2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

A Self-Organising Map Approach for Clustering of XML Documents

F. Trentini, and M. Hagenbuchner, *Member, IEEE*, and A. Sperduti, *Member, IEEE*, and F. Scarselli, *Member, IEEE*, and A.C. Tsoi

***Abstract*— The number of XML documents produced and available on the Internet is steadily increasing. It is thus important to devise automatic procedures to extract useful information from them with little or no intervention by a human operator. In this paper, we investigate the efficacy of an unsupervised learning approach, namely Self-Organising Maps (SOMs), for the automatic clustering of XML documents. Specifically, we consider a relatively large corpus of XML formatted data from the INEX initiative and evaluate it using two different self-organising map models. The first model is the classical SOM model, and it requires the XML documents to be represented by real-valued vectors, obtained using a "bag of words" (or better a "bag of tags") approach. The other model is the SOM for structured data (SOM-SD) approach which is able to cluster structured data, and it is possible to feed the model with tree structured representations of the XML documents, thus explicitly preserving the structural information in the documents. The experimental results show that the SOM model exhibits quite a poor performance on this problem domain which requires the ability to encode structural properties of the data. The SOM-SD model, on the other hand, is able to produce a good clustering and generalization performance.**

I. Introduction

The World Wide Web presents one of the most challenging applications for many machine learning tasks. This is because of its sheer size, and its dynamic nature. The Web is currently estimated to contain approximately 57.59 billion HTML documents [9], and continues to grow in size at a phenomenal rate [13]. The finding of relevant information in the Web is becoming an increasingly difficult problem as it represents probably one of the largest man-made storage system ever built. This is overcome to some extend by the use of general Internet search engines, e.g. Google, Yahoo!. However, it is known that these general purpose search engines access only a limited portion of the Web, and leave a sizable portion of the Web yet unexplored. From this perspective, machine learning techniques can be very useful to automatically process data which would be left unused or unexplored otherwise.

In addition, the Web can be viewed as a large domain of unlabeled data, and is most appropriately represented as graphs. Similarly, Web documents are generally formatted syntactically using meta languages such as HTML or XML. Thus, the documents are more suitably represented as graph data structures with tags[1] representing nodes, and relationships between the tags define the links between the nodes.

It should be possible to cluster portions of the Web, or Web documents into clusters such that elements within the same cluster share similarities. This will be very useful for information retrieval tasks. For example, if we have a sample Web document and wish to find similar documents (with respect to some similarity criterion) then a clustering algorithm will retrieve very quickly similar documents simply by returning documents which are located in the same cluster as the original sample document.

For the above reasons, machine learning approaches capable of performing unsupervised clustering of structured data are of great interest.

An example of unsupervised learning model is given by Kohonen's self-organizing map (SOM) [10], which is a well-known and popular neural network model for applications that require dimension reduction or the clustering of possibly large sets of data. So far, extensions of the SOM concept produced SOMs for data sequences (see [8], [10]), a SOM for (tree-like) data structures (SOM-SD) [1], [8], and two contextual SOM-SDs (CSOM-SD) [2], [3] which are capable to encoding more general types of graphs.

The classic SOM architecture has already been applied for processing Web documents, e.g. WEBSOM [4] is a method for organizing miscellaneous text documents into meaningful maps for exploration and search; or to help the organization of a digital library, e.g. SOMLib [5]. None of these systems, however, take into account any structural information, such as the ones encoded in XML documents. It would be intuitively clear that if somehow the structural content of the documents, e.g., XML documents is also taken into account in the clustering process, it should be possible to obtain improved clustering results. The incorporation of such structural information in an adaptive learning system like the SOM provides a number of advantages:

- It reduces or completely removes the need to pre-process the data. This is because in many problem domains, the data is naturally represented as a graph, and hence, a vector representation does not need to be extracted.
- It adds the capability of encoding the topology inherent in the data represented by graphs. The graph topology is one of the most important properties in characterizing a graph based data structure.

F. Trentini and F. Scarselli are with the Dipartimento di Ingegneria dell'Informazione, Siena University, Italy; M. Hagenbuchner is with the University of Wollongong, Australia; A. Sperduti is with Dipartimento di Matematica Pura ed Applicata, Padova University, Italy; A.C. Tsoi is with Monash University, Australia.

[1]Elements of HTML or XML are called *tags*.

0-7803-9490-9/06/$20.00/©2006 IEEE

In addition, using neural networks (NNs) for the clustering or the dimension reduction task has many advantages:

- NNs are trained on a set of examples and, once trained, are able to generalize to previously unseen data.
- NNs are fault tolerant and are relatively insensitive to noise in the data. In other words, even if the data contains some noise, the classification results remain relatively unchanged.

With SOMs in particular, the computational complexity of the approach grows linearly in time with the increase of size of the dataset. This is an important issue if we are dealing with a large dataset, such as a Web. Thus, NNs can be particularly useful for data mining tasks. The cost is low due to the linear computational complexity, and training needs to be performed only on a relatively small subset of the problem[2] domain due to its ability to generalize.

