Efficient Duplicate Detection and Elimination in Hierarchical Multimedia Data

Manjusha R. Pawar
Department of Computer Engg.
Late G. N. Sapkal College of Engg.
Nashik,
Savitribai Phule Pune University, Pune, India.

J. V. Shinde
Department of Computer Engg.
Late G. N. Sapkal College of Engg.
Nashik,
Savitribai Phule Pune University, Pune, India.

ABSTRACT
Today's important task is to clean data in data warehouses which has complex hierarchical structure. This is possibly done by detecting duplicates in large databases to increase the efficiency of data mining and to make it 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 semi-structured or hierarchical structure and the problem arose is how to detect duplicates on XML data. Also due to differences between various data models, the algorithms which are for single relations cannot be applied on XML data. The objective of this project is to detect duplicates in hierarchical data which contain textual data and multimedia data like images, audio and video. It also focuses on eliminating the duplicates by using elimination technique such as delete. Here Bayesian network is used with modified pruning algorithm for duplicate detection, and experiments are performed on both artificial and real world datasets. The new XMLMultiDup method is able to perform duplicate detection with high efficiency and effectiveness on multimedia datasets. This method compares each level of XML tree from root to the leaves computing probabilities of similarity by assigning weights. It goes through the comparison of structure, each descendant of both datasets and find duplicates despite difference in data.

General Terms
Duplicate detection, Data cleaning.

Keywords
XML Data, Bayesian network, Pruning**.**

1. INTRODUCTION
XML is popular for data storage in data warehouses, but it comes with errors and inconsistencies to real-world data [1], hence, there is a need of XML data cleansing [2]. By recognizing and eliminating duplicates in XML data [3] could be the solution; thus strategy based on Bayesian Network to detect duplicates and the method to eliminate that duplicates can be used with pruning technique.

Various algorithms [4] and techniques have been proposed and implemented for duplicate detection [1] on single relations. But XML data [5] has complex and hierarchical structure therefore the techniques which are being used for single relations cannot be applied on XML data. Although there is a long line of work on identifying duplicates in relational data, only a few solutions focuses on duplicate detection in more complex structures [6], 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[7] 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 difficult 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, the <$film_1$> 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, first compare the corresponding structure, values and contents of both objects. In this work, this paper proposes that if structure is found similar first then next step is to find similarity of the values and further proceed for the duplicate detection in multimedia data. Also if multimedia data in both trees found similar then there is elimination to the trees so as to minimize size of memory space 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.

The main contribution compared to previous work and objectives of proposed system are 1) to detect duplicates in hierarchical data which contain multimedia data e.g. images, audio and video using XMLMultiDup method. 2) To compare

datasets according to user choice and display results e.g. only structure or contents to be compared. 3) To increase efficiency and effectiveness of duplicate detection in comparison of multimedia databases. 4) To eliminate duplicates, to reduce size of databases in data warehouses. 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.

Structure

This paper is organized as follows: Section 2 presents related work. Section 3 summarizes methodology of the proposed system. The strategy of proposed system is presented in Section 4. Working environment and results of proposed system over existing system using artificial and real world dataset are presented in Section 5. Finally, Section 6 and 7 concludes and presents suggestions for future work respectively.

