1. INTRODUCTION

1.1 INTRODUCTION

Databases plays an important role in today’s IT industry. The shear growth in volume of data has increased the importance of problem such as duplicate data detection from the perspective of the user as well as retrieval system. Data mining, also referred to as knowledge discovery in databases (KDD), is a process of finding new, interesting, previously unknown, potentially useful, and ultimately understandable patterns from very large volumes of data. Data mining is a discipline which brings together database systems, statistics, artificial intelligence, machine learning, parallel and distributed processing and visualization between other disciplines. This becomes more difficult when an entity has two or more representations in the database. Duplicate records do not share a common key and/or they contain errors that make duplicate matching a difficult task. Errors are introduced as the result of transcription errors, incomplete information, lack of standard formats or any combination of these factors [1]. Use of shortcuts such as “Rd.” for “Road”, “M.G.” for “Mahatma Gandhi” and “P.M.” for “Prime Minister” and use of local salutations increases the complexity of the problem of duplicate detection.

Nowadays, one of the most important and challenging problems in data mining is the definition of the prior knowledge; this can be originated from the process or the domain. This contextual information may help select the appropriate information, features or techniques, decrease the space of hypothesis, represent the output in a most comprehensible way and improve the whole process.

Recognizing similarities in large collections of data is a major issue in the context of information integration systems. An important challenge in such a setting is to discover and properly manage duplicate tuples, i.e., syntactically different tuples which are actually identical from a semantic viewpoint, for they referring to the same real-world entity. There are several application scenarios involving this important task.

A typical example consists in the reconciliation of demographic data sources in a data warehousing setting. Names and addresses can be stored in rather different formats, thus raising the need for an effective reconciliation strategy which could be crucial for decision making. In such cases the problem is the analysis of a (typically large) volume of small strings, in order to reconstruct the semantic information on the basis of the few syntactic information available.
1.2 NECESSITY

Databases frequently contain field-values and records that refer to the same entity but are not syntactically identical. Variations in representation can arise from typographical errors, misspellings, abbreviations, as well as integration of multiple data sources. Variations are particularly pronounced in data that is automatically extracted from unstructured or semi-structured documents or web pages [2, 3]. Such approximate duplicates can have many deleterious effects, including preventing data-mining algorithms from discovering important regularities. This problem is typically handled during a tedious manual data cleaning, or “de-duping”, process. Some previous work has addressed the problem of identifying duplicate records, where it was referred to as record linkage [4, 5], the merge/purge problem [6], duplicate detection [7, 8], hardening soft databases [3], reference matching [9], and entity name clustering and matching [10].

Typically, standard string similarity metrics such as edit distance [11] or vector-space cosine similarity [12] are used to determine whether two values or records are alike enough to be duplicates. Some more recent work [10, 8, 13] has investigated the use of pairing functions that combine multiple standard metrics. Because an estimate of similarity between strings can vary significantly depending on the domain and specific field under consideration, traditional similarity measures may fail to estimate string similarity correctly. At the token level, certain words can be informative when comparing two strings for equivalence, while others are ignorable. For example, ignoring the substring “Street” may be acceptable when comparing addresses, but not when comparing names of people (e.g. “Nick Street”) or newspapers (e.g. “Wall Street Journal”). At the character level, certain characters can be consistently replaced by others or omitted when syntactic variations are due to systematic typographical or OCR errors. Thus, accurate similarity computations require adapting string similarity metrics for each field of the database with respect to the particular data domain. Rather than hand-tuning a distance metric for each field, we propose to use trainable similarity measures that can be learned from small corpora of labeled examples, and thus adapt to different domains. We have presented some such string similarity measures. The character based distance is best suited for shorter strings with minor variations, while the measure based on vector-space representation is more appropriate for fields that contain longer strings with more global variations.

In Indian scenario, voters database and PAN cards databases are an examples of large and distributed databases. As well, now banking sector is also maintaining the database of its account holders and their details. They are maintaining this record at the
Regional level. These databases at every region combined together are a distributed database with structural difference.