There are a number of other NN approaches for the processing of general types of graph data structures. These include: the Graph Neural Networks (GNN) [14], Contextual Recursive Cascade Correlation networks (CRCC) [11], and more recently, the Neural Network for graphs (NN4G) [12]. All of these models are trained in a supervised fashion, and hence, cannot be employed with unlabeled data, such as a Web. In this paper, we present, to the best of our knowledge, the only known unsupervised neural network approach on the task of clustering or dimension reduction of general graph data structures as represented by XML documents.

Here, we investigate the ability of the above mentioned SOM-based models to cluster a relatively large set of XML documents which belongs to the XML Document Mining Challenge (accessible via *http://xmlmining.lip6.fr*) of the INitiative for the Evaluation of XML Retrieval (INEX). We show that, for the learning to be effective, i.e. to generalize to previously unseen new XML documents, it is important to preserve the structural information contained in the data.

The structure of this paper is as follows: in Sections II and III, we will present the underlying concepts behind the SOM and SOM-SD, respectively. Performance measures used to evaluate the quality of the models are defined in Section IV. Experimental results in applying the proposed approach to a set of XML documents used in the INEX initiative are given in Section V, and finally some conclusions are drawn in Section VI.

II. The Self-Organizing Map

A SOM consists of a number of neurons which are organized on a regular grid called the *map* or the feature map. A codebook vector $\boldsymbol{m}$ is associated with every neuron where the dimension of $\boldsymbol{m}$ is equal to the dimension of the i-th input vector $\boldsymbol{x}_i$. The neurons on the map are bounded together by a topology, and are updated according to a neighborhood function $f(.)$. Often, the topology used is either hexagonal or rectangular. In general, the SOM is a model which is trained on a set of examples as follows:

For every input vector $\boldsymbol{x}_i$ in a training set, obtain the best matching codebook by computing

$$c = \arg\min_j \|\boldsymbol{x}_i - \mathbf{m}_j\| \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm. Then, the codebooks on the map are updated as follows:

$$\Delta\mathbf{m}_j = \alpha(t) f(\Delta_{jc})(\mathbf{m}_j - \mathbf{x}_i) \tag{2}$$

where Δ_{jc} is the topological distance between neuron j and the winning neuron c, and $\alpha(t)$ is a learning coefficient which decreases to zero with time t. Equations (1) and (2) are executed for a set number of iterations. The result is a SOM which maps data onto a $n-$dimensional map, where typically $n = 2$. It is shown in [10] that the strength of the SOM is in its ability to map high dimensional data onto a low dimensional *display space* while preserving the topology of the input data. The SOM is trained in an unsupervised fashion, though some supervised approaches to SOM exist [6], [7], [10]. The supervised methods will not be considered in this paper.

It should be noted that this model can be applied to structured objects only by representing each object by a set of predefined and computable features collected in vectorial form.

III. The SOM for data structures

The SOM for Data Structures (SOM-SD) extends the SOM in its ability to encode directed tree structured graphs [1]. This is accomplished by processing individual nodes of a graph rather than by processing the graph as a whole.

Let

$$\boldsymbol{y}_v = f(\boldsymbol{x}_v) \tag{3}$$

be the response (output) of the network when processing the information available in a node (vertex) v, where $f(\cdot)$ is a general nonlinear function. The network input $\boldsymbol{x}_v$ is defined as a vector obtained by concatenating the data label $\boldsymbol{l}_v \in \mathbb{R}^p$ which is local to the node v and the coordinates of the mapping of children vertexes $y_{\mathrm{ch}[v]}$ such that $x_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ch}[v]}]$. In a self-organising map, the output of a child node is represented by the display space (map) of the child node. Normally such a display space is assumed to be a two-dimensional plane. This two dimensional plane is assumed to be represented by a two dimensional grid. The co-ordinates of the winning neuron of the child node based on the response of its inputs are the inputs to the parent node. Note that in this case the information to be processed by the local node is drastically reduced compared with the need to process the vector representing the entire graph structure. Secondly, the topological relationships among the nodes will be preserved by the links among these nodes. Such topological relationships would be lost if the graph structure is represented as a single vector, instead of a number of sub-vectors representing the inputs to individual nodes.

[2]While the size of the training data set should be small, it is advisable that such data set should be chosen with some care, which should contain most of the features expected in the data set.

These input vectors can be made constant in size if its maximum out-degree is known. Where the maximum out-degree is not known, then by assuming a maximum out-degree, graphs with more out-degrees than assumed will be truncated. If the maximum out-degree is o, then for nodes with less than o children, padding is performed using an impossible coordinate $(-1,-1)$ for a 2-dimensional map. This procedure generates fixed sized vectors, one for each vertex. Hence, such input can be processed in a similar fashion to a standard SOM with two differences

(1) the need to update the $\boldsymbol{x}_v$ at every iteration, and
(2) the introduction of weighting parameters which balance the contribution of $\boldsymbol{l}$ and $\boldsymbol{y}_{\mathrm{ch}[v]}$ in computing the best matching codebook entry:

$$r = \arg\min_i(\|(\boldsymbol{x}_v - \boldsymbol{m}_i)\boldsymbol{\Lambda}\|) \tag{4}$$

where $\boldsymbol{x}_v$ is the input vector for vertex v, $\boldsymbol{m}_i$ the i-th codebook, and $\boldsymbol{\Lambda}$ is a $m \times m$ dimensional diagonal matrix with its diagonal elements $\lambda_{1,1} \cdots \lambda_{p,p}$ set to μ_1, and $\lambda_{p+1,p+1} \cdots \lambda_{m,m}$ set to μ_2. The constant μ_1 influences the contribution of the data label component to the Euclidean distance in (4), and μ_2 controls the influence of the children's coordinates to the same Euclidean distance.