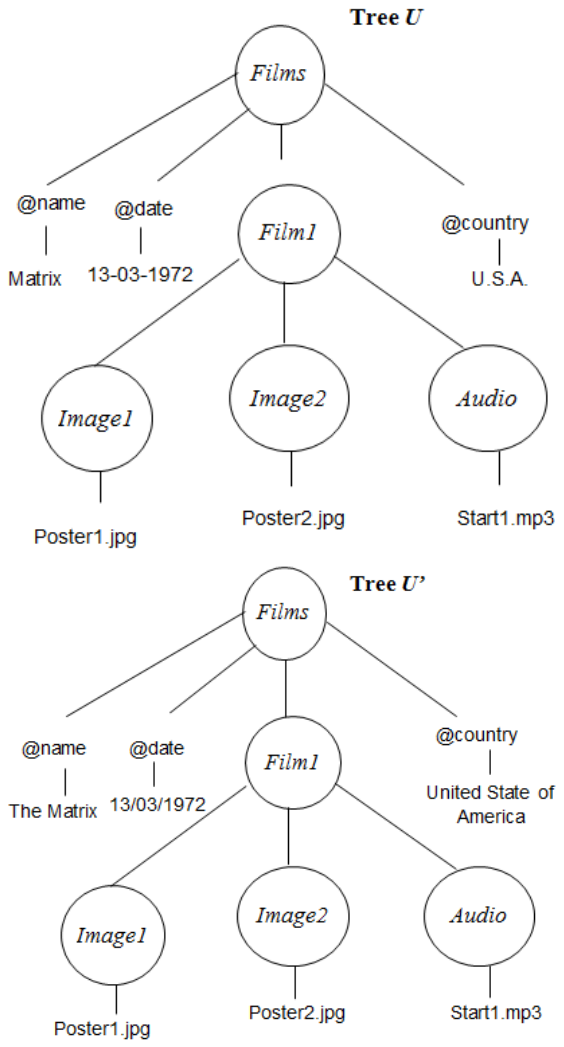

Figure 1: Two XML elements U and U' that represent a same Film. Nodes in circle are labeled by their XML tag name.

2. RELATED WORK

In [8] 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 [9] 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. This paper presents four different strategies to define the similarity function using the vector space model and describes experimental results that show, for Web sources of three different application domains, this approach is quite effective in finding objects with similar identities, achieving precision levels above 75%.

DogmatiX[10] is a generalized framework for duplicate detection, 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 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, second scenario yields poorer results.

Milano Propose a novel distance measure for XML data, the structure aware XML distance [11] 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; and normalize this value with the sum of their lengths

In [12] 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. Figure 1 shows two XML trees which contain duplicate data although value represented differently.

Base for proposed system presented in [1], has extended work done in [12] by adding pruning algorithm to improve the efficiency of the network evaluation. This pruning technique is used to reduce the no. of comparisons where the pairs which are incapable of reaching a given duplicate probability threshold are discarded. 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.

3. 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 real world data and artificial dataset is used. The first phase of this XMLMultiDup method is to input XML data 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. The system first computes prior, computational and final probabilities using Bayesian Network. Algorithm 1 is used for this whole recursive process and shown in figure 3, but there is an issue of complexity of O (n X n'). Hence a pruning technique is used which reduces no. of comparisons using pruning algorithm. If structure and values found duplicate, the contents of multimedia data will be later compared by MD5 hash Key algorithm, which can be replaced by any signature based algorithm to compare multimedia data. Figure 2 shows the architecture of proposed system, which includes combination of three algorithms.

1. Bayesian network
2. Proposed Pruning Algorithm
3. MD5 Hash key Algorithm

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 is elimination 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.

4. DESIGN AND SPECIFICATION

4.1 Bayesian network Construction

This algorithm assumes 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 example of figure 1, the real-world types are *Tfilms = movie*, *Timage = posters*, *Taudio= audioclip*. For simplicity, in the subsequent definitions, it is necessary to 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.

