

# Visualizing Changes in Data Collections Using Growing Self-Organizing Maps

Andreas Nürnberger and Marcin Detyniecki  
University of California at Berkeley  
EECS, Computer Science Division  
Berkeley, CA 94720, USA  
E-mail: {anuernb,marcin}@eecs.berkeley.edu

**Abstract - In previous work we have proposed a modification of the standard learning algorithm for self-organizing maps that iteratively increases the size of the map during the learning process by adding single neurons. The main advantage of this approach is an automatic control of the size and topology of the map, thus avoiding the problem of misclassification because of an imposed size. In this paper we discuss how this algorithm can be used to visualize changes in data collections. We illustrate our approach with some examples.**

## I. INTRODUCTION

Self-organizing maps (SOMs) are a special architecture of neural networks that cluster high-dimensional data vectors according to a similarity measure [11]. The clusters are arranged in a low-dimensional topology – usually a grid structure – that preserves the neighborhood relations in the high dimensional data. Thus, not only objects that are assigned to one cluster are similar to each other as in every cluster analysis, but also objects of nearby clusters are expected to be more similar than objects in more distant clusters. Self organization maps are frequently used to cluster, visualize and interpret large high-dimensional data sets of process data, image or document features as well as commercial or financial data [2].

Unfortunately, the standard learning model of self-organizing maps requires a predefined map structure. The complete learning process has to be repeated if the size of the map was to small and very dissimilar vectors are assigned to the same unit or to large and similar vectors spread out on the map. Meanwhile some methods have been proposed to compute the size of the map automatically by a growing process (see, e.g. [1, 5, 15]).

As noted above self-organizing maps are an interesting tool for visualization. But this feature is – in the standard model – regrettably static. On the one hand changes in data collections can not be observed and on the other hand if new data is added, the map is usually trained from scratch losing the initial topology. Thus, the potential user will not retrieve a similar distribution of the data in the new map. This is especially desirable in applications in document retrieval, where a user might want to use the mapping as starting point for searching.

In this paper we discuss how the growing SOM approach proposed in [15] can be used to visualize changes in document collections. The model proposed in [15] was especially developed for the analysis of text databases. However, this model can also be applied to arbitrary collections of documents that can be described by a numerical feature vector [16].

In the following we first discuss briefly the theoretical aspects of our approach. To make us clear, we will start by explaining some aspects of text classification and retrieval. Secondly we illustrate with some examples the viability of our model for the visualization of changes in document collections.

### A. Analyzing text document collections

For searching and navigating in large document collections (except of e.g. simple keyword searches) it is necessary to pre-process the documents and store the information in a data structure, which is more appropriate for further processing than an unstructured text file. The currently predominant approaches are the vector space model [22], the probabilistic model [19], the logical model [18] and the Bayesian net model [24]. Despite of its simple data structure without using any explicit semantic information, the vector space model enables very efficient analysis of huge document collections and is therefore still used in most of the currently available document retrieval systems. The most popular retrieval methods that are based on this model are Latent Semantic Indexing (LSI) [4], Random Projection [9] and Independent Component Analysis (ICA) [8]. The vector space description is also used for self-organizing maps. In the following we briefly describe the vector space model and corresponding document encoding techniques.

### B. The Vector Space Model

The vector space model represents documents as vectors in a  $t$ -dimensional space, i.e. each document  $i$  is described by a numerical feature vector  $D_i = \{x_1, \dots, x_t\}$ . Thus, documents can be compared by use of simple vector operations. For instance queries can be encoded in a query vector of the same

---

\* **Acknowledgements.** The work presented in this article was partially supported by BTextact Technologies, Adastral Park, Martlesham, UK.

space. The query vector can then be compared to each document and a result list can be obtained by ordering the documents according to the computed similarity [20].

The main problem of the vector space representation of documents is to find an appropriate encoding of the feature vector. Each element of the vector usually represents a word (or a group of words) of the document collection, i.e. the size of the vector is defined by the number of words (or groups of words) of the complete document collection.

