A Novel Framework and Model for Data Warehouse Cleansing

Daya Gupta
Computer Engineering Department
Delhi Technological University (Formerly Delhi College of Engg.), New Delhi, India

Payal Pahwa
Computer Engineering Department
BPIT, Guru Gobind Singh Indraprastha University
New Delhi, India

Rajiv Arora
Ph.D Scholar, Computer Engineering Department
Delhi Technological University (Formerly Delhi College of Engg.), New Delhi, India

ABSTRACT
Data cleansing is a process that deals with identification of corrupt and duplicate data inherent in the data sets of a data warehouse to enhance the quality of data. 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' attribute of the data sets of an already existing data warehouse. The key features of the research includes its proposal of a novel framework through a well defined sequence of algorithms and refining the application of alliance rules [1] by incorporating the use of previously existing and well defined similarity computation measures. The results depicted show the feasibility and validity of the suggested method.

Keywords
Data warehouse, data cleansing, fuzzy logic, data mining.

1. INTRODUCTION
A data warehouse is a subject-oriented, integrated, non - volatile and time- variant collection of data in support of management's decisions [2]. It is the central point of data integration for business intelligence. An enterprise generally runs numerous operational systems which is a major source of data, together with external agencies, for its data warehouse. These operational systems store data by individual applications and these applications work more or less on similar entities. For example, in a telecommunication company, applications like customer billing system , customer credit verifying system, customer personal information management system etc. would have many attributes in common, for example - customer name, addresses , but these common attributes may or may not be represented in exactly same formats in their associated databases. In such a scenario, when data is integrated for storage in a data warehouse, unintentional duplication of records created from millions of data entries from heterogeneous sources can hardly be avoided. This leads to redundancies in the data and degradation of its quality.

Data Warehouse suffers from dirty data. Dirty data can be understood as the data which is not consistent with already residing data of data warehouse [3,4]. For e.g. missing data can be considered as dirty. The data that is unknown or not clear at the time of data entry is also considered dirty data. Data warehouses, after their construction by merging and amalgamation of numerous operational systems, are also updated periodically to store/accommodate new data. The possibility and probability of dirty data induction increases with more and more regular updates. Even the most carefully and cautiously planned updating cannot give a full-proof protection from dirty data [5].The presence of dirty data leads to many serious problems because data warehouse is used for decision making and strategic planning of an organization.

Data cleansing is the most obvious and also, the only viable solution that can address these problems. Cleaning of data is an activity which detects redundancy or unwanted data and corrects it for increasing the quality of the data .The need, therefore, is of an automated data cleaning tool [6] with capabilities of maintaining as well as improving data quality in the data warehouse. It also requires general framework in [7] for data cleaning process as well as some methods that can be used to address the problem like statistical outlier detection, pattern-matching, clustering and data mining techniques.

'Name' field as given in [8] is one of the most common attribute which is used in query processing. Finding and matching personal names is at the core of an increasingly large no. of applications: from text and web mining, information retrieval, search engines to de-duplication and data linkage systems. A large amount of data in most warehouses pertains to people. Hence, personal names are often encountered being used as search criteria thus, the significance and importance of the personal name field cannot be stressed too strongly. The problems in the name field arises because of several reasons ranging from spelling variation for personal names, use of nicknames, change of names with time etc. It also presents an overview of a comprehensive no. of name matching techniques and their experimental results.

In some cases, it is not easy to find out whether a name variation is a different spelling as in [9] of the same name or a different name altogether. There are various types of variations which are:

• Spelling variations: These generally include interchanged or mislaid script due to typographical errors, substitute letters (as in Rajiv and Rajeev), supplementary letters (such as Smythe).Basically such variation does not influence the phonetic configuration of the name but then also causes problems in respective names. These variations mainly

come from mis-reading or mis-hearing by a human or an automated machine.

• Phonetic variations: Where the alphabets of the name are altered, e.g. through mis-hearing. The structure of the name is basically altered.

• Double names: There are some cases where the last name consists of two elements but both are not always shown. For example, a double surname such as Raj -Mohan may suggest in full, as Raj or as Mohan.

Alliance rules based on mathematical association rules are proposed in [1] which detected duplicity in the name field of the data warehouse. It uses the score matching mechanism for calculating the duplicity.

The above described approach considers an arbitrary threshold value of 50% for confidence when none of the scores matched. However, the present research work does not rely on arbitrary threshold values. Rather, it introduces the concept of fuzzy match similarity (FMS) [10, 11] for identification of non-identical duplicates through a more reliable, logical and well-calculated approach. Another major feature of our work is that we have developed a totally novel framework for name-field matching. The framework is simple to understand and yet, it is robust and flexible enough for modeling a potential data cleaning tool or being a part of such a tool.

