Review on Duplicate Detection in Hierarchical Data using Network Pruning Algorithm

Amita Fulsundar
Computer Department,
MET BKC Adgaon, Nashik,SavitribaiPhule Pune University, Maharashtra India.

ABSTRACT
The goal of the data mining process is to extract information from various data sources. Different sources can provide documents that contain data with different structure may be considered as representing the same conceptual information. Solution to this is duplication detection. Duplicate detection is detection of same real world entity in the data sources. Duplicate detection is a necessary task in data cleansing. Various algorithms are proposed for detection of duplicates in relational data, but very few solutions are focused on hierarchical data like XML. Duplicate Detection exactly identifies whether the data is duplicated or not. A peculiar method XMLDup is introduced for duplicate detection in XML data. XMLDup uses Bayesian network to evaluate probability of two XML elements being duplicates. It considers not only the content within the elements but also the way that content is structured. To improve the run time efficiency of network evaluation, a lossless pruning strategy is used. The algorithm achieves high accuracy and recall score in several data sets. The XMLDup perform state-of-the-art in duplicate detection in terms of both effectiveness and efficiency.

Keywords
Duplicate detection, XML, Bayesian networks, data cleaning, and optimization.

1. INTRODUCTION
XML is widely used for data exchange between networks and it is also used for uploading data on web. It has ability to represent data from wide variety of sources. XML data have vital advantage over a relational database, as it acts as a communication medium between different applications. Most of the time XML documents from different sources may contain errors and inconsistencies. It is important to ensure quality of data uploaded on web. But quality of data can be compromised due to introduction of different types of errors, like fuzzy duplicates[1]. Duplicates are multiple representations of same real world entities. Errors are introduced due to typos, misspelling and different representation format. Duplicate data is one of the biggest problems in any data analyst's life. Duplicate detection has practical relevance in many applications, including data cleaning, data integration. Identification of duplicates has become important task as duplicates are not exactly equal. Data may be represented in various formats. It is essential to use a correct matching strategy for identifying if they refer to the same real world entity or not.

Duplicate detection has been studied extensively for relational data stored in a single table. Algorithms performing duplicate detection in a single table generally compare based on attribute values. However, data usually comes in more complex structures. In relational structure the tuples are compared and their similarity scores are computed based on their attribute values[8]. The methods devised for duplicate detection in a single relation does not directly apply to XML data, due to differences between the two data models [5]. The hierarchical relationships in XML provide useful additional information that helps improve both runtime and quality of duplicate detection. The goal of duplicate detection is to detect that XML objects are duplicates, despite the differences in the data.The problem as *duplicate detection* in hierarchical data is discussed.

Consider the two XML elements depicted as trees in Fig. 1. Both represent movie objects and are labeled *mov.* These elements have two attributes, namely the *year* and *title*. They nest further XML elements representing director(*dir*) and casts (*cast*). A casts consists of several actor or actresses (*act*), represented as children XML elements of cast. Leaf elements have a text node which stores the actual data. For instance, year has a text node containing the string "*2006*" as its value.

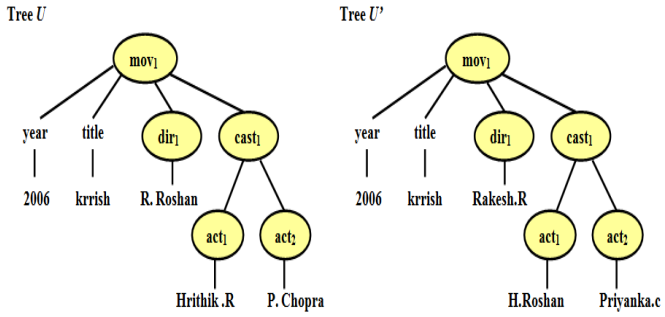

Fig 1: Two XML trees that represent the similar movie.

In this example, the aim of duplicate detection is to detect that both movies are duplicates, despite the differences in the data. To do this, comparisonof the corresponding leaf node values of both objects is done. The hierarchical organization of XML data helps in detecting duplicate *mov*elements, since descendant elements can be detected to be similar, which increases the similarity of the ancestors, and so on in a top-down fashion.

2. RELATED WORK
In this section various algorithms and techniques which were proposed for duplicate detection is explained.

