Efficient Clustering of Structured Documents Using Graph Self-Organizing Maps

Markus Hagenbuchner[1], Ah Chung Tsoi[2], Alessandro Sperduti[3], and Milly Kc[1]

[1] University of Wollongong, Wollongong, Australia
{markus,millykc}@uow.edu.au
[2] Hong Kong Baptist University, Hong Kong
act@hkbu.edu.hk
[3] University of Padova, Padova, Italy
sperduti@math.unipd.it

Abstract. Graph Self-Organizing Maps (GraphSOMs) are a new concept in the processing of structured objects using machine learning methods. The GraphSOM is a generalization of the Self-Organizing Maps for Structured Domain (SOM-SD) which had been shown to be a capable unsupervised machine learning method for some types of graph structured information. An application of the SOM-SD to document mining tasks as part of an international competition: Initiative for the Evaluation of XML Retrieval (INEX), on the clustering of XML formatted documents was conducted, and the method subsequently won the competition in 2005 and 2006 respectively. This paper applies the GraphSOM to the clustering of a larger dataset in the INEX competition 2007. The results are compared with those obtained when utilizing the more traditional SOM-SD approach. Experimental results show that (1) the GraphSOM is computationally more efficient than the SOM-SD, (2) the performances of both approaches on the larger dataset in INEX 2007 are not competitive when compared with those obtained by other participants of the competition using other approaches, and, (3) different structural representation of the same dataset can influence the performance of the proposed GraphSOM technique.

1 Introduction

In general, structured objects can be described by graphs, e.g. acyclic directed graphs, cyclic graphs, un-directed graphs, etc. Graphs are generalizations of the more common vectorial representation as a graph can encode relationships among structural elements of objects, or provide contextual information concerning data points which may be described in vectorial form.

The machine learning community recognizes that any model which is capable of dealing with structured information can potentially be more powerful than approaches which are limited to the processing of vectorial information. This observation motivates us to develop machine learning methods which are capable of encoding structured information. A noteworthy result of such efforts is the Graph Neural Network (GNN) which is a supervised machine learning method capable of learning from a set of graphs [1]. The GNN is probably one of the more powerful supervised machine learning methods devised since it is capable of processing arbitrary types of graphs, e.g. cyclic, un-directed, where (numeric) labels may be attached to nodes and links in the graph. In other words, a GNN can encode the topology of a given set of graph structures as well as the numerical information which may be attached to the nodes or links in the graph.

Supervised machine learning methods require the availability of target information for some of the data, and are typically applied to tasks requiring the categorization or approximation of information. Unsupervised machine learning methods have no such requirement on the target information, and are typically applied to tasks requiring the clustering or segmentation of information. Unsupervised machine learning techniques for graph structured information are often based on the well-known Self-Organizing Maps [2] and are called Self-Organizing Maps for Structures (SOM-SD) [3]. While a SOM-SD is restricted to the processing of bounded positional acyclic directed graphs, it is found that this is sufficient for many practical applications. An application of the SOM-SD to the clustering of XML structured scientific documents at an international competition on document mining: Initiative for the Evaluation of XML Retrieval (INEX) was conducted, and this technique won in the year 2005 [4].

The introduction of a contextual SOM-SD (CSOM-SD) extended the capabilities of the SOM-SD model to allow for the contextual processing of bounded positional directed graphs which may contain cycles [5]. The SOM-SD and CSOM-SD were again applied to document mining tasks at INEX 2006. Both approaches produced winning results albeit amongst a fairly small group of participants [6]. However, it was observed that the CSOM-SD has a nonlinear computational complexity; in most cases, this is close to quadratic. This would limit the application of the CSOM-SD technique to small datasets. In this paper we will use a modification of the CSOM-SD method which we called Graph Self-Organizing Map (*GraphSOM*) [7], which (1) has a linear computational complexity, and (2) allows the encoding of more general types of graphs which may be unbound, cyclic, undirected, and non-positional. This paper demonstrates the efficiency and capability of the GraphSOM technique. Comparisons are made with the existing machine learning method: SOM-SD [3].

A drawback of the SOM-SD is that it does not scale well with the size of a graph. In particular, the computational demand increases quadratically with the maximum outdegree of any node in the dataset. Moreover, the SOM-SD requires prior knowledge of the maximum outdegree, and hence, has limitations in problem domains where the maximum outdegree is not known a priori, or for which the outdegree cannot be fixed a priori. The GraphSOM addresses these shortcomings through a modification of the underlying learning procedures [7]. The effect is that the computational complexity is reduced to a linear one, and, as a side effect, allows the processing of much more general types of graphs which may feature loops, undirected links, and for which the maximum outdegree is not known a priori. A more detailed theoretical analysis of the computational complexity of the GraphSOM is presented in [7].

This paper is structured as follows: Section 2 introduces to the SOM-SD and GraphSOM, and offers some comparisons. The experimental setting and experimental findings are presented in Section 3. Conclusions are drawn in Section 4.

