Reliable and Consistent Data De-duplication In Hierarchical Structure

M. Krishna Kumar
M.Tech ,Department of Computer Science and Engineering
Kakatiya Institute of Technology and Science,
Warangal , Andhra Pradhesh

M. Venugopal Reddy
Assistant Professor,Department of Computer Science
Kakatiya University
Warangal , Andhra Pradhesh

P.Niranjan Reddy

Professor ,Department of Computer Science and Engineering
Kakatiya Institute of Technology and Science,
Warangal , Andhra Pradhesh

Abstract

Although there exists a long distinctive line of work with identifying duplicates in relational files, only a couple of solutions give attention to duplicate diagnosis in more advanced hierarchical buildings, like XML files. In this paper, current a novel way of XML duplicate detection, termed XMLDup. XMLDup works on the Bayesian network to look for the probability connected with two XML aspects being duplicates, considering besides the information in the elements, but also the approach that details is structured. In inclusion, to help the efficiency in the network evaluate, a book pruning technique, capable connected with significant gains on the un optimized version in the algorithm, is actually presented. As a result of experiments, our algorithm is able to achieve high precision as well as recall scores in several data pieces.

Keywords- *Duplicate Detection, Data Cleaning, XML, Bayesian network , Network Pruning*

1.INTRODUCTION

Data plays an important role in many applications. As the volumes of the data increases its quality is compromised due to the presence of errors in it. These errors may occur due to many reasons. In this paper the focus is on duplicates which is a specific type of error. Finding the duplicates in data is not an easy task because duplicates are not exactly equal hence we cannot find them by normal comparison. We have to compare the object representation to find the duplicates.

In case of relational data stored in table duplicate detection can be done by comparing pairs of tuples by finding the similarity between their attribute values. If the similarity is above the threshold value then those tuples are said to be duplicates otherwise not. This approach will not consider the data stored in another table through foreign keys. XML is well suited for data storage and exchange of data.Inconsistencies occur due to the errors in xml data. These are due to the presence of duplicates in data. The hierarchical relationships in XML provide useful information that helps to improve the runtime and the quality of duplicate detection There are some algorithms proposed for detecting the duplicates in xml data. In our paper we discuss about XMLDup algorithm.

XMLDup is first proposed in [2]. This algorithm is used to detect the duplicates present in the data.

2. LITERATURE SURVEY

There are different application areas for duplicate detection such as Customer relationship management, Bioinformatics, Catalog integration etc. Early work in XML duplicate detection was concerned with the efficient implementation of XML join operations. Guha presented an algorithm to perform joins in xml databases. Their main concern was on how to perform join on two sets of similar elements efficiently. They focused on implementation of a tree edit distance [4], which could be applied in an XML join algorithm. Carvalho and Silva proposed a solution to the problem of integrating tree structured data extracted from the Web. Two hierarchical representations of person elements are compared by transforming both into a vector of terms and by using a cosine similarity measure to find the similarity measure between them. A linear combination of weighted similarities are taken into account , because of its more general nature this approach does not take advantage of the useful features existing in XML databases.

2.1. DogmatiX:

DogmatiX is used previously to find the duplicates in the XML data. As XML data is semi structured and organized hierarchically the object identification is complicated.

Object identification task consists of three main components

1) Candidate Definition

2) Duplicate Definition

3) Duplicate Detection

The aim of candidate definition is to find which objects are relevant for object identification such that those objects are compared. It is formally stated as duplicate candidates describe set of objects of same real-world type which can be extracted from the entire dataset by selection and projection operations. In Duplicate definition duplicates are characterized by their description and a classifier for a pair of objects using a similarity measure. Duplicate detection uses an algorithm to find the duplicate objects in the data. We use this framework to detect duplicate detection in XML data.

Object identification in XML data has additional challenges when compared to the relational data. The definition of objects and their description is not clear in xml data. This algorithm takes an XML document, its Schema S, and a file describing a mapping M of element XPaths to a real world type T as input. To find the duplicates in a specific object the user selects that particular object from the list of objects and then the candidate selection is done. Dogmatix requires heuristics for description selection and similarity threshold used by the duplicate classifier[6]. There are two basic heuristics namely r-distant ancestors and r-distant descendants. It defines a domain-independent similarity measure that is used to classify the pair of objects as duplicates or non-duplicates. Duplicates are then detected by pairwise comparisions by the similarity measure.these pairwise comparisions is reduced by using a filter. After detecting duplicates DogmatiX outputs the XML document. For every cluster of duplicate objects, a dupcluster element is generated and which is identified by a unique object identifier. The duplicate elements within a cluster are identified by their XPaths.