The simplest way of document encoding is to use binary term vectors, i.e. a vector element is set to one if the corresponding word is used in the document and to zero if the word is not. This encoding will result in a simple Boolean search if a query is encoded in a vector. Using Boolean encoding the importance of all terms for a specific query or comparison is considered as similar.

To improve the performance usually term weighting schemes are used, where the weights reflect the importance of a word in a specific document of the considered collection. Large weights are assigned to terms that are used frequently in relevant documents but rarely in the whole document collection [21]. Thus a weight  $w_{ik}$  for a term  $k$  in document  $i$  is computed by term frequency  $tf_{ik}$  times inverse document frequency  $idf_k$ , which describes the term specificity within the document collection. In [20] a weighting scheme was proposed that has meanwhile proven its usability in practice. Besides term frequency and inverse document frequency – defined as  $idf_k := \log(N/n_k)$  –, a length normalization factor is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

$$w_{ik} = \frac{tf_{ik} \log(N/n_k)}{\sqrt{\sum_{j=1}^I (tf_{ij})^2 (\log(N/n_j))^2}},$$

where  $N$  is the size of the document collection  $C$  and  $n_k$  the number of documents in  $C$  that contain term  $k$ .

Based on a weighting scheme a document  $i$  is defined by a vector of term weights  $D_i = \{w_{i1}, \dots, w_{ik}\}$  and the similarity  $S$  of two documents (or the similarity of a document and a query vector) can be computed based on the inner product of the vectors, i.e.

$$S(D_i, D_j) = \sum_{k=1}^m w_{ik} \cdot w_{jk}.$$

Thus, this model fulfills a prerequisite for the usability in self-organizing maps. For a more detailed discussion of the vector space model and weighting schemes see, e.g. [3, 6, 21, 22].

### C. Visualization

Visualization of document collections requires methods that are able to group documents based on their similarity and furthermore that visualize the similarity between discovered groups of documents. Clustering approaches that are frequently used to find groups of documents with similar content [23] usually do not consider the neighborhood relations between the obtained cluster centers. Self-organizing maps are

an alternative approach which is frequently used in data analysis to cluster high dimensional data and which considers also the similarities of neighboring clusters [11]. They can be seen as a special architecture of neural networks that cluster high-dimensional data vectors according to a similarity measure. The clusters are arranged in a low-dimensional topology that preserves the neighborhood relations of the corresponding high dimensional data vectors. Thus, not only objects that are assigned to one cluster are similar to each other, but also objects of nearby clusters are expected to be more similar than objects in more distant clusters. Usually, two-dimensional arrangements of squares or hexagons are used for the definition of the neighborhood relations. Although other topologies are possible for self-organizing maps, two-dimensional maps have the advantage of intuitive visualization and thus good exploration possibilities.

In document retrieval, self-organizing maps can be used to arrange documents based on their similarity (e.g. [7, 14]). This approach opens up several appealing navigation possibilities. Most important, the surrounding grid cells of documents known to be interesting can be scanned for further similar documents. Furthermore, the distribution of keyword search results can be visualized by coloring the grid cells of the map with respect to the number of hits. This allows a user to judge e.g. whether the search results are assigned to a small number of (neighboring) grid cells of the map, or whether the search hits are spread widely over the map and thus the search was – most likely – too unspecific.

## II. THE USED MODEL STRUCTURE

Self-organizing maps are trained in an unsupervised manner (i.e. no class information is provided) using a set of high-dimensional sample vectors. The network structure has two layers (see Figure 1). The neurons in the input layer correspond to the input dimensions. The output layer (map) contains as many neurons as clusters needed. All neurons in the input layer are connected with all neurons in the output layer. The weights of the connection between input and output layer of the neural network encode positions in the high-dimensional data space. Thus, every unit in the output layer represents a prototype.