2 Self-Organizing Maps

This section gives an overview to how unsupervised learning of graph structured information is achieved when using Self-Organizing Map techniques[1]. Another unsupervised neural network method capable of learning from graphs is [9] which realizes an auto-associative memory for graph structures, and hence, is quite different to clustering methods discussed in this paper. An alternative approach is constituted by the use of standard clustering methods in conjunction with metrics explicitly defined on graphs or induced by kernels for graphs, such as in [10] where a version of SOM that uses a version of the edit distance for graphs is presented. We are not aware of papers where kernels for undirected and unbounded graphs are used within a traditional clustering method. Finally, MLSOM [11] is an improved self-organizing map for handling tree structured data and cannot deal with graphs.

Traditionally, Self-Organizing Maps (SOMs) are an extension of the Vector Quantization technique [2] in which prototype units are arranged on an n-dimensional lattice. Each element of the lattice is associated with one unit which has adjustable weights. SOMs are trained on vectorial inputs in an unsupervised fashion through a suitable adjustment of the associated weights of the best matching prototype unit and its neighbors. Training is repeated for a number of iterations. The result is a topology preserving mapping of possibly high-dimensional data onto a lower dimensional one, often 2-dimensional mapping space. In practice, SOMs have found a wide range of applications to problem domains requiring the clustering or projection onto lower dimensional space of unlabeled high dimensional vectors. Self-Organizing Maps (SOMs) are a classic concept in machine learning allowing the mapping of high-dimensional data onto a low-dimensional display space [2].

An extension to data which can be described by graphs was made with the introduction of the SOM-SD [3]. With SOM-SD it has become possible for the first time to have an unsupervised machine learning method capable of mapping graph structures onto a fixed dimensional display space.

2.1 Self-Organizing Maps for Structured Data

Approaches to enable SOMs to map graph structured information were proposed relatively recently in [3,8]. The approach in [3] extends the classical SOM method by processing individual nodes of a graph, and by incorporating topological information about a node's offsprings in a directed acyclic graph. Nodes in the graph can be labeled so as to encode properties of objects which are represented

[1] This section does not contain any new material, but simply pulls together information which were published by us [3,7,8] in a coherent and self consistent manner to explain the basic motivation of using GraphSOM in this paper.

by the node. One of the main advantages is that the SOM-SD is of linear computational complexity when processing graphs with a *fixed* out-degree, and hence, the SOM-SD is capable of performing tasks such as clustering of graphs and subgraphs in linear time. The SOM-SD is an extension of the standard SOM in that the network input is formed through a concatenation of the node label with the mappings of each of the node's offsprings. This implies that the SOM-SD is restricted to the processing of ordered acyclic graphs (ordered trees), and requires that the trees have a fixed (and relatively small) outdegree. The computational complexity of the SOM-SD grows quadratically with the out-degree[2], and hence, the processing of trees with a large outdegree becomes quickly a very time consuming task. Moreover, the processing of nodes in a tree must be performed in an inverse topological order so as to ensure that the mapping of child nodes is available when processing a parent node.

The approach was extended through the introduction of CSOM-SD [8]. The CSOM-SD incorporates topological information on all the node's neighbors, and hence, the method is capable of processing undirected and cyclic graphs. Both approaches include the standard SOM as a special case, and when applied to graphs, are restricted to learning domains for which the upper bound of any node's connectivity (outdegree) is known a priori (e.g. the maximum number of neighbors for any node in a graph is known a priori). It was found that the computational demand for learning problems involving a high level of connectivity can be prohibitively high for both methods.

In the following, we will explain some of the basic mechanisms of SOM, and SOM-SD as a prelude on the modifications introduced in GraphSOM [7] later. Let us explain the underlying procedures as follows: The basic SOM [2] consists of a q-dimensional lattice of *neurons* representing the display space. Every neuron i of the map is associated with an n-dimensional codebook vector $\mathbf{m}_i = (m_{i1}, \ldots, m_{in})^T$, where T transposes the vector. Figure 1 gives an example of a simple SOM. The neurons are shown with a hexagonal neighborhood relationship; the most commonly used arrangement. This hexagonal neighborhood is used in the training of the SOM.

The SOM is trained by updating the elements of $\mathbf{m}_i$ as follows:

Step 1: One sample input vector $\mathbf{u}$ is randomly drawn from the input data set and its similarity to the codebook vectors is computed. When using the Euclidean distance measure, the winning neuron is obtained through:

$$r = \arg\min_i \|\mathbf{u} - \mathbf{m}_i\| \tag{1}$$

[2] As is shown in [7], the computational demand of a SOM-SD is $Nk(p + qn)$, where N is the total number of nodes in the data set, k is the number of neurons on the map, p is the dimension of the data label attached to the nodes in a graph, q is the dimension of the map (typically $q = 2$), and n is the maximum number of neighbors of any node in a graph. N, k, and n are often interdependent. An increase in N, p, or n often requires a larger mapping space k. In many data mining applications, $n >> k$ which in turn can require a large k. Thus, the computational complexity for large scale learning problems is close to a quadratic one.

