International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Survey on Duplicate Detection in Hierarchical Data

Nikhil Gawande[1], S. R. Todamal[2]

[1]Department of Computer Engineering, JSPM's Imperial College of Engineering & Research, Wagholi, Pune, India

[2]Professor, Department of Computer Engineering, SPM's Imperial College of Engineering & Research, Wagholi, Pune, India

Abstract: ***Although there has been a lot work done on identifying duplicates in relational data, but only a few solutions focus on identifying duplicates in more complex hierarchical structures, like XML data. In this paper, we have demonstrated the novel method for XML duplicate detection, called XMLDup. XMLDup method implements the Bayesian network to calculate and determine the probability of two XML nodes, considering not only the information within the XML nodes, but also the way that the information is structured. In addition, to increase the efficiency of the network evaluation, a novel pruning strategy, capable of significant gains over the unoptimized version of the algorithm, is presented. Through experiments and comparisons, we show that our algorithm is able to achieve high precision and recall scores in several datasets. XMLDup method helps us to improve both efficiency and of effectiveness.***

Keywords: duplicate detection, record linkage, entity resolution, XML, Bayesian networks, data cleaning, optimization

1. Introduction

ELECTRONIC data plays a central role in numerous business processes, applications, and decisions. But it is also important to ensure the quality evaluation algorithm. Data quality, however, can be compromised by many different types of errors in this paper; we focus on a specific type of error, namely fuzzy duplicates. Duplicates are multiple representations of the same real-world object that differ from each other because, consider an example; one representation stores an outdated address. What makes duplicate detection a non-trivial task is the fact that duplicates are not exactly equal, often due to errors in the data. Consequently, we can't use comparison algorithms that detect exact duplicates which are used to identify duplicates in structural data. Instead, we have to evaluate all object representations, using a complex matching strategy, to decide if they pointing to the same real-world object or not. Methods devised for duplicate detection in a single relation can't be directly using for XML data, due to the differences between the two data models. For example, instances of a same object type may have a different structure at the instance level, whereas tuples within relations always have the same structure. But, more importantly, the hierarchical relationships in XML provide useful additional information that helps improve both the runtime and the quality of duplicate detection. We shown this fact based on the following example that we will use throughout the paper.

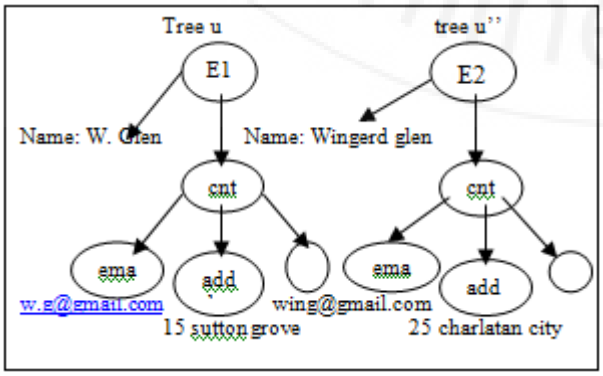

Figure 1: XML elements represents same employee

Consider the two XML elements depicted as trees in *E*. These elements have one attribute, *name*. They nest further XML elements elements of *cnt*. Leaf elements have a text node which stores the actual data.In this example, the goal of duplicate detection is representing email address (*ema*). A contact consists of several addresses (*add*) and an email (*ema*), represented as children XML to detect that both persons are duplicates, despite differences in the data. To do this, we can compare the corresponding leaf node values of both objects. In this work, we propose that the hierarchical organization of XML data helps in detecting duplicate *E* elements, since descendant elements (e.g., *ema* or *add*) can be detected to be similar, which increases the similarity of the ancestors, and so on in a top-down fashion.

Evolved from numerous research communities, especially those from developed countries, the analytical engine within these solutions and software are driven by artificial immune systems, artificial intelligence, auditing, database, distributed and parallel computing, econometrics, expert systems, fuzzy logic, genetic algorithms, machine learning, neural networks, pattern recognition, statistics, visualization and others. There are plenty of specialized fraud detection solutions and software1 which protect businesses such as credit card, e-commerce, insurance, retail, telecommunications industries.

2. Bayesian Network for Duplicate Detection

Bayesian Networks provide a concise specification of a joint probability distribution. They can be seen as a directed acyclic graph, where the nodes represent random variables and the edges represent dependencies between those variables. We first outline how the Bayesian Network for XML duplicate detection is constructed. Afterwards, we explain how probabilities are computed in order to decide if two objects are in fact duplicates. For a more detailed description of Bayesian Networks and their applications.

