ISSN:2231-0711

Rajshree Y. Patil et al | IJCSET(www.ijcset.net) | September 2016 | Vol 6, Issue 9, 335-338

Development of Data Cleaning algorithm by Finding Fuzzy Duplicates in Insurance Data Warehouse

Rajshree Y. Patil
Vivekanand College,Kolhapur

Dr. R. V. Kulkarni
SIBER, Kolhapur

Abstract-Data cleaning is the method of adjusting or eradicating information that is wrong, unfinished, inappropriately formatted or reproduced. A business in a data-intensive profession like banking, insurance, trade and telecommunication might use a data cleaning engine to methodically inspect data for errors by using set of laws, business rules and algorithms. Data cleaning is a vital undertaking for data warehouse experts, database managers and developers in insurance industry. In today's competitive environment, there is a need to make available a program for insurance industry which emphasizes on data quality by using methods and techniques for data cleaning. This will help companies to improve data preparation and reporting capabilities. This paper aims to facilitate the data cleaning process by addressing the problem of duplicate records detection pertaining to the name attributes of the data sets. It provides a sequence of algorithms through a novel framework for identifying duplicity in the 'name' attributes of the data sets of an already existing data warehouse.

INTRODUCTION

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, tables, or database. Used mainly in databases, the term refers to identifying incomplete, incorrect, inaccurate, irrelevant, etc. parts of the data and then replacing, modifying, or deleting this dirty data. After cleaning, a data set will be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores.
Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at entry time, rather than on batches of data.
The database structure used to store insurance database, the framework used to design an algorithm and algorithms used to clean insurance data. Decision support analysis on data warehouses influences important business decisions; therefore, accuracy of such analysis is crucial. However, data received at the data warehouse from external sources usually contains errors, e.g., spelling mistakes, inconsistent conventions across data sources, missing fields. Consequently, a significant amount of time and money are spent on data cleaning, the task of detecting and correcting errors in data. A prudent alternative to the expensive periodic data cleaning of an entire data warehouse is to avoid the introduction of errors during the process of adding new data into the warehouse. This approach requires input tuples to be validated and corrected before they are loaded.

Dataset used:
The researchers collected the data of policy holders from agent of insurance office by using questioner and stored in the database where each policy tuple consists of different attributes related to customer profile.

MATCHING TUPLES:

In the following example a policy holder "$Kulkarni\,Jyoti\,Narayan$" have three different policies. In the tuple with $tid = 1$ all the data is correct but in the tuple where $tid = 2$ Name is entered slightly different and in the tuple where $tid = 3$ email address is different. The system could not identify that all the three tuples are of same person. Insurer stores all the three policies as different tuples. In some situations when we want to find a customer who have more than one policy for campaigning some new offers this customer did not get this opportunity. So researchers develop the following algorithm to find similarity between records to correct errors or to find number of policies belonging unique customer.

Tid	Policyno	Name	Address1	Area	Email	Mobileno	Bdate	Policytype
1	*945755760*	*KulkarniJyoti Narayan*	*243,Flat No. G-1, Safalya Appt.*	*Tarabai Park*	*jyoti_kulkarni@yahoo.com*	*9420493151*	*30-Dec-65*	*Jeevan Sangam*
2	*945755761*	*Kulkarni Joti Narayan*	*243,Flat No. G-1, Safalya Appt.*	*Tarabai Park*	*jyoti_kulkarni@yahoo.com*	*9420493151*	*30-Dec-65*	*20 years Money Back*
3	*945755762*	*KulkarniJyoti Narayan*	*243,Flat No. G-1, Safalya Appt.*	*Tarabai Park*	*joti_kulkarni@yahoo.com*	*9420493151*	*30-Dec-65*	*149 Jeevan Anand*