It becomes clear that the mapping coordinates of $\boldsymbol{y}_{\mathrm{ch}[v]}$ are a representation of sub-structures headed by the nodes in $\mathrm{ch}[v]$. Hence, we can refer to the mapping of any node more appropriately as its *state* as it summarizes the contribution of the graph structure sub-tending to the current node, but not that beyond the current node.

A network input $\mathbf{x}_v$ requires the availability of all states of the children of node a v. This dictates the processing of nodes in a reverse topological order (i.e., from the leaf nodes towards the root.) Graph structures which support this mode of processing are commonly known as *trees*.

The training algorithm of the SOM-SD can be given as follows:

Step 1: Choose a node v from the training set ensuring that all of the children of node v have already been processed. Initialize $\boldsymbol{x}_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ch}[v]}]$, and compute the best matching codebook r by finding the most similar codebook entry $\mathbf{m}_r$. This can be achieved, e.g., by using the Euclidean distance as follows:

$$r = \arg\min_i \|(\mathbf{x}_v - \mathbf{m}_i)\boldsymbol{\Lambda}\| \tag{5}$$

Step 2: Update network parameters by using Equation 2. Repeat Step 1 and Step 2 for every node in the training set.

The algorithm cycles through Steps 1 to 2 until a given number of training iterations is performed, or when the mapping precision has reached a given prescribed threshold.

Once trained, information can be retrieved efficiently from a SOM-SD. This is performed by using a set of data in place of the training set, and by executing Step 1 on this dataset. This will compute the mapping of all nodes in a dataset. Given a sample document (represented by a graph) we can now retrieve similar documents by returning graphs which were mapped near the location on which the known document was mapped.

IV. Performance Measures

It is evident that a simple quantization error is an insufficient indicator of the performance of a SOM-SD since such an approach ignores the fact that structural information is being mapped. In fact, there are a number of criteria with which the performance of a SOM-SD can be measured. These are performance indicators on its retrieval capability, classification performance, clustering performance, and its mapping precision. The retrieval capability reflects the accuracy of retrieved data from the SOM-SD, the classification performance shows how well the map performs on a classification task, the clustering performance shows how well data is grouped together on the map, and the mapping precision shows how accurately structural information is encoded in the map. Computing retrieval capability and clustering performance require the availability of target values, while the mapping precision can be computed without further requirements.

Retrieval capability (R): This can be computed quite simply if class/cluster membership information is available for a set of test data, i.e., for each XML document d_j a target class $y_j \in \{t_1, \ldots, t_q\}$ is given. Since each XML document is represented by a tree, in the following we focus our attention just on the root of the tree. Thus, with r_j we will refer to the input vector for SOM-SD representing the root of the XML document d_j. The R index is computed as follows: the mapping of every node in the dataset is computed; then for every neuron i the set $win(i)$ of root nodes for which it was a winner is computed. Let $win_t(i) = \{r_j | r_j \in win(i) \text{ and } y_j = t\}$, the value $R_i = \max_t \frac{|win_t(i)|}{|win(i)|}$ is computed for neurons with $|win(i)| > 0$ and the index R computed as $R = \frac{1}{W}\sum_{i,|win(i)|>0} R_i$, where $W = \sum_{i,|win(i)|>0} 1$ is the total number of neurons which were activated at least once by a root node.

Classification performance (C): If class labels are available for the data in the training set and the test set, then a classification performance can be computed as follows:

$$C_j = \begin{cases} 1 & \text{if } y_j = t_r^*, \quad t_r^* = \arg\max_t |win_t(r)| \\ 0 & \text{else} \end{cases},$$

where r is the index of the best matching codebook for document d_j (typically measured at the root node). Then,

$$C = \frac{1}{N}\sum_{j=0}^{N} C_j, \tag{6}$$

where N is the number of documents (graphs) in the test set.

Clustering performance (P): A more sophisticated approach is needed to compute the ability of a SOM-SD to suitably group data on the map. The following approach is proposed:

1) Compute the quantities R_i as defined above, and let $t_i^* = arg\max_t |win_t(i)|$
2) For any activated neuron compute the quantity:

$$P_i = \frac{\sum_{j=1}^{|\mathcal{N}_i|} \frac{|win_{t_i^*}(j)|}{|win(j)|} + \frac{|win_t(i)|}{|win(i)|}}{|\mathcal{N}_i| + 1} = \frac{\sum_{j=1}^{|\mathcal{N}_i|} \frac{|win_{t_i^*}(j)|}{|win(j)|} + R_i}{|\mathcal{N}_i| + 1} \quad (7)$$

where $\mathcal{N}_i = \{v | v \in \mathrm{ne}[i], win(v) \neq \emptyset\}$, and $\mathrm{ne}[i]$ are the direct neighbors of neuron i.
3) The overall network performance is then given by:

$$P = \frac{\sum_i P_i}{W} \quad (8)$$

Performance values close to 1 indicates a perfect grouping, while a value closer to 0 indicates poor clustering results. Thus, this measure indicates the level of disorder inside a SOM-SD.

