International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 33 • 2017

Efficient Clustering of XML Documents using Content and Structural Similarities

Sujitha P[a] and Vijaya M.S[b]

[a]*Research Scholar, PSGR Krishnammal College for Women Coimbatore, India*

E-mail: sssujipsg@gmail.com

[a]*Associate Professor, PSGR Krishnammal College for Women Coimbatore, India*

E-mail: msvijaya@psgrkc.ac.in

Abstract : XML is exponentially becoming the standard for knowledge representation, exchange and retrieval. As XML data is more copious, its heterogeneity and structural irregularity limit the speed of XML based search engines in information retrieval. Data mining techniques such as clustering can enhance XML document storage and information retrieval by grouping according to their similarity. This paper presents an adequate methodology to cluster the XML documents using structure and content similarities. Structure and content similarity features are extracted and clustering of XML documents is performed based on similarity measures. Clustering algorithms such as Affinity Propagation, DBScan, and Hierarchical clustering are used for clustering the XML documents. Affinity propagation algorithm out performs well when compared with other clustering techniques. The performance of these algorithms are measured with various evaluation metrics. Experimental results are based on similarity measures show effectiveness of proposed method, when it is applied on XML documents.

Keywords : *Heterogenous XML data; Similarity measures; XML document clustering.*

1. INTRODUCTION

Web is one of the most valuable resource for information retrieval and knowledge discovery. The advancement of the data usage in web requires automatic tools to allow the transformation of large amounts of data into information and knowledge intelligently. XML has gained popularity for information representation, exchange and retrieval. XML documents are semi-structured and it contains structure elements and content data.

The digital libraries, online news feeds and weblogs are stored as XML documents. The exponential growing of XML data accessible on the web has raised the need of developing clustering techniques for XML documents. As the volume of documents grow fast and enormous, it decreases the speed of search engines in information retrieval. These limitations can be overcome by data mining techniques become necessary to facilitate the organization of the documents for browsing. XML clustering is a task which can be applied to organize the massive amounts of XML documents into groups.

XML clustering is grouping the similar data contained in heterogeneous environment. The clustering task of data mining can be used to identify the structure and content similarities among XML documents. Clustering the XML documents will increase the speed of XML based search engines by retrieving the relevant concern of data.

Various researches are being carried out for making XML document clustering better. A brief report is presented in this section about the methodologies used in XML document clustering in the recent years.

Alishahi et al., [1] proposed an improved adaptation of the algorithm called XCLS+ (XML Clustering by Level Structure) for clustering both homogeneous and heterogeneous XML documents. In the previous XCLS algorithm common elements of the two objects could not be identified and also the real similarity cannot be obtained. The new XCLS+ algorithm uses a new method for matching the common elements between two objects. XCLS+ algorithm finds all common elements in one iteration. Comparative results shown that XCLS+ performed better than XCLS.

Marry Posonia et al., [3] proposed a feature clustering mechanism to find the pattern match with the number of relevant data present in the database. XML benchmark dataset was taken for text classification that stores a set of documents into categories from a predefined set. Feature clustering was a powerful method to reduce the dimensionality of feature vectors for text classification. The authors evaluated that the pattern match approach archives better cluster than use of incremental clustering approach.

Brzezinski et al., [4] proposed approach by implementing an algorithm called PathXP, which mines maximal frequent paths and groups them into pro-files. Books and journal articles XML documents were used. PathXP was found to match, in terms of accuracy, other XML clustering approaches, while requiring less parameterization and providing easily interpretable cluster representatives. This framework consists of four steps: choosing a pattern definition, pattern mining, pattern clustering and document assignment. The frameworks distinguishing feature was the combination of pattern clustering and document-cluster assignment, which allows to group documents according to their characteristic features rather than their direct similarity. The authors concluded use of XPattern XML document clustering by structure archives better result and also it detected outliers.

Muralidhara et al., [6] proposed hybrid approach that discovers the frequent XML documents by association rule mining using the real data of wikipedia. Classical k-means algorithm is employed to cluster the XML documents. Association rule based mining discovers the temporal associations among XML documents. Finding the properties for set of similar documents was better idea rather than to find the property of a single document. Hence, the key contribution of the work was to find the meaningful clustered based associations by association rule based clustering.

Samaneh et al., [7] proposed a method which makes use of structural elements to create the document feature vector for classification with INEX dataset. The document structure was represented by the tags of the XML document. The SVM algorithm was used for learning and classification. These documents were strongly structured and contain the elements like tables and schemas.

From the above study it is observed that many research works are carried out using patterns-based XML clustering based on content, which does not improve the relevant information retrieval. Some other works are aimed at identifying salient data using hybrid approach and clustering of XML documents using K-means algorithm and also documents are clustered using XCLS+ algorithm with structural elements. The techniques considered in existing work to build either content similarity or structure similarity is used for document classification and clustering to improve the information retrieval, speed and accuracy of XML based search engines. The above factor emphasize the need of research in clustering XML documents that leaves behind a challenging task of identifying similarities of both content and structure of XML documents. Hence, the content and structure similarity measures can be used to improve the XML document handling and relevant information

retrieval in XML search engines. This motivated to propose the research work of extracting structure and content similarities and implementing clustering task such as affinity propagation, agglomeration and dbscan. The objectives of this research work is to cluster the XML documents effectively by extracting both structure and content similarities.

