International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
ISSN (Online) 2349-6967
Volume 2 , Issue 1(Jan-Feb 2015), PP42-46

XML clustering using Balanced K-Means Algorithm

Nithin N.[1], Shyama Srivatsa Bhatt[1]
Department of CSE, SDM Institute of Technology, Ujire, Karnataka, India

Abstract :- XML is considered as standard way of representing and sharing the data across the network of different platforms. The amount of data is increasing the exponentially and efficient data management techniques are required. In this paper, an XML clustering strategy based on Balanced K-Means algorithm is presented. For testing the clustering methodology custom XML dataset is created. The clustering algorithm takes set of XML files and clusters them separately based on the depth of the similar elements across different files.

Keywords: - XML Clustering, Data Mining, K-Means, Balanced K-Means

I. INTRODUCTION

There are more and more eXtensible Markup Language (XML) documents released on the web. Clustering XML documents is helpful to XML search engine. Since element and its arrangement in the hierarchy not only describe the XML document structure, but also implicitly provide its semantic meaning, clustering XML documents needs to consider both element and its structure. The XML language is becoming the standard Web data exchange format, providing interoperability and enabling automatic processing of Web resources. Because of this reason managing and handling XML documents is a current research issues.

Clustering is a process of collecting data into classes of clusters, so that objects within a group have high similarity in comparison to one another but are very dissimilar to objects in other clusters. Dissimilarities are assessed based on the attribute value based on the object. A simple path represents the tags from root node to an internal node or terminal node; it not only includes the XML tags but also reflecting the structure of the XML documents. Using path as feature of XML document is a good method to compute the similarity among XML documents.

To reduce the number of path features used a prior algorithm to find the frequent paths and take these frequent paths as XML document feature that called common Xpath, each XML document was represented with common Xpath feature vector, selected the distance function to compute the similarity matrix. But only using common Xpath as feature is not always efficient especially to those XML documents with many different structures [1]. For example, in the following three XML documents, if support count is 2, then the frequent path is university/teacher/name, which is denoted by CXP1. First do the data set grouping based on likenesses, and assign labels to the groups. Advantage of this type of clustering techniques is easy to change the useful features that distinguish different groups.

The clustering of XML documents facilitates a number of application such as improve information retrieval, document classification analysis, structure summary, improve query processing, and XML search engine. XML is different from other web documents such as HTML or text because it contains the hierarchical structure and relationships between elements. In the future work, there will have many XML repositories. XML, like HTML, makes use of tags and attributes. While HTML predefines the meanings of tags and attributes, XML can define their own document formats and allows users to specify elements (tags) and attributes using their own words. The elements and their arrangement in the hierarchy describe the document structure and imply its semantic meanings [2].

The rest of the paper is organized as follows. In section 2, describes related research works. In section 3, Balanced K-Mean clustering algorithm is explained. Then discussed result analysis in detail. Finally a brief conclusion is given.

II. RELATED WORK

In most of the XML document sorting and organizing operations uses clustering techniques. We will briefly describe some of the techniques being suggested by the researchers.

Ramasubramanian and Paliwal in [3] proposed fast search algorithms and studied for vector quantization encoding using the K-dimensional (K-d) tree structure. Here, the emphasis is on the optimal design of the K-d tree for efficient nearest neighbor search in multidimensional space under a bucket-Voronoi intersection search framework. Efficient optimization criteria and procedures are proposed for designing the K-d tree, for the case when the test data distribution is available (as in vector quantization application in the form of training data) as well as for the case when the test data distribution is not available and only the Voronoi intersection information is to be used. The criteria and bucket-Voronoi intersection search procedure are studied in the context of vector quantization encoding of speech waveform. They are empirically observed to achieve constant search complexity for O (log N) tree depths and are found to be more efficient in reducing the search complexity. A geometric interpretation is given for the maximum product criterion, explaining reasons for its inefficiency with respect to the optimization criteria.

Selim and Ismail are addressed several questions about the K-means algorithm in [4]. The clustering problem is first cast as a non-convex mathematical program. Then, a rigorous proof of the finite convergence of the K-means-type algorithm is given for any metric. It is shown that under certain conditions the algorithm may fail to converge to a local minimum and that it converges under differentiability conditions to a Kuhn-Tucker point. Finally, a method for obtaining a local-minimum solution is given.

Tapas et al. in [5] have specified that in k-means clustering, we are given a set of n data points in d-dimensional space Rd and an integer k and the problem is to determine a set of k points in Rd, called centers, so as to minimize the mean squared distance from each data point to its nearest center. A popular heuristic for k-means clustering is Lloyd's algorithm. They have presented a simple and efficient implementation of Lloyd's k-means clustering algorithm, which we call the filtering algorithm. This algorithm is easy to implement, requiring a kd-tree as the only major data structure. The authors have established the practical efficiency of the filtering algorithm in two ways. First, they present a data-sensitive analysis of the algorithm's running time, which shows that the algorithm runs faster as the separation between clusters increases. Second, we present a number of empirical studies both on synthetically generated data and on real data sets from applications in color quantization, data compression, and image segmentation.

