A Self-Organising Hybrid Model for Dynamic Text Clustering

Chihli Hung and Stefan Wermter
Centre for Hybrid Intelligent Systems
The University of Sunderland
[chihli.hung; stefan.wermter]@sunderland.ac.uk
www.his.sunderland.ac.uk

Abstract

A text clustering neural model, traditionally, is assumed to cluster static text information and represent its inner structure on a flat map. However, the quantity of text information is continuously growing and the relationships between them are usually complicated. Therefore, the information is not static and a flat map may be not enough to describe the relationships of input data. In this paper, for a real-world text clustering task we propose a new competitive Self-Organising Map (SOM) model, namely the Dynamic Adaptive Self-Organising Hybrid model (DASH). The features of DASH are a dynamic structure, hierarchical clustering, non-stationary data learning and parameter self-adjustment. All features are data-oriented: DASH adjusts its behaviour not only by modifying its parameters but also by an adaptive structure. We test the performance of our model using the larger new Reuters news corpus based on the criteria of classification accuracy and mean quantization error.

1 Introduction

Clustering by document concepts is a helpful way for linking a query to relevant information. One well-known project is WebSOM [1]. WebSOM employs a Self-Organising Map (SOM) for clustering documents and presents them on a 2-dimensional map. The SOM, proposed by Kohonen in the 1980s, applies a pre-defined topological structure and a time-based decaying learning rate to function as a powerful tool for non-linear projection, vector quantization, and data clustering tasks [2]. In terms of a real-world text clustering task, however, the quantity of text information is continuously growing so the information is not static. This information usually has some relationship with time, for instance, news. Some specific events often occur during a specific period and the recent information is

more important. Moreover, it is not easy to decide the number of clusters for a complicated text set, which should be further analysed based on a hierarchical architecture.

Therefore, a SOM clustering model with a very large map is not preferred. A model which contains a time-based decaying learning function and pre-defined topological structure is not suitable for such a real-world text clustering task. Several alternative models have been proposed to enhance the practicability of the SOM. However, none of the existing models meet all the needs of the features required for a real-world text clustering task. This leads to the development of a new algorithm, the Dynamic Adaptive Self-Organising Hybrid model (DASH).

The remainder of this paper is organised as follows. In Section 2, we give a brief survey of the related competitive neural learning models. In section 3, we introduce the features of the DASH and its algorithm. Section 4 includes the experiments and comparisons using three scenarios which test our model based on the static data set and non-stationary data set. A conclusion is given in section 5.

2 The Competitive Neural Learning Models

Due to the deficiencies of the SOM, several related unsupervised neural learning models have been proposed. They are based on the competitive learning technique whose learning adjustments are confined to a neuron that is most activated to the stimulus currently being presented [2, 3]. This pure competitive learning has a feature of "winner-take-all". Compared with a neural system, the neighbours of the winner neuron are also activated to a stimulus. The lateral relationships of neurons and the winner affect the extent of activation for the neighbours of the winner. Thus both the winner and its neighbours are activated to a stimulus, which form a "winner-take-most" model.

In a clustering task, we use a unit to represent the neuron, a connection to represent the relationship and an input data vector to represent the stimulus. All the relationships of the units form a topology of a competitive neural model. The winner is represented by the Best Matching Unit (BMU) which is defined as the unit of the model with the shortest Euclidean distance to its associated input vectors. A common goal of these algorithms is to map a data set from a high-dimensional space onto a low-dimensional space, and keep its inner structure as faithful as possible. We divide these models into four groups, which are static models, dynamic models, hierarchical models and non-stationary distribution learning models.

Static models, such as the Neural Gas (NG) [4], and dynamic models, such as the Growing Grid (GG) [5], Growing Cell Structure (GCS) [6], Growing Neural Gas (GNG) [7], Incremental Grid Growing (IGG) [8] and Growing SOM (GSOM) [9], try to define a new architecture with no need of prior knowledge for a topological structure or the number of output units. They develop the map periodically. Some models, e.g. the GCS, GNG and IGG also contain a unit-pruning or connection-trimming function which is based on a pre-defined constant threshold, to further tune the structures. Hierarchical models, such as the TreeGCS [10], Multilayered Self-Organising Feature Maps (M-SOM) [11] and Growing Hierarchical Self-Organizing

Map (GHSOM) [12], offer a detailed view for a complicated clustering task. Non-stationary distribution learning models, such as the Growing Neural Gas with Utility criterion (GNG-U) [13], Plastic Self Organising Map (PSOM) [14] and Grow When Required (GWR) [15] are focused on the ability of continuous learning under a dynamic environment.

