Mining XML Data: A Clustering Approach

Mohamad Saraee
The University of Salford
Salford, Greater Manchester
M5 4WT

Joanna Moaiad Aljibouri
The University of Salford
Salford, Greater
Manchester
M5 4WT

Abstract
XML data has become very popular to represent semi structured data. This has resulted in a growing amount of XML data on the web. This raises a need for languages and tools to manage collections of XML documents as well as to mine interesting information from them. Several attempts at developing XML mining techniques have been proposed. However the topic of mining XML data has received little attention as the data mining community has focused on the development of techniques for extracting common structure from heterogeneous XML data. This project aims to data mine XML data using the XML Query language XQuery. The data mining technique used is the clustering technique of the Nearest Neighbour Algorithm.
This algorithm will be incorporated into XQuery expression which, when implemented using an XQuery implementation tool, will cluster distance based data within the XML document into groups, where the distance between the data is set by a given threshold.
The implementation of the Nearest Neighbour algorithm hopes to be generic and implement a user interface which allows the user to load a XML document for its data to be clustered, choose the data to be clustered within that document, input the threshold and receive the clustered result in an output file. This work would allow XML distance data to be clustered with the Nearest Neighbour algorithm using XQuery, therefore providing a needed data mining implementation on XML data.

Keywords
XML, Data Mining, Nearest Neighbour, XQuery

1.0 Introduction

It has been increasingly important to be able to mine XML as it has become very popular as a standard to represent semi structured data. XML has also received a lot of attention from the database community and there has been a huge amount of work aimed at coping with XML data. An example of this work would be work on storing XML data, indexing and querying XML content [2,3]updating XML data [4] and benchmarking XML applications[5]. But little attention has been made to mining the actual data within XML. The data mining community has concentrated more on the development of techniques for extracting common structures from the XML data.
It was thought that in order to mine XML data it would first need to be pre or post processed from an XML format to a relational format. A lot of research has gone into data mining XML documents after the format has been changed from XML format to a relational format. Little has been done on directly mining the xml data and integrating SQL, without this post processing the XML data.

At first the focus was to store XML data. Today the focus is more on creating powerful query and retrieval methods. This has led to the new XML Query language, XQuery[10]. XQuery is the first language to receive industry-wide attention and support. It is currently being developed by the W3C XML Query Working Group. Industry experts expected XQuery to do for XML and XML databases what SQL did for relational data and relational database systems: provide a vendor independent, powerful and easy-to-use method for query and retrieval of XML data

It makes sense to try and use this valuable tool to implement a data mining algorithm to mine XML data. XML data is different from relational data in several important respects that influence the design of a query language. Relational data tends to have a regular structure, which allows the descriptive meta-data for this data to be stored in a separate catalogue. XML data in contrast is often quite heterogeneous and distributes the meta data throughout the document. XML documents may contain many levels of nested elements, whereas relational data is flat. XML documents have an intrinsic order whereas relational data is unordered.

XQuery operates on the abstract, logical structure of an XML document, rather than its surface syntax. This logical structure is known as the data model. XQuery Version 1.0 is an extension of XPath Version 2.0. The data model that XQuery uses is based on that of XPath and defines each XML document as a tree of nodes. The data model is not only capable of handling documents but is also designed to work on well-formed document parts (a.k.a. "fragments"), collections of documents, or collections of fragments.

XQuery is a functional language where each query is an expression. There are 7 types of expressions in XQuery: path expressions, element constructors, FLWR expressions, expressions involving operators and functions, conditional expressions, quantified expressions and expressions that test or modify data types. The various expressions can be used together both sequentially and nested.
The main contributions of this paper are as follows:

- Develop an implementation of a data mining technique to perform mining on XML data. This is needed as there has been little work done on mining this large data source
- Use the XML query language XQuery to implement the technique, which is a relatively new query language

This would contribute greatly to the area of data mining XML data as there is lots of XML data. By providing a implementation of a data mining technique this will help to facilitate the possible patterns not yet discovered in XML data

2 Problem statement

The problem is to implement a Data mining technique in XQuery to be used on XML data.
This paper is concerned with the clustering technique of the Nearest Neighbour algorithm.
Clustering is the process of grouping physical or abstract objects into classes of similar objects. It is an example of unsupervised learning.
Clustering examples are:

- Identifying similar web usage pattern from web usage logs
- Grouping houses into a town into neighbourhoods based on similar features.