The layout of the paper is as follows: Section (II) contains literature review. A novel framework for duplicity identification in the name field is proposed in section (III). A sequential algorithm defining various phases of the framework is defined in Section (IV). Section (V) includes Implementation details. Section (VI) is the concluding section.

2. LITERATURE REVIEW

The issues raised in [12] related to data quality emphasizing their importance and impact respectively. It also explains the data quality problems related to merging and investigate the relationship between data quality and consistency.

In [13] A classification of data quality problems and their solutions in data sources differentiating between single and multiple source and between schema and instance level problems have been provided.

[14] presents a survey of data cleansing problems, approaches and methods. It classifies various types of anomalies occurring in data (like syntactical, semantically and coverage) that have to be eliminated and also define a set of quality criteria that comprehensively cleansed data has to accomplish . It also evaluates and compares existing approaches for data cleaning.

As suggested in [15],the goal of data cleansing is not just to clean up the data in a database but also to bring consistency to different sets of data that have been merged from heterogeneous databases.[3,4,6] analyzes the problem of data cleansing. It provides an independent domain tool for data cleansing. For this, application of association rules are also used for removing redundancy from data sets.

Duplicates tuples are removed from dimension tables of data warehouse as in [16] by using high scalable, quality and efficient algorithm so that cleansed data is obtained.

In [12], Phonex algorithm is used for name matching i.e. which attempt to find name spelling variations.[16] presents a highly comprehensive survey of the existing techniques used for identification of non-identical duplicate entries in database records. It also discusses similarity metrics that are commonly used to detect similar field entries and presents an extensive set of duplicate detection algorithms.

All of the above proposals defined the errors and explains the matching from reference tables. In [1], an obscure involvement of the q-gram concept in the name matching has been given. It did not clearly define how Q-grams were actually used in matching or identifying non-identical name duplicates. It also assumed an arbitrary threshold of 50% for confidence in alliance rules. i.e. if 50% letters did not match the word was considered an outlier value.

In this paper, we overcome the shortcomings and limitations of the previous work and improve it to take data cleansing to next higher level. We have clearly defined as well as refined the exploitation of the q-gram concept. We do not assume arbitrary thresholds rather we have incorporated reliable FMS [10,11] techniques and existing similarity measure for forming the basis of accepting or discarding strings (name field) as duplicates or unique values. The newly proposed and enhanced algorithm has the potential of giving more reliable, logical and well-calculated results. We have proposed our ideas through a novel and robust framework which has the flexibility and potential to fit in the model of a future powerful data cleaning tool.

3. FRAMEWORK DESIGN

The proposed framework consists of following phases whose flow and sequence could be understood through figure 1.

- Preprocessing
- Alliance rules application
- Transformation cost calculation
- Token/q-gram formation
- Weight assignment
- Similarity computation

In preprocessing, Name field is converted into unique number (called scores) of first and second data mart. This number is passed to rules application field where scores are compared with reference table .If Last name scores match then results is further saved in one cluster otherwise stored in second cluster and give this output to FMS for calculation of duplication using Transformation cost ,Token q-gram formation and weight assignment.

The steps for proposed work are as follows:

1. FRAMEWORK DESIGNING

The various stages have been morphed to a framework so as to facilitate in the development of data cleansing tools.

2. ALGORITHM FOR EACH PHASE

An algorithm has been proposed for each phase and explained properly through flow charts and diagrams.

3. USE OF FUZZY MATCH SIMILARITY

It includes incorporation of string comparison measures [17,18] using Fuzzy Match Similarity (FMS) [10,11] to find duplicity in the name fields.