We focus on the models with a continuous learning or hierarchical training function. For a model to offer automatic hierarchical clustering, it needs a function to further prune the map by removing unsuitable units to form several partitions on a map. Hodge and Austin [10] use this technique to form synonym clusters as an automatic thesaurus. However, the unit-pruning function seriously depends on a pre-defined constant threshold. Based on the unknown data distribution, this threshold is very difficult to determine. Second, the partition is formed because of the nature of the input data. We cannot foresee that a hierarchy must be built by a competitive model with the unit-pruning function. A proper policy may build such a hierarchy by further developing a whole map from a unit with many input data mapped to this unit or with higher error information, e.g. [11] and [12].

For the non-stationary data set, a trained unit or training unit should be replaced by a unit which is trained with new input samples. A model with the unit-pruning function or with the connection-trimming function which should be based on the global consideration can handle this task. That is, a model, e.g. the GNG and GWR, using the connection-trimming function based on a local aged consideration can be treated as an incomplete non-stationary model only. On the other hand, the stop criterion of models should not be a time-dependent threshold, such as iteration or epoch. However, this stop criterion is used for all models in our survey. Moreover, if a model does not use a time-dependent stop criterion, an unsuitable constant unit-pruning or a connection-trimming threshold may make the model train forever but learn nothing. This constant value can be very small or very large, which is totally dependent on trial-and-error. However, it is not a good idea to use such a constant threshold for a big data set. We argue that a unit-pruning or a connection-trimming threshold should be automatically adjusted to suit different data sets during training.

3 Dynamic Adaptive Self-organising Hybrid (DASH) Model

3.1 The Features of DASH

DASH is a growing self-organising model which contains a dynamic structure, hierarchical training, non-stationary data learning and parameter self-adaptation. All features are data-oriented. We need to define three main parameters, which have an impact on the style of the DASH architecture. The first one is τ, which influences how well DASH represents the current data set. The second one is S_{min}, a minimum number of input samples which a map represents. The third one is a connection-trimming threshold, β. These three parameters are percentage-like. S_{min} can also be a real number. For example, a child map is not built when a unit is associated with

the input vectors whose number is less than S_{min}. β is a self-adjusting variable when the units do not grow continuously to meet the requirement of a map quality, the AQE, which is defined as the average distance between every input vector and its Best Matching Unit (BMU) [16].

DASH starts with two units and stops when all units represent their associated input vectors well, or the number of inputs is too few to build a sub-map. The recursive training continues for the individual unit whose AQE does not meet the requirement of DASH. We use the Competitive Hebbian Learning (CHL) principle to connect the BMU and the Second Matching Unit (SMU) for an input stimulus [17]. A connection is trimmed if it is relatively old compared to other connections and a unit without any connection is removed. However, if the model has met its quality requirement, the connection-trimming function is restrained. This is not a problem for training non-stationary data sets because the stop criterion is based on the map quality. Thus, τ affects the size of a single map, S_{min} influences the depth of a hierarchy, and β controls the separation criterion of the clusters. An example of the DASH structure is given in (Figure 1a).

DASH also offers a cue to decide the number of clusters, which is usually determined by subjective human judgement in other competitive learning models. Because of the features of structure separation and hierarchical training, DASH treats a single training map as a two-level hierarchy, i.e. one for a whole sub-map and the other for the partitions in a sub-map. For example, a map in layer 1 contains four partitions (Figure 1a). This map is treated as a local root that has four branches and each branch has a different number of stems. Finally, DASH offers the potential for shorter training. A hierarchical training function can be seen as a distributed model, which trains a whole input set by training several smaller input sub-sets separately.

3.2 The DASH Algorithm

DASH consists of two main iterations and seven processes (Figure 1b). The inner iteration is a learning procedure for each map in a hierarchy. The outer iteration offers a stop criterion for the whole model. In terms of the concept, DASH is a combination of GNG and GHSOM but with several unique features mentioned in the previous section. For convenience, we describe the main structure of the model as follows:

Let $A=\{L_1,L_2,\ldots L_l\}$, where A is the set of sub-maps. Let $L=\{U_1,U_2,\ldots U_u\}$, where U_i is the unit i in the map L. Each U_i has an error variable, err_i. Let C_{ij} be the binary connection between U_i and U_j. Each C_{ij} has a variable, age_{ij}, to store the connection age. Let the input distribution be $p(X)$ for the input set X. Let $X=\{x_1,x_2,\ldots x_n\}$, where x_i is the input sample i in the input set X. We define the weight vectors for an input sample and for a unit as x_i and w_i respectively. Then the precise processes of the DASH algorithm are as below.