Our overall system employs a two-level learning approach. First, is to generate the ontology and shortcut dictionary and second is to apply the process of field matching and record classification for the dataset based on ontology and shortcuts.

1.3 OBJECTIVES

If we can measure similarity of two entities to each other then if they exceed some threshold then we can say that they are same. In the research, we were to design an algorithm for duplicate record detection in large distributed databases so that it can be applied for any database. This includes similarity function to compare two entities \textit{i.e.} first of all field and then complete record containing multiple fields and then finding an agreement on the threshold that these two entities are same.

Our overall system employs a two-level learning approach. First, is to generate the ontology and shortcut dictionary and second is to apply the process of field matching and record classification for the dataset based on ontology and shortcuts.

In the literature, the problem of tuple de-duplication has been dealt with mainly from an accuracy viewpoint, by taking care to the minimization of incorrect matching: false positives \textit{(i.e.}, object recognized as similar which actually do not correspond to the same entity) and false negatives \textit{(i.e.}, objects corresponding to the same entity but which are not recognized as similar). However, efficiency and scalability issues do play a predominant role in many application contexts where large data volumes are involved, especially when the object identification task is part of an interactive application, calling for short response times.

When we want de-duplication in distributed database, then the size of the database and the use of local short forms and similar words in the database make it difficult. Here in our work, we worked on the problem and suggested a novel way of using short forms of the words and similar words (ontological words).

1.4 THEME

We have designed an algorithm which uses a dictionary where ontological words and the shortcuts are stored. Though ontological words are region dependant, shortcuts are almost common for English language. For example, “Rd” for “Road”, “St” for “Street” and
“Saint”. As these shortcuts are stored and the blocks are created depending on this shortcuts and ontology, the chances of having false negative can be reduced.

In our work, we proved that if our algorithm is used with the available algorithms, then algorithms where multiple words are considered shows improvement in the similarity measure calculation for short cut prone and similarity word prone databases. Also we have suggested the use of web services for parallel processing of this data depending on the blocks of the data created on all of the available systems in the network.

We have designed a tool for all this functionalities. They are implemented as a web services. Use of web services has helped it use across the network without any type of installations and settings making it possible to use all the systems in the network.

1.5 ORGANIZATION OF THE THESIS

Chapter 1: Introduction

This chapter will describe the concepts of Data Cleaning and De-Duplication.

Chapter 2: Background and Related Work (Literature Survey)

This chapter will give details about the current work in the field of deduplication. The software like Weka and Febrl are available. It will give an introduction about this software.

Chapter 3: Algorithm for Finding Duplicates in Distributed Database using Ontology and Shortcuts Dictionary

A dictionary structure is designed for storage of similar words and shortcuts. This chapter will throw a light on it and list the sample data in it. Also algorithm used for computation of weight using the dictionary is going to be discussed in this chapter.

Chapter 4: Enhancements in the Algorithms for finding the Weights in Numerical Data

A few of algorithms are available for finding the similarity between the numeric data. Basic algorithms include Numeric Percentage, Numeric Absolute, Age difference, Date of Birth difference and Time difference algorithm. At higher end WHO has designed hit miss mixture model for detecting similarity between numeric fields like a salary, date of birth, age and time in adverse drug reaction database. Some enhancements in numeric data search are considered here in the chapter.
Chapter 5: A Proprietary Tool for Record Classification

Simple tool is designed in VB.Net using web services for demonstration all of the algorithms and to evaluate the performance. This tool is discussed in this chapter. Use of web services helps to access the library across the network.

Chapter 6: Testing and Analysis

This chapter gives testing and analysis of the theories and code developed. Analysis includes comparison with existing codes and methodologies and comparison of performances.

Chapter 7: Conclusions and Future Work

Conclusion of the work and future work that is proposed to carried out in this direction.

References: References are added at the end of every chapter.
REFERENCES:
