2012

Data driven encoding of structures and link predictions in large Xml document collections

M Hagenbuchner
University of Wollongong, markus@uow.edu.au

A C. Tsoi
Macau University, act@uow.edu.au

M Kc
University of wollongong, millykc@uow.edu.au

Shujia Zhang
University of Wollongong, shujia@uow.edu.au

Publication Details

Hagenbuchner, M., Tsoi, A. C., Kc, M. & Zhang, S. (2012). Data driven encoding of structures and link predictions in large Xml document collections. In A. Tagarelli (Eds.), XML data mining : models, methods, and applications (pp. 219-241). Hershey, PA: Information Science Reference.

Data driven encoding of structures and link predictions in large Xml document collections

Abstract
In recent years there have been some significant research towards the ability of processing related data, particularly the relatedness among atomic elements in a structure with those in another structure. A number of approaches have been developed with various degrees of success. This chapter provides an overview of machine learning approaches for the encoding of related atomic elements in one structure with those in other structures. The chapter briefly reviews a number of unsupervised approaches for such data structures which can be used for solving generic classification, regression, and clustering problems. We will apply this approach to a particularly interesting and challenging problem: The prediction of both the number and their locations of the in-links and out-links of a set of XML documents. In this problem, we are given a set of XML pages, which may represent web pages on the Internet, with in-links and out-links. Based on this training dataset, we wish to predict the number and locations of in-links and out-links of a set of XML documents, which are as yet not linked to other existing XML documents. To the best of our knowledge, this is the only known data driven unsupervised machine learning approach for the prediction of in-links and out-links of XML documents.

Keywords
data, driven, encoding, structures, link, collections, document, xml, large, predictions

Disciplines
Engineering | Science and Technology Studies

Publication Details
Hagenbuchner, M., Tsoi, A. C., Kc, M. & Zhang, S. (2012). Data driven encoding of structures and link predictions in large Xml document collections. In A. Tagarelli (Eds.), XML data mining : models, methods, and applications (pp. 219-241). Hershey, PA: Information Science Reference.

Data Driven Encoding of Structures and Link Predictions in Large XML Document Collections

M. Hagenbuchner [a], A.C. Tsoi [b], M. Kc [a], S. Zhang [a].

[a] *University of Wollongong, Wollongong, Australia.*

[b] *Macau University of Science and Technology, Macau.*

ABSTRACT

In recent years there have been some significant research towards the ability of processing related data, particularly the relatedness among atomic elements in a structure with those in another structure. A number of approaches have been developed with various degrees of success. This chapter provides an overview of machine learning approaches for the encoding of related atomic elements in one structure with those in other structures. The chapter briefly reviews a number of unsupervised approaches for such data structures which can be used for solving generic classification, regression, and clustering problems. We will apply this approach to a particularly interesting and challenging problem: The prediction of both the number and their locations of the in-links and out-links of a set of XML documents. In this problem, we are given a set of XML pages, which may represent web pages on the Internet, with in-links and out-links. Based on this training dataset, we wish to predict the number and locations of in-links and out-links of a set of XML documents, which are as yet not linked to other existing XML documents. To the best of our knowledge, this is the only known data driven unsupervised machine learning approach for the prediction of in-links and out-links of XML documents.

INTRODUCTION

The traditional approach to processing data is by using numeric vectors of some fixed dimension as a means of representing the data, where the vector elements describe features of an associated data item (Haykin, 1994). Such representation is suitable when the data items are independent or when any existing dependencies are not relevant for the solving of a given problem. A common situation in which data items are dependent is in time series information processing (Haykin, 1994). In this case, a data entry may depend on another data entry which occurred in a prior time instance, and such dependency may be relevant for a given problem (Haykin, 1994). Data items which are sequentially organized are referred to as *sequences*. For example, in financial forecasting or in speech recognition, it is important that such dependency can be represented and encoded (Haykin, 1994). Note that unlike data vectors which are often of the same dimension, which in the cases where they are not of the same dimension, then the technique of "zero padding" is used to ensure that all data vectors are of the same dimension (Haykin, 1994), there is not normally a restriction to the length of sequences. When data items are "related" to several other data items, then such dependencies are popularly described by a *graph*, and hence such instances are referred to as *graph data structures*. Note that here, the data items are "related" in that the data objects, represented by nodes of a graph, are connected by links, directed or un-directed, cyclic, or acyclic, which express the relationships among the data objects. A very large number of practical problems can be represented as a graph data structure. For example, problems in molecular chemistry, image processing, computer security systems, weather forecasting, etc. can be addressed by processing the dependencies which are represented in graph structured data. Since sequences are a special case of graphs (graphs with an out-degree of 1), and since vectors are also a special case of graphs (a graph with a single node), and, hence, graph data structures provide the most generic means for data representation. Any

approach capable of encoding graphs can also solve time series problems and problems in which the data can be represented in vector spaces.

Simple graphs consist of two basic elements, namely, *nodes* and the binary relations called *links*. The nodes represent atomic entities of a domain. For example, a document may be represented by a node. A node may be labeled to describe the associated entity. For example, a node label may contain descriptive features of the associated document. These features could be the number of words in the document, the occurrence of certain words in the document, the frequency with which certain words occurred in the document, etc. A link exists between any two nodes if these nodes are "related" in some way. For example, a document may contain a hyperlink to another document. In this case, there is a directed link from the node representing the source document to the node which represents the target document of the hyperlink. A link may also be labeled in order to describe its properties. For example, a label associated with a link may indicate the strength of a link. Links may be directed (such as in the hyperlink case) or undirected (such as the links indicating atomic bindings of a molecule). The total number of all outgoing links of a node is called the *outdegree* of the node, while the total number of all incoming links is called the *indegree* of a node. In the case of undirected links, this is referred to as the *degree* of a node. The size of a graph G is the total number of nodes in G and is denoted by $n = |G|$. If there exists a path along the links in G from one node to another then these nodes are said to be *connected*, otherwise they are *disconnected*. The shortest path along the link structure between a pair of nodes in G is called the *distance* between these nodes [1].

The importance of an ability to process graph structured information is highlighted by the fact that most documents are created electronically nowadays. The content of a document is stored as normal text whereas the layout, appearance, and structure of the text is described by a machine readable annotating text. In this context, the Extensible Markup Language (XML) has established itself as one of the most important mechanisms for describing layout, appearance, and structure of an electronic document. The increasing popularity of XML is due to its simplicity and generality. Since XML documents are becoming more and more widespread and thus the development of tools capable of dealing with XML documents is more and more important. An important observation is that XML imposes a logical tree-like structure on a document meaning that XML provides a means to represent documents naturally as a data tree. Since XML describes the structure and appearance of a document, and hence, the associated tree representation suitably represents the structure and appearance of a document. Moreover, XML allows the linking of documents via the so called *hyperlinks*. This means that it is possible to represent a collection of documents by means of a set of interlinked trees. This chapter addresses possible machine learning approaches to dealing with such types of data, and proposes a means to predict inter-document links.

A number of data mining techniques, e.g., syntactical grammar (Fu, 1977), have been developed to deal with these types of input data structures. This chapter will focus on machine learning techniques. The traditional approach to processing data vectors in machine learning is through Multi-Layer Perceptron Neural networks (MLP) (Haykin, 1994), and Self-Organizing Maps (SOMs) (Haykin, 1994), whereas the traditional approach to processing sequences is by using Elman Networks (Elman, 1990; Kohonen, 1984; Rumelhart et al., 1986). Machine learning approaches for the processing of graphs have only recently been introduced (Hagenbuchner et al., 2003; Kc et al., 2010; Scarselli et al., 2009a), and have already been very successful in solving several benchmark problems (Hagenbuchner et al., 2006; Kc et al., 2007; Zhang et al., 2009).

This chapter describes a general framework in which graph structured information can be encoded for machine learning applications. The organization of this chapter is such that it will first give a detailed view of one particular class of machine learning methods which, as we will find, is particularly suitable for solving the challenging task of predicting links in an interlinked domain. Then, an explicit description of such an application domain is provided. A practical deployment and associated experimental findings to a benchmark problem, viz., the prediction of the number and locations of in-links and out-links of a set of XML documents are then to demonstrate the capabilities of the deployed machine learning methods. Finally, a summary of this chapter and some future directions are provided.

Encoding Structures in Machine Learning

The general approach to encoding structured information in machine learning is by encoding individual nodes of a given graph one at a time. When processing a particular node, the node's direct neighbors are taken into account by using an internal representation of these neighbors as an additional input to the current node. The repeated recursive

[1] In this chapter, we are not dealing with hypergraphs, in which a link may connect to more than one node in a graph.

processing of the nodes in a graph ensures that information is eventually passed from any node to any other node in the graph. The main difference between the various machine learning methods is in the way by which such internal representation is produced. In the following, we will refer to the internal representation of a node as the *state* of the node. The nomenclature is derived from the fact that the internal representation of a node is the network's response to a given set of inputs to the node together with its neighbours. In other words, it reflects the network's *state* in response to a given set of input nodes linked to the node.