2. PROPOSED WORK

Clustering of XML documents are built using similarity measures with three different techniques such as affinity propagation, hierarchical clustering and DBSCAN. The proposed work include different phases such as data collection, tree construction and segmentation, pre-processing, extraction of structure and content similarities, similarity matrix generation and clustering. Heterogeneous XML documents are collected and stored in text file for further processing in data collection phase. XML documents are transformed into tree format and segmented using xpath. In pre-processing stage contents are processed to remove stop words and stem words to extract similarities. Structure and content similarities are extracted and finally clustering of XML documents is performed based on similarity measure. The architecture of proposed model is shown in Fig. 1

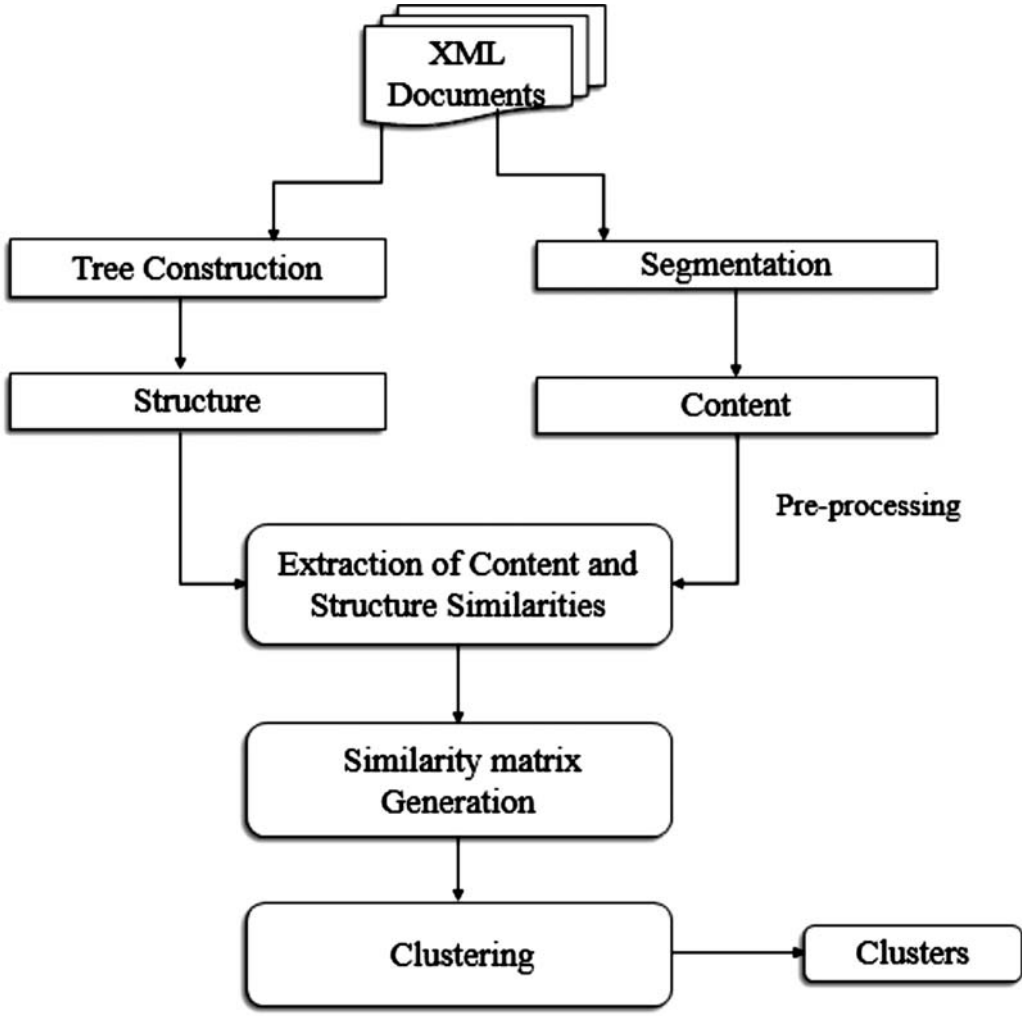

Figure 1: Frame work of the proposed work

2.1. Data Collection

Data are collected from distinct websites. Totally 85 XML documents are collected from websites related to eBay, yahoo, consumer database, food hygiene, road works and medline datasets from cs.washington.edu website. Food hygiene and road works are collected from data.gov.uk website. Medline datasets is collected from Medline plus website.

3.2. Tree Construction and Segmentation

XML documents hold a hierarchical structure and can conceptually be translated as a tree structure, called an XML tree. XML documents must enclose a root element (parent of all other elements). All elements in an XML documents can hold sub elements, text and attributes. A well-formed XML document shows a tree structure and processed by XML parser. Tree represented by XML documents starts at the root element and branches to the lowest level of elements.

Tree is constructed for 85 XML documents using DOM (Document object model), a tree based model and SAX (Simple API for XML Parsing), an event based model. SAX parser reads an XML document and generates events as it finds elements and data in the document. There are events for document start, document end, element start-tags, element end-tags, attributes, text context, entities, processing instructions, comments and others. Click-event on a particular node displays only the sub-nodes rather than loading all the nodes.

