A Flexible Structured-based Representation for XML Document Mining

Anne-Marie Vercoustre, Mounir Fegas, Saba Gul, Yves Lechevallier

A Flexible Structured-based Representation for XML Document Mining

Anne-Marie Vercoustre, Mounir Fegas, Saba Gul, and Yves Lechevallier

INRIA, Rocquencourt, France
Firstname.Lastname@inria.fr,
WWW home page: http://www.inria.fr/index.en

Abstract. This paper reports on the INRIA group's approach to XML mining while participating in the INEX XML Mining track 2005. We use a flexible representation of XML documents that allows taking into account the structure only or both the structure and content. Our approach consists of representing XML documents by a set of their sub-paths, defined according to some criteria (length, root beginning, leaf ending). By considering those sub-paths as words, we can use standard methods for vocabulary reduction, and simple clustering methods such as K-means that scale well. We actually use an implementation of the clustering algorithm known as *dynamic clouds* that can work with distinct groups of independent variables. This is necessary in our model since embedded sub-paths are not independent.

1 Introduction

XML documents are becoming ubiquitous because of their rich and flexible format that can be used for a variety of applications. Standard methods have been used to classify XML documents, reducing them to their textual parts. These approaches do not take advantage of the structure of XML documents that also carries important information.

Recently much attention has been drawn towards using the structure of XML documents to improve information retrieval, classification and clustering, and more generally information mining. In the last four years, the INEX (Initiative for the Evaluation of XML retrieval) has focused on system performance in retrieving elements of documents rather than full documents and evaluated the benefits for end users. Other works are interested in clustering large collections of documents using representations of documents that involve both the structure and the content of documents, or the structure only.

Approaches for combining structure and text range from adding a flat representation of the structure to the classical vector space model [6] or combining different classifiers for different tags or media [5], to defining a more complex structured vector models [17], possibly involving attributes and links [10].

When using the structure only, the objective is generally to organize large and heterogeneous collections of documents into smaller collections (clusters) that can be stored and searched more effectively. Part of the objective is to

identify substructures that characterize the documents in a cluster and to build a representative of the cluster [7], possibly a schema or a DTD.

Since XML documents are represented as trees, the problem of clustering XML documents is the same as clustering trees. One can identify two main approaches: 1) identify frequent common sub-patterns between trees and group together documents that share the same patterns [16], [19], [7]; 2) define a similarity measure between trees that can be used with a standard clustering algorithm. A possible distance can be calculated by associating a cost function to the edit distance between two trees [8], [14], [2]. However, it is well known that the algorithms for calculating the edit distant have complexity issues. Therefore some models replace the original trees by structural summaries [3] or s-graphs [12] that only retain the intrinsic structure of the tree: for example, reducing a list of elements to a single element, flattening recursive structures, etc.

A common drawback of the two approaches above is that they reduce documents to their intrinsic patterns (sub-patterns, or summaries) and do not take into account an important characteristic of XML documents, - the notion of list of elements and more precisely the number of elements in those lists. While it may be fine for clustering heterogeneous collection, suppressing lists of elements may result in losing document properties that could be interesting for other types of XML mining.

Our idea is therefore to use a document representation that takes into account the frequency of structure within the documents, while not been as costly as the edit distance.

In this paper we propose a generic model for clustering documents that involves either their structure or both their structure and content. We represent documents by flattening their trees into their sets of sub-paths of length between n and m, two values given *a priori*. We retain the frequency of paths and we consider sub-paths as words. Therefore we can apply standard clustering methods usually used for text. When considering document content as well as structure, sub-paths are extended with the individual words of the text contained in the terminal node of each path. For specific values of m and n, our model is equivalent to models that have been proposed before, so we offer a more general framework.

We evaluate our model using the collections proposed in the INEX XML mining track, while being aware that our approach may not be appropriate for some of the proposed collections, in particular those where the order of elements is significant for clustering.

In section 2, we present our document model for clustering and compare it, in section 3, to previous models for specific values of m and n. Section 4 describes our clustering method and some additional feature selection. Section 5 details the evaluation metrics we use, while sections 6 and 7 present our experiments and the results. In section 8 we propose our conclusions.