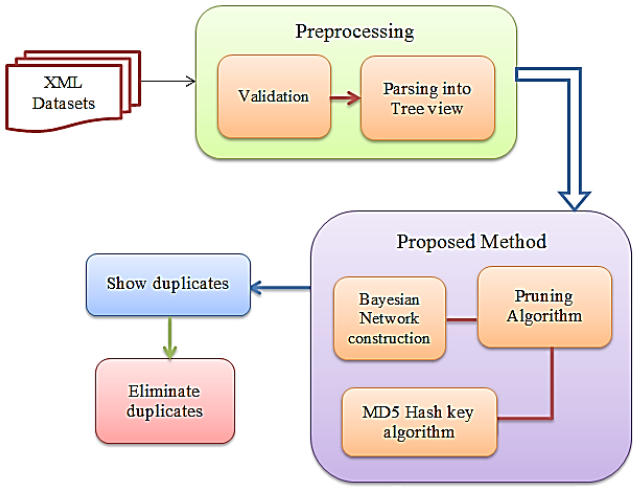

Figure 2: Architecture of Proposed System

Algorithm 1 ***BNGen(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: A directed graph G = (N, E)
/* ---------- 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|T_{ti} 3 = r};
4. S'_r = {t'_i∈ S'|Tt'_i = r};
/* ---------- BN Construction ---------- */
5. foreach type r ∈ $S \cup S'$ do
/* Nodes with single occurrence */
6. if $|S_r| \leq 1$ *and* $|S'_r| \leq 1$ then
7. Insert into N a node t_{ii};
8. if $V_i \cup V'_i \neq \emptyset$ then
9. Insert into N a node Vtii ;
10. Insert into E an edge from this node to node t_{ii};
11. if $C_i \cup C'_i \neq \emptyset$ then
12. Insert into N a node Ct_{ii} ;
13. Insert into E an edge from this node to node t_{ii};
14. if *node* Vt_{ii} *was created* then
15. foreach *attribute* $a \in V_i \cup V'_i$ do
16. Create a node t_{ii}[a];
17. Insert an edge from this node to node Vt_{ii};
18. if *node* C_{tii} *was created* then
19. G' = (N',E') ← *BNGen*(C_i,C'_i);
20. foreach *node* $n \in N'$ do
21. Insert n into N;
22. foreach *edge* $e \in E'$ do
23. Insert e into E;
24. foreach *node* $n \in N'$ *without outgoing edges* do
25. Insert an edge in E from n to node C_{tii} ;
/* Nodes with multiple occurrences */
26. else if S_r *or* S'_r *contain more than one tag each* then
27. Insert into *N* a node t_{**};
28. foreach tag $t_i \in S_r$ do
29. Insert into *N* a node t_{i*};
30. Insert into *E* an edge from this node to node t_{**};
31. foreach tag $t'_j \in S'_r$ do
32. Insert into N a node t_{ij} ;

33. Insert into E an edge from this node to node ti∗;
34. foreach newly created node tij do
35. Similar to processing of node tii (lines 8-25), second subscript i is replaced by j…

In summary

1. Input: Two sets of xml trees u and u'.

   u contains (t,v,c) and u' contains(t',v',c')

   where t-root,v-(attribute,value),c-sub tree

5 If t has value node then create a node of label V and place it as left child of t.

6 Place the attributes value as a child of node V

7 If t has children, create a node of label c and repeat step 1

8 If nodes are of same type, create a node ac and create children as equal to number of same type in left tree.

Repeat the step 1.

9 Bayesian Network is constructed.

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. In following trees U and U' of Fig. 1, this process can be represented by the Bayesian Network of Fig. 2.

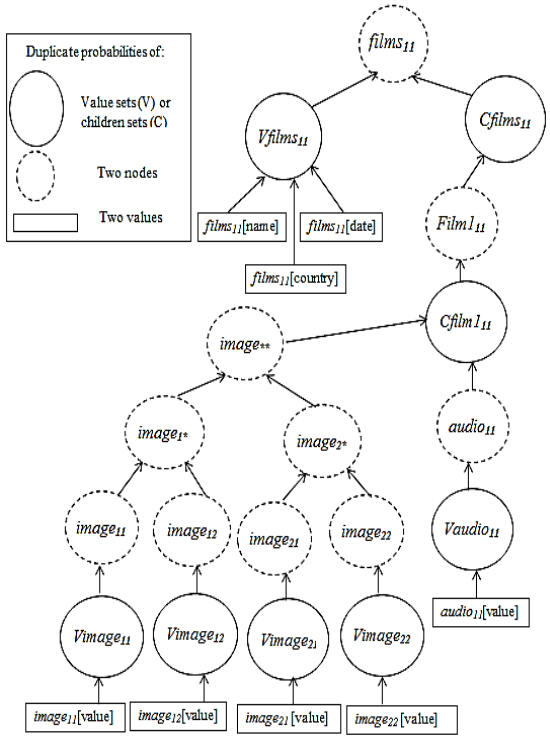

Figure 3: BN to compute similarity of trees in Fig. 1.

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.

4.2 Proposed Pruning Technique

To improve the efficiency and effectiveness of algorithm 1, following algorithm is used which uses some factors which effects 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 for Proposed Pruning Method

Algorithm: XMLMulDup(N)

Input: The node or subtree N for which algorithm will detect duplicates.

Output: Exact and Partial Matching Pair(s) of N' with N and their count.