Our approach for XML duplicate detection is centered around 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 persons represented in Figure 1 are duplicates. This means that the two person objects, represented by nodes tagged *E*, are duplicates depending on whether or not their children nodes (tagged *ema* and *add*) and their values for attributes *name* is duplicates. To identify such duplicates in XML data we calculate the probabilities between these elements.

3. BN Structure foe Duplicate Detection

As we have seen, we assign a binary random variable 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. In our example, this corresponds to computing $P(E1 = 1)$, which can be interpreted as a similarity value between the two XML elements. To obtain this probability, the algorithm propagates the prior probabilities associated to the BN leaf nodes, which will set the intermediate node probabilities, until the root probability is found. In the following, we explain how these probabilities can be defined. For simplicity, we will use the notation $P(x)$ to mean $P(X = 1)$.

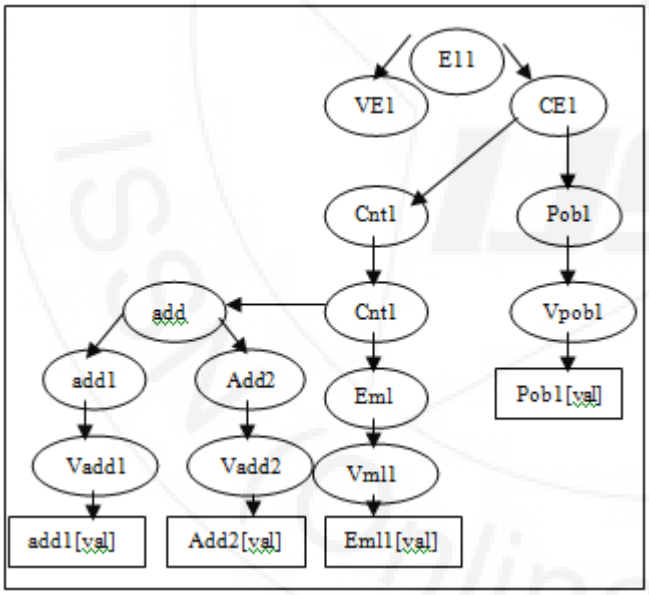

Figure 2: Information extraction

Let us first consider the XML nodes tagged *E1*. As illustrated in Figure 2, the BN will have a node labeled *E*1 representing the possibility of node *E*1 in the XML tree *U* being a duplicate of node *E*1 in the XML tree *U''*. Node *E*1 is assigned a binary random variable. This variable takes the value 1 (*active*) to represent the fact that the XML *E* 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 our assumption, the probability of the two XML nodes being duplicates depends on (i) whether or not their values are duplicates, and (ii) whether or not their children are duplicates. Thus, node *E1* in the BN has two parent nodes, as shown in Figure 2. Node *VE*1 represents the possibility of the values in the *E* nodes being duplicates. Node *CE*1 represents the possibility of the children of the *E* 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.

We assume 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 *VE*1 , represented as rectangles in Figure 2. In this case, these new nodes represent the possibility of the *name* values in the *E* nodes being duplicates. Similarly, the probability of the children of the *E* 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 *CE*1 node *pob*1 represents the possibility of node *pob*1 in tree *U* being a duplicate of the node *pob*1 in tree *U''* node *cnt*1 represents the possibility of node *cnt*1 in tree *U* being a duplicate of node *cnt*1 in tree *U''*. We can now repeat the whole process for these two nodes. However, a slightly different procedure is taken when representing multiple nodes of the same type, as is the case for the XML nodes labeled *add*. In this case, we wish to compare the full set of nodes, instead of each node independently. Thus, we say that the set of *add* nodes being duplicate depends on each *add* node in tree *U* being a duplicate of *any add* node in tree *U''*. This is represented by nodes *add*, *add*1, and *add*2 in the BN of Figure 2.

4. Accelerating the BN Evaluation

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 $O(n \times n\_)$, where *n* and *n''* 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.

4.1 Network Pruning

In order to improve the BN evaluation time, we propose a lossless pruning strategy. 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 be1 (i.e., the maximum possible score). 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.

