Efficient Duplicate Detection and Elimination in Hierarchical Multimedia Data

Ms. Manjusha R. Pawar
Department of Computer Engg.
Late G. N. Sapkal Collage of Engg.
Nashik, India.
pawar.manjusha@gmail.com

Prof. J. V. Shinde
Asst. Prof., Department of Computer Engg.
Late G. N. Sapkal Collage of Engg.
Nashik, India.
jvshinde@rediffmail.com

Abstract- Today's important task is to clean data in data warehouses which has complex hierarchical structure. This is possibly done by detecting duplications in large databases to increase the efficiency of data mining and to make data mining effective. Recently new algorithms are proposed that consider relations in a single table; hence by comparing records pairwise they can easily find out duplications. But now a day the data is being stored in more complex and semistructured or hierarchical structure and the problem arose is how to detect duplicates on this XML data. Due to differences between various data models we cannot apply same algorithms which are for single relation on XML data. The objective of this paper is to detect duplicates in hierarchical data which contain textual and multimedia data, like images, audio and video. Also to act according to user choice on that data like delete, update etc. Also to prune the duplicate data by using pruning algorithm that is included in proposed system. Here Bayesian network will be used for duplicate detection, and by experimenting on both artificial and real world datasets the XMLMultiDup method will be able to perform duplicate detection with high efficiency and effectiveness. This method will compare each level of XML tree from root to the leaves. The first step is to go through the structure of tree comparing each descendant of both datasets inputted and find duplicates despite difference in data.

***Keywords*: Duplicate detection, Data cleaning, XML Data, Bayesian network, Pruning.**

I. INTRODUCTION

XML is popular for data storage in data warehouses, but it comes with errors and inconsistencies to real-world data, hence, there is a need of XML data cleansing [8]. By recognizing and eliminating duplicates in XML data [12] could be the solution; thus strategy based on Bayesian Network to detect duplicates and the method to eliminate that duplicates can be used.

Various algorithms[11] and techniques have been proposed or implemented for duplicate detection [1] on single relations. But XML data [10] has complex and hierarchical structure therefore we cannot directly apply those techniques which are being used for single relations on XML data. Although there is a long line of work on identifying duplicates in relational data, only a few solutions focus on duplicate detection in more complex structures [7], like XML databases.

Moreover hierarchical data which contain multimedia data like images and videos has very difficult structure and detecting duplication in such a data become complicated. The proposed method is a novel method for duplicate detection in XML data. Detecting duplicates[9] in hierarchical multimedia data is more challenging than detecting duplicates in relational and simple XML data, because comparing tuples and computing probabilities has no ambiguity of text but the data such as images and videos is more challenging because of its need of space on web for publishing and structural diversity. On the other hand, XML duplicate detection allows exploiting the hierarchical structure for efficiency in addition to effectiveness, which is not the case when detecting duplicates in simple data.

Consider the two XML elements shown with hierarchical structure in Fig. 1. Both represent films objects and are labeled Films. These elements have three attributes, namely the name of film, release date and country where the film is released. These are tags within XML trees and they nest further XML elements representing the contents of film. As film contains series of several images or posters and audios, this film1 tag contains the paths of all these contents where the images and audios are being stored. All leaf elements have a text which may be simple value or the path of any multimedia file which stores the actual multimedia data. For instance, Poster1.jpg in both trees may be same posters of film or may not be. Again audio1.mp3 may be different in second tree if the film is found not duplicate of film in first tree.

In this example, the goal of duplicate detection is to detect that both Films are duplicates, even if values within tree are different. To do this, we can compare the corresponding structure, values and contents of both objects. In this work, this paper proposes that if structure is found similar first then we can proceed to find similarity of the values and further we proceed for the duplicate detection in multimedia data. Also if multimedia data in both tree s found similar then there is elimination or update to the trees so as to minimize size of data within data warehouses or databases.

Contributions: This proposed method, present a probabilistic duplicate detection method for hierarchical multimedia data called XMLMultiDup. This method considers all parameters and aspects for comparison of XML datasets

which contain multimedia database like images, audio and videos. The algorithm presented here extends work in [1] significantly improving level of detecting duplication and efficiency and this paper also considers relational databases for finding duplications by converting it into hierarchical datasets.