1. *WN* ← 0 {Initially Weight of selected node N}
2. **if** N *is Value or N is Value Node* **then**
3. WN ← *getSimilarityScore(*N*)* {Similarity Value of N}
4. **else if** N is a Sub Tree **then**
5. P ← *getParents*(N) {collect all Parents P of node N)
6. **for each** *parent* $p_i \in$ P **do**
7. **if** p_i *is value node* **then**
8. WN ← WN + p_i's *similarity Value.*
9. **else if** *it's value is a Multimedia Data* **then**
10. *calculate Hash Key matching status and*
11. WN ← WN+Status value.
12. **else**
13. **go to** *step 5. and calculate WN*
14. **end if**
15. **end for**
16. **end if**
17. P(N) ← WN/*No. of parents.*{ calculate probability of N}
18. **if** P(N) = 1 **then**
19. *Show Matching Pair as Exactly matched.*
20. **else if** P(N) ≥ 0.6 **then**
21. *Show Matching Pair as Partially matched*.
22. **else**
23. *End network evaluation.*
24. **end if**
25. **return** *count of Exact and Partial or none matched pairs*.

4.3 MD5 Hash key Algorithm

The proposed system will use this algorithm for comparing the contents of multimedia path 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 the pruning algorithm is used for increasing the efficiency and effectiveness of the Bayesian network algorithm. But while doing this some datasets may contain the multimedia databases and it is needed 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, duplicates within multimedia files which are included in XML datasets are detected..

The main advantage of using this algorithm is it finds hash key for given four conditions 1) the path of multimedia is same but contents are different, 2) the path of multimedia is different but contents are same and 3) the path of multimedia is different and contents are also different.

5. EXPERIMENTS ON DATASETS

5.1 Datasets

Tests were performed using two different data sets, having simple text and having multimedia data representing different data domains. The first two data sets, CD2 and Cora consist of XML objects taken from a real database and remaining two artificially polluted by inserting multimedia data. One more dataset i.e. CD which is artificially polluted and different types of errors, such as typographical errors, missing data, and duplicate erroneous data [6]. The data sets vary in size from 9,763 objects (CD 2) through 1878 (Cora). All data sets contained objects nested in a hierarchy of up to three levels. The Cora and CD data sets are available at the Hasso Plattner Institute website.

5.2 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. Thus, the XML datasets which contain multimedia data is necessary as input for this system and it can be artificial dataset or real world dataset which contain multimedia data. The proposed system is implemented in integrated development environment of Microsoft Visual Studio 2010 and Dot Net Framework 4.0 with windows platform and on Intel dual core CPU at 1.9 GHz , 2 GB of RAM and 40 GB HDD.

5.3 Results

The precision and recall measures are applied to evaluate effectiveness. And for efficiency measure both XMLDup and XMLMultiDup algorithms are compared with respect to their runtime. The precision of both methods is high for the datasets which contain only textual data but XMLDup drops its precision when datasets contain any type of Multimedia data. While the proposed XMLMulDup method shows high precision for the same dataset with multimedia data. Fig. 3 and fig. 4 shows the Precision/Recall results obtained for each experiment.

When proposed system is used on Real datasets CD2 and Cora, the precision and recall graph curves are near 100 percent as shown in fig. 3 and fig. 4. Both figures shows the precision/recall results obtained by proposed and existing method on real and artificially polluted dataset by adding multimedia data. The proposed system shows 100 percent of precision on multimedia data while this facility is not contributed by existing system.

Table 1 presents the average precision on each dataset which is above 95 for proposed system and suddenly drops for existing system as it operate on multimedia data. R-precision is the precision taken at cut off R, where R is no. of duplicates in dataset. It shows the high precision when the path of value is same and contents are different and vice versa.

Table 1. Performance Achieved Using Proposed Method on Real and Artificial Dataset

Dataset	Average Precision		R-Precision		Maximum Recall	
	XMLDup	XMLMulDup	XMLDup	XMLMulDup	XMLDup	XMLMulDup
CD 2	96	99	82	82	99	99
Cora	87	98	75	75	80	80
MultiCD2	68	99	64	82	71	99
MultiCora	52	98	56	75	68	80

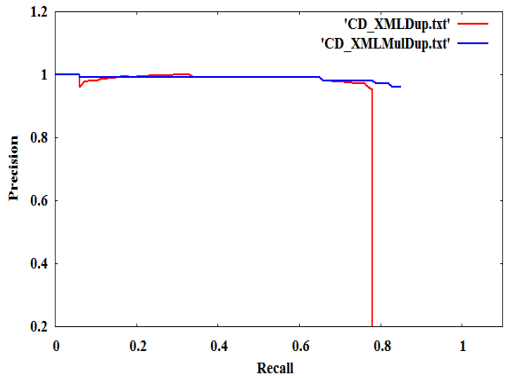

Figure 3: Comparison Result for XMLDup and XMLMulDup representing precision and recall values for CD2 Datasets.