R. Ananthakrishna, S. Chaudhuri, and V. Ganti, developed an algorithm for eliminating duplicates in dimensional tables in a data warehouse, which are usually associated with hierarchies. Hierarchies to develop a high quality, scalable duplicate elimination algorithm have been exploited, and evaluation is done on real datasets from an operational data warehouse [3].

Guha et al., proposed a novel approach to perform approximate joins in XML databases. The main goal was how

to efficiently join two sets of similar elements. Thus, focused was on an efficient implementation of a tree edit distance[2].

M. Weis et.al proposed Dogmatix framework which aims at both efficiency and effectiveness in duplicate detection [5]. The framework consists of three main steps. 1]candidate definition; 2]duplicate definition, which provides the definitions necessary for duplicate detection (i.e., the set of object representations to compare and the duplicate classifier to use) and 3] duplicate detection, which includes the actual algorithm, an extension to XML data [3].In this framework, XML elements based not only on their direct data values, but also on the similarity of their parents, children, structure, etc.

Milano et al. propose a distance measure between two XML object representations that is defined based on the concept of overlays [4]. An overlay between two XML trees U and V is a mapping between their nodes, such that a node uϵU is mapped to a single node vϵV 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. Identification of the object is difficult for XML data because of its structural pliability. For approximate comparisons among XML trees, tree edit distances have been used. However, such distances don't consider the semantics implicit in XML data structure, and their use is computationally infeasible for unordered data.A new distance for XML data called as structure aware XML distance is defined, which solves the problems, together with a polynomial-time algorithm to calculate it.

3. PROPOSED METHOD

For hierarchical data, a probabilistic duplicate detection algorithm called XMLDup is proposed. For duplicate detection, a Bayesian Network model is constructed, and this model is used to compute the similarity between XML object representations. Given this similarity, two XML objects are as duplicates if it is above a predefined threshold [7].

A major result in that XMLDup exceed the performance of existing algorithm more efficiently for XML duplicate detection. A schema mapping step has preceded duplicate detection is assumed, so that all XML elements compared comply with the same schema. It provides a more extensive evaluation of algorithms than in previous work. A distance measure between two XML object representations that is defined based on the concept of overlays is more effective algorithm.

3.1 Construction of Bayesian Network

Bayesian Networks is a directed acyclic graph, where the nodes represent random variables and the edges represent dependencies between those variables [6].

3.1.1 Bayesian Network structure

A proposed approach for XML duplicate detection is based on one basic assumption: The fact that two XML nodes are duplicates depends only on the fact that their values are duplicates and that their children nodes are duplicates. Thus, we say that two XML trees are duplicates if their root nodes are duplicates. To illustrate this idea, consider the goal of detecting that both movies represented in Fig.1.are duplicates. This means that the two movie objects, represented by nodes tagged *mov*, are duplicates depending on whether or not their children nodes (tagged *dir* and *cast*) and their values for attributes year and title are duplicates. Furthermore, the nodes tagged dir are duplicates depending on whether or not their values are duplicates, and the nodes tagged cast are duplicates depending on whether or not their children nodes (*act*) are duplicates. 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. Let us first consider the XML nodes tagged *mov*. As illustrated in Fig.2, the BN will have a node labeled mov_{11} representing the possibility of node mov_1 in the XML tree U being a duplicate of node mov_1 in the XML tree U'.

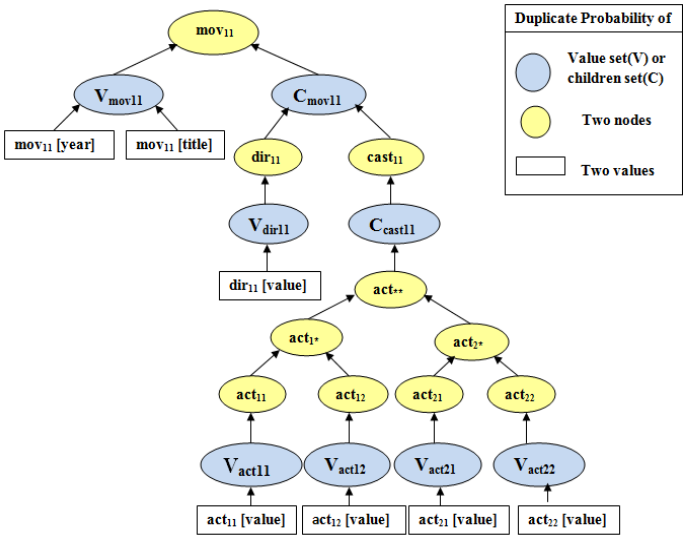

Fig 2: Bayesian Network to compute the similarity of the trees in Fig.1.

