ISSN:2320-0790

An International Journal of Advanced Computer Technology

Duplicate Detection in Hierarchical XML Multimedia Data Using Improved Multidup Method

Prof. Pramod. B. Gosavi [1], Mrs. M.A.Patel [2]
Associate Professor & HOD- IT Dept., Godavari College of Engineering, Jalgaon
Student- ME- Computer Science & Engineering, Godavari College of Engineering, Jalgaon

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 MULTIDUP 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 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.

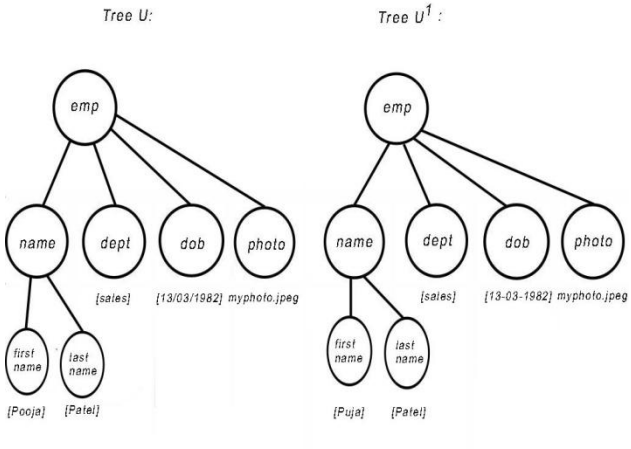


Figure. 1: Two XML elements that represent the same Employee. Nodes are labelled by their XML tag name.

Consider the two XML elements shown with hierarchical structure in Fig. 1. Both represent films objects and are labeled EMP. These elements have four attributes, namely the name of EMP, dept, dob, photo. In emp tree name attribute is further divided into firstname, lastname. These are tags within XML trees and they nest further XML elements representing the contents of employee. emp tree contains one multimedia attribute photo.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, myphoto.jpeg in both trees may be same photo of EMP or may not be.

In this example, the goal of duplicate detection is to detect that both EMP subtree are duplicates, even if values within tree are somewhat different. To do this, we can compare the corresponding structure, values and contents of both trees. 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.

Contributions: This proposed method, present a probabilistic duplicate detection method for hierarchical multimedia data called MULTIDUP. This method considers all parameters and aspects for comparison of XML datasets which contain multimedia database like images, audio and videos. To obtain better result in terms of effectiveness a network pruning is applied. 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. In this paper performance is evaluated by comparing the recall and precision values of MULTIDUP & XMLDUP with a proposed method MULTIDUP.

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 MULTIDUP method. 2) To convert relational database into XML and then detect duplicates as above. 3) To compare datasets according to user choice by compare level & node path selection and display results 4) To increase efficiency and effectiveness of duplicate detection in comparison of multimedia databases. 5) 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.6) To improve efficiency of system by network pruning.

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] Anantha krishna 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 XML Data and with databases, which contains multimedia attributes. Here the input will be two XML trees – one original and other is duplicate (dirty). In proposed system, we have on worked on real world datasets (Cora, CD, and Country).

The working of proposed system starts by giving input of two XML Files. First XML file is original (U) & another XML file is duplicate (U') . After giving this input user of the system selects choice of comparision. We can find duplicate record by three types. We can also give path for comparision.

- **Structure only**- This option is used if we want to search the structure of selected node for finding duplication. With this option only structure of selected node is compared, attribute values are not taken into consideration.
- **Structute & Text**- This option is used if both structure & attribute values are required at the time of comparision.
- **Multimedia Attribute**- This option is used when both files (Orginal (U), duplicate (U')) contains multimedia attribute.Here MD5 Hash key algorihtm is used to compute hash key of multimedia attribute of both files.

Node Path Selection- It has two options

- **Single** - This option starts searching of seleected subtree from root to leaf node in another file.If Matching occurs then subtrees are highlighted in another file.
- **Multiple**- This option also strats searching of seleected subtree from root to leaf node in another file but if some attribute of selected tree are matching in another file eventhough their roots are different, they are highlighted as duplicates.

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.Following figure shows the architecture of proposed system, which includes combination of two algorithms.