1) Global network initialisation

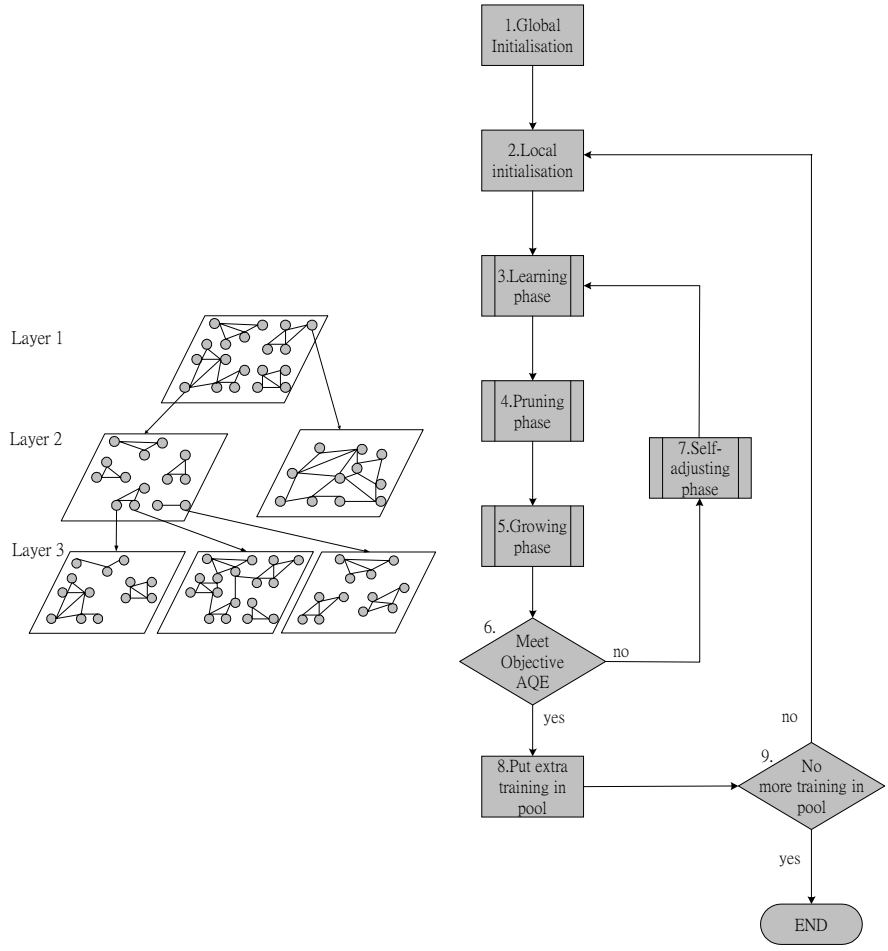


Figure 1. a) The hierarchical structure of DASH b) The flow chart of DASH algorithm

1.1) Define a map quality index, τ, where $0 < \tau \leq 1$. τ decides the objective AQE for a child map. It controls the extent of the size for a single map and is also the stop criterion for the child map training. A smaller τ builds a bigger map. We suppose that before training there is a virtual map L_0 above the first map L_1. L_0 contains only one unit whose weight vector, w_0, is the mean value of the untrained input data set, X, which contains N input samples.

$$w_0 = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{(Eq. 1)}$$

Thus, the AQE of L_0 is:

$$AQE_0 = \frac{1}{N}\sum_{i=1}^{N} \left\| x_i - w_0 \right\| \qquad \text{(Eq. 2)}$$

1.2) Define learning rates α_b and α_s for the Best Matching Unit *b*, and its neighbours *s*, respectively, where $0 < \alpha_s < \alpha_b < 1$.

1.3) Define an age threshold, β, for a connection C_{ij}, where $0 < \beta \leq 1$. The β cooperates with the current highest age of connection to decide whether a connection is too old. A β adjusting parameter, J_β, is defined as well, which is used to modulate β based on the current data samples.

1.4) Define S_{min}, a minimum number of input samples which a map represents. The default value is two because the minimum number of units is two in a map. S_{min} can also be set as a proportion of the size of input data, where $0 < S_{min} < 1$. In this case, S_{min} will be found by $S_{min} \times N$. S_{min} influences the depth of a DASH hierarchy. A smaller S_{min} makes DASH expand deeper down a hierarchy.

