Enhanced Technique for 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, West Bengal University of Technology
Kolkata, West Bengal, India
www.rcciit.in

Abstract
Data cleaning is a process of correcting or removing of erroneous data caused by contradictions, disparities, keying mistakes, missing bits, etc to create consistent and reliable information. Text files are used to store simple information and which can be also deceptive in terms of dirty data. In this paper we have provided a solution to cleanup simple text file using some data cleaning processes. Though we use text files so often but there is no such robust method exist to clean up text files. As data cleaning plays a crucial role for decision management which is depend on high quality data. So we have implemented a set of methods to clean text files. Here we use text files to store data in tabular format and our system checks whether there exist any error and finally try to correct or remove the errors according to different algorithms.

Keywords: *ETL, Data Dictionary, Metaphone, Date Validation Rules, Gender Validation Rules.*

1. Introduction

At the beginning of our cleaning process, we fetch data from text file and then we apply several algorithms to rectify the erroneous record and after modification we put back the corrected data to the same source text file and may be kept in newly created text files as per user request. For implementing the process we have used **ETL** model (Extract, transformation and load) [1], [6], [7].

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.

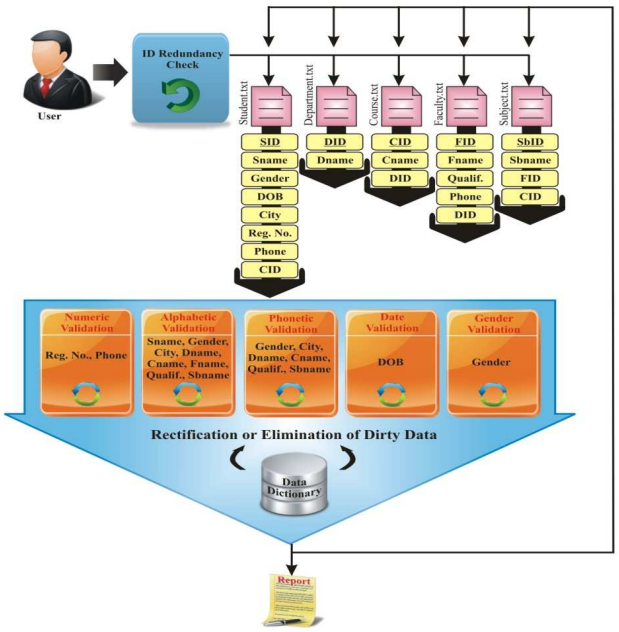
Load – The process of putting back the transformed data to a target location (May be source text file or other text file).

We categorize different type of error that can be occurred due to various reasons. And use respective rules to correct the data. Cleaning processes use various data dictionary (text file format) to match with the nearest correct data and to replace the erroneous data with the correct one. Finally, a report consist of detailed information about the rectified data along with the percentage of modification is being generated by the system.

As high quality data is essential for accurate data analysis and decision making, this data cleaning process ensures users to get correct and quality data. In our project we have used simple text file for keeping the information as it reduces the overhead of maintaining the storage and cost complexity of other database packages and making the system portable.

2. System Architecture

(Fig 1) describes the overall system working principal. Our system provides a user interface (UI) where user gives requested input. Here we have considered college information system as a demo process. First of all, system validates the ID field *(At entry level we are checking the redundancy and pre-defined format of Id fields. If the ID is redundant or empty, the system will request user for a unique ID. The entry will not be submitted to the input text file until user gives a valid ID, If user input an ID of improper format, system will try to rectify it into pre-defined pattern)* [1]. The data are extracted from the input text files *(student.txt, course.txt, department.txt, faculty.txt, subject.txt)* based on attributes and system categorizes them to process through some functions namely Numeric Validation, Alphabetic Validation, Metaphone Phonetic Validation, Date Validation and Gender Validation. These functions use different Data Dictionaries for valid data reference and the system finally generates a report containing all modifications to the original files.



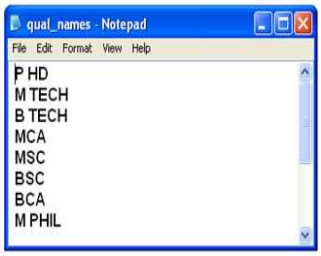
. **Fig 1: System Architecture [1]**

3. Data Dictionary

We have included some Data Dictionaries (DD) for some specific fields like city, qualification, course name, department name, subject name etc to rectify and replace with specified form or most nearest form for the erroneous data. To incorporate this transformation we first generate the phonetic code of the erroneous data and match with each phonetic code of each and every value in corresponding data dictionary. Two examples of such Data Dictionary are shown below.

