Evaluating the Performance of XML Document Clustering by Structure Only

Tien Tran and Richi Nayak

Faculty of Information Technology, Queensland University of Technology
Brisbane, Australia
t4.tran@qut.edu.au, r.nayak@qut.edu.au

Abstract. This paper reports the results and experiments performed on the INEX 2006 Document Mining Challenge Corpus with the PCXSS clustering method. The PCXSS method is a progressive clustering method that computes the similarity between a new XML document and existing clusters by considering the structures within documents. We conducted the clustering task on the INEX and Wikipedia data sets.

Keywords: Clustering, XML document mining, Structural mining, INEX, XML, Structural similarity.

1 Introduction

With the emergence of XML standard, XML documents are widely accepted by many industries such as business, education, entertainment and government [2]. With the continuous growth of XML data, many issues concerning with the management of large XML data sources have also arisen. For efficient data management and retrieval, a possible solution is to group XML documents based on their structure and content. The clustering of XML documents facilitates a number of applications such as improved information retrieval, document classification analysis, structure summary, improved query processing [1, 8] and so on.

The clustering process categorizes the XML data based on a similarity measure without the prior knowledge on the taxonomy [4]. Clustering techniques have frequently been used to group similar database objects and text data. However, clustering of XML documents is more challenging because a XML document has a hierarchical structure and there exist relationships between element objects at various levels.

We propose to use the PCXSS algorithm [7] that is developed to deal with the heterogeneous XML schemas to cluster the INEX 2006 Document Mining Challenge Corpuses [3]. The PCXSS (*P*rogressively *C*lustering *X*ML by *S*tructural *S*imilarity) algorithm employs a global criterion function *CPSim* (common path coefficient) that measures the similarity between an XML document and existing clusters of XML documents, instead of computing the pair-wise similarity between two data objects. The PCXSS, originally developed for the purpose of clustering of heterogeneous XML schemas, has been modified and applied to cluster the INEX 2006 XML documents by considering only the structure of XML documents.

Our philosophy is based on the common usage of XML that is, XML is mainly used for representing the text data in the structured format. Based on this, we assume that a clustering algorithm should group the documents that share a similar structure. For example, documents from the publication domain would have different structure from the documents from the movie domain. Our initial work has shown that the structure of the documents plays a prominent role in grouping the similar XML documents [6]. The semantic difference in tag names can be avoided during the clustering process. In these experiments, we also have not included the instances. The inclusion of instances (the contents within the tag) incurs an additional computing cost. We would like to test the hypothesis such as how important is the structure of the XML documents when categories of documents are mainly based on theme such as the INEX 2006 Document Mining Challenge Corpuses.

The next section gives an overview of the PCXSS methodology. Interested readers can read [7] for a more detailed discussion on this methodology. Phases of the PCXSS method are then described further in Sections 3 and 4. Section 5 reports the results, experiments and data analysis performed on INEX and Wikipedia data sets. The paper is then concluded and further work is outlined in Section 6.

2 The PCXSS Method: Overview

Fig. 1 illustrates a high level view of the PCXSS method. The pre-processing phase decomposes every XML document into the structured path information called node paths. Each path contains the node properties from the root node to the leaf node. The first stage of the clustering phase i.e., ‘structure matching,’ measures the structural similarity between node paths of a XML document and other objects (the existing clusters). This stage determines the similarity between two objects according to the nodes they share common in their paths. The output of the structure matching stage is the common path coefficients (*CPSim*) between the document and all existing clusters. The second stage of the clustering phase groups the XML document into an existing cluster with which it has the maximum *CPSim* or assigns it to a new cluster.

A number of modifications have been made to the PCXSS method in order to experiment with the INEX 2006 corpus. Firstly, the pre-processing phase extracts the structure of every XML documents into X_Paths where only the name of the element is considered. Other information such as data type and constraints are ignored. Secondly, the structure matching of the clustering phase measures the structural similarity between X_Paths of a document and of clusters considering only the exact match between element names. We do not consider the various semantic and syntactic meanings that an element name can have during the structure matching. We have shown elsewhere that semantics of an element name (such as person vs. people) in XML documents do not make any significant contribution when determining similarity between two XML documents [6].

