AN OPTIMIZED TECHNIQUE FOR DUPLICATE DETECTION IN XML DATA

Suvarna B. Kale[1], Basha Vankudothu[2]

[1, 2] Department of Computer Engineering, Pune University, GSMCOE, Balewadi, Pune, (India)

ABSTRACT

Duplicate detection means detecting multiple representations of a same object and also for every object represented in a dataset. Duplicate detection is important in data cleaning and data integration applications and has studied extensively for relational data describing a single type of object in a single table. Presently data is being stored in more complex and semi structured or hierarchical structure and the problem arose is how to detect duplicate objects on XML data. Also due to differences between various data models we cannot apply same algorithms which are use for single relation on XML data. So main aim of our project is duplicate object detection in XML data. In this paper we considering detection of duplicates in multiple types of objects related to each other and devise methods adapted to semi-structured XML data. Our method called XMLDup uses a Bayesian network which is used to determine the probability of two XML elements which are duplicates and considering not only the information within the elements, but also considers the way that information is structured in it. And to improve the efficiency of the network evaluation we are using a novel pruning strategy which is capable of significant gains over the unoptimized version of the algorithm is presented.

Keywords: ***Duplicate Detection, XML Data, Pruning Strategy, Hierarchical Structure***

I. INTRODUCTION

XML is mostly used in most of the business applications to exchange data within the network and it also used for publishing the data on web. Most of the times XML data comes with errors and inconsistencies inherent in the real world data. It is necessary to ensure that the quality of data published on web as it is created from distributed and heterogeneous datasets. Data quality however can be compromised by different types of errors as it can have various origins. Identifying and eliminating duplicate objects has become one of the challenging tasks as the data may not look exactly similar. Presently data's are given different representation as a result identifying different representation of the same entity turn out to be a problem in the field of duplicate detection in xml. So it is essential to use a correct matching strategy for identifying if they refer to the same real world entity or not. What makes duplicate detection an important task is the fact that duplicates are not exactly equal and often due to errors in the xml data. Consequently, we cannot use previous comparison algorithms that detect exact duplicates. Instead of that, we have to compare all object representations, using a possibly complex matching strategy, to decide if they refer to the same object or not. Due to the numerous applications in various fields, duplicate detection has been studied profoundly for data stored in relational datasources. In detecting duplicates in relational tables the tuples are compared and their similarity scores are computed based on their attribute values.

In most of the cases, it deletes some of the data's as foreign keys are used to connect tables. Various algorithms presented for detecting duplicates in hierarchical and semistructured data, most notably, on XML data [1][2].

II. PROBLEM STATEMENT

Duplicates detection and resolution in xml is an important task of data cleaning process which we aim at identifying multiple representation of the same real world entity. Duplicate detection in the world of XML is not easy and it suffers from many problems as object definition which refers to the problem of defining which data values actually describe an object (i.e. which values to consider when comparing two objects). Structural diversity which discusses the problem of multiple structures of the same XML elements unlike relational tupples that makes detecting duplicates in XML more complex process than any other data store, also element dependency which means that XML elements relate to their ancestors and descendants. So these relationships can be considered to improve duplicate detection. For example, two <city> elements with text Los Angeles are nested under different <country> elements with text USA and Chile. However the city names are identical, we can detect that they are not the same city in the real-world, because they are in different countries.

III. LITERATURE SURVEY

In this section we have explained various duplicate detection algorithms and techniques.
Delphi is used to identify duplicates in data warehouse which is hierarchically organized in a table. In this work it doesn't compare all pairs of tuples in the hierarchy as it evaluates the outermost layer first and then proceeds to the innermost layer [4].

D. Milano et.al, suggested a method for measuring the distance of each XML data with one another which is known as structure aware XML distance. Using this edit distance measure and similarity measure can be evaluated. This method compares only a portion of XML data tree whose structure is similar in nature [5].

M. Weis et.al proposed Dogmatix framework which consists of three main steps: candidate definition, duplicate definition and duplicate detection. The method Dogmatix compares XML elements based on the similarity of their parents, children and structure. It also takes into account difference of the compared elements [6].

S. Puhlmann proposed SXNM (Sorted XML Neighborhood Method) is a duplicate detection method that adapts the relational sorted neighborhood approach (SNM) to XML data. Similar to the original SNM, the idea is to avoid performing useless comparisons between objects by grouping together those that are more likely to be similar [7].

Detecting the duplicates between XML entities involves detecting similarity between entities, which is measured using their edit distance as in [8].

But researches in [9] develop an efficient algorithm for detecting duplicates in complex XML using MD5 algorithm. As well presents a novel method for XML duplicate detection, called XMLDup which uses a Bayesian network to determine the probability of two XML elements being duplicates, considering not only the information within the elements, but also the way that information is structured.