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

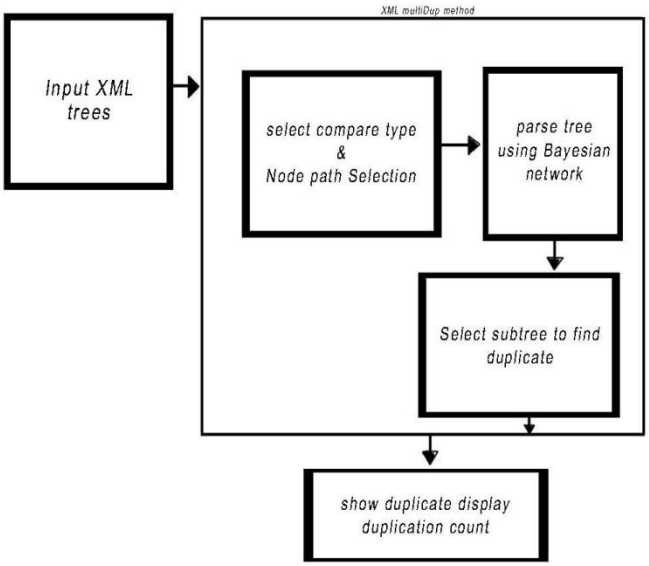


Figure 2: Architecture of Proposed System

Proposed system uses all these algorithms but needs small user intervention. User has to provide the parameter by which comparison will perform. 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 for finding Duplicates

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 *Tname* = fitstname & lastname, *Tphoto* = myphoto, *Tdob*= *date_of_birth*, *Tdept=deptname*. 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 emp trees shown in Fig. 1 are same. This means that the two emp objects, represented by nodes tagged emp are duplicates depending on whether or not their children nodes tagged name, dob, dept, photo and their values are same. Furthermore, the nodes tagged photo is duplicate depending on whether or not its content is duplicate. Here the path represents value and the file contained in path has the content of node.We are also highlighting the inner nodes duplication even if subtree roots did not match.

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. 1, 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.

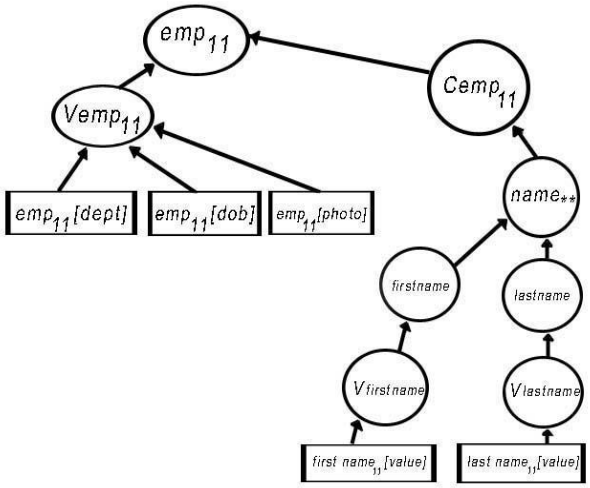


Figure 3: BN to compute similarity of trees in Fig 1.

Algorithm for finding duplcation in dataset without multimedia attributes

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

Input: U = {(t_1, V_1,C_1), (t_2, V_2, C_2), . . . },
U' = {(t'_1, V'_1, C'_1), (t'_2, V'_2, C'_2), . . . }
Output: Duplicate sub trees are highlighted with red color, Count of total duplication is obtained.

/* ---------- Initialization ---------- */
/* Root node tags of all XML trees in U and U'*/
1. S ← {t_1, t_2, . . .};
2. S' ← {t'_1, t'_2 , . . .};
/* Tags in S and S' representing real-world type r */
3. Sr = {ti ∈ S|Tti 3 = r}
S'_r = {t'_i∈ S'|Tt'_i = r};
/* ---------- BN Construction ---------- */
4. Use BN Construction algorithm for constructing BN netwok
5. Select Compare type & Node path selection
Where- compare type-i) Structure Only
ii) Structure & Text
iii) Multimedia Column
Node Path Selection-
i) Single
ii) Multiple
6. Select Subtree from XtreeSetU
7. Compare selected subtree of XtreeSetU
to each subtree of XtreeSetU'
8. If <Comparision fails>
Return false
Else
Goto next subtree in XtreeSetU'
9. Repeat step. 8 until last node
10. Highlight duplicate nodes with red color- show count of total duplication

The above algorithm take to XML trees(***XTreeSet U, XTreeSet U')*** as input and highlight duplicate node. The algorithm works by comparing selected subtree from ***XTreeSet U*** to each subtree of ***XTreeSet U'.*** Each node in subtree1 is comapared to all nodes of subtree2. If match found System assign digit '1' to that node otherwise '0'. This process goes on recursively until the leaf nodes are reached. At end final probability is calculated and depending on its score subtree is termed as duplicate or nonduplicate.

Algorithm for finding duplcation in dataset contains multimedia attributes

Algorithm 2: MultiDup (*XTreeSet U, XTreeSet U'*)

Input: U = {(t_1, V_1,C_1), (t_2, V_2, C_2), . . . },
U' = {(t'_1, V'_1, C'_1), (t'_2, V'_2, C'_2), . . . }
Output: Duplicate subtrees are highlighted with red color, Count of total duplication is

/* ---------- Initialization ---------- */
/* Root node tags of all XML trees in U and U'*/
1. S ← {t_1, t_2, . . .};
2. S' ← {t'_1, t'_2 , . . .};
/* Tags in S and S' representing real-world type r */
3. Sr = {ti ∈ S|Tti 3 = r}
S'_r = {t'_i∈ S'|Tt'_i = r};
/* ---------- BN Construction ---------- */
4. Use BN Construction algorithm for constructing BN netwok
5. Select Compare type & Node path selection
Where- compare type-i) Structure Only
ii) Structure & Text
iii) Multimedia Column
Node Path Selection-
i) Single
ii) Multiple
6. Select subtree from XtreeSetU
7. Select multimedia attribute from selected subtree
8. Compare selected subtree of XtreeSetU
to each subtree of XtreeSetU'
9. If <Comparision fails>
Return false
Else
Goto next subtree in XtreeSetU'
10. Compute hash key of selected multimedia attribute.
11. If key is unique then apply colors to dupliacte subtree
12. Repeat step. 9, 10, 11 until last node
13. Show count of total duplication

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. In 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. But while doing this some datasets may contain the multimedia databases and we need to compare them for finding duplicate.

MD5 Hash Key algorithm is 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

This section gives a brief description about the experiments performed using the dataset. The same dataset used in [1] are taken for the process of duplicated detection. Result Analysis is performed on integrated development environment of Microsoft Visual Studio 2010 and Dot Net Framework 4.0. with Windows XP and on Intel core two Duo CPU at 2.00 GHz , 3 GB of RAM and 40 GB HDD.

- **Effectiveness Evaluation-**

To measure the effectiveness proposed method is compared with XMLDUP method. Two parameters are used here recall and precision

Table 1-Recall & Precision values

Pruning Factor	XMLDUP(pf=1)		MULTIDUP(pf=1)	
	Precision	Recall	Precision	Recall
Dataset				
CD	99	95	100	100
Cora	99	90	100	100
MultiCora	80	75	95	95
MultiCD	80	70	97	97

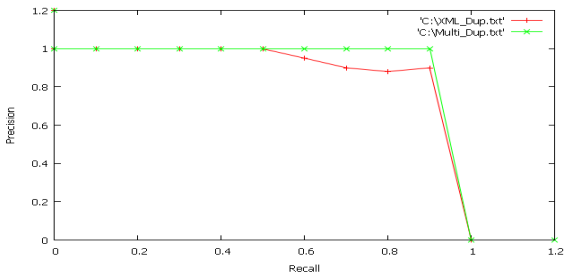
- **Efficiency Evaluation**

To evaluate efficiency we performed experiments using MultiDup using our proposed pruning algorithm.

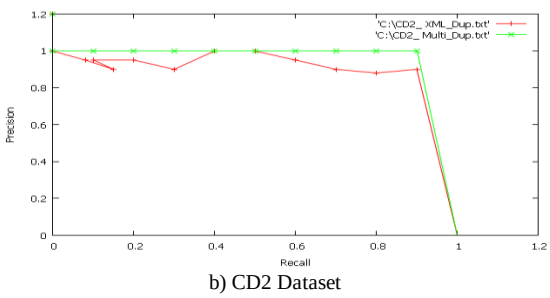
Table 2- Performance Achieved Using the Pruning Method on Real Data.

Dataset	XMLDup(pf=1)	MULTIDUP(pf=1)
CD	02:07:17:50	00:48:40:00
Cora	00:02:41:30	00:01:45:31
MultiCora	00:03:45:00	00:01:40:00
MultiCD	01:06:16:40	00:50:57:00

Time in format: hh:mm:ss:ms



a) Cora Dataset



b) CD2 Dataset

Fig 3. Comparision results for MultiDup & XMLDup representing precision & recall values for a) Cora dataset b) CD2 Dataset

VI. CONCLUSION

The new method MULTIDUP 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. The proposed method run faster than previous methods described in[1][2] with accurate results.

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.