The key idea in our work is to represent each node using a neural network. While the same neural network is used to encode all the nodes in a dataset, it is the network's *activation* that is used to represent each node. Then, this node's response (the state of the node) is obtained through the inputs which are derived by nodes which are linked to the current node. Each of the nodes connected via a link to the current node is in turn represented by a response (state) which is the result of nodes which are incident on it. Thus, by processing each node of a graph in turn, it is possible to recursively process the nodes in the graph, and eventually, all nodes will be considered.

There are two main types of neural networks, namely, supervised neural networks and unsupervised neural networks. Supervised neural networks are deployed when a teacher (target) signal is available for a given set of data (the training dataset consists of a set of graphs together with an associated teacher signal at an output node, or a set of teacher signals in a set of output nodes). The teacher signal is utilized as the target information for supervised neural networks. Unsupervised neural networks are deployed if no supervising (teacher) signal exists. In this case the training dataset simply consists of a set of graphs, but there is no teacher signal associated with any of the nodes. The training procedures (the learning of the unknown parameters by the processing of the training dataset) of these two types of neural networks differ significantly.

Supervised Learning of Structured Data

The availability of ground truth information (teacher signals) for a given set of nodes allows for the learning of neural networks with specific targets. MLP based systems are parameterized systems which can be trained if the target information is available (Haykin, 1994). This is performed by comparing the desired target value with the actual output produced by an MLP to form an error signal. A gradient descent method is adopted to minimize the sum of error signals through the entire training dataset during the training process. Formal proofs exist which show that such methods can produce optimal results under some conditions (Hornik et al., 1989).

The supervised encoding of graph structured data is achieved by extending an MLP by a recursive element (Scarselli et al., 2009b). The error is computed as in the MLP case, and the gradients are propagated back through the recursive structure of the neural network (Scarselli et al., 2009b).

Since in this chapter, we will not be using supervised approaches, and hence we will not consider this approach further, except to indicate that there are developments in supervised approaches to graph structured input data (Bianucci et al., 1998; Scarselli et al., 2009b), which had been proved to be very successful in processing graph structured data (Scarselli et al., 2009a).

Unsupervised Learning of Structured data

Target information may not exist with many real world problems. This is particularly the case for data mining problems. This rules out the application of supervised machine learning approaches. Unsupervised machine learning algorithms exist when there is a set of vectors $\boldsymbol{u}$ for which there is no associated target. One of the most popular and most widely adopted algorithms is the Self-Organizing Map (SOM) (Kohonen, 1984). A SOM performs a mapping from a high dimensional input space to a low dimensional display space in such a way that input data which are close to one another in the input space are mapped near to each other on the display space (Kohonen, 1984). Hence, a SOM can be used for the clustering or dimension reduction of a set of input data. One of the most appealing properties of the SOM is that the algorithm is computationally very efficient, and hence, a SOM can be easily applied to large scale applications (Kohonen, 1997). Moreover, the training algorithm of a SOM does not rely on the computation of a gradient, and hence, the SOM does not suffer from any long term dependency problems that afflict gradient descent learning methods (Bengio et al., 1994).

The usefulness of SOM can be observed through its widespread adoption by practitioners, however, the representation of the input data in the form of disjointed sets limits its ability to effectively encode complex or structured data. As such, various extensions of the standard SOM were created, one of which is the extension which allows

the encoding of graph structured information (Hagenbuchner et al., 2003). This is an interesting aspect since this allows the mapping of graphs onto a fixed dimensional display space. One approach uses the edit distance to define similarity in the competitive step of the training algorithm (Bunke, 1990). However, the approach is computationally extremely expensive, and hence, it is restricted to solving very small toy problems only. The approaches (Hagenbuchner et al., 2003, 2001) described in this chapter are much more scalable and have already been applied to a variety of data-mining problems. But more importantly, the SOM based methods described in this chapter are not only capable of encoding relatedness that may exist between vectors and graphs but also they are capable of predicting the number and location of links.

SELF-ORGANIZING MAP FOR STRUCTURES

The basic SOM defines a mapping from an input data space onto a regular q-dimensional array of the so called *neurons* (Kohonen, 1995), alternatively known as a display space of q-dimensions. It is very common that $q = 2$. Every neuron i of the map is associated with an n-dimensional codebook vector $\mathbf{m}_i = (m_{i1}, \ldots, m_{in})^T$, where $\mathbf{m}_i$ is said to be a weight vector and its elements are said to be weights, and T denotes the transpose of a vector. The neurons of the map are connected to adjacent neurons by a neighborhood relation where rectangular and hexagonal are the most common ones (Kohonen, 1995).

Adjacent neurons belong to the neighborhood N_i of a neuron i. Neurons belonging to N_i are updated according to a neighborhood function $f(\cdot)$, where the *Gaussian-bell* shape function is most commonly used (Kohonen, 1995). The number of neurons determines the granularity of the mapping, which has an effect on the accuracy and generalization capability of the SOM (Kohonen, 1995).

The training algorithm of the weights associated with each neuron is performed in two steps:

Competitive step: One sample vector $\boldsymbol{u}$ is randomly drawn from the input data set and its similarity to the codebook vectors is computed. The minimum Euclidean distance is typically used for this purpose. A winning neuron is then computed through the following operation:

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

Cooperative step: The best matching codebook $\mathbf{m}_r$ 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 $\mathbf{m}_i$. 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 a learning coefficient, and $f(\cdot)$ is the neighborhood function which controls the amount which the weights of the neighbouring neurons are updated. The neighborhood function $f(\cdot)$ can take the form of a Gaussian function:

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

where σ is the spread, and $\mathbf{l}_r$ is the location of the winning neuron, and $\mathbf{l}_i$ is the location of the i-th neuron in the lattice. Other neighborhood functions are possible (Kohonen, 1984).

The two steps together constitute a single training step and they are repeated until the training ends. As the learning proceeds and new input vectors are input to the map, the learning rate $\alpha(t)$ gradually decreases to zero according to a specified learning rate function type. Along with the learning rate, the neighborhood radius decreases as well. This means that the number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated accordingly.

The Self Organizing Map for Structured Data

The Self-Organizing Map for Structured Data (SOM-SD) extends the SOM algorithm so as to allow the encoding of causal relations that may exist between a set of input vectors. The basic idea is analogous to the supervised

machine learning approach: Take the network's response to directly related vectors as an additional input. The main difference lies in the form by which a SOM produces a response. The response of a SOM to a given input is produced by Equation 1. The SOM-SD uses the 2-dimensional coordinates of the winning neuron as the state information for an associated input vector, and the input vectors are formed by concatenating the j-th node's label $\boldsymbol{u}_j$ with the state of the node's neighbors $\boldsymbol{c}_{ne[j]}$ forming an input vector $\boldsymbol{x} = [\boldsymbol{u}_j, \boldsymbol{c}_{ch[j]}]$, where $\boldsymbol{c}_{ch[j]}$ are the concatenated coordinates of the winning neurons of the children of node j. The newly formed input vectors $\boldsymbol{x}$ are of dimension $n = p + 2o$, where $p = |\boldsymbol{u}|$ and o is the maximum out-degree of a graph G. The codebook vectors $\boldsymbol{m}$ are of the same dimension. Training a SOM-SD is analogous to the training of the SOM where Equation 1 is modified as follows:

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

where $\boldsymbol{\Lambda}$ is a $n \times n$ dimensional diagonal matrix. Its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to μ, all remaining diagonal elements are set to $1 - \mu$. The constant μ influences the contribution of the data label component to the Euclidean distance, and $1 - \mu$ controls the influence of the children's coordinates to the Euclidean distance. Equation 2 in this case is modified as follows:

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

Since the order of the state vector elements in $\boldsymbol{x}$ matters, and since a SOM-SD can only take the state information of offsprings, and hence, a SOM-SD can encode causal relationships in ordered tree structured data only. Many problems can be solved using directed ordered trees. This is because objects in computer science are popularly described by various types of parsing trees (Fu, 1977). The SOM-SD has been very successful in a range of application domains and has set the state-of-the-art performances (Hagenbuchner et al., 2006; Kc et al., 2007) in this type of problems.

