ISSN:0975-9646

DFT: A Novel Algorithm for Data Cleansing

Shweta Taneja , Ishita Ashri , Shipra Gupta , Mehak Sharma

Computer Science Department
Bhagwan Parshuram Institute of Technology, New Delhi

***Abstract*- Data cleansing is an essential part of data mining and has become a pre-requirement before analysing any kind of data. The data collected by an organisation is enormous and full of errors and inconsistencies, which degrades the quality of data and affects the results of mining. Many algorithms have been proposed by several authors to deal with such inconsistencies. But, a little work has been done on the date type field. Being an integral part of any data we need to ensure that the date field associated with a database is consistent in all aspects. This paper addresses the various problems related with date type fields and different types of errors that can occur due to different date formats. We propose an algorithm DFT for the transformation of varying date formats into a unique consistent format to avoid any ambiguities. The data set for implementation of the algorithm is taken from the causelists of Supreme Court of India. The algorithm shows good results and transforms each date record into a unified format to avoid noise in the database.**

***Keywords*— Data Cleaning, Normalisation, Inconsistent Date Formats, Disguised Date values**

I. INTRODUCTION

A Data Warehouse is a subject oriented, integrated, non-volatile and time variant collection of data in support of management's decisions[1]. The Data Warehouse of an enterprise consolidates data from multiple sources in order to support enterprise wide decision making, reporting, analyzing and planning. The processes performed on data warehouse for above mentioned activities are highly sensitive to the quality of data. They depend on the accuracy and consistency of data. Data cleansing, data cleaning or data scrubbing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database [2]. Degraded quality of data leads to wrong conclusions of these processes which ultimately lead to wastage of all kinds of resources and assets. Thus data cleansing plays an important role in data mining. A lot of work has been done in handling various types of inconsistencies in data like Name, Address, Phone number, etc but very little work has been done on the date type format till now.

Every key structure in the data warehouse contains an element of time, explicitly or implicitly. Date field is very crucial for the data warehouse as whenever data would be loaded into the warehouse, the date on which data is loaded would be saved either explicitly or implicitly.

There are different types of inconsistencies which can occur with date type fields. They are:

- Time Zone Differences
- Inconsistent Date Formats
- System Date Errors
- Disguised Date Values
- Integration Errors
- Data Entry Errors

Thus we need to avoid inconsistencies and ambiguities arising with the date type. In this paper we propose an algorithm Date Field Transformation Algorithm (DFT) in order to remove the inconsistencies in the date type field. In our algorithm data from multiple sources having different date formats is integrated. Date is normalized and is converted to a uniform format in order to maintain consistency.

The organisation of paper is as follows. Related work is presented in Section II which summarizes the work done earlier by various authors related to date type field. Section III details the various kinds of problems or errors related to date field. Section IV explains the proposed rules for our algorithm (DFT) which is proposed in section V along with its proposed framework. The algorithm normalizes the date type format after integration from multiple sources. The date normalization framework defines a system which takes data as input from different software and converts the data into a normalized and consistent format. Experiments conducted and results obtained are dealt with in section VI. Some of the limitations in existing approaches which can be worked on in near future are defined in section VII along with the conclusion.

II. RELATED WORK

In order to maintain accuracy and consistent records, we need to ensure that our database is clean. Here data cleansing comes into play. Data cleansing is of utmost importance as it not only removes duplicate data but also corrects inaccurate data. Several papers have been read dealing with the inconsistencies of data. A comprehensive algorithm for data cleansing is proposed in [3]. The target of the authors is to correct and detect most of the error types and problems such as lexical errors, domain format errors, irregularities, integrity constraint violations and duplicates. J. J. Tamilselvi suggests two algorithms to handle noisy data [4]. The noisy data errors include duplicate records, data entry records & spelling errors. The algorithms will handle this noisy data by expanding abbreviations, removing unimportant characters & eliminating duplicates. The two algorithms used are Attribute selection algorithm and token based algorithm. G.Beskales has provided a solution to deal with duplicate words via ProbClean [5]. It generates multiple repairs by setting different parameters and allows clustering of