2 Our Model for Document Representation

XML documents are usually represented as trees where each node corresponds to an XML tag. The hierarchy of the nodes reflects the embedding of the tags, and leaf nodes have associated text. Attributes can be represented the same way as sub-elements, i.e. as additional descendants to the node they are attached to.

```
<article>
  <fm>
    <au>
      <fnm>werner</fnm>
      <snm>buchholz</snm>
      <role>editor</role>
    </au>
    <abs>This department offers an opportunity to
      comment on previously published articles ...
    </abs>
  </fm>
  <bdy>
    <sec sno="01">
      <st>the stored program concept: a reprise</st>
      <p>for historians interested in establishing ... </p>
      <p>when a small number of us under the leadership...</p>
      <bibl>
        <h>additional material on the subject in annal</h>
        <bb>
          <au>n. metropoli</au>
          <au>j. worlton</au>
          <atl>a trilogy of errors in the history of computing</atl>
        </bb>
      </bibl>
  </bdy>
</article>
```

Fig. 1. An example of XML document

Figure 1 gives an example of an XML document, extracted from the IEEE collection, that we will use along the paper.

Our model is based on a tree linearization that represents a document as its set of paths. The precise definition of paths to consider is defined below and correspond, in fact, to a family of possible representations that take into account the structure, the text, or both. By regarding paths as simple words we can use the vector model to represent documents from their structure.

The motivation for a flexible choice of paths or sub-paths in the document is that some analysis or clustering tasks may be interested in the top part of the

tree, the lower parts of the tree, or possibly parts in the middle. An example would be clustering very heterogeneous collections based on the structure, where the partition can be done by looking at the top level elements only. To the contrary, if one wants to cluster documents based mostly on the text, it could be appropriate to add some limited context just above the text.

Before presenting our structured document representations, we introduce some definitions:

Definition 1. *The path of a node n is the sequence of nodes from the root to this node, when traversing the tree from child to child. We note it p(n). It is also called a complete path*[1]*.*

Definition 2. *The length of a path is the number of nodes in the path.*

Definition 3. *A* sub-path s of length l *on a path p is a sequence of l consecutive nodes along the path p. (i.e. a sub-path does not necessarily start at the root). We note $|s|$ the length of the sub-path s.*

Table 1 shows examples of paths and sub-paths of length 3.

To take into account the text of the documents, we introduce "text paths" defined as follow:

Definition 4. *A text path (resp. sub-path) is a path (resp. sub-path) that ends with a word contained in the text associated with the last node in the path (resp. sub-path).*

Paths	Tf
article.bdy.sec	1
article.fm.au	1
bdy.sec.p	2
bdy.bb.au	2
bdy.sec.sno@	1

Table 1. Paths and sub-paths of length 3; the character @ marks an attribute.

Paths	Tf
article.fm.abs."offer"	1
bdy.sec.p."historian"	1
bdy.sec.sno@."01"	1
article.fm.au."werner"	1
bdy.sec."historian"	1

Table 2. Textuel paths of length 4 and 3.

Table 2 shows some text paths or sub-paths of length 4 and 3 corresponding to the example in figure 1. The last two paths are non terminal paths extended with words that are not directly associated with their final node but with one of their descendants.

We can now define a family of representations for a XML document tree d as:

$$R(d) = \sum_i w_i p_i \tag{1}$$

[1] We use the terminology used in [13] for complete paths, root-beginning paths, leaf-ending paths

for all sub-paths p_i in d, where $m \leq |p_i| \leq n, 1 \leq m \leq n$, w_i is the frequency of sub-paths p_i

The actual representations are defined by a few parameters:

- m and n are two *a priori* fixed integers. The value "n" can be replaced with the symbol "*", meaning that the maximum value would be, for each sub-path, the length of its supporting path.
- when the parameter **root** is set on, only the sub-paths starting from the root (root-beginning paths) are generated.
- when the parameter **leaf** is set on, only the sub-paths ending at leaf nodes (leaf-ending path) are generated.
- with the parameter **text** set on, only "text paths" are generated.
- with the parameter **text-and-node** set on, both text sub-paths and node sub-paths are generated.
- with the parameter **attribute** set on, attributes, as well as nodes, are considered for path generation.

By setting different parameter values, we can use a variety of document representations for different clustering tasks. Before presenting our clustering approach, we are going to interpret the models for some specific values of the parameters, and compare them with other existing models.