Structural mapping precision (e): This index measures how well structural (e) information is encoded in the map. A suitable method for computing the structural mapping precision was suggested in [3]. In this case, just the skeleton of the trees is considered, i.e. the information attached to vertexes is disregarded, and only the topology of the trees is considered. Notice that this measure does not consider the information about the class to which an XML document (i.e., a tree) belongs to. For this reason, all the neurons of a map are now considered, since we are also interested in neurons which are winners for sub-structures. This measure is computed as follows:

$$e = \frac{1}{N} \sum_{i=1, n_i \neq 0}^{N} \frac{m_i}{n_i} \quad (9)$$

where n_i is the number of sub-structures mapped at location i, m_i is the greatest number of sub-structures which are identical and are mapped at location i. N is the total number of neurons activated by at least one sub-structure during the mapping process. Hence, e is an indicator of the quality of the mapping of sub-structures. Values in e close to 1 indicate a very good mapping (indeed a *perfect* mapping if the value is 1), and values closer to 0 indicate a poor mapping.

V. Experiments

Performances of the two Self-Organizing Map methods are evaluated by using a relatively large set of XML formatted documents which were made available as part of the INEX Initiative (INitiative for the Evaluation of XML Retrieval), and was obtained from *http://xmlmining.lip6.fr*

A. The dataset

The corpus (`m-db-s-0`) consists of $9,640$ XML formated documents. The documents contained XML tags only, no further textual information was available. All documents had target values available. $4,824$ of these documents were marked as being the *training set*, all remaining documents formed the test set, which was in no way involved in the training and model selection process, but was used at the end to evaluate the quality of the selected networks.

A tree structure was extracted for each of the documents in the dataset by following the general XML structure within the documents. The resulting dataset featured $9,640$ tree structured graphs, one for each XML document in the dataset. The maximum depth of any graph is 3, the maximum outdegree is $6,418$, and the total number of nodes in the dataset is $684,191$. Hence, the dataset consists of shallow tree structures which can be very wide. A three-dimensional data label is attached to every node in the dataset indicating the XML-tag it represents (more on this below). Each of the graphs in the training set is associated with one of 11 possible clusters (the target information).

For the classical SOM model, the XML documents are represented by real-valued vectors. These are obtained by a pre-processing step that uses the "bag of words" (or better a "bag of tags") approach. It processes the text and obtains some representations before using them in a standard SOM application in what is referred to as the WEBSOM approach[3]. While for the SOM-SD there is no specific need to pre-process the XML documents, we decided to apply a pre-processing step in order to reduce the dimensionality of the dataset. This allows for a reasonable turn around time for a comprehensive set of experiments. Dimension reduction was achieved by consolidating XML tags as follows: repeated sequences of tags within the same level of a structure are collapsed. For example, the structure:

```
<BB>
      <a> </a>
      <b> </b>
      <a> </a>
      <b> </b>
      <a> </a>
      <b> </b>
</BB>
```

is consolidated to

```
<BB>
      <a> </a>
      <b> </b>
</BB>
```

A justification for taking this step can be found by using regular expressions. For example, the expression $(ab)^n$ can be simulated by repeatedly presenting ab for n-times. Hence, it suffices to process the consolidated structure n times.

There were many trees which exhibit such repeated sequences of tags. The consequence of this pre-processing step was that the maximum outdegree is reduced to just 32.

A further dimension reduction is achieved by collapsing simple sub-structures which have the property of a data sequence into a single node. For example, the sequential structure `<A><b><c></c></b></A>` can be collapsed to `<A><b&c></b&c></A>`, and even further to `<A&b&c>`. Since the deepest graph is of depth 3, this implies that the

[3]The WEBSOM which was applied to Web documents in [4] is not suited for the learning problem presented in this paper. This is due to the fact that WEBSOM clusters general text documents by removing any structural or non-textual information in a pre-processing step. Thus, WEBSOM does not cluster documents according to the structure of a document as may be defined by an encapsulating meta language such as HTML or XML. Furthermore, the dataset used in the experiments contained only structural information. No textual information was available.

longest sequence that can be collapsed is of length 3. This pre-processing step reduces the total number of nodes in the dataset to $124,359$.

A unique ID is associated with each of the possible 197 XML tags. In order to account for nodes which represent collapsed sequences, we attach a three dimensional data label to each node. The first element of the data label gives the ID of the XML tag it represents, the second element of the data label is the ID number of the first tag of a collapsed sequence of nodes, and consequently, the third element is the ID of the tag of the leaf node of a collapsed sequence. For nodes which do not represent a collapsed structure, the second and third element in the data label is set to zero.