duplicate records. It allows user to choose the source and the method of repair. Distance function is applied to each unclean attribute and final U-clean relationship. Each repair with its running time is shown. A tool Febrl is developed in [6]. It is an open source tool with Graphical User Interface. It can be used to compare a record linkage technique with the existing record linkage techniques. This tool is useful in development, implementation and testing of new record linkage algorithms and techniques. It supports many text based file formats including CSV. It also contains nine encoding methods for comparison. Problems encountered in the instance level of a database or schema is targeted in [7]. The problems are missing values, duplicate tuples, null values, values outside the domain, among others. This paper delivers an archetype contribution on optimization of algorithm for the detection of duplicate tuples in databases. For this it uses phonetic based on multithreading without the need for trained data and an independent environment of language to be supported. The detection and correction of erroneous data based on error detection rules is dealt with in [8]. These rules have been proposed by analyzing the detection process. The optimized detection algorithm is implemented using SQL statements. In [9], the approach is to holistic repair which improves the quality of the cleaned database with respect to the same database treated with a combination of existing techniques. This paper combines existing formalisms and expresses rules involving numerical values and predicates such as “greater than” and “less than”. The holistic approach holistic approach outperforms previous algorithms in terms of quality and efficiency of the repair. A hybrid approach HADCLEAN for cleaning data is put forward in [10], which is a combination of modified PNRS and transitive closure algorithm. The contemporary PNRS algorithm was correcting the spelling mistakes in data by using a standard dictionary or any other language specific dictionary. Conditional functional dependencies are discovered from relations and used as rules for cleaning relational data in [11]. It has implemented three algorithms for discovering minimal CFDs: CFDMiner (for mining minimal constant CFDs), CTANE (for discovering general minimal CFDs based on the level wise approach) and FastCFD (for discovering general minimal CFDs based on a depth-first search strategy). Z. Yuhang introduces a new method for cleaning new and rare word which are, word frequency i.e. number of times a word is repeated in whole text and weightings i.e. importance of the text, in [12]. They use C-means algorithm for text clustering. Text clustering is used to increase flexibility, scalability and to make it easy to use. This method is more efficient and decrease probability of mistake in filtering rare word. It also increases accuracy of finding these words which make result of text cluster more accurate. A framework based on the ETL process (Extract, Transform & Load) is proposed in [13] .Three parts which are used in the cleaning process are: extract the invalid value, matching attributes with valid values and data cleaning algorithm. K-nearest neighbour algorithm is used for data cleaning in this paper. Payal Pahwa in her paper provides association rules for data mining and specifically deals with the date type field by defining the type of errors and their identification [14]. Alliance rules based on the principle of association in mining are explained in [15]. They provide a method to handle duplicity in the name type field. Few of them target the inconsistencies associated with date type field. A.Marcus and J.I.Maletic have defined the potential errors that may creep in any database such as non-numerical values in data type field, outlier values, date earlier than DOB , empty fields for DOB, outlier values etc [16]. It also proposes data quality metrics to deal with such inconsistencies.

In the above proposals, errors are defined and some rules are proposed. Our approach differs from their proposals .We present Date Field Transformation Algorithm (DFT) to deal with varying formats in the date type field.

III. PROBLEMS IN DATE FIELD

There are various problems that can occur in date field while integrating date from multiple sources. The problems associated with date field are explained below:

a. *No standard format is followed for date therefore there is a need for a normalized representation :* While integrating data from multiple sources , dates might be in different formats like :

FORMAT	*EXAMPLE*
MM-DD-YYYY	12-06-2012
DD-MM-YYYY	06-12-2012
YYYY-MM-DD	2012-12-06
YYYY-DD-MM	2012-06-12
MM-DD-YY	12-06-12
DD-MM-YY	06-12-12
DD-YY-MM	06-12-12
DD-MON-YY	6-Dec-12
DD-MONTH-YY	06-December-12

These dates can be interpreted differently.

b. *Disguise Date Values:* Disguise data refer to fake or incorrect data such as the attribute birth date is often required in many customer account registration forms. However, many customers do not want to disclose their privacy. Popularly, one may choose January 1 (the first value in the pop-up lists of month and day, respectively) in order to pass. Here, January 1 is a disguise for the missing data.[17]

c. *Time Zone Differences:* Time zone may differ due to difference in regions. Different time zones such as BST (British Summer Time), GMT (Greenwich Mean Time), DST (Daylight saving time), UTC (Coordinated Universal Time), etc. are used.

d. *Difference in Abbreviations:* In text or string format user might use different abbreviations. E.g. Jan, January, jn , june, jun, etc.

e. *Different kinds of separators* are used for date values.

f. *De-duplication Errors:* redundant entries for same date/time referring to same entity.[5]

g. *System Date Errors:* Verification of system date, system date checks according to ISO standard.