3 Comparison with other models for structured documents

Our document model integrates various representations that have been proposed in other works:

- Case min=1, max=1, text= true;
  This case corresponds to representing a document by its text only (standard vector model)
- Case min=1, max=1, text= false, [attribute= (false | true)];
  Corresponds to representing a document by the list of its tags. This is the model used, with or without attributes, in [6], for the case "Tag feature only", based on the vector model. It is also used in [7] where both elements and attribute names are considered. Moreover if node-and-text is set to true, we get the "Tag and text features" used in [6].
- Case min=1, max=*, root=true, leaf= true, [text= (false | true)];
  In this case XML documents are represented by the set of their paths from the root to the leaves. In [18] they use a bitmap matrix where lines represent the documents and columns represent the different terminal paths in the collection. The frequency is not used. They also extend their bitmap model by adding quadruplets (document, path, term, b) where b is true if the path contains the term, which corresponds in our case with text set to true. When text=true, it is also the representation used in [17] for the "flat with structured tag" experiments, where each term (document word) is replaced by its *text path* in a flat vector.

- Case min=1, max=*, root=true, leaf=false, text= true;
  A document is represented by the set of all the root text paths (of any length) in the document, where a term will belong to its parent node and all the embedding nodes. This model is equivalent to the Structure Vector Model proposed in [17], where a document is represented by all its paths of length between 1 and the height h of the document tree. The frequency of terms associated with a path is relative to the subtree associated with that path.
- Case min=1, max=L, root=false, leaf=false, text=false;
  One of the representation proposed in [13] is based on paths of length smaller than L, although they can also fix the level in the tree where the paths must start. In our case paths will start at the root or at any level in the tree.
- Case min=n, max=n, root=false, leaf=true, text= true;
  This case corresponds to representing documents by leaf-ending sub-paths of length n, and therefore providing a limited context to the terms in the documents. One of the representations developed in [13] includes the definitions of leaf-ending paths as well root-beginning paths, of length less than L. They seems to use text as well, but this is not clearly described.

Our representation of XML documents using sub-paths is therefore flexible enough to subsume many of the representations used in other works.

4 Clustering Approach

Since we represent XML documents as a set of paths seen as words, we can use traditional clustering methods for flat texts. However we have to deal with two issues: first, reducing the number of paths in case they are too many; secondly, the possible dependency of paths. Before presenting the clustering approach we address these two issues.

4.1 Further feature selection

Algorithms for clustering such as the k-means are linearly dependent on the size of the data, that is, the number of words that represent the documents. In our case the document will be represented not only by the different words in the text but possibly by their contextual path (i.e. generating as many different occurrences of a word as the different contexts in which it occurs). Moreover they may be extra "words" corresponding to any sub-path in the document trees (node paths). Obviously there will be much more leaf-ending paths than root-paths since trees are expending from the root to the leaves. It is therefore necessary to limit the number of paths that represent the documents to reduce the clustering time.

We apply two levels of feature selection in the path generation: structure level and text level. Then we reduce the number of paths by applying standard selection on words using their relative frequency (Tf/Idf).

Structure level Usually we reduce the number of generated paths by regrouping some tags in more semantic categories, using our knowledge of the DTD or the collections. For example we will replace tags for different sections (ss1, ss2, sec), by a single tag "sec", or ignore presentation tags. Since the INEX collections were specially preprocessed for the XML mining track, we did not have to take care of these semantic groupings.

Text level For the textual content of the document, we use standard reduction methods:

- stop list word for suppressing insignificant word
- suppression of terms shorter than 4 characters
- stemming using the Porter's stemmer [15].

Frequency of paths As said before, the documents are represented by a set of paths that depends on the chosen parameters. First, the frequency of a path is calculated and normalized using the $TfIdf$ formula (number of path occurrence in the document over the number in the whole collection) Paths that are too frequent or too rare will be suppressed. In particular paths that occur only once in the collection will be suppressed since it will not affect the clustering process (those paths will not contribute).

For the remaining paths, we calculate their normalised weight in the document by dividing the number of occurrence of the path by the number of the paths in the document (standard vector normalization to take into account the length of documents).

