IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Volume 14, PP 43-46

A Web-Based Application For Merging XML Files

Arko Sarkar[1], Saurabh Patil[2], Shubham Sharma[3], Neha Kunte[4]
[1]*Student, Computer Engineering, Atharva College of Engineering, Mumbai, India.*
[2]*Student, Computer Engineering, Atharva College of Engineering, Mumbai, India.*
[3]*Student, Computer Engineering, Atharva College of Engineering, Mumbai, India.*
[4]*Assistant Professor, Computer Engineering, Atharva College of Engineering, Mumbai, India*.

Abstract: *Today various applications are developed which are performing various complex operations simultaneously. Now, it is not important what applications are being developed, rather the importance here is what data is being used, and how these data are stored. So the main aim in every type of applications is to reduce the overall time and space complexities.*
XML plays an important role in these type of web applications. XML document comparison becomes a central issue in database and information retrieval, during development of application. Many heterogeneous XML sources have similar content and different structures or vice-versa. Hence to centralize all related information about a topic, merging of XML files according to certain standards is of greatest need and many solutions are emerging. Integration of XML documents that are similar in terms of structure and semantic content is needed. Therefore we are developing an application based on the needs above, where the program will be able to compare the documents having different content structures and then give a proper unified document which will be compact and easier to understand and store. Thus this will make information transfer and retrieval efficient.
Keywords: *Clustering, Data analysis, Similarity Score, XML, XRel Schema.*

I. Introduction

In common usage, XML is used to store data. A lot of programs are written and various operations are performed by computer programmers, developers, students and sometimes even a common user. Now the data which will be generated out of this can be written in XML format and this data can sometimes be enormous. This kind of data can also be categorized into different sections, where if we look properly these sections will have their own sub-sections and it'll continue. Now the problem arises when you start to exceed the pre-defined limit or it takes huge time to load the data as it is too much for the system to retrieve. In such times, it's a humongous task to go through all the data, as it can have more than thousands of lines of code. This data isn't supposed to do anything, as XML data aren't supposed to do anything or particularly perform any function, so this isn't like a website code or something. Rather, it is just a raw format of the data and it is important to have this stored properly and it shouldn't get deleted as it'll affect not only the data, but all the applications which are dependent on this data.

So, this project tries to operate on all the data that are stored by first identifying the different nodes of the document, where the nodes are actually the different elements of the document which are represented in a tree diagram. Then according to our algorithm, we first determine the leaf nodes and their respective parents then consider these nodes as our clustering points. Now, these clusters are then compared with each other, where different roots of these clusters are checked to find any values which match amongst them, also the values here will be the value of the attribute element where before getting to the comparison we need to first check whether the elements being compared are same, and the attributes too should be same.

In this way, we are recursively traversing bottom-up and determining various nodes and their parents, then comparing and finding values which are equal, and then traveling up until we reach the root node and all the nodes are compared and optimal number of matches are found. The algorithm which is being used here will help us to determine the most efficient way of getting the matches in the shortest time possible. About the efficiency of using the algorithm, using this will help us to find the best number of matches which is similar in contents and also in their semantics.

1.1 Need:

XML documents have been very crucial for data representation and exchange. But these documents store the raw data of the website and this data may be redundant. As the popularity of a website increases, the number of users increases and so do the data generated. This again increases the redundancy and introduces other problems like website getting time out during retrieval of the data, or users are unable to reach the website or use it because data cannot be loaded, the storage is getting exceeded after storing so much data acquired after this increase in usage and other reasons. Hence all of these issues have resolutions like buying new servers to

store the data, also these servers should be quicker than before to reduce the access time etc. But, this kind of resolutions if implemented will just keep on getting repeated and nothing is being done actually on why the data being generated is too much larger in size and requires more space and time to store and obtain respectively etc. So, a need for optimizing the data is required where applying an algorithm on the data is the best option!

1.2 Basic Concept:

Basic concept of this project that user will write or have a set of XML documents which will have large data, these documents will then be uploaded to the web-based application where a predefined document standards needs to be followed and implemented (i.e. pre-processed by user before uploading). System will then check for the document's integrity, calculate a similarity score if it follows every rule defined, and will merge the documents and return to the user.