Before the learning phase of the network, the two-dimensional structure of the output units is fixed and the weights are initialized randomly. During learning, the sample vectors are repeatedly propagated through the network. The weights of the most similar prototype  $w_s$  (*winner neuron*) are modified such that the prototype moves toward the input vector  $w_i$ . As similarity measure usually the Euclidean distance or scalar product is used. The weights  $w_s$  of the winner neuron are modified according to the following equation:

$$\forall i: w'_s = w_s + \delta \cdot (w_i - w_s), \text{ where } \delta \text{ is a learning rate.}$$

To preserve the neighborhood relations, prototypes that are close to the winner neuron in the two-dimensional structure are also moved in the same direction. The strength of the modification decreases with the distance from the winner neu-

ron. Therefore, the adaptation method is extended by a neighborhood function  $v$ :

$$\forall i: w_{s,i} = w_{s,i} + v(c,i) \cdot \delta \cdot (w_{s,i} - x_i)$$

where  $\delta$  is a learning rate. By this learning procedure, the structure in the high-dimensional sample data is non-linearly projected to the lower-dimensional topology. Finally, arbitrary vectors (i.e. vectors from the sample set or prior ‘unknown’ vectors) can be propagated through the trained network and are mapped to the output units. For further details on self-organizing maps see [12].

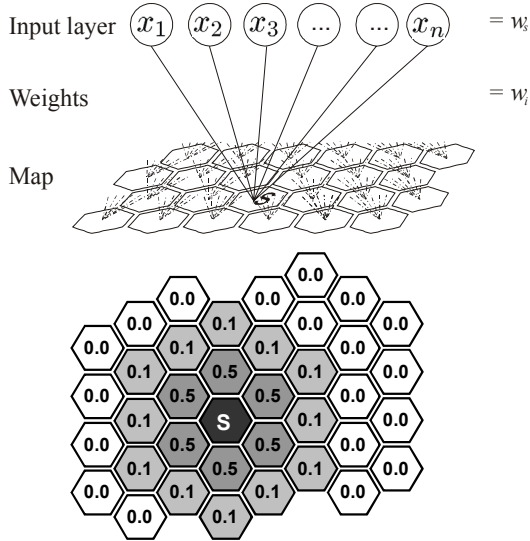


FIGURE 1. STRUCTURE OF A RECTANGULAR SELF-ORGANIZING MAP (TOP) AND POSSIBLE NEIGHBORHOOD FUNCTION FOR A STRUCTURE BASED ON HEXAGONS (BOTTOM).

Unfortunately, the standard model of self-organizing maps requires a manual predefined map structure. Therefore, the size of the map is usually too small or too large to map the underlying data appropriately. If the size of the map was too small (the classification error for every pattern is usually very high and thus very dissimilar vectors are assigned to the same unit) or too large (similar vectors spread out on the map). Therefore, the complete learning process has to be repeated several times until an appropriate size is found.

Growing self-organizing map approaches try to solve this problem by modifying the size (and structure) of the map by adding new units to the map, if the accumulated error on a map unit increases a specified threshold. In the following we briefly describe the approach that we used in the presented application.

#### A. A growing self-organizing map approach

The proposed method is mainly motivated by the growing self-organizing map models presented in [1, 5]. In contrast to these approaches we use hexagonal map structure and restrict

the algorithm to add new units to the external units if the accumulated error of a unit exceeds a specified threshold value.

The algorithm can be described as follows:

1. Predefine the initial grid size (usually  $2 \times 2$  units)
2. Initialize the assigned vectors with randomly selected values. Reset error values  $e_i$  for every unit  $i$ .
3. Train the map using all inputs patterns for a fixed number of iterations.
4. For all input patterns  $i$  increase the error values of their winner unit  $s$  by the current error value for the pattern  $i$ .
5. Identify the unit with the largest accumulated error.
6. If the error does not exceed a threshold value stop training.
7. Identify the external unit  $k$  with the largest accumulated error.
8. Add a new unit to the unit  $k$ . If more than one free link is available select the unit at the nearest position to the neighboring unit which is most dissimilar to  $k$ . Initialize the weights of the new unit with respect to the vectors of the neighboring units so that the new vector is smoothly integrated into the existing vectors (see Figure 2).
9. Continue with step 3.
10. Continue training of the map for a fixed number of iterations. Reduce the learning rate during training.