The number of samples for each of the 11 pattern classes varies as is shown in Table I. It is observed that the smallest pattern class denoted by "4" features 172 samples whereas the largest class "8" is a collection of 769 samples. Furthermore, the properties of the graphs within each pattern class varies considerably as is illustrated in Figure 1. In Figure 1 it is seen that the maximum outdegree of a graph can vary considerably within a pattern class, and can form quite distinct groups of graphs: narrow graphs as produced by patterns in class 6, 7, 8, 9, 10, and 11, and the group of wide graphs as produced by patterns in the classes 1, 2, 3, and 4. Thus, the dataset is rather unbalanced in its features.

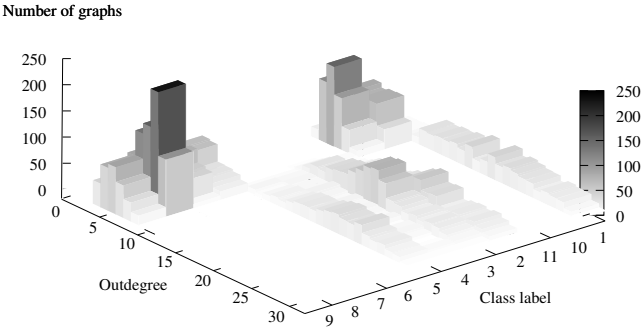

Fig. 1. The distribution of outdegrees in the dataset. Plotted are the number of graphs (z-axis) which have a given outdegree (y-axis) and belong to a given class (x-axis).

No particular effort has been undertaken to compensate this imbalance in the training set.

We emphasize that class information is not used during the training phase which is executed in a strict unsupervised fashion. However, we used the class labels to analyze the properties of the dataset, as well as the quality of trained SOM models.

TABLE I

THE TABLE ON SHOWS THE NUMBER OF GRAPHS IN EACH OF THE 11 CLASSES. TOTAL NUMBER OF PATTERNS IS 4820

Label	1	10	11	2	3	4	5	6	7	8	9
Frequency	598	386	448	486	701	172	435	231	261	769	333

TABLE II

TRAINING PARAMETERS FOR WHICH PEAK PERFORMANCES WERE OBSERVED.

Model	Network size	#Training iterations	$\alpha(0)$	$r(0)$	μ_1	μ_2
SOM	64×48	120	1.0	32	1.0	–
SOM-SD	110×81	32	1.0	21	0.85	0.15

B. Experimental results

The SOM is trained on 4,820 data vectors, each one represents an XML document. The i-th element in the data vector represents the frequency of the i-th XML tag. Thus, the input vectors for the SOM are 197 dimensional containing the complete set of information about the XML tags in a document but does not contain any information about the underlying structure between the XML tags.

In contrast, there are 4,820 graphs with a total of $124,359$ nodes in the training set for the SOM-SD. The data label is of dimension 3 while the structure component of the input vectors is of dimension $2 \times 32 = 64$.

Thus, the SOM is trained on relatively few high-dimensional data vectors while the SOM-SD is being trained on a very large number of nodes which are represented by a relatively small sized vector. This has implications on how to choose the size of the networks. SOMs of size $64 \times 48 = 3072$ were used as a basis. Such networks feature a total of $3072 \times 197 = 605,184$ network parameters which are used to encode the $4,820$ data in the training set. Given that the SOM-SD is to encode $124,359$ 67-dimensional nodes, this would imply that it would be justified to train SOM-SDs of size $(605,184 \times 124,359)/(4820 \times 67) = 233,047 \approx 582 \times 400$ for a direct comparison with the SOM. However, for demonstration purposes, and in order to reduce the computational time, it suffices to train SOM-SDs of size $110 \times 81 = 8910$. A SOM-SD of size 110×81 implies that on average 14 nodes from the training set are mapped onto a single neuron location. In other words, the training set is compressed down to about 7.14% of its original size.

All maps illustrated in this section used a hexagonal topology and a Gaussian neighborhood function.

A large number of experiments were conducted on the training set in order to obtain a good set of training parameters, and to be able to record good performances of each of the models.

The experiments aimed at maximising the e-performance index for the SOM-SD, and at minimizing the quantization error for the SOM. These indices were chosen since these do not rely on the availability of labelled data. When training the SOM and SOM-SD models we observe peak performances when using training parameters as shown in Table II.

In Table II it is observed that a SOM-SD requires less training iterations, and a smaller initial neighborhood radius despite the fact that a larger network is trained. This is most likely due to the fact that the SOM-SD is being trained on a much larger set of input vectors. Training times were about 4 hours for the SOM-SD, and about 63 minutes for the SOM

