XML Multidup: A Probabilistic Duplicate Detection Method for Hierarchical Multimedia Data

Mrs.K.Valarmathi[1] K.Priyangaa[2]
[1]Assistant Professor & Head
[1,2]Department of Computer Science
[1,2]Trinity College for Women, Namakkal

Abstract— Even though present is a long line of work on identify duplicate in relational data, only a few solutions focus on duplicate detection in more composite hierarchical structures, like XML data. In this article, we there a novel method for XML replacement detection, called XML Dup. XML Dup uses a Bayesian network to decide the probability of two XML elements being duplicate, allowing for not only the information surrounded by the elements, but also the way that information is structured. In addition, to get better the good organization of the network evaluation, a novel pruning policy, capable of major gains over the un optimized version of the algorithm, is presented. Through experiment, we show that our algorithm is able to accomplish high exactitude and remember scores in quite a lot of datasets. XMLDup is also able to do better than another state of the art replacement detection solution, both in terms of competence and of usefulness.

Key words: XMLDup, Bayesian Network, Duplicate

I. INTRODUCTION

Outstanding to its extremely practical application in data clear out and data mixing scenarios, replacement detection has been premeditated extensively for relational data stored in a single fig.3. In this case, the detection strategy typically consists in comparing pairs of tuples 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 pairwise 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 [5]. 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 example that we will use throughout the article.

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 by Guha et al. [11], 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. Thus, they focused on an efficient implementation of a tree edit distance, which could later be applied in an XML join algorithm. The concern with accuracy was later approached by Carvalho and Silva, in [12]. Although not specifically focused on XML, their work proposes a solution to the problem of integrating tree-structured data extracted from the Web. Two object representations, e.g., two hierarchical representations of person elements, are compared by transforming each into a vector of terms and using a variation of the cosine measure to evaluate their similarity [13]. The hierarchical structure of object representations is mostly ignored, and a linear combination of weighted similarities is used to account for the relative importance of the different fields within the vectors. The authors show that this simple strategy manages to achieve high precision values in a collection of scientific publications. Nevertheless, and because of its more general nature, their approach does not take advantage of the useful features existing in XML databases, such as the element structure or tag semantics. Only more recently has research been performed with the specific goal of discovering duplicate object representations in XML databases.

These works differ from previous approaches since they were specifically designed to exploit the distinctive characteristics of XML object representations: their structure, textual content, and the semantics implicit in the XML labels. We briefly describe the main features of these methods here, and refer readers to [9] for a detailed theoretical and experimental comparison of these approaches. The DogmatiX framework aims at both efficiency and effectiveness in duplicate detection [5] . The framework consists of three main steps: candidate definition, duplicate definition, and duplicate detection. Whereas the first two provide the definitions necessary for duplicate detection (i.e., the set of object representations to compare and the duplicate classifier to use), the third component includes the actual algorithm, an extension to XML data of the work of Ananthakrishna et al. [3].

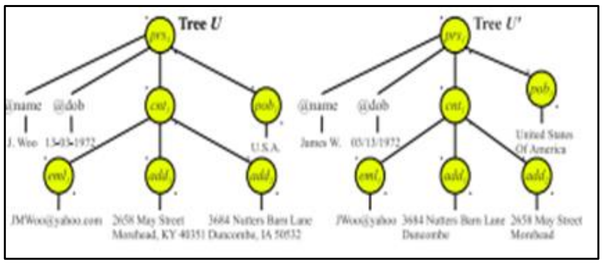

Fig. 1: Two XML elements that represent the same person.

II. RELATED WORKS

In this section, 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 by Guha et al. [11], 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. Thus, they focused on an efficient implementation of a tree edit distance, which could later be applied in an XML join algorithm.