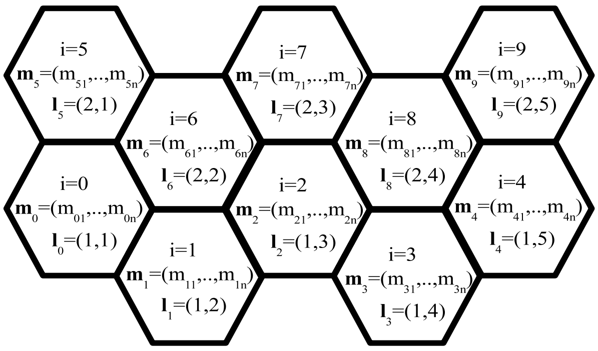

Fig. 1. A simple 2-dimensional map of size 2 × 5. Each hexagon marks a neuron. Number, associated codebook vector, and coordinate values for each neuron is shown.

Step 2: $\mathbf{m}_r$ itself as well as its topological neighbours are moved closer to the input vector in the input space. The magnitude of the attraction is governed by the learning rate α and by a neighborhood function $f(\Delta_{ir})$, where Δ_{ir} is the topological distance between $\mathbf{m}_r$ and $\mathbf{m}_i$. Here topological distance is used to described the distance between the neurons topologically. In our case we simply used the Euclidean distance to measure the distance topologically. This is the most commonly used method. The updating algorithm is given by:

$$\Delta\mathbf{m}_i = \alpha(t)f(\Delta_{ir})(\mathbf{m}_i - \mathbf{u}), \tag{2}$$

where α is the learning rate decreasing to 0 with time t, $f(.)$ is a neighborhood function which controls the amount by which the codebooks are updated. Most commonly used neighborhood function is the Gaussian function:

$$f(\Delta_{ir}) = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_r\|^2}{2\sigma(t)^2}\right), \tag{3}$$

where the spread σ is called neighborhood radius which decreases with time t, $\mathbf{l}_r$ and $\mathbf{l}_i$ are the coordinates of the winning neuron and the i-th neuron in the lattice respectively. It is worth noting that $\sigma(t)$ must always be larger than 1 as otherwise the SOM reduces to Vector Quantization and no longer has topology preserving properties [2].

The steps 1 and 2 together constitute a single training step and they are repeated a given number of times. The number of iterations must be fixed prior to the commencement of the training process so that the rate of convergence in the neighborhood function, and the learning rate, can be calculated accordingly.

After training a SOM on a set of training data it becomes then possible to produce a mapping for input data from the same problem domain but which may not necessarily be contained in the training dataset. The level of ability of a trained SOM to properly map unseen data (data which are not part of the training set) is commonly referred to as the *generalization* performance. The generalization performance is one of the most important performance measures. However, in this paper, rather than computing the generalization performance of the SOM, we will evaluate the performance on the basis of micro purity and macro purity. This is done in order to comply with guidelines set out by the INEX-XML mining competition.

To allow for the processing of structured data this training algorithm is extended in [3] by incorporating the information about a node's neighbors. If $\mathbf{u}_i$ is used to denote the label of the i-th node in a graph, then an input vector for the SOM is formed by concatenating the label with the coordinates of the winning neuron of all the present node's neighbors. These coordinates are referred to as the *states* of neighbors. A hybrid input vector, defined as a vector $\mathbf{x}_i = (\mathbf{u}_i, \mathbf{y}_{ch[i]})$ is formed, where $\mathbf{y}_{ch[i]}$ is the concatenated list of states (coordinates of the winning neuron) of all the children of a node i. These states summarise the information which is contained in the child nodes. Note that here we assume a Markov assumption, in that the information on previous child nodes, the child nodes of those child nodes, etc are contained in the states. Since the size of vector $\mathbf{y}_{ch[i]}$ depends on the number of offsprings, and since the SOM training algorithm requires constant sized input vectors, padding with a default value is used for the missing offsprings or for nodes which have less than the maximum outdegree on a graph. Thus, the dimension of $\mathbf{x}_i$ is $p + qw$, where p is the dimension of the data label $\mathbf{u}$, q the dimension of the lattice, and w the maximum outdegree value. The training algorithm of a SOM is altered to account for the fact that an input vector now contains hybrid information (the data label, and the state information of offsprings). Equation 1 and Equation 2 are respectively replaced by the following:

$$r = \arg\min_i \|(\mathbf{x}_j - \mathbf{m}_i)^T \mathbf{\Lambda}\| \tag{4}$$

$$\Delta\mathbf{m}_i = \alpha(t) f(\Delta_{ir})(\mathbf{m}_i - \mathbf{u}) \tag{5}$$

where $\mathbf{\Lambda}$ is a $n \times n$ dimensional diagonal matrix; its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to μ_1, all remaining diagonal elements are set to μ_2. The constant μ_1 influences the contribution of the data label component to the Euclidean distance, while μ_2 controls the influence of the states on the same distance measure. Thus, if μ_1 is large relative to μ_2 then the contribution of data labels is more important in the Euclidean distance measure relative to that exerted by the states (past information contained in the child nodes), and vice versa. In reality, it is the ratio $\frac{\mu_1}{\mu_2}$ that is important rather than their relative values. For simplicity, $\mu_2 = 1 - \mu_1, \quad 0 \le \mu_1 \le 1$ is normally used. Then, the SOM-SD adds a new step to the training algorithm [3]:

Step 3: The coordinates of the winning neuron are passed onto the parent node which in turn updates its vector $\mathbf{y}$ accordingly.

