Detection of Duplicate Objects in Semi Structured Data like XML

Kiran Lokhande
Department of Information Technology
Pune Institute of Computer Technology
Pune, India
e-mail: kiranlokhande199@gmail.com

Tushar Rane
Department of Information Technology
Pune Institute of Computer Technology
Pune, India
e-mail: ranetushar@yahoo.com

Dr. S. T. Patil
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune, India

***Abstract*— Duplicate detection is the process of finding the duplicate objects in the data. This is the important part of data cleansing step of data mining. Duplication occurs when some real world object has multiple representations in data source. Significant amount of work has been done in duplicate detection of relational data, but only recently the researchers have shifted their focus towards duplicate detection in hierarchical and semi-structured data e.g. XML. A novel method called XMLDup uses Bayesian network to effectively detect the duplicates in XML. However this method does not consider the XML objects with different structure. In this work, we will propose a modified version of XMLDup so that it can also detect the objects with structural differences.**

Keywords-Data cleansing, duplicate detection, hierarchical data, XML, Bayesian networks.

I. INTRODUCTION

Data cleansing is the process of detecting and removing errors and inconsistencies from data in order to improve the quality of data [1]. These errors include typographical errors, missing values, duplicate objects etc. Duplicates are multiple representations of same real world object [2]. Duplicate detection is the process of finding such a duplicate data. Duplicate detection is not just about simple comparisons between two objects to decide whether they represent the same real world object or not, but this is a very complex process, because the two objects being compared are not exactly same.

Considerable amount of work has been done on duplicate detection in relational databases. But this work cannot be applied to hierarchical and semi-structured data such as XML. Duplicate detection in XML is more difficult than that in relational data because there is no schematic distinction between object types among which duplicates are detected and attribute types describing objects [3]. Also this task becomes more challenging due to flexible format of XML [4].

Consider the examples shown in Fig. 1. In this, two XML elements are shown. These two XML elements represent the person object from the real world. These two XML elements will be considered as a two different elements even though they represent the same real world object. This is due to the typographical errors as well as structural differences between them. For example the address in first element is divided in two parts addr1 and addr2 while in second element, address and email id are given under the element contact. In first element, date of birth is 18-08-1991 while in second element, it is 18/08//1991. Similarly class of the student is represented as Master of Engg in first element while in second element acronym is used.

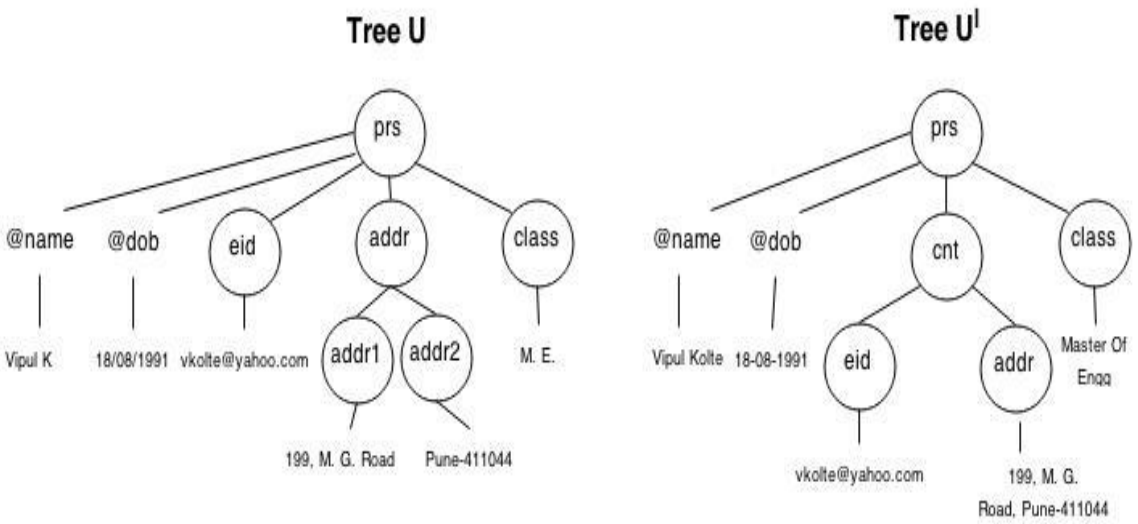

Figure 1: Example of duplication in XML

II. RELATED WORK

Most of the earlier approaches [5]-[10] of duplicate detection in XML were based on performing join operations in XML. These approaches used tree edit distance to perform join operation. Tree edit distance is defined as the minimum number of tree edit operations require to transform one tree into another which includes node insert, node delete and node rename operations [11].