1.5) Let O_l be a temporal maximum number of units in a map for the layer *l*. It is defined by Equation 3. We use 3 as the minimum of O_l because a sub-map of DASH starts with 2 units, which allows the model with one spare unit to grow. We apply a constant, 100, in (Eq. 3) for two reasons. The first is that the model is better if it can achieve the quality requirement using a smaller map. We force the model to train properly rather than adding units to pursue a smaller AQE. The second reason is that a very large map is not preferred because it is hard to analyse or visualise. Besides the parameter, O_l, we also define a γ adjusting parameter, J_γ, to modify O_l, where $0 < J_\gamma \leq 1$. O_l will be modified in the self-adjusting phase, if a map contains O_l units but does not meet the map quality.

$$\begin{cases} O_l = \max(3, \min(100, \frac{S_{\min}}{2})), \textit{where } l = 1 \\ O_l = \max(3, \min(O_{l-1} \times \frac{\tau}{2}, \frac{S_{\min}}{2})), \textit{where } l > 1 \end{cases} \quad \text{(Eq. 3)}$$

2) Local network initialisation

2.1) Determine an objective AQE based on the AQE in the direct parent map.

$$\textit{objective } AQE_l = AQE_{l-1} \times \tau \quad \text{(Eq. 4)}$$

2.2) Based on O_l, define how often the unit grows as follows:

$$\textit{IterGrow} = \frac{N_l}{O_l - 1}, \text{ where } N_l \text{ is the number of current input data.} \quad \text{(Eq. 5)}$$

2.3) Create two units and initialise weights randomly from *p(X)*.

2.4) Re-order the current data set randomly.

3) Learning Phase

3.1) Generate a data sample x_i for the model.

3.2) Calculate the Euclidean distance of each unit to x_i and decide the Best Matching Unit, *b*, and the Second Best Matching Unit, *s*, by

$$b = \arg\min_{n \in L} \|x_i - w_n\| \text{ and} \quad \text{(Eq. 6)}$$

$$s = \arg\min_{n \in L/\{b\}} \|x_i - w_n\| \quad \text{(Eq. 7)}$$

and connect them as C_{bs}, if it does not exist.

3.3) Update the weights to the BMU b, and other units n, with a connection from b:

$$w_b(t+1) = w_b(t) + \alpha_b \cdot (x_i - w_n) \text{ and} \quad \text{(Eq. 8)}$$

$$w_n(t+1) = w_n(t) + \alpha_s \cdot (x_i - w_n) \quad \text{(Eq. 9)}$$

3.4) Add 1 to the age variables for all connection C, but zero to C_{bs}.

$$\begin{cases} age = age + 1 \\ age_{bs} = 0 \end{cases} \quad \text{(Eq. 10)}$$

3.5) Increase the error to the BMU error variable, err_b:

$$err_b(t+1) = err_b(t) + \|x_i - w_b\| \quad \text{(Eq. 11)}$$

4) Pruning Phaseat: each n *IterGrow* iteration, where n is ≥ 1.

4.1) Find the maximum age of connections currently.

age_{max}=arg max(age) (Eq. 12)

4.2) Remove any connection whose age is larger than a portion of the maximum age of connections currently. This will be carried out if the number of units is more than two.

$$\begin{cases} remove\ connection\ C_{ij}, & if\ \ current\ number\ of\ \ units > 2\ and\ \dfrac{age_{ij}}{age_{\max}} > \beta \\ do\ nothing, & otherwise \end{cases} \quad \text{(Eq. 13)}$$

4.3) Prune any unit without any connection but still keep the minimum number of units, 2.

5) Growing Phase: at each *IterGrow* iteration, insert a new unit as follows:

5.1) Find the unit q with maximum accumulated error:

$$q = \arg\max_{u \in L}(err) \quad \text{(Eq. 14)}$$

5.2) Find the unit f, the unit with the highest accumulated error amongst the neighbours of q.

$$f = \arg\max_{u \in Neighbours_q}(err) \quad \text{(Eq. 15)}$$

5.3) Insert a new unit r to a map and initialise its weight by interpolating weight vectors q and f.

$$w_r = \frac{w_q + w_f}{2} \quad \text{(Eq. 16)}$$

5.4) Set up the err variables for units q, f and r.

$$err_q = \frac{err_q}{2} \quad \text{(Eq. 17)}$$

$$err_f = \frac{err_f}{2} \quad \text{(Eq. 18)}$$

$$err_r = \frac{err_q + err_f}{2} \quad \text{(Eq. 19)}$$

5.5) Connect unit *r* to unit *q* and *f*. Set up the age variable for these two connections, i.e. C_{rq} and C_{rf}.

$$age_{rq} = age_{rf} = age_{qf} \quad \text{(Eq. 20)}$$

5.6) Remove the connection between unit *q* and unit *f*.