5. Experiments on duplicate detection

In this section we present an evaluation of the XMLDup algorithm described in the previous sections. We evaluate the algorithm both in terms of effectiveness and efficiency. First, we evaluate effectiveness by comparing it to a state of the art duplicate detection system, called DogmatiX , that proved to be the most competitive so far . We then evaluate the efficiency of XMLDup when using our proposed pruning optimization, node ordering heuristics, varying the pruning factor, and automatically selecting the most adequate pruning factors. The experiments are concluded with a discussion of the results. Testing the impact of data quality on duplicate detection is important to confirm the effectiveness of a given algorithm. In previous work we have shown that XMLDup manages to cope with errors like typos or duplicate erroneous elements without any significant degradation of the results and even performs effectively when dealing with reasonable amounts of missing data. Due to space constraints, those experiments will not be repeated here.

Our tests were performed using seven different datasets, representing five different data domains. The first three datasets, *Country*, *CD* and *IMDB*, consist of XML objects taken from a real database and artificially polluted by inserting duplicate data and different types of errors, such as typographical errors, missing data, and duplicate erroneous data. Experiments were performed to compare the effectiveness and efficiency of the tested algorithms. We now present the experimental results for the XMLDup algorithm, both in terms of effectiveness and efficiency.

5.1 Effectiveness evaluation-

To evaluate effectiveness, we present a comparison of XMLDup to the DogmatiX duplicate detection system. We chose DogmatiX because, although it does not use the object structure to compute similarities, it was shown to be the most competitive in previous tests. In this experiment, only the real world datasets were used. The attributes used for the *Restaurant* dataset were etc. Information visualization process is depicted in figure.

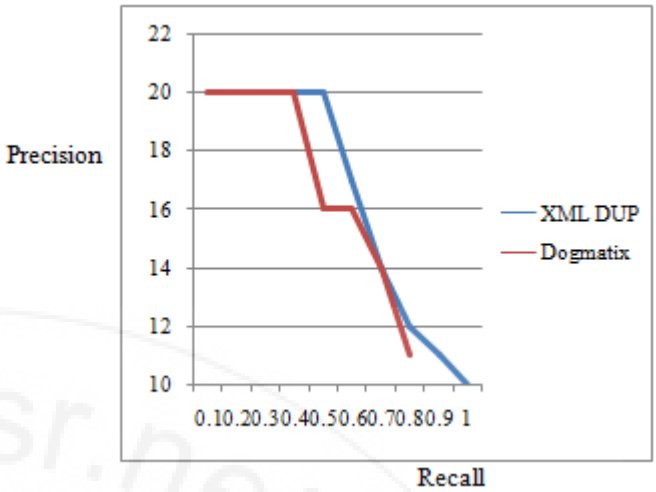

Figure 3: Restaurant Dataset

name, *address*, *phone* and other four attributes representing geographic coordinates. For the *Cora* dataset we used the attributes *author*, *title* and *venue name*. For the *CD 2* dataset, the attributes *artist*, *disc title* and *track title* were used. For the *IMDB+FilmDienst* dataset we used the attributes *title*, *aka-title* and *cast name*. Figure 3 shows the precision/recall results obtained for each experiment.

6. Merits

- XML DUP method provides the consistent capacity to maintain higher precision scores until later recall values.
- It improves the overall 60% of performance over unsorted approach.

7. Demerits

XMLDup requires little user intervention, since the user only needs to provide the attributes to be considered, their respective default probability parameter, and a similar Threshold

8. Conclusion

In this review paper we showed a novel method for XML duplicate detection called XMLDup. Our algorithm implements a Bayesian Network to calculate the probability of two XML objects which are duplicates. The Bayesian Network model is build from the structure of the objects being compared, thus all probabilities are calculated considering not only the information the objects contain, but also the way such information is structured.

XMLDup requires little bit user's intervention, since the user only needs to provide the attributes to be considered, 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. To improve the runtime efficiency of XMLDup, a network pruning strategy is also presented. This strategy can be used in two ways. One is lossless approach, with no impact on the accuracy of the final result set, and a lossy approach, which slightly decreases the recall. In addition to that, the second approach

can be performed automatically without any user intervention. Both strategies have achieved the significant gains in efficiency over the unoptimized version of the algorithm. Experiments done on both artificial and real world data showed that our algorithm has capability to achieve high precision and recall scores in several contexts. When evaluated against another state-of-the-art XML duplicate detection algorithm, XMLDup consistently showed better results in terms of both efficiency and effectiveness. The success presented in experimental results leaves motivation for future enhancement. One of the important works would be to compare XML objects with different structures and apply machine learning mechanisms to derive the conditional Probabilities and network structure, based on the existing data.