IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 2 (Jan. - Feb. 2013), PP 17-21
www.iosrjournals.org

Data Cleaning in Text File

Arup Kumar Bhattacharjee[1], Atanu Mallick[2], Arnab Dey[3],
Sananda Bandyopadhyay[4]
(Dept. of MCA, RCC Institute of Information Technology, India)

Abstract : *Data cleaning is an automated process of detecting, removing and correcting incomplete, incorrect, inaccurate and irrelevant data from a record set. Our system works on simple text (*.txt) files using Extract, Transform and Load (ETL) model. In this paper we present a set of algorithms to correct errors such as alpha-numeric errors, invalid gender, invalid ID pattern and redundant ID error. The text files are used as data storage which stores data in a tabular format and the algorithms are applied on each field value depending on its nature.*

Keywords - *ETL, Dirty Data, ID Validation, Alphabetic Validation, Numeric Validation*

I. INTRODUCTION

The system that we have designed eliminates errors from text files and rectifies them. The erroneous data which are called dirty data may have been originally caused by user entry errors. Redundant data may cause inconsistency in the data set [2]. Incorrect data leads to all kind of unpleasant and unnecessary expenses. We need to prepare quality data by pre-processing the raw data because it is used in the process of decision making in an organization [3]. There exist some severe data quality problems that can be resolved by data cleaning and transformation.

Our data cleaning approach satisfies several requirements. First of all, it detects and removes all major errors and inconsistencies in data sources. Algorithms for data cleaning and data transformation are specified in a declarative way and are reusable for other data sources. Data cleaning is an essential task in order to get correct and qualitative data.

This paper presents a solution to handle data cleaning in text files by using different cleaning functions.

II. FRAMEWORK DESIGN

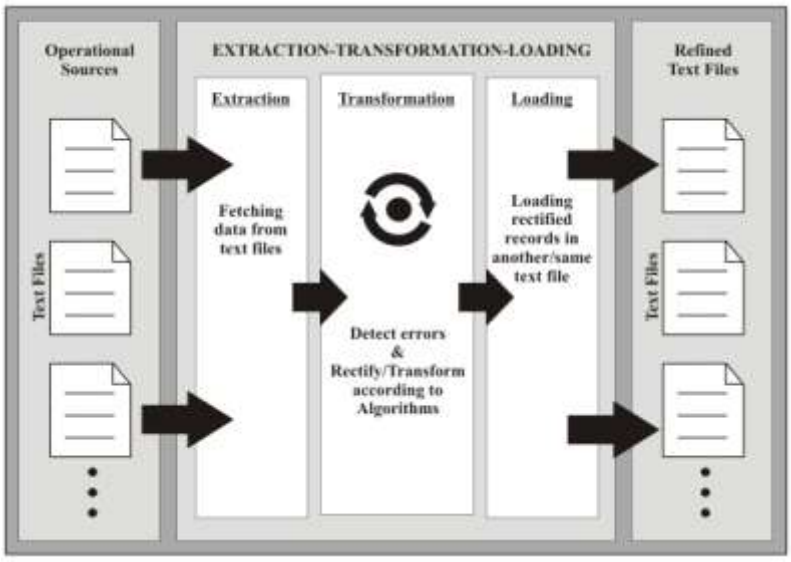


Fig 1: Framework [5], [6]

Extract - The process of fetching data from external sources (*Text files*).
Transform – In this process, several rules are applied on the fetched data for validation.
Load – The process of putting back the transformed data to a target location (*May be source text file or other text file*).