DOM parser loads the XML document, builds an object model in the form of a tree comprised of nodes. The DOM API is used for traversing the tree elements and nodes. All nodes are loaded and tree model is created using DOM. DOM parsers use a SAX parser to create the document tree.

For representing XML documents as tree DOM and SAX parsers are used. The result of tree construction is visualized using SAX tree validator. Tree construction for a sample XML document is illustrated in Fig 2

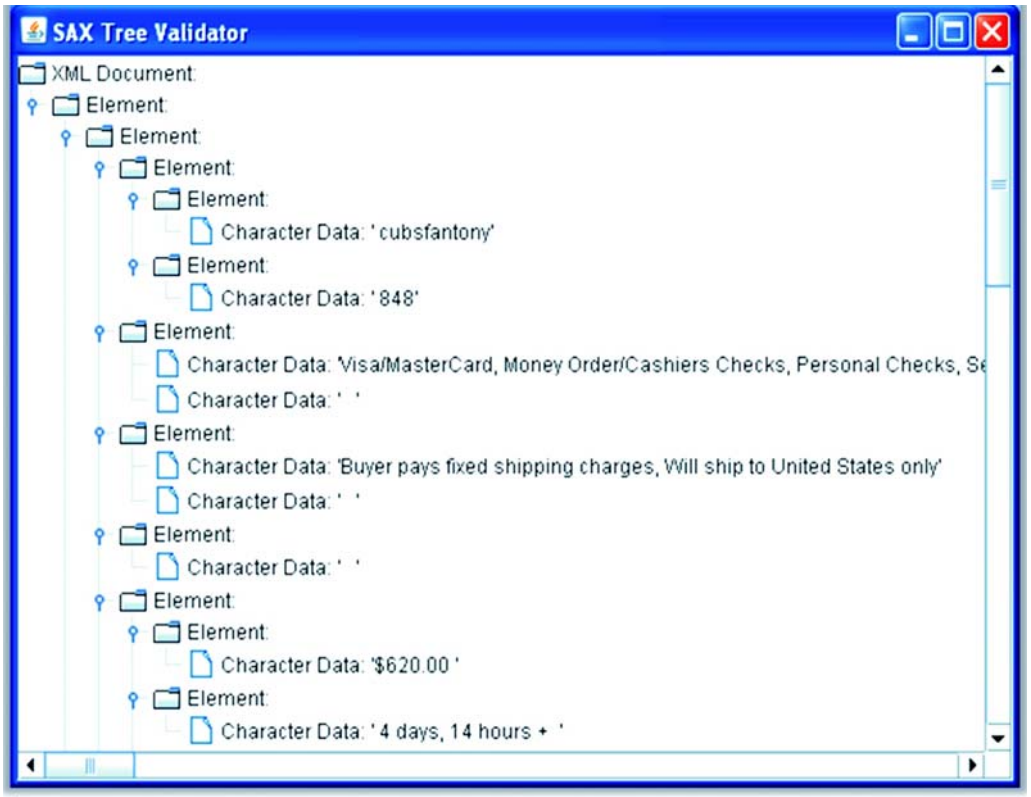

Figure 2: Tree representation of an XML document

Xpath (XML Path Language) is a query language for selecting nodes from XML documents. Xpath is based on tree representation of XML documents and provides the ability to navigate around the tree. Path expression is used to navigate in XML documents. Content and structure of XML documents are segmented using Xpath.

Elements of XML documents are extracted using path expression by selecting a node in XML tree. So the elements of all documents are stored separately in text file and content (text) of all documents are stored separately in text file. Content and structure of XML documents are segmented for extracting similarity. A sample XML document segmentation is illustrated in Table 1

Table 1
Segemented Structure And Contents

Structure segmentation	*Content segmentation*
<root> <listing> <seller_info> <seller_name> </seller_name> <seller_rating> </seller_rating> </seller_info>	Ubsfantony 848 Visa/MasterCard, Money Order/Cashiers Checks, Personal

2.3. Extraction of Structure Similarity

Structure (element) similarity extraction is based on XML tree and structure segmentation. Structure similarity is extracted by two methods:

