

# User Adaptive Clustering and Visualization of Document Collections

**Andreas Nürnberger**

University of Magdeburg,  
FIN IWS – IR Group  
39106 Magdeburg, Germany  
nuernb@iws.cs.uni-magdeburg.de

**Marcin Detyniecki**

CNRS, Laboratoire d'Informatique de Paris 6,  
University of Paris 6,  
8, rue du Capitaine Scott, 75015 Paris, France  
Marcin.Detyniecki@lip6.fr

## Abstract

An interesting way of accessing collections of documents is by methods of visualization and clustering. Growing self-organizing maps provide such a solution, by adapting their structure automatically to the structure and size of the underlying database. Unfortunately, the result of the clustering process greatly depends on the definition of the describing features and the employed similarity measure and therefore often does not fulfill the users expectancies. In this paper, we present an approach to improve the obtained clustering by considering user feedback (in the form of drag-and-drop) to adapt the underlying topology of the self-organizing map.

## 1 Introduction

Today large archives of text, images, audio and/or video sequences are available. In order to access a specific object a great number of indexing methods have been developed. Almost all of these methods are based on a numerical data space, i.e. an index is computed, which is a numerical feature vector that describes the objects. For instance, text documents are frequently indexed by selected key terms and then each term is represented by a number in a dictionary vector [14]. Other examples are images that can be described by color or texture histograms that are also represented as numerical vectors [2, 9].

Unfortunately, the existing methods are not aware of either the context or the user preferences. In prior work, we implemented a document retrieval system based on growing self-organizing maps [10, 11]. These maps are built based on the provided database taking into account not only neighboring documents, but also introducing new words to the index, if they are considered as relevant. All this, depending

completely on the learning context (i.e. the database used for training). Thus, the system automatically enables a user to visualize, search and navigate in arbitrary document collections that can be represented by numerical vectors.

In this article we go one step further. We present an extension of our system that makes it aware of user-feedback in order to adapt the classification of documents. In the following we give a brief introduction to self-organizing systems and the used document retrieval system. Then, we outline two complementary learning methods based on user-feedback and a generalized version of these methods. We conclude with a brief presentation of an interactive image retrieval system, in which the discussed algorithms have been implemented.

## 2 Self-Organizing Systems

Self-organizing maps [5] are a special architecture of neural networks that clusters high-dimensional data vectors according to a similarity measure. The clusters are arranged in a low-dimensional topology that preserves the neighborhood relations in the high dimensional data space. 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 of distant clusters. Usually, two-dimensional grids of squares or hexagons are used. Although other topologies are possible, two-dimensional maps have the advantage of an intuitive visualization and thus good exploration possibilities.

Several applications have been proposed for the usage of self-organizing maps for classification and exploration of collections of text documents, images or speech data [1, 7, 8]. Our approach for document retrieval [4] combines conventional keyword search methods with several SOM-based views of the document collection, to allow interactive exploration.

In this paper, we will focus on different extensions of the basic algorithm in order to allow

the integration of user-feedback. Other previous extensions of the model include the implementation of growing self-organizing maps, which eliminates the necessity to define the map size manually, leading to more appropriate mappings of the objects [10, 11]. In [12] we discussed how the system could be applied to collections of objects other than text documents.

## 2.1 Using the Maps

The SOM algorithm can be applied to arbitrary document collections, as far as a vector description of the considered objects is given. However, it is essential that the vector consists of object features that represent the characteristics of the objects appropriately and that the used vector-similarity translates a real similarity between the objects. More on the construction of the maps, for instance on the training algorithm, can be found in [6]. Here we focus on the proposed extensions.

A trained map represents a clustering of the object collection. A browsable list of objects can be assigned to every grid cell. A screenshot of our implementation for text document retrieval is shown in Figure 1. Based on this architecture two main querying possibilities are offered: by keyword and by example.