3 PCXSS Phase 1: Pre-processing

All documents in the INEX collection or in the Wikipedia collection conform to only one DTD schema. As a result, we do not perform the pre-processing of element

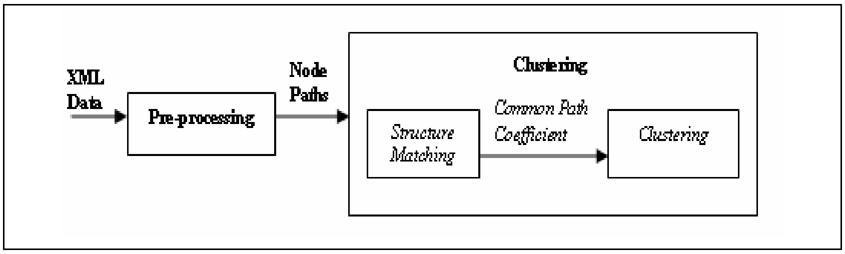


Fig. 1. The PCXSS Methodology

names while inferring the structure of the documents. Only a simple pre-processing step has been applied on the XML documents. An XML document is first parsed and modelled as the labelled tree (Fig. 2). The attribute of an element is modelled exactly the same way as its child elements. The tree is then decomposed into X_Paths to represent the structure of the XML document.

An X_Path is formally defined as an ordered sequence of tags from a root to a leaf node which includes hierarchical structure. An XML document consists of many X_Path sequences and the order of X_Paths is ignored because each X_Path is considered as an individual item in the XML document structure. Moreover, duplicated X_Paths in a document structure are eliminated. After the pre-processing of XML documents, documents are represented as a collection of distinct X_Paths.

4 PCXSS Phase 2: Clustering

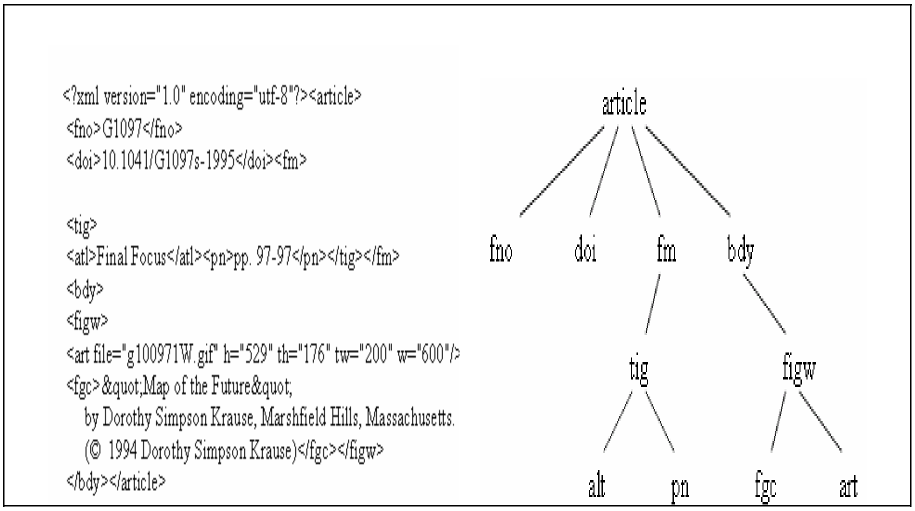
The clustering phase consists of two stages: structure matching and clustering. At structure matching stage, the similarity between a XML document and existing clusters is measured. The output of this stage is a similarity value called *CPSim* (Common Path Similarity) between an XML document and a cluster. *CPSim* is then used in the clustering stage to group the XML document into an existing cluster with which it has the maximum *CPSim,* or assigns it to a new cluster if (1) the clustering number has not yet exceeded and (2) *CPSim* does not exceed the clustering threshold.

4.1 Structure Matching Stage

Each node in a node path of a document is matched with the node in a node path of the clusters, and then aggregated to form the node path (or structure) similarity.

4.1.1 Node Matching

The node matching process measures the similarity between the nodes in node paths by considering the name similarity only. While clustering XML schemas, PCXSS also includes the data type similarity (Tsim) and constraints similarity (Csim). As the INEX 2006 documents follow the same schema, neither semantic nor syntactic similarity computation is needed on the element name matching. Additionally, the exact matching process on element names saves a significant computation effort. Consequently, node matching depends on the exact match of the node names. For example, the last node at level 2 in Fig 2 is ‘bdy’. Consider another tree that contains a node named as ‘body’. If we compare these two trees, these two nodes will not be considered similar; however, they are syntactically similar. In a similar fashion, a node named as ‘person’ in one tree and a node named as ‘people’ in another tree will not be considered similar, although, they are semantically similar. The *NodeSim* value between element names is equal to 1 if they have an identical name else it is assigned with a 0.



```
<?xml version="1.0" encoding="utf-8"?><article>
<fno>G1097</fno>
<doi>10.1041/G1097s-1995</doi><fm>

<tig>
<atl>Final Focus</atl><pn>pp. 97-97</pn></tig></fm>
<bdy>
<figw>
<art file="g100971W.gif" h="529" th="176" tw="200" w="600"/>
<fgc>&quot;Map of the Future&quot;
   by Dorothy Simpson Krause, Marshfield Hills, Massachusetts.
   (© 1994 Dorothy Simpson Krause)</fgc></figw>
</bdy></article>
```