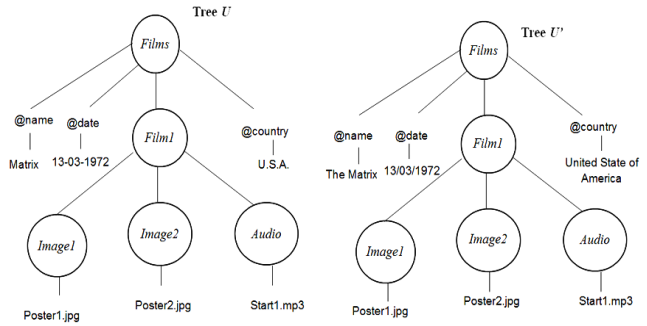

Figure 1: Two XML elements that represent the same Film. Nodes are labelled by their XML tag name.

Here the main contribution compared to previous work is and objectives of this system are 1) to detect duplicates in hierarchical data which contain multimedia data e.g. images, audio and video using XMLMultiDup method. 2) To convert relational database into XML and then detect duplicates as above. 3) To compare datasets according to user choice and display results e.g. limited no. of levels to be compared. 4) To increase efficiency and effectiveness of duplicate detection in comparison of multimedia databases. 5) To eliminate or update duplicates, to reduce size of databases in data warehouses. 6) To consider all probabilities of XML trees for comparison for example part of tree, structure of trees, levels of tree, values and contents within trees and complete subtrees to find duplications.

Structure: This paper is organized as follows: Section 2 presents related work. Section 3 summarizes methodology of the proposed system. Our strategies to this algorithm are then presented in Section 4.Working environment of these techniques over artificial and real world data in Section 5. Finally, in Section 6 we conclude and present suggestions for future work.

II. RELATED WORK

In [2] Ananthakrishna has exploited dimensional hierarchies typically associated with dimensional tables in data warehouses to develop duplicate elimination algorithm called Delphi, which significantly reduces the number of false positives without missing out on detecting duplicates. He rely on hierarchies to detect an important class of equivalence errors in each relation, and to efficiently reduce the number of false positives.

Carvalho and Silva proposed a similarity-based approach in [3] to identifying similar identities among objects from multiple Web sources. This approach works like the join operation in relational databases. In the traditional join operation, the equality condition identifies tuples that can be joined together. In this approach, a similarity function that is based on information retrieval techniques takes the place of the equality condition. In this paper, we present four different strategies to define the similarity function using the vector space model and describe experimental results that show, for Web sources of three different application domains, that our approach is quite effective in finding objects with similar identities, achieving precision levels above 75%.

DogmatiX is a generalized framework for duplicate detection [4], dividing the problem into three components: candidate definition defining which objects has to be compared, duplicate definition defining when two duplicate candidates are actually duplicates, and duplicate detection means how to efficiently find those duplicates. The algorithm is very effective in the first scenario: Edit distance should compensate typos, and our similarity measure is specifically designed to identify duplicates despite missing data. On the other hand, synonyms, although having the same meaning, are recognized as contradictory data and the similarity decreases. They are more difficult to detect without additional knowledge, such as a thesaurus or a dictionary. Thus, we expect the second scenario to yield poorer results.

Milano Propose a novel distance measure for XML data, the structure aware XML distance [5] that copes with the flexibility which is usual for XML files, but takes into proper account the semantics implicit in structural schema information. The structure aware XML distance treats XML data as unordered. The edit distance between tokens t1 and t2 is the minimum number of edit operations (delete, insert, transpose, and replace) required to change t1 to t2; we normalize this value with the sum of their lengths

In [6] author has proposed a novel method for detecting duplicates in XML which has structural diversity. This method uses a Bayesian network to compute the probability of any two XML objects being duplicates. Here author has considered not only children elements but also complete subtrees. Computing all probabilities, this method performs accurately on various datasets. Following figure shows two XML trees which contain duplicate data although value represented differently.

Base for proposed system presented in [1], has extended work done in [6] by adding pruning algorithm to improve the efficiency of the network evaluation. That is to reduce the no. of comparisons where the pairs which are incapable of reaching a given duplicate probability threshold are decreased. It requires user to give input, since the user only needs to provide the attributes to be compared, their respective default probability values, and a similarity value. However, the system worked in good manner that it allows to use different similarity measures and different combinations of probabilities.