II. Existing System

W. Viyanon [3] has proposed a system to merge XML documents, where the document is first represented in a tree structure and then the root and its children are divided into sub trees. In this paper, the author is comparing the leaf node parents which have unique values which are in the keys section. These sub trees are compared to other sub trees of documents already stored in the database to find out whether there exists similarity between them by using their key values.

The subtrees thus found to be similar are matched according to their content and semantic similarity. Also the leaf-node parents compared are then shown in the merged document, and then it traverses recursively bottom-up till it reaches the root node and it also performs same algorithm comparison to find all the possible matched subtrees. Matched subtrees are then merged which in turn merges the documents giving one single document consisting of resultant similar data. [3]

M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura [4] have described XRel, an approach to store and retrieve XML documents using relational databases. In this approach, an XML document is decomposed into nodes based on its tree structure, and stored in relational tables according to the node type, with path information from the root to each node. XRel enables us to store XML documents using a fixed relational schema without any information about DTDs and element types, and also enables us to utilize indices such as the B+-tree and the R-tree supported by database management systems.

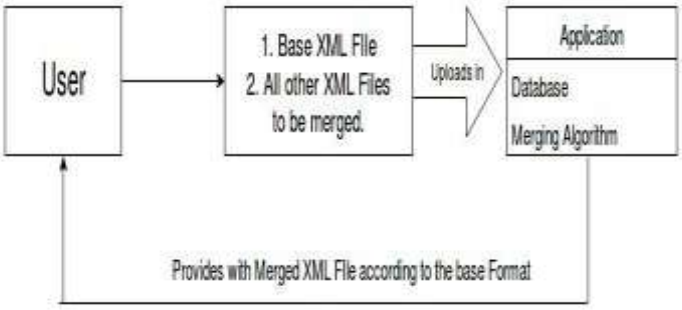
For the processing of XML queries, we present an algorithm for translating a core subset of XPath expressions into SQL queries. Thus, XRel does not impose any extension of relational databases for storage of XML documents, and query retrieval based on XPath expressions can be realized in terms of a pre-processor for database query language. [4]

III. Proposed System

The functionality of this system will consist of three main steps. In the initial step, the user will open the web application, which is the system's user interface which user can interact with to perform various operations as given by the system and required by the user. The user will now upload document(s) under the basic assumption that the document has some similarity. If the document doesn't have similarities in between them or if there are some other issues, then these will be shown in the further steps. This entire step can be said to be the initiation stage. The user will now upload the document by locating the document upload option in the dedicated area given in the application. The system will know do a pre-processing operation on the document, where the actual merging process isn't carried out first, rather the similarity score is calculated, which is a score based on the similarities between the document(s) which can be in the form of a simple score or even a percentage, and the user can now understand after this output how much the system can be useful to it or the user can go back a step and re-edit the documents to make them similar. Now, after the first two steps the user is familiar with the system and how it will respond to its document, and now user will confirm through a system pop-up if it wishes to continue. When the user responds affirmatively, the system operates on this data given and after successful operation the system will return a downloadable output file or the file text will be given in a text-field of the website. User will have the option to choose any one of them, the differences between them are that the direct output pasted in the text-field will be easier to create and faster to achieve than creating a downloadable file, and user can also export the outputted data in any word-processor he chooses to get the file in his desired format. Thus, the system will process the entire document with the proposed algorithm.

IV. Methodology

In the proposed system, system will take the user data in XML format and process it through the system and return proper merged file. This is explained above and the functional block diagram of the system is:



This section explains the block diagram shown above in detail:

4.1.1 Designing the front and back ends of the system.

First, the front-end and back-end of the website are designed and they are interlinked with each other such that there is a proper system available for the user to upload his document(s), and then this document is stored in the server (a relational database particularly) properly such that the website administrator is able to access this document whenever he requires or the algorithm can fetch this document for further processing.

This can be implemented in two variants, one where PHP will be used for designing the backend of the system and thus will be an important link between the two ends. This system is useful for operating on larger files (i.e. storing larger files and processing them) and for scalability of the system in future. Thus, this can be implemented as a large-scale or commercial system.