Fig. 2. An XML Document (article) & its Tree Representation

4.1.2 Structure Similarity

The frequency of common nodes appearing in two XML structures is not sufficient to measure the similarity of XML data. XML is different from other web documents such as HTML or text because it contains the hierarchical structure and relationships between elements. The order of where the element resides in the structure is important in determining the structural similarity between the XML document and existing clusters.

The structural similarity between two XML documents is measured by first finding the common nodes between two paths and then finding the common paths between two trees. The structure matching process in PCXSS is advanced by starting at leaf node between two paths to detect more similar elements within structures.

Common nodes finding. The degree of similarity between two node paths, defined as path similarity coefficient (*Psim)*, is measured by considering the common nodes coefficient (*CNC*) between two paths. The *CNC* is the sum of *NodeSim* of the nodes between two paths P_1 and P_2 as shown in Fig. 3. *Psim* of paths, P_1 and P_2 is the maximum similarity of the two *CNC* functions (P_1 to P_2 and P_2 to P_1) with respect to the maximum number of node in both paths, P_1 and P_2, defined as:

$$Psim(P_1, P_2) = \frac{Max(CNC(P_1, P_2), CNC(P_2, P_1)}{Max(|P_1|, |P_2|)} \tag{1}$$

```
Function: $CNC\ (P_1, P_2)$
Sim:= 0; for each $n_i \in P_1$
  while j not end of $P_2$ length
    if (NodeSim($n_i, n_j$)) ==1
        Sim += NodeSim($n_i, n_j$)
        j--
                  break from 'while' loop
    else
       j--
    end if
  end while
end for
return Sim
```

Fig. 3. The CNC function

Fig. 4 shows an example of traversing through the *CNC* function.

Consider two paths: Path1 (1/2/3/4/5/6) and Path2 (1/2/4/5/6). Path1 contains 6 element names that are showed as numbers for convenience. The following steps are iterated when calculating the *CNC* function:

1. Start at the leaf element of both paths (*j*=5, *i*=4). If the NodeSim coefficient of the leaf elements equals to 1 (a match) then increase Sim by NodeSim coefficient and go to step 2 else go to step 3.
2. Move both paths to the next level (*j*--, *i*--) and start element matching at this level. If the NodeSim coefficient of these elements equals to 1 (a match) then increase Sim by NodeSim coefficient and repeat step 2 else move to step 3.
3. Move only Path 1 to the next level (*j*--) then start element matching in the original level of Path 2 (*i*) to the new element of Path 1.

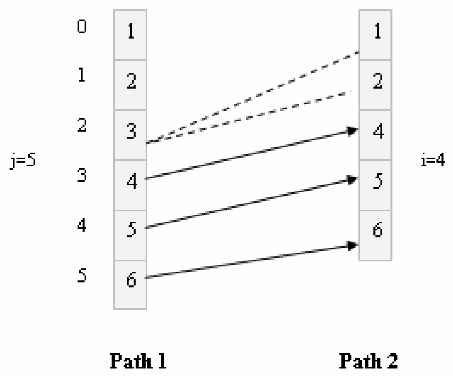


Fig. 4. Example of CNC Matching

The CNC function is not transitional. It means that *CNC(* P_1, P_2) is not equal to *CNC*(P_2,P_1). This is due to the fact that if the leaf element from P_1 can not be found in P_2 then no further matching is required. However, in some cases, one path may be a sub-path of the other. If P_2 is a sub-path of P_1, and if the leaf element can not be found in P_2 then the *CNC(*P_1, P_2*)* returns 0. However *CNC(*P_2, P_1*)* will return a value according to the matching. As a consequence, both *CNC(* P_1, P_2) and *CNC*(P_2,P_1) are computed and the maximum of the two is used to measure the degree of similarity between the two paths.

