Clustering XML Documents Using Self-organizing Maps for Structures

M. Hagenbuchner[2], A. Sperduti[3], A.C. Tsoi[4], F. Trentini[1], F. Scarselli[1], and M. Gori[1]

[1] University of Siena, Siena, Italy
[2] University of Wollongong, Wollongong, Australia
[3] University of Padova, Padova, Italy
[4] Monash University, Melbourne, Australia

Abstract. Self-Organizing Maps capable of encoding structured information will be used for the clustering of XML documents. Documents formatted in XML are appropriately represented as graph data structures. It will be shown that the Self-Organizing Maps can be trained in an unsupervised fashion to group XML structured data into clusters, and that this task is scaled in linear time with increasing size of the corpus. It will also be shown that some simple prior knowledge of the data structures is beneficial to the efficient grouping of the XML documents.

1 Introduction

In many scientific and practical situations, there is often a need to visualise, if possible, the relationships, e.g. cluster formation, among high-dimensional data items. Kohonen's [6] self-organizing map (SOM) is one of the most well known methods to help users to achieve this goal. It was developed to help identify clusters in multidimensional, say, p-dimensional datasets. The SOM does this by effectively packing the p-dimensional dataset onto a q-dimensional display plane, where we assume for simplicity $q = 2$ throughout this paper. The SOM consists of a discrete display space with $N \times N$ grid points, each grid point is associated with a p-dimensional vector, often referred to in this paper, as an artificial neuron, or simply a neuron [1]. The contents of these vectors are updated with each presentation of samples from the p-dimensional original data set. The contents of these vectors encode the relationships (distances) among the p-dimensional data. The result is that data items that were "similar" or "close" to each other in the original multidimensional data space are then mapped onto nearby areas of the 2-dimensional display space. Thus SOM is a topology-preserving map as there is a topological structure imposed on the nodes in the network. A topological map is simply a mapping that preserves neighbourhood relations.

Thus, in the SOM, there are $N \times N$ grid points, or neurons, each neuron has an associated p-dimensional vector, often called a codebook vector. This codebook vector $\boldsymbol{m}$ has the same dimension as the i-th input vector $\boldsymbol{x}_i$. The neurons on the map are bound together by a topology, which is often either hexagonal or rectangular. In general, the SOM is trained on a set of examples in an unsupervised fashion as follows:

[1] This is called a neuron for historical reasons.

N. Fuhr et al. (Eds.): INEX 2005, LNCS 3977, pp. 481–496, 2006.

For every $\boldsymbol{x}_i$ 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.

After the best matching unit $\mathbf{m}_c$ is found, the codebook vectors are updated. $\mathbf{m}_c$ itself as well as its topological neighbours are moved closer to the input vector in the input space i.e. the input vector attracts them. The magnitude of the attraction is governed by a learning rate α and by a neighbourhood function $f(\Delta_{jc})$, where Δ_{jc} is the topological distance between $\mathbf{m}_c$ and $\mathbf{m}_j$. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to a specified learning rate function type. Along with the learning rate, the neighbourhood radius decreases as well. 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 α is a learning coefficient, and $f(.)$ is a neighbourhood function which controls the amount by which the weights of the neighbouring neurons are updated. The neighbourhood function can take the form of a Gaussian function $f(\Delta_{jc}) = \exp\left(-\frac{\|\mathbf{l}_j - \mathbf{l}_c\|^2}{2\sigma^2}\right)$, where σ is the spread, and $\mathbf{l}_c$ and $\mathbf{l}_j$ is the location of the winning neuron and the location of the j-th neuron respectively. Other neighbourhood functions are possible.

Equations (1) and (2) are computed for every input vector in the training set, and for a set number of iterations. It is shown in [7] that the strength of the SOM is in its ability to map high dimensional input data onto a low dimensional display space while preserving the topological relationships among the input data. The SOM is trained unsupervised, though some supervised approaches to SOM exist [4, 5, 7].

While there are extensions of the SOM algorithm to allow the processing of data sequences [7], this paper is concerned with more recent developments which extended the capabilities of the SOM towards the processing of graph structured information in a causal manner [1], and a more general approach which is capable of capturing contextual dependencies among the input data [2, 3].

This paper addresses the specific problem of grouping graph structured data into clusters. The task will be executed in an unsupervised fashion (i.e. during network training no knowledge is available about how graphs should be clustered) by using a Self-Organizing Map approach. A collection of XML formatted documents (belonging to the INEX competition dataset) will be used to evaluate the approaches. The performance of these self-organizing methods [2, 3] will be addressed in this paper.

This paper is organized as follows: an introduction to the processing of graphs using a SOM for data structures is given in Section 2. The contextual SOM-SD, capable of encoding more general types of graphs, is given in Section 3. Methods for measuring the performances of SOM-SD based models are defined in Section 4. Results produced when engaging the SOM-SD and the CSOM-SD to the clustering task is presented in Section 5. Finally some conclusions are drawn in Section 6.