This process creates an incremental growing map. Furthermore, it allows training the map incrementally by adding new data, since the training algorithm affects mainly the winning units to which new data vectors are assigned. If these units accumulate high errors, which means that the assigned patterns cannot be classified appropriately, this part of the map starts to grow. Even if the considered neuron is an inner neuron, then the additional data pushes the prior assigned patterns to outer areas to which new neurons had been created. This can be interpreted as an increase of the number of data items belonging to a specific area or cluster in data space, or if text documents are assigned to the map as an increased number of publications concerning a specific topic. Therefore also dynamic changes can be visualized by comparing maps, which were incrementally trained by, e.g. newly published documents. In the following section we discuss these characteristics of the algorithms more detailed.

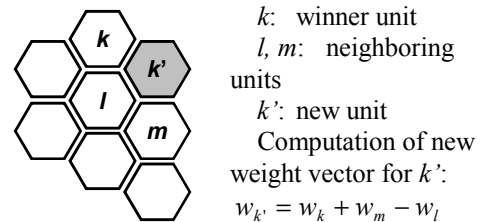


FIGURE 2. EXAMPLE OF A COMPUTATION OF A VECTOR FOR A NEW INSERTED UNIT

### III. VISUALIZING CHANGES

In the following we discuss some examples to clarify the effects of growing and the obtained visualization. For simplicity we use only low dimensional data sets. We trained the self-organizing maps using the growing learning method discussed above. Since the learning algorithm is designed for incrementally increasing data bases – which is the most common case for document collections – we discuss only the addition of new data to an existing document collection.

#### A. Introducing data to an existing cluster

For the following two examples we used a data set consisting of 1000 feature vectors defining 6 randomly generated clusters in 3-dimensional data space. Each data cloud is generated around a center point, placed at a fixed distance to the center of the axes (see Figure 3 A). Using this structure we can easily control the overlapping of the classes. An initial map was trained using this data set. The map is depicted in Figure 3 (D). As label of a cell we selected the class describing the majority of data patterns assigned to this cell.

Let us assume that the data describe before was the initially available data. Let imagine that we receive later new data but mostly for one of the classes we observed. What would be the effect in the growing self-organizing map? To show this, we

added 150 randomly generated data points of class 4 to the training data set (see Figure 3 B) and retrained the map. The resulting map is shown in Figure 3 (E). The number of cells assigned to class 4 increased and the other classes are pushed away from this class. The algorithm not only correctly represents the changes in the data, but also a user might obtain visual information of the changes by comparing the two maps, since the algorithm does not change the initial topology.

#### B. Introducing a new cluster

For this example we used once more the map obtained by use of the initial data set (6 classes; Figure 3 D). But this time we introduced data from a new class and we look at the effects on the map. More precisely, we create a new class of 150 data points at the center in-between of the 6 classes, which were used for training (see Figure 3 F). The data points describing the new class were added to the training data set and the original map was retrained. As expected the learning algorithm placed the new class in the center of the exiting classes and pushed them away by creating new cells at the outside of the map. It has to be noticed that this result is remarkable since in the ‘growing’ algorithm cells are added at the borders of the map. Again a user is able to appropriately interpret the changes in the underlying data by visual comparison of the resulting map and the initial one.