In [12], R Ananthakrishna, S Chaudhuri and V Ganti have proposed a framework which eliminates the duplicates in dimensional tables. Dimensional tables are represented hierarchically. In [13], M. Weis and F. Naumann proposed a framework called DogmatiX. It consists of three steps: *candidate definition, duplicate definition and duplicate detection.* In [14], S. Puhlmann, M. Weis, and F. Naumann, have proposed a method called Sorted XML Neighbourhood Method (SXNM). It is based on the Sorted Neighbourhood Method [15] in relational database.

In [3], M. Weis, L. Leitao and P. Calado have proposed a method for duplicate detection in XML, called as XMLDup. This method uses Bayesian network for duplicate detection. Authors have extended their work in [2]. In this they have proposed a pruning algorithm which reduces the number of comparisons in XMLDup.

III. PROPOSED WORK

Our proposed work is based on the XMLDup. The problem with XMLDup is that it does not work with XML elements with different structures e.g. elements in Fig. 1. XMLDup assumes that the schema matching step has preceded the duplicate detection procedure. We have modified the XMLDup so that it can detect the duplicates even when there are structural differences in XML elements. Fig. 2 shows the system architecture for our proposed work which is same as the XMLDup. Only thing we have modified is the Bayesian network construction.

This method starts with construction of Bayesian network for given elements to be compared. It uses this Bayesian network to find the probability of child nodes and descendants being duplicates for a given pair of XML elements to be compared. This process is repeatedly in bottom up fashion till the probability of root nodes being duplicate is found. This probability indicates the similarity between two XML elements.

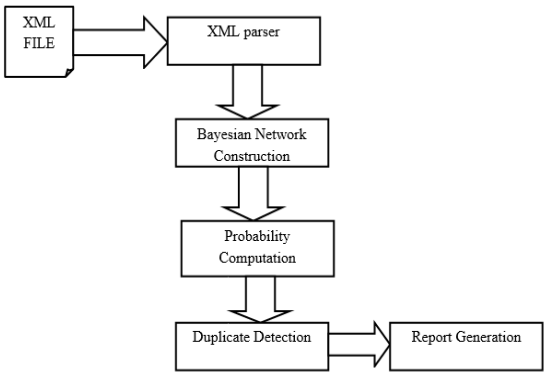

Figure 2: System Architecture

A. *Bayesian Network Construction*

XMLDup is based on the assumption that *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.* So two XML elements are duplicates if their root nodes are duplicates and these root nodes are duplicates if their children and values are duplicates.

Consider the two XML trees of Fig. 1. These two are duplicates if the root node prs in both the trees are duplicates. We will use this example to show how Bayesian network will be constructed in our proposed work. Fig 3 shows the Bayesian Network (BN) for the two XML trees from Fig 1. We will start with root node. We will add the root node prs11 in BN. This node represent the probability of node 'prs' from tree U and

prs from tree U| being duplicates. As stated earlier, XMLDup assumes that two XML nodes are duplicate if their values are duplicate and their children are duplicate. So the node prs11 in BN has two parent nodes Vprs11 and Cprs11. Node Vprs11 represents the possibility of the values in the prs nodes being duplicates. Node Cprs11 represents the possibility of the children of the prs nodes being duplicates. Similarly node Vprs11 has two parent nodes prs11[name] and prs11[dob] which are represented by rectangle. These two nodes represent the probability of attributes name and date of birth (dob) being duplicate in two XML trees of Fig. 1.

Cprs11 will have parent node $class_{11}$ which represents the probability of node 'class' from both the trees being duplicate. Up till now the procedure we explained is same as the original XMLDup. Because there was no structural difference. Now the children nodes of node prs i.e. email and addr from tree U have no matching nodes in the children nodes of node prs from tree U|. And original approach will fail in this case. In our approach we try to find the matching nodes for email and addr from tree U in tree U|, and they proceed with rest of the Bayesian network construction. This process is repeated in recursive manner. Fig 3 shows the BN for our example in Fig. 1.

B. *Probability Calculation*

Once the BN is constructed, we start with calculating the probabilities of leaf nodes being duplicates. Leaf nodes are nothing but the values. For computing the probabilities for values, we use token based similarity metrics algorithms such as Monge-Elkan. This algorithm divides the two strings to be compared into tokens and then compares these tokens to find the similarity between the two strings. This approach makes our duplicate detection process independent of whether there are multiple nodes of same type in the given XML tree.