1. Level similarity
2. Structure similarity

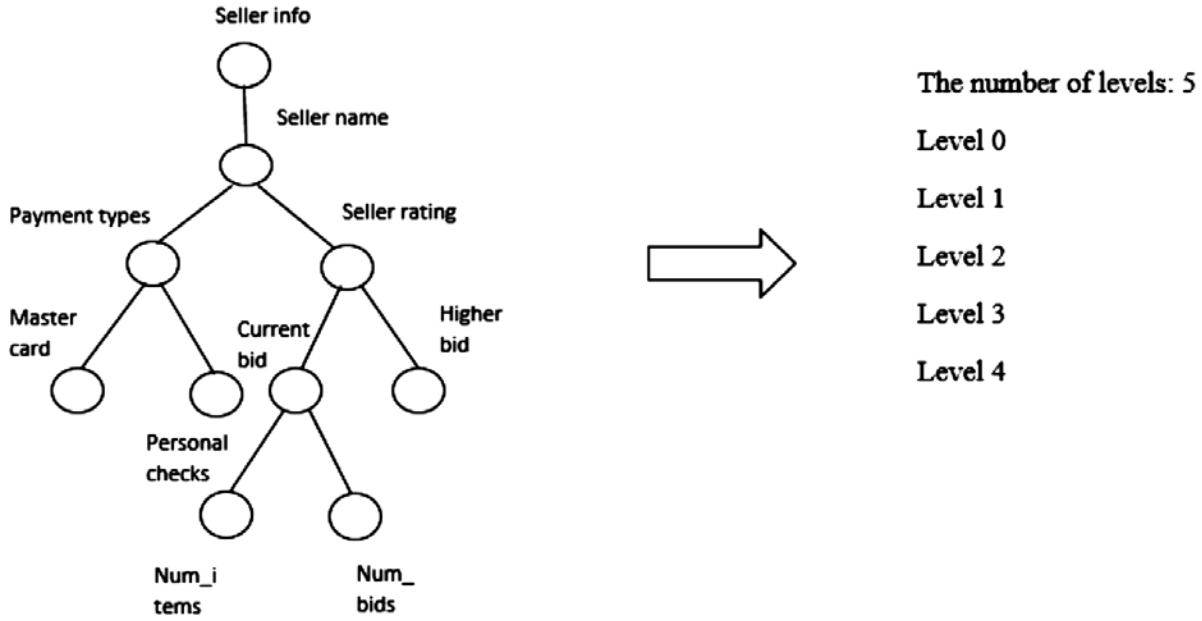

Figure 3: Level Structure of eBay document

Measuring Level Similarity : Level structure uses the elements or tags of the XML documents and ignores their contents and attributes. Level structure is acquired from tree representation. Level of XML document in tree varies from document to document. Computing similarity is a considerable point in clustering. Every document contains parent, child and leaf node. Level similarity is measured based on common nodes in different levels of object. Similarity between two documents is measured based on common nodes of each level of two object. By comparing levels between each documents and similarity is measured. Extraction of level similarity generates similarity matrix in excel file. Level structure of XML document and level structure re-labelled nodes to integer is shown in Fig. 3.

Measuring Structure Similarity : Structure similarity is measured using parent node of XML documents. Structure of XML documents are extracted based on the representation of tree. Structure or tag of XML documents is stored separately in text file. Text file contains parent node, child node and leaf node. First identified the parent node of collected documents and stored in separate file. By using the parent node of each document is compared with all other document. Based on comparison similarity is measured. Similarity matrix is calculated for all documents in excel file. Procedure for comparing document is,

```
for(int t = 0; t < vPaths.size(); t ++)
MyDocument tdoc = new MyDocument((String)vPaths.
get(t));
documents.addDoc(tdoc);
```

2.4. Extraction of Content Similarity

Content of XML documents are extracted and stored separately in text file. Extracted content file contains words, phrases and symbols, which are not required for similarity extraction. Pre-processing includes steps such as tokenization, stop word removal and stem word removal.

Tokenization : The first step in most retrieval systems is to identify keywords for representing documents, a preprocessing step often called tokenization. Tokenization is the process of breaking a stream of content into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing in XML content mining.

Stop word removal: A stop list is a set of words that are deemed irrelevant. For example, a, the, of, for, with and so on are stop words. Even though they may appear frequently. Stop lists may vary per document set. For example, database systems could be an important keyword in a newspaper. However, it may be considered as a stop word in a set of research papers presented in a database systems conference. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on. In this case, stop words can cause problems when searching for phrases that include them, particularly in names such as “The Who”, “The” or “Take That”. To avoid indexing useless words, a data retrieval system often associates a stop list with a set of documents. Search engines remove the most common words from a content file in order to improve performance. stopwordFilter.check(word) expression is used for removing stop words

Stem word removal : A group of different words may share the same word stem. A text retrieval system needs to identify groups of words where the words in a group are small syntactic variants of one another and collect only the common word stem per group. For example, the group of words drug, drugged and drugs, share a common word stem, drug and be viewed as different occurrences of the same word. Stemming is the process for reducing inflected words to their word stem. After filtering out the stop words, can stem the remaining words. List of rules is stored which provides a path for the algorithm, given an input word form, to find its root form. After removing stop words and stem words, the content file is meaningful for extracting similarity. 50 keywords are selected manually and occurrence of selected keywords in every XML document is computed as term count.

Keywords and Term count : From pre-processed content file 50 keywords are manually selected. Knowing the number of words or characters in a document is important to provide some relevance to the document. Searching results of search engine is related to better keywords of XML documents. For this process term count is calculated, which in turn helps in the computation of *tf-idf* weights. For each document, selected keywords are counted, which results in term count value. By using term count values, term frequency (*tf*) and inverse document frequency (*idf*) weights are computed.

Tf-Idf and Cosine similarity : Term frequency-Inverse document frequency(*tf-idf*), is numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and content mining. Weight is calculated for every keywords in documents using term count. Cosine similarity is measured using *tf-idf* weights.