The Contextual SOM-SD

A SOM-SD can only encode causal relationships in a directed acyclic graph. This means that the mappings produced by the SOM-SD are independent of a node's relationship with any parent node. The ability of encoding contextual information in which not only the information about child nodes is available, but also the information on the parents of the current node is of relevance in a number of application domains. For example, in recommender systems (Takács et al., 2009), it is important to be able to encode the context within which a causal relationship occurred. The Contextual SOM-SD (CSOM-SD) (Hagenbuchner et al., 2005) extends the SOM-SD algorithm in order to encode contextual relationships in directed ordered tree structured learning problems. This is achieved by concatenating the state information of a node's offsprings as well as the state information of a node's parents such that an input vector becomes $\boldsymbol{x} = [\boldsymbol{u}, \boldsymbol{c}_{ch[i]}, \boldsymbol{c}_{pa[i]}]$, where $\boldsymbol{c}_{pa[i]}$ are the coordinates of the winning neurons of the parents of node i, and $\boldsymbol{c}_{ch[i]}$ is as before. The input vectors are made constant in size by padding with an illegal coordinate (such as $(-1, -1)$) for nodes with an outdegree or indegree smaller than the maximum outdegree or smaller indegree respectively. The training algorithm remains unchanged; it is the same as the one used for SOM-SD with the exception that $\boldsymbol{\Lambda}$ in Equation 4 is now a $m \times m$ dimensional diagonal matrix. Its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to μ_1, a constant, the elements $\lambda_{p+1p+1} \cdots \lambda_{nn}$ are set to μ_2, another constant, and all remaining diagonal elements are set to $1 - \mu_1 - \mu_2$.

The Probability Mapping Graph SOM

The most advanced unsupervised machine learning approach to encoding structures is the Probability Measure Graph SOM (PMGraphSOM). The PMGraphSOM is capable of encoding general graphs which may feature directed or undirected links (or a mixture of both), have an arbitrary degree, and may contain cyclic dependencies (Hagenbuchner et al., 2009; Kc et al., 2010).

The PMGraphSOM achieves this through a two-fold modification of the SOM-SD algorithm. First, the PMGraphSOM modifies the interpretation of the *state* vector. Instead of concatenating the coordinates (the *state*) associated with the winning neuron of each child, parent, or neighbor, the PMGraphSOM consolidates the mappings of all a node's neighbors by considering the mappings produced by all neighbors in the display map $\boldsymbol{M}$. The map $\boldsymbol{M}$

consists of elements (the neurons) which can be active if a node is mapped to a location. Thus, the PMGraphSOM forms input vectors by concatenating the node label with the activation of the map when mapping all the node's neighbors. Since the size of $\boldsymbol{M}$ is fixed, and hence, the input dimension is fixed and independent of the degree of any node in a set of data.

The similarity measure in the competitive step in Equation 4 does not work as desired when using the activations of $\boldsymbol{M}$ instead of the coordinate values as with SOM-SD or CSOM-SD. The problem is that the Euclidean distance measure used in Equation 4 does not make a distinction as whether any change of a mapping during the cooperative step has been mapped to a nearby location or to a location far away on the map. A *soft coding* of the mappings is applied to account for the likelihood of any changes in the mapping of nodes in subsequent iterations (Hagenbuchner et al., 2009). In other words, instead of *hard coding* the mappings of nodes to be either 1 if there is a mapping at a given location, or 0 if there is no mapping at a given location, the likelihood of a mapping in a subsequent iteration is encoded by a probability value. It is known that due to the effects of the training algorithm it is most likely that the mapping of a node will be unchanged in the next iteration. But since all vectors associated with the grid points in the display map are updated, and since those vectors which are close to a winning entry (as measured by the Euclidean distance) are updated more strongly (controlled by the Gaussian function), and, hence, it is more likely that any change of a mapping will be to a nearby location rather than to a location far away from the last update. These likelihoods are directly influenced by the neighborhood function and its spread. The PMGraphSOM incorporates the likelihood of a mapping as follows:

$$M_i = \frac{e^{-\frac{\|\{i_1,j_1\}-\{i_k,j_k\}\|^2}{2\sigma(t)^2}}}{\sqrt{2\pi}\sigma(t)}, \tag{6}$$

where M_i refers to the i-th element of the map, $\sigma(t)$ decreases with time t towards zero, and $\{i_k, j_k\}$ are the coordinates of the winning vector, while $\{i_1, j_1\}$ are the coordinates of the vector in the display space. The computation is cumulative for all the i-th node's neighbors. The term $\frac{1}{\sqrt{2\pi}\sigma(t)}$ normalizes the states such that $\sum_i M_i \approx 1.0$. It can be observed that this approach accounts for the fact that during the early stages of the training process it is likely that mappings can change significantly, whereas towards the end of the training process, as $\sigma(t) \rightarrow 0$, the state vectors become more and more similar to the hard coding method. It was shown that this approach helps to significantly improve the stability of the GraphSOM [2], and allows the setting of large learning rates while providing an overall improvement in the clustering performance (Hagenbuchner et al., 2009).

Figure 1 illustrates the general idea underlining the encoding of hyperlinked XML documents by a SOM. The figure illustrates that a set of hyperlinked XML documents can be readily represented as a graph. Each node of the graph represents a document. Attached to a node is a label which describes the content of properties of the associated XML document. Note that the dimension of a node label can be arbitrary. This means that it is possible to describe any XML document to any arbitrary detail. Sometimes it can be useful to set the dimension of the node label to zero in cases where only the hyperlink structure is of relevance for a given task. Each node in the graph is then processed by a SOM by concatenating the label of the node with the state of the node's neighbors. As an example, let us assume that node 2 and node 3 (the Figure has labeled the nodes by a unique value for ease of reference) have already been processed. Let us further assume that node 2 was mapped to location $\mathbf{l}_3$ and node 3 was mapped to location $\mathbf{l}_5$ of the map, then when processing node 1 in the case of SOM-SD the input vector would be $\mathbf{u}_1 = (12, 4.12, 7.2, 1, 4, 2, 1)$. Thus, it can be observed that the network input is simply the concatenation of the node label with the network's response (in the SOM-SD's case the coordinates of the codebooks activated by the node's offsprings). Step 2 of the training algorithm updates the codebooks so as to make them more similar to the input. This means that the codebook encodes both the data label and context of a node. Thus, if we know the activation of a node without explicit knowledge of the node's context, then we can extract the context from the state vector component of the codebook vector (in the previous example, the last 4 elements of the codebook vector hold information on up to two offsprings of a node). This property is important later in this chapter when describing an approach to link prediction. Given the same scenario described before, the input for a PMGraphSOM

[2] GraphSOM is an earlier version of SOM specifically designed to handle graph inputs through the *state* representation of the winning nodes in the map M. This was before we used the probability mapping idea as indicated in this chapter.

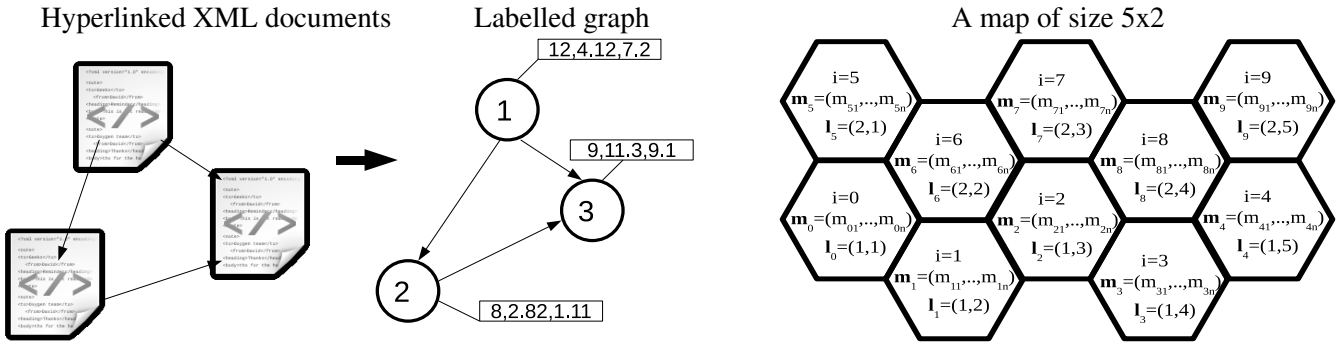

Fig. 1. A set of 3 hyperlinked XML documents (left), the associated graph representation (center), and a SOM of size 6×3 *(right).*

when processing node 1 would be $\mathbf{u}_1 = (12, 4.12, 7.2, m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9)$, where the m_i are computed according to Equation 6 and by using the coordinates of the winners. Again, it can be observed that the state vector component of the codebook vectors, once trained, would encode the context of a given node.

INCOMING AND OUTGOING LINK PREDICTION IN XML DOCUMENTS

While the SOMs described in this chapter are popularly applied to learning problems requiring the clustering or dimension reduction of related data vectors, we will show that it is also possible to predict incoming or outgoing links (in the case of a domain containing directed links) or links to neighbors (in the case of domains containing undirected links).

The input vectors of a SOM-SD contain hybrid information consisting of a data label element and a state vector element. Equation 4 determines the codebook vector which is most similar to an input vector, and Equation 5 reinforces the similarity by adjusting the elements of the codebook vector accordingly. This means that the codebook vectors become more and more similar to the input vectors. Ideally, at the end of the training process there would be one codebook vector for each distinct input vector and the value of the codebook vector would be identical to the associated input vector. This means that the codebook vectors encode the data label components as well as the state information about a node's neighbors. Since the state information refers to a coordinate value of the map, and hence, it is possible to consider the codebook vectors located at such coordinates in order to determine the properties of the child nodes. Recursively applied, it becomes possible to start from a codebook entry at a given coordinate value and then to unfold the associated structural information. Now, the interesting aspect is as follows: assume there is a new node n_k for which the relationship with other nodes in a graph is unknown, then it is possible to compute the location of the best matching neuron (using Equation 4) and then to investigate the state component of the associated codebook vector. The procedure will reveal how n_k is related to other nodes in the training dataset. We refer to this as *codebook-based link inference*.

