IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VIII (Mar-Apr. 2014), PP 101-105
www.iosrjournals.org

An Effective Solution to Adequate and Operative Duplicate Detection in Stratified Data

A. Baladhandayutham, S. Roselin Mary,
P.G Scholar, Department of Computer Science and Engineering, Anand Institute of Higher Technology, Chennai, India.
Associate Professor, Department of Computer Science and Engineering, Anand Institute of Higher Technology, Chennai, India.

Abstract: *Data Mining is considered as a nontrival extraction of implicit, previously unknown and potentially usefull information from data. Although there is a long line of work on identifying duplicates in relational data, only a few solutions focus on duplicate detection in more complex hierarchical structures, like XML data. A novel method for XML duplicate detection, called XMLDup. XMLDup uses a Bayesian network to determine the probability of two XML elements being duplicates, considering not only the information within the elements, but also the way that information is structured. In addition, to improve the efficiency of the network evaluation a novel pruning strategy, capable of significant gains of the unoptimized version of the algorithm, is presented. Through experiments Bayesian Network proves that it can achieve high precision and recall scores in several data sets. XMLDup is also able to outperform another state-of-the-art duplicate detection solution, both in terms of upto 80% and of effectiveness.*

Keywords: *Duplicate Detection, Record Linkage, XML, Bayesian Network, Data Cleaning, Data Integration, Pruning Optimization.*

I. Introduction

Electronic data play a central role in numerous business processes, applications, and decisions. As a consequence, assuring its quality is become essential. Data quality however can be compromised by many different types of errors, which can have various origins. They focus on a specific type of error, namely fuzzy duplicates or duplicates for short. Duplicates are multiple representations of the same real-world object (e.g., a person) that differ from each other because, for example one representation stores an outdated address. What makes duplicate detection a nontrivial task is the fact that duplicates are not exactly equal, often due to errors in the data. Consequently, they cannot use common comparison algorithms that detect exact duplicates. Instead, they have to compare all object representations, using a possibly complex matching strategy, to decide if they refer to the same real-world object or not. Due to its highly practical relevance in data cleaning and data integration scenarios, duplicate detection has been studied extensively for relational data stored in a single table. In this case, they detect strategy typically consists in comparing pairs of tuples (each tuple representing an object) by computing a similarity score based on their attribute values. Then, two tuples are classified as duplicates if their similarity is above a predefined threshold. However, this narrow view often neglects other available related information as, for instance, the fact that data stored in a relational table relates to data in other tables through foreign keys. The opportunity of considering such relations during pair wise comparisons has recently been realized and new algorithms have been proposed. Among these, several focus on the special case of detecting duplicates in hierarchical and semi-structured data, most notably, on XML data. Methods devised for duplicate detection in a single relation do not directly apply to 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.

Consider the two XML elements depicted as trees. Both represent person objects and are labeled prs. These elements have two attributes, namely the date of birth (dob) and name. They nest further XML elements representing place of birth (pob) and contacts (cnt). A contact consists of several addresses (add) and an email (eml), represented as children XML elements of cnt. Leaf elements have a text node which stores the actual data. For instance, dob has a text node containing the string “13-03-1972” as its value. In this example, the goal of duplicate detection is to detect that both persons are duplicates, despite the differences in the data. To do this, they compare the corresponding leaf node values of both objects. They propose that the hierarchical organization of XML data helps in detecting duplicate prs elements, since descendant elements (e.g., eml or add) can be detected to be similar, which increases the similarity of the ancestors, and so on in a top-down fashion. They first present a probabilistic duplicate detection algorithm for hierarchical data called XMLDup.

This algorithm considers both the similarity of attribute contents and the relative importance of descendant elements, with respect to the overall similarity score. The algorithm presented here extends the previous work by significantly improving the efficiency and showing a more extensive set of experiments. The contributions, especially compared to the previous work, can be summarized as follows 1) we address the issue of efficiency of the initial solution by introducing a novel pruning algorithm and studying how the order in which nodes are processed affects runtime. A major result is that XMLDup now outperforms DogmatiX, a previously more efficient state-of-the-art algorithm for XML duplicate detection 2) they describe how to increase efficiency when a slight drop in recall, i.e., in the number of identified duplicates, is acceptable. This process can be manually tuned or performed automatically; using known duplicate objects from other databases and 3) they provide a more extensive evaluation of the algorithms than in the previous work. More specifically, demonstrate the effectiveness of the algorithm on a larger number of data sets, from different domains than those used. The efficiency is extensively evaluated here. Unlike previous works, such as, where the goal is to reduce the number of pair wise comparisons performed, here and concerned with how to efficiently perform each pair wise comparison. In fact, the approach could be combined with such solutions to achieve an even for more effective result.