Term frequency(TF): Term frequency is also known as TF measures. It counts the number of times each term occurs in each document and sums them all together. The number of times a term occurs in a document is called its term frequency. For selected 50 terms, term frequency is computed for every XML document. The following formula is used to calculate TF

$$\text{TF}(f) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

Inverse document frequency(IDF): The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient. The following formula is used to calculate IDF

$$\text{IDF}(t) = \log\_e \frac{\text{Total number of documents}}{\text{Number of document with term } t}$$

Tf-Idf : A high weight in *tf–idf* is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents. The weights tend to filter out common terms. Since the ratio of idf log function is always greater than or equal to 1, the value of idf (and *tf–idf*) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and *tf–idf* closer to 0. Tf-*idf* weights is generated for all XML documents with pairwise distance (matrix form). The tf-idf weight of a term is the product of its tf weight and its idf weight. Then tf-idf is calculated as

$$\text{T}fidf(t, d, \text{D}) = tf(t, \text{d}).\ idf(t, \text{D})$$

Cosine similarity : Cosine Similarity is used to calculate the similarity between different documents. Cosine similarity between two documents is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude, a comparison between documents on a normalized space because not taking into the consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the documents. Using the formula given below the cosine similarity between two documents is computed.

$$\text{Cosine Similarity}(d1, d2) = \text{Dot product}(d1, d2) / ||d1|| * ||d2||$$
$$||d1|| = \text{Square root}(d1[0]2 + d1[1]2 + ... + d1[n]2)$$
$$||d2|| = \text{Square root}(d2[0]2 + d2[1]2 + ... + d2[n]2)$$

Documents	D1	D2	D3	D4	D5	D6	D7
D1	1	0.4932	0.0789	0.2367	0.7288	0.3366	0.7828
D2	0.4932	1	0.654	0.7787	1	0.5778	0.7745
D3	0.0789	0.654	1	1	0.864	0.7441	1
D4	0.2367	0.7787	1	1	0.6454	0.8997	0.8687
D5	0.7288	1	0.8645	0.6454	1	1	1
D6	0.3366	0.5778	0.7441	0.8997	1	1	0.6332
D7	0.7828	0.7745	1	0.8687	1	0.6332	1

Figure 4: Cosine similarity matrix for XML documents

3. CLUSTERING ALGORITHMS

This research focuses on clustering XML document by structure and content using a hierarchical distance-based XML clustering technique. XML document clustering differs from clustering any other type of data set because of the hierarchical structure of XML. Three clustering algorithms, dbscan, agglomeration and affinity propagation were used for clustering the heterogeneous XML documents.

3.1. DBSCAN technique

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density- based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise. It defines a cluster as a maximal set of density-connected points. DBSCAN requires two parameters: ε (eps) and the minimum number of points required to form a dense region (minPts). It starts with an arbitrary starting point that has not been visited. This point's ε-neighborhood is retrieved and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. If a point is found to be a dense part of a cluster, its ε-neighborhood is also part of that cluster. Hence, all points that are found within the ε-neighborhood are added, as is their own ε-neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

3.2. Agglomeration technique

A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed. The agglomerative approach, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups

are merged into one the topmost level of the hierarchy or until a termination condition holds. This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this method. They differ only in their definition of inter-cluster similarity.

3.3. Affinity propagation

Affinity Propagation creates clusters by sending messages between pairs of samples until convergence. Affinity Propagation chooses the number of clusters based on the data provided. For this purpose, the two important parameters are the preference, which controls how many exemplars are used, and the damping factor. The messages sent between points belong to one of two categories. The first is the responsibility $r(i, k)$, which is the accumulated evidence that sample k should be the exemplar for sample i. The second is the availability $a(i, k)$ which is the accumulated evidence that sample i should choose sample k to be its exemplar, and considers the values for all other samples that k should be an exemplar. Exemplars are chosen by samples if they are similar enough to many samples and chosen by many samples to be representative of themselves. More formally, the responsibility of a sample k to be the exemplar of sample i is given by:

$$r(i, k) \leftarrow s(i, k) - \max_{k' \neq k} \{a(i, k') + s(i, k')\}$$

Where $s(i, k)$ is the similarity between samples i and k. The availability of sample k to be the exemplar of sample i is given by:

$$a(i, k) \leftarrow \min(0, r(k, k) + \sum_{i' \notin \{i, k\}} \max\{0, (i', k))\} \text{ for } i \neq k \text{ and}$$

$$a(k, k) \leftarrow \sum_{i' \neq k} \max\{0,\ r(i', k))$$

Values of r and a are set to zero, and the calculation of each iterates until convergence. The affinity propagation algorithm is simple to implement and customize. It is also computationally efficient, scaling linearly in the number of similarities or quadratically in the number of data points if all possible pairwise similarities are used. Computing pairwise similarities typically takes more computation than does clustering them.

The clustering models can be evaluated using various criteria such as homogeneity, completeness, *v*-measure, adjusted rand index, adjusted mutual rand index and silhouette coefficient.

Homogeneity: Homogeneity is satisfied by clustering results if all of its clusters contain only data points which are members of single class. Homogeneity is calculated using the following formula.

$$H = 1 - \frac{H(C|K)}{H(C')}$$

Completeness : Completeness is satisfied by clustering results if all the data points that are members of a given class are elements of the same clusters. Completeness is calculated using the following formula.

$$C = 1 - \frac{H(K|C)}{H(K)}$$

V- measure : V-measure is the harmonic mean between homogeneity and completeness. V measure is calculated using the following formula.

$$V = 2 * (\text{homogeneity} * \text{completeness}) / (\text{homogeneity} + \text{completeness})$$

Adjusted Rand Index : The adjusted Rand index is a function that measures the similarity of the two assignments, ignoring permutations and with chance normalization. ARI is calculated using the following formula.