Once the probabilities for values are calculated, these probabilities are propagated in upward direction in BN until the probability of the root nodes being duplicate is calculated.

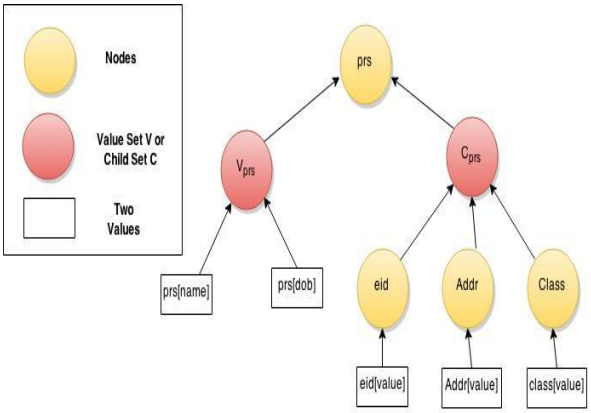

Figure 3: BN for example in Fig 1.

IV. RESULTS

We have tested our system on some sample XML elements which had structural differences and it was able to detect the duplicates. We tested the original XMLDup on same samples but it failed to detect the duplicates. So we can say that our modified version of XMLDup detects duplicates even when there are structural differences. We will be testing our system on some real-world data sets. Some of these data sets will have structural differences while others will have schema matching step already performed on them. We expect our system to effectively detect the duplicates in both the cases.

V. CONCLUSION

In this paper, we have proposed modified version of XMLDup for duplicate detection in XML. Our proposed method will detect the duplicates even when there are structural differences in XML elements. Focus of the proposed work is to effectively detect duplicates. We have not focused on efficiency part. So in future, we will like to extend our work to have efficient as well as effective duplicate detection method.

REFERENCES

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

[2] L. Leita˜o, P. Calado, and M. Weis. "Efficient and Effective Duplicate Detection in Hierarchical Data." IEEE Transactions on Knowledge and Data Engineering, Vol. 25, May 2013.

[3] L. Leita˜o, P. Calado, and M. Weis. "Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection." in Proc. 16th ACM Int'l Conf. Information and Knowledge Management, pp. 293-302, 2007.

[4] A.M. Kade, C.A. Heuser. "Matching XML Documents in Highly Dynamic Applications." in Proc. ACM Symp. Document Eng. (DocEng), pp. 191-198, 2008.

[5] P. N. Klein, "Computing the edit-distance between unrooted ordered trees," in Proceedings of the 6th European Symposium on Algorithms, ser. Lecture Notes in Computer Science, vol. 1461. Venice, Italy:Springer, 1998, pp. 91-102.

[6] W. Chen, "New algorithm for ordered tree-to-tree correction problem," Journal of Algorithms, vol. 40, no. 2, pp. 135-158, Aug. 2001.

[7] E. D. Demaine, S. Mozes, B. Rossman, and 0. Weimann, "An optimal decomposition algorithm for tree edit distance," in Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP 2007), Wroclaw, Poland, 2007.

[8] K. Zhang, R. Statman, and D. Shasha, "On the editing distance between unordered labeled trees," Information Processing Letters, vol. 42, no. 3, pp. 133-139, 1992.

[9] S. Guha, H.V. Jagadish, N. Koudas, D. Srivastava, and T. Yu. "Approximate XML Joins." in Proc. ACM SIGMOD Conf. Management of Data, 2002.

[10] J.C.P. Carvalho and A.S. da Silva. "Finding Similar Identities among Objects from Multiple Web Sources." in Proc. CIKM Workshop Web Information and Data Management (WIDM), pp. 90-93, 2003.

[11] K.-C. Tai, "The tree-to-tree correction problem," Journal of the ACM (JACM), vol. 26, no. 3, pp. 422-433, July 1979.

[12] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. "Eliminating Fuzzy Duplicates in Data Warehouses." in Proc. Conf. Very Large Databases (VLDB), pp. 586-597, 2002.

[13] M. Weis and F. Naumann. “Dogmatix Tracks Down Duplicates in XML.” in Proc. ACM SIGMOD Conf. Management of Data, pp. 431-442, 2005.

[14] S. Puhlmann, M. Weis, and F. Naumann. “XML Duplicate Detection Using Sorted Neighborhoods.” in Proc. Conf. Extending Database Technology (EDBT), pp. 773-791, 2006.

[15] M.A. Herna´ndez and S.J. Stolfo. “The Merge/Purge Problem for Large Databases.” in Proc. ACM SIGMOD Conf. Management of Data, pp. 127-138, 1995.