Data clustering identifies clusters or densely populated regions according to some distant measurement in a large data set. Clustering identifies the sparse and crowded places and discovers the overall distribution patterns of a data set. Using a distance based approach, means that for each clustering decision all data points are inspected equally and use global measurements within requires scaling all data points or all currently existing clusters.

The Nearest Neighbour algorithm employs distance functions to determine the closeness between data entities. It is assumed that points in space that are close together have similar properties. So for new data, computes its point in space based on some predefined computation, then find out how close it is to known data points and then determine it properties. Items are iteratively merged into existing clusters that are close. This technique uses a threshold t, to determine if items are added to existing clusters or a new cluster is created. It is incremental and uses a time of $o(n^2)$ and space of $0(n^2)$.The algorithm itself can be seen below:

Nearest Neighbour Algorithm
Input:
D = {t1, t2, ..., tn} // Set of elements
A //adjacency matrix showing distance between elements
Output:
K // Set of clusters.
Nearest Neighbour Algorithm:
$K_l = \{t_1\};$
$K = \{K_1\};$
k =1;
for *i* = 1 to *n* do
find the tm in some cluster *K*m in *K* such that dis (ti, tm) is the smallest;

if dis $(t_i,t_m) \leq$ t then
$K_m=K_m \cup t_i$
Else
k=k+1;
K_k = {ti};

2.1 Related Work

Data mining XML data has received little attention. As previously mentioned the data mining community has focused on other areas such as XML storage and its structure [6].This paper involved trying to classify XML data using a rule based classifier called XRules. When this classifier was compared to a association based classifier CBA [7] and an index rule IR classifier, it out performed both of them. They concluded that since the XRules performed better than the CBA classifier this indicated that the system relied on the classification information hidden in the structures for an effective rule generation process. They also stated that it outperformed the IR based method in spite of the greater amount of input used by this method. Their results showed that structural mining can provide new insights into the process of XML
Other work has concentrated on extracting association rules from XML data such as in [1, 8] both using XQuery.
Braga *et al* [8] presented a *XMINE* operator a tool they developed to extract XML association rules for XML documents. XMINE was based on XPath and inspired by the syntax of XQuery, which allowed them to express complex mining tasks, compactly and intuitively. They thought that XMINE could be used to specify indifferently (and simultaneously) mining tasks both on the content and on the structure of the data. They found that their research provided a good starting point it left many open issues for further research which need to be addressed to have a fully functional and efficient implementation
of the XMINE operator. It was pointed out the one of the most important open issues was that of support evaluation as There are currently no XQuery interpreters to be used in filtering XML fragments and

although XPath processors are quite advanced, they do not yet provide all the required functionalities either. They suggested that this extra functionality should be included in the new XPath 2.0.
Wan and Dobbie [1] have shown that it is possible to extract association rules (data mining technique) from XML documents without any need for pre or post processing by using XQuery to implement the well known Apriori algorithm. Their ideas was that XML data can be mined using XQuery and then integrate the data mining technique of association rule mining, into XML native databases. Association rule mining is a popular data mining technique with uses implementation of the Apriori algorithm. This algorithm finds associations between items in a database. They concluded that XQuery implementation of the Apriori algorithm was not efficient in comparison to a C++ implementation.

3 Methodology

The XML clustering software using XQuery takes an Object Oriented Programming approach. It is implemented in Java. This uses a open source XQuery implementation tool called Qexo [9]. The software is divided up into the following classes Start, GUI, ExpressionBuilder, XQueryExec and XMLTreeBuilder. How these classes work together can be summarized as follows:
1) The start class initiates the start of the program. It creates an instance of the GUI class. This provides a GUI interface that allows the user to load a XML Document they wish to be clustered.
2) This document is then parsed using a SAX API. This deals with the document sequentially and builds a Tree model from the elements nodes as it goes through the document. As it does this it also inputs the element nodes into an array. The nodes are tested for whether they contain numerical or textual data. If the data is textual it is put into an array which contains nodes that contain textual data. These elements will not be input into the XQuery expression but will be used purely for display purposes. If the data is numeric then, the element will be put in a separate array. This distinction needs to occur to prevent elements with textual data being input into the Nearest Neighbour algorithm which only works on numerical data.
3) Both of these arrays are then displayed in the Gui as a list of tick boxes with labels. If the data is from the numeric array then a threshold box will appear giving the user an area to input a threshold. If from the textual array a threshold box will not be displayed. This allows the user to choose which elements in the document to cluster together and also to display other attributes to make it easier to make sense of the results.
4) The user will also be asked for a path to the output file for the results to be sent to.
5) From the user's options, the path for the chosen elements and the chosen elements will be incorporated into the XQuery expression. The expression will be built suing the XQueryBuilder class.
6) This will result in the full expression being evaluated in the XQueryExec class. It does this by using the runXQuery().
7) The output results will be sent to the output path specified in the form of a HTML page with an embedded table of clustered results.
8) On completion the display will inform the user it has completed, it will display the output path and give the user the option to exit or to cluster again.