2 The SOM for Structured Data

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 one at a time rather than by processing a graph as a whole. The network response to a given node v is a mapping of v on the display space. This mapping is called the *state* of v and contains the coordinates of the winning neuron. An input vector representation is created for every node in a graph G by concatenating a numeric data label $\boldsymbol{l}_v$ which may be attached to a node v with the *state* of each of the node's immediate offsprings such that $\boldsymbol{x}_v = [\boldsymbol{l}_v \ \boldsymbol{y}_{\mathrm{ch}[v]}]$, where $\mathrm{ch}[v]$ denotes the children of node v, and $\boldsymbol{y}_{\mathrm{ch}[v]}$ denotes the states or mappings of the children of v. The dimension of $\boldsymbol{x}$ is made constant in size by assuming a maximum dimension for $\boldsymbol{l}$ together with a maximum out-degree of a node. For nodes with less dimensions than the assumed, padding with a suitable value is applied. Since the initialization of $\boldsymbol{x}$ depends on the availability of all the children states, this dictates the processing of nodes in an inverse topological order (i.e. from the leaf nodes towards the root nodes), and hence, this causes information to flow in a strictly causal manner (from the leaf nodes to the root nodes).

A SOM-SD is trained in a similar fashion to the standard SOM with the difference that the vector elements $\boldsymbol{l}$ and $\boldsymbol{y}_{\mathrm{ch}}$ need to be weighted so as to control the influence of these components to a similarity measure. Equation (1) is altered to become:

$$c = \arg\min_j(\|(\boldsymbol{x}_v - \boldsymbol{m}_j)\boldsymbol{\Lambda}\|) \tag{3}$$

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 constants μ_1 and μ_2 control the influence of $\boldsymbol{l}_v$ and $\boldsymbol{y}_{\mathrm{ch}[v]}$ to the Euclidean distance in (3).

The rest of the training algorithm remains the same as that of the standard SOM. The effect of this extension is that the SOM-SD will map a given set of graphs, and all sub-graphs onto the same map. The SOM-SD includes the standard SOM and the SOM for data sequences as special cases.

Since the SOM-SD maintains its ability to cluster input data according to some topology, it is found that leaf nodes (which do not feature any outlinks) are mapped in well distinct areas compared to root nodes. Moreover, since the SOM-SD processes graphs in a causal manner, the root nodes are a representation of the graph as a whole.

The increased encoding capability of a SOM-SD has a drawback. While a single vector is sufficient to represent data in a standard SOM, in a SOM-SD the data is represented by a number of nodes within a graph. Since the SOM-SD maps all nodes, this implies an increased demand in the display space. In general, a SOM-SD requires larger maps in order to perform satisfactorily. This is not a major issue since the algorithm scales linearly in complexity with the size of the map, and the size of the dataset.

This paper will deploy the SOM-SD to cluster XML formatted documents into various clusters. The graphs extracted from the XML documents naturally form trees. Loops or un-rooted structures are not possible. While a SOM-SD is fully capable of encoding such data, this paper also addresses a further extension to the SOM algorithm which is capable of encoding more general types of graphs [2, 3]. This is performed in order

to provide an overview of Self-Organizing map methods which can be deployed to the task of clustering XML structured documents. In fact, this is a very recent extension which is applied for the first time to this real world learning problem.

3 The Contextual SOM-SD

With contextual SOM for graphs (CSOM-SD), the network input is formed by additionally concatenating the state of parent nodes and children nodes to an input vector such that $\boldsymbol{x}_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ch}[v]}\ \boldsymbol{y}_{\mathrm{pa}[v]}]$, where $\boldsymbol{y}_{\mathrm{pa}[v]}$ are the states of the parent nodes and $\boldsymbol{y}_{\mathrm{ch}[v]}$ are the states of the children nodes. The problem with this definition is that a circular functional dependency is introduced between the connected vertices v and $\mathrm{pa}[v]$, and so, neither the state for node v nor the state of its parents $\mathrm{pa}[v]$ can be computed. One possibility to compute these states could be to find a joint stable fix point to the equations involving all the vertices of a structure. This could be performed by initializing all the states with random values and then updating these initial states using the above mentioned equations, till a fixed point is reached. Unfortunately, there is no guarantee that such a fixed point would be reached. Moreover, even if sufficient conditions can be given over the initial weights of the map to guarantee stability, i.e. the existence of the fixed point, there is no guarantee that training will remain valid on such sufficient conditions over the weights.