III. SYSTEM ARCHITECTURE

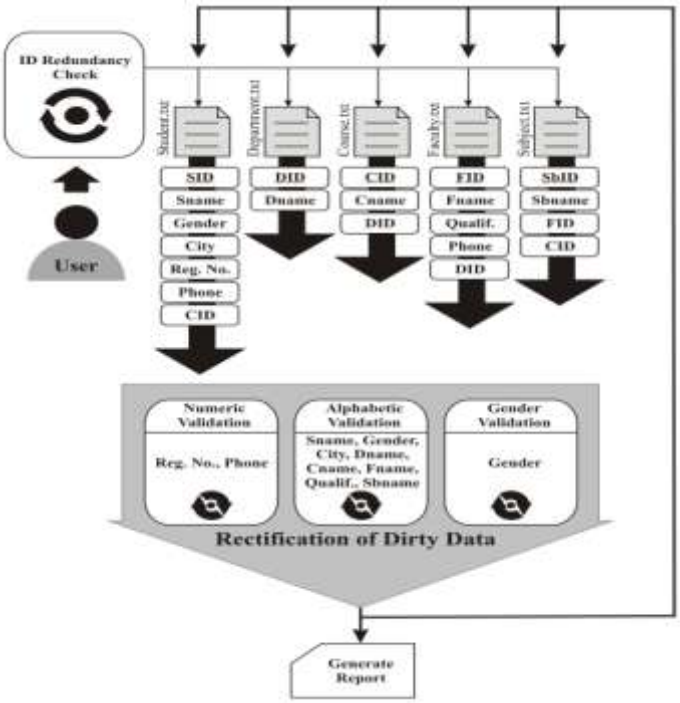


Fig 2: System Architecture

In order to implement this particular problem we have considered college information system as an example. The above figure (Fig 2) describes how our system works as a whole. Initially user is giving input in the user interfaces (UI). After ID validation (*At entry level we are checking the redundancy and pre-defined pattern of Id field. If the ID is redundant or empty then the system will prompt a message to the user to give a unique id. If the ID doesn't match the defined pattern then ID validation function generates the most nearest pattern with the error user input. The entry will not be submitted to the input text file until a valid Id is found*) the values are stored in corresponding text files namely student.txt, course.txt, department.txt, faculty.txt and subject.txt and fields are separated in the text files with proper delimiter (e.g. "|") and organized in a tabular format. Then each field value is extracted and distinguished in different categories in order to apply some cleaning processes such as numeric validation, alphabetic validation and gender validation on them as per requirement. Finally the system generates a report containing all modifications to the original files.

IV. TAXONOMY OF ERROR

Here we have classified the types of error that can occur in the input text file, those are:-

(1) Numeric Value in alphabetic token (e.g. Name, Gender, City)
(2) Alphabets in Numeric token (e.g. Phone no, registration no, Date)
(3) Invalid ID pattern (e.g. SID, CID, DID)
(4) Redundant ID
(5) Invalid gender [1]

V. RULES AND ALGORITHMS

(a) ID Validation Algorithm:
INPUT: String ID
OUTPUT: String finID
Step 1: Eliminate all alphabets from the ID.
Step 2: Concatenate ID with preceding Zero(s) according to following rules.

I. If length of ID equals to 1 then
ID = "00"+ID
II. If length of ID equals to 2 then
ID = "0"+ID
III. If length of ID greater than 3 then
Take only first 3 characters and eliminate the rest.

Step 3: Concatenate ID with preceding character according to following rules:

I. If ID is Student ID then
finID="S"+ID
II. If ID is Department ID then
finID="D"+ID
III. If ID is Course ID then
finID="C"+ID
IV. If ID is Faculty ID then
finID="F"+ID
V. If ID is Subject ID then
finID="B"+ID

Step 4: Return finID.
Step 5: END.

[Above algorithm only produces a valid ID pattern. To prevent redundancy of ID (*primary key*) we match all the ID's in the corresponding text file with the "finID" returned by the above algorithm at the entry time. If a match found, it ensures redundancy in ID field and system prompts a message to the user to renter a unique ID]

(b) Alphabetic Validation Algorithm:
Step1: Input String.
Step 2: Extract character while index=0 until string length.
Step3: If character is not an alphabet then check whether the pattern matches with
0, 5, $, &, @ , i, I, l
Transform:
0(Zero) --> o)
[5$&] --> s
@ --> a
[!1lI] --> i
Else
Remove the character.
Step 4: Index++. Go to Step 2.
Step 5: Return the Formatted string.
Step 6: End.

