Duplicate Detection Algorithm In Hierarchical Data Using Efficient And Effective Network Pruning Algorithm

Ashwini G Rathod  
FG Student, CSE, JSPM’S ICOER,Wagholi,Pune, India  
ashwinigrathod@gmail.com

Vinod S Wadne  
Assistant Professor, CSE, JSPM’S ICOER,Wagholi,Pune, India, vinods1111@gmail.com

Abstract—There is a long line of work on identifying duplicates in relational data, only a few solutions are there which focus on duplicate detection in more complex hierarchical structures, example XML data. The main goal of duplicate detection is to either on improving the quality of the detected duplicates (effectiveness) or on saving computation time (efficiency). This paper, focused on a novel method for XML duplicate detection, which is called XMLDup. XMLDup algorithm which uses a Bayesian network to determine the probability of two XML elements, considering information within the elements, and the way that information is structured. This algorithm improve the efficiency of the network evaluation, a novel network pruning strategy, which is capable of significant gains over the un-optimized algorithm, is presented. Through experiments on the data set, we show that our algorithm using network pruning strategy is able to achieve high precision and recall scores in data sets. XMLDup is having better performance on state-of-the-art duplicate detection solution. The proposed system also compares XML object with different structures and apply Dice coefficient algorithm for similarity check, which is the easiest way to detect the duplicate data quickly and efficiently.

Keywords—Duplicate detection, record linkage, XML, Bayesian networks, Data cleaning, Dogmatix, Dice Coefficient.

INTRODUCTION
Electronic data play a central role in numerous business processes, applications, and decisions. As a consequence, ensuring its quality is essential. Data quality, however, can be compromised by many different types of errors, which can have various origins [1]. In this paper, we focus on a specific type of error, namely fuzzy duplicates, or duplicates for short. Data cleansing is an issue of critical practical importance. It is required to ensure high data quality in scenarios such as report generation over data warehouses and CRM. Another application is data integration, where data from distributed and heterogeneous data sources should be combined into a unique, complete, and correct representation for every real-world object. A crucial subtask in data cleansing is duplicate detection (duplicate detection for short). It resolves which entries in a data source actually represent the same real-world object.

For examples, In figure 1 consider the two XML elements describe as tree. Both are correspond to person object and are labeled prs. These elements have two attribute, namely date of birth and name. Advance XML element representing place of birth (pob) and contact (cnt). A contact consist of several address (add) and an email (eml), represented as a children of XML element of cnt. Each leaf element has text node which store actual data. The objective of duplicate discovery is to detect the both persons are duplicates, regardless of the variation in the data. By comparing the corresponding leaf node values of both objects. Hierarchical association of XML data helps to detecting duplicate prs element, since successor elements can be detected to be similar. The goal is to reduce the number of pair wise comparison and to increase the efficiency of pair wise comparison. To compare two candidates, an overlay between their two sub trees is computed. It is not possible to match the same XML elements in different contexts. The weight is assigned to a match is based on a distance measure, e.g., edit distance for string values in leaves. The goal is to determine an overlay with minimal cost and not a proper substructure of any other possible overlay. To construct the Bayesian network, taking two XML elements as input, each rooted in the candidate element and having a sub tree that corresponds to the description. Nodes in the Bayesian network
represent duplicate probabilities of a set of simple XML element, a set of complex XML elements and a pair of complex elements or pair of simple elements.

Paper Organization

Structure of the paper: This paper is organized as follows: Section 2 presents related work of the system. Our strategies to accelerate this algorithm are then presented in Section 3. Section 4 represent system architecture. Section 5 represent the experimental setup and result. Finally, in Section 6 we conclude and present suggestions for future work.

Figure 1. Two XML elements that represent the same person. Nodes are labeled by their XML tag name and an index for future reference.