The other variant of the system will be by using front-end development technologies only, like HTML/CSS/JS where there won't be a dedicated server and hecen a dedicated database for user needs, and the user needs to do all operations by using his file system as the main source of documents. This system too is locally hosted and won't be much extensible, and will be advantageous for any document processing where file size is small or when the system isn't expected to be big or scalable later.

4.1.2 Document upload by the user.

User will upload his document and the server will have the document's copy. User will have the option to store multiple documents. If he doesn't upload anymore or fails to upload then the website will simply process the document uploaded originally as it is to the server and it'll be returned to the user, there won't be any changes done.

Once sufficient documents are uploaded on the server, where sufficiency will be decided as per user's requirements, then the main processing will start where the algorithm will do the further processing.

4.1.3 Pre-processing results of the document

Here, the system will return a similarity score calculated by it on comparing the set of documents, and it'll let the user know how similar these documents are if he wishes to do further processing, and how much changes he can make if he wants to improve the similarity output, by taking the document and going a step back, and then re-editing the document to make it more similar.

4.1.4 Actual merging operation on the data.

The final steps of the method will be that the algorithm will return the processed document after calculating the similarity score, where how much the document is processed as per the user's requirements will be decided on whether any matches were found or not, what was the similarity score which is the main factor etc.\

4.1.5 Different user roles described in the system.

Also, administrator will have a separate login to manage the documents uploaded and to perform further maintenance and upgrade in the system.

User will be able to interact with the interface properly and upload document in the system and also retrieve the output file as a downloadable or text-only format from the system.

4.1.6 Algorithms used in Merging:

Our main algorithm will be a combination of previously known algorithms XDoI and XDI-CSSK(which are stated in appendix) and will use some concepts from XML, tree structures. The algorithm will implement the concepts from the above.

Using concepts of XML-Merge:

The XDoI and XDI-CSSK algorithms basically use the relationship between the node and its parent to find the clustering point and then compare this with the other tree such that the key values of both of these match. [7] When matches between the clustering points are found, it means that the all the child value of the current node's can be merged with the other, as the parent's have same values. More about these algorithms are given in the appendix. But the issue in these algorithms are that they consider the content similarity only between the nodes, they don't consider the semantic similarity at all and then proceed with the further steps, Now, in this algorithm it does further improvement over the first two by considering the semantic similarity too over the content similarity. This semantic similarity will be calculated here. This scanning at an earlier stage will improve have further improvement over the performance. [6]

V. Conclusion

Hence, for a very large system where there are multiple XML files which has large data stored in it, merging them will enable fast access of the system. It will become more convenient for the developer to access file of a specific type as the storage of all files related to a particular field are centralized in one whole document.

Multiple developers may find additional information effortlessly in the merged file. It will also make the storage more flexible as there is no need to allocate space separately for files that are already merged in the database. The time required to access files separately is too high and hence merging reduces the complexity of file search by providing all information that is needed by the user in a single document itself.

References

[1] JiSim Kim, Wol Young Lee, KiHo Lee , "The Cost Model for XML Documents in Relational Database Systems", in *Computer Systems and Applications, ACS/IEEE International Conference* on. 2001

[2] Sarvadee Sae Tam and Gae Keng Hoon, "An Efficient Similarity Matching for Clustering XML Element," in 2016 *Third International Conference on Information Retrieval and Knowledge Management*

[3] W. Viyanon, MXML: Implementation of a Web-based Application for Merging XML Documents using XML-SIM, in *ICT and Knowledge Engineering (ICT & Knowledge Engineering 2015), 13th International Conference*.

[4] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, "XRel: a path-based approach to storage and retrieval of XML documents using relational databases," *ACM Transactions on Internet Technology*, vol. I, no. I, pp. 110-141, 2001.

[5] W. Viyanon, S. K. Madria, and S. S. Bhowmick, "XML data integration based on content and structure similarity using keys," in On the Move to Meaningful Internet Systems: OTM 2008. Springer, 2008, pp. 484-493.

[6] W. Viyanon and S. K. Madria, "XML-SIM: Structure and content semantic similaritydetection using keys," in On the Move to Meaningful Internet Systems: OTM 2009. Springer, 2009,p p. 1183-1200.