III. METHODOLOGY

A method described in [1], the author has extended his previous work by increasing efficiency and effectiveness for duplicate detection in hierarchical data, but proposed system will be useful for both simple and multimedia data. Here the

input will be two XML trees or datasets; for this we can use real world data or standard dataset. The first phase of this XMLMulitDup method is to input either XML data or simple relational data which will be converted into XML for comparison and duplicate detection. The choice of user will be taken for comparison i. e. whether to compare structures of tree, values of tree or contents of the tree. The second contribution of proposed system is to input dataset which contain any type of multimedia data which contain images, audio or videos. We will first compute prior, computational and final probabilities using Bayesian Network. The algorithm and formulae given below will be used for this whole recursive process. But while doing this, if we use Bayesian network there is an issue of complexity of O(n X n'). Hence a pruning technique will be used which reduces no. of comparisons using pruning algorithm. Here the contents of multimedia data will be later compared by MD5 hash Key algorithm, if structure and values found duplicate. Following figure shows the architecture of proposed system, which includes combination of three algorithms.

1. Bayesian network
2. Pruning Algorithm
3. MD5 Hashkey Algorithm

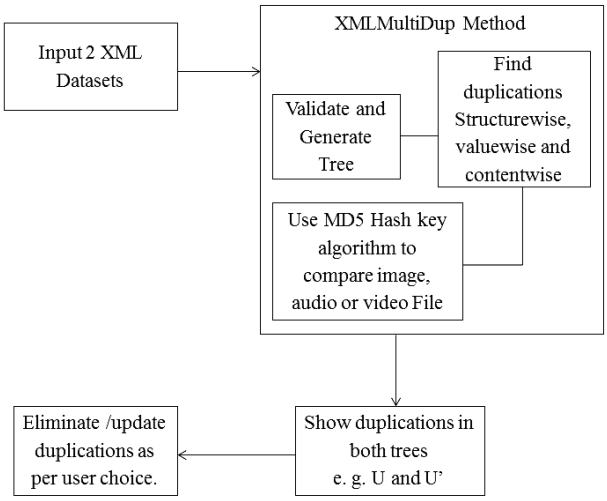

Figure 2: Architecture of Proposed System

Here the proposed system uses all these algorithms but needs small user intervention. User has to provide the parameter by which comparison will perform. And second the action to be performed after duplicate detection which may be elimination, updation, or any other operation. Next section will describe all algorithms and example which show how the original trees shown in figure 1 will be converted to Bayesian network.

IV. DESIGN AND SPECIFICATION

Bayesian network Construction

We assume that two trees can only be duplicates if they are of the same type. Also, two nodes can be compared only if they are of the same type. In our example, the real-world types are *Tfilms = movie, Timage = posters, Taudio= audioclip*. For simplicity, in the subsequent definitions we assume that nodes with the same real world type also have the same tag. That is, a relabeling step has preceded the construction of the BN. To illustrate this idea, consider the goal of detecting that both films shown in Fig. 1 are same. This means that the two movies objects, represented by nodes tagged films, are duplicates depending on whether or not their children nodes, tagged name, date, film and country and their values are same. Furthermore, the nodes tagged image and audio are duplicates depending on whether or not their contents are duplicates. Here the path represents value and the file contained in path has the content of node.

Bayesian Network Construction

Formally, an XML tree is defined as a triple U = (t, V, C), where t is a root tag label, e.g., for tree U in Fig. 6, t = prs1. V is a set of (attribute, value) pairs. If the node itself has a value and C is a set of XML trees, i.e., the sub-trees of U. We say that two XML trees are duplicates if their root nodes are duplicates.

Algorithm 1:BNGen (*XTreeSet U, XTreeSet U'*)