A (partial) solution to this dilemma has been proposed in [2]. The approach is based on an K-step approximation of the dynamics described above: Let

$$\boldsymbol{y}^t = h(\boldsymbol{x}_v^{t-1}), t = 1, \ldots, K \tag{4}$$

where $h(\cdot)$ computes the state of node v by mapping the input $\boldsymbol{x}_v^{t-1}$, and $\boldsymbol{x}_v^{t-1} = [\boldsymbol{l}_v\ \boldsymbol{y}_{\mathrm{ch}[v]}^{t-1}\ \boldsymbol{y}_{\mathrm{pa}[v]}^{i-1}]$. The algorithm is initialized by setting $\boldsymbol{y}_{\mathrm{ch}[v]}^0 = \boldsymbol{y}_{\mathrm{pa}[v]}^0 = k$, where $k = [-1, -1]$, an impossible winning coordinate. In other words, the approach iteratively recomputes the states of every node in a graph K-times. Then, the network input can be formed by setting $\boldsymbol{x}_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ch}[v]}^K\ \boldsymbol{y}_{\mathrm{pa}[v]}^K]$. A suitable value for K could be, for instance, the maximum length of any path between any two nodes in the graph. Although such a choice does not guarantee the full processing of contextual information due to possible latency in the transfer of contextual information from one vertex of the structure to its neighbors vertices, this value for K seems to be a good tradeoff between contextual processing and computational cost.

Training is performed similar to the training of SOM-SD with the difference that $\boldsymbol{\Lambda}$ is now a $n \times n$ matrix, $n = \dim(\boldsymbol{x})$ with $\lambda_{m+1,m+1} \cdots \lambda_{n,n}$ set to the constant μ_3. All other elements in $\boldsymbol{\Lambda}$ are the same as defined before.

Note that this approach is a generalization of a CSOM-SD which operates on undirected graphs. With un-directed graphs, each node v has a set of neighbors such that the network input would be $\boldsymbol{x}_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ne}[v]}^K]$, where $\mathrm{ne}[v]$ denotes the state of neighboring nodes. Accordingly, $\mu_2 = \mu_3$ in $\boldsymbol{\Lambda}$ in Eq. (3).

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

Step 0. Initialize all $\boldsymbol{y}$ with k, where $k = [-1, -1]$ the impossible output coordinate.

Step 1. For every $v \in G_i$ compute $\boldsymbol{y}^t = h(\boldsymbol{x}_v^{t-1})$, where $\boldsymbol{x}_v^{t-1} = [\boldsymbol{l}_v \ \boldsymbol{y}_{\mathrm{ch}[v]}^{t-1} \ \boldsymbol{y}_{\mathrm{pa}[v]}^{t-1}]$. Repeat this step K times, where K is the maximum path length between the two most distant nodes in G_i, and G_i is the i-th graph in the training set. Apply this step to all graphs in the training set.

Step 2. Choose a node v from the training set, initialize $\boldsymbol{x}_v^K = [\boldsymbol{l} \ \boldsymbol{y}_{\mathrm{ch}[v]}^K \ \boldsymbol{y}_{\mathrm{pa}[v]}^K]$, 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:

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

Step 3. Update network parameters as follows:

$$\Delta\mathbf{m}_i = \alpha(t)g(\Delta_{ir})(\mathbf{m}_i - \mathbf{x}_v^K) \tag{6}$$

where t is the current training iteration, α is a learning rate which gradually decreases to zero, $g(.)$ is the neighborhood function depending on Δ_{ir} which is the topological distance between neuron i and neuron r. This step is identical to the traditional SOM updating step shown in Eq. (2). Repeat Step 2 and Step 3 for every node in the training set.

The algorithm cycles through Steps 1 to 3 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 CSOM-SD. This is performed by using a set of data in place of the training set, and by executing Step 0 to Step 2 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.

4 Performance Measures

It is evident that a simple quantization error is an insufficient indicator of the performance of a SOM-SD or a CSOM-SD since such an approach neglects to take into account the fact that structural information is being mapped. In fact, there are a number of criteria with which the performance of a SOM-SD or a CSOM-SD can be measured. These are performance indicators on the clustering performance, the mapping precision, and the compression ratio. The clustering performance shows how well data are grouped together on the map, the mapping precision shows how accurately structural information is encoded in the map, and the compression ratio indicates the degree of network utilization. In addition, if target labels are available then the network can also be evaluated on the classification performance, and the retrieval capability.

Retrieval capability (R): This reflects the accuracy of retrieved data from the various SOM models. This can be computed quite simply if 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 will focus our attention just on the root of the tree. Thus, with r_j we will refer to the input vector for SOM,

SOM-SD or CSOM-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): This 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,

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

where N is the number of documents (graphs) in the test set. Values of C and R can range within $(0 : 1]$ where values closer to 1 indicate a better performance.

Clustering performance (P): A more sophisticated approach is needed to compute the ability of a SOM-SD or a CSOM-SD to suitably group data on the map. In this paper 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:

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

   where $\mathcal{N}_i = \{v | v \in \text{ne}[i], win(v) \neq \emptyset\}$.
3. The overall neural network performance is then given by:

$$P = \frac{\sum_i P_i}{W}.$$

A performance value close to 1 indicates a perfect grouping, while a value closer to 0 indicates a poor clustering result. Thus, this measure indicates the level of disorder inside a SOM-SD or CSOM-SD.