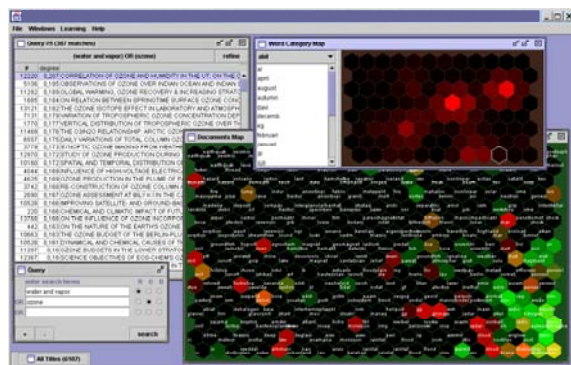


Figure 1. A text document retrieval system based on the proposed approach [4].

## 2.2 Query by Keyword

For a collection of text documents – or for a multimedia database that provides besides numerical features textual information about the objects – a keyword search can be performed resulting in a list of objects that are ranked according to their similarity to the given keywords.

Since a user might also be interested in objects dealing with related keywords the document map – a self-organizing map that is trained by the textual features of the objects – can additionally be used. This map provides a visualization of the search results. Therefore, the search keywords can be associated with colors. The nodes of the document

map are highlighted with blends of these colors to indicate how well the documents assigned to a node match the search terms. This feature enables the user to see how wide the results of his query are spread and thus can give him an idea, if he should further refine the search. E.g. if the highlighted nodes build clusters on the map we can suppose that the corresponding search term was relevant for the neighborhood relations in the learning of the self-organizing map. In this case the probability to find documents with similar topics in adjacent nodes can be expected to be higher.

Furthermore, if the user selects a document in a result list, the node in the map to which this document is assigned is marked and the user can search for similar documents in the surrounding area. The labels of the nodes, which classify the documents that are assigned to the specific nodes, can be used as hints for navigation.

## 2.3 Query by Example

If the user is looking for objects similar to a given sample object (associative search), this sample object can be directly mapped to the document map. The grid cell that is selected as winner unit refers to the objects that are most similar to the provided sample. The user can also use this winning unit as a starting point for navigating to similar objects in its neighborhood.

To improve the visualization, the map can be colored with respect to the distance to surrounding grid cells [3]. In this case, e.g. darker colors visualize great differences to the neighboring cell prototypes while lighter colors represent homogenous regions. Another method for coloring is to visualize the distance between every cell and the sample object provided [7]. Both coloring approaches give the user hints about the structure of the underlying database.

## 3 Incorporating User Feedback

To allow the user to give feedback information, the tool was extended such that a user can drag one or several objects from one node of the map to another, which in his opinion is more appropriate for the considered objects. Furthermore, the user can mark objects that should remain at a specific node, thus preventing the algorithm from moving them together with the moved object during the re-computation of the document allocations.

In the following, we describe two user-feedback models that have been designed to solve specific clustering problems [13]. These two approaches modify the underlying similarity measure by increasing or decreasing the importance of individual features.

### 3.1 Learning Global Feature Weighting

For the implementation of a global feature weighting scheme, we replaced the traditional Euclidean similarity function used for the computation of the winner nodes by a weighted similarity measure. Therefore, the distance of a given feature vector to the feature vectors of the prototypes is computed by

$$e_s = \left( \sum_i w^i \cdot (x_s^i - y_k^i)^2 \right)^{\frac{1}{2}} \quad (1)$$

where  $w$  is a weight vector,  $y_k$  the feature vector of an document  $k$  and  $x_s$  the prototypical feature vector assigned to a node  $s$ .

We update the global-weight vector  $w$  based on the differences between the feature vectors of the moved document and the vectors of the origin node; and of the target node. The goal is to increase the weights of similar features between the document and target node and to decrease the weights of similar features between the document and its current node. And symmetrically decrease and increase the weights of dissimilar features.