A second approach to link prediction has been proposed (Kc et al., 2010): It is known that during training a best matching codebook has been obtained for each of the nodes in a graph. Since each codebook may be activated by a number of nodes, and hence, these codebooks are said to be a *representation* of these nodes. Due to the topology preserving ability of the SOMs, it can be said that all nodes which activate the same (or nearby) codebook are most closely related to each other in terms of content and topology. This means that a node mapped at a neuron location is likely to exhibit the same links and link structure as other nodes which are mapped to the same neuron. This gives rise to the possibility of mapping a new node n_k for which the relationship with other nodes in a graph is unknown and then to select a candidate from the training dataset which was mapped to the same location. This candidate can then be expected to exhibit a close resemblance to the features of n_k, and will provide complementary information about the (proposed) topology of n_k. This approach is called a *content-based link inference*.

We note that both approaches can only work if the nodes are labeled by a vector whose dimension is non-zero. Moreover, the former approach is computationally more demanding since a recursive unfolding of the codebook vectors is required in order to obtain the complete view of a graph whereas the latter approach requires considerably more storage since the mapping of all nodes and all graphs in a training dataset needs to be remembered.

Both approaches can also be applied to the CSOM-SD and the PMGraphSOM. In the case of the CSOM-SD, the codebooks of a trained CSOM-SD contain an encoding of the structural relationships with the parents as well as the

children of a node, while in the PMGraphSOM the codebooks contain information about the structural relationships with all the neighbors of a given node.

EXPERIMENTAL RESULTS

Processing and encoding structural information remains a challenging problem and a very dynamic research area. There exist several initiatives which organize annual international competitions on data mining tasks requiring the processing of structured and/or semi-structured information. For example, the Text Analysis Conference (TAC) issues annual challenges on the clustering, classification, or summarization of XML structured document collections. Moreover, the Initiative for the Evaluation of XML Retrieval (INEX) organizes annual competitions on the clustering, classification, as well as on the link prediction tasks in large scale XML document collections. The advantage of such challenges is that they provide an open platform where anyone can participate so that a multitude of comparative results are available. They provide interested researchers a summary of available approaches, and an overview of the state-of-the-art approaches and their results.

In this chapter we will be using a large dataset provided by the INEX 2009 competition in order to demonstrate the capability of the SOMs to perform in link prediction tasks. Here, the prediction will focus on the links between documents within the provided dataset.

Dataset characteristics

The dataset used for the experiments in this chapter is that provided for the INEX 2009 Link-the-wiki challenge. The collection consists of a data dump of the Wikipedia web site taken on 8 October 2008; it contains $2,666,190$ articles, is compressed to 50.7GB in size, and is annotated with the 2008-w40-2 version of YAGO, which is a knowledge base developed at the Max-Planck-Institute Saarbrucken. The knowledge base contains information harvested from Wikipedia and linked to Wordnet. YAGO contains more than 2 million entities (like persons, organizations, cities, etc.), and contains 20 million facts about these entities. YAGO has a manually confirmed accuracy of 95%. It can be queried online. The documents are formatted in XML.

Encoding and Prediction of inter-document links

Given a new Wikipedia document, the aim of this experiment is to extract and analyze the document content, and recommend a set of incoming and outgoing links from the particular document to other documents in the collection. A set of $5,000$ Wikipedia documents was said to have been randomly selected from the collection for the purpose of serving as the testing dataset for the experiment on inter-document link discovery.

Since these selected documents are not truly orphaned documents (as there are orphaned documents which contain no incoming or outgoing links, a situation similar to the newly introduced documents onto the Web), we removed these orphaned documents and their hyperlinks from the collection to simulate the situation of genuinely introducing new documents into the Web. This results in a training dataset consisting of $2,661,190$ documents, all of which do not contain any link information about the $5,000$ documents in the testing dataset.

The training dataset is represented as a directed graph where each node is a web page. Each page is then represented by a state vector encoding both its contextual and link structural information. Before the data can be trained by PMGraphSOM, decisions need to be taken regarding the selection of features to be used to describe each document in the form of node labels. It was important to select a feature which could represent each document, but such features cannot contain any link information, as otherwise the testing process would not be able to use the same feature as node labels to represent the test data.

Some analysis revealed that the category information associated with Wikipedia documents could be used as a representative feature. The category extraction process identified $8,918,924$ categories in total, which could be consolidated to $362,251$ unique categories. The maximum number of categories to which a single document belonged is $2,022$, whereas there are $118,209$ documents with no associated category information at all. If the category information is to be used without any processing, the large dimension would prevent the training process from completing within a reasonable amount of time. The experiments were carried out on machines fitted with 2.3GHz AMD Opteron CPUs. We considered 36 hours reasonable for training a network (A faster CPU or the parallelization of the implementation would significantly reduce the training times, and hence, would help to avoid the dimension reduction step used here); therefore, we decided to perform some dimension reduction first.

	Max	Min	Standard deviation	Num. of documents without links
Out-degree	5295	0	92.96	36,978
In-degree	549,658	0	476.18	304,518

Table 1. Statistics of the link structure of the training dataset.

	Total	Max	Min	Mean	Std. dev.	Num. of docs. without links
Out-degree	461,741	245	1	92.35	46.05	0
In-degree	311,423	3095	0	62.28	131.50	372

Table 2. Statistics of the link structure of the test-dataset.

Singular Value Decomposition (SVD) was considered for a dimension reduction step (Golub and Kahan, 1965). However, SVD would require the building of a $2,661,190$ x $362,251$ matrix which is far too large for the capacity of computers which we had available for this project. Hence, another approach to dimension reduction was taken. This second approach utilizes a well known Multi-Layer Perceptron (MLP) algorithm (Haykin, 1994) to assist in dimension reduction. The MLP algorithm is generally applied to neural network architectures which consist of an input layer, followed by one or more hidden layers of neurons, and then an output layer of neurons, where all neurons in a layer are fully connected to all neurons in the next layer (Haykin, 1994). To utilize MLP for dimension reduction we use an architecture known as the "Auto-Associative Memory" (AAM) architecture (Rumelhart and McClelland, 1986). In an AAM, both the input and output dimensions are the number of unique categories ($362,251$), and the dimension of the single hidden layer will be the dimension with which we would like to reduce it to. Using this configuration, the number of neurons (dimension) in the hidden layer is less than the dimension of the input layer or the output layer, so the encoding process in the hidden layer is a compression of information from the input. Then the connection between the single hidden layer neurons and the output layer neurons can be seen as un-compressing the information back to the original dimension.

For training purposes, the input data is also used as the target, so that the MLP can learn a mapping which loses the least amount of information through the compression (hidden) layer of an auto-associative memory. It is known that an MLP trained in this fashion results in a dimension reduction which is qualitatively equivalent to those obtained by using a SVD algorithm but without the need of having to store a large matrix in memory.

After the node labels are reduced to a more manageable dimension (here we use 16) using the MLP technique as indicated, attention was shifted to the incorporation of link structures within the training dataset. PMGraphSOM is capable of incorporating link information to assist in the training process, therefore the link structure of the training dataset could be used for training purposes without further processing. However, some analysis on the links within the dataset revealed that the total number of links is $136,304,216$; this equates to a mean of approximately 51.22 links per page. These links are unlikely to be distributed equally, therefore, separate analysis of the out-links and in-links were also conducted. The statistical results can be found in Table 1. The standard deviation indicates that the number of in-links varies much more than the number of out-links. This property is important since it implies that the dataset is unbalanced with respect to the incoming and outgoing links. Such imbalances in the training dataset are known to potentially cause problems with any machine learning approach.

Although the link structure of the testing dataset will not be used during training, some analyses were carried out to ascertain if the testing dataset and the training dataset are comparable. This is especially important in machine learning, as a training dataset, which is representative of the testing dataset, will be able to provide more accurate results for the documents the network has not encountered previously. The statistics of the link structure for the testing dataset are included in Table 2.

As can be observed from comparing the statistical information of the link structure, the training dataset and the testing dataset have a number of significant differences.

- The number of documents with no links - It can be observed that a little more than 1% of documents in the training dataset have no out-links. Based on this ratio, approximately 50 documents in the testing dataset are expected to

have no out-links, but the testing dataset contains no such type of documents. The in-link also has a similar problem: with the training dataset containing approximately 11% of documents with no links, in comparison with 7% in the testing dataset.
- A higher number of links in the testing dataset - The average number of out-links and in-links per document in the testing dataset are consistently higher than the 51.22 links per page average in the training dataset.