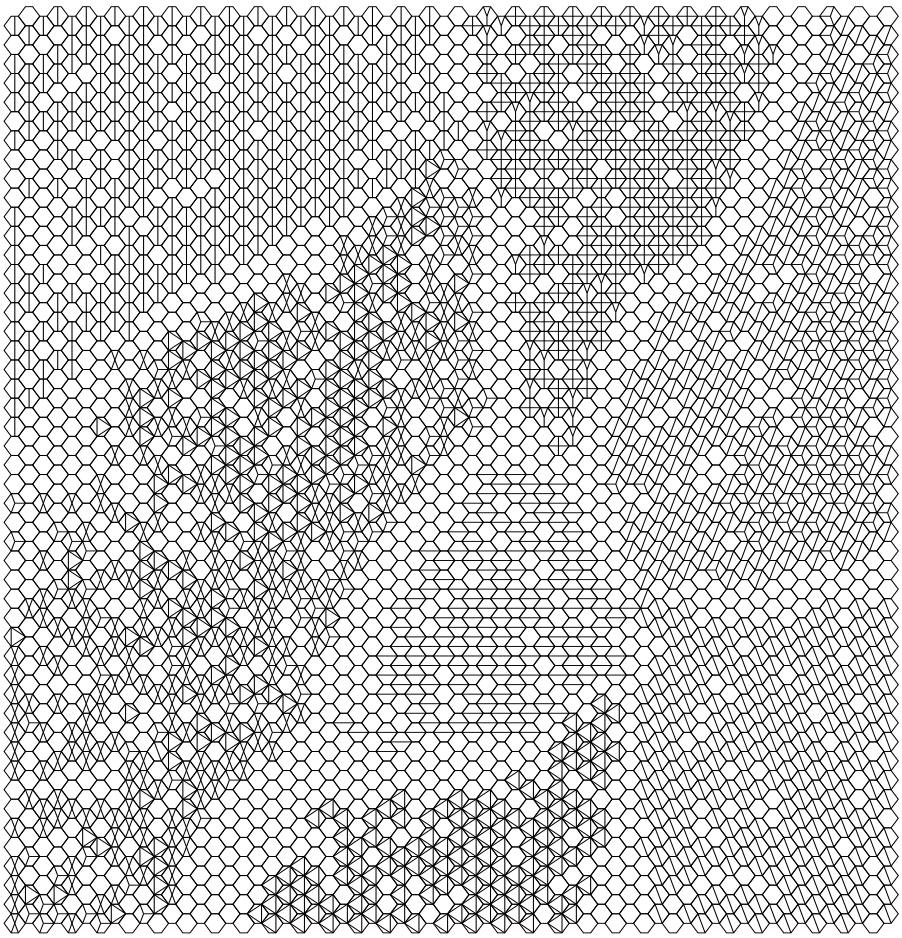

Fig. 2. The mapping of vectors on a standard SOM

when executed on a 3GHz Intel based (single core) CPU.

Figure 2 shows the mapping of vectors in the training set on a trained SOM. The hexagons in Figure 2 refer to the neurons on the map. Each neuron is filled with a pattern indicating the class that most frequently activated the neuron. There are 11 different fill patterns for the 11 possible classes. Neurons which are not filled are not activated by any vector in the training set. It can be seen that a number of well distinct clusters have formed on the map, most of which correspond very nicely with the target label that is associated with the training data. Most clusters are separated from each other by an area of neurons which were not activated. This is a good result since the presence of such border regions should allow for a good generalization performance. An assumption which will be confirmed later in this paper.

The mapping of root nodes in the training set on a trained SOM-SD is shown Figure 3. Neurons which are not filled are either not activated by any node, or are activated by a node other than the root node. It can be observed in Figure 3 that large sections of the map are not activated by any root node. This is due to the fact that root nodes are a minority in the dataset. Only $4,824$ nodes out of the total $124,468$ nodes in the training set are root nodes. Hence, only a relatively small portion of the map is activated by root nodes. It is also observed that graphs belonging to different classes form clear clusters some of which are very small in size. Given this observation, it may be expected that the SOM-SD will be able to generalize well.

TABLE III

BEST RESULTS OBTAINED DURING THE EXPERIMENTATION WITH MAPS OF SIZE 64×48 (SOM), AND FOR MAPS OF SIZE 89×67 (SOM-SD)

	train set					test set			
	R	e	P	C	Z	R	e	P	C
SOM	87.2%	N/A	0.87	13.2%	2.39	89.9%	N/A	0.85	11.8%
SOM-SD	95.29%	0.73	0.81	95.3%	11.96	95.28%	0.73	0.80	93.9%

In Table III we compare the performance indices as was suggested in Section IV. The results shown in Table III reflect the visual observations made earlier. The SOM-SD performs considerably better on the classification performance. This is an indication that the classification of the documents relies heavily on the ability to encode the underlying structure of the XML documents. For most other performance measures, the SOM and the SOM-SD perform at similar levels. In