DEPARTMENT DD

QUALIFICATION DD



4. Taxonomy of Errors

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

(1) Numeric Value in alphabetic token [1] *(e.g. Name, Gender, City)*
(2) Alphabets in Numeric token [1] *(e.g. Phone no, registration no, Date)*
(3) Phonetic error *(e.g. City, Course Name)*
(4) Invalid or Redundant ID pattern [1] *(e.g. SID, CID, DID)*
(5) Misspelling *(e.g. City)*
(6) Invalid date *(e.g. DOB)*
(7) Invalid gender [1], [2]

5. Rules and Algorithms

(A) ID Correction Rules: In this ID correction rules we assume that the id will be always of 4 characters *(e.g. S007 for student ID, C276 for Course ID).*

Step 1: Eliminate all alphabets from the given string id.
Step 2: If length of id equals to 1 or 2 then we will add "00", "0" in front of the number respectively. In all other cases we will take only first three digits.
Step 3: After that, for the case of Student, Department, Course, Faculty id we will add S, D, C, F respectively in front of their ids. (e.g. for Student ID - S001, S002. For faculty -F001, F002. For Course - C001, C002 etc)
Step 4: After transformation if cause any redundancy, request user for unique one.

(B) Alphabetic Rules:

Step 1: Extract each character from the given string.
Step 2: Check whether the character is an alphabet or not. If it is digit, then check whether it matches with 0, 5, $, &, @ , i, I, l.
Step 3: If matches with the above digits, then we will transform those digits into the resembling character. If it is 0(Zero) then transform into O, if 5 or $ or & then transform into S, if @ then transforms into A, if ! or 1 or l or | then transform into I.
Step 4: In case of all other characters remove them all from the given string.

(C) Numeric Rules:

Step 1: Extract each character from the given string.
Step 2: Check whether the character is numeric or not. If it is alphabet, then check whether it matches with o, O, i, I, l, !, s, S..
Step 3: If matches with the above alphabets, we will transform those digits into the resembling digit. If it is 'o' or 'O' then transform into 0(Zero), if 's' or 'S' then transform into 5, if '!' or 'I' or 'i' or '|' then transform into 1(one).
Step 4: In case of all other character remove them all from the given string.
Step 5: In case of Phone number first we check the length of the number.
(1) If Phone No length less than 9 then initialize Null to the string.
(2) If Phone No length equals to 9 then append 0 in the end of the string.
(3) If Phone No length greater than 10 then return the First 10 digit.
(4) If Phone No length equals to 10 then take the whole string.

(D) Metaphone Phonetic Rules:

1. Drop duplicate adjacent letters, except for C.
2. If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter.
3. MB → B only if MB at the end of word.
4. CIA→ X; CH → X; SCH→K; C[IEY] → S; Otherwise C→K
5. DG[EIY] → J Otherwise D→T
6. Drop 'G' if followed by 'H' and 'H' is not at the end or before a vowel. GN -> N; GNED→NED and is at the end.

7. 'G' transforms to 'J' if before 'I', 'E', or 'Y', and it is not in 'GG'. Otherwise, 'G' transforms to 'K'.
8. Drop 'H' if after vowel and not before a vowel.
9. CK → K
10. PH → F
11. Q → K
12. SH → X; SIO → X; SIA → X
13. TI[AO] → XI[AO] ; TH → O ; TCH → CH
14. V → F
15. 'WH' transforms to 'W' if at the beginning. Drop 'W' if not followed by a vowel.
16. Drop 'Y' if not followed by a vowel.
17. Z → S
18. Drop all vowels unless it is in the beginning.

(Here → implies transformation from left side to the right hand side)

(E) Date Validation Rules:

We have specified our pre defined date format as dd/mm/yy. We assume year is in the last portion of our input always and the age of the student is greater than 12. Following steps summarize most of the rules from the original implementation in our project.

Step 1: Input Date.
Step 2: The system will take only the input string containing minimum 6 digits and it will check whether the use has given delimiters or not. Otherwise system will return null.
Step 3: We parsed the input string into three parts- day, month, year respectively; which is separated by delimiter.
Step 4: If length of the year string is equals to 4 then we extract last 2 characters and if the length of the year string is equals to 2 then we extract the year string.
Step 5: We convert the year string into integer, and check whether the age is greater than 12 or not. If the year is in between 1900 and 1999 then it will execute following steps otherwise it will return nothing.
Step 6: If user gives the month input using name of the month instead of numbers, the system will convert the month in system defined format even if the month string starts with or ends with few characters. Those rules are given below

Months	Starts With	Ends With	Replace With
January	Ja	nuary	01
February	F	ruary	02
March	Mar	ch	03
April	Ap	il	04
May	M[aeiou]y	ay	05
June	Jun	ne	06
July	Jul	uly	07
August	Au	ust	08
September	S	tember	09
October	O	ober	10
November	N	vember	11
December	D	cember	12