Node mov_{11} is assigned a binary random variable. This variable takes the value1 (active) to represent the fact that the XML *mov* nodes in trees *U* and *U'* are duplicates. It takes the value 0 (inactive) to represent the fact that the nodes are not duplicates. In accord with assumption, the probability of the two XML nodes being duplicates depends on 1) whether or not their values are duplicates, and 2) whether or not their children are duplicates. Thus, node mov_{11} in the BN has two parent nodes, as shown in Fig.2. Node V_{mov11} represents the possibility of the values in the *mov* nodes being duplicates. Node C_{mov11} represents the possibility of the children of the *mov* nodes being duplicates. As before, a binary random variable, that can be active or inactive, is assigned to these nodes, representing the fact that the values and children nodes are duplicates or non duplicates, respectively. It is assumed that the probability of the XML node values being duplicates depends on each attribute independently. This is represented in the network by adding new nodes for the attributes as parents of node V_{mov11}, represented as rectangles in Fig.2. In this case, these new nodes represent the possibility of the year values in the *mov* nodes being duplicates and of the title values in the *mov* nodes being duplicates. Similarly, the probability of the children of the *mov*nodes being duplicates depends on the probability of each pair of children nodes being duplicates. Thus, two more nodes are added as parents of node C_{mov11}: node dir_{11} represents the possibility of node dir_1 in tree *U* being a duplicate of the node dir_1 in tree *U'*; node $cast_{11}$ represents the possibility of node $cast_1$ in tree *U* being a duplicate of node $cast_1$ in tree *U'*. Whole process is repeated for these two nodes. However, a slightly different procedure is taken when representing multiple nodes of the same type, as is the cases for the XML nodes labeled *act*. In this case, the full set of nodes is compared, instead of each node independently. Thus, the set of act nodes being duplicate depends on each *act* node in tree *U* being a duplicate of any *act* node in tree *U'*. This is represented by nodes *act*, *act1*, and

act2 in the BN of Fig.2.This approach is not symmetrical, i.e., the network obtained for *U* and *U'* is not always the same as that obtained for *U'* and *U*. However, it would be difficult to satisfy this feature without a significant decrease in efficiency, while we would not expect a high increase in effectiveness.

3.2 Computing Probabilities

As a binary random variable is assigned to each node, which takes the value 1 to represent the fact that the corresponding data in trees *U* and *U'* are duplicates, and the value 0 to represent the opposite. Thus, to decide if two XML trees are duplicates, the algorithm has to compute the probability of the root nodes being duplicates.

3.2.1 Prior Probabilities

In the network of Fig.2, we need to define the prior probabilities of values being duplicates in the context of their parent XML node, i.e., P$(mov_{11}[year])$P$(mov_{11}[title])$ P$(dir_{11}[value])$and P$(act_{ij}[value])$. These 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. Thus, we define P *(tij [a]) = sim (vi[a], vj [a])* if the similarity was measured, and P (*tij* [*a*]) =*ka* if otherwise, where *Vi[a]* is the value of attribute a of the *ith* node with tag *t* in the XML tree. For instance, for the name attribute in the *mov* nodes, we can define *sim (n1, n2) = 1-ned (n1, n2),* where *ned( )* is the normalized edit distance between the strings. The default probability ka can be derived from the distribution of attribute values in the database, or simply be set to a small number.

3.2.2 Conditional Probabilities

CP1: The probability of the values of the nodes is duplicates, given that each single pair of values contains duplicates.

CP2: The probability of the children nodes is duplicates, given that each single pair of children are duplicates. The more child nodes in both trees are duplicates, the higher the probability that the parent nodes are duplicates.

CP3: The probability of two nodes is duplicates given that their values and their children are duplicates.