4.2 Word Dependency

Clustering algorithms based on the vector model rely on the independence of the various dimensions (words) for calculating distance between the vectors. Although it is not always verified in practice with words in texts, it usually works fine. In our case, where words are sub-paths in the document tree, there is an obvious dependency between embedded sub-paths. For example, the two paths bdy.sec and bdy.sec.st are not independent since the second one can exist only if the first one exists, the first one being embedded in the second one. However, two overlapping paths, such as bdy.sec and sec.st would not be regarded as dependent. We only consider structural dependency here, not dependency that would derive from the DTD itself, for example if two siblings are mandatory according to the DTD definition. This later dependency would not affect the clustering results since the two paths would then be present in all the documents and therefore eliminated as very frequent.

To deal with the first case of dependency, we partition the paths by their length and treat each set of paths as a different modality in the clustering algorithm as explained below.

4.3 Clustering Method

Our clustering algorithm is based on the partitioning method proposed by [1], where the distance between clusters is based on the frequency of the words of the selected vocabulary. This approach is equivalent to the k-means algorithm. As for the k-means we represent the clusters by prototypes which summarize the information (paths) of the documents belonging to each of them.

More precisely, if the vocabulary counts p words, each document s is represented by the vector $x_s = (x_s^1, ..., x_s^j, ..., x_s^p)$ where x_s^j is the number of occurrences of word x_j in the document s, then the prototype g for a class U_i is represented by $g_i = (g_i^1, ..., g_i^j, ..., g_i^p)$ with $g_i^j = \sum_{s \in U_i} x_s^j$.

Finally, the prototype of each class been fixed, every element is assigned to a class according to its proximity to the prototype. The proximity is measured by a classical distance between distributions (e.g. euclidean distance):

$$d(x, y) = \sqrt{\sum_{j=1}^{m} (x_j - y_j)^2}, \; m \text{ is the number of modalities.}$$

When there are dependencies between paths[2], we replace the above formula by the following:

$$d(x, y) = \sqrt{\sum_{k=1}^{p} \sum_{j=1}^{m_k} (x_j^k - y_j^k)^2}$$

where p is the number of variables, and m_k is the number of modalities for the variable k

5 Evaluation and metrics

Clustering evaluation is always a bit difficult, since, unlike classification, clustering is supposed to discover significant clusters whose number is not known in advance. However standard evaluation can be made on well known collection were some existing classification can be used as a reference. Since training sets are provided for the XML tracks we are able to evaluate our clustering approaches using them. We used different standard measures and compared their behavior when increasing the number of clusters and using different path lengths. We recall below the definition of the four metrics we use: F-measure, entropy, purity and corrected rand.

- The **F-measure** proposed by [11] combines the precision and recall measures from information retrieval and treats each cluster as if it were the result of a query and each class as if it were the desired answer to that query. It is the *harmonic mean* between precision and recall.

[2] For complete paths (option root=true and leaf=true), there are no embedded paths so (1) can be used.

– The **Corrected Rand Index** has been proposed by [9] to compare two partitions.
Let $U = \{U_1, \ldots, U_i, \ldots, U_r\}$ and $P = \{C_1, \ldots, C_j, \ldots, C_k\}$ be two partitions of the same data set having r and k clusters respectively. The Corrected Rand is calculated by:

$$CR = \frac{\sum_{i=1}^{r}\sum_{j=1}^{k}\binom{n_{ij}}{2} - \binom{n}{2}^{-1}\sum_{i=1}^{r}\binom{n_{i.}}{2}\sum_{j=1}^{k}\binom{n_{.j}}{2}}{\frac{1}{2}[\sum_{i=1}^{r}\binom{n_{i.}}{2} + \sum_{j=1}^{k}\binom{n_{.j}}{2}] - \binom{n}{2}^{-1}\sum_{i=1}^{r}\binom{n_{i.}}{2}\sum_{j=1}^{k}\binom{n_{.j}}{2}} \quad (2)$$

where $\binom{n}{2} = \frac{n(n-1)}{2}$ and n_{ij}, $n_{i.}$, $n_{.j}$ and n are defined as above.

– **Entropy**: it measures the class distribution of each cluster. The smaller the entropy value, the better the clustering solution. A perfect clustering solution would be the one that leads to clusters that contain documents from only a single class, in which case the entropy will be zero [20].
– **Purity**: measures the percentage of documents in a cluster that belong to the largest class of documents in this cluster. In general the larger the value of purity, the better the clustering solution [20].