Input: $U = \{(t_1, V_1, C_1), (t_2, V_2, C_2), \ldots\}$,
$U' = \{(t'_1, V'_1, C'_1), (t'_2, V'_2, C'_2), \ldots\}$
Output: A directed graph G = (N, E)
/* ---------- Initialization ---------- */
/* Root node tags of all XML trees in U and U'*/
$S \leftarrow \{t_1, t_2, \ldots\}$;
$S' \leftarrow \{t'_1, t'_2, \ldots\}$;
/* Tags in S and S' representing real-world type r */
Sr = {ti ∈ S|Tti 3 = r};
$S'_r = \{t'_i \in S' | Tt'_i = r\}$;
/* ---------- BN Construction ---------- */
foreach type r ∈ S ∪ S' do
/* Nodes with single occurrence */
if $|S_r| \leq 1$ *and* $|S'_r| \leq 1$ then
Insert into N a node t_{ii}*;*
if $Vi \cup V'_i \neq \emptyset$ then
Insert into N a node Vtii ;
Insert into E an edge from this node to node t_{ii}*;*
if $Ci \cup C'_i \neq \emptyset$ then
Insert into N a node Ct_{ii} *;*
Insert into E an edge from this node to node t_{ii}*;*
if *node Vtii 14 was created* then
foreach *attribute a* $\in Vi \cup V'_i$ do
Create a node $t_{ii}[a]$*;*
Insert an edge from this node to node Vt_{ii}*;*
if *node* Ct_{ii} *was created* then
$G' = (N', E') \leftarrow BNGen(Ci, C'_i)$;
foreach *node n* $\in N'$ do
Insert n into N;
foreach *edge e* $\in E'$ do
Insert e into E;
foreach *node n* $\in N'$ *without outgoing edges* do
Insert an edge in E from n to node Ct_{ii} *;*
/* Nodes with multiple occurrences */

else if *Sr or S'*r contain more than one tag each then
Insert into N a node t∗∗;
foreach tag ti 28 ∈ Sr do
Insert into N a node tᵢ∗;
Insert into E an edge from this node to node t∗∗;
foreach tag t'j ∈ S'r do
Insert into N a node tij ;
Insert into E an edge from this node to node ti∗;
foreach newly created node tij do
Similar to processing of node tii (lines 8-25),
second subscript i is replaced by j..

Therefore there is next step to verify path and contents of the specified multimedia database. This process goes on recursively until the leaf nodes are reached. If we consider trees U and U' of Fig. 1, this process can be represented by the Bayesian Network of Fig. 2.

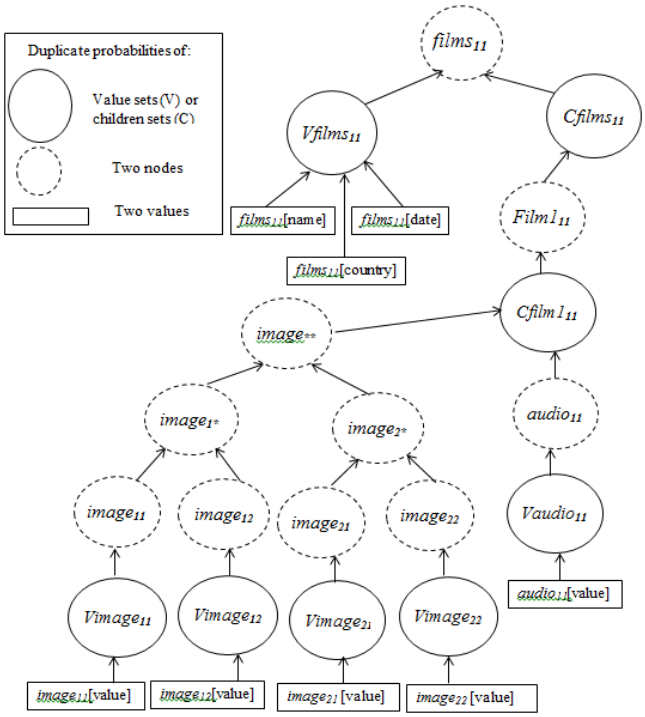

Figure 3: BN to compute similarity of trees in Fig. 1.

While the system finds any path for multimedia data then it will be compared by using MD5 Hashkey algorithm. To construct the above tree, the system will use following probabilities:
Four types of conditional probabilities:
(1) The probability of the values of the tags being duplicates, given that each individual pair of values contains duplicates data;
(2) The probability of the descendant tags being duplicates, given that each individual pair of descendants is duplicates;
(3) The probability of two tags being duplicates given that their values and their descendant are same; and
(4) The probability of a set of nodes of the same type being duplicates given that each pair of individual tags in the set are same.

Pruning technique
To improve the efficiency and effectiveness of algorithm 1, following algorithm will use some factors which will effect on time of execution of algorithm 1. Such as order of nodes on pruning, features such as uniqueness, content length, format, absence and occurrence features.