The concern with accuracy was later approached by Carvalho and Silva, in [12]. Although not specifically Focused on XML, their work proposes a solution to the problem of integrating tree-structured data extracted from the Web. Two object representations, e.g., two hierarchical representations of person elements, are compared by transforming each into a vector of terms and using a variation of the cosine measure to evaluate their similarity[13]. The hierarchical structure of object representations is mostly ignored, and a linear combination of weighted similarities is used to account for the relative importance of the different fields within the vectors. The authors show that this simple strategy manages to achieve high precision values in a collection of scientific publications. Nevertheless, and because of its more general nature, their approach does not take advantage of the useful features existing in XML databases, such as the element structure or tag semantics. Only more recently has research been performed with the specific goal of discovering duplicate object representations in XML databases. These works differ from previous approaches since they were specifically designed to exploit the distinctive characteristics of XML object representations: their structure, textual content, and the semantics implicit in the XML labels.

We briefly describe the main features of these methods here, and refer readers to [9] for a detailed theoretical and experimental comparison of these approaches. The Dogmati X framework aims at both efficiency and effectiveness in duplicate detection The framework consists of three main steps: candidate definition, duplicate definition, and duplicate detection. Whereas the first two provide the definitions necessary for duplicate detection (i.e., the set of object representations to compare and the duplicate classifier to use), the third component includes the actual algorithm, an extension to XML data of the work of Ananthakrishna et al. [3]. The XML Dup system first proposed in [6] uses a Bayesian Network model for XML duplicate detection. Its approach is the basis for the algorithms proposed in this paper, and is further described in Section 3. Milano et al. propose a distance measure between two XML object representations that is defined based on the concept of overlays [8]. An overlay between two XML. trees U and V is a mapping between their nodes, such that a node u ∈ U, is mapped to a single node v ∈ V if, and only if, they have the same path from the root.

This measure is then used to perform a pair wise 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. Finally, SXNM (Sorted XML Neighborhood Method) [10] is a duplicate detection method that adapts the relational sorted neighborhood approach (SNM) [14] to XML data. Like the original SNM, the idea is to avoid performing useless comparisons between objects by grouping together those that are more likely to be similar.

III. METHODOLOGY

A. *Bayesian Network Constructions*

Bayesian Network provide a concise specification of Joint probability distribution. They can be seen as a going to acyclic graph, where the nodes stand for random variables and the edges represent dependencies between those variables. We first summarize how the Bayesian Network for XML duplicate detection is constructing. Afterwards, we give particulars how odds are compute in arrange to make a decision if two objects are in fact duplicate. For a more detailed clarification of Bayesian Networks and their application, the reader is referred.

B. *BN Duplicate Detection*

Our presents for XML duplicate detection is centered on one basic supposition: The fact that two XML nodes are duplicates depends only on the fact that their standards are duplicates and that their children nodes are duplicates. Thus, we say so as to two XML trees are duplicating if their root nodes are duplicate. This earnings that the two person objects, represent by nodes tagged, are duplicate depending on whether or not their children nodes and their values for attributes name and DOB are duplicates. Furthermore, the nodes tagged are duplicates depending on whether or not their values are duplicates, and the nodes tagged are duplicates depending on whether or not their children nodes are duplicates. This procedure goes on recursively until the leaf nodes are reached.

Let us first consider the XML nodes tagged prs. As illustrate in Figure 2, the BN will have a node labeled prs11 on behalf of the possibility of node prs1 in the XML tree U being a replacement of node prs1 in the XML tree U 0. Node prs11 is assigned a binary chance variable. This variable takes the value 1 (active) to symbolize the fact that the XML nodes in trees U and U 0 are duplicate. It takes the value 0 (inactive) to characterize the fact that the nodes are not duplicates.

In agreement with our belief, the prospect of the two XML nodes being duplicates depends on (i) whether or not their ethics are duplicates, and (ii) whether or not their children are duplicates. Thus, node prs11 in the BN has two parent nodes, as shown in Figure 2. Node Vprs11 represents the possibility of the values in the nodes being duplicates. Node Cprs11 represent the possibility of the children of the prs nodes being duplicate. As before, a binary random variable, that can be active or inactive, is assigned to these nodes, on behalf of the fact that the values and brood nodes are duplicate or non duplicates, respectively.

C. Compute the Probabilities

As we have seen, we allocate a binary random variable to each node, which take the value 1 to represent the fact that the corresponding data in trees U and U 0 are duplicate, and the value 0 to represent the opposite. Thus, to decide if two XML trees are duplicates, the algorithm has to calculate the probability of the root nodes being duplicates. In our example, this corresponds to computing, which can be interpreted as a connection value between the two XML. Computes. To obtain this probability, the algorithm spread the prior probabilities associated to the BN leaf nodes, which will set the middle node probabilities, until the root likelihood is found.