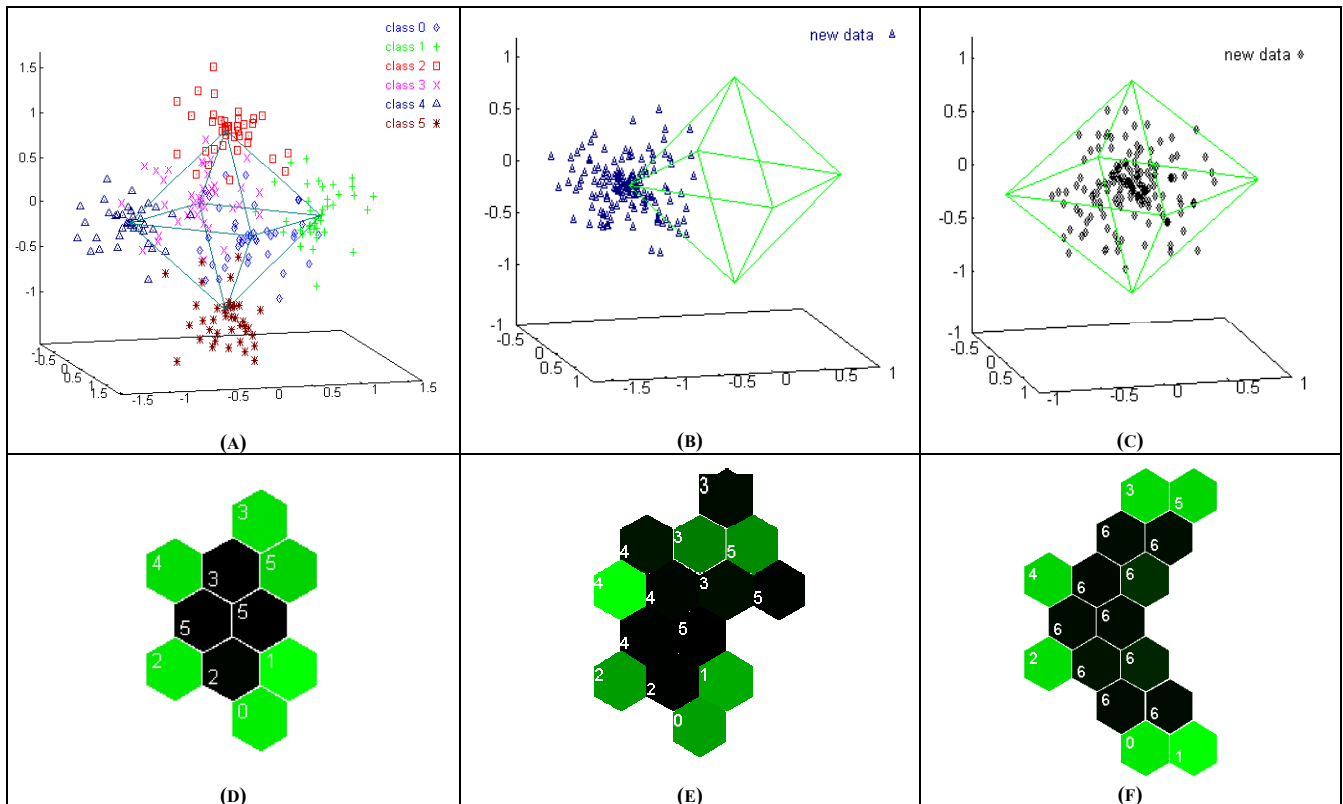


FIGURE 3. RANDOMLY GENERATED TRAINING DATA AND RESULTING MAPS. UPPER ROW: INITIAL DATA USED FOR TRAINING (A), INCREASED NUMBER OF ELEMENTS FOR CLASS 4 (B) AND INTRODUCED CLASS IN THE CENTER OF THE TRAINING DATA CLOUDS (C). LOWER ROW: RESULTING MAPS (NUMBERS DEPICT CALSSES ASSIGNED BY MAJORITY LABELLING; COLORS DEPICT AMOUNT OF DATA ASSIGNED TO A CELL, BRIGHTER COLORS DEPICT MORE DATA)

