Xml Duplicate Detection over Bayesian Network Using Decision Tree Induction Algorithm

S. Krishnakumar
IV Semester, ME, Dept of CSE
k.Ramakrishnan college of Technology
Trichy

Abstract— **Duplicate detection is a nontrivial task because the duplicates are not exactly equal they are more or less equal and also structure of the xml also varies from one xml data with other which represent the same real world object. To find duplicates in xml data is done by Decision tree induction algorithm which removes accurate matching xml data and incomplete data. Thus it reduces the number of records to classify. From the output of Decision tree induction algorithm, Bayesian network is created to obtain the probability of two xml objects being duplicates. To improve the efficiency of the evaluation, a novel pruning strategy is used. The efficiency and effectiveness is high because duplicate detection is done in two levels one by the Decision tree induction algorithm and then by Bayesian network which is reduced by pruning.**

Keywords— **Decision tree induction algorithm, Bayesian network, Network pruning, Data cleaning**

1. INTRODUCTION

XML is increasingly popular, especially for data published on the Web and data exchanged between organizations. XML data is semi-structured and is organized hierarchically.

Fuzzy Duplicate detection is of critical practical relevance in many applications, including data cleaning data integration and personal information management. The most prominent application area for duplicate detection is customer relationship management (CRM), where multiple entries of the same customer can result in multiple mailings to the same person, incorrect aggregation of sales to a certain customer,etc. Other application areas include bioinformatics, catalog integration, and in general any domain where independently collected data is integrated. Ironically, the problem has been considered under various names, e.g., record linkage, merge/purge, reference reconciliation, or entity resolution

In this paper, we refer to the problem as fuzzy duplicate detection, or duplicate detection for short.The word Fuzzy implies that the result of matching two xml datas is not exactly true or false.the result lies between true and false therefore it is between 0 to 1. If the result obtained is more than Threshold value then it is taken as true and less than threshold value means it is taken as false. Duplicate detection has been studied extensively for relational data stored in a single table . In this case, the detection strategy typically consists in comparing pairs of tuples (each tuple representing an object) by computing a similarity score based on their attribute values.This will not suit duplicate detection in xml data because the content of xml data may be different but still represents the same object. The xml data is taken from different source, the content and structure of the xml data may vary but they represent same entity. The content of the xml varies due to the errors, different semantics and misspelling. Example of XML data with content and structural difference is shown below.

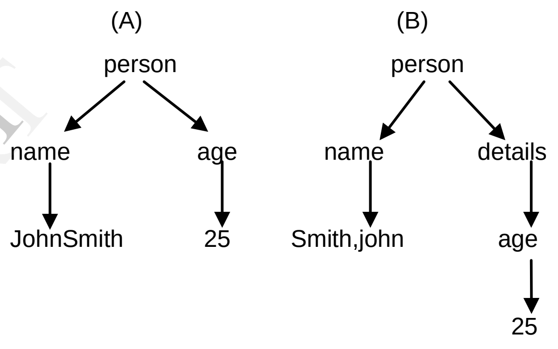

Fig. 1.1 Two xml elements represent same person

Fig. 1.1 represents the xml data difference occurs in the name element, which is spelled in document (A) as John Smith, while in document (B) it is Smith, John. To find out that John Smith and Smith, John represent the same name, a simple string comparison is not sufficient. A string similarity function is required to discover that these two strings represent the same information. Such similarity functions get two strings as input and return a value indicating how similar (or how different) they are. Some well-known similarity functions are Edit Distance, Jaro-Winkler, Monge-Elkan and SoftTF.IDF, among others. When such functions are used, we usually consider that the input strings match if their similarity value is above a certain threshold. Structural difference exits in the above example. In document (A) the age element is embedded in to person element,but in document(B) the age element is embedded in to details element. The structural difference can be identified from the DTD(Document Type Descriptor).

The method we present in this paper uses the Decision induction tree to remove the xml records which has missing values and the xml records which are exactly duplicate. The classifier Decision induction tree reduces the number of records that is stored thus it reduces the number of comparison made in the Bayesian network. string similarity function takes longer time to compare the strings, to avoid unnecessary comparison a technique called Network pruning is used.