The SOM-SD [3] which represents a first attempt in incorporating graph information in the SOM approach requires the processing of data in a strict causal order from the leaf nodes towards the root. Thus strictly speaking it is only applicable to processing tree structures, rather than the more general graph structures which may contain loops (where the Markov assumption that the information in the sub-tree processed so far is contained in the states of the child nodes breaks down). We process nodes in a strict leaf nodes to root node order so that in Step 3 all states of all neighbors are available. The SOM-SD approach has a computational complexity which for large graphs could be quadratic [7]. Hence the SOM-SD approach can only be applied to process tree structured data with small number of nodes.

However, there are many situations in which loops occur in graphs. Hence an extension to circumvent this problem of having to process the information in a strict leaf nodes to root order is necessary. A first attempt was proposed in the CSOM-SD [8]. The CSOM-SD builds on the SOM-SD by adding new steps to the training procedure to address the fact that both ancestors and descendants of the node may need to be taken into account in some applications. There are various ways to do this, but basically these make use of the information stored from previous processing steps to be used as a proxy of the lack of information imposed by the non-causal manner to process the graph as dictated by the graph structure (which may contain loops). Fundamentally the approach is similar to SOM-SD as it introduces yet another constant to take the additional requirement of representing the ancestors (the nodes which are ahead of the current node in the graph structure, and the information of which are represented by the stored states obtained from previous processing steps) into account, and the CSOM-SD can be reduced to the SOM-SD accordingly. Thus, the CSOM-SD suffers the same computational complexity issue which arises in the SOM-SD approach, Nevertheless, for small graphs, it can be applied to situations when there are loops, or un-directed links in the graph structure.

2.2 The GraphSOM

The GraphSOM [7], a very recent development addresses some of the shortcomings of SOM-SD. This is made possible by making a key observation in the SOM-SD and CSOM-SD processing of graphs: much of the information presented to the network may be redundant because it concatenates state information of every neighbor to the network input. Redundancies occur when several neighbors are mapped onto the same location on the map. This likelihood increases with increasing n, and becomes unavoidable when $n > k$. The GraphSOM processes the graph structured data by concatenating the data label with the activation of the map when mapping all of a node's neighbors. Note that here it is the activation of the map rather than the states of the map that is being presented to the GraphSOM inputs. Since the dimension of the map remains static, independent of the size of a training set, and independent of the outdegree of graphs, this implies that the GraphSOM's computational complexity is reduced to a linear one with respect to the outdegree of graphs. In other words, a GraphSOM can process graphs with a large outdegree much more efficiently than a SOM-SD. It is found that the GraphSOM includes the SOM-SD as a special case, and hence, one can expect that the clustering performances of the GraphSOM can be at least as good as that for the SOM-SD.

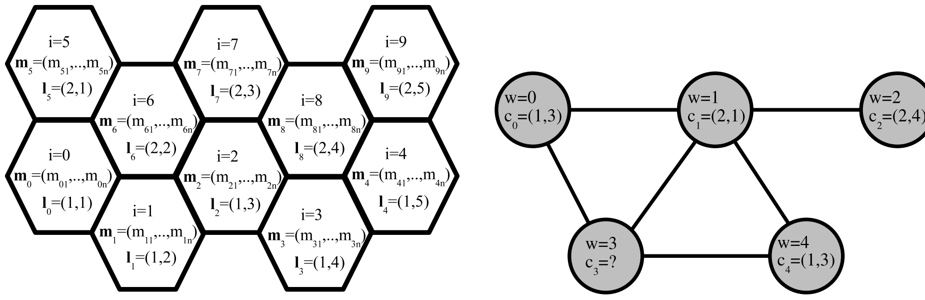

Fig. 2. A 2-dimensional map of size 2×5 (left), and an undirected graph (right). Each hexagon is a neuron. ID, codebook, and coordinate value for each neuron is shown. For each node, the node number, and coordinate of best matching codebook is shown.

Thus, the GraphSOM approach is to simply concatenate the activation (state) of the map with respect to all of the node's neighbors [7]. Formally, when processing a node i the input to the SOM is $\mathbf{x}_i = (\mathbf{u}_i, \mathbf{M}_{ne[i]})$, where $\mathbf{u}_i$ is defined as before, and $\mathbf{M}_{ne[i]}$ is a k dimensional vector containing the activation of the map $\mathbf{M}$ when presented with the neighbors of node i. An element M_j of the map is zero if none of the neighbors are mapped at the j-th neuron location, otherwise it is the number of neighbors that were mapped at that particular location.