6) Check the condition whether the map AQE meets the objective AQE at each *IterGrow* iteration.

6.1) Evaluate the AQE for a map *l*:

$$AQE_l = \frac{1}{N_l}\sum_{i=1}^{N_l}\left\|x_i - w_{b(i)}\right\|, \quad \text{(Eq. 21)}$$

where $w_{b(i)}$ is the weight vector of BMU for input sample *i*.

6.2) If $AQE_l \le objective\ AQE_l$, then stop training for this map, or go to the self-adjusting phase.

7) Self-adjusting Phase: at some *IterGrow* iteration, modify some parameters to suit the object AQE_l.

7.1) Increase the age threshold, β by the adjusting parameter, J_β, if units are not growing.

$$\beta(t+1) = \beta(t) \times (2 - J_\beta), \text{where } 0.5 < J_\beta \le 1. \quad \text{(Eq. 22)}$$

7.2) Decrease the age threshold, if the number of units reach O_l, which is the reference number of units in a map:

$$\beta(t+1) = \beta(t) \times J_\beta, \quad \text{(Eq. 23)}$$

7.3) Increase the reference number of units in a map, if the number is reached.

$$O_l(t+1) = O_l(t) \times (2 - J_\gamma), \text{where } 0.5 < J_\gamma < 1. \quad \text{(Eq. 24)}$$

8) Put all units whose AQEs are greater than the objective AQE in the same layer into the training pool if the number of their associated input vectors is greater than S_{min}.

9) Continue the hierarchical training until there are no training requirements in the training pool.

4 Experiments and Comparisons

4.1 The New Reuters Corpus

We work with the new version of the Reuters corpus, RCV1, since this news corpus is a representative test for text classification, a common benchmark and a fairly recent comprehensive data source [18]. In this paper, we initially concentrate on the

8 most dominant topics and the first 10,000 full-text articles associated with these topics for our data set. Because a news article can be pre-classified as more than one topic, we consider the multi-topic as a new combination of topics in our task. Thus the 8 chosen topics are expanded into 40 combined topics for the first 10,000 news articles. We use a traditional vector space model such as TFxIDF to represent a full-text document as a numeric vector [19]. Some feature selection techniques are necessary because of the huge dimensionality of vectors. We remove the stop words, confine words shown in WordNet [20], which only contains open-class words, i.e. nouns, verbs, adjectives and adverbs and lemmatise each word to its base form. After this pre-processing, we still have 16,122 different words in the master word list. We further pick up the 1,000 most frequent words from the master word list since this method is as good as some dimensionality reduction techniques [21]. We evaluate our model by the AQE and classification accuracy, which have also been used in the work of Kohonen et al. [22]. The AQE tests the distortion of the representation for the model. Classification accuracy is used since it tests the ability of the model to simulate human categorisation when tackling pre-labelled data sets.

4.2 Static Data Set

We use 10,000 full-text documents as our test-bed. We use a normalized TFxIDF as the vector representation approach [23]. In this experiment, three different τ, i.e. 95%, 90% and 85% of DASH are applied. For convenience, they are termed DASH95, DASH90 and DASH85 in this paper. Different τ means different objective AQEs for DASH, which affects the size of maps. We compare the results with five other models, i.e. SOM, NG, GG, GCS and GNG. We use 15x15 = 225 units for each model, but this number is only an estimate for dynamic models, i.e. the GG, GCS and GNG. All learning rates of models are initialised to 0.1. Such a learning rate is decayed in some models, such as SOM and NG. SOM fine-tuning training starts with a 0.001 learning rate. Other models, such as the GG, GCS, GNG and DASH, also use an extra learning rate which is 0.001 for training runner-up units of BMU. Except DASH, all models stop at 50,000 iterations. The DASH stop criterion is defined by objective AQE. According to the results in (Table 1), the classification accuracy of GG is the lowest. Other models have similar results.

Another evaluation criterion used in this paper is the AQE, which can be used to describe the degree of distortion for models. The smaller AQE, the more cohesive the cluster. We notice that the stricter structure of models, e.g. SOM and GG have higher AQEs. That is, the performance of vector quantization is a significant feature for clustering. The pre-defined structure for models, e.g. SOM and GG, is not the same as the real structure of the input data set.