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

Adjusted Mutual Rand Index : Mutual Information is a function that measures the agreement of the two assignments, ignoring permutations. Two different normalized versions of this measure are available, Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI). NMI is often used in the literature while AMI was proposed more recently and is normalized against chance. AMI is calculated using the following formula.

$$AMI = \frac{MI - E[MI]}{\max(H(U), -H(V)) - E[MI]}$$

Silhouette Coefficient : The Silhouette coefficient is an evaluation, where a higher Silhouette coefficient score relates to a model with better defined clusters. The Silhouette coefficient is defined for each sample and is composed of two scores:

1. ***a*:** The mean distance between a sample and all other points in the same class.
2. ***b*:** The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette coefficient S for a single sample is then given as:

$$S = \frac{b - a}{\max(a, b)}$$

4. EXPERIMENTS AND RESULTS

Various experiments have been carried out by implementing clustering algorithms namely affinity propagation, agglomeration and dbscan. These techniques are implemented for clustering XML documents using structural features, content features, both content and structure similarity features. Totally 85 XML documents are collected from different websites related to ebay, consumer database, food hygiene, road works and medline data sources. Tree construction is done based on java routines of DOM and SAX Parsers. From the tree, Xpath is used for segmentation. Structure similarity is extracted using level and structure similarity measures based on tree representation. Contents are pre-processed using stop word and stem word removal. Content similarity between the documents is extracted using tf-idf and cosine similarity measures.

Extraction of content and structure similarities generates similarity matrix. The performance of clustering are evaluated in terms of Homogeneity, Completeness, V- measure, adjusted rand index, mutual rand index and silhouette coefficient. The results of the experiments are summarized in Table 3 to V and the comparative performance of clustering is shown in Fig. 1 to Fig. 3.

4.1. Document clustering using Affinity propagation

In this experiment, XML documents are clustered using affinity propagation algorithm. The results of affinity propagation algorithm, based on similarity matrices: (*i*) content similarity matrix (*ii*) structure similarity matrix (*iii*) merged content and structure similarity matrix is analyzed and the comparative results with respect to various metrics are given in Table 2

Table 2
Comparative Results of Affinity Propagation Algorithm

Affinity propagation Performance	*Content*	*Structure*	*Content and structure*
Homogenity	0.959	1	1
Completeness	0.959	0.709	1
V-measure	0.959	0.830	1
Adjusted rand index	0.970	0.729	1
Adjusted mutual index	0.958	0.701	1
Silhouette coefficient	0.718	0.618	0.956

4.2. Document clustering using Agglomeration algorithm

In this clustering experiment, XML documents are clustered using agglomerative algorithm. The results of agglomeration algorithm, based on similarity matrices: (*i*) content similarity matrix (*ii*) structure similarity matrix (*iii*) merged content and structure similarity matrix is analyzed and the comparative results with respect to various metrics are given in Table 3

Table 3
Comparative Results Of Agglomerative Algorithm

Agglomeration Performance	*Content*	*Structure*	*Content and structure*
Homogenity	0.789	0.369	0.437
Completeness	0.539	0.773	0.798
V-measure	0.640	0.514	0.565
Adjusted rand index	0.601	0.372	0.425
Adjusted mutual index	0.522	0.390	0.432
Silhouette coefficient	0.604	0.528	0.538

4.3. Document clustering using DBScan algorithm

This experiment is performed to cluster XML documents using DBSCAN algorithm. The results of DBSCAN algorithm, based on similarity matrices: (*i*) content similarity matrix (*ii*) structure similarity matrix (*iii*) merged content and structure similarity matrix is analyzed and the comparative results with respect to various metrics are given in Table 4

Table 4
Comparative Results of Dbscan Algorithm

DBSCAN Performance	*Content*	*Structure*	*Content and structure*
Homogenity	0.660	0.159	0.001
Completeness	0.531	0.431	0.007
V-measure	0.589	0.232	0.002
Adjusted rand index	0.448	0.078	0
Adjusted mutual index	0.520	0.151	0
Silhouette coefficient	0.298	0.177	0.211

4.4. Comparative analysis

The results of all the three algorithms: affinity propagation, agglomeration and DBSCAN, based on (*i*) Content similarity (*ii*) Structure similarity (*iii*) Content and Structure similarity are compared. The clustering algorithms are implemented and performance metrics are measured. Comparative results of clustering techniques with various performance measures are shown in Table 5 and illustrated in Fig. 1 to Fig 3.

Table 5
Performance of Clustering Algorithms

	Content			*Structure*			*Content and Structure*		
Measure	*DB-SCAN*	*Hierachical clustering*	*Affinity Propagation*	*DB-SCAN*	*Hierachical clustering*	*Affinity Propagation*	*DB-SCAN*	*Hierachical clustering*	*Affinity Propagation*
N	3	1	3	1	5	5	1	3	3
H	0.66	0.789	0.959	0.159	0.369	1	0.001	0.437	1
C	0.531	0.539	0.959	0.431	0.733	0.709	0.007	0.798	1
V-Measure	0.589	0.640	0.959	0.232	0.514	0.830	0.002	0.565	1
ARI	0.448	0.601	0.970	0.078	0.372	0.729	0	0.425	1
AMI	0.520	0.522	0.958	0.151	0.390	0.701	0	0.432	1
SC	0.298	0.604	0.718	0.177	0.528	0.618	0.211	0.538	0.956