(c) Numeric validation Algorithm:
Step 1: Input String.
Step 2: Extract character while index=0 until string length.
Step 3: If character is not an numeric then check whether the pattern matches with o,O, i , I , l , ! , s , S ,

Transform :
[oO]-->0
[iIl!]--> 1
[sS]-->5
Else
Remove the character.
Step 4: if the input is phone number then
If Phone No length less than 9
Initialize Null to the string.
If Phone No length equals to 9
Append 0 in the end of the string.
If Phone No length greater than 10
Return the First 10 digit.
If Phone No length equals to 10
Take the whole string.
Step 5: Index++ . Go to Step 2.
Step 6: Return the Formatted string.
Step 7: End.

(d) Gender Validation Rules:

1. First go through Alphabetic validation.
2. If the return string starts with 'm' or 'M' then convert the string to "Male".
3. If the string starts with 'f' or 'F' then convert the string to "Female".

VI. USE CASE DIAGRAM

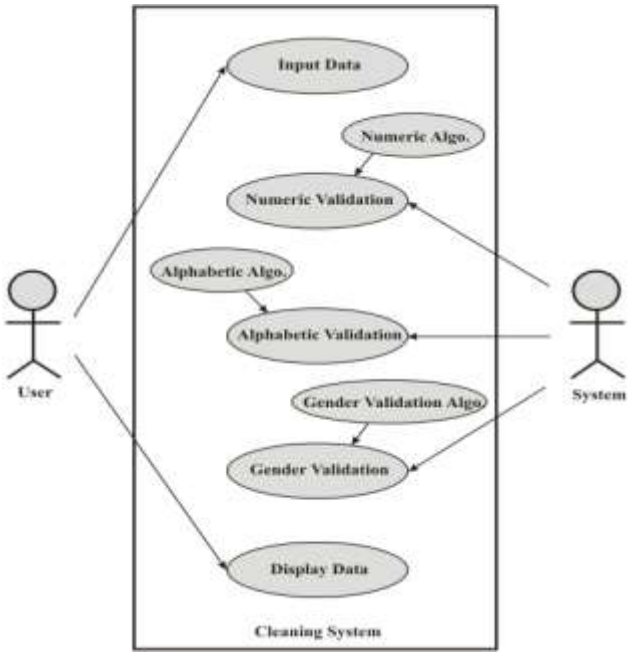


Fig 3: Use Case

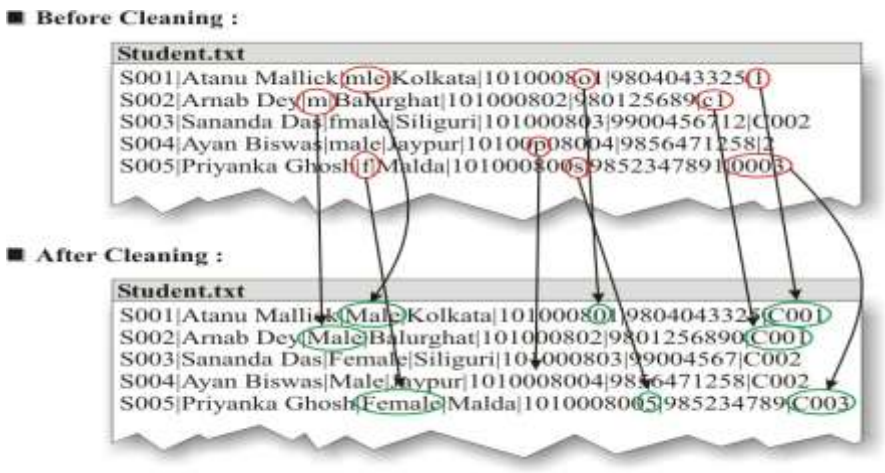
USE CASE represents the different ways in which a system can be used by the user. For our data cleaning process in text file the use cases are:

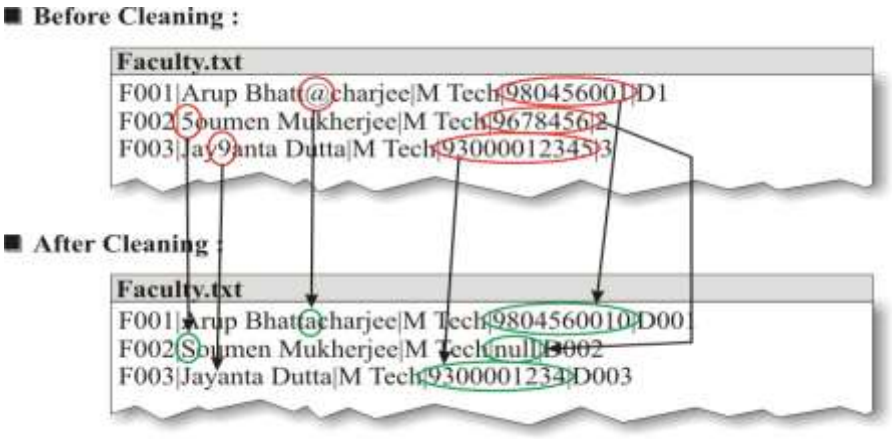
- Input Data
- Numeric Validation
- Alphabetic Validation
- Gender Validation
- Display Data

Here we have two actors:

- User
- System

VII. SAMPLE OUTPUT





VIII. FUTURE SCOPE

As of now, our algorithms have been tested on data with only few records. This could be tested on huge Enterprise Data that can give us better knowledge of Performance and efficiency of those algorithms. Here we have considered only ID pattern and redundancy error, alpha-numeric and invalid gender errors. We can improve this cleaning process by introducing Date validation, phonetic validation etc. System can be enhanced to incorporate better redundancy checking. Data Dictionary may be used to replace a field value with proper form.

IX. CONCLUSION

Incorrect and misleading data lead to all sorts of unpleasant and unnecessary expenses. In this paper we have proposed some algorithms for data cleaning in text file. It can detect data errors, programmatically create valid values and perform rectification of erroneous field values. The main advantages of this system are that, it is platform independent and no need of any DBMS software. Hence reduces space and cost overheads. There are some inadequacy still needs to improve. This cleaning process can be enhanced to clean any type of system.

REFERENCES

[1] R. Cody, "Data cleaning 101," Proceedings for the Twenty-Seventh SAS User Group International Conference. Cary, NC: SAS Institute Inc,2000.
[2] Dr. Mortadha M. Hamad and Alaa Abdulkhar Jihad, "An Enhanced Technique to Clean Data in the Data Warehouse". Computer Science Department. University of Anbar, Ramadi, Iraq.
[3] Hasimah Hj Mohamed, Tee Leong Kheng, Chee Collin and Ong Siong Lee, "E-Clean: A Data Cleaning Framework for Patient Data". School of Computer Sciences. University Sains Malaysia Penang, Malaysia.
[4] Arindam Paul, Varuni Ganesan, Jagat Sesh Challa and Yashvardhan Sharma, "HADCLEAN: A Hybrid Approach to Data Cleaning in Data Warehouses". Department of Computer Science & Information Systems . Birla Institute of Technology & Science, Pilani, Rajasthan, India – 333031.
[5] Erhard Rahm and Hong Hai Do. "Data Cleaning: Problems and Current Approaches". University of Leipzig, Germany.
[6] Srivatsa Maddodi, Girija V. Attigeri and Dr. Karunakar A. K, "Data Deduplication Techniques and Analysis". Manipal Institute of Technology, Manipal, India.
[7] R. Kimball and J. Caserta, "The Data Warehouse ETL Toolkit". Wiley,2004.
[8] V. Raman and J. M. Hellerstein, "Potter's Wheel: An Interactive Framework for Data Transformation and Cleaning.," in Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.
[9] K. Kukich, "Techniques for Automatically Correcting Words in Text", ACM Computing Surveys, vol. 24, no. 4, pp.377-439, 1992.
[10] R. Bheemavaram, J. Zhang and W. N. Li, "Efficient Algorithms for Grouping Data to Improve Data Quality", roceedings of the 2006 International Conference on Information & Knowledge Engineering (IKE 2006), CSREA Press, Las Vegas, Nevada, USA, pp. 149-154, 2006.