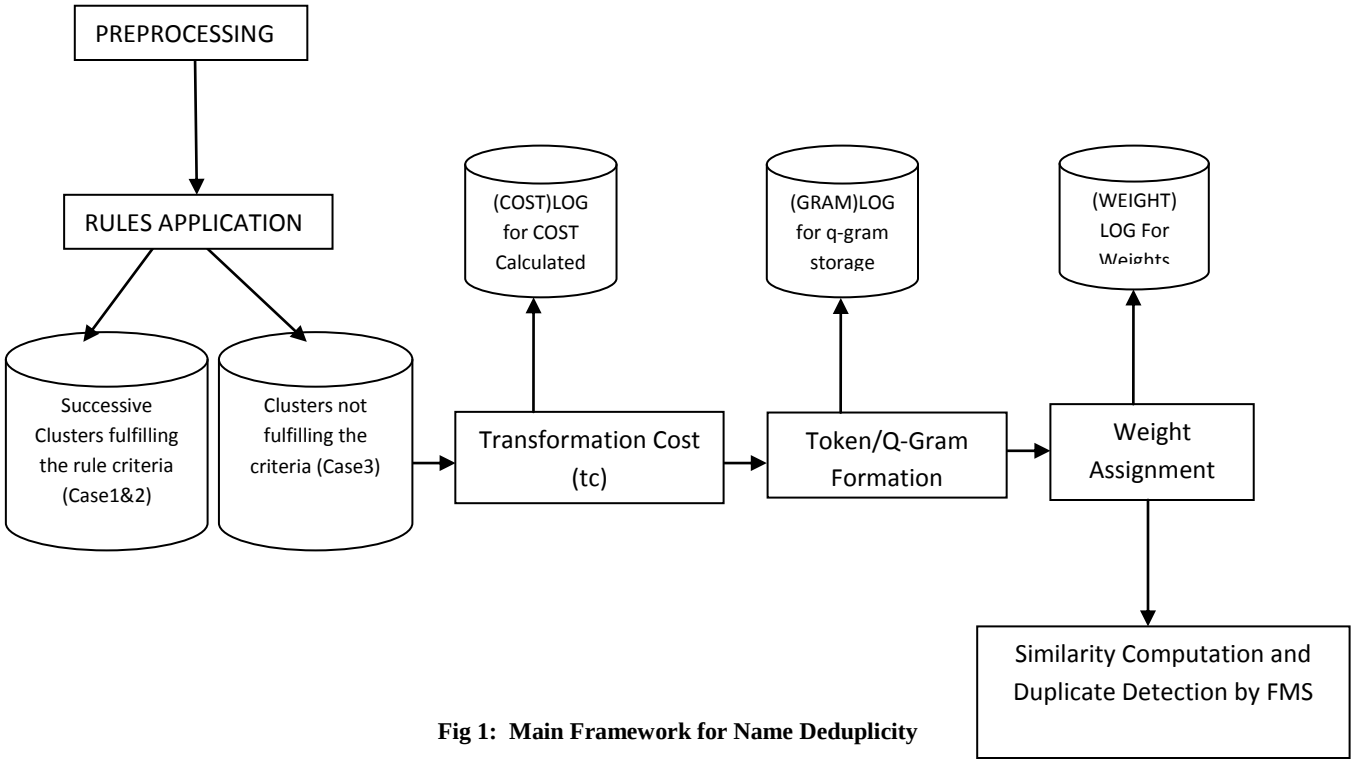

Fig 1: Main Framework for Name Deduplicity

4. PROPOSED ALGORITHM

The various phases as shown in Figure 1 of the framework are defined as follows –

4.1 Preprocessing

The algorithm for preprocessing is as follows in fig.2:

Preprocessing
INPUT: Input data sets, Reference Tables from Data Marts
OUTPUT : Score File
Begin
1 .For each name in the input data set (1…n)
a. Calculate the no. of words 'N'
2. for each individual word as well as the combination of initials
a. Calculate 'scores' using the relation
[(radix) ^ (place value)* face value] mod m
End for
End for
3. Store the *scores* in Score file.

Fig 2: Preprocessing

This step involves the conversion of string data type (name fields) to numerical values. The data sets taken from input data marts (say DM1) and reference tables' data marts (say DM2) consists of name fields which are converted to numerical integer values. The domain D is defined for integers which provide domain boundaries and help in providing a solution for error detection or specifically duplicity detection that is domain independent.

A string (name) is converted to numbers using the relation [(radix) ^ (place value)*face value] mod m where radix is defined as a set of 27 characters (26 alphabets +special symbol'.'(Period)).The letters are taken as case-insensitive. The face value of each character is marked by the sequence number with which they arrive in alphabetic order starting with 0 –a ----- 25-z and 26 - (.) and m is any large prime number. The place value of letters is marked from right to left starting with 0. The converted integer values are stored in a file called Score and are called scores of the name. The conversion is done by first evaluating the total number of words in a name say, John F. Kennedy has 3 words in its name. The number of scores that would be calculated for a name with N words is N+1. The N+1 scores consists of N scores for each word in the name and N+1th score for the initials of the name. This step has the advantage that it makes the comparison domain-independent. All these evaluated scores for each name corresponding for each word in the name is stored in tabular form in the Score file. The calculation of score is illustrated with the following example in fig 3.