Figure 1: An example record matching task

For this type of example researchers develop an algorithm by using fuzzy matching lookup table. A common technique validates incoming tuples against reference relations consisting of known-to-be-clean tuples. The reference relations may be internal to the data warehouse (e.g., customer relations) or obtained from external sources (e.g. valid address relations from postal departments). Attributes(e.g. $Name, birth\ date$ etc.) The errors in the input customer tuple need to be corrected before it is loaded. The information in the input tuple is still very useful for identifying the correct reference customer tuple, provided the matching is resilient to errors in the input tuple. We refer to this error-resilient matching of input tuples against the reference table as the fuzzy match operation. The reference relation, Customer, contains tuples describing all current customers. If the fuzzy match returns a target customer tuple that is either "$exactly\ equal$" or "$reasonably\ close$" to the input customer tuple, then we would have validated or corrected, respectively, the input tuple. The notion of closeness between tuples is usually measured by a $similarity\ function$. If the similarity between an input customer tuple and its closest reference tuple is higher than some $threshold$, then the correct reference tuple is loaded. Otherwise, the input is routed for further cleaning before considering it as referring to a new customer. A fuzzy match operation that is resilient to input errors can effectively prevent the proliferation of fuzzy duplicates in a relation, i.e., multiple tuples describing the same real world entity. Each match includes a similarity score and a confidence score. The similarity score is a mathematical measure of the textural similarity between the input record and the record that Fuzzy Lookup transformation returns from the reference table. We can think of it as the similarity point out of 1. The confidence score is a measure of how likely it is that a particular value is the best match among the matches found in the reference table. The confidence score assigned to a record depends on the other matching records that are returned.

Understanding the Jaccard Index of Similarity:

The magic behind the Fuzzy Lookup Add-in comes from the Jaccard Index of Similarity. The Jaccard index, also known as the Jaccard similarity coefficient, was developed by Paul Jaccard as a way to document the distribution of different types of flora (yawn). Jaccard's index gave him a statistical way to measure similarities between sample sets.

Jaccard Index is calculated statistically as follows:

The $dist(x, y)$ is calculated using Jaccard index known as Jaccard similarity coefficient is a static used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ Where $A \cap B = \{x | x \in A\ both\ x \in B\}, A \cup B = \{x | x \in A\ or\ x \in B\ or\ x \in A \cap B\}$, Similarity function $d(x, y) = 1 - J(A, B)$. Confidence level is also a score between $0\ and\ 1$; it describes the likelihood that each possible match record returned by the Fuzzy Lookup is a match to the input record. This measure indicates the probability that each returned reference record for any one input record is actually a match. A high similarity factor does not necessarily mean that a definitive match has been found, but the combination of high similarity and confidence scores usually does. For reference tables that do not remain static, the Fuzzy Lookup transformation can be configured to maintain the match index as the underlying reference table changes. The Fuzzy Lookup Table Maintenance feature allows the error-tolerant index to be updated each time the reference table changes and a Fuzzy Lookup task is executed against it. This feature involves adding a trigger to the reference table itself, so performance implications should be weighed carefully when considering this technique. As an alternative to both of these methods, the match index can simply be recreated each time the Fuzzy Lookup transformation is executed. The next steps are setting the similarity threshold and maximum number of results per input record. Record matching is the problem of identifying matching or duplicate records, records that corresponds to the same real-world entity. The standard approach to record matching is to use textual similarity between the records to determine whether or not two records are matched. The input to the data cleaning a relation, R is a relation Reference Table with $R_1, \dots, R_k$ attributes

$R = \{R_1(r_{11}, r_{12}, \dots, r_{1n}), R_2(r_{21}, r_{22}, \dots, r_{2n}), \dots, R_k(r_{k1}, r_{k2}, \dots, r_{kn})\}$

For the above relation x_1 is defined as $x = \{$"945755760", "$Kulkarni\ Jyoti\ Narayan$", "$243, Flat\ No.G - 1, Safalya\ Appt$", "$jyoti\_kulkarni@yahoo.com$", " 9420493151", "$30 - Dec - 65$", "$Jeevan\ Sangam$"$\}$, S is a relation Look up table with $S_1, \dots, S_k$ attributes

$S = \{S_1(s_{11}, s_{12}, \dots, s_{1n}), s_2(s_{21}, s_{22}, \dots, s_{2n}), \dots, S_k(s_{k1}, s_{k2}, \dots, s_{kn})\}$

For the lookup table y_1 is defined as $y = \{$"$Kulkarni\ Jyoti\ Narayan$","$jyoti_{kulkarni}@yahoo.com$", "$9420493151$","$30 - Dec - 65$"$\}$. In above example for tuple $id = 1\ x\ and\ y$ are similar, for tuple $id = 2\ x\ and\ y$ are different because $x.name \neq y.name$ and for tuple $id = 3\ x\ and\ y$ are different as $x.email \neq y.email$ so to find matching tuples and similarity between two tuples researchers used a fuzzy lookup table algorithm. The matching pair (r_{11}, s_{11}) is textually similar, while the matching pair (r_{21}, s_{11}) is not. The record matching problem is identifying all pairs of matching records$(x, y) \in R \times S$, for given two record set R and S.Two records match if they represent same entity. Let a fuzzy lookup relation $S(S_1, S_2, \dots, S_k)$. The output is a partition of the records in R which we capture through selection of attributes from relation S. We refer to each equivalence class in the partition as a group of tuples. The result of matching between two input relations r_1 and s_1 is required to be less than some ϵ is a set: $\{(x, y, dist'(x, y)) | x \in r_1 \wedge y \in r_2 \wedge dist'(x, y) \leq \epsilon\}$. The distance $dist'(x, y)$ is calculated by using Jaccard Index formula. The distance filtering optimization mapping f (e.g. get the letters of a string up to delimiter space) over sets $R_1\ and\ S_1$, with a distance function $dist'$ much cheaper then $dist$, such that: $\forall x,\ \forall y, dist'(f(x), f(y)) \leq$