Oktie Hassanzadeh [10] Evaluate and compares several unconstrained clustering algorithms for duplicate detection by extensive experiments over various sets of string data with different characteristics. As well provide survey of duplicates detection methods and identify their strength and shortcoming.

The XMLDup system first proposed in [11] uses a Bayesian Network model (BN) for XML duplicate detection. Approach presented in this is the basis for the algorithms proposed in this paper.

IV. METHODOLOGY

The main aim of duplicate detection in XML data is to identify the XML element which represents the same real world object. XML document is considered as duplicates not only based on their structure but also on the way in which contents are represented n XML. Two methods we have presented described as follows.

4.1 A Bayesian Network Model for Duplicate Detection

Method for detecting duplicate data detection is to construct the Bayesian network then compute the similarity between XML objects and using this similarity we can classify two XML objects as duplicates if it falls above a threshold.

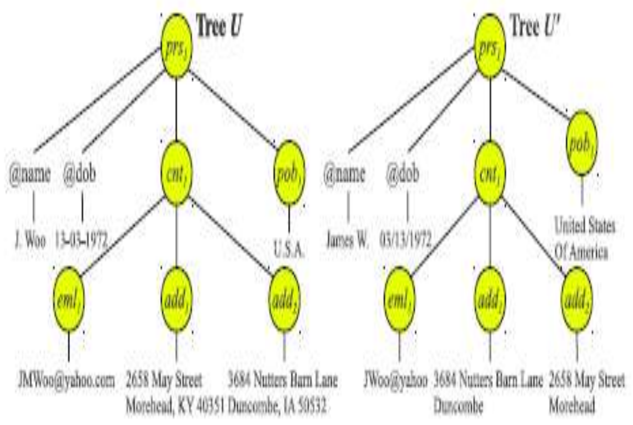

Fig 1 Two XML Elements That Represent the Same Person Nodes Are Labeled By Their XML Tag Name

The Bayesian Networks gives us a specification of a joint probability distribution and it can be seen as a directed acyclic graph, where the nodes represent random variables and the edges represent dependencies between those variables.

4.2 Bayesian Network Constructions

The basic assumption for XML duplicate detection is the fact that two XML nodes are duplicates depends only on the fact that their values are duplicates and their children nodes are duplicates. So two XML trees are duplicates if their root nodes are duplicates. An XML tree is defined as a triple $U = (t, V, C)$

Where,

t is a root tag label, e.g., for tree U

V is a set of (attribute, value) pairs. If the node itself has a value then we can consider it as a special (attribute, value) pair.

C is a set of XML trees that is the sub-trees of U. These subtrees are again each described by a triple.

First consider the XML nodes tagged prs. As illustrated in Following figure, the BN will have a node labeled prs11 representing the possibility of node prs1 in the XML tree U being a duplicate of node prs1 in the XML tree U0. Node prs11 is assigned a binary random variable and this variable takes the value 1 (active) to represent the fact that the XML prs nodes in trees U and U' are duplicates and It takes the value 0 (inactive) to represent the fact that the nodes are not duplicates. Then compute prior probability, conditional probability and final probability