These differences suggest that the testing dataset is not a random sampling from the problem domain, but rather a subset selected by some (unknown) criteria. Again, this can impose some added challenges to a machine learning approach.

Learning for link discovery

We learn patterns of the Wikipedia links by feeding the training dataset to a PMGraphSOM. After training, all Wikipedia pages will be mapped onto a 2-dimensional display space where pages with similar contextual content and link structure will ideally be mapped onto the same codebook or onto nearby codebooks. Based on this property of the SOM training algorithm, we are able to discover links for the $5,000$ testing pages by inferring from the codebooks of a trained map.

We used two approaches to infer from the trained map, and they are the codebook-based and the content-based link inference approaches described in Section 2. Each of these two approaches can produce an arbitrary number of links which is only limited by the size of the map (for the codebook-based approach), or by the existing link structure of the training dataset (for the content-based approach). These inferred links are then to be ranked in descending order from the most likely link to the least likely link. Then we will truncate the list of links to the maximum allowable 250 for both the in-links as well as the out-links. Note that the computed rank values will also play an important role in the evaluation of the results.

Three ranking algorithms were considered for this task.

(1) The first is based on the energy flow of a page (Page et al., 1998). This is calculated by accumulating scores when a page receives in-links, but distributing scores when a page contains out-links. The list of proposed out-links for each of the documents in the test dataset are ordered according to their associated scores. The inverse of accumulating scores from the out-links, and distributing scores to the in-links, is used to order the list of proposed in-links for each test document.
(2) The second ranking algorithm is based on frequency. For example, if many of the training documents indicate that a link is a likely in-link or out-link, then it has a higher likelihood of being proposed, and therefore will be ranked higher.
(3) The third ranking algorithm is based on the Euclidean distance of the test document and the training documents. This ensures that the training documents with more contextual feature similarity are ranked higher.

We obtained results for each of the two link inference approaches, and investigated the impact of the three ranking mechanisms.

Unless specified otherwise, the training of the PMGraphSOM used the training parameters of 3 iterations, a radius of 10, an initial learning rate of 0.9 and a seed number of 7. Other parameters such as the size of the map, and the weight μ were varied as indicated later in the paper.

From the above list of training parameters, the number of iterations refers to the number of times the entire training dataset was processed during training. Due to the large number of training data, and due to the fact that a PMGraphSOM is updated on every node of an input graph, and hence, within each of the training iteration, the PMGraphSOM is updated $2,661,190$ times. Since the dataset contains a large degree of redundant (or very similar) information, it suffices to train the map for a relatively small number of iterations. The radius refers to the distance from the winning neuron within which update will take place. Moreover, since the mapping is expected to be more randomized in the early phase of training, and then become more stable and more representative as training progresses, the number of codebooks around the winning neuron which need updating is also expected to decrease. This is in fact implemented in practice by decreasing the radius as training progresses, therefore, the specified radius value only refers to the initial radius, and subsequent radius values are calculated based on the training progress until it approaches a value of 1. Learning rate governs how much update should take place, and uses a similar approach as radius, in terms of its strength as training progresses, so the learning rate specified is the initial learning rate. Through the training

iteration, learning rate will approach 0; this is commonly done with SOM based algorithms to aid convergence. The grouping concept was introduced with the PMGraphSOM approach. The grouping of codebooks allows the granularity of the map to be altered for training, and improves training efficiency as this reduces the dimension of the codebook vector. Training with groupings were attempted, but they did not produce improvements for the testing dataset, therefore for the experiments conducted, no grouping was used. The seed value was used to simply allow the purpose of being able to reproduce the results, as if it was not specified, a random seed value would have been used every time the algorithm is applied. The seed influences the initial condition of the network.

Fixing the above mentioned training parameters allows other parameters to be varied and tested. The parameters under investigation here are the map size and the weights μ. We attempted a large number of combinations of map sizes and weights, the resulting trained maps were analyzed on the testing dataset. We selected the five most representative results submitted by us to the INEX LinkTheWiki track for the purpose of visualizing the results in this chapter. The map size and weights for corresponding submissions are indicated in Table 3. These submitted training tasks were the only results produced from unsupervised machine learning approaches amongst the approaches attempted by other researchers (He and de Rijke, 2010; Itakura and Clarke, 2010).

Submission ID	Map size	μ_1	μ_2	μ_3	Trained map
01	-	-	-	-	-
02	20x40	0.999	0.0005	0.0005	Fig. 2
03	10x30	1.0	0.0	0.0	Fig. 3
04	20x40	0.991	0.0045	0.0045	Fig. 4
05	20x40	0.991	0.0045	0.0045	Fig. 4

Table 3. Statistics of the test set's link structure

The first submission does not have any associated information about the training process, because it was not trained, but merely ranked. PMgraphSOMs were trained for submission associated with the IDs 02 to 05. The map size refers to the horizontal and vertical extent of the 2-dimensional map $\boldsymbol{M}$. The parameters μ_1, μ_2 and μ_3 are weights,

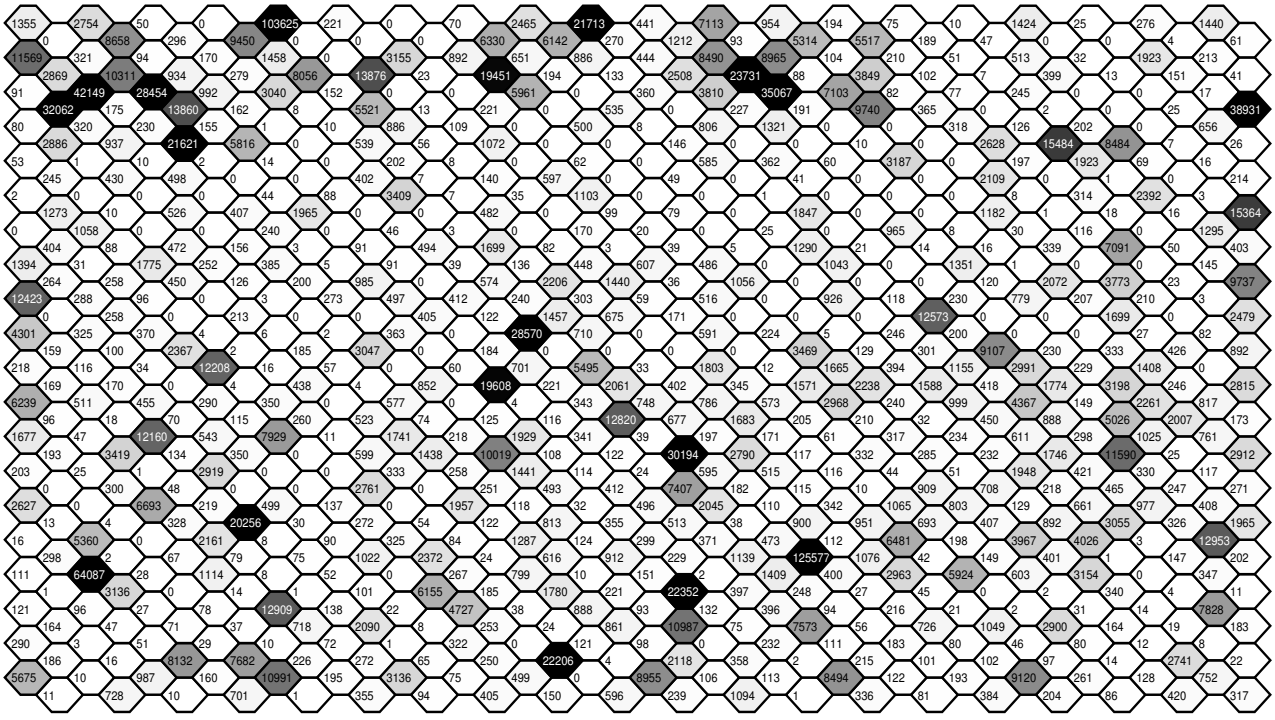

Fig. 2. The trained map corresponding to submission 02. Larger maps generally produced a refined clustering.

and $\sum_{i=1}^{3} \mu_i = 1$. μ_1 is the weight associated with the node labels, μ_2 is the weight associated with the out-links (the vectors associated with the out-links), and μ_3 is the weight associated with the in-links (the vectors associated with the in-links). In other words, we use the state of $\boldsymbol{M}$ for the children of a node, and a second state of $\boldsymbol{M}$ for the parents of a node. This is because otherwise the PMGraphSOM considers the links as being undirected whereas the task is to detect both, in-links and out-links rather than all the neighbors of a node. A large variation of the weights were attempted during training, and it was observed that the weights associated with the links are generally significantly smaller than the weights assigned to node labels; this could be attributed to the differences in the dimensions, and the magnitude of the vectorial elements. For example, in this experimental set up, a 16 dimensional vector is used to represent the node labels, whereas the information about the links (both in-links and out-links) are represented by much larger vectors that are dependent on the map size. It should be noted that submissions 04 and 05 are based on the same trained map, but different ranking algorithms were applied to produce different sets of results.