$dist(x, y)$. Having determined $f\ and\ dist'$, the optimization consists of computing the set of pairs (x, y) such that $dist'(f(x), f(y)) \le \epsilon$, which is a superset of the desired result: $Dist\_Filter = \{(x, y) | x \in S_1 \wedge y \in S_2 \wedge dist'((f(x), f(y)) \le \epsilon\}$. Given this, the set defined by (1)is equivalent to: $\{(x, y, dist(x, y)) | (x, y) \in Dist\_Filter \wedge dist(x, y) \le \epsilon$ 3. Calculation of Jaccard Index for insurer data. For our algorithm Input relation R i.e. Reference Table (Table containing details of policy holders)

$$R = \{policyno, name, birthplace, nationality, address1, address2, taluka, birthdate, mobileno, email\}$$

Input relation S i.e. Lookup table (Table containing lookup data of customer)

$$S = \{name, birthdate, mobileno, email)$$

Entering these two relations researchers allow to enter attributes on which matching rules are to be applied. If matching algorithm is to be performed on names only then only name attribute has to be selected. If matching is to perform on all attributes $\{name, birthdate, mobileno, email\}$ then all attributes are to be selected. After selection of attributes the tokens are partitioned on the basis of delimiter space. Each token is used as a set and the similarity between two data is checked. Eg. "$Kulkarni\ Jyoti\ Narayan$" from the reference table is partitioned as Attribute $A = \{A_1 = KULKARNI, A_2 = JYOTI, A_3 = NARAYAN\}$ with delimiter as space. A_1 set is refereed as set $A = \{KULKARNI\}$ and again lookup table is also partitioned into $\{B_1 = KULKARNI, B_2 = JYOTI, B_3 = NARAYAN\}$ with delimiter space. S_1 is referred as set $B = \{KULKARNI\}$ and then we find the similarity between these two tokens by using Jaccard Similarity Index. Once the similarity index is found it is stored and the similarity index for new token is calculated. This process is repeated until the last attribute. Average similarity index is calculated and then it is checked whether it is lies between $0\ and\ 1$. If similarity index is 1 then algorithm gives the output as "$Exactly\ Match$" if it is greater than 0.80 algorithm displays the ouput as"$Likely\ Match$".

$$A = \{KULKARNI, JYOTI, NARAYAN, \dots\} and\ B = \{KULKARNI, JYOTI, NARAYAN, \dots\}$$

Researchers calculate $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$

Similarity Index $= 1 - J(A, B)$, distance between A, B. Researchers used two real datasets in experiments. Each dataset contains two tables ($R\ and\ S$) over which record matching are performed. The first dataset, called $REFTABLE$, the second dataset called $LOOKUPTABLE$ with number of records. Follwing figure lists the columns of these records; for brevity, we abbreviate column names by their first letters. For record matching, we use 4 similarity dimensions listed in following figure In Figure J denotes Jaccard similarity.

Size	$\lvert R\rvert = 150$, $\lvert S\rvert = 150$
Columns	$Name(N), BirthDate(D), Email(E), Mobileno(M$
Similarity	$J(N), J(N, D), J(N, D, E), J(N, D, E, M)$
Blocking	$J(N) >= 0.90$, $J(N, D) >= 0.90$, $J(N, D, E) >= 0.90$, $J(N, D, E, M) > 0.90$