Calculate SCORES using formula
[(radix) ^ (place value) * face value] mod m
Let m= 731 (AJEET)
Score=[0*(27)^4+9*(27)^3+4(27)^2+4*(27)^1+19*(27)^0]mod 731= (0+177147+2916+108+19)mod 731=180190 mod 731=364 [This is the score of the word AJEET (in both cases -upper and lower)]

Fig 3: Score Calculation

The preprocessing step involving conversion to integer scores has the benefit that it enables the maintenance of domain independency and achieving advantages of memory

concerns also. This also allows for statistical analysis if needed (On conversion of strings into numbers, statistical analysis can also be done).

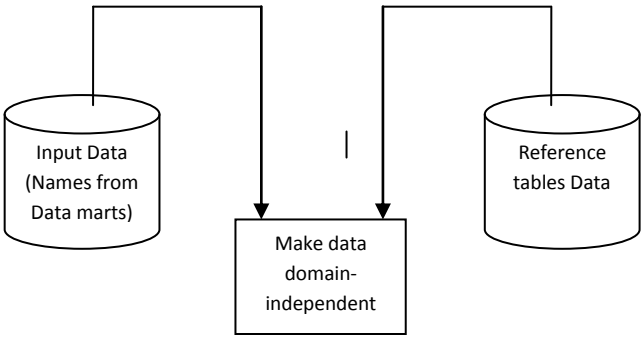

Fig 4: Flow diagram for preprocessing.

4.2 Alliance rules application

The algorithm for detecting duplicity in the name field [1] of the data warehouse is as follows in fig 5:

Alliance rules application
INPUT: Score File
OUTPUT: Data sets with errors and without errors
Begin
1. for each name in DM1
2. Compare Sn(last name score of DM1)(input data set) with all Sn's of DM2(reference table) where Nr=Ni i.e. no. of words in reference names= no. of words in input name
3. Store rows with same Sn(last name score) in Log file
4. Compare S1 of each name with S1's of all rows of LOG file
Case 1: Perfect Match
4.1 If single entry
NO ERROR
4.2 If multiple entries
for i=2 to n-1
4.2.1. Compare Si of DM1 with Si of all rows of LOG
4.2.2. Store results in successive Log files.
1. If single entry Then NO ERROR
2. If multiple entries
Check (DOB+Address)
3. If match Then ERROR
Else OK
Case 2: No Match
4.3 If no entries in Log file
Compare Sn+1 of DM1 with Sn+1 of DM2
4.3.1 If match Then Check (DOB+Address)
Case 3: None of score matches from Sn to Sn+1
4.4 If entry exists with errors in name

Fig 5: Alliance rules application

The alliance rules defined in Section V are used for error identification and detection in the data warehouse. These Rules provide a rule based criteria to identify the errors of duplicity of data of string type in data warehouse in different data marts. The name fields of the data sets taken from the input data marts (DM1) are checked and matched for duplicity with all names in the reference tables in other data marts (DM2) on the basis for alliance between scores of names calculated in preprocessing stage. The rules to be followed for this checking, matching and detection of duplicates are guided by the data alliance rules. The duplicity between names is calculated by first matching the last name scores 'Sn' of each name of input data mart DM1 with all names in the reference tables' data mart DM2 where no. of words in reference names(Nr) is same as no. of words in input name(Ni). This results in further reduced cluster of names that has same score value of Sn. This cluster is stored in a file called scoring file 1. Now, matching of the first name score 'S1' is done for further reducing the cluster which is stored in another file called scoring file 2.

Now, there exists three possible cases for 'S1' score matching (refer fig 4). This score matching finally helps in detecting the duplicates. Till now Sn has been matched and using S1 the further algorithm will be followed for concluding the duplicity detection.

Case 1:-Perfect Match

Single entry match: In case of single entry, there is no duplicity and hence no error is detected.

Multiple entry matches: In this case, match other scores of the name which are S2, S3 up to Sn-1. If a single entry is resulted out by matching all available scores then there is no error. But, if multiple entries are skimmed out, then check the value for (DOB + Address). If the value matches then it's the same person and data set is infected with duplicate records.

Case 2:-No match

If S1 score does not match then match the score of initials i.e. Sn+1. Now, this can result in two conditions:

a. Same person
b. Different person with same initials

In this case, check the value for (DOB + Address). If the value matches then it's the same person and hence error is identified as duplicity due to different name formats in two data marts. If the value does not match then no error exists.