The *Psim* value is monitored by a path similarity threshold. The threshold determines whether the two node paths are similar. If the *Psim* of two node paths exceeds the path similarity threshold then it is used to determine the structural similarity between the trees and existing clusters.

Common paths finding. PCXSS measures common paths (1) between two trees and (2) between a tree and a cluster.

Tree to Tree Matching: The tree to tree matching is the matching between a new tree and a cluster that contains only one tree. This is defined as:

$$CPSim(Tree_1,Tree_2) = \frac{\sum_{i=1}^{|TPath_1|} \max(\int_{j=1}^{|TPath_2|} Psim(P_i,P_j))}{Max(|TPath_1|,|TPath_2|)} \tag{2}$$

CPSim is the common path similarity between two XML trees. The *CPSim* of trees, $Tree_1$ and $Tree_2$ is the sum of the best path similar coefficient (*Psim*) of paths, P_i and P_j with respect to the maximum number of paths, $|TPath_1|$ and $|TPath_2|$ of trees, $Tree_1$ and $Tree_2$, respectively. The clustering process in PCXSS works on the assumption that only one path from $Tree_1$ matches with one path in $Tree_2$. Thus, it only selects the maximum *Psim* between each pair of paths of $Tree_1$ and $Tree_2$.

Tree to Cluster Matching: The tree to cluster matching is the matching between a new tree and the common paths in a cluster. The common paths are the similar paths that are shared among the trees within the cluster (normally a cluster must contain at least 2 or more trees in the cluster to have the common paths or else the tree to tree matching is required). Initially, the common paths are derived in the tree to tree matching. Then every time a new tree is assigned to the cluster, the similar paths are added to the cluster if paths are not already in the cluster. The tree to cluster matching is defined as:

$$CPSim(Tree,Cluster) = \frac{\sum_{i=1}^{|TPath|} \max(\int_{j=1}^{|CPath|} Psim(P_i,P_j))}{Max(|TPath|)} \tag{3}$$

Similar to the tree to tree matching, *CPSim* between a tree and a cluster is the sum of the best *Psim* of paths, P_i and P_j w. r. t. the number of paths, $|TPath|$ in the *Tree*.

4.2 Clustering Stage

PCXSS is an incremental clustering method. It first starts off with no cluster. When a new tree comes in, it is assigned to a new cluster. When a next tree comes in, *CPSim* is computed between the tree and the existing cluster. If *CPSim* exceeds the clustering threshold and the cluster has the largest *CPSim* with the tree then the tree is assigned to that cluster else it is assigned to a new cluster. The node paths of the tree that are used to compute the *CPSim* are then added to the cluster. The node paths in the cluster are referred to as common paths. The common paths in the cluster are then used to measure the *CPSim* between the cluster and new trees. Since the common paths (instead of all the node paths of the trees held within a cluster) are used to compute *CPSim* with new trees, the computation time reduces significantly. In addition, the cluster contains only the distinct common paths (duplicate paths are removed from the cluster).

5 Experiment and Discussion

Test data. The data used in the experiments are the INEX corpus and Wikipedia corpus from the INEX XML Mining Challenge 2006. Table 1 shows the properties of the experimental corpus.

Table 1. Test Data Sets

Test Data	No. of Classes	No.of XML documents	Size (MB)
INEX	18	6054	259
Wikipedia	60	75047	530

Evaluation methods. For the INEX XML Mining Challenge 2006, the clustering solutions are measured using the f1-measures: micro-average f1 and macro-average f1. These measures are used to evaluate multi-labeled classification (more than 2 labels). To understand how micro-average f1 and macro-average f1 are measured, it is necessary to revisit the precision, recall and f1-measure. For example, for binary classification, the precision (p), recall (r) and f1-measure are defined below, where A stands for the number of positive samples which are predicted as positive, B stands for the number of false negative samples which are predicted as positive, and C stands for the number of false positive samples which are predicted as negative

$$p = A / A + B \qquad r = A / A + C \qquad f1 = 2pr / p + r$$

In a multi-label classification, summing up A, B and C values from all binary classifications respectively and then these values are used to calculate f1 value is called micro-average f1 measure. The macro-average f1 is derived from averaging the f1 values from all binary classifications. Refer to paper [5] for more information on f1 measure for multi-label classification. Micro and macro f1 measures are applied directly on multi-label classification solutions for evaluation. However, to measure the clustering solutions, the clustering solutions are first converted to classification solutions before calculating the micro and macro f1 measures.