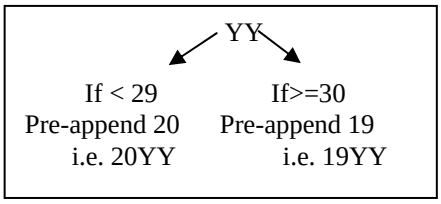
h. *Integration Errors:* Errors which may occur while integrating from various sources, each using a different format for date type.

i. *Data Entry Errors:* Errors which may occur as a consequence of human carelessness and negligence while entering data into the system.

IV. PROPOSED RULES

In our algorithm we have proposed a set of rules which are indicated as follows:

- *R_1 (Rule 1):* The rule 1 places a delimiter between date, month and year fields of Integer type is a must, be it a space. It can be avoided only in the case when month field is of string type and its position is in between the date and year field.
- *R_2 (Rule 2):* The year field should be in YYYY format. It can be used in the YY format only when it is used at the last position. If YY format is used then the rules applied for normalization are given below.



- *R_3(Rule 3):* In case of ambiguity between the month and date field when no record in the table for date field is greater than 12 and no day field is provided then we assume that date follows the month field.
- We have not considered data entry errors in our algorithm.

All the rules stated above are made to avoid any kind of ambiguity while combining data from multiple sources or different date formats obtained from a single source.

V. PROPOSED ALGORITHM- DFT

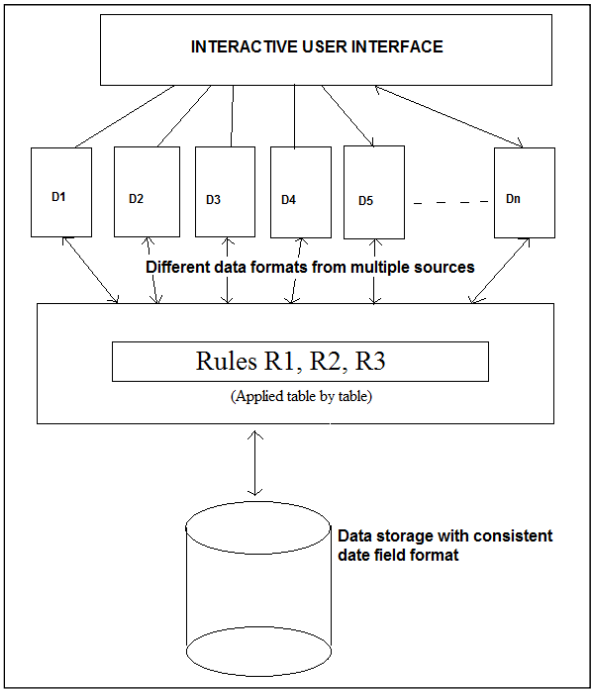


Figure 1: Figure shows the framework of DFT

In this section, we present our proposed algorithm along with its framework and steps.The framework for the algorithm is shown in figure 1 given above. A user can choose a file with the help of an interactive user interface. Files can be taken from multiple sources and can have various file formats. Now the three rules stated above are applied on each table individually. After applying the rules dates are normalised in one consistent and unique format. Now this data will be stored in the database. Since all the dates are in a consistent format, any operation can be applied on it.

The algorithm for date field transformation is proposed as follows. It transforms the date field for a table before integrating it with the other tables.

The following is a list of variables that are used in our algorithm:

D_1, D_2: Delimiters (Can be any symbol, other than letters or digits. Even space and special characters are allowed)

A: Position after D_2

B: Position between D_1 and D_2

C: Position before D_1

V_A: Array of all values of fields after A.

V_B: Array of all values of fields after B:

Vc: Array of all values of fields after C.