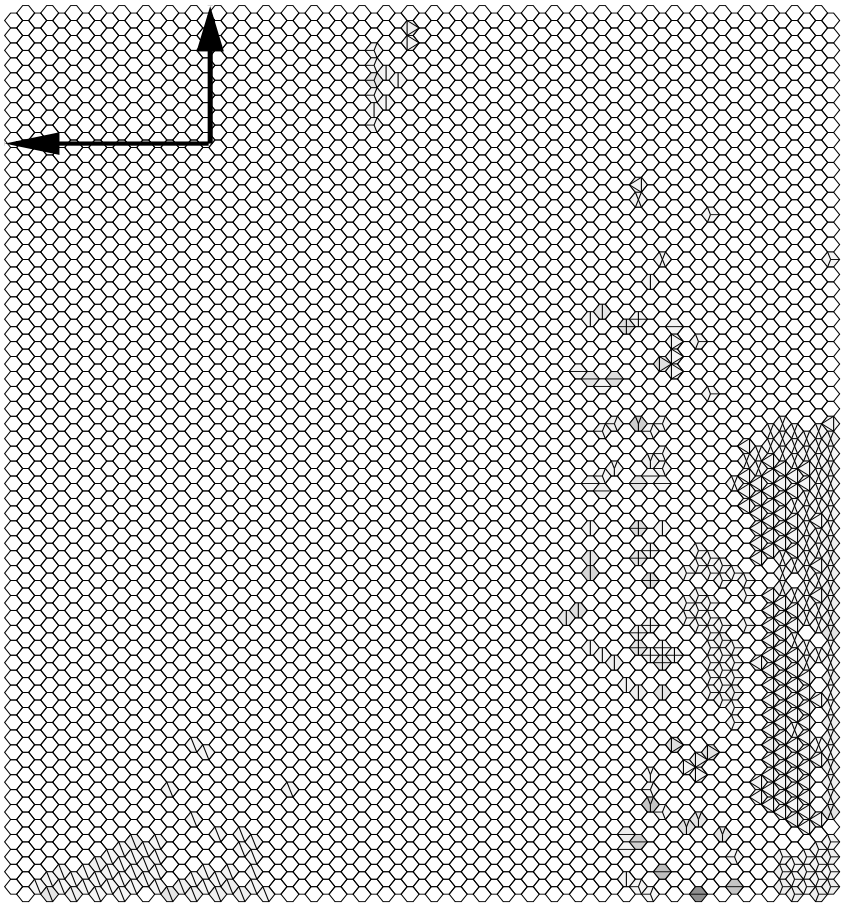

Fig. 3. The portion of the map which mapped root nodes on the SOM-SD. 403 neurons were activated by root nodes in the training set. The arrows indicate that the direction in which the map extends further. The actual map is of size 110×81.

addition, Z in Table III shows the compression ratio[4]. This emphasizes the efficiency of the SOM-SD since a very good performance is achieved despite the fact that information is compressed 5-times stronger.

Since the training data were labelled, a confusion matrix can be computed. This is presented in Table IV. It is found that the SOM-SD classifies most classes at an accuracy well above 90%. Exceptions to this rule are the classes "6" and "11" which are often confused with each other. By considering the data properties as was shown in Figure 1, it is observed that the confusion occured between two classes which feature very similar structures. In addition, the worst class "6" is amongst the smallest in the dataset. This implies that the performance of the SOM-SD can be affected by imbalances in the dataset.

TABLE IV
CONFUSION MATRIX AS OBTAINED FROM THE SOM-SD WHEN USING THE TRAINING SET.

Label	1	10	11	2	3	4	5	6	7	8	9	Perf %
1	597	1	0	0	0	0	0	0	0	0	0	99.83
10	0	384	0	0	0	0	0	2	0	0	0	99.48
11	0	1	381	0	0	0	0	66	0	0	0	85.04
2	0	0	0	459	27	0	0	0	0	0	0	94.44
3	0	0	0	21	680	0	0	0	0	0	0	97.00
4	0	0	0	1	0	160	11	0	0	0	0	93.02
5	0	0	0	0	0	8	427	0	0	0	0	98.16
6	0	0	71	0	0	0	0	157	0	3	0	67.96
7	0	0	0	0	0	0	0	0	261	0	0	100.0
8	0	0	4	0	0	0	0	3	0	762	0	99.08
9	0	0	0	0	0	0	0	0	8	0	325	97.59

C. Comparison with alternative methods

The XML dataset used for the experiments has been used by other researchers in order to test and develop unsupervised classification, and clustering methods for structured documents. The performance comparison illustrated in Figure 4 shows that the SOM-SD method described in this paper outperformed all other methods which are known to have been applied to this dataset. A short summary of these alternative approaches is given as follows:

Candillier et al. [15] propose to transform the XML trees into sets of attribute-values, so as to be able to use standard vectorial-based clustering methods. The structural attribute-values they consider are a set of parent-child relations,

[4]The ratio between total number of (root) nodes, and the number of neurons activated by these nodes.

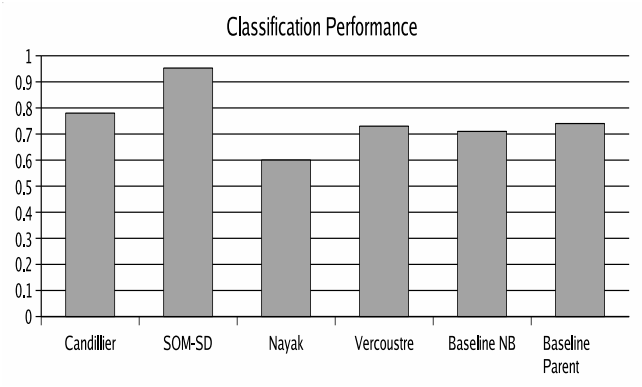

Fig. 4. Performances of unsupervised methods applied to the XML dataset.

the set of "next-sibling" relations, the set of paths starting from the root, and the arity of the nodes. XML documents represented according to these features were then clustered by the Statistical Subspace Clustering (SSC) algorithm.