II. Existing Works

Data cleaning is especially required when integrating heterogeneous data sources and should be addressed together with schema-related data transformations [1]. The correctness and effectiveness of a transformation workflow and the transformation definitions should be tested and evaluated. Multiple iterations of the analysis, design and verification steps may be needed. Single source errors are removed the cleaned data should also replace the dirty data in the original sources in order to give legacy applications the improved data and to avoid redoing the cleaning work for future data extractions. Schema level and Instance level Problems are discussed for transform a data.

Identifying multiple representation [2] of a same real-world object is must be defined. Data model and detection model are two dimensions for comparing with attributes of relations. Using a top-down approach DELPHI regards not only the immediate attributes of objects, but also their children and parents in a complex warehouse schema. In an XML document every XML element is considered to represent an object, and every object can be subject to duplicate detection, in which case it is called a candidate. RECONA and ADAMA are two algorithms which use for the ascending order of r and exploit relationships between objects for the purpose of duplicate detection. Duplicate detection is the problem of determining that different representations of entities actually represent the same real-world object. The problem of XML duplicate detection is particularly tackling in applications like catalog integration or on-line data cleansing.

The duplicate elimination problem of detecting multiple tuples, which describe the same real world entity, is an important data cleaning problem [3]. It reduce the number of pair wise comparisons between tuples by pruning out many tuples that do not have any duplicates Translation tables can be significantly smaller than the database if the number of duplicates is much less than the number of tuples in the database. Hence, potential duplicate filtering enhances efficiency. The problem of detecting and eliminating duplicated data is one of the major problems in the broad area of data cleaning and data quality.

A domain-independent data cleaning approach for reference disambiguation, referred to as Relationship-based Data Cleaning (RELDC), which systematically [4] exploits not only features but also relationships among entities for the purpose of disambiguation. The generic approach therefore first discovers connections between the entity, in the context of which the reference appears, and the matching candidates for that reference. RELDC will exploit relationships for further disambiguation and will output a resolved graph G in which each entity is fully resolved.

Object identification has received much attention in the relational world, and first exists in identifying duplicate objects in hierarchical and XML data [5]. The similarity measure, is both symmetrical and takes into account not only similarity but also differences of the compared elements. DogmatiX is rendered domain-independent in its description selection by using specialized heuristics. The only remaining part of the framework that relies on expert input is the candidate selection. The XML query formulation component takes as input the set of XPaths and returns an XQuery the result of which is the description of a candidate duplicate as XML. After the candidate duplicate specification and duplicate definition phases, perform candidate query formulation and execution.

Duplicate detection is the process of discovering the same real world object with different representations, in one or more databases [6]. Duplicate detection in XML, in some aspects, is similar to duplicate detection in relational databases. In both cases, one has to deal with strings of text, which must be compared in order to check if they carry the same content or not. Two records from two different databases can store the same information, but their fields can be organized in different ways. Neumann and Herschel proposed duplicate detection in a single relation, tree data, and graph data. They mainly find works focusing on the XML data model. The only exception is Eliminating Fuzzy Duplicates in Data Warehouses, which focuses on hierarchical tables in a data warehouse. In previous XML duplicate detection was mostly concerned with the efficient implementation of XML join operations. Guha, who suggested an algorithm to perform approximate, joins in XML databases. However, their main concern was on how to efficiently join two sets of similar elements, and not on how accurate the joining process was.

III. PROPOSED MODEL

The goal of duplicate detection is to detect that both persons are duplicates, despite the differences in the data. The corresponding leaf node values of both objects must be compared and the proposed hierarchical organization of XML data helps in detecting duplicate prs elements, since descendant elements (e.g., eml or add) can be detected to be similar, which increases the similarity of the ancestors, and so on in a top-down fashion. It is first presented a probabilistic duplicate detection algorithm for hierarchical data called XMLDup. Bayesian Network is unable to detect a duplicate detection in a large size of xml data. To overcome this problem, we implement the proposed Priority algorithm for detect duplicate data's in large XML data. The algorithm considers both the similarity of attribute contents and the relative importance of descendant elements, with respect to the overall similarity score.

