SIMILARITY DETECTION FOR XML DATA

Ashu Narendra Mehta

Computer Engineering,

Dr. Babasaheb Ambedkar Technological University- Lonere, (India)

ABSTRACT

Similarity Detection is critical task of any database or datasets to any organization. Finding similarity means to find out the same real time objects that are presented in different from the structure and the formats. In past there is lots of work already presented in the past for finding out the similar data in the relational data. But nowadays there is more focus on finding similarity in the XML data. Because of XML is very popular for data storing and extensively used for data exchange between the organizations. In this paper, we have done an extensive literature survey on this topic and proposed a project work by combining the existing paper's ideas and some of our original ideas. We present the unique method which gives similarity detection for XML data. Here we use Bayesian network to find out the probability of two XML elements are similar with each other. In addition to improve the efficiency and effectiveness, we also checks for its typographical errors when comparing the two XML elements. To test the correctness of our method, we are comparing it with existing system. While comparing the results we are giving more focus on how to get higher precision and recall values in the various datasets we are used.

Keywords: Bayesian Network, Data Similarity, Data Mining XML, KDD

I. INTRODUCTION

Similarity detection is subtask of Data mining where data mining is the process of extracting data from large amount of database. It is same as like of finding out of the gold from sands and rocks. Data mining helps to find out hidden knowledge, the new rules and expected or unexpected patterns from a large database. Data mining is related to data analysis and also used for software techniques which are helpful for finding patterns in database.

Finding similarity between two elements is part of data cleaning where data cleaning is the first step of Knowledge Data Discovery (KDD). KDD is used for the finding out of the data that is used for gaining knowledge for any system. So our ultimate goal is to find out similar representation of same real world object in the datasets.

XML (eXtensible Markup Language) is a markup language much like HTML (Hypertext Markup Language). It is used as designed to carry data, not to display data as HTML. XML is designed to be self-descriptive. XML is W3C Recommendation (World Wide Web Consortium). XML is a software and hardware independent language. It has no predefined tags. Its best advantage is that it allows a user to create his/her own document structure. It is increasingly popular in recent days especially for data published on the web and used for data exchange between organizations.