Figure 2 :Characteristics of Blocking

Algorithm

```
Step 1 :Input relation $R = (R_1, R_2, \dots, R_k)$, $S = (S_1, S_2, \dots, S_k), dist, \epsilon, dist', f$
Step 2 : $x = set\ of\ partitions\ of\ R_1\ according\ to\ f$
Step 3 : $y = set\ of\ partitions\ of\ S_1\ according\ to\ f$
Step 4 : For each partition $x \in R_1$
Step 5 : For each partition $y \in S_1$ such
Step 6 : Find $dist'\ (f(x), f(y))$
         If $dist'(f(x), f(y)) = 1$ then
         Output=$Exactly\ match$
         Else
         Output=$Likely\ match$
Step 7 : goto 5
Step 8 : goto 4
Step 9 : For each element $r_1 \in R_1$
Step 10: For each element $s_1 \in P_1$ do
Step 11 :Enter $threshold\ value$
         If $dist(r1, s1) >= threshold\ value$ then
         clean data
Step 12:goto 10
Step 13: goto 9
Step 14: End.
```

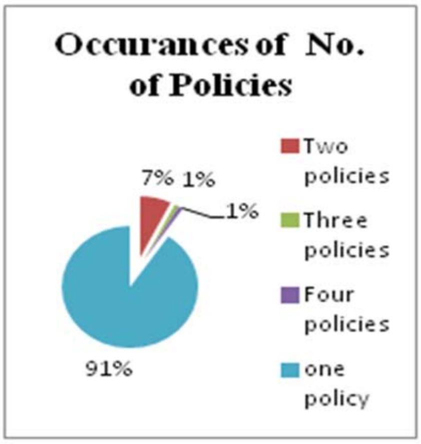
Data Analysis

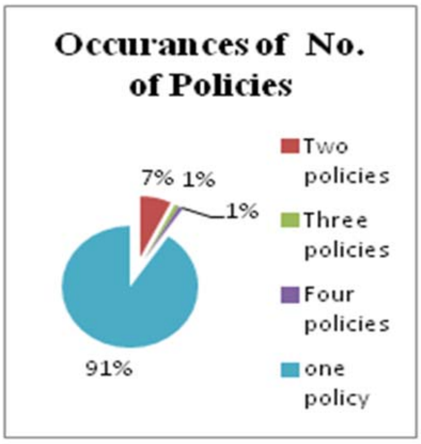
This covers analysis of existing data warehouse system, the architecture and the storage methods used in an insurance industry. It checks efficiency of algorithms developed to clean dirty data. One customer in insurer data can have number of policies. In this module researchers finds Matching records in reference table by using data given in fuzzy lookup table. Here researchers demonstrate finding matching in reference table by giving 4 attributes to search $Name, emil, mobileno, birth\ date$. Insurer database consists of 3 records related to policyholder "$Kulkarni\ Jyoti\ Narayan$". Three records found with same name and their similarity index was given above. In first and third records all contents are same but for second record it contains wrong email id son it displayed similarity index 0.966. With the help of clean option researchers allow insurer to clean that particular data. Researchers compared the duplicate data given by the tool with Pivot table data in excel. From 150 records researchers found following duplicates. Researchers used the following evaluation matrices to calculate efficiency of algorithm.

Accuracy: The percentage of input tuples for which a match algorithm identifies the seed tuple, from which the erroneous input tuples was generated, as the closest reference table is its accuracy.

Parameter Settings: In our entire experiments researchers set $K = 1$ (i.e. we only retrieve $similarity\ threshold\ si = 1$ and the $confidence\ level = 1$). Researchers find accuracy of algorithm with pivot table generated by Excel for the same dataset.

Finding matching duplicates by using tool By using Pivot table





CONCLUDING REMARKS

We have introduced Fuzzy logic to find out duplicate and eliminate duplicate. From this we can find integrated customer view.

REFERENCES

[1] "A Data Cleaning Method Based on Association Rules" by Weijie Wei, Mingwei Zhang, Bin Zhang ,www.atlantis-press.com
[2] Applied Brain and Vision Science-Data cleaning algorithm
[3] "Data Cleansing for Web Information Retrieval using Query Independant Features" by Yiqun Liu, Min Zhang, Liyun Ru, Shaoping Ma- www.thuir.cn
[4] "An Extensive Framework for Data Cleaning " by Helena Galhardas, Daniela Flor escu, Dennis Shasha, Eric Simon
[5] "A Token-Based Data Cleaning Technique for Data Warehouse" by Timothy E. Ohanekwu International Journal of Data Wrehousing and Mining Volume 1
[6] "The role of visualisation in effective data cleaning" by Yu Qian,Kang Zhang – Proceddins of 2005 ACM symposium on applied computing
[7] "A Statistical Method for Integrating Data Cleaning and Imputation" by Chris Mayfield, Jennifer Neville, Sunil Prabahakar- Purdue University(Computer Science report-2009)
[8] "Data cleansing based on mathematical morphology" by Sheng Tang published in ICBBE 2008 The second International Conference-2008