Structural mapping precision (e and E): These indices measure how well structural (e) and contextual structural (E) information are encoded in the map. A suitable method for computing the structural mapping precision was suggested in [2]. In this case, just the skeleton of the trees is considered, i.e. the information attached to vertices is disregarded, and only the topology of the trees is considered. Notice that these measures do not consider the information about the class to which an XML document (i.e., a tree) belongs. 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. These two measures e and E are respectively computed as follows

$$e = \frac{1}{N}\sum_{i=1,n_i\neq 0}^{N}\frac{m_i}{n_i} \qquad \text{and} \qquad E = \frac{1}{N}\sum_{i=1,n_i\neq 0}^{N}\frac{M_i}{n_i}$$

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. Similarly, M_i is the greatest number of identical complete trees which are associated with the sub-structure 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, and E indicates the quality of the contextual mapping process. Values of e and 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.

Compression ratio: This is the ratio between the total number of root nodes in the training/test set, and the number of neurons actually activated by root nodes in the training/test set. The higher the compression, the fewer the number of neurons are involved in the mapping process. Extremely high or extremely low compression ratios can indicate a poor performance. The compression ratio can vary between 0 and N, where N is the number of root nodes in the training/test set.

5 Clustering Results

The corpus (`m-db-s-0`) considered consists of $9,640$ XML formatted documents which were made available as part of the INEX Initiative (INitiative for the Evaluation of XML Retrieval), and was obtained via the Web site: *http://xmlmining.lip6.fr*. Each of the XML formatted documents describes an individual movie (e.g. the movie title, list of actors, list of reviewers, etc.). It was built using the IMDB database. 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. Note that the target labels are used solely for testing purposes, and hence, are ignored during network training.

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). There were a total of 197 different tags in the dataset.

While for the SOM-SD and CSOM-SD there is no specific need to pre-process this set of graphs, 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 the experiments. Dimension reduction was achieved by consolidating XML tags as follows: Repeated sequences of tags within the same level of a structure are consolidated. 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 is inspired by operations in regular expressions. For example, the expression $(ab)^n$ can be simulated by repeatedly presenting ab n-times. Hence, it suffices to process the consolidated structure n times. There were many trees which exhibited such repeated sequences of tags. The consequence of this pre-processing step is that the maximum outdegree is reduced to just 32.

A further dimension reduction is achieved by collapsing sequences 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 further to `<A&b&c>`. Since the deepest graph is of depth 3, this implies that the longest sequence that can be collapsed is of length 3. This pre-processing step reduces the total number of nodes in the dataset to $247,140$.

A unique identifier (ID) is associated with each of the 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 will be set to zero.

The resulting dataset consists of $4,820$ graphs containing a total of $124,360$ nodes (training set), and 4,811 graphs containing a total of $122,780$ nodes (test set). The training set was analysed for its statistical properties, results are illustrated in Figure 1. It is observed that the training set is unbalanced. For example, the table on the left of Figure 1 shows that there are only 172 samples of the pattern instance denoted by "4" but over 700 instances of patterns from the instance denoted by "3". Also, the 3-D plot in Figure 1 shows that the distribution of outdegrees can vary greatly. For example, there is only one instance in the pattern class denoted by "8" which has an outdegree of 10 while there are over 270 instances for the same pattern class with outdegree 5. There are also a number of pattern classes which are similar in features such as class "10" and class "11" which are of similar size and are of similar structure.

There are $2,872$ unique sub-structures in the training set. This is an important statistical figure since it gives an indication to how much more information is provided to a SOM-SD when compared to the flat vectors used for the SOM. And hence, the larger the number of unique sub-structures in the training set, the greater the potential diversification in the mapping of the data will be. Similarly, there are $96,107$ unique nodes in different contextual configurations in the training set. This shows that the CSOM-SD is provided with a greater set of diverse features in the training set, and

Label	Frequency
1	598
10	386
11	448
2	486
3	701
4	172
5	435
6	231
7	261
8	769
9	333
Total	4820

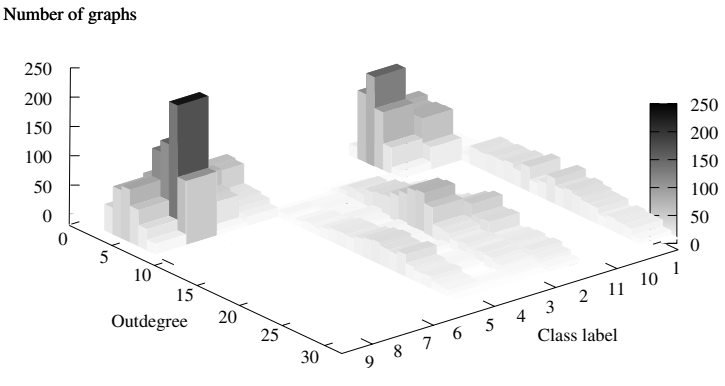

Fig. 1. Properties of the training set: The table (left) shows the number of graphs in each of the 11 classes. The plot (right) shows the distribution of outdegrees in the dataset. Shown are the number of graphs (z-axis) which have a given outdegree (y-axis) and belong to a given class (x-axis).

hence, may be capable to diversify in the mapping of the data even further. Thus, this dataset provides a challenging learning problem on which various SOM models will be tested.