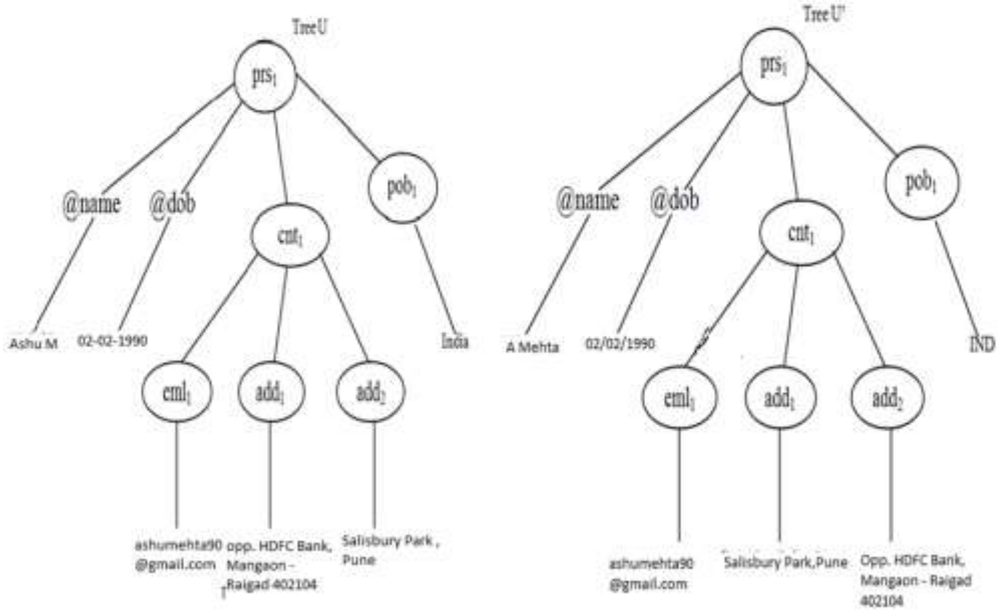

Figure 1 : Two XML elements that represent the same person

Let us consider the two XML elements shown in tree format that represent the same persons which are shown as trees in Fig 1. Here both represent object person named as ‘prs’. These elements have two attributes that are name and dob (date of birth) and also two child nodes as pob (place of birth) and cnt (contact). Again at next level contact contains add1 (correspondence address), add2 (permanent address) and eml1 (email). Here leaf elements have a text node which contains the data. For example, name has a text node which consists of a text as a “Ashu M” as its value. In this example the ultimate goal of our method is to detect the both XML elements as similar with respect to their minor differences data values. To carry out this we compare first value of attributes of both tree with each other i.e. name and dob and then check for its children node i.e pob and cnt. Furthermore, pob node are similar depending on whether or not their values are similar, and the cnt node are similar depending on whether or not their child nodes i.e eml and add are similar. This process carried out until the leaf nodes are reached.

1.1 Structure

This paper is organized as follows: Section 2 presents Releated Work. Section 3 summaries our Proposed Work which consist of our base algorithm which presented in [1][2] and [4]. We will evaluate our methods and compare with existing work in section 4 as Experimental Evaluation and Results. Finally, we conclude our work in section 5.

II. RELEATED WORK

There is lots of work was already performed for similarity detection for relational data, but we cannot directly use those methods to find out similarity in hierarchical and complex data (XML) [2]. But there are several authors who focused on similar data detection for XML data on the basis of the idea of finding similar data in relational data [3][4][5][6]. In this section, we summarize various recent methods that are introduced till date for similarity detection in hierarchical XML data [1][2][3][4][6][8].

The XMLDup system [2] was first given in [4] which use a Bayesian Network (BN) model which calculates the joint probability of two XML elements. They can be seen as DAG (Directed Acyclic Graph) which allows the relationship between nodes of the XML tree that gives random variable and their edges gives the dependencies between the variable. This approach is used as the basis for our algorithm proposed in this paper in section 3.

The DogmatiX framework gives efficiency and effectiveness in similarity detection [3]. The framework consists of three parts : 1) Candidate Definition 2) Duplicate Definition 3) Duplicate Detection. First two parts are carried out in offline mode while system setup and last part is carried out in online mode where actual algorithm works. Candidate definition allows choosing of objects that are checked for similarity detection.

Duplicate definition allows choosing of which portion of actual data is used for selection to find out similarity detection. Duplicate detection performs six various sub-steps on actual data to detect similarity in duplicate candidates that are an extension to XML data of the work of R. Ananthakrishna et al. [10].

The XMLDup system was first introduced in [4] which use a Bayesian Network (BN) model for data duplication in XML data. This approach is basis for refined version XMLDup which was stated in [2]. M. Weis et al.[7], proposed a similarity detection method called as SXNM (Sorted XML Neighbourhood Method). SXNM gives the relational sorted neighborhood approach (SNM) [9] to XMLL data. Here comparison was performed on a grouping of similar objects with each other.

M. Scannapieco et al. proposed a distance measure between two XML object representations that are depends on the concept of overlays [6]. If the distance measure crosses the certain threshold value then the given pair will call as similar.