Algorithm 2: NetworkPruning(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]* ← 1; 8n e 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*

MD5 Hash key algorithm
The proposed system will use this algorithm for comparing the contents of multimedia contents contained in both the trees. All previous methods just detect the textual and structural duplications. But the proposed method extends the duplicate detection within the multimedia databases, which are included in datasets. In construction of Bayesian network tree, there will be computation of probabilities of node values being duplicates. Next we use the pruning algorithm for increasing the efficiency and effectiveness of the Bayesian algorithm. But while doing this some datasets may contain the multimedia databases and we need to compare them for finding duplicate.
MD5 is specifically used to generate hash keys of both files each present in individual tree. It then compares tree and check for duplication. Means even if path are different may the files are same. Hence by using MD5 we can detect duplicates within multimedia files which are included in XML datasets.

V. EXPERIMENTAL SETUP

In previous method the author has considered all attribute values as textual strings but proposed system will consider the attribute values as path of any multimedia database such as path of image, path of video or path of audio. That is XML datasets which contain multimedia data, is necessary as input for this system. For this we can either create artificial dataset or real world dataset which contain multimedia dataset will be used. It will be performed in integrated development environment of Microsoft Visual Studio 2010 and Dot Net Framework 4.0. with any windows and on Intel two core CPU at 2.53 GHz , 2 GB of RAM and 40 GB HDD.

VI. CONCLUSION

The new method XMLMultiDup presents a novel procedure for XML duplicate detection which contains various type of multimedia database. Using a Bayesian network model, this method is able to accurately determine the probability of two XML objects in a given database being duplicates. This model is derived from the structure of the XML objects being compared and all probabilities are computed taking into account not only the values contained in the objects but also their internal structure. To improve the runtime efficiency of XMLMultiDup, a network pruning strategy is also used as basis. This XMLMultiDup can be applied in two ways. Direct on the datasets which are consist of XML tags and Relational dataset. Second approach will need conversion of relations to the XML data and then go for first approach. Further there is need of parsing the XML data into trees and then apply above discussed algorithms.

VII. FUTURE WORK

We can extend the presented method to avoid user intervention with high accuracy, effectiveness and efficiency. The use of domain dependent similarity measures for prior probabilities, extend the BN model construction algorithm to compare XML objects with different structures, experiment with more collections and different network configurations, and apply machine learning methods to derive the conditional probabilities, based on multimedia data.

REFERENCES

[1] Luis Leitao, Pavel Calado, and Melanie Herschel, "Efficient and Effective Duplicate Detection in Hierarchical Data" IEEE Trans. on Knowledge and Data Engineering, Vol. 25, No. 5, May 2013.

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating Fuzzy Duplicates in Data Warehouses," Proc. Conf. Very Large Databases (VLDB), pp. 586-597, 2002.

[3] J.C.P. Carvalho and A.S. da Silva, "Finding Similar Identities among Objects from Multiple Web Sources," Proc. CIKM Workshop Web Information and Data Management (WIDM), pp. 90-93, 2003.

[4] M. Weis and F. Naumann, "Dogmatix Tracks Down Duplicates in XML," Proc. ACM SIGMOD Conf. Management of Data, pp. 431-442, 2005.

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

[6] L. Leitao, P. Calado, and M. Weis, "Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection," Proc. 16th ACM Intl Conf. Information and Knowledge Management, pp. 293-302, 2007.

[7] K.-H. Lee, Y.-C. Choy, and S.-B. Cho, "An efficient algorithm to compute differences between structured documents" IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 16, no. 8, pp. 965979, Aug. 2004.

[8] E. Rahm and H.H. Do, "Data Cleaning: Problems and Current Approaches" IEEE Data Eng. Bull., vol. 23, no. 4, pp. 3-13, Dec. 2000.

[9] L. Leitao and P. Calado, "Duplicate Detection through Structure Optimization," Proc. 20th ACM Intl Conf. Information and Knowledge Management, pp. 443-452, 2011.

[10] M.A. Hernandez and S.J. Stolfo, "The Merge/Purge Problem for Large Databases," Proc. ACM SIGMOD Conf. Management of Data, pp. 127-138, 1995.

[11] S. Guha, H.V. Jagadish, N. Koudas, D. Srivastava, and T. Yu, "Approximate XML Joins," Proc. ACM SIGMOD Conf. Management of Data, 2002.

[12] Joe Tekli, Richard Chbeir, Kokou Yetongnon "An overview on XML similarity: Background, current trends and future directions",Computer Science Review, Volume 3, Issue 3, August 2009, Pages 151173