Other parameters for three DASH models are the same. However, the β variable is self-adjusting based on the input data set. DASH85 contains only one map whose units have all reached the stop criterion. Compared with other DASH models, DASH85 applies a bigger map with 239 units. The β parameter is adapted from 95% to 17.6%. Thus, this is a flat DASH. The number of units in the first map for the DASH95 and DASH90 are 58 and 124, respectively. Several units in the first map of the DASH95 and DASH90 do not meet the stop criterion. Therefore, they develop the recursive training until the stop criterion is met. In our experiments,

DASH95 and DASH90 contain 30 and 14 sub-maps, respectively. The performance of the DASH is comparable to other models and offers an extra hierarchical structure based on the static data set.

	SOM	NG	GG	GCS	GNG	DASH95	DASH90	DASH85
AQE	0.930	0.837	0.881	0.820	0.823	0.818	0.790	0.802
Accuracy	69.16%	69.54%	69.82%	68.18%	68.60%	66.60%	70.42%	68.37%

Table 1. A comparison of several competitive methods based on the criteria of classification accuracy and AQE for 10,000 full-text news articles. Parameters for DASH are: S_{min}: 1% and β: 95%. Please note that β is an adjustable parameter during training.

4.3 Knowledge Acquisition

In this section, we keep the same pre-processing and vector representation approaches as that in the previous section but we want to test the ability of models to handle the non-stationary data set. We use three experiments in this section. We treat the new data set as new knowledge that complements the existing knowledge. The first 5,000 full-text documents are applied as an existing data set and the second 5,000 full-text documents as a new data set. The existing data set is used for all experiments in the beginning. The new data set is introduced in experiment 1 at iteration 10,000, experiment 2 at iteration 30,000 and experiment 3 at iteration 50,000. SOM rough-training is stopped at iteration 30,000 and its fine-tuning training is stopped at 50,000 iterations. The stop criterion of DASH is finding the objective AQE. According to our experiments, the SOM does not suffer from the new data set seriously, if the distribution of the new data set is similar to that of the existing data set or the new data set comes from the same collection of the existing data set (Table 2). In this case, the performance of the SOM is comparable to the non-stationary model, DASH.

		10,000 iterations	30,000 iterations	50,000 iterations
SOM	AQE	0.937	0.938	0.940
	Accuracy	69.06%	68.44%	69.08%
DASH90	AQE	0.771	0.780	0.802
	Accuracy	72.23%	71.42%	70.39%

Table 2. A comparison of SOM and DASH. A new data set is added to the existing data set at iteration 10,000 in experiment 1, iteration 30,000 in experiment 2 and iteration 50,000 in experiment 3. Parameters for DASH are: τ: 90%, S_{min}: 1% and β: 95%. Please note that β is an adjustable parameter during training.

4.4 Knowledge Update

In this section, we want to compare the performance of models under a non-stationary environment as well, but we treat the existing knowledge as out-of-date, which should be updated by the new knowledge, i.e. the new data set. We use the same pre-processing procedure mentioned above and use 10,000 full-text documents as our test-bed. To mimic the non-stationary data set, we use the normalized TFxIDF vector representation as our new data set but use the non-normalized TFxIDF as the existing data set. The averaged weights of the existing set are much

higher than those of the new data set. Thus, the AQEs are also much higher when models deal with the existing data set. Like the strategy mentioned in the previous section, we introduce the new data set at iteration 10,000 for experiment 1, iteration 30,000 for experiment 2 and iteration 50,000 for experiment 3. According to our experiments, the SOM clearly suffers from the decayed learning rate (Table 3). The new data samples are not learnt completely, so the accuracy drops while the AQE increases. On the other hand, DASH removes all unsuitable trained units very fast and adjusts its new objective AQE automatically. Thus, there is no big difference between the performance at each point when a new data set is introduced during training for DASH.

		10,000 iterations	30,000 iterations	50,000 iterations
SOM	AQE	0.948	1.352	2.513
	Accuracy	64.37%	21.52%	25.75%
DASH90	AQE	0.784	0.776	0.793
	Accuracy	71.40%	72.25%	69.30%

Table 3. A comparison of SOM and DASH. A new data set substitutes for the existing data set at iteration 10,000 in experiment 1, iteration 30,000 in experiment 2 and iteration 50,000 in experiment 3. Parameters for DASH are: τ: 90%, S_{min}: 1% and β: 95%. Please note that β is an adjustable parameter during training.

4.5 An Analysis of the Non-Stationary and Hierarchical DASH

We use the second experiment in the previous section, which updates the existing data set at iteration 30,000 to further the analysis of our model. Please note that a 15×15 SOM is used in this experiment. The DASH starts with 2 units and adjusts its architecture and parameters based on the current data set. The DASH satisfies the objective AQE by developing 23 sub-maps. A part of the hierarchical structure is shown in (Figure 2). The concepts of units in a neighbourhood are similar. We use two terms whose weights are the most significant to represent the labels of the unit in the root map and use the second and third significant terms to represent its child map. Thus, a unit of the map in the lower layer of a hierarchy is associated with more terms, which represent news articles with more specific concepts.