6 Experimentations with the INEX collections

The INEX XML mining track provides a number of collections for evaluating clustering methods. Some of them consist only of document tree structure (structure only collections), while the others correspond to XML documents with textual content (structure and content collections). The training sets consist of a subset of the documents in each collection, together with the class they belong to. As a consequence the expected number of clusters for each collection is known in advance. Below we give a short summary of the test collections we use for our experiments. Unfortunately we were not able to carry all the experiments before the date of the workshop. We hope to get more results by the time of the final version of this paper.

6.1 The IEEE collections

From the standard IEEE collection used in INEX ad-hoc retrieval experiments, two collections, INEX-s and INEX-cs, have been derived for XML mining. Both collections have been preprocessed by the XML mining track's organisers in order to eliminate useless tags, as well as remove information (the name of the Journal) that would identify obviously the class the document belongs to. For INEX-s, the clustering task is to identify the two classes that correspond, first to *Transactions* Journals, second to other Journals. It is expected that the two types of Journals use different parts of the IEEE DTD and that articles could be easily partitioned into the two classes.

For INEX-cs, the clustering task, using both the structure and the content of the articles, is to identify the six classes proposed in [4] and built from the 18 existing Journals.

Table 3. Results for Inex-s (training collection) for path length 3 and 4, and cluster number set to 2 or 4

Path length	Root	Leaf	Nb.of Clusters	Fmeasure	Corr.Rand	Entropy	Purity
3	T	F	2	0.662	0.098	0.755	0.663
3	F	T	2	0.667	0.105	0.745	0.667
3	**T**	**F**	**4**	**0.661**	**0.423**	**0.185**	**0.963**
3	F	T	4	0.549	0.005	0.728	0.682
4	T	F	2	0.625	-0.044	0.871	0.650
4	F	T	2	0.655	0.087	0.757	0.655
4	T	F	4	0.542	0.208	0.457	0.857
4	F	T	4	0.545	-0.001	0.737	0.675

6.2 The Movie database collections

The MovieDB corpus is a corpus of XML documents describing movies. It was built using the IMDB database. It contains 9643 XML documents. Each document is labelled by one thematic category which represents the genre of the movie in the original collection and one structure category. There are 11 thematic categories and 11 possible structure categories which correspond to transformations of the original data structure. There are four resulting test collections for clustering based only on structure, and two test collections for clustering using both content and structure.

7 Results

Since training sets were provided, we use them to evaluate our approach before getting the official results from the track organisers.

7.1 IEEE collections

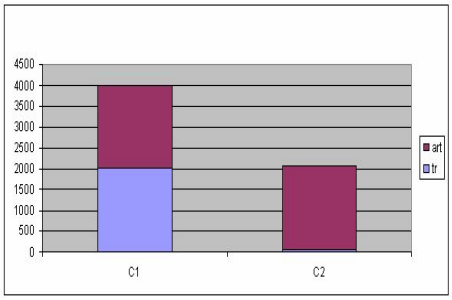

Fig. 2. The repartition of classes *Articles* and *Transactions* on two clusters.

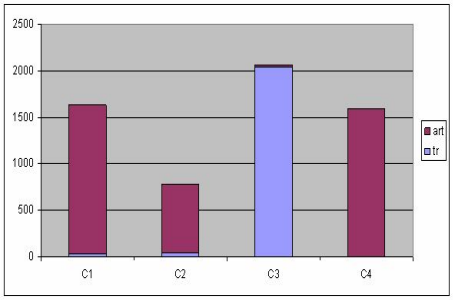

Fig. 3. The repartition of classes *Articles* and *Transactions* on four clusters.

We first test our approach with different path lengths either starting at the root or ending at the leaves. We present only a few results in Table 3: The best values (especially entropy and purity) are obtained for documents represented by the set of paths of length 3 starting from the root. It must be noticed that it happens for four clusters, not the two that were expected. There is nothing wrong with this result since there is no intrinsic reason why some articles would not have an overall structure more dissimilar to other articles than to *Transactions*.

Figures 2 and 3 shows the repartition of the two (resp. four) clusters on the two expected classes. We can see that in the case of four clusters, one class (Transaction) maps quite closely to cluster 3, while the other three clusters contain mostly articles. We have not tried to analyse more deeply what could be the similarities between articles within these three clusters.