All SOMs illustrated in this section used a hexagonal topology, and a Gaussian neighborhood function. For the SOM-SD and the CSOM-SD, when generating the input vectors $\boldsymbol{x}_i$ for nodes with less than the maximum outdegree, padding was performed using the impossible coordinate $[-1, -1]$.

The standard 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 within a document. Thus, the input vectors for the SOM are 197 dimensional containing the complete set of information about the XML tags in a document but do not contain any information about the topological structure between the XML tags.

Thus, the SOM is trained on relatively few high-dimensional data vectors while the (C)SOM-SD is being trained on a large number of nodes which are represented by a relatively small size vectors. For the SOM we chose $64 \times 48 = 3,072$ as the size of the network. The total number of network parameters for the SOM is $3,072 \times 197 = 605,184$. Since the codebook dimensions for the SOM-SD is $3 + 32 \times 2 = 67$, this implies that a SOM-SD needs to feature at least $9,033$ codebooks to allow a fair comparison. Accordingly, the CSOM-SD should feature at least $8,771$ neurons. However, since the SOM-SD (and to an even greater extent the CSOM-SD) are to encode a larger feature set which includes causal (contextual) information about the data, this implies that the SOM-SD (CSOM-SD) will potentially diversify the mapping of the data to a greater extent than a SOM would do. Hence, this would justify the choice of even larger networks for the SOM-SD and CSOM-SD respectively for the comparisons. However, we chose to use the network sizes as indicated in Table 1 as these suffice to illustrate the principal properties of the models.

Table 1. Network parameters used for the training procedure

	size	# iterations	$\alpha(0)$	$r(0)$	μ_1	μ_2	μ_3
SOM	64×48	32	1.0	4	1.0	–	–
SOM-SD	110×81	62	1.0	38	0.11	0.89	–
CSOM-SD	110×81	12	0.7	15	0.11	0.88	0.01 [2]

A number of SOMs, SOM-SDs, and CSOM-SDs were trained by varying the training parameters, and initial network conditions. We used the classification measure C as a general benchmark on which to optimize the performance of the various models. A total of 56 experiments were executed for each of the SOM models, and every experiment was repeated 10 times using a different random initialization of the map as a starting point. The experiments varied the following training parameters: number of training iterations i, initial neighborhood radius $r(0)$, initial learning rate $\alpha(0)$, and the weight values μ (in this order). The set of training parameters which maximised the classification performance of the three models is shown in Table 1. It is observed that the SOM-SD required more training iterations and a larger initial neighborhood radius to achieve optimum classification performance (on the training set). It was also observed that the classification performance of the CSOM-SD improved with smaller values for μ_3 reaching an optimum for $\mu_3 = 0.0$. However, setting μ_3 to zero would reduce the CSOM-SD to a SOM-SD, and hence, would be an unsuitable choice for the comparisons. Further details regarding this observation are given below.

The performances of the three SOM models are illustrated in Table 2. The performance indices are those as defined in Section 4. From Table 2 it can be seen that a standard SOM is able to classify over 90% of patterns in the training set correctly despite of no information about the underlying causal or contextual configuration of XML tags is provided to the training algorithm. However, it was found that the SOM generalizes poorly. In comparison, the SOM-SD improved the classification rate by a noticeable amount, and was able to generalize over unseen data very well. As is seen from the compression ratio Z, the performance increase of the SOM-SD comes despite a doubling of the compression ratio. This is a clear evidence that causal information about the order of XML tags allows to a.) diversify the mapping of nodes to a considerably larger extend, and b.) the diversification in the mappings can result in an overall im-

Table 2. Best results obtained during the experimentation with maps of size 64×48 (SOM), and for maps of size 110×81 (SOM-SD and CSOM-SD)

	train set						test set					
	C	R	P	e	E	Z	C	R	P	e	E	Z
SOM	90.5%	0.90	0.73	1.0	1.0	2.45	76.5%	0.92	0.73	1.0	1.0	2.45
SOM-SD	92.5%	0.92	0.78	0.77	0.50	5.13	87.3%	0.93	0.79	0.76	0.50	4.9
CSOM-SD	83.9%	0.87	0.73	0.91	0.30	8.53	78.6%	0.88	0.71	0.90	0.37	8.54

[2] Smallest non-zero value tried. Setting $\mu_3 = 0.0$ resulted in a better classification performance but would reduce the CSOM-SD to a SOM-SD.

provement of the classification or clustering performances. In contrast, the inclusion of contextual information did not help to improve on the classification performance as it is seen from the results obtained from the CSOM-SD. It is found that contextual information helped to diversify the mapping of nodes by almost double when compared to the SOM-SD. This is indicated by the larger compression ratio. Thus, it is evident that a correct classification of the graphs in the dataset is independent to contextual information about the XML tags within the original documents. Paired with the greater data compression which is the result of a greater diversification in the mapping of nodes, this produced a relative overall reduction in classification performance for the CSOM-SD, and explains the observation that the performance optimum of the CSOM-SD is at $\mu_3 = 0$.