### C. Visualizing text data (introducing new topics)

The structure of the data when using a vector space model for text documents is slightly different to the two examples before. In fact, the dimension of the space is usually very large (equals the cardinality of the indexing dictionary). Also frequently some keywords appear in one text and not in the others, leaving the data vector with a lot of zeros (sparse vector). In order to test the growing self-organizing map algorithm in this type of application we created the following data. We generated feature vectors that define documents dealing with the topics (keywords) *fuzzy*, *neuro*, *genetic* and five arbitrary keywords. We created two data sets, each consisting of 500 feature vectors. The first data set (A) describes documents that use just one of the keywords *fuzzy*, *neuro* or *genetic*. The five remaining keywords are arbitrarily chosen. To encode the documents in a feature vector to each word a term weight is assigned, i.e. for each selected word  $i$  a random value  $x_i \in [0,1]$  is computed and assigned to the document vector. All other elements are set to zero.

The second data set (B) describes documents that contains exactly two of the keywords *fuzzy*, *neuro* or *genetic*. Thus, these documents are considered to describe a combination of these topics. The distribution of the keywords in the documents is shown in Table 1.

TABLE 1. DISTRIBUTION OF KEYWORDS IN TEXT DATA

	data set A			data set B		
	fuzzy	neuro	genetic	neuro- genetic	fuzzy- genetic	fuzzy- neuro
<b>fuzzy</b>	179	-	-	-	169	161
<b>neuro</b>	-	167	-	170	-	161
<b>genetic</b>	-	-	154	170	169	-
<b>keyword 4</b>	80	79	77	98	86	84
<b>keyword 5</b>	90	80	85	93	77	83
<b>keyword 6</b>	93	84	70	77	82	75
<b>keyword 7</b>	91	78	73	92	89	87
<b>keyword 8</b>	83	66	84	91	88	85

We trained a self-organizing map using the first data set (A). The resulting map is shown in Figure 4 (left). As label of a cell we selected the keyword describing the majority of documents assigned to this cell. (For document collections more refined labeling methods have been proposed that also consider the distribution of documents in neighboring cells, see e.g. [13, 17]. However, since we just want to describe the document distribution, we used the majority labeling method.)

In a second run we added the data set B that contains documents with keyword combinations as described above to the training data set A and retrained the map. The resulting map is shown in Figure 4 (right). The documents using keyword combinations where placed in-between the ‘pure’ classes which where pushed out during the learning process. The underlying structure of the new map still reflects the

structure of the old map and thus a user gets an idea – considering the labels as well as the old map – of what might have changed in the underlying document collection. In Table 2 and Table 3 the confusion matrices using the node label as a class are given to describe the obtained classification performance of this approach. However, it has to be considered that the goal of these experiments was not to evaluate the classification performance, which depends on the error function and threshold used in the learning method and even more on the quality of the feature vectors used to describe the objects as discussed for text documents in Section I.C., but to demonstrate the effectiveness of the growing approach for the visualization of document collections.

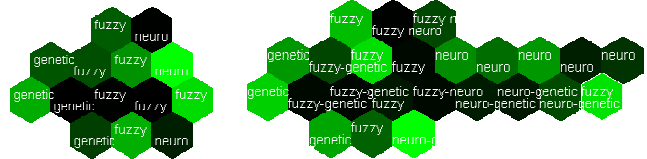


FIGURE 4. LEARNED MAPS FOR TEXT DATA.