4 Experimental Setup

This software was developed and tested using an iris data set. Here is a sample of the data:

```
<?xml version="1.0" encoding="UTF-8" ?>
<irisdata>
-<Iris>
  <sepal_length>5.10</sepal_length>
  <sepal_width>3.50</sepal_width>
  <petal_length>1.40</petal_length>
  <petal_width>0.20</petal_width>
  <class>Iris-setosa</class>
</Iris>
```

As it can be seen the element values consist of both numerical data, i.e. that is contained in the element <petal_width>, and textual data e.g. the data within the <class> element.
The software works out the time taken to cluster the results. A n experiment was conducted that wanted to show how the software processing time altered when the number of variables clustered increased. It entailed requesting that one variable i.e. petal_width was clustered with a threshold of <0.1,the time taken and this repeated 5 times to get a number of results. This was then repeated for 2,3 & 4 variables with different thresholds. The Iris document contained 152 <iris> nodes which means that the algorithm (whose performance is based on O(n^2). Regardless of the number of variables clustered the algorithm reads the document 152^2 times. However, the work done increases, as the number of variables to cluster, increases.

4 Experimental Results

As previously mentioned the results will be in the form of a HTML page with an embedded table of clustered results. The headers of the table will be taken from the element name within an array.
A sample of the output of clustering 2 elements and displaying the class is as follows:

Nearest Neighbour clustered results
This data is your clustered results from the XML Document you have chosen. The chosen elements and thresholds were:

Element : petal_length
Threshold : 0.2
Element : petal_width
Threshold : 0.5

These have been put through the XQuery Nearest Neighbour algorithm and produced these results.