III. PROPOSED WORK

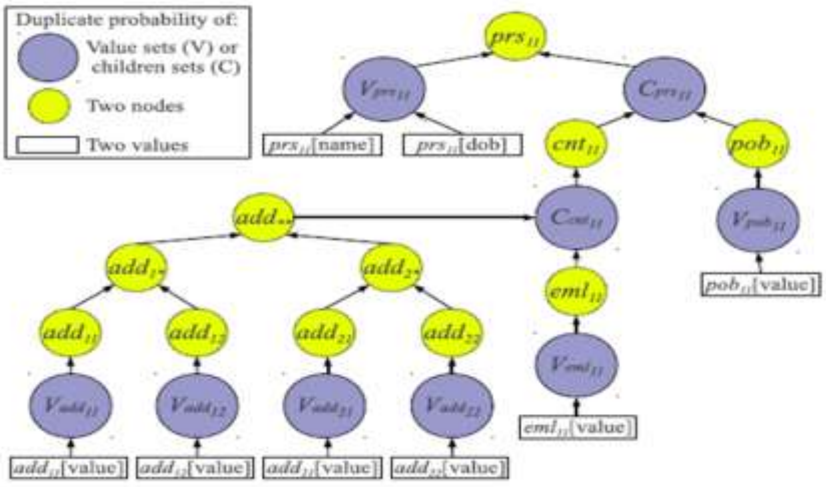

Figure 2 : BN to compute the similarity of the trees in Figure 1

In Fig2 represent the Bayesian Network [2] to compute the similarity between 2 XML objects that represent same elements which is shown in Figure 1. In this type of network ,the node prs11 represent the possibility of the node prs1 in the XML TreeU with respect to the similarity node prs1 in XML TreeU'. This root node prs11 has two children nodes VPrs11 and CPrs11 where node VPrs11, that represent the possibility of child nodes of the prs nodes being similar. In detailed, node Vprs11 contains value of attributes of as prs11[name] and prs11[dob] as shown in a rectangle that checks for its similarity. Node Cprs11 contains node pob11 and cnt11 where node pob11 and cnt11 represents the possibility of node pob1 in the XML TreeU with respect to the similar node cnt1 in the XMLTreeU'.

There is slightly different procedure is to be followed when there is a same type of multiple nodes are present in given XML element just like as a add field in our example of figure 1. So in this case, we want to compare the full set of nodes, instead of each node independently [2], for this reason we create node add** under that node we compared add11, add12, add21, add22 of TreeU and TreeU'. Pseudo code of construction of BN algorithm is given by author [4].

3.1 Computing Probabilities

There are two types of probabilities are used in this method that are : Prior Probability and Conditional Probability. Prior Probabilities are the probabilities of values being similar with respect to their parent XML node, i.e. , P(prs11[name]), P(prs11[dob]), P(pob11[value]), P(eml11[value]), and P(addij[value]). This type of probability can be defined with the help of a similarity function as sim(.) whose values are ranges from the 0 to1 [2],[4].

There are four types of conditional probabilities are 1) the probabilities of the value of nodes being similar, given that each pair of value is similar. Here we not directly consider all the attributes, but we focus on the attribute whose values are being similar. 2) The probability of the child nodes being similar depends on an every pair of children is similar. 3) The probability of two nodes is similar depend on their values, and their child becomes similar. 4) The probability of a same type of nodes is there in BN then for checking similarity we want to take each and every pair of individual nodes to be considered [2].

3.2 Proposed Algorithm

In this section we discussed the actual algorithm which we follow to find out similarity in XML data which we follow to find out similarity in XML data which based on the [1][2][11][12]. In our algorithm instead of taking all attributes for comparison, we will only consider the attributes which are useful. On the selection of that attribute we can say that it is a similar object and we will not consider other attributes. So it will give better performance than previous methods. If attribute value contains any white space then we will first remove the white spaces and after that perform the comparison for finding out the similarity between two objects. This is one of the parts of typographical errors. We also check other type of typographical error where spelling of any word is differs as per the pronunciation of the individual. For example Asmi and Ashmi, Asoke and Ashoke, Ashu and Aashu, Color and Colour and so on. But for this we want to analyses the whole dataset to carefully to find all this types of typographical errors.

The details of an algorithm as follows : First of all algorithm takes an input as (N,T) where N stands for nodes and T stands for predefined threshold value upon which we can say that a given node is similar or not. Then it takes a list of all parents nodes of N and assumes that their probability score is 1. Then a next step the actual probability value of each node of parents of N are computed. If a node n is value node then we compute the probability score by finding the similarity of the values it represents. Selection of nodes and attributes are depends on the user given at runtime. On the other hand if node n is not a value node then we compute recursively until the leaf node not found with updated threshold value T with respect to that particular node. Ones the score for node n is calculated then our algorithm compares the total score for N with threshold value T and then decides whether to continue or stop the algorithm. Here function compute Probability() consists by applying one of the conditional probabilities.

IV. EXPERIMENTAL EVALUATION

In this section we will compare our method with XMLDup approach [2] and DogmatiX approach [3].