It is much easier to express the underlying concepts in the GraphSOM[3] by considering the following example as shown in Figure 2. This figure shows a SOM. For simplicity, we assume further that no data label is associated with any node in the graph. Then when processing node $w = 3$ the network input is the k dimensional vector $\mathbf{x}_3 = (0, 0, 2, 0, 0, 1, 0, 0, 0, 0)$. This is because two of the neighbors of node 3 are mapped at the coordinate $(1, 3)$ which refers to the $2-$nd neuron, and the third neighbour of node 3 is mapped at $(2, 1)$ which refers to the $5-$th neuron. No other neuron is activated by a neighbour of node 3. These activations are listed in sequential order so as to form the vector $\mathbf{x}_3$. It can thus be observed that $\mathbf{x}_3$ summarizes the mappings of the neighbors of node 3 by listing the two activations: one at coordinate $(1, 3)$, the third neuron, and one at coordinate $(2, 1)$, the sixth neuron. Then the mapping of node 3 can be computed, and the information be passed onto its neighbors in the same way as is done for the SOM-SD. Note that the input remains unchanged regardless of the order of the neighbors; and that the dimension of $\mathbf{x}$ remains unchanged regardless of the maximum outdegree of any node in the dataset.

In comparison, the dimension of input vectors for the same graph when processed using SOM-SD is 6 (because $q = 2$ and $n = 3$). Thus, the number of computations with the GraphSOM is similar to that of SOM-SD when processing small graphs. But, the GraphSOM becomes more and more efficient as the connectivity (and consequently the size) of a graph increases. When dealing with large maps then the GraphSOM approach [7] can be approximated quite well

[3] This is the same example used in [7] to illustrate the developing of the thought process behind the proposed GraphSOM approach.

by consolidating neighboring neurons into groups. This approximation is valid since it exploits a general property with SOMs where only those nodes, which are mapped in close proximity to one another, share the greatest similarities, and hence, little information is lost by grouping them together. This leaves the granularity of the mappings unchanged but allows for a more compact representation of $\mathbf{M}_{ne[i]}$. For example, when using a 2×5 grouping for the map in Figure 1 then the network input for node 3 becomes $\mathbf{x}_3 = (1, 2, 0, 0, 0)$. A more coarse grouping can be chosen to achieve greater speed gains at the cost of some inaccuracy in the representation of the state of the map.

A summary of differences between the GraphSOM when compared to SOM-SD (and CSOM-SD) can be given as follows: (1) GraphSOM has a lower computational complexity when dealing with large graphs than SOM-SD; (2) GraphSOM's computational complexity can be scaled arbitrarily through a grouping of neurons at the cost of some inaccuracy in the representation of the graph while for SOM-SD it grows quadratically; (3) the input dimension of GraphSOM is independent of the level of connectivity in a graph while in SOM-SD it is a variable; and (4) the GraphSOM allows the processing of cyclic, positional, and non-positional graphs while SOM-SD can only be applied to trees, and CSOM-SD can be applied to cyclic graphs, graphs with un-directed links with low dimensions.

In fact, the GraphSOM is capable of processing the same classes of graphs as the GNN [1] with the exception that the labeling of links is currently not supported. It is obvious to note that the GraphSOM includes the SOM-SD as a special case.

We note in passing that a related idea has been proposed by some researchers in the context of sequence processing. In particular, the Merge SOM model [12] stores compressed information of the winner in the previous time step, whereby the winner neuron is represented via its content rather than its location within the map. The content of a neuron i consists of the weight w_i and the context c_i. These two characteristics of neuron i are stored in a merged form, i.e. as a linear combination of the two vectors. It should be stressed that this approach is different from the GraphSOM approach [7].

3 Experiments

This paper applies the SOM-SD and the GraphSOM to a large dataset consisting of structured documents from the web as contained in the INEX 2007 dataset (for the XML mining competition task). More specifically, the methods are applied to cluster a subset of documents from Wikipedia. The task is to utilize any of the features (content or structure) of the documents in order to produce a given grouping (clustering). We decided to investigate the importance of structural information to the clustering of these documents. Such an approach had been very successful at previous INEX clustering events. The documents are formatted in XML, and, hence, are naturally represented as tree structures. The dataset is made available in two parts, a training set and a test set. Each data in these sets is labeled by one of 21 unique numeric labels. The label indicates the desired grouping of the document. Hence, the task is to cluster pages into 21 groups. The machine learning method used for the experiments is trained *unsupervised*. This means that the data label is not used when training the SOM. The labels are used to evaluate whether the trained SOM groups the training data as desired. The SOMs' ability to perform the desired grouping of the training data is maximised by tweaking the various training parameters through trial and error. After determining the best set of training parameters, the SOM is used to map the unseen data in the test set. The performances reported in this paper are computed based on the mappings of the test set.

Since the data is represented in a tree form, and hence, the application of the SOM-SD is possible. However, it will be found that the outdegree of the dataset is prohibitively large. The GraphSOM has a reduced computational complexity relative to the SOM-SD. In applying the GraphSOM, the computational time required, compared with the ones using SOM-SD, is reduced significantly. This is especially important, as the INEX 2007 contains a relatively larger dataset.