Structure. This paper is organized as follows: Section 2 presents related work. Section 3 summarizes overall architecture of system. Section 4 describes the classifier Decision tree induction. Section 5 describes the Bayesian network formation. To accelerate the evaluation of Bayesian network by pruning is presented in Section 6.Finally, in Section 7 we conclude and present suggestions for future work.

2. RELATED WORK

Only more recently research been performed with the specific goal of discovering duplicate object representations in XML databases [1], [2], [3], [4], [5]. These works differ from previous approaches since they were specifically designed to exploit the distinctive characteristics of XML object representations: their structure, textual content, and the semantics implicit in the XML labels. We briefly describe the main features of these methods here.Comparison of xml elements based on parents, children and structure is discussed in [1].Heuristics is proposed to choose any one of the above. It consists of three modules namely candidate definition, duplicate definition and duplicate detection. the first two provide the definitions necessary for duplicate detection (i.e., the set of object representations to compare and the duplicate classifier to use), the third component includes the actual algorithm. Demerit is Heuristics selection is non automatic. The similarities between two nodes are calculated by element's content, node's name and node's path in [2]. The problem of defining which parts of two xml data contain the same information is solved. It focuses only on effectiveness not in efficiency.

In [3], not only the duplicate status of children is considered, also the Probability of descendants being duplicate is also considered. Bayesian Network is able to accurately determine the probability of two xml objects being duplicate in [3].Demerit is run time is high. It is effective but has less efficiency. The work done in our proposed method is to increase the efficiency of the duplicate detection done in Bayesian network by using Decision tree induction classification and pruning Network.

The problem of integrating xml data through correlations realized as join operations is done in [4]. Lower and upper bounds for the tree edit distance metric between two trees are calculated. It focuses only on Focus only on efficiency and not in effectiveness. In [5] Tree edit distance is computationally infeasible for unordered data. Hence new distance for xml data called structure aware xml distance is proposed based on the concept of overlays. An overlay between two XML trees U and V is a mapping between their nodes, such that a node u which belongs to U, is mapped to a single node v which belongs to V if, and only if, they have the same path from the root. This measure is then used to perform a pairwise comparison between all candidates. If the distance measure determines that two XML candidates are closer than a given threshold, the pair is classified as a duplicate.

3. OVERALL ARCHITECTURE

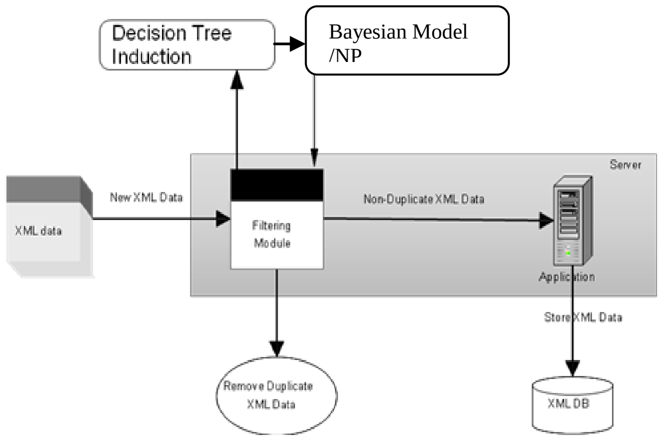

Fig.3.1. Overall Architecture of duplicate detection.

Fig.3.1 represents the overall architecture of duplicate detection. The xml data is processed by Decision tree induction to remove accurate duplicates and xml data whose values are missing. This process reduces the amount of work done by Bayesian network and Network pruning by reducing the xml data to compare.

The classified xml data are given to construct the Bayesian model, from where the four types of conditional probabilities are calculated. To evaluate the conditional probabilities string similarity function is used.string similarity function is a costly process because it takes more time and reduces the efficiency of the system. To improve the efficiency of the system Network pruning is used. It reduces the number of comparison by following strategy like sorting and computing the conditional probability of the system whenever the string similarity function is calculated. If the conditional probability has reached less than the threshold value then the comparison is stopped and the xml data is declared as non duplicate. Thus the comparison of xml data is stopped in midway and increases the efficiency of the system.