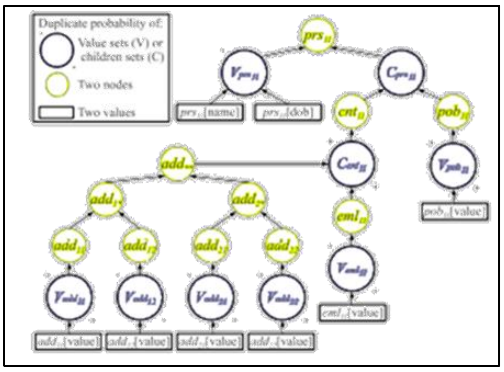

Fig. 2: BN to compute the similarity of the trees in

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 0. This is represented by nodes add⇤⇤, add1⇤, and add2⇤ in the BN of Figure 2.

IV. Evaluation

In this division we there an assessment of the XML Dup algorithm describe in the previous sections. We assess the algorithm both in terms of effectiveness and competence. First, we evaluate effectiveness by compare it to a state of the art replacement detection organization, call DogmatiX, that prove to be the most spirited so far. We then assess the efficiency of XML Dup when using our proposed pruning optimization, node ordering heuristics, varying the pruning feature, and routinely selecting the nearly all adequate pruning factors. The experiment is concluded with a discussion of the results.

Testing the impact of data superiority on duplicate detection is important to substantiate the effectiveness of a given algorithm.

In preceding work we have shown that XML Dup manage to cope with errors like typos or duplicate incorrect elements without any significant dilapidation of the results and even performs effectively when commerce with reasonable amounts of lost data. Due to space constraints, that experiment

A. Datasets

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 err such as typographical errors, missing data, and duplicate erroneous data [6].

1) Experimental Setup

Experiments were performed to compare the effectiveness and efficiency of the tested algorithms. To assess effectiveness, we applied the commonly used precision, recall, and r-precision measures. Precision measures the percentage of correctly identified duplicates, over the total set of objects determined as duplicates by the Recall measures the percentage of duplicates correctly identified by the system, over the total set of duplicate objects system.

Dataset	Average Precision		R-Precision		Maximum Recall	
	XMLDup	XMLMulDup	XMLDup	XMLMulDup	XMLDup	XMLMulDup
CD 2	96	99	82	82	99	99
Cora	87	98	75	75	80	80
MultiCD2	68	99	64	82	71	99
MultiCora	52	98	56	75	68	80

Table 1: Performance Achieved Using Proposed Method on Real and Artificial Dataset

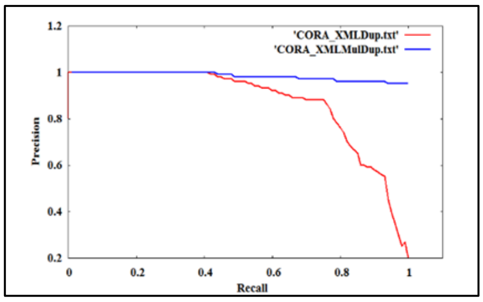

Fig. 3: Recall values for Multi CORA Datasets.

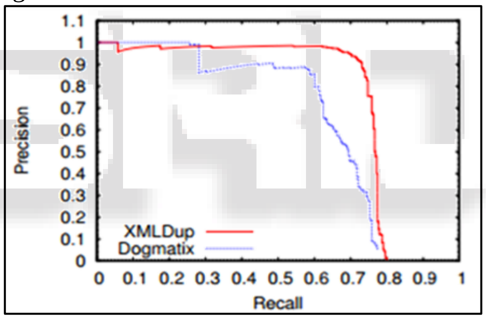

Fig. 4: Recall value for CD2 datasets

V. Conclusions

This algorithm uses a Probabilistic Duplicate Detection Method to establish the probability of two XML objects living being duplicates. A Probabilistic Duplicate Detection Method model is composed from the structure of the objects being compared, thus all probabilities are compute considering not only the information the substance contain, but also the way such information is structured. XML Dup require little user interference, since the consumer only needs to provide the attribute to be considered, their respective default prospect parameter, and a comparison threshold. However, the model is also very flexible; allow the use of different resemblance measures and different ways of combine probabilities.