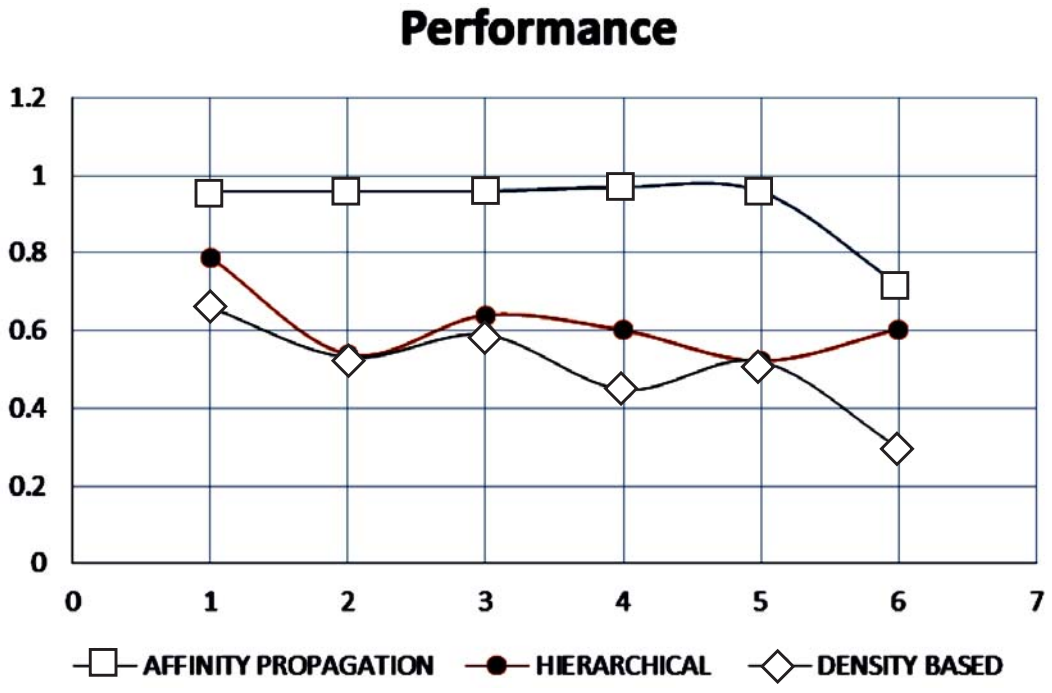

Figure 6: Comparative results of content similarity based clustering

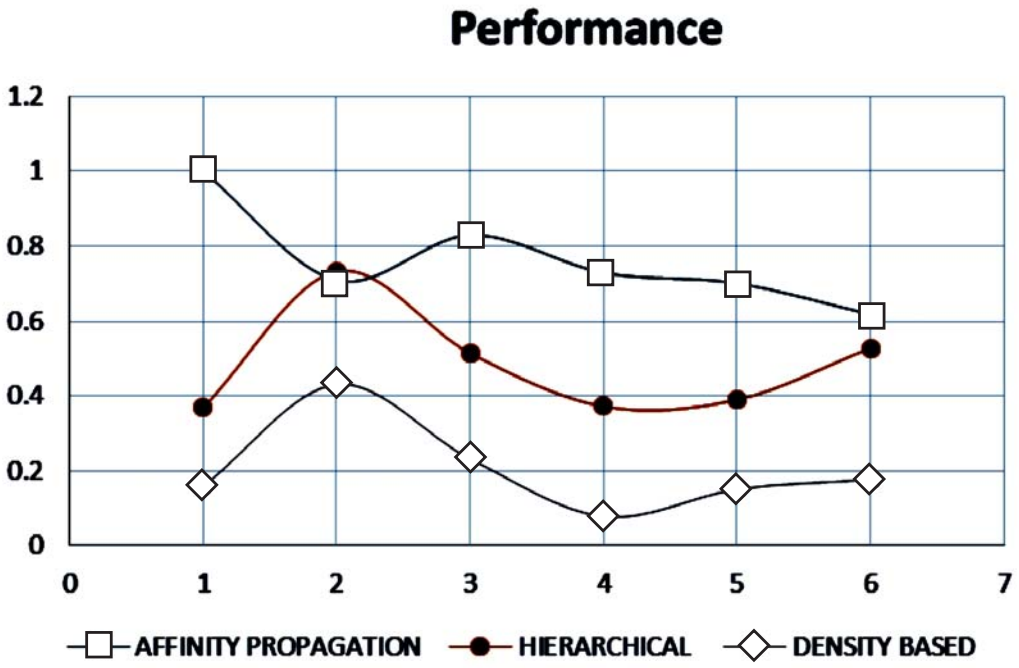

Figure 6: Comparative results of structure similarity based clustering

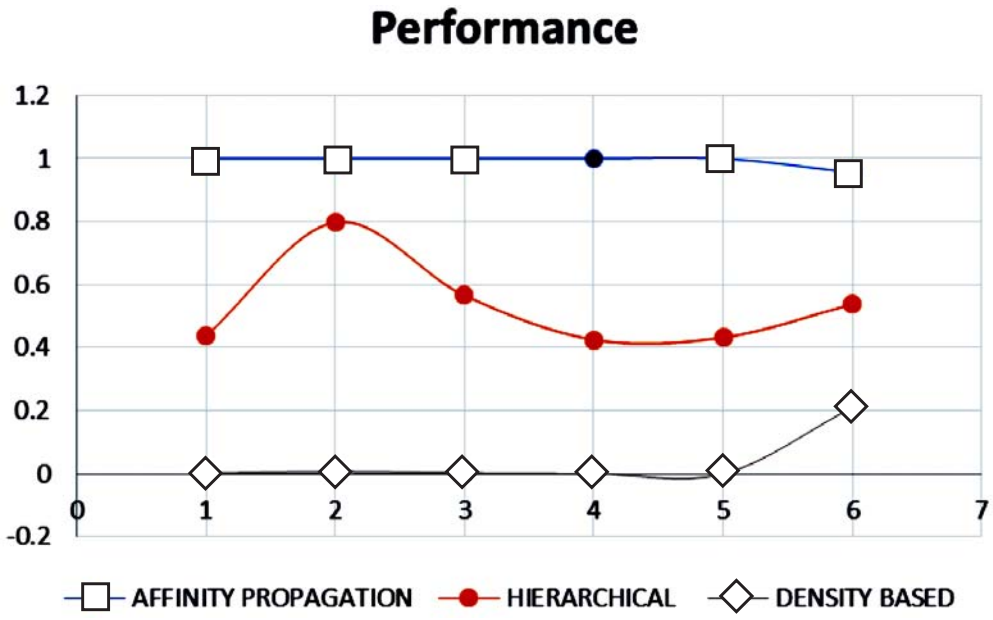

Figure 7: Comparative results of both content and structure similarity based clustering

From the above comparative results it is observed that fusing of content and structure similarity matrix performs better, when compared to algorithm which takes only content or structure similarity features. The results of affinity propagation based clustering using both content and structure matrix is higher than other clustering techniques such as DBSCAN and agglomeration, which proves that affinity propagation algorithm clusters XML documents more effectively.

5. CONCLUSION

This paper demonstrates the modeling of XML document clustering using content and structure similarity features. XML documents are collected from six different domains and similarity features are extracted in the form of similarity matrix. The proposed model is implemented using affinity propagation, agglomeration and DBScan algorithm. The comparative results indicate that affinity propagation yield better performance by taking both content and structure features, when compared to other clustering algorithms with respect to evaluation metrics. The proposed model will be highly vital in search engines that search immense resources of the web.

REFERENCES

[1] Mohamad Alishahi, Mahmoud Naghibzadeh, Baharak Shakeri Aski (2010), "*Tag Name Structure-based Clustering of XML Documents*", International Journal of Computer and Electrical Engineering, Vol. 2, No. 1, 1793-8163.

[2] Somayeh Ghazanfari, Hassan Naderi (2015), "A Hybrid Algorithm for Improvement of XML Documents Clustering", International Journal of Computer Science and Business Informatics, ISSN: 1694-2108,Vol. 15, No. 3.