Step 7: Then we will check for different condition for the day and month field. It can be in dd/mm/yy format or mm/dd/yy format.
Step 8: Here if the day is greater than 31 then first we are reversing the number and then check whether the number is greater than 31 or not. If not, then we will consider the value. (*Our system considers that, it may be possible that the user can give wrong input for certain reasons*)
Step 9: Check for the April, June, September, November month. Those months cannot exceed 30 days if user gives wrong input as 31. Then our system will consider as typing mistake and will give value of 13.
Step 10: Check for February month. And if the year is leap year then only the month can have 29 days. For normal year the system won't take day as 29.
Step 11: After all checking, the system will give input in dd/mm/yy format. If none of the rule satisfies the user input then system will return null.
Step 12: End.

(F) 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.

.

6. Class Models

We have modularized our project through some classes for ease of understanding. Our system has two set of classes.

1. Classes that represents the input text files *(Student, Department, course, Faculty, Subject).*
2. Validation Classes *(Date, Gender, Numeric, Alphabetic, Phonetic).*

Whenever system retrieves a tuple from a text file, a new object of corresponding class gets created, populating each property with the respective field values of that tuple. Now related validation objects *(instantiated validation classes)* takes this object as input and validate properties and finally

return back the object with rectified field values which can be inserted into output text files.

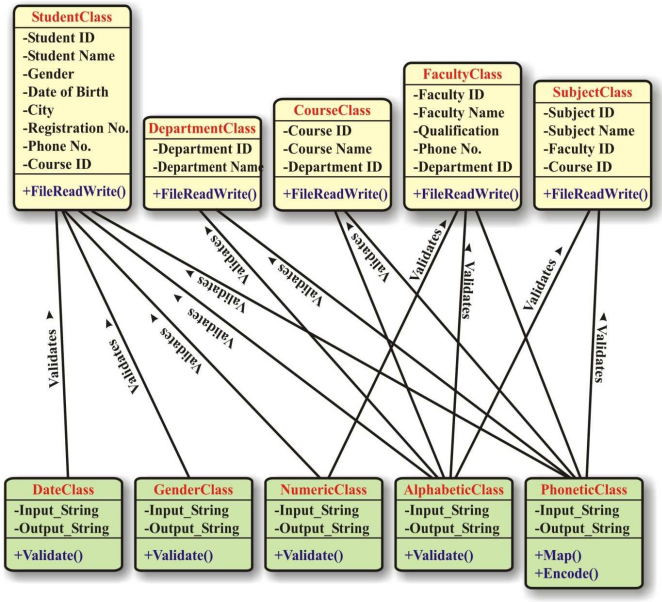
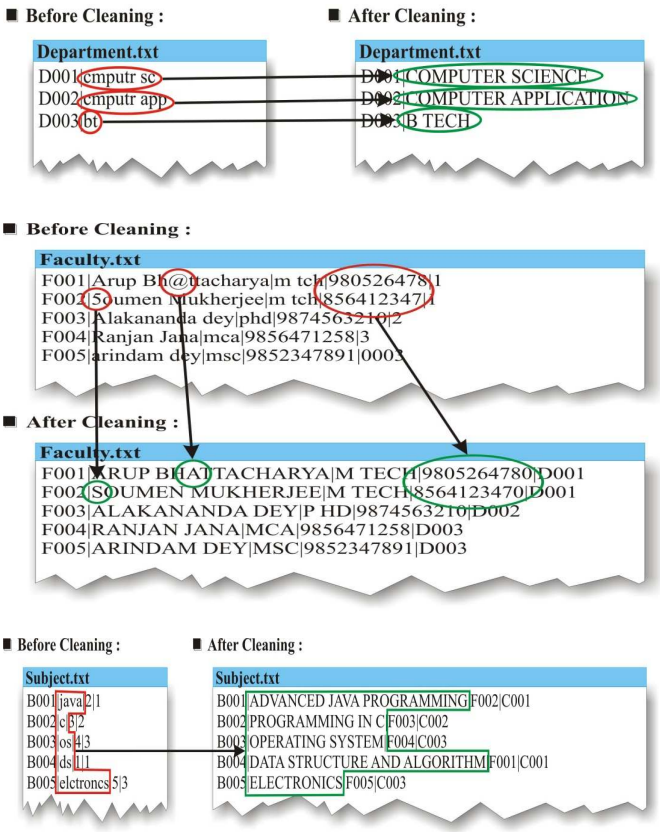
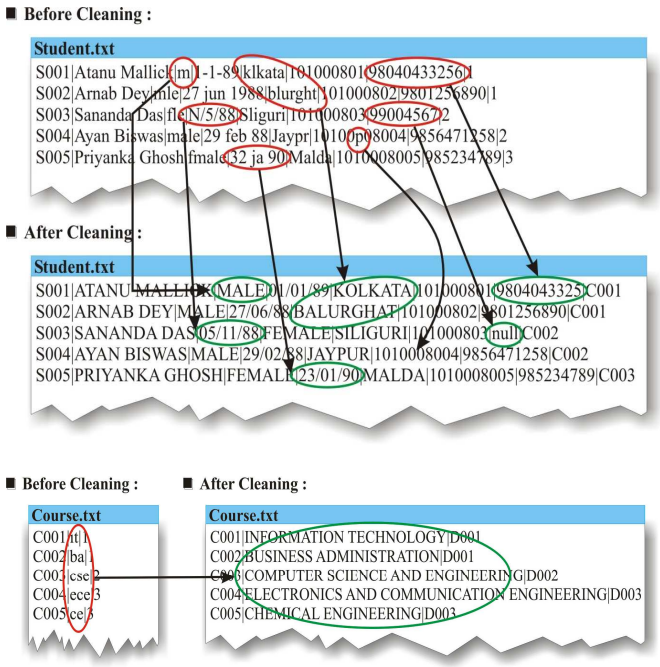