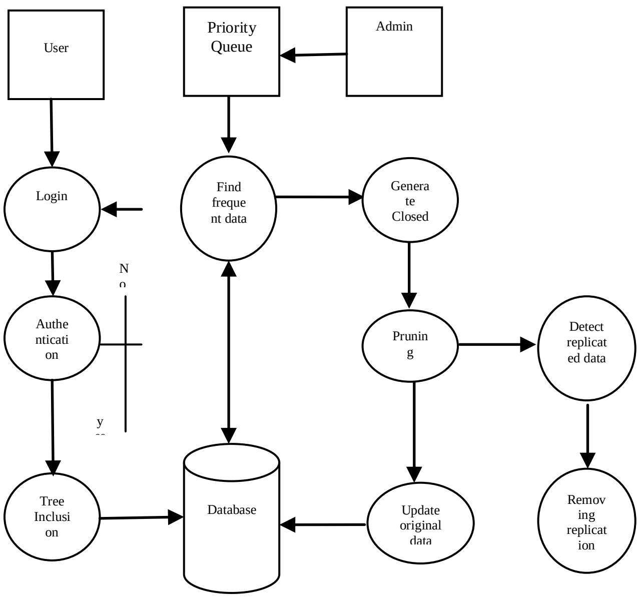

Fig 1.Overall Design Model

i) Algorithm

Network Pruning (N,T)

The node N, for which we intend to compute the probability score; threshold value T, below which the XML nodes are considered non-duplicates.

Duplicate probability of the XML nodes represented by N.

```
1: L ← getParentNodes (N){Get the ordered list of parents}
2: parentScore[n] ← 1, ∀n ∈ L {Maximum probability of each
   parent node}
3: currentScore ← 0
4: for each node n in L do{Compute the duplicate probability}
5: if n is a value node then
6: score ← getSimilarityScore (n) {For value nodes, compute
   the similarities}
7: else
8: new Threshold ← getNewThreshold(T, parentScore)
9: score ← NetworkPruning (n, new Threshold)
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
```

Above code explains the proposed algorithm of Priority Bayesian Network.

ii) Process of Proposed Model

In this paper, we survey the state of the art for duplicate detection in hierarchical data, which is the focus of this paper. For a more complete discussion of related work, we
refer readers to the book by Naumman and Herschel [2], which includes duplicate detection in a single relation, tree data, and graph data.

Among studies that deal with hierarchical data, we mainly find works focusing on the XML data model. The only exception is [3], which focuses on hierarchical tables in a data warehouse. Early work in XML duplicate detection was mostly concerned with the efficient implementation of XML join operations. A pioneering approach was presented.

a. Tree Inclusion Module

In Tree Inclusion Module the User have to include the Tree structured data. Use tree mining concept to determine the frequent patterns tree. This means determining if a pattern tree is included in any tree of the data.

b. Bayesian Network Construction

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

c. Frequent Trees Finding Module

The Frequent Trees Finding Module contains searching process in Tree Structured data. It means finding frequent attribute trees in a tree database. Here the User Using kpruning probability propagation algorithm concept to find out the frequent tree patterns in the Tree structured data. To keep track of all frequent occurrences and will use the occurrence support for processing.

d. Generate Closed Frequent Tree

In Closed Frequent Tree generation module the frequent trees contain redundant Information find all the closed frequent attribute trees for a given data tree. The number of closed frequent attribute trees is much smaller than the number of all frequent attribute trees. Finally the frequent attributes trees can be easily deduced from the closed frequent attribute trees.

e. Pruning Module

In Pruning module 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.

V. Conclusion

In this paper, we presented a novel 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. How-ever, 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 for 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. Experiments performed on both artificial and real-world data showed that our algorithm is able to achieve high precision and recall scores in several contexts. When compared to another state-of-the-art XML duplicate detection algorithm, XMLDup consistently showed better results regarding both efficiency and effectiveness.

The success demonstrated in experimental results leaves motivation for future work. Among other tasks we intend 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.

References

[1] E. Rahm and H.H. Do, “Data Cleaning: Problems and Current 2000. Approaches,” IEEE Data Eng. Bull., vol.
23, no. 4, pp.
3-13, Dec.

[2] Relationship-Based Duplicate Detection “Melanie Weis and Felix Naumann”Natural Language Processing and Knowledge Engineering (NLP-KE), 2010 International Conference.

[3] Eliminating Fuzzy Duplicates in Data Warehouses “Rohit Ananthakrishna, Surajit Chaudhuri Venkatesh Ganti”,Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[4] Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph” DMITRI V. KALASHNIKOV and SHARAD MEHROTRA”, ACM Transactions on Database Systems, Vol. 31, No. 2, 2006.

[5] DogmatiX Tracks down Duplicates in XML “Melanie Weis, Felix Neumann”, SIGMOD '05 Proceedings of the 2005 ACM SIGMOD international conference on Management of data.

[6] Matching XML Documents in Highly Dynamic Applications”Adrovane Marques Kade and Carlos Alberto Heuser”,DocEng '08 Proceedings of the eighth ACM symposium on Document engineering on 2008.