The steps used for normalization of date field in a table are given below:

1. Search for a 4 digit continuous number ->YYYY
   1.1. If not found, then the last two digits YY is the year.
   1.2. Rule 2 is applied on the year field YY
   1.3 If YY<29 prefix 20
   1.4 Else If YY>30 prefix 19
2. If a string of at least 3 characters is found
   2.1 Call checkValidMonth() to check if it is a valid month.
   2.2 If step (a) returns true then go to next step.
   2.3 Else go to step 4.
3. If remaining digits <=2, then set it equal to dd.
   3.1 Call checkDate() to check if it is a valid date.
4. Check for the position of year relative to delimiter:
   4.1 If YYYY or YY after D_2
      Case A ->Call $f_1(V_C, V_B)$
   4.2 Else If YYYY between D_1 and D_2
      Case B-> Call f_1 (V_A, V_C)
   4.3 Else If YYYY before D_1
      Case C-> Call f_1 (V_A, V_B)

The following is a list of definition of functions used in the algorithm:

f_1(a [], b []):

1. If any value of a>1
   1.1 a-> dd & b->mm
   1.2 Call dateCheck() to check if it is a valid date.
   1.3 Call checkValidMonth() to check if it is a valid month.
2. Else if any value of b>12
   2.1 a-> mm and b->dd
   2.2 Call dateCheck(false) to check if it is a valid date.
   2.3 Call checkValidMonth() to check if it is a valid month.

3. Else
    3.1 Else a->mm & b-> dd
    3.2 Call checkDate() to check if it is a valid date.
    3.3 Call checkValidMonth() to check if it is a valid month.

checkDate():

1. If Month=(Jan, Mar, May, Jul, Aug, Oct, Dec) || (January, March, May, July, August, October, December) || (01, 03, 05, 07, 08, 10, 12)
    1.1 If 0<Date<32 return true
2. If Month= (Apr, Jun, Sep, Nov) || (April, June, September,
November) || (04, 06, 09, 11)
    2.1 If 1<Date<31
        Return true
3. If Month= Feb || February || 02
    3.1 If (YYYY % 4==0)
        3.1.2 If 0<Date<30
            Return true
    3.2 Else
        3.2.1 If 0<Date<29
            Return true

checkValidMonth():

1. If mm= (January) || (February) || (March) || (April) || (May) || (June) || (July) || (August) || (September) || (October) || (November) || (December) || (Jan) || (Feb) || (Mar) || (Apr) || (May) || (Jun) || (Jul) || (Aug) || (Sep) || (Oct) || (Nov) || (Dec)
    1.1 Return true

2. Else return false

The algorithm successfully transforms the date field of the records into a unique consistent format as per the rules of the algorithm.

VI. EXPERIMENT CONDUCTED AND RESULTS OBTAINED

The proposed algorithm is implemented using java (JDK 1.7) as frontend and MS-SQL 2005 as backend. The data set for testing is taken from the causelists of Supreme Court of India [18]. It is a collection of dates of hearing on court cases. We have taken a random subset of 100 records. Here we present a sample of 10 records in figure 2 and the results obtained after the execution of algorithm is displayed in figure 3.

Sno	Lawyers_Name	Date_of_Listing	Petition_Number	Judge
1	Vijay Kumar	25 july 13	141/2013	Justice G.S Sanghvi
2	Pragati Neekhra	17-10-2013	21796/2013	Justice A.K. Patnaik
3	Prashant Kumar	05-01-2013	20227/2013	Justice Jagdish Singh Khehar
4	Dushyant Parashar	17-Sep-13	4772/2013	Justice A.K. Patnaik
5	Bina Madhavan	08-10-2013	4917/2013	Justice S.A. Bobde
6	Sushma Suri	16-07-2013	552/2009	Justice Ranjana Prakash Desai
7	D.S. Mahra	16August2013	1810/2009	Justice Ranjana Prakash Desai
8	Arjun Krishnan	Aug-23-2013	2114/2007	Justice Ranjan Gogoi
9	Abhijeet Sengupta	2013-August-26	7293/2009	Justice V. Gopala Gowda
10	S.K. Sabharwal	30-04-2012	5270/2012	Justice R.M Lodha