Case 3:- None of score matches from Sn to Sn+1

This can be due to two reasons:

a. That entry does not exist.
b. Entry exists with some errors in name

For entry existing with some error in names, concept of fuzzy match similarity (FMS) based on q-grams is exploited. This is a more refined, elaborate and well-calculated approach for matching non-identical duplicates than the previous one. Application of FMS further enhances and improves the previous approach. The clustering based approach reduces the no. of comparisons in successive steps which enhances the performance by saving time.

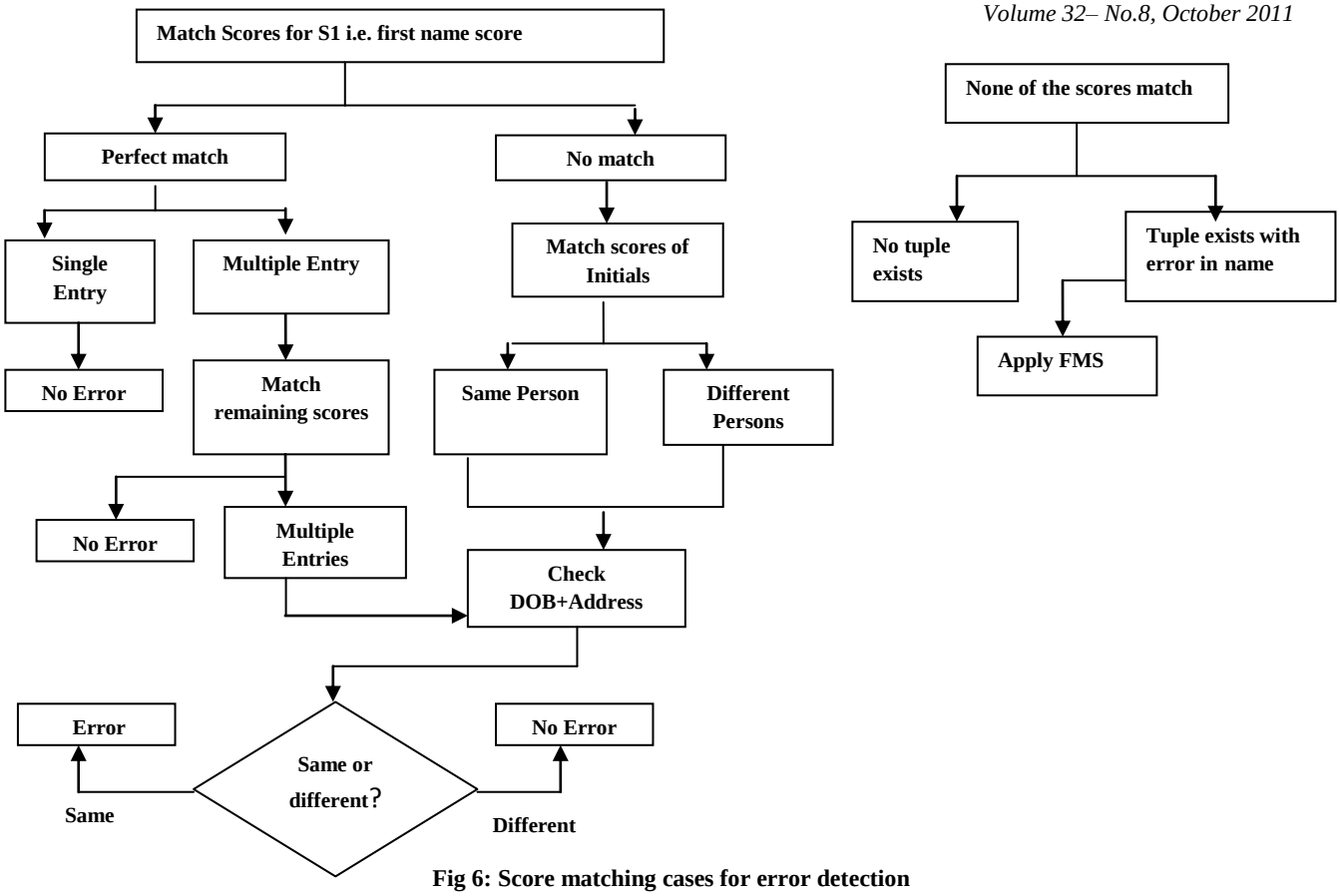

Fig 6: Score matching cases for error detection

5. TRANSFORMATION COST (tc) CALCULATION

The algorithm for transformation cost is as follows in fig 7:

Transformation cost calculation
INPUT : Strings from Input data sets and Reference Tables from Data Marts (say DM1 and DM2 respectively).
OUTPUT : Transformation Cost (tc)
Let ni= names from input data set
nr= names from reference tables
Begin
1. for each ni and nr
2. tc(ni, nr)
Number of matches/max (| ni |, | nr |)
End for

Fig 7: Transformation Cost Calculation

The idea behind this phase is to take into account those strings which are duplicates but not apparently. For if they can be transformed to another string that is under consideration within some reasonable cost then they could be considered as potential duplicates. A variety of string distance metrics can be found in literature [8, 18, 19]. These step calculations the cost of transforming a string (in this context, names) to another which is referred as the transformation cost. This step presents the algorithm that calculates the cost required to transform one string to another based on the number of matches between the string and the maximum length out of the two strings.