Recent surveys show that most of the researchers who works on data mining projects spends much time in cleaning and preparing the data. Data cleaning problem arises because information from various heterogeneous sources is merged to create a single database [1]. Many different data cleaning challenges have been identified as dealing with missing data, handling erroneous data, record linkage etc. Here we address one challenge called reference disambiguation, which is also known as "fuzzy match" and "fuzzy lookup". This problem arises when entities in a database contain references to other entities. If entities were referred to using unique identifiers, then disambiguating those references would be straightforward. But if entities are represented using properties/descriptions that may not uniquely identify them leading to ambiguity. For instance, a database may store information about two distinct individuals 'Steve L. Wards' and 'Steve E.

Wards', both of whom are referred to as 'S. Wards' in another database. References may also be ambiguous due to differences in the representations of the same entity and errors in data entries. The goal of reference disambiguation is for each reference to identify the unique entity it refers to. The reference disambiguation problem is related to the problem of record deduplication or record linkage that often arise when multiple tables (from different data sources) are merged to create a single table. The causes of record linkage and reference disambiguation problems are similar. The differences between the two can be viewed using as follows: while the record linkage problem consists of determining when two records are the same, reference disambiguation corresponds to ensuring that reference in a database point to the correct entities.

Because of the tight relationship between these two data cleaning tasks and similarity between their causes the existing approaches to record linkage can be applied for reference disambiguation. In particular, feature-based similarity (FBS)[5] methods which analyses the similarity of record attribute values (to determine whether two records are the same) can be used to determine whether a particular reference corresponds to a given entity or not. The quality of disambiguation can be significantly improved by exploring additional semantic information.For instance, 'S. Wards' might be used to refer to an author in the context of a particular publication.But This publication might also refer to different authors, which can be linked to other sites or organizations etc., this form chains of relationships among entities. Such knowledge can be exploited along attribute-based similarity which results in improved accuracy of disambiguation

2.2.XMLDup:

Xmldup algorithm is also used to detect the duplicates in the XMLdata. It uses Bayesian network model for detecting duplicates[3]. This network model is a directed acyclic graph. The nodes in the graph represents the random variables and the edges in the graph represents the dependencies between the nodes.

Two nodes are said to be duplicates if their values are duplicates and their children nodes are duplicates. To find the similarity between two xml trees a Bayesian network is constructed for it. Each node in the Bayesian network represents the possibility of a node is duplicated in both trees. The applications of Bayesian networks is present in [7].

For example: If two trees consist of a root node 'N1'. Bayesian network contains the node 'N11' which represents the possibility of node N1 is duplicate in both trees. A binary random variable 0 or 1 is assigned to this node. The value 1 represents this node is duplicate in both treesl. The value 0 represents that node is not duplicate.

The node N1 will have two parent nodes such as VN1 and CN1 where VN1 represents the possibility that the values of the node N1 are duplicates and the node CN1 represents the possibility of the children nodes of N1 are duplicates. These nodes are also assigned binary random variables 1 or 0 which represents that their values and children nodes are duplicates or not. In this way Bayesian network is constructed for all the nodes.

To assign the random values 0 r 1 to the nodes we need to calculate the probabilities of the nodes. For obtaining these probabilities the algorithm first finds the prior probabilities associated with leaf nodes such that it sets the probabilities of the intermediate node and hence the root node probabilities will be set. prior proababilities can be defined by using a similarity function between the values.

Conditional probabilities are also defined further. After calculating both prior and conditional probabilities Bayesian network can be used to find the final probability of two xml trees being duplicates.

2.3 Network Pruning:

As mentioned before prior probabilities are found by applying the similarity measure to the pair of values represented by the leaf nodes, computing the

Similarity measure for all the values is time consuming hence we introduce a pruning strategy which avoids calculating the prior probabilities until they are strictly needed. This strategy works as follows:

The similarity values are assumed to be 1 before comparing the two objects. Whenever a new

Similarity is computed, the final probability is found by considering already known similarities and the unknown similarities which we have assumed to be 1. When we got to know that root node probability cannot achieve a score greater than the duplicate threshold value then that object pair is discarded and hence those remaining calculations are avoided. So that the performance of the network evaluation increases.

3.RESEARCH ELABORATIONS

3.1. Xml duplicates deletion

After completing the Bayesian network evaluation by implementing the pruning technique it displays the list nodes with complete node structure. From those list we have to select any node on which we are interested to find the duplicates and to delete them. After selecting the node the delete option has to be selected to delete that particular node. It first compares that node structure with existing dataset by using string tokenizer to compare the values at each level to find whether that node structure has duplicated value. If it contains a duplicate value then that node is displayed in red colour as abnormal node. If it does not have any duplicates when compared with entire dataset then that node is displayed in blue colour. The duplicate nodes of the abnormal node are deleted from the list.