The importance of this application is manifold:

- XML is an increasingly popular language for representing many types of electronic documents.
- An application to data mining tasks can help to demonstrate the capabilities of the GraphSOM, or the lack of it, over previous machine learning methods, e.g. SOM-SD approach which are capable of clustering graphs.
- The datasets considered (viz. the INEX Wikipedia dataset) is a benchmark problem used INEX 2007.

This paper gives some preliminary results[4]. Results presented here were obtained from training the SOMs for eight runs each. The best result is presented in this paper. Note that under normal circumstances, a SOM would have to be run under possibly hundreds of training conditions in order to determine its peak performance. This is due to the fact that a number of training parameters need to be determined through trial and error (for any SOM training algorithm). Amongst these parameters are the dimensionality of the map, the geometry of the map, the type of neighborhood relationship between the codebook entries of the map, a learning rate, the number of training iterations, weighting measures for the data label and structural component of the inputs, and several others. A suitable choice of training parameters is essential in obtaining a well performing GraphSOM (similar to all SOM approaches of which GraphSOM is one).

A first set of experiments utilizes the XML structure of the documents. A node in the graph represents an XML tag, the links represented the encapsulation of the tags. For example, the XML sequence `<a><b></b><c></c></a>` produced a graph with a root node representing the tag `<a>` and its two offsprings `<b>` and `<c>`. We assigned a unique numerical ID number to each unique XML tag, then added the ID number as a label to the corresponding node in the graph. For example, if we assign the ID number 101 to tag `<a>`, then the node representing this tag will have been assigned the numeric label 101. In other words,

[4] The GraphSOM has been developed very recently[7]. Time constraints and implementation issues prevented us from conducting experiments more thoroughly.

Table 1. Results when clustering the test dataset into 21 clusters

SOM-SD	
Micro average purity	0.262457
Macro average purity	0.26159

GraphSOM	
Micro average purity	0.26885
Macro average purity	0.26635

the SOMs are trained to cluster XML formatted documents solely on topological information, and on the type of XML tag embedded in the document. No further information is provided to the training algorithm.

The results when clustering the dataset into 21 clusters are summarized in Table 1. In comparison, the performances of the clustering task obtained by other participants of the competition is shown in Table 2. It is observed that (1) despite successes of the SOM-SD method in earlier INEX competitions [4,6] the performances obtained here are well below those obtained by others, and (2) both the GraphSOM and the SOM-SD perform about equally well. A difference between the latter is the training times needed. Given that the training dataset contained $48,306$ tree structures, one for each document, with a maximum out-degree of 1,945, this size of outdegree would require an estimated 40 years of training time for the SOM-SD on a top end desktop computer with 2.8 GHz clock rate! To avoid this, and to enable the use of the SOM-SD for comparison purposes, we pruned the graphs to have a maximum outdegree of 32 by truncating nodes with a larger outdegree. This reduced the training time for the SOM-SD to a more reasonable 36 hours[5]. In comparison, the GraphSOM is capable of processing the graphs without pruning in about 48 hours by using a 8×8 grouping of nodes. The results shown were obtained when using for both networks a size of 160×128, the number of training iterations equal to 200, $\alpha(0) = 0.8$, the weights μ_1, μ_2 were $(0.05, 0.95)$, and $\sigma(0) = 20$.

Table 2. Results when clustering into 21 clusters obtained by competing groups

Name	Micro avg. purity	Macro avg. purity
Guangming Xing	0.62724	0.571855
Jin YAO & Nadia ZERIDA	0.51530897	0.61035

Pruning can have a negative impact on the clustering performance since some relevant information may be removed. The GraphSOM allows the processing of large graphs without requiring pruning, and hence, can be expected to produce performances which are at least as good as those obtained by a SOM-SD if not better. While this has been confirmed by these experiments, it is also found that

[5] This is one of the major disadvantages of using SOM-SD in processing larger datasets. In order to ensure a reasonable turn-around time it is necessary to prune the data back so that the training time is kept to reasonable duration. In doing the pruning, some information would be lost. In contrast, the GraphSOM does not require as extensive pruning or no pruning at all, and hence it is far more efficient than the SOM-SD.

the pruning did not reduce the SOM-SD performance. These observations (generally relative poor performance, and pruning without effect on results) caused us to suspect that the XML structure of the documents may not be a feature which leads to a desired clustering result as was defined by INEX-2007.