Fig 2: Class Models

7. Sample Outputs





8. Future Scope

Here we have used text file in tabular format, it is possible to correct errors without the tabular structure. In this paper we have used Metaphone (phonetic algorithm) for pronunciation of word; it can be implemented by double Metaphone and Metaphone 3 algorithm which are latest algorithms. Here we have improved the knowledge base according to our requirements, anyone can change that for their own suitable condition. Our algorithms for ID, Alphabetic, Numeric, Date, Gender and Phonetic validation can be improved or replaced depending upon situation arises.

9. Conclusion

Our data cleaning framework preserves the quality error free data in text file. Data accuracy is very hard to achieve through data-cleansing in the general case, because it requires accessing an external source (Data Dictionary) of data that contains the true value. So our approach is quietly based on this predefined knowledge base which can be improved further with a best possible outcome of true value for the erroneous data. As text files are used massively it can flaw a decision making process. So by enhancing the process of data cleaning in text file we can resolve the situation and can be later used as analytics software for decision making or removing dirty data in text file.

References

[1] Arup Kumar Bhattacharjee, Atanu Mallick, Arnab Dey and Sananda Bandyopadhyay, “Data Cleaning in Text File”, IOSR Journal of Computer Engineering (IOSR-JCE), ISSN: 2278-0661, Volume 9, Issue 2 (Jan. - Feb. 2013).

[2] R. Cody, “Data cleaning 101,” Proceedings for the Twenty-Seventh SAS User Group International Conference. Cary, NC: SAS Institute Inc,2000.

[3] 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.

[4] 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.

[5] 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.

[6] Erhard, Rahm and Hong Hai Do. “Data Cleaning: Problems and Current Approaches”. University of Leipzig, Germany.

[7] Srivatsa Maddodi, Girija V. Attigeri and Dr. Karunakar A. K, “Data Deduplication Techniques and Analysis”. Manipal Institute of Technology, Manipal, India.

[8] R. Kimball and J. Caserta, “The Data Warehouse ETL Toolkit”. Wiley,2004.

[9] 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.

[10] K. Kukich, “Techniques for Automatically Correcting Words in Text”, ACM Computing Surveys, vol. 24, no. 4, pp.377-439, 1992.

[11] 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.

Author Biography

Arup Kumar Bhattacharjee received his MCA Degree from University of Kalyani and his M.Tech from West Bengal University of Technology. He has contributed to 15 books and coauthored 2 publications. He is an Assistant Professor of Computer Application at RCC Institute of Information Technology which is affiliated to West Bengal University of Technology in Kolkata, West Bengal. His research interests include Software Engineering, Object Technology and Parallel Computing.

Atanu Mallick has completed his Bachelors in Computer Science (*Hons.*) from Surendranath College under Calcutta University in Kolkata, West Bengal. Currently he is pursuing his Masters in Computer Application from RCC Institute of Information Technology which is affiliated to West Bengal University of Technology in Kolkata, West Bengal, India.

Arnab Dey received his Bachelors degree in Computer Application from Pailan College of Management & Technology, Kolkata under West Bengal University of Technology. Currently he is pursuing his Masters in Computer Application from RCC Institute of Information Technology which is affiliated to West Bengal University of Technology in Kolkata, West Bengal.

Sananda Bandyopadhyay has completed her Bachelors in Computer Application from Techno India (salt lake) under West Bengal University Of Technology. Now she is pursuing her Masters in Computer Application from RCC Institute of Information Technology which is affiliated to West Bengal University of Technology in Kolkata, West Bengal.