6. TOKEN/Q-GRAM FORMATION:

The algorithm for token/Q-gram formation is as follows in fig 8:

Token/ q-gram formation
INPUT : Strings from Input data sets and Reference Tables from Data Marts (say DM1 and DM2 respectively).
OUTPUT : Array GRAM containing all q-grams
Begin
1. for each ni
2. For (i=1 to |ni|-q+1)
3. Store q=grams in GRAM
End For
End For

Fig 8: Token/Q-Gram Formation

Q-grams, also called n-grams [8, 10, 18], are sub-strings of length q in longer strings. The length q can be any value such that it is smaller than the length of string. Commonly used q-grams are unigrams (q = 1), bigrams (q = 2, also called diagrams) and trigrams (q = 3). For example, 'peter' contains the bigrams 'pe', 'et', 'te' and 'er'. A q-gram similarity measure between two strings is calculated by counting the number of q-grams in common (i.e. q-grams contained in both strings) and divide by either the number of q-grams in the shorter string (called Overlap coefficient2), the number in the longer string (called Jaccard similarity) or the average number of q-grams in both strings (called the Dice coefficient). This algorithm calculates q-grams of the strings and stores the result in an

array GRAM. The value of q=3 is a reasonable one and considered most appropriate since it is neither too long nor too short.

7. WEIGHT ASSIGNMENT

The algorithm for weight assignment is as follows in fig 9:

Weight assignment
INPUT : Array GRAM containing the q-grams
OUTPUT : Array Weight containing the weight of each string
Begin
1. for each q-gram in GRAM
2. Calculate weight (Wi) using any one if the mentioned three methods
End for
4. Store the weights in array WEIGHT
5. for each ni
6. Total Weight (W) is:
End for

Fig 9: Weight Assignment

This algorithm assigns weight to the q- grams calculated in previous phases. There are various methods available for weight assignment, namely: Inverse Document Frequency, Overlap coefficient, Jaccard Similarity [8, 18]. Depending on the name characteristics and their origin in the context, choices are made for best results. The output of this phase is an array weight storing the weights of each string. The idea behind this algorithm is that the weights assigned indicate the significance of each q-gram and taking into consideration that significance duplicity between the names is detected.

The three popular approaches for weight assignment allocate weight according to the following relations:

1. Overlapping Coefficient:
Overlapping Coefficient (O.C.) = (No. of matching q-grams)/ (No. of q-grams in shorter string)

2. Inverse Document Frequency (IDF)
IDF= (Total no. of strings)/ (Strings with that q-gram)

3. Jaccard Similarity
J = (No. of Matching q=grams)/ (No. of q-grams in longer string).

8. SIMILARITY COMPUTATION BY APPLICATION OF FUZZY MATCH:

The algorithm for similarity computation by application of fuzzy match is as follows in fig 10:

Similarity computation based on fuzzy match
INPUT : Array WEIGHT, Array GRAM
OUTPUT : Array FMS
Begin
1. For (each ni and nr)
2. FMS = 1- min (tc (ni, nr)/Weight (nr))
End for

Fig 10. Similarity computation based on fuzzy match

While validating the incoming tuples against reference relations consisting of known-to-be-clean tuples, it is a common scenario that input tuples fails to match exactly with any tuple in the reference relations. One possible reason is that no entry exists corresponding to the input name. Another possible reason which is also more common is due to errors in the input tuples. In such a scenario, we apply a technique called Fuzzy Matching which is defined as an error resilient matching of input tuples against the reference tuples[11]. In our paper, we incorporate an efficient FMS approach for fuzzy matching which takes into consideration the problems encountered in traditional fuzzy match where due importance is not given to various tokens while checking for similarity. Our approach views the string field as a sequence of q-grams (tokens) by explicitly associating weights (through step E) and quantifying their importance. Our notion of similarity between 2 names depends on the minimum cost of transforming one name into another through a sequence of transformation operations where the cost of each transformation operation is a function of weights of q-grams/names involved.

