatched values in 2 columns. 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 Figure 3, De-Duplicator algorithm describes the complete procedure of comparison between records and identifying duplicated records. The time complexity of de-duplicator algorithm is \( \mathcal{O}(kn) \) where \( k \) represents the total number of row in a cluster. Symbols used in De-Duplicator algorithm are explained in Table 6.

**Experimental Result:** The dataset has been taken from [25] for designing an experiment on the de-duplicator algorithm. The data set named “Restaurant” contains 864 records where 112 records are duplicated. Approximately 13 percent records of total records are duplicated. The experiment is performed on computer having following specifications: CPU P4, 3.0 Ghz processor, 1G RAM, 120GB Hard disk. The operating system used is window XP and DBMS is Oracle 10g.

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 Figure 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.

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

To show the effectiveness of our technique, we measured the accuracy of our algorithm based on True positives (duplicated records identified by algorithms), False positives (non duplicated records considered as duplicated records), False negative (duplicated records considered as non duplicated records), Precision, Recall, F-score.

\[
Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \times 1 \tag{2}
\]

\[
precision = \frac{True\ Positive}{True\ Positive + False\ Negative} \times 1 \tag{3}
\]

\[
FScore = 2 \times \frac{precision \times Rec}{precision + Rec} \tag{4}
\]

**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 \( d \) and go to step 8  
6. else go to step 13  
7. divide \( p \), \( q \) and go to step 2  
8. if \( d = 100\% \) then discarded the duplicated record and go to step 13  
9. else if \( d = ? \) 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  
**End**

Fig. 3: De-Duplicator Algorithm
number of clusters, duplicated records move from one cluster to another, resulting in an increase in the value of false negative and decrease in true positive. Thus the number of clusters has an overall affect on the values of true positive, false positive and also recall.

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 in parallel in all these clusters to identify and remove the duplicated records. Before this technique sequential process is used for matching among the values of records. But proposed technique takes each single value and match with the corresponding value and all the value of row match at a time.

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. This technique provide the accuracy and improve the number of true positive values provide zero value in all cases. To provide higher efficiency this technique identify maximum numbers of duplicated records which improve the value of recall.

REFERENCES