Then we sent two runs to the XML document mining tracks. The parameters we used and the official results are shown in table 4.

Table 4. Official Results for inex-s (test collection) for two runs

Run	Path-length	Root	Leaf	Nb. of Clusters	Micro Entropy	Macro Entropy	Micro Purity	Macro Purity
Run 1	3	F	T	2	0.744	0.627	0.663	0.627
Run 2	4	T	F	4	0.109	0.137	0.984	0.878

These results confirms the results with the training set that clustering in four clusters give better results than clustering in two clusters.

7.2 MovieDB

We did the same type of experiments for the four structured collections built from the Movie databases. For each of the four collections we set the path length alternatively to three and four, with either root paths or leaf-ending paths. We cluster the documents into 9, 11 or 13 clusters respectively, the expected classes being 11.

Table 5 shows the measure values when clustering the training collection m-db-s0. The results are always better when using leaf-ending paths over root paths, unless they are identical when the path length is set to 4. The best value for the purity is obtained when clustering into 11 clusters, but the differences for other measures may not be all significant.

Tables 6, 7, 8 show the evaluation measure values when clustering the collection m-db-s1, m-db-s2, m-db-s3 respectively.

8 Conclusion

In this paper we proposed a flexible representation of XML documents by a set of their paths that are generated according to specified parameters. We evaluate

Table 5. Results for m-db-s0 (training collection) for path length 3 and 4, and cluster number set to 9, 11 and 13

Path length	Root	Leaf	Nb.of Clusters	Fmeasure	Corr.Rand	Entropy	Purity
3	T	F	9	0.541	0.370	0.286	0.632
3	F	T	9	0.708	0.575	0.154	0.819
3	T	F	11	0.509	0.357	0.285	0.640
3	**F**	**T**	**11**	**0.642**	**0.506**	**0.151**	**0.820**
3	T	F	13	0.465	0.328	0.284	0.640
3	F	T	13	0.595	0.473	0.151	0.819
4	T	F	9	0.714	0.576	0.158	0.813
4	F	T	9	0.714	0.576	0.158	0.813
4	**T**	**F**	**11**	**0.663**	**0.532**	**0.157**	**0.821**
4	**F**	**T**	**11**	**0.663**	**0.532**	**0.157**	**0.827**
4	T	F	13	0.648	0.519	0.155	0.820
4	F	T	13	0.648	0.519	0.155	0.820

Table 6. Results for m-db-s1 (training collection) for path length 3 and 4, and cluster number set to 9, 11 and 13

Path length	Root	Leaf	Nb.of Clusters	Fmeasure	Corr.Rand	Entropy	Purity
3	T	F	9	0.477	0.292	0.357	0.554
3	F	T	9	0.722	0.600	0.152	0.816
3	T	F	11	0.446	0.277	0.355	0.553
3	F	T	11	0.652	0.520	0.151	0.816
3	T	F	13	0.422	0.264	0.355	0.557
3	F	T	13	0.615	0.486	0.150	0.816
4	T	F	9	0.751	0.639	0.156	0.812
4	F	T	9	0.751	0.639	0.156	0.812
4	T	F	11	0.692	0.587	0.154	0.813
4	F	T	11	0.692	0.587	0.154	0.813
4	T	F	13	0.633	0.515	0.152	0.813
4	F	T	13	0.633	0.515	0.152	0.813

Table 7. Results for m-db-s2 (training collection) for path length 3 and 4, and cluster number set to 9, 11 and 13

Path length	Root	Leaf	Nb.of Clusters	Fmeasure	Corr.Rand	Entropy	Purity
3	T	F	9	0.534	0.384	0.301	0.605
3	F	T	9	0.495	0.177	0.538	0.551
3	T	F	11	0.502	0.367	0.300	0.607
3	F	T	11	0.510	0.217	0.477	0.611
3	T	F	13	0.468	0.348	0.301	0.605
3	F	T	13	0.566	0.328	0.399	0.696
4	T	F	9	0.514	0.278	0.402	0.589
4	F	T	9	0.459	0.166	0.517	0.520
4	T	F	11	0.585	0.378	0.310	0.696
4	F	T	11	0.522	0.282	0.436	0.602
4	T	F	13	0.544	0.331	0.317	0.690
4	F	T	13	0.501	0.278	0.426	0.606