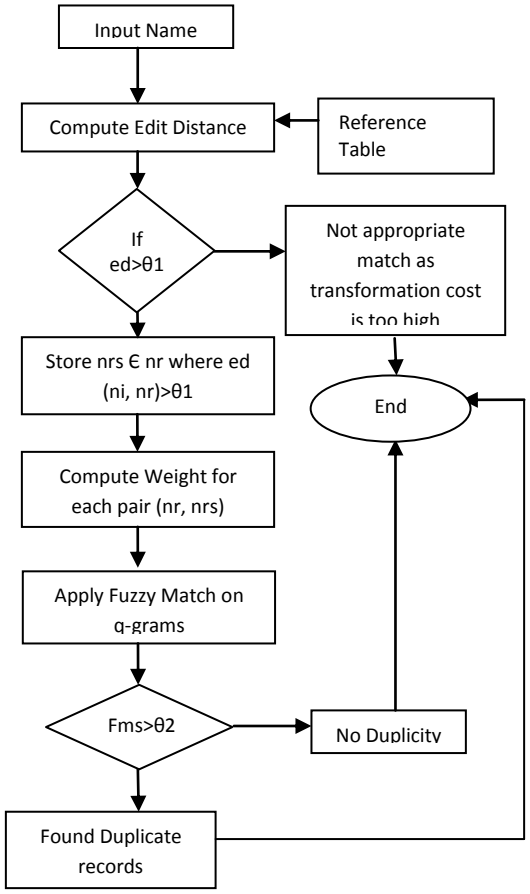

Fig 11: Flow diagram for detection of errors through phases C, D, E and F

This algorithm calculates the FMS value for each string. If FMS values calculated is greater than certain threshold, strings can be considered as duplicates. A major advantage of applying Fuzzy Match Concept is that the strings which are not exact duplicates are also considered[11.16] .The flow diagram (fig 11) above illustrates the application of fuzzy match based on q-grams for similarity matching and duplicate identification.

9. COMPARISON OF PREVIOUS APPROACH AND PROPOSED APPROACH

Table 1: Comparison

Approach / Attribute	Previous	Proposed
Reliability	Low	Comparably high
Accuracy	Low	Comparably high
Efficiency	Low	Comparably high

10. IMPLEMENTATION

Prototype has been developed for pre-processing in which scores are calculated of reference data mart. Mat Lab has been used for implementation of Alliance rules application. We have used the real data sets (1000 tuples) of telecommunication industry. The final result of the algorithm is calculated using fuzzy match similarity function fms (u, v). We define the fuzzy match similarity function, fms (u, v), between an input tuple u and reference tuple v in terms of the transformation cost and weight metrics as follows [16]:

fms (u, v) = 1 - min [(tc (u, v) / w (u), 1.0]

Given an input tuple, u, the goal of the fuzzy match similarity function is to identify the fuzzy matches- the _k'reference tuples closest to u.The qualitative significance of the fuzzy match similarity function explains how the closeness between the input names and reference names is decided while personal names matching, in specific, or any pattern matching in general. It can be observed from the function definition that as the cost of transforming an input tuple u to a reference tuple v increases, the possibilities of closeness between u and v decreases. The figures 12 through 13 shows the results of the algorithms proposed as follows and the relations observed between the various evaluation metrics and corresponding fms values. High fms values signify that corresponding input name is a possible match to the corresponding reference name. The fms gives more accuracy in comparison of q-gram concept of alliance rules algorithm.

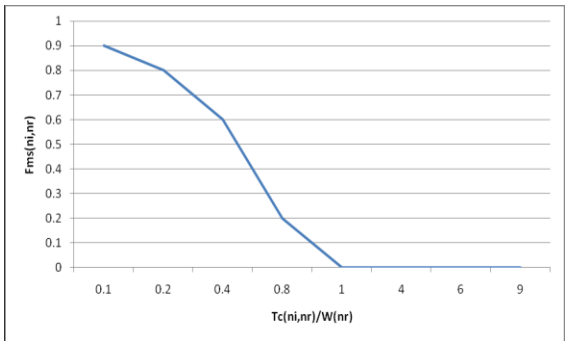

Fig 12: Fms Vs (Tc/W)

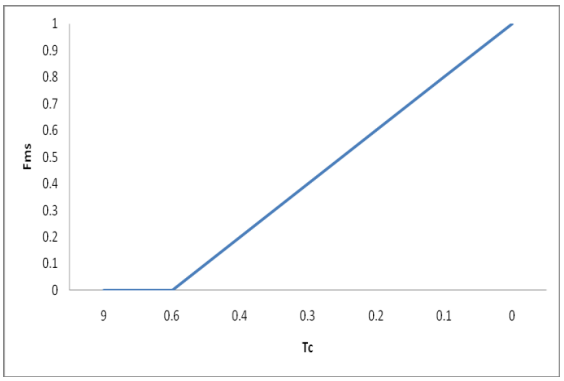

Fig 13: Fms Vs Tc

11. CONCLUSION

In this paper, we have tried to highlight the major problems encountered when data sets are integrated from multiple heterogeneous sources. This data is referred as 'dirty data'. The presence of dirty data in data marts dangerous for business decisions.