Related Work

In an existing Duplicate detection has been studied extensively for relational data stored in a single table. Algorithms performing duplicate detection in a single table generally compare tuples (each of which represents an object) based on feature values. Data regularly comes in more intricate structure, e.g., data stored in a relational table relates to data in other tables through strange keys. Detect duplicate in XML is more difficult than detecting duplicates in relational data because there is no schematic distinction between object types among which duplicates are detected and attribute types recitation substance. What makes spare detection a nontrivial mission is the fact that duplicates are not precisely equal, frequently due to error in the data. Therefore, cannot use common comparison algorithms that detect accurate duplicates. Evaluate all object representation using a possibly complex identical approach, to choose if they refer to the similar real-world object or not. By computing a similarity score based on their attribute values we can detect the duplicates which consist comparing pairs of tuples. In this paper various duplicate detection algorithms and techniques for detection are explained. Delphi is used to identify duplicates in data warehouse which is hierarchically organized in a table. Delphi algorithm first evaluate outermost layer and then go for the innermost layer also this algorithm does not compare all pairs in hierarchy. D. Milano et.al[6], suggested a method for measuring the distance of each XML data with one another, known as structure aware XML distance. Similarity measure can be evaluated by using edit distance structure. Whose structure is in similar nature only those portion compare by this method. M. Weis et.al[3] proposed Dogmatix framework which comprises of three main steps: defining candidate, definition of duplicates and detection of duplicate. Dogmatix algorithm compares based on the similarity of their parents, children and structure. It also consider the account difference of the compared elements.

Proposed System

XMLDup system was proposed using Baysian Network and networking pruning, Estimating probabilities value and pruning factor allocation. We now present the XMLDup approach to XML duplicate detection. We first present how to construct a Bayesian Network model for duplicate detection. We present the XMLDup approach to XML duplicate detection. We present how to construct a Bayesian Network model for duplicate detection. We show that how this model compute the similarity between XML object representations.

BN Structure for Duplicate Detection

Approach for BN creation is centered which containing one. 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. Bayesian network provide a succinct requirement of a joint probability distribution. BN is directed acyclic graph, in which node represents random variables and edges represent dependencies between those variables. Two XML nodes are duplicates depends only on their values are
duplicates and their children nodes are duplicates.

XML nodes being duplicates depend on
1) Whether or not their assessment is duplicates.
2) Whether or not their children are duplicates.

A binary random variable can be active or inactive that can be assigned to each node, representing the values and children node is duplicate or non duplicates.

**Contributing the Probabilities:**
Assign a binary random variable to each node which takes the value 1 to represent the corresponding data in tree U and U’ are duplicates and the value 0 to represent opposite. To decide if two XML tree are duplicates, the desire algorithm has to compute the probability of the root nodes being duplicates. To obtain the probabilities associated with the Bayesian Network leaf nodes, which will set the intermediate node probabilities, until the root probability is found between the nodes.

**Network Pruning:**
To improve the BN evaluation time by using lossless pruning strategy. By using lossless approach in the sense of no duplicate object are lost. Network evaluation is performed by doing a propagation of the prior probabilities, in bottom up fashion. Prefer the suitable order by which to evaluate the nodes, it makes the minimal number

of estimate before choose if a pair of object is to be superfluous.

**Pruning Factor Allocation:**
Before evaluation, every node is assumed to have a duplicate probability of 1. This assumed probability is called as the pruning factor. Pruning factor equal to 1 which guarantees that the duplicate probability estimated for a given node is always above the true node probability. Therefore, no duplicate pair is lost. By lowering the pruning factor, this guarantee will be loose. Hence a object pair can be already discarded, even if they are true duplicates. By lower pruning factor, all probability estimates will be small, this will cause the defined duplicate threshold to be reached earlier and the network evaluation to stop sooner. Although we observed a higher loss of recall for the artificial data sets, the same was not observed in the real data sets. The number of comparisons was always lower. Thus, when there is little knowledge of the database being processed, or when manually tuning the pruning factor is not viable.

**Duplicate Detection:**
Evaluate the algorithm both in terms of efficiency and effectiveness. To evaluate the efficiency of XMLDup using node ordering heuristics, varying the different pruning factor, pruning optimization, selecting pruning factors. Data quality on duplicate detection is essential.