Table 2. Performance Achieved Using Proposed Method

Dataset	XMLDup (pf=1)	XMLDup (pf=0.4)	XMLMulDup
Cora	00:02:41	00:02:04	00:01:06
CD 2	02:07:17	00:48:32	00:03:38
MultiCora	00:02:52	00:02:09	00:01:15
MultiCD2	01:25:06	00:42:22	00:03:53

Table 3. Performance Achieved on Artificial Dataset CD after using proposed pruning algorithm

Sorting strategy	Time	
	XMLDup	XMLMulDup
Unsorted	00:00:43	00:00:30
Depth	00:00:43	00:00:32
AvgSS	00:03:12	00:01:52
Depth+ AvgSS	00:00:43	00:00:39

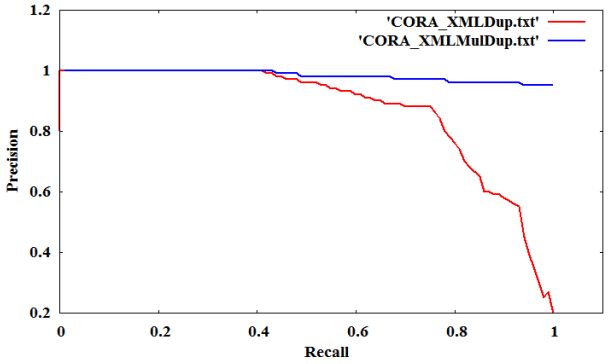

Figure 4: Comparison Result for XMLDup and XMLMulDup representing precision and recall values for CORA Datasets.

Table 2 shows the time performance values with respect to pruning factor of existing system and proposed system. It shows that if pruning factor is increased the runtime also increases but in proposed system there is no user intervention to provide pruning factor. Hence there is lossless strategy used for proposed system. It also shows the maximum recall achieved for each dataset which slightly drop for CD2 dataset when it contains multimedia data.

Table 3 shows the runtime of both methods on CD which is artificial dataset polluted by some dirty data. It shows result on unsorted CD dataset and also compares with the result if test performed on dataset with respect to depth and average string size. When depth is considered, the node having more important information is kept nearer to the root, hence it is evaluated first. And average string size means the value having smaller string size is kept first so as to evaluate first i.e. cheaper comparison first and fast. Thus in both cases it shows the small improvement as compared to unsorted detests using both methods.

6 CONCLUSION AND FUTURE WORK

The new method XMLMultiDup presents a procedure for XML duplicate detection which contains various types of multimedia databases. 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 XML datasets and Relational database. Second approach will need conversion of relations to the XML data and then go for first approach and further apply above discussed algorithms.

The proposed method can be extended 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.

7 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] 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.

[3] 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

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

[5] 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.

[6] 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.

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

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

[9] 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.

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

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

[12] 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.

[13] F. Naumann and M. Herschel, "An Introduction to Duplicate Detection. Morgan and Claypool, 2010.

[14] A.M. Kade and C.A. Heuser, "Matching XML Documents in Eng. Highly Dynamic Applications," Proc. ACM Symp. Document Eng.

[15] M. Weis and F. Naumann. Duplicate detection in xml. In SIGMOD Workshop on Information Quality in Information Systems (IQIS), pages 10–19, Paris, France, 2004.

[16] http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/.

[17] http://www.cs.utexas.edu/users/ml/riddle/data.html.

[18] https://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_Naumann/

[19] http://www.researchgate.net/publication/225867479_An_Overview_of_XML_Duplicate_Detection_Algorithms

[20] http://se-pubs.dbs.uni-leipzig.de/files/Weis2006ADuplicateDetectionBenchmark.pdf

[21] http://www.morganclaypool.com/doi/abs/10.2200/S00262ED1V01Y201003DTM003

[22] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.8263&rep=rep1&type=pdf#page=14

[23] http://www2.cs.uni-paderborn.de/cs/ag-boettcher/lehre/SS05/sem-ss05/SIGMOD05