TABLE 2. CONFUSION MATRIX FOR TEXT DATA (3 CLASSES)

	fuzzy	genetic	neuro	$\Sigma$
<b>fuzzy</b>	138	40	1	179
<b>genetic</b>	14	139	1	154
<b>neuro</b>	15	40	112	167
$\Sigma$	167	219	114	500

TABLE 3. CONFUSION MATRIX FOR TEXT DATA (6 CLASSES)

	fuzzy	fuzzy-genetic	fuzzy-neuro	genetic	neuro	neuro-genetic	$\Sigma$
<b>fuzzy</b>	138	-	-	40	1	-	179
<b>fuzzy-genetic</b>	60	37	-	58	-	14	169
<b>fuzzy-neuro</b>	50	-	36	2	37	36	161
<b>genetic</b>	14	-	-	139	1	-	154
<b>neuro</b>	15	-	-	40	112	-	167
<b>neuro-genetic</b>	11	-	-	45	62	52	170
$\Sigma$	288	37	36	324	213	102	1000

## IV. A SOFTWARE IMPLEMENTATION

The retrieval and learning methods discussed above have been implemented in a tool for interactive document retrieval. A more detailed description of the implemented methods for searching and navigation can be found in [10] and [15]. The tool combines iterative keyword search with the visualization and navigation capabilities of a self-organizing map to which the documents of the underlying database are assigned.

In the current release we also support retraining of the map using new data. Thus, dynamic changes can be visualized as described above.

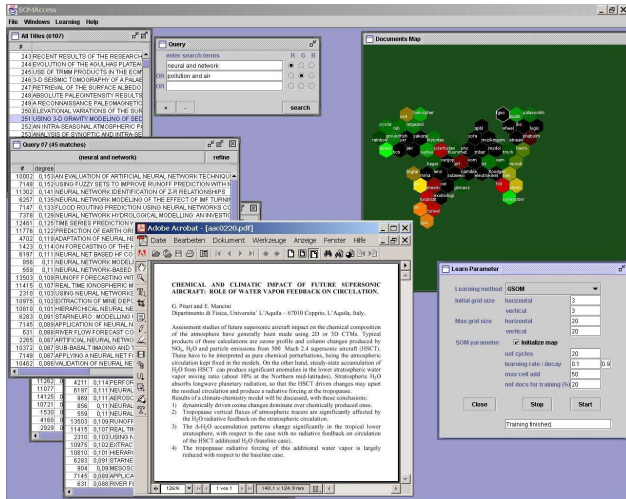


FIGURE 5. A SCREENSHOT OF THE SOFTWARE TOOL: DOCUMENT MAP (TOP RIGHT) COLORED ACCORDING TO THE SEARCH HITS OF A QUERY (TOP CENTER); SEARCH RESULT LISTS (LEFT); RETRIEVED DOCUMENT (CENTER).

## V. CONCLUSION AND FUTURE WORK

As shown above, self-organizing maps can be a useful method for clustering and exploration of document collections. In contrast to the standard model, which usually has to be retrained several times until an appropriate size has been found, growing self-organizing maps are able to adapt their size and structure to the data. As described in this paper, growing self-organizing maps can be used to visualize changes in data collections. If document collections are analyzed, the dynamic information is shown by a growing of the map in specific areas. Thus, this method can provide valuable hints, e.g. concerning upcoming topics or additional subjects that are recently introduced to a document database. Since the basic map data structure is unchanged after training with additional data, a user will still be able to retrieve documents from prior 'known' areas of the map.

A further advantage of the presented approach is that it does not require manually defined lists of index terms or a classification hierarchy as other document retrieval models. These usually require expensive maintenance. Especially in rapidly changing document collections – like collections of publications of scientific research – classification systems that are not frequently updated are usually not accepted by the users, who are naturally highly interested in current and upcoming topics.

Applied to text documents especially the combination of interactive keyword search and self-organizing maps in an interactive environment provides a valuable tool for document retrieval purposes.

## VI. REFERENCES

[1] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery, *IEEE Transactions on Neural Networks*, 11(3), pp. 601-614,2000.

[2] N. Allinson, H. Yin, L. Allinson, and J. Slack (eds.), *Advances in Self-Organizing Maps*, Proc. of the third Workshop on Self-Organizing Maps (WSOM 2001), Springer,2001.