The duplicate xml data is removed and only non duplicate is stored. This improves the memory usage utility and increases the searching process faster while retrieving the data. Thus it increases the optimization of memory. The duplicate detection of xml data is done effectively and efficiently. Recall and precision are used to evaluate the effectiveness of our system. Time consumption is evaluated to know the efficiency of the system. If it takes less time to execute, then the algorithm is more efficient.

4. DECISION TREE INDUCTION

Decision tree is formed to match the new xml data with the existing xml data. Comparing new data with every xml data takes longer time so it reduces the efficiency of the duplicate detection. To reduce the time, Decision tree induction is formed from the existing xml data. The new xml data is compared with the Decision tree from the root to the leaf where the new data matches the path from root to the leaf.

4.1 Decision Tree Induction Construction

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data

4.1.1 Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).entropy is used to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one. If the entropy is more than zero then the attribute has to be split and if it is zero then the attribute becomes leaf.

$$\text{ENTROPY (T)} = \sum_{j=1}^{m} p_i \log_2 p_j$$

$$\text{ENTROPY (T,X)} = \sum_{v \in values(A)} \frac{|X_v|}{|T|} \text{Entropy}(X_v)$$

4.1.2 Information Gain

Information gain is calculated to select the attribute as a node from the set of attributes. Otherwise it is also said that it can be used to calculate the splitting attribute. Other methods to find the splitting attribute are Gain Ratio, Gini Index and Minimum Description Length. In this paper we use Information Gain as the splitting attribute.

$$\text{GAIN (T, X)} = \text{ENTROPY (T)} - \text{ENTROPY (T, X)}$$

Where T is the Target attribute and X is one of the attribute of the xml data. The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).The attribute which has highest information gain is considered as a node.

4.1.3 Splitting Criteria

The splitting criteria is set to make branch out of the node. It depends on the value of the attribute i.e. node. If the value is discrete then the node is split according to the values of the node.If the value is continuous then split point is found. The branches are made by values greater than split point and values lesser than split point.

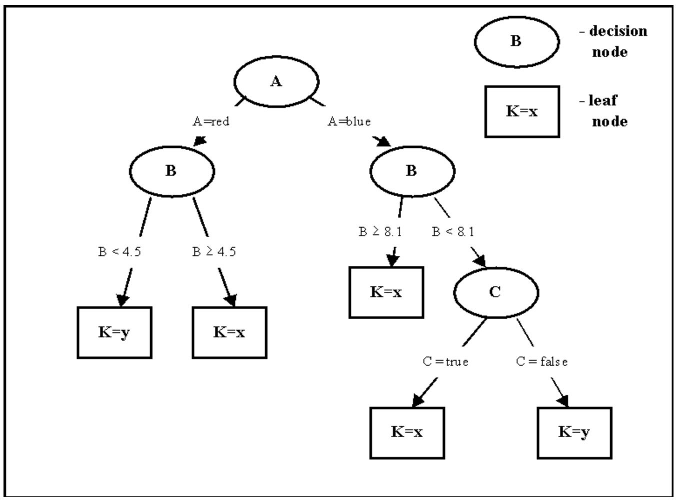

Fig.4.1. Example of Decision Tree

4.2 Decision Tree Induction Algorithm

Input: Dataset of xml data

Step 1: Calculate entropy of the target.

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain.

Step 3: Choose attribute with the largest information gain as the decision node.

Step 4: Branches from decision node is done by split criteria.

Step 5: A branch with entropy of 0 is a leaf node. Then further splitting of node is avoided.

Step 6: A branch with entropy more than 0 needs further splitting.

Step 7: The algorithm runs recursively on the non-leaf branches, until all data is classified.

5. BAYESIAN NETWORK MODEL

Bayesian networks provide a graphical formalism to explicitly represent the dependencies among the variables of a domain, thus providing a concise specification of a joint probability distribution. This representation is based on a directed acyclic graph where a set of random variables makes up the nodes of the network and a set of directed links connects pairs of nodes. In this graph, an edge from one node to another means that the first has a direct influence on the second. This influence is quantified through a conditional probability distribution function correlating the states of each node with the states of its parents.