4.1 Experimental Setup

To know the effectiveness of our method , we applied commonly used Precision and Recall [13]. Precision measures the percentage of correctly identified duplicates, over the total set of duplicate objects. This method will run on minimum hardware requirement which include Intel 2 Core CPU at 2.53 GHz and 4 GB of RAM. Having any windows XP/7/8 as operating system or Ubuntu as an open source system. The algorithm will be implementing in Java programming language which uses DOM API to deal with XML objects.

4.2 Data Sets

To perform the similarity detection we are using four different type datasets which represent different data domains. These data sets are Restaurant, Cora, CD2, IMDB+FilmDienst. These datasets are a real database, so we want to add some polluted data related to the data objects. And also make sure that it contains different types of errors such as a typographical error, missing data and similar erroneous data[4]. The size of the dataset varying from 9,763 objects (CD2) through 864 objects (Restaurant). For all experimental results, we will be used threshold value as 0.3 for Cora dataset, 0.4 for CD2 dataset and 0.5 for remaining datasets i.e. for Restaurant and IMDB+FilmDienst. If possible we will be varying the threshold value T to achieve high accuracy in the results.

4.3 Expected Results

After implementing our results with the XMLDup that is given in [2]. We choose XMLDup because we are using it as our basis of our paper, and it also already compared it with their previous Duplicate detection system called as DogmatiX [3][7].

V. CONCLUSION

In this paper, we proposed the unique method which can find out similarities in the XML data elements. Bayesian network will determined the probability of two similar XML elements. The Bayesian Network is used to form the structure of elements that are being compared with each other. In addition to improving the efficiency and effectiveness, we check for one of the types of typographical errors in which two elements are compared by removing its white spaces. We also check for another type of typographical error where spelling of any word is differing as per pronunciation of that individual. It requires little user intervention because user only needs to give attribute list to be compared and their threshold value.

REFERENCES

[1] Luis Leitão, Pável Calado and Melanie herschel, "Efficient and Effective Duplicate Detection in Hierarchical Data," IEEE Transaction On Knowledge and Data Engineering, Vol. 25, N0 5,pp. 1028-1040, 2013.

[2] Melanie Weis and Felix Naumann, "Dogmatix Tracks Down Duplicates in XML," Proc. ACM SIGMOD Conf. Management of Data, pp. 431-442, 2005.

[3] Luis Leitao, Pavel Calado, and Melanie Weis, "Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection," Proc. 16th ACM Int'l Conf. Information and Knowledge Management, pp. 293-302, 2007.

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

[5] Diego Milano, Monica Scannapieco and Tiziana Catarci, "Structure Aware XML Object Identification," Proc. VLDB Workshop Clean Databases (CleanDB), 2006.

[6] Marcin Szymczak, Sławomir Zadrożny, Guy De Tré, "Coreference detection in XML metadata," IEEE International Conference on IFSA World Congress and NAFIPS Annual Meeting, pp. 1354-1359, 2013.

[7] Thandar Lwin and Thi Thi Soe Nyunt, "An Efficient Duplicate Detection System for XML Documents," IEEE International Conference on Computer Engineering and Applications (ICCEA) , pp. 178-182,2010.

[8] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti, "Eliminating Fuzzy Duplicates in Data Warehouses," Proc. Conf. Very Large Databases (VLDB), pp. 586-597, 2002.

[9] Mauricio A. Hernández and Salvatore J. Stolfo, "The Merge/Purge Problem for Large Databases," Proc. ACM SIGMOD Conf. Management of Data, pp. 127-138, 1995.

[10] Luis Leitão, and Pável Calado, "Duplicate Detection through Structure Optimization," Proc. 20th ACM Int'l Conf. Information and Knowledge Management, pp. 443-452, 2011.

[11]Luis Leitão, and Pável Calado, "Efficient XML Duplicate Detection Using an Adaptive Two-level Optimization," Proc. 28th Annual ACM Symposium on Applied Computing, pp. 832-837, 2013.

[12]Sutheetutt Vacharaskunee, Sarun Intakosum, "XML Path Matching for Different Hierarchy Order of Elements in XML Documents," 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 82-86,2010.

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