The following algorithm is used to find the duplicates and delete them. The input of the algorithm are two object type X and Y and maximum number of iterations. We get the best combination of YX as output.which is the duplicate of XY and it is deleted.

Algorithm

```
1.      for  (each object type X)
2.       n ← total number of different object types
   related to objects of type X
3.      for (each related object type Y)
4.      end for
5.      t ← a large number
6.      do
7.        for (each object type X)
8.    for (each object type Y)
9.     repeat
10.        repeat
11.         Randomly select in Neighbor (XY)
12.        Diff ← f(YX) - f('YX)
13.      If diff >0  then YX ← 'YX
14.     else generate random x in (0,1)
15.   until iteration count = max no of iteration
16.   until iteration count = max no of iteration
17.    end for
18.   end for
19.   until timeout
20.  return the best combination of YXs
21.  End
```

4.RESULT AND DISCUSSION

The concept of this paper is implemented and different results are shown below, The proposed paper is implemented in Java technology on a Pentium-IV PC with minimum 20 GB hard-disk and 1GB RAM. The propose paper's concepts shows efficient results and has been efficiently tested on different Datasets.

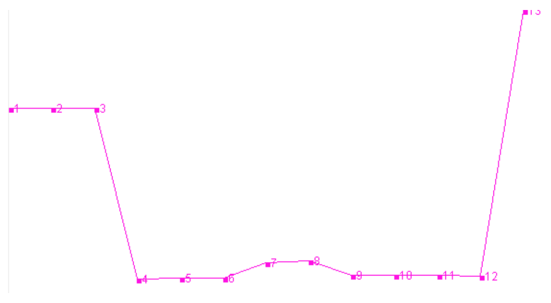

Fig. 1 detecting duplicates.

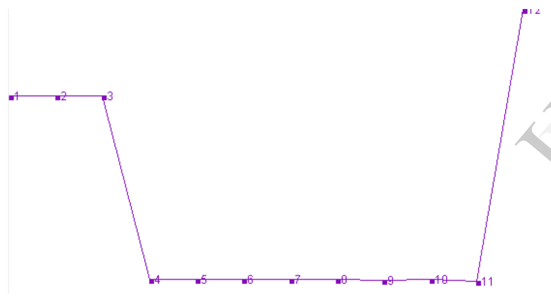

Fig. 2 Proposed system detecting duplicates at different instance

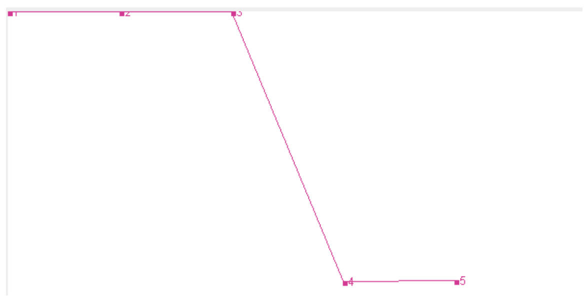

Fig. 3 detecting duplicates for different instance

5.CONCLUSION

In this paper, we presented a method for XML duplicate detection called XMLDup. Our algorithm uses a Bayesian Network to determine the probability of two XML objects being duplicates. The Bayesian Network model is composed from the structure of the objects being compared, thus all probabilities are computed considering not only the information the objects contain, but also the way such information is structured. XMLDup requires little user 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 applied in two ways. A lossless approach, with no impact on the accuracy of the final result, and a lossy approach, which slightly reduces recall.

Furthermore, the second approach can be performed automatically, without needing user intervention. Both strategies produce significant gains in efficiency over the unoptimized version of the algorithm. After finding the duplicates in the dataset we use an algorithm to delete those duplicates from the dataset.

The future work intends to extend the BN model construction algorithm to compare XML objects of different structures and apply different machine learning methods to derive the conditional probabilities and network structure, based on the data used.

REFERENCES

[1] D.V. Kalashnikov and S. Mehrotra, "Domain-Independent DataCleaning via Analysis of Entity-Relationship Graph."ACM Trans.Database Systems, vol. 31, no. 2, pp. 716-767, 2006.

[2] L. Leit˜ao, P. Calado, and M. Weis, "Structure-based inference of XML similarity for fuzzy duplicate detection," in Proceedings of the 16th ACM international conference on Information and knowledge management, 2007, pp. 293-302.

[3] Lu´?s Leit ˜ao, P´avel Calado, Melanie Herschel, "Efficient and Effective Duplicate Detection in Hierarchical data"

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

[5] Hideki Hirakawa, Zhonghui Xu, Kenneth Haase: "Inherited Feature-based Similarity Measure Based on Large Semantic Hierarchy and Large Text Corpus".

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

[7] J. Pearl, Probabilistic Reasoning in Intelligent Systems:Networks of plausible inference, 2nd ed. Morgan Kaufmann Publishers, 1988.