[3] Mary Posonia A , Jyothi V.L (2013), "Context-based Classification of XML Documents in Feature Clustering", Indian Journal of Science and Technology, Vol 7(9), 1355–1358.

[4] Maciej Piernik, Dariusz Brzezinski (2016), "Clustering XML documents by patterns", Knowl Inf. Syst. 46:185–212.

[5] Shashirekha H.L, Vanishree K.S, Sumangala N (2011), "Content and structure based classification of XML documents", International Journal of Machine Intelligence, Volume 3, Issue 4, pp-376-380.

[6] Muralidhara A , Pattabiramanb V (2015), "An Efficient Association Rule Based Clustering of XML Documents", 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15).

[7] Samaneh Chagheri, Sylvie Calabretto (2012), "Document Classification Combining Structure and Content", International Journal of Machine Intelligence, Volume 3, Issue 4, pp-566-580.

[8] Vidhya S, Asir Antony Gnana Singh D, Jebamalar E (2015), "Feature Extraction for Document Classification", International Journal of Innovative Research in Science, Engineering and Technology, Vol. 4, Special Issue 6.

[9] Nayak G.R (2008), "Fast and effective clustering of XML data using structural information", Knowl. Inf. Syst. 14(2), pp-197-215.

[10] Nayak R (2008), "XML Data Mining: Process and Applications", in Song, Min and Wu, Yi-Fang, Eds, Idea Group Inc. / IGI Global.

[11] Nayak R, Tran T (2007), "A Progressive Clustering Algorithm to Group the XML Data by Structural and Semantic Similarity", IJPRAI 21(4), pp-723-743.

[12] Denoyer L, Gallinari P (2007), "Report on the XML mining track at INEX 2005 and INEX 2006: categorization and clustering of XML documents", SIGIR Forum 41(1), pp.79-90.

[13] Abiteboul S, Buneman P, Suciu D (2000), "Data on the Web: From Relations to Semistructured Data and XML", Morgan Kaumann, CA.

[14] Flesca S, Manco G, Masciari E, Pontieri L, Pugliese A (2005), "Fast detection of XML structural similarities", IEEE Trans. Knowl. Data Engin.7(2), pp.160–175.

[15] Dalamagas T, Cheng T, Winkel K, Sellis T.K (2006), "A methodology for clustering XML documents by structure", Inf. Syst. 31(3), pp.187-228.