To get better the runtime competence of XMLDup, a network prune strategy is also presented. This strategy can be practical in two ways. A lossless move toward, with no impact on the accuracy of the final result, and a lossy move toward, which slightly reduces recall. in addition, the second approach can be performed automatically, without needing user intervention. Both strategies make important gains in competence over the unoptimized version of the algorithm. Experiments perform on both mock and real world data show that our algorithm is able to achieve high precision and bring to mind scores in a number of contexts. As soon as compare to another state-of-the-art XML duplicate detection algorithm, XML Dup consistently showed improved results regarding both competence and effectiveness.

The success confirmed in untried results leaves enthusiasm for future work. Surrounded by other tasks we plan to lengthen the BN model manufacture algorithm to contrast XML objects with different structure and apply machine knowledge methods to get the restrictive probabilities and network structure, based on the presented data.

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] F. Naumann and M. Herschel, An Introduction to Duplicate Detection. Morgan and Claypool, 2010.
[3] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating fuzzy duplicates in data warehouses," in Conference on Very Large Databases (VLDB), Hong Kong, China, 2002, pp. 586–597.
[4] D. V. Kalashnikov and S. Mehrotra, "Domain-independent data cleaning via analysis of entity-relationship graph." ACM Transactions on Database Systems (TODS), vol. 31, no. 2, pp. 716–767, 2006.
[5] M. Weis and F. Naumann, "Dogmatix tracks down duplicates
[6] in XML," in Conference on the Management of Data (SIGMOD), Baltimore, MD, 2005, pp. 431–442.
[7] L. Leitao, 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.
[8] A. M. Kade and C. A. Heuser, "Matching XML documents in highly dynamic applications," in ACM Symposium on Document Engineering (DocEng), 2008, pp. 191–198.
[9] D. Milano, M. Scannapieco, and T. Catarci, "Structure aware XML object identification," in VLDB Workshop on Clean Databases (CleanDB), Seoul, Korea, 2006.
[10] P. Calado, M. Herschel, and L. Leitao, "An overview of XML ˜ duplicate detection algorithms," Soft Computing in XML Data Management, Studies in Fuzziness and Soft Computing, vol. 255, 2010.
[11] S. Puhlmann, M. Weis, and F. Naumann, "XML duplicate detection using sorted neigborhoods," in Conference on Extending Database Technology (EDBT), Munich, Germany, 2006, pp. 773–791.
[12] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava, and T. Yu, "Approximate XML joins," in Conference on the Management of Data (SIGMOD), 2002.
[13] J. C. P. Carvalho and A. S. da Silva, "Finding similar identities among objects from multiple web sources," in CIKM Workshop on Web Information and Data Management (WIDM), New Orleans, Louisiana, USA, 2003, pp. 90–93.
[14] R. A. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.
[15] M. A. Hernandez and S. J. Stolfo, "The merge/purge problem or large databases," in Conference on the Management of Data (SIGMOD), San Jose, CA, 1995, pp. 127–138.
[16] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of plausible inference, 2nd ed. Morgan Kaufmann Publishers, 1988.
[17] L. Leitao and P. Calado, "Duplicate detection through structure ˜ optimization," in Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, pp. 443–452.
[18] E. H. Simpson, "Measurement of diversity," Nature, vol. 163, p. 688, 1949.
[19] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," Advances in Neural Information Processing Systems (NIPS), vol. 9, pp. 155–161, 1996.
[20] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671–680, 1983.
[21] T. Joachims, Making large-scale support vector machine learning practical. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184.
[22] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma, "Object-level ranking: bringing order to web objects," in International conference on World Wide Web (WWW), 2005, pp. 567–574.
[23] L. Chen, L. Zhang, F. Jing, K.-F. Deng, and W.-Y. Ma, "Ranking web objects from multiple communities," in Proceedings of the 15th ACM international conference on Information and knowledge management, 2006, pp. 377–386.