The frequency of activation of each codebook after the training process (the trained map) can be observed in Fig. 2, Fig. 3 and Fig. 4 respectively. In these figures, the number in each cell (codebook) indicates the number of activation, and to provide a better overview of the maps, the cells are coloured in grey scale with darker cells indicating higher activation. A group of adjacent dark cells surrounded by light-coloured cells indicates a cluster. Some clusters seem to have clear boundaries, such as the cluster in the lower-right quadrant in Fig. 3, while some clusters may appear to overlap with another cluster, such as the clusters stretching horizontally across the centre towards the right-hand side of the map in Fig. 3. The ideal map would be one with several distinct clusters formed by similar documents mapped in the same or in neighbouring cells, while a poorly trained map would show a more random activation, which often makes the identification of clusters difficult. The trained maps illustrated here show reasonable clustering on all maps, exhibiting some distinct clusters.

Results and evaluation

The performance measure for participating in the challenge includes Precision, Recall, Mean Average Precision (MAP) and interpolated precision (Precision @R). The evaluation mechanism used to compare the results across the participating groups is the interpolated precision. Therefore the evaluation mechanism used for this task is also based on the interpolated precision.

Interpolated precision is precision evaluated on different granularities to observe the performance achievable when varying the number of top ranked links considered. If precision is defined as $\frac{|t \cup r|}{|r|}$, calculating precision for the top n links for each page would be calculated by restricting the size of t to a maximum of n. This is expected to reveal the effectiveness of the ranking algorithms used.

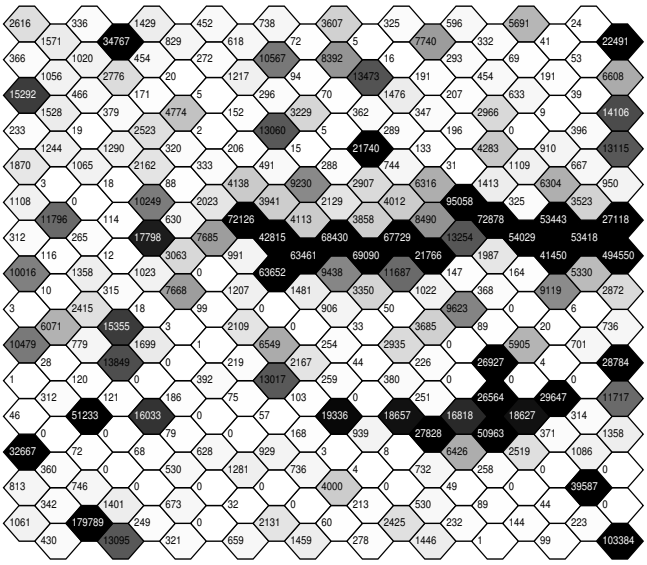

Fig. 3. The trained map corresponding to submission 03.

Submission ID	01	02	03	04	05
Link inference method	Consider all	Content-based	Content-based	Codebook-based	Content-based
Ranking algorithm	Energy flow	Link frequency	Link frequency	Euclid. distance	Euclid. distance
Recall for out-links	0.00377	0.02942	0.03202	0.00070	0.00448
Precision@250 out-links	0.00136	0.01057	0.01817	0.00025	0.00161
Precision@100 out-links	0.00168	0.01795	0.02162	0.00002	0.00147
Precision@20 out-links	0.00040	0.03941	0.04489	0.00003	0.00268
Recall for in-links	0.00032	0.00050	0.00095	0.00010	0.00029
Precision@250 in-links	0.00008	0.00012	0.00025	0.00002	0.00007
Precision@100 in-links	0.00007	0.00008	0.00037	0.00003	0.00008
Precision@20 in-links	0.00007	0.00018	0.00107	0.00001	0.00008

Table 4. The performance of proposed links as predicted by our proposed algorithms.

The performance measured using interpolated precision and recall for the 5 submissions are included in Table 4. Each of the submissions use a different combination of training configuration and ranking algorithm. Although the results may appear generally low, with the Precision@250 ranging from 0.00002 to 0.01817, however, this marks a vast improvement over the results obtained by using a random process. The accuracy of a random process is less than 0.00001 on an average, and hence the proposed approach performs by 3 to 5 orders of magnitude better. It is important to note that these results were obtained using an unsupervised machine learning approach, so it cannot be expected to perform as well as supervised approaches that utilize teacher signals for guidance and correction during training.

Submission 01 is the best result achievable without using a trained map, but instead, simply apply the ranking algorithm over the entire dataset, and identify the top few ranked pages. As may be observed from Table 4, this simple procedure produces one of the worst results when compared with those obtained using the training process.

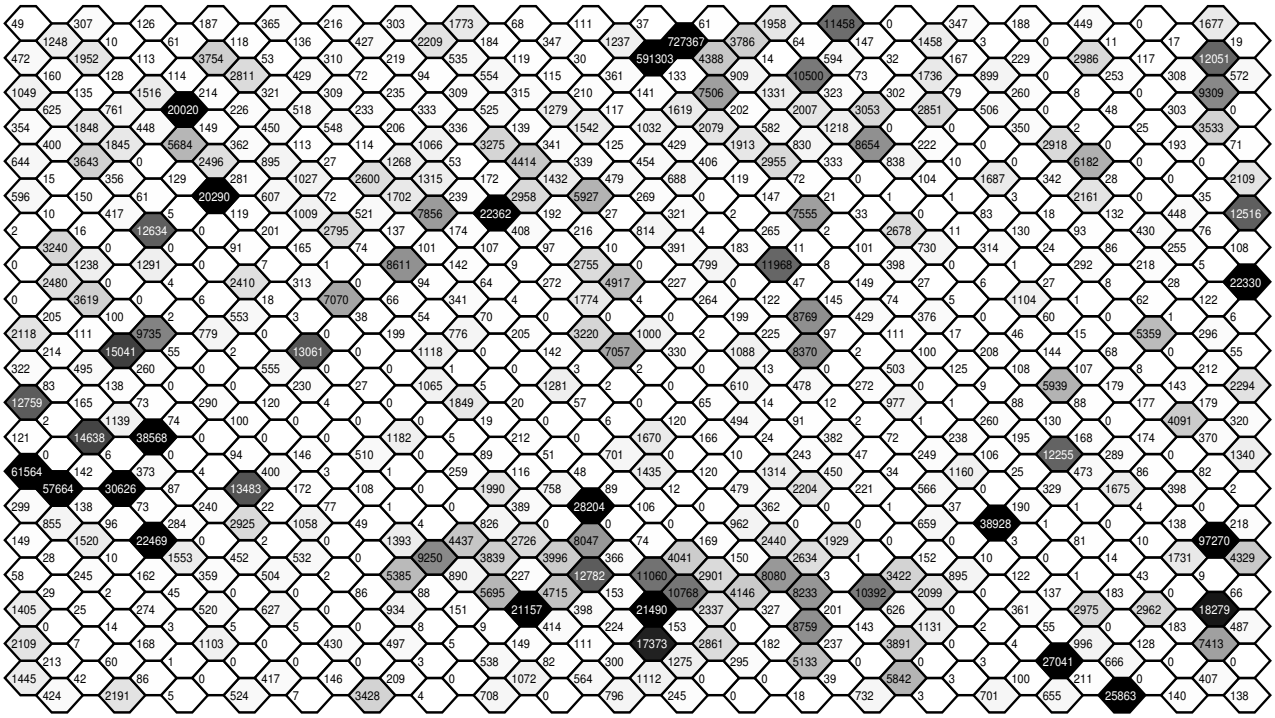

Fig. 4. Visualization of the mappings on the trained map corresponding to submissions 04 and submission 05.

Although most of the results from a trained map out-performed the un-trained result, the best performance using the trained data, is obtained from submission 03, which uses content-based link inference approach, and is ranked by link frequency. The associated map is visualized in Fig. 2. The visualization indicates that the increased size of the display space allows for a more sharply defined clustering of the nodes. When training the same size of a map by strengthening the weight μ_2 and μ_3 (the influence of the state vectors) this produced a mapping as shown in Fig.4. The effect is a reduction in the number of clusters while strengthening the remaining clusters (the number of activations within a cluster increased sharply) causing an overall reduction in the accuracy of the predicted links. Hence, this indicates the importance of the information provided by the data label (weighted by μ_1) in link prediction tasks. The main reason for this is that the test data does not contain any link information, and hence, the similarity with codebook values is based solely on the content based similarity provided through the data label component.

These representative results show that the content-based link inference approach produces better performance than the codebook- based link inference approach, and that the ranking by link frequency produces better results than ranking by Euclidean distance. Another observation from Table 4 is that no matter which inference method or ranking algorithm is used, the in-links proposed are less accurate than the out-links proposed. These results and observations allow us to draw the conclusion that encoding the link structure and incorporating textual information about the documents in the training process is able to achieve a significantly improved performance; provided that the link inference approach and ranking approach are appropriately chosen. It was also shown that by increasing the size of the SOM, this will enhance the precision of the prediction at the cost of additional computational time. Thus, the precision of the SOM for link prediction tasks is constrained by the discrete nature of the mapping space.