5.1 Bayesian Network Construction Algorithm

Input: Two sets of xml trees u and u1.u contains (t,v,c)and u1 contains(t1,v1,c1)

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

Step 1: If t has value node then create a node of label V and place as left child of t.

Step 2: Place the attributes value as a child of node V

Step 3: If t has children .create a node of label c and repeat step1.

Step 4: 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.

Step 5: Bayesian Network is constructed.

5.2 Computing the Probabilities

5.2.1 Prior Probabilities

Prior probabilities can be defined based on a similarity function sim(.) between the values, normalized to fit between 0 and 1. However, it is sometimes not possible, or not efficient, to measure the similarity between two attribute values. In this case, we define the probability as a small constant ka, named the default probability, representing the possibility of any two values being duplicates before we observe them. Other similarity function exists are Edit Distance, Jaro-Winkler, Monge-Elkan and SoftTF.IDF.choosing efficient similarity function will enhance the effectiveness and efficiency of the duplicate detection.

5.2.2 Conditional Probability

Conditional probability 1 (CP1): The probability of the values of the nodes being duplicates, given that each individual pair of values contains duplicates. if all attribute values are duplicates, we consider the XML node values as duplicates. if none of the attribute values are duplicates, we consider the XML node values as nonduplicates; 3) if some of the attribute values are duplicates, we determine that the probability of the XML nodes being duplicates equals a given value, w_a.

Conditional probability 2 (CP2): The probability of the children nodes being duplicates, given that each individual pair of children are duplicates. The more child nodes in both trees are duplicates, the higher the probability that the parent nodes are duplicates.

Conditional probability 3 (CP3): The probability of two nodes being duplicates given that their values and their children are duplicates.

Conditional probability 4 (CP4): The probability of a set of nodes of the same type being duplicates given that each pair of individual nodes in the set is duplicates.

5.2.3 Final Probability

Once all prior and conditional probabilities are defined, the Bayesian network can be used to compute the probability of two XML trees being duplicates, i.e. *P*(*t*), where *t* is the tag for the root node of both trees. Final probability is calculated by product of value node and child node of the root node. If the final probability is more than the threshold value then the xml data is duplicate else the xml value is not duplicate.

6. NETWORK PRUNING

To compute the final probability one needs to analyze the whole network and calculate the probabilities for every node. This process, which has a complexity of O(n*m), where n and m are the number of nodes in each XML tree being compared, can be time consuming, especially if we are dealing with a large network. However, when performing duplicate detection, we are usually interested only in objects whose duplicate probability is above a given threshold. This allows us to optimize the network evaluation process. In this section, we propose a novel strategy to reduce the time spent on the BN evaluation.

6.1 Network Pruning Algorithm

Input: Node N from the Bayesian network and has user defined threshold value T.

Step1: Get the parent nodes of N, sort it and assign duplicate probability score as 1.

Step2: If the parent node is a value node, it's probability score is it's similarity value.

Step3: Multiply the probability score of value with the assigned children's probability (1).assign this value to current score.

Step4: If the current score is less than the T then stop network evaluation.

Step5: Else if current score is greater than T then compute recursively with new threshold value

Step6: The output is current score.

Note: New Threshold value =T/current score

6.2 Sorting Nodes

By choosing the appropriate order by which to evaluate the nodes, we can assure that the algorithm makes the minimal number of calculations, before deciding if a pair of objects is to be discarded. we propose three such heuristics: sorting by depth, by average string size, and by distinctiveness. Each of these heuristics corresponds to a different way of ordering the nodes in step 1 of Network Pruning Algorithm, as explained in the following.

The most important information is usually stored in nodes that are placed closer to the root, while nodes with less distinctive power are stored in deeper levels. Therefore, by ordering the nodes according to the depth of the branch to which they belong, we cause the more distinctive nodes to be evaluated first. First evaluating nodes whose values have a smaller average string length. The idea is simply to perform the cheaper comparisons first, expecting that non duplicate nodes to be discarded before the longer strings have to be compared. We should note that, since shorter strings are more likely to be similar than larger strings, this node ordering could delay the reaching of the cut-off threshold. However, we expect the cheaper string comparisons to compensate for the increment in the number of compared strings.