In the beginning of the training stage, the SOM has a smaller AQE because the number of units is much larger than that of the DASH (Figure 3a). However, the AQE of the DASH is smaller than that of the SOM from iteration 8,000 (Figure 3). At this point, the DASH only contains 34 units while the SOM has 255 units (Figure 4a). When the existing data set is replaced by the new data set, the AQEs of both models are much smaller. Based on the criterion of the AQE, the DASH outperforms the SOM.

The non-stationary learning feature of the DASH can be illustrated by Figure 4. The DASH adjusts its architecture by the unit-growing and global connection-trimming functions. The number of units for the DASH is continuously growing in general. When the existing data set is replaced by the new data set at iteration 30,000, many unsuitable units are removed (Figure 4a). This is performed by the connection-trimming variable, β. We set an initial value of 95% for β. It is adjusted automatically based on the current data set. The final value of β is about 0.4 (Figure 4b). Some non-stationary competitive models such as GNG-U and GWR

have been tried in these experiments. However, it is not possible to use their unit-pruning and connection-trimming constant thresholds for both data sets. When a proper threshold is set for the existing data set, this threshold is always too large for the new data set. Thus, models do not grow. Conversely, if a threshold is suitable for the new data set, this threshold is always too small for the existing data set. However, we should not set such a threshold by presuming the distribution of the new data set. Therefore, we only present our model and SOM in these experiments.

The Root Map of the DASH

yen ml	yen specify		yen billion	billion crown	crown billion	profit crown	profit earnings	profit earnings		net loss	net share
				franc billion					share dividend	share net	
				franc swiss		yuan million	yuan million		rupee share		share rupee
beat game		sept latest	budget sept			yuan million	percent mark		markka singapore		share rupee
			officer chief		tax hotel	hotel percent	percent hotel	pct percent			peso rate
play match		drug company		software company		gold hotel		pct pc	pct pc		
		drug study		tobacco company	tobacco company		cattle source		pct bank		rate bank
england test				airline tobacco	tobacco power					bank rate	bank rate
iraq mother	nuclear treaty	china nuclear		airline european		import oil		coffee crude			bond bank
iraq king		china taiwan	china court		port cargo		oil tonne		cent coffee		bond coupon
			court minister				tonne gas		tonne wheat		bond coupon
police refugee	police rebel	minister russian	minister election	clinton minister		gas british	tonne rice	tonne wheat	wheat tonne		trader price

Oil and Tonne

tonne price	tonne crude	import tonne		chinese petroleum	chinese petroleum	company chinese
		win million				company percent
gas output	gas tonne	gas tonne	gas tonne	gas crude		company tender
		gas field		company field		
field crude			field petroleum		company iran	iran company
oil production	field production	field barrel	field barrel	barrel field	venture energy	energy venture

Figure 2. A part of the hierarchical structure of a DASH. The two most significant terms are the shown in the upper map and the second and third most significant terms are show in its child map.

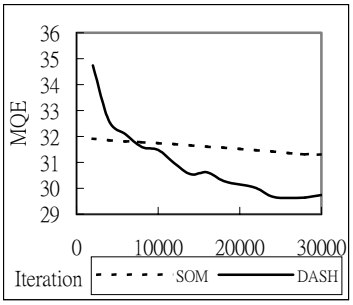
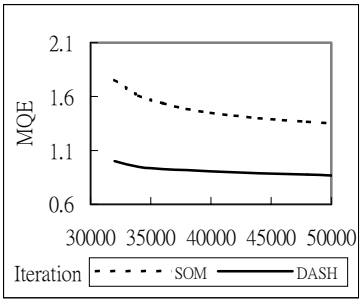


Figure 3. a) AQEs of SOM and DASH for the existing data set



b) AQEs of SOM and DASH for the new data set

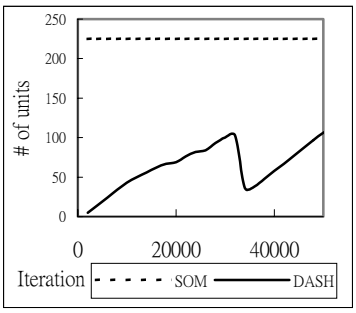
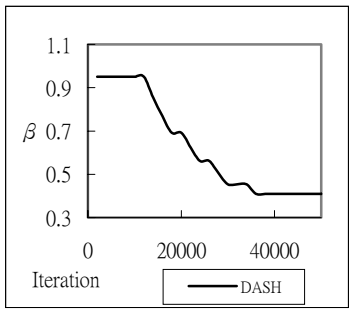