[3] R. Baeza-Yates, and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley Longman,1999.

[4] S. Deerwester, S. T. Dumais, G. W. Furnas, and T. K. Landauer, Indexing by latent semantic analysis, *Journal of the American Society for Information Sciences*, 41, pp. 391-407,1990.

[5] B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, *Neural Networks*, 7(9), pp. 1441-1460,1994.

[6] W. R. Greiff, A Theory of Term Weighting Based on Exploratory Data Analysis, In: *21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, New York, NY,1998.

[7] T. Honkela, Self-Organizing Maps in Natural Language Processing, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland,1997.

[8] C. L. Isbell, and P. Viola, Restructuring sparse high dimensional data for effective retrieval, In: *Proc. of the Conference on Neural Information Processing (NIPS'98)*, pp. 480-486,1998.

[9] S. Kaski, Dimensionality reduction by random mapping: Fast similarity computation for clustering, In: *Proc. Of the International Joint Conference on Artificial Neural Networks (IJCNN'98)*, pp. 413-418, IEEE,1998.

[10] A. Klose, A. Nürnberger, R. Kruse, G. K. Hartmann, and M. Richards, Interactive Text Retrieval Based on Document Similarities, *Physics and Chemistry of the Earth, Part A*, 25(8), pp. 649-654, 2000.

[11] T. Kohonen, Self-Organized Formation of Topologically Correct Feature Maps, *Biological Cybernetics*, 43, pp. 59-69,1982.

[12] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin,1984.

[13] K. Lagus, and S. Kaski, Keyword selection method for characterizing text document maps, In: *Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks*, pp. 371-376, IEEE,1999.

[14] X. Lin, G. Marchionini, and D. Soergel, A selforganizing semantic map for information retrieval, In: *Proceedings of the 14th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pp. 262-269, ACM Press, New York,1991.

[15] A. Nürnberger, Interactive Text Retrieval Supported by Growing Self-Organizing Maps, In: T. Ojala (ed.), *Proc. of the International Workshop on Information Retrieval (IR'2001)*, pp. 61-70, Infotech, Oulu, Finland,2001.

[16] A. Nürnberger, and A. Klose, Interactive Retrieval of Multimedia Objects based on Self-Organising Maps, In: *Proc. of the Int. Conf. of the European Society for Fuzzy Logic and Technology (EUSFLAT 2001)*, pp. 377-380, De Montfort University, Leicester, UK,2001.

[17] A. Rauber, LabelSOM: On the Labeling of Self-Organizing Maps, In: *In Proc. of the International Joint Conference on Neural Networks (IJCNN'99)*, pp. 3527-3532, IEEE, Piscataway, NJ,1999.

[18] C. J. van Rijsbergen, A non-classical logic for Information Retrieval, *The Computer Journal*, 29(6), pp. 481-485,1986.

[19] S. E. Robertson, The probability ranking principle, *Journal of Documentation*, 33, pp. 294-304,1977.

[20] G. Salton, J. Allan, and C. Buckley, Automatic structuring and retrieval of large text files, *Communications of the ACM*, 37(2), pp. 97-108,1994.

[21] G. Salton, and C. Buckley, Term Weighting Approaches in Automatic Text Retrieval, *Information Processing & Management*, 24(5), pp. 513-523,1988.

[22] G. Salton, A. Wong, and C. S. Yang, A vector space model for automatic indexing, *Communications of the ACM*, 18(11), pp. 613-620,1975, (see also TR74-218, Cornell University, NY, USA).

[23] M. Steinbach, G. Karypis, and V. Kumara, A Comparison of Document Clustering Techniques, In: *KDD Workshop on Text Mining,2000*, (see also TR #00-034, University of Minnesota, MN).

[24] H. Turtle, and W. Croft, Inference Networks for Document Retrieval, In: *Proc. of the 13th Int. Conf. on Research and Development in Information Retrieval*, pp. 1-24, ACM, New York,1990.