Let  $y_i$  be the feature vector of an document  $i$ ,  $s$  be the source and  $t$  the target node,  $x_s$  and  $x_t$  be the corresponding prototypes, then  $w$  is computed as described in the following. First we compute an error vector  $e$  for each object based on the distance to the prototypes

$$e_{ji}^k = |d_{ji}^k|, \forall k, \text{ where } d_{ji} = \frac{y_i - x_j}{\|y_i - x_j\|}. \quad (2)$$

If we want to ensure that an object is moved from the source node to the target node using feature weights, we have to assign higher weights to features that are more similar to the target than to the source node. Thus for each object we compute the difference of the distance vectors

```

Compute the weight vectors  $w_i$ ;
If the global weight vector  $w$  is undefined
  create  $w$ ;
  initialize all elements to one;
end if;
cnt = 0;
Repeat until all documents are moved
  or cnt > max
  cnt++;
  For all documents  $i$  to be moved do
    Compute the winning node  $n$  for  $i$ ;
    if  $N \neq ti$  (target for  $i$ ) then
       $w^k := w^k \cdot w_i^k, \forall k$ ;
      normalize  $w$ ;
    end if;
  end for;
end repeat;

```

**Figure 2. Pseudocode description of the computation of a global weight.**

$$f_i = e_{si} - e_{ti} \quad (3)$$

The global weight vector is finally computed iteratively. For the initial weight vector we choose  $w^{(0)} = w_I$ , where  $w_I$  is a vector where all elements are equal one. Then we compute a new global weight vector  $w^{(t+1)}$  by doing a by element multiplication:

$$w^{k(t+1)} = w^{k(t)} \cdot w_i, \forall k \text{ with } w_i = (w_i + \eta \cdot f_i), \quad (4)$$

where  $\eta$  is a learning rate. The global weight is modified until – if possible – all moved objects are finally mapped to the target node. A pseudocode description of this approach is given in Figure 2.

Obviously, this weighting approach also affects the assignments of all other documents. The idea is to interactively find a feature weighting scheme that improves the overall classification performance of the map. Without a feature weighting approach the map considers all features equally important.

Notice that the global weights reflect the user preferences and therefore can be used to identify features that the user considers important or less important. If, for example, text documents are used where the features represents terms, then we might get some information about the keywords that the user seems to consider important for the classification of the documents.

### 3.2 Learning a Local Weighting Scheme

The global weighting scheme emphasizes on general characteristics, which support a good overall grouping of the data collection. Unfortunately, this may lead to large groups of cells with quite similar documents. In this case some features – which are of less importance on a global scope – might be useful for distinguishing between local characteristics. Thus, modifying locally the weights assigned to these features might improve the assignment of the documents to more specific *local* classes.

The proposed learning method used is quite similar to the method described above. However, instead of modifying the global weight  $w$ , we modify local weights assigned to the source and the target nodes (noted here  $w_s$  and  $w_t$ ).

As before we first compute an error vector  $e$  for each document based on the distance to the prototypes, as defined in equation (2).

Then we set all elements of the weight vectors  $w_s$  and  $w_t$  to one and compute local document weights  $w_{si}$  and  $w_{ti}$  by adding (subtracting) the error terms from the neutral weighting scheme  $w_I$ . Then we compute the local weights iteratively similar to the global weighting approach:

$$w_s^{k(t+1)} = w_s^{k(t)} \cdot w_{si}, \forall k, \text{ with } w_{si} = w_i + \eta \cdot e_{si} \quad (5)$$

and

$$w_t^{k(t+1)} = w_t^{k(t)} \cdot w_{ti}, \forall k, \text{ with } w_{ti} = w_i - \eta \cdot e_{ti} \quad (6)$$

where  $\eta$  is a learning rate. The weights assigned to the target and source node are finally normalized such that the sum over all elements equals the number of features in the vector, i.e.

$$\sum_k w_s^k = \sum_k w_t^k = \sum_k 1 \quad (7)$$

In this way the weights assigned to features that achieved a higher (lower) error are decreased (increased) for the target node and vice versa for the source node.

### 3.3 A Generalized Learning Model

With the local approach we just modified weighting vectors of the source and target nodes. However, as adjacent map nodes should ideally contain similar documents, one could demand that the weights should not change abruptly between nodes. Thus, it is a natural extension of this approach to modify the weight vectors of the neighboring map units accordingly with a similar mechanism as in the learning of the map. Depending on the radius  $r$  of the neighborhood function, the result would lie between the local approach ( $r = 0$ ) and the global approach ( $r = \infty$ ). In the following, we present such an extension.