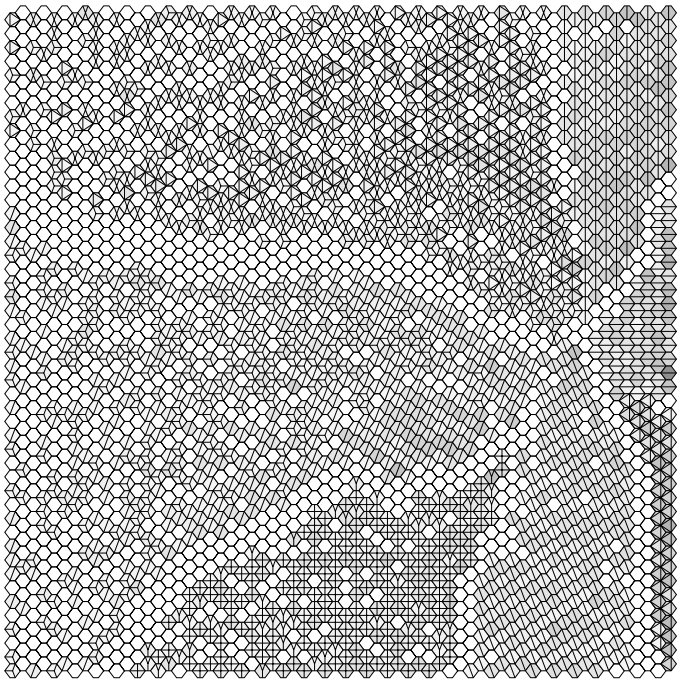

Fig. 2. The mapping of the training vectors on a standard SOM

In addition, it is observed that a CSOM-SD performs worse on the performance measure E than a SOM-SD. This is a result which arose out of the fact that the experiments were to optimize the classification performance C. It was found that a CSOM-SD improves on C when using $\mu_3 \to 0$. However, setting $\mu_3 = 0$ would reduce the CSOM-SD to a SOM-SD and would have denied us from making a comparison between the models. Instead, we chose a small value for μ_3 so as to allow such comparisons, and still produce reasonable classification performances. Using a very small μ_3 reduces the impact of contextual information to the training algorithm. Paired with the increased

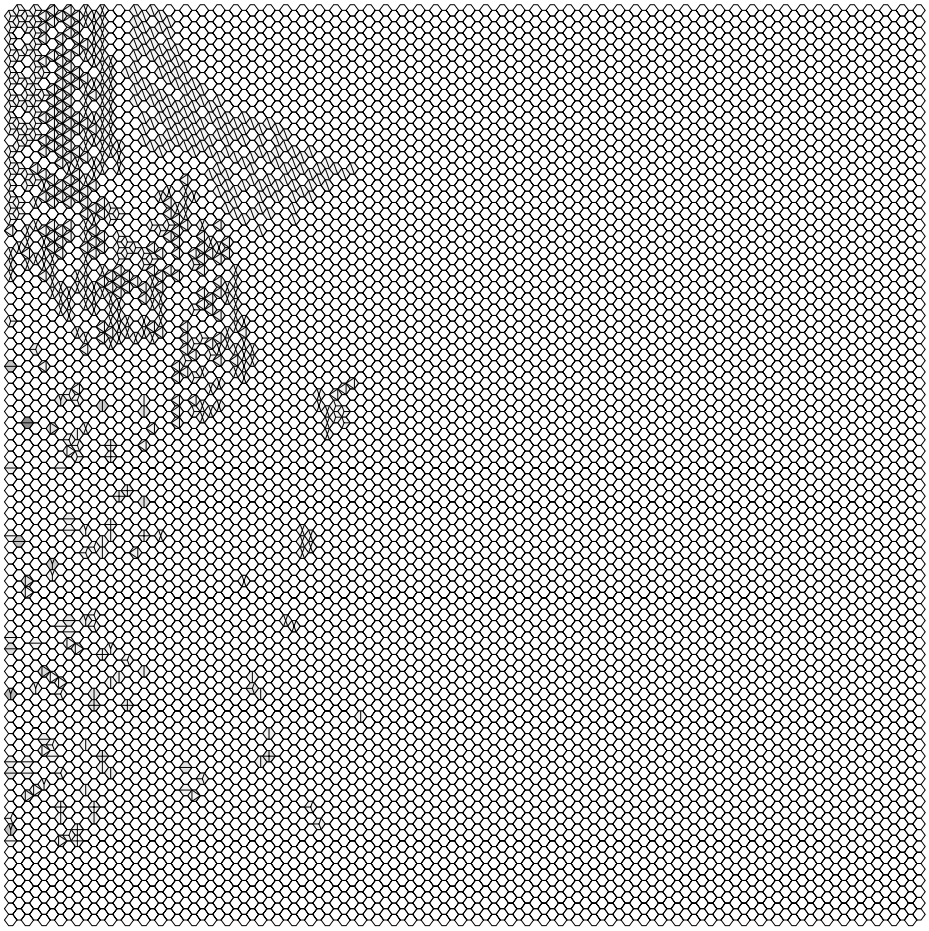

Fig. 3. The mapping of root nodes (training set) on the SOM-SD

compression ratio in the mapping of root nodes, this resulted in a relative decrease in the performance on E. Note that the standard SOM performed at $e = E = 1$. This is due to the fact that a SOM handles the simplest type of data structures (viz. single nodes). These render all structures in the dataset identical, resulting in the observed performance values.