Experiments and Results. We submitted 3 results for the INEX test data and 1 result for the Wikipedia test data to the INEX XML Document Mining track 2006. The varied submissions were made due to the results obtained by setting different thresholds during experiments. The results of the clustering solutions performed by PCXSS are shown in Table 2.

Table 2. Results from INEX XML Mining Track 2006

Clustering Threshold	Categories Discovery	Micro F1	Macro F1
0.5 (INEX)	7	0.072944	0.039460
0.7 (INEX)	6	0.088004	0.044307
0.8 (INEX)	7	0.088824	0.044641
0.3 (Wikipedia)	20	0.120460	0.037375

The F1 measure of the clustering solutions obtained with PCXSS on the INEX and Wikipedia test data are low. We examined the results and our experimental setups to find out why the clustering solutions have low performance. Firstly, we used the different thresholds to see whether does the threshold value is a reason for poor performances. The results do not seem to improve much by varying the threshold values.

Secondly, we eliminate attributes of an element to see whether it can improve the clustering solutions. The results in Table 3 show that the removals of the attributes of the elements somewhat improve the clustering results using the same thresholds. However, the results are not yet satisfactory. The reason for the improvement may be that the attributes contained by the Wikipedia and INEX corpuses do not play an importance in understanding the structure of the XML document itself.

Table 3. Clustering Solution without the Attributes

Clustering Threshold	Categories Discovery	Micro F1	Macro F1
0.5 (INEX)	7	0.149186	0.090254
0.7 (INEX)	10	0.150553	0.096187
0.8 (INEX)	10	0.150553	0.096187

The clustering solution using a clustering threshold of 0.8 in table 2 is further analysed. This clustering solution has discovered 7 out of 18 true categories. Table 4 below shows the mapping between 18 clusters that have been generated by PCXSS and the true categories.

Table 4. Mapping of 18 Clusters Discovered by PCXSS to its True Category

18 Clusters Discover by PCXSS	True Category
11	11
10	3
13	17
12	13
15	3
14	3
17	5
16	3
18	14
1	3
3	13
2	3
5	3
4	3
7	3
6	12
9	3
8	5

It shows that the documents in category 3 are widely spread out over the 18 clusters that have been discovered by PCXSS. This can happen due to many reasons. Firstly the XML documents from same category (in this case 3) are not grouped together into one cluster by PCXSS due to the differences in structure and size. The PCXSS algorithm mainly derives the solution based on structure similarity. Moreover, the contents within tags play a significant role in measuring the similarity between documents of the INEX corpus in which documents conform to only one schema. We have ignored the contents within tags in our experiments.

To achieve some success, we tried another modification to the clustering algorithm. The principle is to increase the time performance while maintaining the accuracy. Since the accuracy obtained is not very high, we decided to measure the similarity between a XML document with the first tree in the cluster without using common paths. We only consider the first tree that formed the cluster instead of comparing with all the common paths (of all trees) that are included in the cluster. The results of the INEX corpus are shown in Table 5.

The clustering solutions achieve somewhat better results than those in Table 3. It shows that the clustering on common paths on these kinds of data may not be sufficient enough without including the contents within tags.

PCXSS with the Iteration Phase. The XML documents are grouped according to CPSim between an XML document and existing trees. We do not include further iterations to refine the clustering process. Due to the absence of iteration phase, the clustering process highly depends upon the order of the data set and the clustering threshold. Consider this scenario: the clustering threshold is firmly set as 0.8 in the

Table 5. Results from the Modification of the Clustering Alogrithm in PCXSS

Clustering Threshold	Categories Discovery	Micro F1	Macro F1
0.8 (INEX)	9	0.179525	0.115392
0.9 (INEX)	9	0.174740	0.118604
0.3 (INEX)	6	0.103753	0.051152
0.4 (INEX)	7	0.126618	0.086362
0.4 (Wikipedia)	18	0.121828	0.050716
0.7 (Wikipedia)	10	0.125178	0.033793
0.6 (Wikipedia)	13	0.126537	0.034368

experiment. CPsim between two documents from the same domain is measured as 0.75 while processing. These documents are not considered to be grouped together according to this predefined threshold.

With the current PCXSS clustering process when the desired number of cluster is reached, for the remaining data set, PCXSS will not use the predefined threshold but will find the best similarity from the existing cluster that this remaining data set can be grouped into. This in turn creates a problem at the start when two documents belong to the same group are split into two different clusters. Due to this problem, the experiment is then extended the PCXSS clustering process by including the iteration phase.