Table 8. Results for m-db-s3 (training collection) for path length 3 and 4, and cluster number set to 9, 11 and 13

Path length	Root	Leaf	Nb.of Clusters	Fmeasure	Corr.Rand	Entropy	Purity
3	T	F	9	0.482	0.305	0.352	0.546
3	F	T	9	0.592	0.332	0.441	0.638
3	T	F	11	0.449	0.287	0.345	0.546
3	F	T	11	0.583	0.303	0.447	0.638
3	T	F	13	0.408	0.236	0.352	0.564
3	F	T	13	0.581	0.310	0.441	0.652
4	T	F	9	0.518	0.289	0.372	0.620
4	F	T	9	-	-	-	-
4	T	F	11	0.527	0.339	0.312	0.674
4	F	T	11	0.403	0.102	0.539	0.501
4	T	F	13	0.528	0.340	0.310	0.676
4	F	T	13	-	-	-	-

the approach on some of the collections proposed by the INEX XML document track. Unfortunately we were not able to run all the tests in time for the run submission and this paper. We hope to go further before the workshop and the final version of the paper. In particular we would like to work with the content and structure collections, although we expect that the large number of generated paths may harm the clustering performance.

References

1. G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, and H. Ralambondrainy. *Classification Automatique des Données, Environnement statistique et informatique*. Dunod informatique, Bordas, Paris, 1989.
2. T. Dalamagas, T. Cheng, K.-J. Winkel, and T. K. Sellis. Clustering XML Documents by Structure. In *SETN*, pages 112–121, 2004.
3. T. Dalamagas, T. Cheng, K.-J. Winkel, and T. K. Sellis. Clustering XML Documents Using Structural Summaries. In *EDBT Workshops*, pages 547–556, 2004.
4. L. Denoyer. *Apprentissage et inférence statistique dans les bases de documents structurés : Application aux corpus de documents textuels*. PhD thesis, L'UNIVERSITÉ DE PARIS 6, December 2004.
5. L. Denoyer, J.-N. Vittaut, P. Gallinari, S. Brunesseaux, and S. Brunesseaux. Structured multimedia document classification. In *ACM Document Engeneering*, Grenoble, Nov. 2003.
6. A. Doucet and H. Ahonen-Myka. Naïve Clustering of a large XML Document Collection. In *INEX Workshop*, pages 81–87, 2002.
7. S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. Detecting Structural Similarities between XML Documents. In *WebDB*, pages 55–60, 2002.
8. F. D. Francesca, G. Gordano, R. Ortale, and A. Tagarelli. Distance-based Clustering of XML Documents. In L. De Raedt and T. Washio, editors, *MGTS-2003 : Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 75–78. ECML/PKDD'03 workshop proceedings, September 2003.
9. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.
10. Y. Jianwu and C. Xiaoou. A semi-structured document model for text mining. *J. Comput. Sci. Technol.*, 17(5):603–610, 2002.
11. B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM Press.
12. W. Lian, D. W.-L. Cheung, N. Mamoulis, and S.-M. Yiu. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. *IEEE Trans. Knowl. Data Eng.*, 16(1):82–96, 2004.
13. J. Liu, J. T. L. Wang, W. Hsu, and K. G. Herbert. XML Clustering by Principal Component Analysis. In *ICTAI*, pages 658–662, 2004.
14. A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *Proceedings of the Fifth International Workshop on the Web and Databases (WebDB 2002)*, Madison, Wisconsin, USA, June 2002.
15. M. F. Porter. An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

16. A. Termier, M.-C. Rousset, and M. Sebag. TreeFinder: a First Step towards XML Data Mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 450, Washington, DC, USA, 2002. IEEE Computer Society.
17. J. Yi and N. Sundaresan. A classifier for semi-structured documents. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 340–344, New York, NY, USA, 2000. ACM Press.
18. J. P. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg. BitCube: A Three-Dimensional Bitmap Indexing for XML Documents. *Journal of Intelligent Information Systems*, 17(2-3):241–254, 2001.
19. M. J. Zaki and C. C. Aggarwal. XRules: an effective structural classifier for XML data. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–325, New York, NY, USA, 2003. ACM Press.
20. Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report 01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.