A closer look at the mapping of (training) data is made in the standard SOM Figure 2. The hexagons in Figure 2 refer to the neurons on the map. The brightness of the grid intersection represents the number of training data which are assigned to the grid point due to their closeness in the original input space. Thus by examining the brightness in the grid, it is possible to gain an appreciation of the way the given training dataset can be grouped together, according to their closeness in the original input space. Every neuron is also filled in with a pattern indicating the class that most frequently activated the neuron. There are 11 different fill in patterns

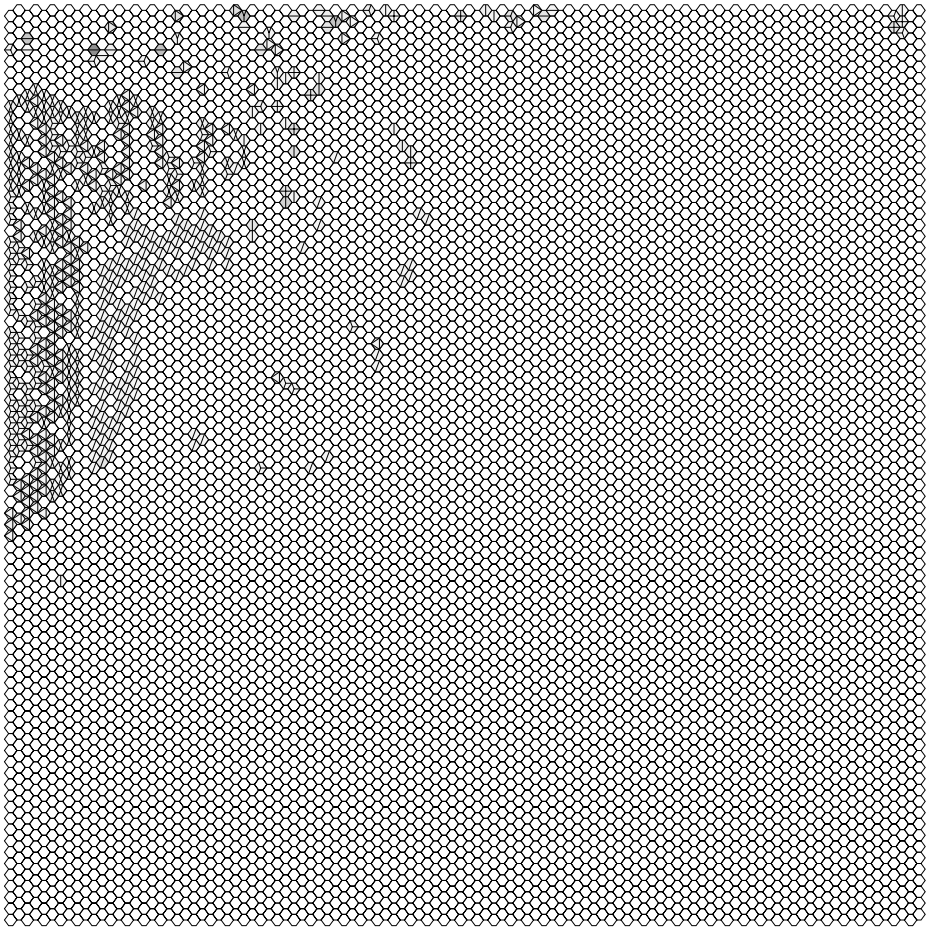

Fig. 4. The mapping of root nodes (training set) on the CSOM-SD

for the 11 possible classes. Neurons which are not filled in are not activated by any vector in the training set. It can be observed 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 may indicate a good result since the presence of such border regions should allow for a good generalization performance; a statement which could not be confirmed when evaluating the test set.

In comparison, the mapping of root nodes in the training set on a trained SOM-SD is shown in Figure 3. Neurons which are not filled in are either not activated by a root 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

Table 3. Confusion table as produced by the best SOM when using the training set

Label	1	10	11	2	3	4	5	6	7	8	9	Perf.
1	598	0	0	0	0	0	0	0	0	0	0	100.0%
10	0	339	33	0	0	0	0	14	0	0	0	87.8%
11	0	69	350	0	0	0	0	29	0	0	0	78.1%
2	0	0	0	362	124	0	0	0	0	0	0	74.4%
3	0	0	0	29	672	0	0	0	0	0	0	95.8%
4	2	0	0	0	0	87	83	0	0	0	0	50.5%
5	0	0	0	0	0	16	419	0	0	0	0	96.3%
6	1	47	12	0	0	0	0	171	0	0	0	74.0%
7	0	0	0	0	0	0	0	0	260	0	1	99.6%
8	0	0	0	0	0	0	0	0	0	769	0	100.0%
9	0	0	0	0	0	0	0	0	0	0	333	100.0%

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. This observation confirms the experimental findings which show that the SOM-SD will be able to generalize well.