The iteration works as follows: after the PCXSS clustering process ends (with the clustering number greater than the predefined one), the iteration phase starts by going through all the existing clusters and merging clusters together if their similarity is greater than the clustering threshold until the desired number of cluster is reached. At the end of the iteration phase if the number of existing clusters is still greater than the desired number of cluster, the iteration phase starts again and the clustering threshold will be decremented by 0.1 until the number of desired cluster is reached. Decrementing the clustering threshold can help to identify two clusters that contain documents from the same domain but have the similarity values lower than the rigid predefined clustering threshold. These two clusters can be merged together.

The experiment uses 0.7 for the clustering threshold and runs the PCXSS with the iteration phase on INEX 6054 test data. The micro and macro F1 of the clustering solution are 0.095 and 0.057 respectively, which are lower than PCXSS with no iteration phase shown in Table 2. We can argue here that the iteration phase proposed in this experiment is not suitable. The reasons are twofold: (1) XML documents from different categories contain many overlapping tags and (2) XML documents from the same category greatly vary in size. For example, XML documents from the 'an' category have an XML document that is 1KB and another document is 276KB in document size, where there is a big gap difference in both tags and content. These two documents surely can never be grouped together if XML documents from different categories have many overlapping tags and content.

During the testing and analysis of the INEX data set, it has been ascertained that even if PCXSS is extended by including contents in the clustering process, the clustering solution will not be that much better if no training or learning is done on the INEX data set because two documents from the same category may contain different content and keywords (where semantic learning of the content or keywords may require). Thus, the INEX test data is more suitable for the classification task rather than for the clustering task.

Based on all the experiments above, it can be ascertained that measuring the structure similarity in the documents derived from the same schema do not show any advantage. The usual methods of matrix computations considering only the contents of documents such as vector space or neural networks may have been more appropriate here. The structure overlapping in the documents of the corpus due to deriving from the same schema and the large variations in the sizes and structures of documents from the same category also downplay the PCXSS clustering process.

6 Conclusions and Future Work

This paper presented the experience of applying the PCXSS clustering method considering only the structure of the XML document to cluster the data of the INEX 2006 document mining challenge. Our aim was to explore whether the structure of the XML documents overplay the instances (contents within tags) of the documents for the clustering task. The experiments show that the structure matching employed by PCXSS alone can not be applied well on the INEX documents especially when the XML documents conform to only one schema. Furthermore, INEX documents are data-centric based where the structure of the document plays a small role in determining the similarity between INEX documents.

The development of the PCXSS clustering algorithm originally meant to cluster the heterogeneous schemas. Use of PCXSS on the XML documents may need a number of extensions such as the learning of instance and data type for a more efficient clustering solution.

For future work, PCXSS will be extended to include the learning of content and to develop a more suitable iteration phase for the clustering process so that it is not highly depended on the predefined threshold. The effect of the size and of the order of the XML documents will also be thoroughly investigated in PCXSS. The PCXSS method will be appropriately modified to reduce those effects.

References

1. Boukottaya, A., Vanoirbeek, C.: Schema matching for transforming structured documents. In: 2005 ACM symposium on Document engineering. Bristol, United Kingdom (November 02-04, 2005)
2. Bray, T., et al.: Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation (2004)
3. Denoyer, L., Gallinari, P.: Report on the XML Mining Track at INEX 2005 and INEX 2006. In: INEX 2006 (2006)
4. Han, J., Kamber, M.: Data Mining. In: Concepts and Techiques, Morgan Kaufmann, Seattle, Washington, USA (2001)
5. Luo, X., Zincir-Heywood, N.: Evaluation of two systems on multi-class multi-label document classification. In: ISMIS05, New York, USA (2005)

6. Nayak, R.: Investigating Semantic Measures in XML Clustering. In: The 2006 IEEE/ACM International Conference on Web Intelligence. Hong Kong (December 2006)
7. Nayak, R., Tran, T.: A Progressive Clustering Algorithm to Group the XML Data by Structural and Semantic Similarity. To be published in International Journal of Pattern Recognition and Artifical Intelligence (Data of Acceptance: 9th October 2006)
8. Nayak, R., Witt, R., Tonev, A.: Data Mining and XML documents. In: The 2002 International Workshop on the Web and Database (WebDB 2002) (June 24-27, 2002)