Thus, one of the reasons for the relatively poor performance may be that the desired clustering of the documents may not rely on the XML structure as much as the learning tasks in previous years did. A difference with the datasets used in previous years is that this year's dataset contained embedded hyperlinks which pointed from one document to another document. It may be that a successful clustering requires the encoding of the link structure of the documents rather than the XML structure. To verify this hypothesis, we created a new graph representation of the dataset where a node represents a document as a whole, and the links represent the hyperlinks pointing from one document to another. This produced one directed graph for each; the training set and the test set containing $48,306$ and $48,305$ nodes respectively. The graphs contained cycles. The maximum outdegree for each of the two datasets were 410 and 544 respectively. Note that some documents have several (redundant) links to another document. We consolidated these by listing only its first occurrence. This was performed since we assumed that such redundancy may not significantly influence the results, and to help reduce the turn around time for the experiments. Had we included redundant links, the maximum outdegree would have increased to 471 and 579 respectively. Note also, that only those links were used which pointed to documents within the same dataset (i.e. for the training set we used only those links which pointed to other documents in the training set). Links to documents outside the same dataset were discarded. Nevertheless, we added the total number of (unique) hyperlinks as a data label to each node in the graph. It is interesting to note that the maximum outdegree of any node in the datasets is $2,118$ for the training set and $2,088$ for the test set if all links (internal, external, and redundant links) are considered. This implies (a) that most hyperlinks are to pages not within the same dataset, and (b) that Wikipedia document hyperlink structure is relatively rich.

For the purpose of training the SOM-SD, we again had to resort to pruning. Here, pruning was done selectively by identifying links which are involved in cycles and pruning these first. This allowed the generation of an acyclic graph with sufficiently small outdegree which allowed the training of a SOM-SD in a timely fashion. No such pruning was necessary for the training of the GraphSOM. Maps were trained with the same parameters as before on these datasets. The results are summarized in Table 3.

It can be observed that this produced an improvement in the performance by 3 to 9 percentage points. It is also observed that the GraphSOM performs slightly better than the SOM-SD which may be attributed to that the GraphSOM is trained on a non-positional graph which was not pruned. The finding indicate that information which is useful for obtaining the desired partitioning of the test data set is embedded in the hyperlink structure of the documents, and that the task does not appear to rely on positional information about the links (i.e. the order of the hyperlinks does not appear to be important). Nevertheless, the performances obtained still fall short of those obtained by other participants of the competition. It shows that structural information may not be a key ingredient to obtain the desired clustering. It is thus likely that textual information within the documents is of importance for this given task. This can be addressed through a suitable labeling of the nodes in the graph so as to encode textual information. To surmise one of the reasons why the addition of content to link structure may not result in improved accuracy is that there are many ways in which information on content can be extracted. For example, in an XML document, content could mean the extraction of information on each paragraph or on each document. Since such information is contained in words, there will need to be a good way to include the information and meaning conveyed by words, or combination of words. This is not a particularly easy task to perform as it is well known that some combination of words may lead to quite different meaning. However, without an accurate representation of the contents, the SOM may not receive sufficient information for it to discriminate web pages. This points to possible ways in which our approach can be improved: through a better representation of the content information of the documents. This is left for a future task and will not be addressed further in this paper.

4 Conclusions

The paper applied the newly developed GraphSOM [7] for the first time to a relatively large clustering application represented by the INEX 2007 competition dataset. Comparisons with its predecessor (the SOM-SD) revealed that the added capabilities of the GraphSOM can help to produce improved results. But most importantly, it was demonstrated that the lower computational complexity of the GraphSOM and its ready scalability to larger datasets allow the processing of graphs without requiring pre-processing measures such as pruning in order to reduce time requirements for the training procedures. This represents a major advance on the SOM-SD approach in terms of its practicality, in that GraphSOM is shown to be able to handle relatively large datasets without any pruning, thus preserving the information content in the dataset. In contrast, the SOM-SD approach is required to prune the training dataset so as to keep the training time to reasonable duration, and thus may lead to a loss of information.

This paper presented some initial experimental results which do not measure up to results obtained by others on this clustering task. It can be assumed that by fine-tuning training parameters, and by providing a richer set of information to the GraphSOM learning procedure will help to improve the performances. For example, the data label attached to nodes may contain a description of the content found in the document. Such tuning may not improve the performance significantly if the decisive criteria for the clustering task is based on structural components. One could, moreover, attempt to train the GraphSOM on a hybrid structure which combines both the underlying XML structure of individual documents as well as the hyperlink structure between documents. This may improve the performance of the method as it takes into account the nature of a dataset which contains structured documents, and a structural dependency between documents. There are various

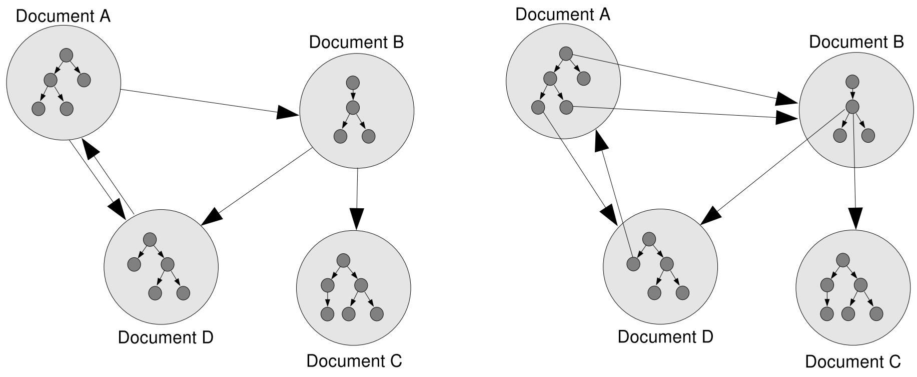

Fig. 3. Hybrid structures. When considering hyperlinks to point from one document to another (left), or when considering hyperlinks to point from within an embedded XML-tag to another document (right).