We note that the approach described in this chapter is *unsupervised*. To the best of our knowledge, there exists no other unsupervised machine learning approach to link prediction at this time. The dataset used is a benchmark problem which has been used by a number of others. The state-of-the-art performance for this benchmark problem is, when $R = 250$, an interpolated precision of approximately 0.05. This was achieved by a supervised approach (Huang et al., 2010).

The experiments in this chapter have demonstrated that extensions of SOM for encoding structured data, in particular PMGraphSOM, is capable of encoding and predicting inter-document links in a dataset, given the appropriate features, inference approaches and ranking methods. This has implications on its potential to solve other real-word problems, as there could be two directions to the link prediction problem: the prediction of inter-document links such as hyperlinks as was presented in this chapter, and the prediction of intra-document links such as the relationship between elements of a single document. It is also possible to predict both types of links at the same time. The latter requires what is known as the Graph-of-graph (GoG) concept (Zhang et al., 2010) though the GoG is yet to be explored for the unsupervised case. The GoG concept recognizes that intra-document links depict the relationships among the nodes of a document. These nodes can be words, or paragraphs in the document. Whereas for inter-document links, they express the relationship between documents. For example, one document may be dependent on another document (it makes a reference to this other document). This other document may in turn be dependent on yet another document (it makes a reference to this other document). If each document is modelled by a graph structure consisting of nodes and links (intra-document links), then the dependency between documents can be represented as a directed link from one document to another. Thus, this creates a model in which one graph is related to another graph, and this other graph is in turn related to yet another graph. This is what we called a GoG situation. This is quite common in XML documents or web pages in which there are references to other documents or web sites. As indicated, we have yet applied to this case. In this chapter we have not considered the GoG models. In other words, each document is considered by itself, without dependency except the number of links and the locations in which they arrive from other XML documents, or going to other XML documents. However, it can be observed that the GoG model can be achieved through extensions to the PMGraphSOM. The main challenge with this is how such an extended PMGraphSOM can be trained in a scalable fashion.

Time complexity of the presented approach has been addressed in another publication (Hagenbuchner et al., 2003). In summary, the computational time complexity of the presented approach grows linearly with the number of nodes in a dataset (Hagenbuchner et al., 2003). This behaviour is analogous to the time complexity of the standard SOM approach (which grows linearly with the size of the training set). Thus, the approach presented in this paper is suitable for large scale data mining tasks.

FUTURE RESEARCH DIRECTIONS

Graph mining, link analysis, and link prediction are hot research topics. Numerous directions are currently being explored. Future research directions as are relevant to the materials presented in this chapter would be as follows:

Continuous mapping spaces: Self-Organizing Map approach consist of a set of codebook vectors arranged on a regular grid. This means that the number of codebook vectors is limited, and that the layout of the grid discretizes the mapping of the display space. The aim of a future research project could be to introduce a mapping space that is continuous. One possibility would be to expand on the "probability measure" idea expressed in this chapter by describing a mapping by a Gaussian function rather than by coordinate values. In other words, each codebook could be interpreted as the representative of a multi-variate Gaussian function. The median of the Gaussian would be fixed at a given location, but the description of a mapping would be continuous. The details to how this can be achieved are yet to be explored.

Convergence theorem and proof: There exists no proof of convergence for any of the many Self-Organizing Map approaches. This is mostly due to the absence of an error function in the training algorithms for a SOM. If one could introduce some form of a continuously differentiable error function to the SOM training algorithm, then it should be possible to formally analyse the convergence properties of a SOM.

Supervised link prediction approaches: Supervised machine learning approaches could be expected to outperform most unsupervised approaches. There is currently no known supervised machine learning approach suitable for link prediction tasks. The main problem is that existing supervised machine learning methods are unable to reverse the information stream through a given model (to conclude from a given output what the associated input would have looked like). It may be possible to explicitly train a supervised machine learning method by adopting an auto-association approach (i.e. by extending the model by a second component which is trained to reconstruct the input from a given output). This approach remains to be explored in details.

Extension to hypergraphs: Hypergraphs are a generalization of graph data structures in that its links can represent a one-to-many relationship. This means that hypergraphs allow nodes to be connected to several other nodes by a single link. Hypergraphs are suitable to encode relationships as they occur in several applications such as, for example, electrical circuits, molecular chemistry. There is currently no known machine learning approach capable of encoding hypergraphs. A possibility may be to represent each link by a neural network. This would be in addition to encoding each node by a neural network as was described in this chapter. Effectively, this could mean that a hypergraph could be encoded by training two maps concurrently; one for the encoding of the nodes, and a second one for encoding the links of a hypergraph. A suitable training algorithm would need to be developed.

CONCLUSION

The work presented in this chapter is the latest in the area of unsupervised machine learning in the domain of graph structured information. The approach presented in this chapter is the only known unsupervised machine learning approach that can encode graph data structures and, once trained, can be used to reconstruct the original structure by starting from a single node. The simplicity and scalability of the algorithm are some of the properties which make its application to very large learning problems on link prediction a possibility. While the accuracy of the methods improves the prediction of links by three orders of magnitude when compared with those obtained by using random selections, it was also shown that the cardinality of the Self-Organizing Map can be an inhibiting factor on the accuracy of a predicted link. The avoidance of this cardinality problem, and the development of supervised approaches to link prediction remain open problems yet to be solved. These would pose as challenges for future research.

The approach presented in this paper is also beneficial for tasks requiring the clustering of structured information. For example, the SOM-SD holds the state-of-the-art performance for several benchmark problems (provided by INEX) on the clustering of relatively large sets of XML documents (Hagenbuchner et al., 2006; Kc et al., 2007), and has been successfully used as a filter on a spam detection task in hyper-linked domain (Di Noi et al., 2010). The versatility of the method underlines the benefits of data driven approaches, and shows that SOM based methods continue to be beneficial for solving real world data mining problems.

Acknowledgements: The authors wish to acknowledge partial financial support from an Australian Research Council Discovery Project grant DP077148 (2007 - 2009) for the work which is reported in this chapter.

REFERENCES

Bengio, Y., Frasconi, P., and Simard, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(5):157–166. Special Issue on Recurrent Neural Networks.

Bianucci, A., Micheli, A., Sperduti, A., and Starita, A. (1998). Quantitative structure-activity relationships of benzodiazepines by recursive cascade correlation. In *Proceedings of the IEEE World Congress on Computational Intelligence (IJCNN'98)*, Anchorage, Alaska.

Bunke, H. (1990). String grammars for syntactic pattern recognition. In Bunke, H. and Sanfeliu, A., editors, *Syntactic and Structural Pattern Recognition, Theory and Applications*, pages 29–55. Ed. World Scientific, Singapore.

Di Noi, L., Hagenbuchner, M., Scarselli, F., and Tsoi, A. C. (2010). Web spam detection by probability mapping graphsoms and graph neural networks. In Diamantaras, K., Duch, W., and Iliadis, L., editors, *Artificial Neural Networks (ICANN 2010)*, volume 6353 of *Lecture Notes in Computer Science*, pages 372–381. Springer Berlin / Heidelberg.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science 14*, pages 179–211.

Fu, K. (1977). Tree languages and syntactic pattern recognition. In Chen, C., editor, *Pattern Recognition and Artificial Intelligence*, pages 257–291. Academic Press.

Golub, G. H. and Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):205–224.

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

Hagenbuchner, M., Sperduti, A., and Tsoi, A. C. (2005). Contextual self-organizing maps for structured domains. In *Workshop on Relational Machine Learning*.

Hagenbuchner, M., Sperduti, A., Tsoi, A. C., Trentini, F., Scarselli, F., and Gori, M. (2006). Clustering xml documents using self-organizing maps for structures. In et al., N. F., editor, *LNCS 3977, Lecture Notes in Computer Science*, pages 481–496. Springer-Verlag Berlin Heidelberg.

Hagenbuchner, M., Tsoi, A. C., and Sperduti, A. (2001). A supervised self-organising map for structured data. In Allison, N., Yin, H., Allison, L., and Slack, J., editors, *WSOM 2001 - Advances in Self-Organising Maps*, pages 21–28. Springer.

Hagenbuchner, M., Zhang, S., Tsoi, A. C., and Sperduti, A. (2009). Projection of undirected and non-positional graphs using self organizing maps. In *European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning*, pages 559–564, Bruges, Belgium.

Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 866 Third Avenue, New York 10022.