**Contribution**
Probabilistic duplicate detection algorithm for hierarchical data called XMLDup. It considers the both similarity of attribute content and generation element, with respect to similarity score. 1) To address the issue of efficiency of initial solution by using novel pruning algorithm. 2) The no of identified duplicates in increased, can be performed manually using known duplicate objects from databases. 3) Extensive evaluation on large number of data sets, from different data domain. The goal is to reduce the number of pair wise comparison is performed, and concerned with efficiently perform each pair wise comparison.

**Dice Algorithm**
The Sørensen–Dice index, also known by other names, is a statistic used for comparing the
similarity of two samples. It was independently developed by the DICE. The index is known by several other names, usually Sørensen index or Dice's coefficient. Both names also see "similarity coefficient", "index", and other such variations. Sørensen's original formula was intended to be applied to presence/absence data, and is

\[ QS = \frac{2|A \cap B|}{|A| + |B|} \]

where \( A \) and \( B \) are the number of species in samples A and B, respectively, and \( C \) is the number of species shared by the two samples; QS is the quotient of similarity and ranges between 0 and 1.

The Sørensen–Dice coefficient is mainly useful for ecological community data. Justification for its use is primarily empirical rather than theoretical. As compared to, Sørensen distance retains sensitivity in more heterogeneous data sets and gives less weight to outliers.

SYSTEM ARCHITECTURE

Probabilistic duplicate detection algorithm for hierarchical data called XML Dup. It considers both the resemblance of attribute content and the relative importance of descendant elements, with respect to similarity score. This algorithm is used to improve the efficiency and effective of run time performance.

The architecture shows the how to find the duplicate in XML data by using information regarding Bayesian network, network pruning and decision tree knowledge. A new XML data can be passed through the filtering module, by using some knowledge about the XML data. After filtering the XML data a noisy or inaccurate data can be removed and stored in a duplicate database.

DICE Algorithm is used for similarity check instead of euclidean distance. In XMLDup Euclidean distance is used for the similarity check. We are using Dice coefficient to calculate the similarity.

Figure 3: System Architecture

EXPERIMENTAL SET UP AND RESULT

We now present the experimental results for the XMLDup algorithm, both in terms of effectiveness and efficiency. To evaluate effectiveness of these algorithm by comparing the XMLDup and Dogmatix state of the art algorithm. Determined the best node arrangement strategy of the Bayesian Node, using arrangement strategy that can be practical to XMLDup on the real world thing, to progress the effectiveness.

Tested numerous variation of pruning factor and evaluate the actual impact on inducing. The unconscious pruning factor optimization provided reliable enhancement in concert. The high thrashing of remind for the synthetic data set is same as original data sets. Precision measures the percentage of correctly identified duplicates, over the total set of objects determined as duplicates by the system. Recall measures the percentage of duplicates correctly identified by the system, over the total set of duplicate objects.

Varying the pruning factor: Another set of experiments was performed to evaluate the improvements achieved by our pruning strategy. We applied the duplicate detection process to all real world data sets, with and without our network pruning strategy. When using the pruning strategy, we tested pruning factor values of 1 (lossless pruning), 0.7, 0.6, 0.5, and 0.4. For values below 0.4 no significant gains in time were achieved and, in some cases, there was a high loss in precision and recall.
VII. CONCLUSION AND FUTURE WORK

In this work network pruning technique derive condition Probabilities are derived from using learning methods; it becomes more accurate results of xmlDup detection than General methods. After that BN Pruning was performed To eliminate or remove duplicate detection of XML data and XML data objects. It produces best efficient results for duplicate detection. The problem of detecting and eliminating duplicated data is one of the major problems occurred in data cleaning and data integration. By using xmlDup provides the effective and efficient result for identifying duplicates. This model is very flexible and allowing different similarity measures. The future implementation is to develop the on different structures and complex hierarchical structures using machine level language.

REFERENCES