As for the local approach we have a weighting vector per node. Then – as before – we start by computing an error vector  $e$  for each object based on the distance to the prototypes, as defined in equation (2).

Based on the error vectors  $e$  weight vectors of each node  $n$  are computed iteratively. For the initial weight vector  $w_n^{k(0)}$  we choose vectors where all elements are equal to one. We then compute a new local weight vector for each node by an elementwise multiplication:

$$w_n^{k(r+1)} = w_n^{k(r)} \cdot w_{ni}^r, \forall k \quad (8)$$

with

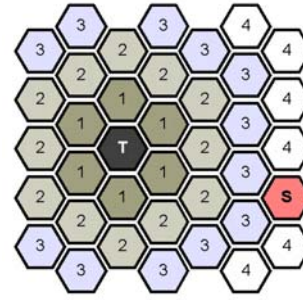
$$w_{ni}^r = w_i + \eta \cdot (g_{sn}^r \cdot e_{si} - g_m^r \cdot e_{ni}).$$

where  $\eta$  is a learning rate and where  $g_{sn}^r$  and  $g_m^r$  are weighting values calculated using a neighborhood function.

Because two quite similar prototypes could be projected in slightly distant cells of the map, e.g. in very dense areas, the neighborhood function should be based on the actual topology of the map. Here we propose to use a linear decreasing function for  $g_{sn}^r$ , which equals one for the source node and equals zero at the hull defined by the radius  $r$ . The same holds for the target node and  $g_m^r$  (see also Figure 3). Notice that more refined functions can be used as for instance Gaussian-like functions.

As above, all weights vectors are modified until – if possible – all moved objects are finally mapped to

the target node. A pseudocode description of this approach is given in Figure 4.



**Figure 3. Neighborhood function centered on target node (decreasing to zero for  $r$ )**

This weighting approach affects the assignments of documents of neighboring cells. The influence of the modification is controlled by the neighborhood function. The idea is that a local modification has a more global repercussion on the map. In this way we can interactively find a feature weighting scheme that improves the classification performance of the map.

## 4 Application Example: An Image Retrieval System

The discussed learning methods were implemented in an interactive tool for image retrieval.

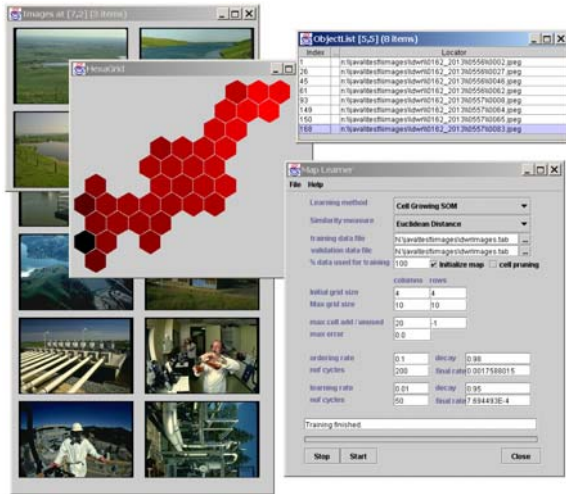
As sample data set we used a picture database provided from the California Department of Water Resources. The database contains more than 17,000 color images of scenery, of buildings, animals, people, etc. They are available from the web server of the *Digital Library Project*, University of California, Berkeley (<http://elib.cs.berkeley.edu/>).

```

For each node n
  compute the weight vectors  $w_{ni}$ 
end for;
If any weight vector  $w_n$  is undefined, then
  create  $w_n$ ;
  initialize all elements to one;
end if;
cnt = 0;
Repeat until all documents are moved
  or cnt > max
  cnt++;
  For all documents  $i$  to be moved do
    Compute the winning node  $N$  for  $i$ ;
    if  $N \neq t_i$  (target for  $i$ ) then
      for all nodes  $n$  of the map
         $w_n^k := w_n^k \cdot w_{ni}^k, \forall k$ ;
      end for;
      normalize  $w_n$ ;
    end if;
  end for;
end repeat;

```

**Figure 4. Pseudocode description of the generalized weighting scheme computation.**



**Figure 5. The learning environment: Resulting map and some grouped images. The map is colored according to the similarity between a selected image and the prototypes assigned to the map nodes.**

The pictures are provided as jpeg images, each picture has a size of 192x128 pixels.