CP4: The probability of two nodes is duplicates given that their values and their children are duplicates.

4. Algorithm for BN Evaluation

In order to improve the BN evaluation time, a lossless pruning strategy is proposed. This strategy is lossless in the sense that no duplicate objects are lost. Only object pairs incapable of reaching a given duplicate probability threshold are discarded. As stated before, network evaluation is performed by doing a propagation of the prior probabilities, in a bottom up fashion, until reaching the topmost node. The prior probabilities are obtained by applying a similarity measure to the pair of values represented by the content of the leaf nodes. Computing such similarities is the most expensive operation in the network evaluation and in the duplicate detection process in general.

Therefore, the idea behind our pruning proposal lies in avoiding the calculation of prior probabilities, unless they are strictly necessary. The strategy follows the premise that, before comparing two objects, all the similarities are assumed to be 1.The idea is to, at every step of the process, maintain an upper bound on the final probability value.At each step, whenever a new similarity is computed, the final probability is estimated taking into consideration the already known similarities and the unknown similarities that we assume to be 1. When we verify that the network root node probability can no longer achieve a score higher than the defined duplicate threshold, the object pair is discarded and, thus, the remaining calculations are avoided. The process is presented in detail in

Algorithm 1. NetworkPruning(N,T)

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

Output:Probability scoreof the XML nodes represented by N

1: Get the ordered list of parents

2: Set Maximum probability of each parent node as 1

3: currentScore← 0

4: for each node n in List do {Compute the duplicate probability}

5: if n is a value node then

6: For value nodes, compute the similarities

7: else

8: Get new threshold

9: 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

The algorithm takes as input a node N from the BN and a user defined threshold T. It starts by gathering a list of all the parent nodes of N and assuming that their duplicate probability score is 1 .It then proceeds to compute the actual probability value of each of the parents of N.If a given parent node n is a value node, its probability score is simply the similarity of the values it represents. If, on the other hand, n also has parent nodes, its probability score depends on the scope of its own parents, which is computed recursively .However, the algorithm should now be called with a different threshold value, that depends on the equation used to combine the probabilities for node N (line 8).Once the score for node n is computed, the algorithm checks if the total score for N is still above T, and decides whether to continue computing or to stop the network evaluation.

5. CONCLUSION

A new method for XML duplicate detection called XMLDup can be used.The probability of two XML objects being duplicates is determined. The algorithm uses a Bayesian Network. The Bayesian Network model is composed from the structure of the objects being compared. In XMLDup the user needs to provide the attributes, their respective default probability parameter, and a similarity threshold. However, the model is also very flexible, allowing the use of different similarity measures and different ways of combining probabilities. The runtime efficiency of XMLDup is improved by using a network pruning strategy. The XMLDup perform state-of-the-art in duplicate detection in terms of both effectiveness and efficiency.It is intended to extend the BN model construction algorithm to compare XML objects with different structures and apply machine learning methods to derive the conditional probabilities and network structure, based on the existing data. The machinelearning and optimization algorithm such as bee, artificial immune system and BAT algorithm to derive conditional probability values can be used in future.

6. REFERENCES

[1] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Engineering Bulletin*, vol. 23, pp. 3–13, 2000.

[2] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava, and T. Yu,"Approximate XML joins," in *Conference on the Management of Data(SIGMOD)*, 2002.

[3] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminatingfuzzy duplicates in data warehouses," in *Conference on Very LargeDatabases (VLDB)*, Hong Kong, China, 2002, pp. 586–597.

[4] D. Milano, M. Scannapieco, and T. Catarci, "Structure awareXML object identification," in *VLDB Workshop on Clean Databases(CleanDB)*, Seoul, Korea, 2006.

[5] M. Weis and F. Naumann, "Dogmatix tracks down duplicatesin XML," in *Conference on the Management of Data (SIGMOD)*,Baltimore, MD, 2005, pp. 431–442.

[6] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks ofplausible inference*, 2nd ed. Morgan Kaufmann Publishers, 1988.

[7] L. Leita o, 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*.

[8] A. M. Kade and C. A. Heuser, "Matching XML documents inhighly dynamic applications," in *ACM Symposium on DocumentEngineering (DocEng)*, 2008, pp. 191–198.