Also we have discussed some of the existing algorithm that has been designed to perform identification of duplicity in 'name' field. But, we have analyzed that this deal with only a limited portion of the problem and do not yield the desired results.

We have proposed a new approach for finding duplicity in 'name' field which is more effective and better in terms of performance. This approach deals with the framework and algorithms gives a reliable, logical and well-calculated approach for identification of name duplicity. This approach is amalgamation of well known string similarity measures on q-grams, fuzzy match and alliance rules. It extends the application of alliance rules to find fuzzy duplicates based on a suitable threshold which helps in finding the errors in efficient manner.

12. REFERENCES

[1] Rajiv Arora, Payal Pahwa, Shubha Bansal," Alliance Rules for Data Warehouse Cleansing", International Conference on Signal Processing Systems IEEE Explore no. D01 10.1109/ICSPS, 133, pages 743-747, 2009.

[2] P.Ponniah, "Data Warehousing Fundamentals- A comprehensive guide for IT professionals", Ist ed.,

second reprint, ISBN-81-265-0919-8, Glorious Printers: New Delhi, India, 2007.

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

[4] A.Marcus, J.I.Maletic," Data Cleansing: Beyond Integrity Analysis" Proceedings of the Conference onInformation Quality (IQ2000). Boston: Massachusetts Institute of Technology, pp. 200-209, 2000.

[5] T. Redman, "The Impact of Poor Data Quality on the Typical Enterprise", Communications of the ACM, Vol. 41. 8, February 1998.

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

[7] A.Marcus, J.I.Maletic and Lin, K.-I.," Association Rules for Error Identification in Data Sets", Proceedings of the 10th ACM Conference on Information and Knowledge Management (ACM CIKM 2001). Atlanta, GA, pp. 589-591, 2001.

[8] Peter Christen," A Comparison of Personal Name Matching: Techniques and Practical Issues" Joint Computer Science Technical Report Series, TR-CS-06-02, September, 2006.

[9] Gérard Bouchard and Christian Pouyez, Name Variations and Computerised Record Linkage, Historical Methods, Vol. 13, No. 2, Springer 1980, pp119-125.

[10] Timothy E. Ohanekwu, C.I. Ezeife," A Token-Based Data Cleaning Technique for Data Warehouse Systems", Ontario, Canada N9B, 3P4.

[11] Surajit Chaudhary, Kris Ganjam, Venkatesh Ganti, Rajeev Motwani, " Robust and efficient fuzzy match for online data cleaning",ACM SIGMOD,2003

[12] Amit Rudra, Emilie Yeo, "Key Issues in Achieving Data Quality and Consistency in Data Warehousing among Large Organisations in Australia," Proceedings of the 32nd Hawaii International Conference on System Sciences – 1999.

[13] E. Rahm, H. H. Do: "Data Cleaning: Problems and Current Approaches", IEEE Techn. Bull. Data Eng., Dec. 2000.

[14] Heiko Müller, Johann-Christoph Freytag, Berlin," Problems, Methods, and Challenges in Comprehensive Data Cleansing", 10099 Berlin, Germany.

[15] A. D.Chapman, "Principles and Methods of Data Cleaning – Primary Species and Species-Occurrence Data, version 1.0. Report for the Global Biodiversity Information Facility, Copenhagen, 2005.

[16] Rohit Ananthakrishna (Cornell University) Surajit Chaudhuri Venkatesh Ganti (Microsoft Research)," Eliminating Fuzzy Duplicates in Data Warehouses".

[17] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, Vassilios S. Verykios, "Duplicate Record Detection: A Survey", IEEE transactions on knowledge and data engineering, vol. 19, no. 1, January 2007.

[18] Oktie Hassanzadeh, Mohammad Sadoghi, Ren´ee J. Miller, "Accuracy of Approximate String Joins Using Grams", University of Toronto 10 King's College Rd.,Toronto, ON M5S3G4, Canada.

[19] Jakub Piskorski_, Marcin Sydow, "Usability of String Distance Metrics for Name Matching Tasks in Polish".