The nearest neighbours of the node(s):
petal_length

1.40

petal_width

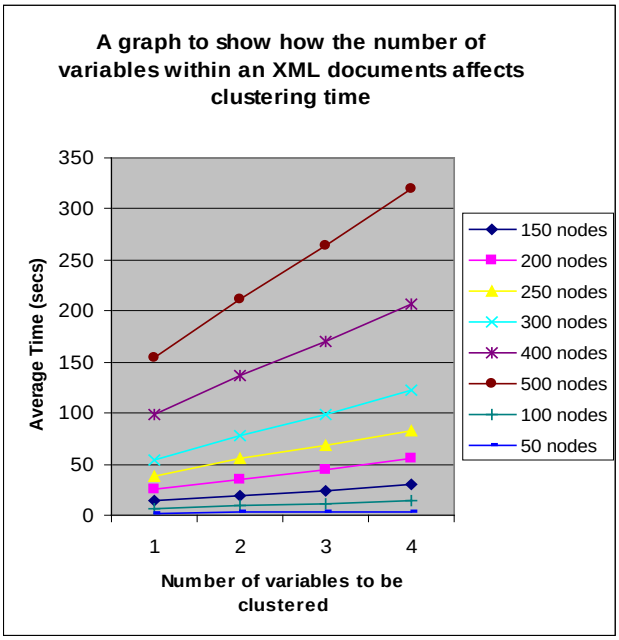
0.20

Class

Iris-setosa

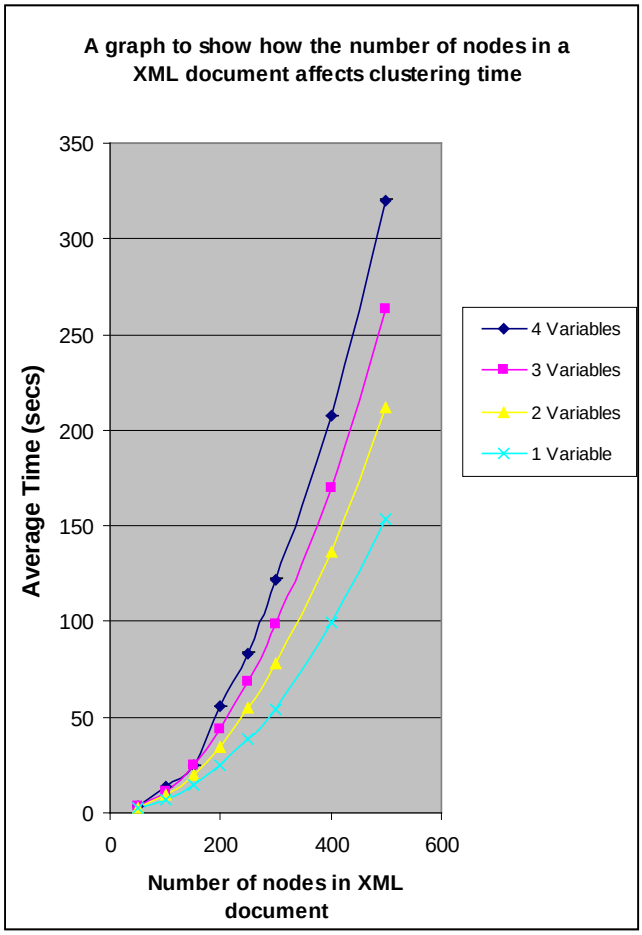
petal_length	petal_width	class
1.40	0.20	Iris-setosa
1.40	0.20	Iris-setosa
1.30	0.20	Iris-setosa

The output was configured to make it easy to read for the user to analyse the results and thus easier to identify possible patterns.
The speed in which the data is clustered depends on many factors such as the size of the document, the specification of the computer and what it has running at the time and the number of variables that are input into the XQuery Algorithm to cluster.
With the sample data, which contained 152 <iris> nodes .The following graph can be displayed to show time against work done. As the number of variables clustered together increases so does how much work the XQuery expression has to do. The time taken to cluster the sample data document with 1-4 variables was recorded 5 times for each variable. Then the average and Standard Deviation was calculated. This was repeated with increasing sample data sizes The results can be displayed as follows:



Graph 1.Number of variables against time

The time was also measured with a differing number of nodes within a XML document. They are displayed as follows:



Graph 2 The number of nodes against time

4.2 Discussion

The results in graph 1 show that there is a linear relationship between the time taken to cluster the data

and number of variables clustered. The algorithm performs fairly constantly with little variation in performance which is demonstrated by a very small standard deviation (with a range of 0-1.3 for the whole data set). As the number of nodes within the XML document increases the gradient of the line becomes greater. Graph 2 demonstrates that the nearest neighbour algorithm works with a performance of O(n^2).It is expected that the time taken to cluster the data will increase exponentially as the number of nodes increases.

5 Conclusions

This software provides a tool to mine XML data into clusters of the data points nearest neighbours according to a given threshold. It provides a GUI interface which makes it much more adaptable for a user. However it has been shown that clustering time greatly increases as the number of nodes within an XML document increases. This is the nature of the nearest Neighbour algorithm implemented. Therefore with larger data sets it will take a much longer time to cluster, greatly increasing CPU usage and virtual memory on the system running the software. The bigger the data set to be clustered the more resources are going to be needed.

6 References

[1] Wan, Jacky W.W . Dobbie, G. Mining Association Rules from XML Data using XQuery, http://crpit.com/confpapers/CRPITV32Wan.pdf

[2] Quanzhong Li and Bongki Moon. Indexing and Querying XML Data for Regular Path Expressions. In *VLDB*, 2001

[3] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. Answering XML Queries on Heterogeneous Data Sources. In *VLDB*, 2001

[4] Yannid Papakonstantinou and Victor Vianu. Incremental Validation of XML Documents. In *ICDT*, 2003

[5] The XML benchmark project. http://www.xml-benchmark.org

[6]Zaki and Aggarwal,XRules:an effective structural classifier for XML data,KDD 2003

[7] B. Liu, W. Hsu, Y. Ma. Integrating Classi_cation and Association Rule Mining. SIGKDD, 1998.

[8] Daniele Braga_ , Alessandro Campi_ , Stefano Ceri_ , Mika Klemettinen_ , Pier Luca Lanzi_,A Tool for Extracting XML Association Rules from XML Documents

[9] http://www.gnu.org/software/qexo/

[10] http://www.w3.org/TR/xquery/