ways in which such combination can be effected. One way in which this can be effected is that if we consider hyperlinks to be a binary relation between two documents as depicted in Figure 3 (left). Each document is represented by its XML (tree-)structure. A SOM could be trained on these tree-structures as was carried out in this paper. Then, once the SOM has converged to a final mapping, a second SOM is trained on the underling hyperlink structure of the dataset where, to each node, a data label which summarizes the activation of the first SOM (trained on XML information) is attached. Since the XML representation of documents is done through a tree structure, and since processing can be done in an inverse topological order, and hence, the root of the XML tree structure can be a representation of the tree structure as a whole. Thus, it would suffice to solely utilize the mapping (state) of the root as a label for the corresponding node in the hyperlink structure.

Alternatively, one may wish to take account of the fact that hyperlinks are embedded within a section defined by an XML tag (Figure 3 on the right), and hence, the source of a hyperlink can be any node in the XML tree. Thus, it may be more appropriate to use the mapping of the XML-tag which is the source of a hyperlink for the label for the associated node.

One may observe that for both of these approaches, the dimensionality of the node labels depends on the outdegree of all nodes in the XML tree. For instance, in Figure 3 (right), Document A has two nodes with outgoing hyperlinks to Document B. To account for the various activations of the nodes involved, concatenation could be used to obtain the associated data labels. However, to avoid problems which would arise with large outdegrees, one could use the Graph-SOM approach for forming the node labels (i.e. to use the activation of the map rather than concatenating individual activations). The second map, once trained on the underlying hyperlink structure, would produce a mapping of documents according to the overall hybrid structure.

These instances show that one may not ignore the underlying structure of a document, whether the inherent structure, or the extended structure as provided

Table 3. Results when clustering into 21 clusters using the hyperlink structure

SOM-SD	
Micro average purity	0.253804
Macro average purity	0.337051

GraphSOM	
Micro average purity	0.27423
Macro average purity	0.353241

through the hyperlink structure if one wishes to cluster the documents. The validation of these hypotheses and intuition is left as a future research task.

Acknowledgment. The first and second authors wish to acknowledge financial support by the Australian Research Council through its Discovery Project grant DP0452862 (2004 - 2006) and DP077148 (2007 - 2009) to support the research reported in this paper.

References

1. Scarselli, F., Yong, S., Gori, M., Hagenbuchner, M., Tsoi, A., Maggini, M.: Graph neural networks for ranking web pages. In: Web Intelligence Conference (2005)
2. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences, vol. 30. Springer, Berlin (1995)
3. Hagenbuchner, M., Sperduti, A., Tsoi, A.: A self-organizing map for adaptive processing of structured data. IEEE Transactions on Neural Networks 14(3), 491–505 (2003)
4. Hagenbuchner, M., Sperduti, A., Tsoi, A., Trentini, F., Scarselli, F., Gori, M.: Clustering xml documents using self-organizing maps for structures. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 481–496. Springer, Heidelberg (2006)
5. Hagenbuchner, M., Sperduti, A., Tsoi, A.: Contextual self-organizing maps for structured domains. In: Workshop on Relational Machine Learning (2005)
6. KC, M., Hagenbuchner, M., Tsoi, A., Scarselli, F., Gori, M., Sperduti, S.: Xml document mining using contextual self-organizing maps for structures. LNCS. Springer, Berlin (2007)
7. Hagenbuchner, M., Sperduti, A., Tsoi, A.: Self-organizing maps for cyclic and unbound graphs. In: European symposium on Artificial Neural Networks, April 23-25 (2008) (to appear)
8. Hagenbuchner, M., Sperduti, A., Tsoi, A.: Contextual processing of graphs using self-organizing maps. In: European symposium on Artificial Neural Networks. Poster track, Bruges, Belgium, April 27-29 (2005)
9. Sperduti, A., Starita, A.: A memory model based on LRAAM for associative access of structures. In: Proceedings of IEEE International Conference on Neural Networks, June 2-6, 1996, vol. 1, pp. 543–548 (1996)
10. Neuhaus, M., Bunke, H.: Self-organizing maps for learning the edit costs in graph matching. IEEE Transactions on Systems, Man, and Cybernetics, Part B 35(3), 503–514 (2005)
11. Rahman, M.K.M., Yang, W.P., Chow, T.W.S., Wu, S.: A flexible multi-layer self-organizing map for generic processing of tree-structured data. Pattern Recogn. 40(5), 1406–1424 (2007)
12. Strickert, M., Hammer, B.: Neural gas for sequences. In: WSOM 2003, pp. 53–57 (2003)