Table 4. Confusion table as produced by the best SOM-SD when using the training set

Label	1	10	11	2	3	4	5	6	7	8	9	Perf.
1	590	1	0	3	4	0	0	0	0	0	0	98.66%
10	0	384	0	0	0	0	0	0	0	0	2	99.48%
11	0	0	363	0	0	0	0	59	0	25	1	81.03%
2	3	0	0	440	16	8	19	0	0	0	0	90.54%
3	4	0	0	10	686	1	0	0	0	0	0	97.86%
4	1	0	0	86	5	65	15	0	0	0	0	37.79%
5	0	0	0	4	0	2	429	0	0	0	0	98.62%
6	0	0	63	0	0	0	0	150	0	18	0	64.94%
7	0	0	0	0	0	0	0	0	257	4	0	98.47%
8	0	0	5	0	0	0	0	3	0	761	0	98.96%
9	0	0	0	0	0	0	0	0	0	0	333	100.0%

Figure 4 gives the mapping of the root nodes as produced by the CSOM-SD. Again, it is found that the largest portion of the map is filled in by neurons which are either not activated or are activated by nodes other than the labelled root nodes. Clear clusters are formed which are somewhat smaller in size when compared to the SOM-SD case. This illustrates quite nicely that the CSOM-SD is compressing the “root” data considerably more strongly than the SOM-SD since contextual information is also encoded which requires additional room in the map. Nevertheless, the observation confirms that the CSOM-SD will also be able to generalize well even though some of the performance indices may be worse than when compared to a SOM-SD of the same size. This can be expected since the CSOM-SD compresses the “root” data more strongly.

Table 5. Confusion table as produced by the best CSOM-SD when using the training set

Label	1	10	11	2	3	4	5	6	7	8	9	Perf.
1	592	1	0	0	5	0	0	0	0	0	0	99.00%
10	0	355	0	0	0	0	0	0	0	0	31	91.97%
11	0	1	274	0	0	0	0	17	0	156	0	61.16%
2	0	0	0	425	17	16	28	0	0	0	0	87.45%
3	15	0	0	7	679	0	0	0	0	0	0	96.86%
4	0	0	0	78	10	68	16	0	0	0	0	39.53%
5	0	0	0	8	3	3	421	0	0	0	0	96.78%
6	0	2	46	0	0	0	0	89	0	94	0	38.53%
7	0	0	0	0	0	0	0	0	261	0	0	100.0%
8	0	0	103	0	0	0	0	2	0	664	0	86.35%
9	0	0	0	0	0	0	0	0	0	0	333	100.0%

A more detailed look at the classification of the data is made in Table 3, Table 4, and Table 5 which give the confusion matrices as produced by the SOM, SOM-SD, and the CSOM-SD respectively. It is seen that all three models perform best on classes which are relatively large in size (compare with Figure 1) The poorest classifications are observed for the smallest classes. This is particularly true for the classes labelled 4 and 6, which in addition of being the smallest classes in the dataset, also share features that are similar to the classes 2 and 11. This shows that the performance of all three models is affected by unbalances in the feature space presented in the training set.

The experiments presented in this paper were executed on 2GHz Intel based CPUs. Training times varied from $2 - 12$ hours depending on the SOM model and training parameters used. Once trained, data retrieval generally took only a few minutes.

6 Conclusions

The clustering of graphs and sub-graphs can be a hard problem. This paper demonstrated that the clustering task of general 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 performances even if the map is considerably smaller than the size of the training set. Using larger maps will generally allow to improve the performance further though this was not illustrated in this paper.

Specifically, it was shown that the given learning problem depends on the availability of causal information about the XML tags within the original document in order to produce a good grouping or classification of the data. The incorporation of contextual information did not help to improve on the results.

The training set used in this paper featured a wide variety of tree structured graphs. We found that most graphs are relatively small in size, only few graphs were either very wide or featured many nodes. This creates imbalances in features represented in a training set which is known to negatively impact the performance of a neural network.

Similarly it is true when considering the way we generated data labels for the nodes. An improvement of these aspects (i.e. balancing the features in the training set, using unique labels which are equiv-distant to each other) should help to improve the network performances. An investigation into the effects of these aspects is left as a future task.

Furthermore, it was shown that the (C)SOM-SD models map graph structures onto a finite regular grid in a topology preserving manner. This implies that similar structures are mapped onto nearby areas. As a consequence, these SOM models should be suitable for inexact graph matching tasks. Such applications are considered as a future task.

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 *Relational Machine Learning*, pages pp. 46–55, 2005.
4. 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.
5. 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.
6. T. Kohonen. *Self-Organisation and Associative Memory*. Springer, 3rd edition, 1990.
7. T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.