Joshua et al. in [6] propose a k-means type clustering algorithm that can automatically calculate variable weights. A new step is introduced to the k-means clustering process to iteratively update variable weights based on the current partition of data and a formula for weight calculation is proposed. The convergence theorem of the new clustering process is given. The variable weights produced by the algorithm measure the importance of variables in clustering and can be used in variable selection in data mining applications where large and complex real data are often involved. Experimental results on both synthetic and real data have shown that the new algorithm outperformed the standard k-means type algorithms in recovering clusters in data.

Hongjun et al. in [7] the authors find that some features influence so much on the results of clustering. For improving the K-means algorithm, the authors design a novel balance K-means algorithm. The main idea is that we normalize all the feature values of dataset before clustering. So all the features play the same important role in the clustering, which make the k-means balanced. There are three contributions to this paper. First the disadvantages of the standard K-means are illustrated in detail. Second we design the balance K-means algorithm which all the values of features are projected into a fix range, so it can take over the disadvantage of the standard K-means and. At last the authors choose some datasets from UCI for experiments. And the results of experiments show that the balance K-means runs better than the standard K-means.

III.METHODOLOGY

3.1. Balanced k-means alogorithm for XML documents

This algorithm partitions X observations of XML documents into k clusters. For each element x_i is calculated as follows:

$$x_i = \frac{x_i - MIN(f)}{MAX(f) - MIN(f)} \qquad (1)$$

Initialize the k centroids as wj = xl, j $\in$ (1, k) and l $\in$ (1, n). In this each cluster cj is associated with the centroid wj. Perform the following operations repeatedly for each input x_i, where $l \in 1,.., n$. The vector x_l is

grouped with the cluster c_j, which is having nearest centroid. For each cluster c_j update the centroid w_j like in equation (2).

$$w_j = \sum_{x_i \in c_j} x_i / |c_j| \qquad (2)$$

The error function is calculated using equation number (3):

$$e = \sum_{j=1}^{k} \sum_{x_i \in c_j} |x_i - w_j|^2 \qquad (3)$$

IV.EXPERIMENTAL SETUP

The proposed algorithm is implemented for XML clustering using C# programming language with .NET plat form and MATLAB tool. The basic file reading and all input/output operations are developed using C# programing language in Microsoft Visual studio 2010 tool. The balanced K-Mean clustering algorithm was developed using MATLAB tool.

Mainly three procedures are used to perform the XML clustering operation like:

- To count number of elements in a given input XML file.
- To find the maximum depth of the given XML file.
- Normalization of row elements.

4.1. Count number of Elements:

For a given source XML file, this function performs operations like, open the file, read its contents and return back the number of XML elements present in the file.

```
1. Open XML file
2. While (File NOT over)
      a. Read next content
      b. Is it an element?
            i. YES: increment counter
3. Return count (number of elements)
```

Assuming that the length of a file can be measured in terms of "contents" (A content may be a reference, image, closing tag … etc), and *N* be the number of contents in a file. This function will has a complexity of *O(N)*.

4.2. Find the maximum depth of XML file:

This function opens the XML source file, read its contents and return back the maximum depth.

```
1. Open XML file
2. DEPTH = 0
3. While (File not over)
      a. Read next content
      b. Read depth of current content
            Is current depth > DEPTH
            YES: Assign DEPTH = current depth
4. Return DEPTH
```

This function also scans over entire file to find the maximum depth. The time complexity is *O(N)*.

4.3. Normalization:

This procedure takes the raw input data and normalizes them.

```
1. Min = first element of data set
2. Max = first element of data set
3. For (i from 0 to number of data elements)
      a. If (Min > current element scan)
      b. Min = current element
      c. If (Max < current element scan)
      d. Max = current element
4. Range = Max – Min
5. For (i from 0 to number of data elements)
      a. Normal of Current = (current element scan – Min) / Range
```

If *M* is the number of elements present, then the 2 loops run over the elements twice. This means the complexity becomes of the order of *O(M)*.

XML files are clustered based on their depth (i.e. based on the number of sub elements). The directory (folder) which contains the XML files to be clustered using the Graphical User Interface (GUI) was created using C# programming language. The GUI will display the list of XML files within that directory, which is shown in Fig. 1.