He, J. and de Rijke, M. (2010). An exploration of learning to link with wikipedia: Features, methods and training collection. In Geva, S., Kamps, J., and Trotman, A., editors, *Focused Retrieval and Evaluation*, volume 6203 of *Lecture Notes in Computer Science*, pages 324–330. Springer Berlin / Heidelberg.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Huang, W., Geva, S., and Trotman, A. (2010). Overview of the inex 2009 link the wiki track. In Geva, S., Kamps, J., and Trotman, A., editors, *Focused Retrieval and Evaluation*, volume 6203 of *Lecture Notes in Computer Science*, pages 312–323. Springer Berlin / Heidelberg.

Itakura, K. and Clarke, C. (2010). University of waterloo at inex 2009: Ad hoc, book, entity ranking, and link-the-wiki tracks. In Geva, S., Kamps, J., and Trotman, A., editors, *Focused Retrieval and Evaluation*, volume 6203 of *Lecture Notes in Computer Science*, pages 331–341. Springer Berlin / Heidelberg.

Kc, M., Chau, R., Hagenbuchner, M., Tsoi, A. C., and Lee, V. (2010). A machine learning approach to link prediction for interlinked documents. In Geva, S., Kamps, J., and Trotman, A., editors, *Focused Retrieval and Evaluation*, volume 6203 of *Lecture Notes in Computer Science*, pages 342–354. Springer Berlin / Heidelberg.

Kc, M., Hagenbuchner, M., Tsoi, A. C., Scarselli, F., Gori, M., and Sperduti, S. (2007). Xml document mining using contextual self-organizing maps for structures. In *Lecture Notes in Computer Science*, volume 4518, pages 510–524. Springer-Verlag Berlin Heidelberg.

Kohonen, T. (1984). *Self-Organisation and Associative Memory*. Springer, 3rd edition.

Kohonen, T. (1995). *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg. (Second Extended Edition 1997).

Kohonen, T. (1997). Exploration of very large databases by self-organizing maps. In *IEEE International Conference on Neural Networks*, pages 1–6.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University.

Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing*, volume 1. MIT Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Cambridge, MA, USA.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009a). Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009b). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Takács, G., Pilászy, I., Németh, B., and Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656.

Zhang, S., Hagenbuchner, M., Scarselli, F., and Tsoi, A. C. (2010). Supervised encoding of graph-of-graphs for classification and regression problems. In Geva, S., Kamps, J., and Trotman, A., editors, *Focused Retrieval and Evaluation*, volume 6203 of *Lecture Notes in Computer Science*, pages 449–461. Springer Berlin / Heidelberg.

Zhang, S., Hagenbuchner, M., Tsoi, A. C., and Sperduti, A. (2009). Self organizing maps for the clustering of large sets of labeled graphs. In et al., N. F., editor, *LNCS 4862, Lecture Notes in Computer Science*, pages 207–221, Berlin. Springer-Verlag Berlin Heidelberg.

ADDITIONAL READING SECTION

Hammer, B., Micheli, A., Sperduti, A., and Strickert, M. (2004). A general framework for unsupervised processing of structured data. *Neurocomputing*, 57:3–35.

Hammer, B. and Sperschneider, V. (1997). Neural networks can approximate mappings on structured objects. In *Proc. of the 2nd International Conference on Computational Intelligence and Neuroscience (ICCIN'97)*, volume 2, pages 211–214.

Frasconi, P., Gori, M., and Sperduti, A. (1998). A general framework for adaptive processing in data structures. In *IEEE Trans on Neural Networks*, volume Vol 9, pages 768–785.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166. Special Issue on Recurrent Neural Networks.

Roja, R. (1996). *Neural Networks: A Systematic Introduction*. Springer-Verlag, New York.

Hagenbuchner, M. (2002). *Adaptive Processing of Structured Information using Artificial Neural Networks*. PhD thesis, University of Wollongong.

Bellman, R. E. (1961). *Adaptive control processes: a guided tour*. Princeton University Press.

Tan, P.-N., Steinbach, M., Tan, V. K., and Pang-Ning, editors (2005). *Introduction to data mining*. Pearson Addison Wesley.

Hartigan, J. A. (1975). *Clustering algorithms*. Wiley.

Papadimitriou, C. (1994). *Computational Complexity*. Addison Wesley, 1st edition.

Aho, A. and Ullman, J. (1992). *Foundations of Computer Science*. Online: http://infolab.stanford.edu-/∼ullman/focs.html.

Erwin, E., Obermayer, K., and Schulten, K. (1992). Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55.

Gonzalez, A., Graña, M., D'Anjou, A., Albizuri, F., and Cottrell, M. (1997). A sensitivity analysis of the self organizing map as an adaptive one-pass non-stationary clustering algorithm: the case of color quantization of image sequences. In *Neural Processing Letters*, volume 6, pages 77–89.

Hagenbuchner, M., Gori, M., Tsoi, A., Bunke, H., and Irniger, C. (2002). Using attributed plex grammars for the generation of image and graph databases. In Vento, M., editor, *Special issue PRL-Graph-based Representations*.

Hassoun, M. (1995). *Fundamentals of Artificial Neural Networks*. MIT Press.

Pollack, J. B. (1989). Implications of recursive distributed representations. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, pages 527–536. Morgan Kaufman, San Mateo, CA.

Yin, H. and Allinson, N. (1995). On the distribution and convergence of feature space in self-organizing maps. *Neural Computation*, 7:1178–1187.

KEY TERMS & DEFINITIONS

Clustering: This refers to the grouping of data. The general aim of clustering is to organize data such that data that share certain similarities are grouped together whereas dissimilar data are grouped more distant. Clustering algorithms can be computationally very efficient. Hence, clustering can be used as a quick way to identify redundancies in a dataset, or as a filtering tool. K-means and SOM are two of the most famous and widely adopted clustering algorithms

Euclidean distance: This is the most commonly used measure for computing the distance of two points in a metric space. The formula computes the square root of the sum of squared differences of all vector elements. The Euclidean distance suffers from the *curse of dimensionality*, a problem caused by the exponential increase in volume with the increase of dimensions. Due to the lack of better alternatives, the Euclidean distance is also used when computing distances in high-dimensional data space.

Graph: A collection of atomic entities for which a binary relationship exists is called a graph. The atomic entities are said to be the nodes of a graph, and the binary relations are said to be the links that connect a pair of nodes of a graph. For example, in molecular chemistry, each atom can be represented as a node, and the chemical binding with other atoms in the molecule can be represented by a link. Graphs are said to be directed if there exists a causal relationship between nodes. For example, if a link has a source and a destination as is the case with hyperlinks and citations, then the links are directed as is the associated graph. Otherwise the graph is said to be undirected.

Inlink and outlink: A directed graph contains directed links, each of which has a unique source node and unique target node. The link originating from a given node is said to be an outlink of the node whereas a link targeting a node is said to be an inlink to the node. The total number of outgoing links of a node is called the *outdegree* of a node whereas the total number of inlinks is the *indegree* of a node.

Link Prediction: A dependency between data is commonly represented as a *link* between related data. For example, a citation within a scientific document links the two documents. Link prediction is the task of assessing whether or not a link should exist between any given data pair.

Machine learning: This is a scientific discipline that aims at developing algorithms that allow computers to learn from data. The discipline is inspired by the ability of biological neural systems (the brain) to adapt itself to sensory information. This has made Artificial Neural Networks the most important research direction in machine learning.

Self Organizing Maps: This is an unsupervised machine learning approach which mimics the ability of biological neural systems to organize and specialize neurons in a particular way. The result is that Self-Organizing Maps perform a clustering of data such that data which share similar properties are grouped together while data which differ from each other are mapped apart from each other. The projections obtained are also popularly used to visualize high-dimensional data.

SOM-SD: A Self Organizing Maps capable of encoding directed acyclic ordered graphs; a class of graphs known as directed trees. The SOM-SD processes graphs by taking one node at a time, and by dynamically forming an input representation of the node. A training algorithm updates network parameters such that a topology preserving mapping is obtained for each node in a given training data set. The SOM-SD is normally trained unsupervised. A SOM-SD is particularly useful for data mining tasks requiring the clustering and projection of tree data structures. Since a SOM-SD encodes the context of each node, and hence, it can also be deployed successfully to link prediction tasks.

Supervised Learning: Learning problems that consist of a training dataset containing data *pairs*. Each data pair consists of an input object and associated target object. The purpose of supervised machine learning is to train a model such that for each input object the desired output object is generated, and once trained, that the model can be used to generate useful output for data with a missing target object. The latter is said to be the models ability to *generalize* to unseen data, and is one of the most significant and sought-after abilities of supervised machine learning algorithms.

Unsupervised Learning: Assume that we have collected data and that we have no knowledge on the meaning of the data. Then, any attempt to engage a machine learning approach would need to be done in an unsupervised fashion. this means that the machine learning approach would not require any a-priory information about the meaning of the data. In contrast, supervised learning method rely on the availability of some relevant knowledge about the meaning of data. For example, given a set of images, we may or may not know what is depicted in those images. If we know what is depicted on a set of images, then we can train a machine learning approach supervised. Otherwise we need to adopt an unsupervised machine learning approach.