The final heuristic consists in sorting the nodes according to their distinctiveness. We define the distinctiveness of a node n as $\log(A/d_n)$, where A is the total number of objects containing node n and d_n is the number of distinct values of node n. This measure follows the idea that nodes with a high number of distinct values are less likely to be similar and thus should be evaluated first. We note that the overhead introduced by the pre computation of these heuristics is negligible.

7. PERFORMANCE EVALUATION

The performance evaluation is done on different set of xml data. The efficiency of our modified XMLDup is calculated, compared with previous method Dogmatix and shown in graphical form as follows

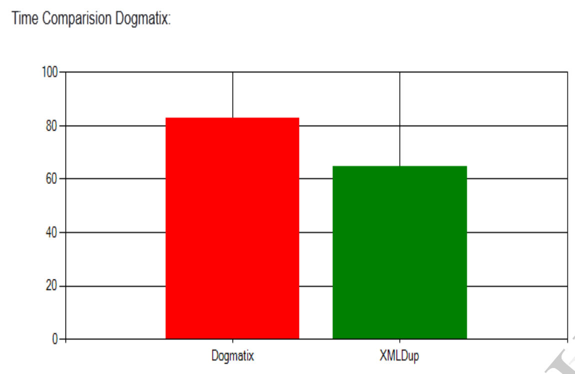

Fig.7.1. Comparison of Dogmatix and Modifed XMLDup

The Fig.7.1 shows the comparison of performance between the Dogmatix and Modified XMLDup.our proposed system, modified XMLDup outperforms previous method Dogmatix in terms of efficiency.That is our proposed method runs faster than the previous method Dogmatix.

Our proposed method runs faster than previous method Dogmatix because of the usage of Decision Tree Induction algorithm which eliminates the accurate matching xml data and missing information, thus avoiding unnecessary formation of Bayesian network, which is followed by Network pruning. Even though usage of Decision Tree Induction algorithm is an overhead, but it still increases the performance of the duplicate detection.

8. CONCLUSION

Decision tree induction is created from the existing xml data. The new xml data is matched with the path of the Decision tree. The node of the path is matched with the value of the xml data. If the values match exactly then the new tuple is not stored. Thus optimization of memory is done. The classified xml data is given to construct the Bayesian network. Prior probability, Conditional probability and Final probability is calculated. Evaluation of probabilities of every node reduces the efficiency hence Network pruning is used to increase the efficiency, sorting of nodes done in the pruning process reduces the computation time thus increasing the efficiency.

The future work of this paper is implementing the xml duplicate detection by other machine learning algorithm which increases the efficiency and effectiveness of the duplicate detection.

REFERENCES

[1]. M. Weis and F. Naumann, “Dogmatix Tracks Down Duplicates in XML,” Proc. ACM SIGMOD Conf. Management of Data, pp. 431-442, 2005.
[2]. A.M. Kade and C.A. Heuser, “Matching XML Documents in Highly Dynamic Applications,” Proc. ACM Symp. Document Eng. (DocEng), pp. 191-198, 2008.
[3]. L. Leitao, P. Calado, 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.
[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]. D. Milano, M. Scannapieco, and T. Catarci, “Structure Aware XML Object Identification,” Proc. VLDB Workshop Clean Databases (CleanDB), 2006.
[6]. P. Calado, M. Herschel, and L. Leitao, “An Overview of XML Duplicate Detection Algorithms,” Soft Computing in XML Data Management, Studies in Fuzziness and Soft Computing, vol. 255.2010
[7]. S. Puhlmann, M. Weis, and F. Naumann, “XML Duplicate Detection Using Sorted Neighborhoods,” Proc. Conf. Extending Database Technology (EDBT), pp. 773-791, 2006.
[8]. L. Leita˜o and P. Calado, “Duplicate Detection through Structure Optimization,” Proc. 20th ACM Int’l Conf. Information and Knowl-edge Management, pp. 443-452, 2011.
[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]. L. Chen, L. Zhang, F. Jing, K.-F. Deng, and W.-Y. Ma, “Ranking Web Objects from Multiple Communities,” Proc. 15th ACM Int’l Conf. Information and Knowledge Management, pp. 377-386, 2006.