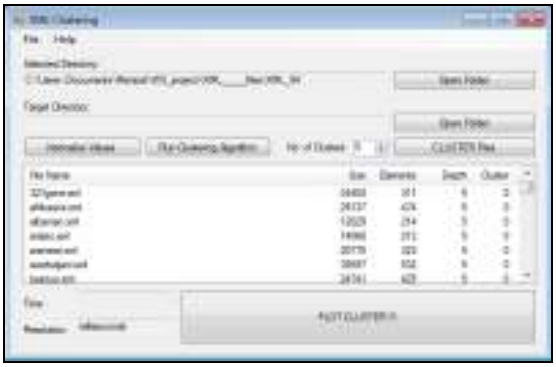

Fig. 1. The directory containing XML files

Enter the number of clusters to be formed in list box and click on the "Run Clustering Algorithm" button to actually run the balanced k-means algorithm. After running the algorithm the GUI shows the list of XML files along with the cluster number to which they belong to. That is shown in fig. 2.

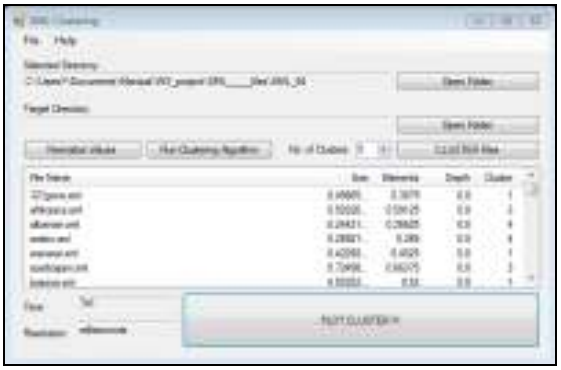

Fig. 2. After execution of clustering algorithm

V.RESULT ANALYSIS

Upon completion of XML clustering result is analyzed by plotting the 3-D cluster graph, in which each dot represents a particular XML file. The dots belonging to the same cluster will be having the same color. The cluster plot is generated by using the three values representing the three axes of the graph which include the Number of elements, Depth and the File Size. By the use of balanced k-means algorithm the depth of the XML files will be clustered based on their corresponding depths. The resulted 3-D cluster plot is shown in Fig. 3.

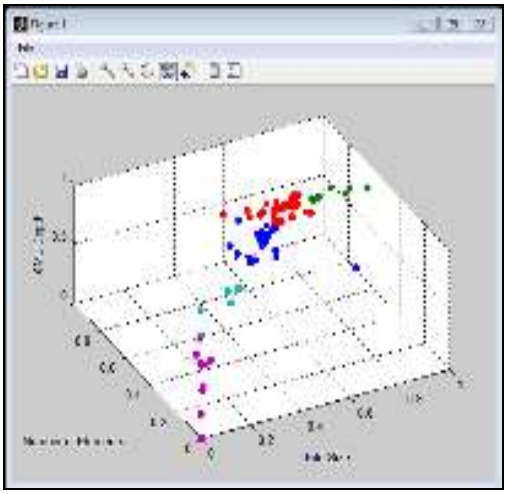

Fig. 3. 3-D XML file clustering

VI.CONCLUSION

In this paper a clustering algorithm for XML files is developed. Clustering of XML files is carried out using balanced k-means algorithm. The algorithm is implemented in Visual Studio and results are obtained in MATLAB simulation tool. The balanced k-means algorithm considers a set of xml files and partitions them into clusters. The syntactic structure of XML file is cluster formation objective. The XML files with similar syntactic structure like number of elements, depth and the file Size are groped to form a single cluster. The algorithm is only tested on custom made XML dataset. In the future the algorithm an attempt will be made to test the algorithm on standard dataset containing large number of XML files.

REFERENCES

[1] M. Khalilian, M. N. Dehkordi and K. Zamanifar, XML Documents Clustering based on frequent and rare tag sequences with automated support thresholds, *International journal of Review on Computers and Software*, vol. 7, issue 4, Aug. 2012, 1-10.

[2] H. Leung, F. Chug, S. C. F. Chan and R. Luk, XML document clustering using common XPath, *IEEE Proceedings of International Workshop on Challenges in Web Information Retrieval and Integration*, April 2005, 91-96.

[3] V. Ramasubramanian and K. Paliwal, Fast K-dimensional tree algorithm for nearest neighbor search with application to vector quantization encoding, *IEEE Transl. on Signal Processing*, vol. 40, issue 3, Mar. 1992, 518-531.

[4] S. Z. Selim and M. A. Ismail, K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality, *IEEE Transl. on Pattern analysis and machine intelligence*, vol. PAMI-6, Jan. 1984.

[5] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, An efficient k-Means clustering algorithm: analysis and implementation, *IEEE Transl. on Pattern analysis and machine intelligence*, vol. 24, July 2002.

[6] J. Z. Huang, M. K. Ng, H. Rong and Z. Li, Automated variable weighting in K-Means type clustering, *IEEE Transl. on Pattern analysis and machine intelligence*, vol. 27, issue 5, May 2005.

[7] H. Wang, J. Qi, W. Zheng and M. Wang, Balance K-means Algorithm, *IEEE international conference on Computational Intelligence and Software Engineering*, Dec. 11-13, 2009, 1-3.