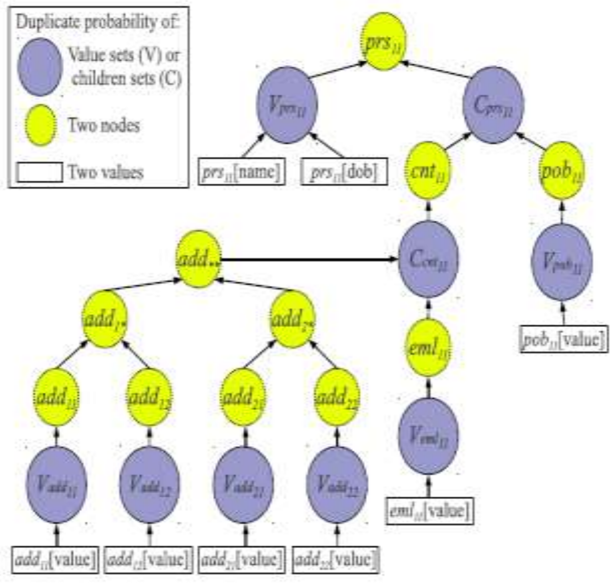

Fig 2 Bayesian Network to Compute Similarity of Trees in Fig 1

4.3 Network Pruning

To increase Bayesian network evaluation time a lossless pruning strategy is used. We can say that this strategy is lossless in the sense that no duplicate objects are lost. Only the object pairs incapable of reaching a given duplicate probability threshold value are discarded. A network evaluation is performed by doing a propagation of the prior probabilities, in a bottom up fashion, until reaching the topmost node of tree. The prior probabilities are obtained by applying a similarity measure to the pair of values represented by the content of the leaf nodes and computing such similarities is the most expensive operation in the network evaluation and in the duplicate detection process in general. Therefore, the idea behind our pruning proposal lies in avoiding the calculation of prior probabilities, unless they are strictly necessary and the strategy is that before comparing two objects, all the similarities are assumed to be 1 (i.e., the maximum possible score). At every step of the process, maintain an upper bound on the final probability value. At each step, whenever a new similarity is computed, the final probability is estimated taking into consideration the already known similarities and the unknown similarities that assume to be 1. When verify that the network root node probability can no longer achieve a score higher than the defined duplicate threshold, the object pair is discarded and, thus, the remaining calculations are avoided. The algorithm takes input as node N from the BN and a threshold T. It starts by gathering a list of all the parent nodes of N and assuming the duplicate probability score is 1. It then proceeds compute the actual probability value. If a parent node n is a value node then its probability score is simply the similarity of the value. If n also has parent node then its probability score depends on its own parent.

Algorithm 1. Network Pruning (N, T)

Require: The node N, for which we intend to compute the probability score; threshold value T, below which the XML nodes are considered non-duplicates

```
Ensure: Duplicate probability of the XML
nodes represented by N
1: L ← getParentNodes(N) {Get the ordered list of parents}
2: parentScore[n] ← n L {Maximum probability of each parent node}
3: currentScore ← 0
4: for each node n in L do {Compute the duplicate probability}
5: if n is a value node then
6: score ← getSimilarityScore(n) {For value nodes, compute the similarities}
7: else
8: newThreshold ← getNewThreshold(T, parentScore)
9: score ← NetworkPruning(n, newThreshold)
10: end if
11: parentScore[n] ← score
12: currentScore
←computeProbability(parentScore)
13: if currentScore < T then
14: End network evaluation
15: end if
16: end for
17: return currentScore
```

Fig.3.Network Pruning Algorithm

V. CONCLUSION

Duplicate detection is defined as the identification of different representations of a same real-world object called as duplicates and these duplicates are due to errors and inconsistencies in the data, like typos and misspellings, missing information, or outdated data. The consequence of this, duplicates are not exactly equal, which makes duplicate detection a challenging task in data cleaning and data integration processes. In this paper, we have presented a method for XML duplicate detection called XMLDup. Our method uses a Bayesian Network to determine the probability of two XML objects being duplicates. Bayesian Network model is construct from the structure of the objects being compared and also all probabilities are computed considering not only the information the objects contain, but also the way such information is structured in it. To improve the runtime efficiency of our method called XMLDup we propose a network pruning algorithm is used. We can say that this strategy is lossless in the sense that no duplicate objects are lost only object pairs incapable of reaching a given threshold are discarded. Every node should have the duplicate probability value as 1 before evaluation, this is called as pruning factor. Slightly lowering the pruning factor the system achieves high performance. The XMLDup does not consider different structure representation and the conditional probability is derived manually. So, in proposed the machine learning algorithm with support vector machine is used to extend the BN model construction. When compare to XML Dup, the machine learning algorithm is expected to calculate the objects with different structures and conditional probability is obtained automatically with the help of Support vector machine.

VI. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their very useful comments and suggestions.

REFERENCES

[1] M. Weis "*Dogmatix Tracks Down Duplicates in XML," Proc. ACM SIGMOD Conf. Management of Data, pp. 431 442, 2005*.

[2] L. Leita˜o and M. Weis, "*Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection*," Proc. 16th ACM Int'l Conf. Information and Knowledge Management, pp. 293-302, 2007.

[3] Melanie Weis,"*Fuzzy Duplicate Detection on XML Data*", Humboldt-University at zu Berlin Unter den Linden 6, Berlin, Germany, 2005.

[4] R .Ananthakrishna, S. Chaudhuri "*Eliminating fuzzy duplicates in data warehouses*" In International Conference on Very Large Databases, Hong Kong, China, 2002.

[5] D. Milano, M. Scannapieco "*Structure Aware XML Object Identification*" Proc. VLDB Workshop Clean Databases (CleanDB), 2006.

[6] M. Weis and F. Naumann, "*Dogmatix Tracks Down Duplicates in XML*" Proc. ACM SIGMOD Conf. Management of Data, 2005.

[7] S. Puhlmann, M. Weis "*XML Duplicate Detection Using Sorted Neighborhoods*" Proc. Conf. Extending Database Technology (EDBT), pp. 773-791, 2006.

[8] Melanie Weis and Felix "*Detecting duplicate objects in XML documents*" Humboldt-Universität zu Berlin, germany,Workshop on Information Quality in Information Systems, 2004

[9] Lwin, T, "*An efficient Duplicate detection system for XML document*" Computer Engineering and Applications (ICCEA) Second International Conference in 2010.

[10] Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee and Ren´ee J. Miller, University of Toronto, "*Framework for Evaluating Clustering Algorithms in Duplicate Detection*",2009.

[11] P. Calado "*Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection*," Proc. 16th ACMInt'l Conf. Information and Knowledge Management, pp. 293-302, 2007.