Figure 2: Original database before execution of algorithm

Sno	Lawyers_Name	Date_of_Listing	Petition_Number	Judge
1	Vijay Kumar	25 July 2013	141/2013	Justice G.S Sanghvi
2	Pragati Neekhra	17 October 2013	21796/2013	Justice A.K. Patnaik
3	Prashant Kumar	1 May 2013	20227/2013	Justice Jagdish Singh Khehar
4	Dushyant Parashar	17 September 2013	4772/2013	Justice A.K. Patnaik
5	Bina Madhavan	10 August 2013	4917/2013	Justice S.A. Bobde
6	Sushma Suri	16 July 2013	552/2009	Justice Ranjana Prakash Desai
7	D.S. Mahra	16 August 2013	1810/2009	Justice Ranjana Prakash Desai
8	Arjun Krishnan	23 August 2013	2114/2007	Justice Ranjan Gogoi
9	Abhijeet Sengupta	26 August 2013	7293/2009	Justice V. Gopala Gowda
10	S.K. Sabharwal	30 April 2012	5270/2012	Justice R.M Lodha

Figure 3: Modified Database after execution of algorithm

VII. CONCLUSION AND FUTURE WORK

Our proposed algorithm provides a solution to deal with the inconsistent data formats which may occur in date type field. The algorithm identifies the different formats and transforms them into a unique consistent format. The unique format obtained avoids any errors and inconsistencies and at the same time maintains readability. The desired date format chosen is "dd month yyyy". The algorithm transforms each date record into this unified format to avoid noise in the database.

Our algorithm has various applications. It can be applied on the manual records of various organisations to unify the various formats used in date field. Not only on manual records ,DFT can also be applied on digital formats where a unique format for date becomes necessary for any kind of further processing such as analysing and searching. In future, the work can be extended to cover other errors associated with the date field such as duplications, disguise data values, etc.

REFERENCES

[1] P. Ponniah, *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*, John Wiley and Sons, Inc., 2001, pp.19.

[2] S.Wu, "A review on coarse warranty data and analysis", *Reliability Engineering and System*, vol.114, pp.1–11, June 2013.

[3] Dr. M. M. Hamad and A. A. Jihad, "An enhanced technique to clean data in the data warehouse" in *IEEE conference on Developments, E-systems Engineering*, 2011.

[4] J. J. Tamilselvi, et-al, "Handling Noisy Data using Attribute Selection and Smart Tokens", in *IEEE International Conference on Computer Science and Information Technology*,2008.

[5] G. Beskales, et-al, "ProbClean: A Probabilistic Duplicate Detection System", in IEEE *ICDE Conference*, 2010

[6] P. Christen, "Febrl – An Open Source Data Cleaning, Deduplication and Record Linkage System with a Graphical User Interface", *ACM*, August 24–27, 2008.

[7] T . L. de Andrade, R. C. G. de Souza, et-al, "Optimization of algorithm to identification of duplicate tuples through phonetic based on multithreading", in *12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2011.

[8] Z. Zhang, et-al, "Research of Error Data Detection Algorithm Based on Rules", *IEEE*, 2011.

[9] X. Chu, et-al, "Holistic Data Cleaning: Putting Violations into Context", in *IEEE ICDE Conference*, 2013.

[10] A. Paul, et-al, "HADCLEAN: A Hybrid Approach to Data Cleaning in Data Warehouses", 2012.

[11] W. Fan, et-al, "Discovering Conditional Functional Dependencies", IEEE *Transactions On Knowledge And Data Engineering*, vol. 23, no. 5, May 2011.

[12] Z. Yuhang, " Research on Data Cleaning in Text Clustering", in *IEEE Conference at International Forum on Information Technologygy and Applications*, 2010.

[13] H.H. Mohamed, et-al, "E-Clean: A Data Cleaning Framework for Patient Data", in *IEEE First International Conference on Informatics and Computational Intelligence*, 2011.

[14] A.Marcus, J.I.Maletic, "Utilizing Association Rules For the Identification of Errors in Data", University of Memphis, *TR-CS-00-04*, 2004.

[15] P. Pahwa, et-al, "Alliance Rules for Data Warehouse Cleansing" in *IEEE International Conference on Signal Processing Systems*, 2009.

[16] A.Marcus, J.I.Maletic,"Automated Identification of Errors in Data Sets", University of Memphis, TR-CS-00-02, 2002.

[17] M. Hua and J. Pei, "DiMaC: A System for Cleaning Disguised Missing Data", *SIGMOD, ACM, Vancouver, BC, Canada*, June 9–12, 2008.

[18] Causelists of Supreme Court of India. Available at http://causelists.nic.in/scnew/index1.html.