Nayak and Xu [16] used a novel clustering algorithm named XCLS. XCLS exploits a so-called level structure representation where information such as element values and their occurrences and levels in the hierarchy defined by an XML document is represented. The clustering is then obtained by exploiting a similarity-based global criterion function, which allows it to incrementally assign/reassign documents to clusters, thus avoiding to compute the pairwise similarity between two individual documents.

Vercoustre et al. [17] represent XML trees via a set of their sub-paths, defined according to length, root beginning, and leaf ending. By representing these sub-paths as words, they are able to perform a step feature selection, based on information retrieval concepts, to reduce their number.

Finally, considering that sub-path representations are not independent, a k-means based clustering is obtained by considering distinct sets of sub-paths defined according to the sub-path length. The baseline models are constituted by a stochastic generative Naive Bayes model restricted to the generation of single nodes of the trees (Baseline NB), or a stochastic generative model where the generation of single nodes is conditioned on its parents (Baseline Parent).

VI. Conclusions

The clustering of graphs and sub-graphs can be a hard problem. This paper demonstrated that the clustering task of common types of graphs can be performed in linear time by using a neural network approach based on Self-Organizing Maps. In addition, it was shown that SOM-SD based networks can produce good results, and showed better efficiency over the standard SOM method in that relatively small maps are sufficient to encode possibly large sets of graph structured data with a good accuracy.

The training set used in this paper featured a wide variety of tree structured graphs. Unbalanced training sets are known to negatively impact the performance of many (supervised) neural network models. A more careful investigation into the effects of the imbalanced dataset on these unsupervised methods is left as a future task.

A more interesting task to investigate is the apparent robustness of the SOM-SD model to initial network conditions.We have no formal proof of convergence for the training procedure of the SOM-SD model, however experimental findings not discussed in this paper for lack of space show that SOM-SD produces an identical performance independently from the initial network conditions.

VII. Acknowledgments

The work presented in this paper received financial support from the Australian Research Council in form of a Linkage International Grant and a Discovery Project grant.

References

[1] M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, May 2003.

[2] M. Hagenbuchner, A. Sperduti, and A. Tsoi. Contextual processing of graphs using self-organizing maps. In *European symposium on Artificial Neural Networks*, Poster track, Bruges, Belgium, 27 - 29 April 2005.

[3] M. Hagenbuchner, A. Sperduti, and A. Tsoi. Contextual self-organizing maps for structured domains. In *Workshop on Relational Machine Learning*, pages pp. 46–55, 2005.

[4] K. Lagus, S. Kaski, and T. Kohonen. Mining massive document collections by the WEBSOM method. *Information Sciences*, Vol. 163/1-3:135-156, 2004.

[5] A. Rauber and D. Merkl. The SOMLib Digital Library System. In *3rd Europ. Conf. on Research and Advanced Technology for Digital Libraries (ECDL'99)*, Paris, France, September 22. - 24. 1999, Lecture Notes in Computer Science (LNCS 1696), Springer, 1999.

[6] M. Hagenbuchner and A. Tsoi. A supervised self-organizing map for structures. In *International Joint Conference on Neural Networks*, volume 3, pages 1923–1928, Budapest, Hungary, 25-29 July 2004.

[7] M. Hagenbuchner and A. Tsoi. A supervised training algorithm for self-organizing maps for structures. *Artificial Neural Networks in Pattern Recognition, Special Issue Pattern Recognition Letters*, 26(12):1874–1884, September 2006.

[8] B. Hammer, A. Micheli, A. Sperduti, and S. M. Recursive self-organizing network models. *Special issue on New Developments in Self-Organizing Systems, Neural Networks*, pages pp 1061–1085, October-November 2004.

[9] W. Kc, M. Hagenbuchner, and A. Tsoi. A distributed lightweight crawler, and some web statistics. In *The 15th International Conference on WWW*, 2006 (submitted).

[10] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.

[11] A. Micheli, D. Sona, and A. Sperduti. Contextual processing of structured data by recursive cascade correlation. *IEEE Transactions on Neural Networks*, Vol. 15(no 6):pp. 1396– 1410, November 2004.

[12] A. Micheli and A. Sperduti. Dealing with graphs using neural network. In *Workshop on Relational Machine Learning*, pages pp. 66–75, 2005.

[13] Web server survey, October 2005. Retrieved from http://-news.netcraft.com/archives/web_server_survey.html on 13/Oct/2005.

[14] F. Scarselli, A. C. Tsoi, M. Gori, and M. Hagenbuchner. Graphical-based learning environments for pattern recognition. In *SSPR/SPR*, pages 42–56, August 2004. keynote paper.

[15] L. Candillier, I. Tellier, and F. Torre. Transforming XML trees for efficient classification and clustering. In *INEX 2005 Workshop on Mining XML documents*, 2005.

[16] R. Nayak and S. Xu. XML documents clustering by structures with XCLS. In *INEX 2005 Workshop on Mining XML documents*, 2005.

[17] A.-M. Vercoustre, M. Fegas, S. Gul, and Y. Lechevallier. A Flexible Structured-based Representation for XML Document Mining. In *INEX 2005 Workshop on Mining XML documents*, 2005.