Figure 4. a) Units of SOM and DASH



b) The β parameter of DASH

5 Conclusion

In this paper, we have presented a new type of self-organising dynamic growing neural network which can deal with non-stationary data sets and represent the inner data structure by a hierarchical view. In terms of the concept, DASH is a hybrid model of GHSOM and GNG. It contains several unique features, such as parameter self-adjustment, hierarchical training and continuous learning. Based on these features, a real-world document clustering task has been demonstrated in this paper. We also analyse the deficiencies of current models. Those models which are designed for the non-stationary data sets may not be suitable for clustering a real-world task. The main reason is the difficulty of determining constant unit-pruning and connection-trimming parameters. Furthermore, those non-stationary models should not use a time-dependent stop criterion. For more complex data sets, such as a document collection, a hierarchical structure is preferable. This hierarchical training also benefits from a distributed model which trains several small maps separately instead of a huge map. That is, the DASH is a new hierarchical neural model which functions as a non-stationary distribution learning facility.

References

1. Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. Newsgroup exploration with WEBSOM method and browsing interface. Report A32, Helsinki University of Technology, 1996
2. Kohonen, T. Self-organization and associative memory. Springer-Verlag, Berlin, 1984

3. Grossberg, S. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. Biological Cybernetics, 1976, 23:121-131
4. Martinetz, T. and Schulten, K. A 'Neural-Gas' network learns topologies. Artificial Neural Network, 1991, 1:397-402
5. Fritzke, B. Growing grid-a self-organizing network with constant neighborhood range and adaptation strength. Neural Processing Letters, 1995, 2(5):9-13
6. Fritzke, B. Growing cell structures – a self-organizing network for unsupervised and supervised learning. Neural Networks, 1994, 7(9):1441-1460
7. Fritzke, B. A growing neural gas network learns topologies. Advances in Neural Information Processing Systems 7, Tesauro, G., Touretzky, D.S. and Leen, T.K. (Eds) , MIT Press, Cambridge MA, 1995: 625-632
8. Blackmore, J. and Miikkulainen, R. Incremental grid growing: encoding high-dimensional structure into a two-dimensional feature map. Proceedings of the IEEE International Conference on Neural Networks (ICNN'93), 1993
9. Alahakoon, D., Halgamuge, S.K., and Srinivasan, B. Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Tractions on Neural Networks, 2000, 11(3):601-614
10. Hodge, V. and Austin, J. Hierarchical growing cell structures: TreeGCS. Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems, 2000
11. Chen, H., Schuffels, C. and Orwig, R. Internet categorization and search: a self-organizing approach. Journal of Visual Communication and Image Representation, 1996, 7(1):88-102
12. Rauber, A., Merkl, D. and Dittenbach, M. The growing hierarchical self-organizing maps: exploratory analysis of high-dimensional data. IEEE Transactions on Neural Networks, 2002, 13(6):1331-1341
13. Fritzke, B. A self-organizing network that can follow non-stationary distributions. Proceedings of ICANN-97, International Conference on Artificial Neural Networks, Springer, 1997:613-618
14. Lang, R. and Warwick, K. The plastic self organising map. IEEE World Congress on Computational Intelligence, 2002
15. Marsland, S., Shapiro, J. and Nehmzow, U. A self-organising network that grows when required. Neural Networks, 2002, 15:1041-1058
16. Kohonen, T. Self-organizing maps. Springer-Verlag, 2001
17. Martinetz, T.M. Competitive Hebbian learning rule forms perfectly topology preserving maps. International Conference on Artificial Neural Networks, ICANN'93, Amsterdam, 1993:427-434
18. Wermter, S. and Hung, C. Selforganizing Classification on the Reuters News Corpus. COLING2002, 19th International Conference on Computational Linguistics, Taipei, Taiwan, 2002:1086-1092
19. Salton, G. Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, USA, 1989
20. Miller, G.A. WordNet: a dictionary browser. Proceedings of the First International Conference on Information in Data, 1985
21. Chakrabarti, S. Data mining for hypertext: a tutorial survey. ACM SIGKDD Explorations, 2000, 1(2):1-11
22. Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V. and Saarela, A. Self organization of a massive document collection. IEEE Transactions on Neural Networks, 2000, 11(3):574-585
23. Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. Information Processing & Management, 1988, 24(5):513-523