For the evaluation of the tool we selected a small subset of 186 images and computed 303 dimensional feature vectors for each picture based on histograms computed on the pixel colors in the HSV color model. Then we trained the growing self-organizing map starting with 4x4 neurons. A resulting map and the learning environment are shown in Figure 5.

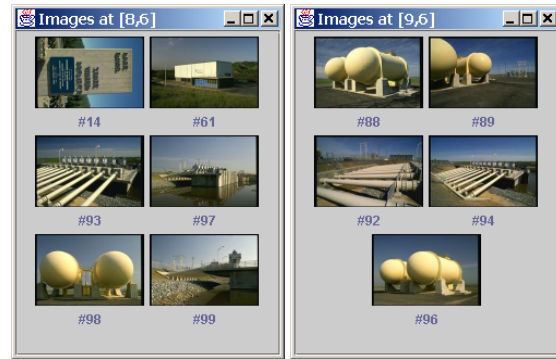
#### 4.1 Discussion of user feedback models

In the following we briefly discuss the effects of the feedback models based on the database described above.

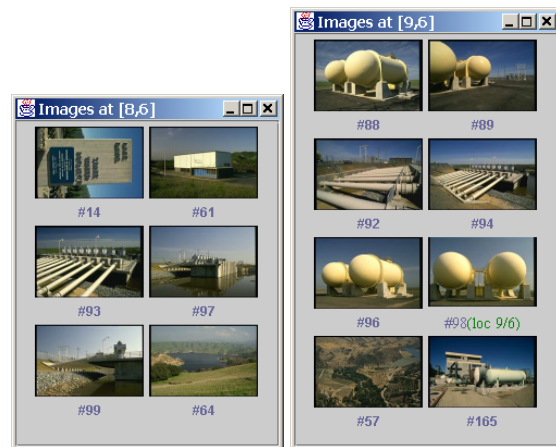
After learning of the image map, we obtained at the neighboring nodes (8,6) and (9,6) the image groups shown in Figure 6. Unfortunately, there are still very similar images distributed over both groups. To improve the grouping we moved the image #98 showing a tank from node (8,6) to node (9,6). Then we retrained the system using all three discussed user feed-back approaches.

In Figure 7 and Figure 8 the obtained image groups after retraining are shown. As we can see, the local weighting approach is more selective and only moves image #98 to the new node. The global weighting approach adds images to both groups, while to the target node (9,6) also a similar tank image (#165) was added.

The color histograms of the moved image (#98) and the prototypes assigned to the source node (8,6) and the target node (9,6) are shown in Figure 9. Furthermore, the learned weighting vectors are shown. As expected, the global weighting vector increases features, where the error between the image feature and the source prototype (8,6) is larger than



**Figure 6. Image groups derived.**



**Figure 7. Image groups after moving image #98 using the global weighting approach.**



**Figure 8. Image groups after moving image #98 using the local weighting approach and the semi-supervised self-organizing map learning.**

between the image feature and the target prototype. Also the local weighting vectors enhance (dampens) errors between the image and the source (target) prototype and vice versa.

If we continue to move several pictures between groups, we can see that the approach is able to reflect the desired changes more specifically than by moving just one picture. However, since the used image features only use color information to describe the images, we cannot expect to obtain every grouping result we would like to have (for instance a semantic classification that do not corresponds to a

color reality). In further research, we will analyze the performance based on more refined image feature models and on text document collections. Nevertheless, the results obtain so far are rather promising.

## 5 Conclusions

Self-organizing maps provide valuable means for visualization and exploration of any object collection that can be described by numerical feature vectors. The combination with traditional search and coloring methods allows the design of interactive and user-friendly object retrieval tools.

The presented learning approaches incorporate user-feedback and thus allowing to refine the map, which was initially trained in an unsupervised manner, with respect to user specific interests. The initial clustering that only depends on the definition of describing features and the similarity measures is

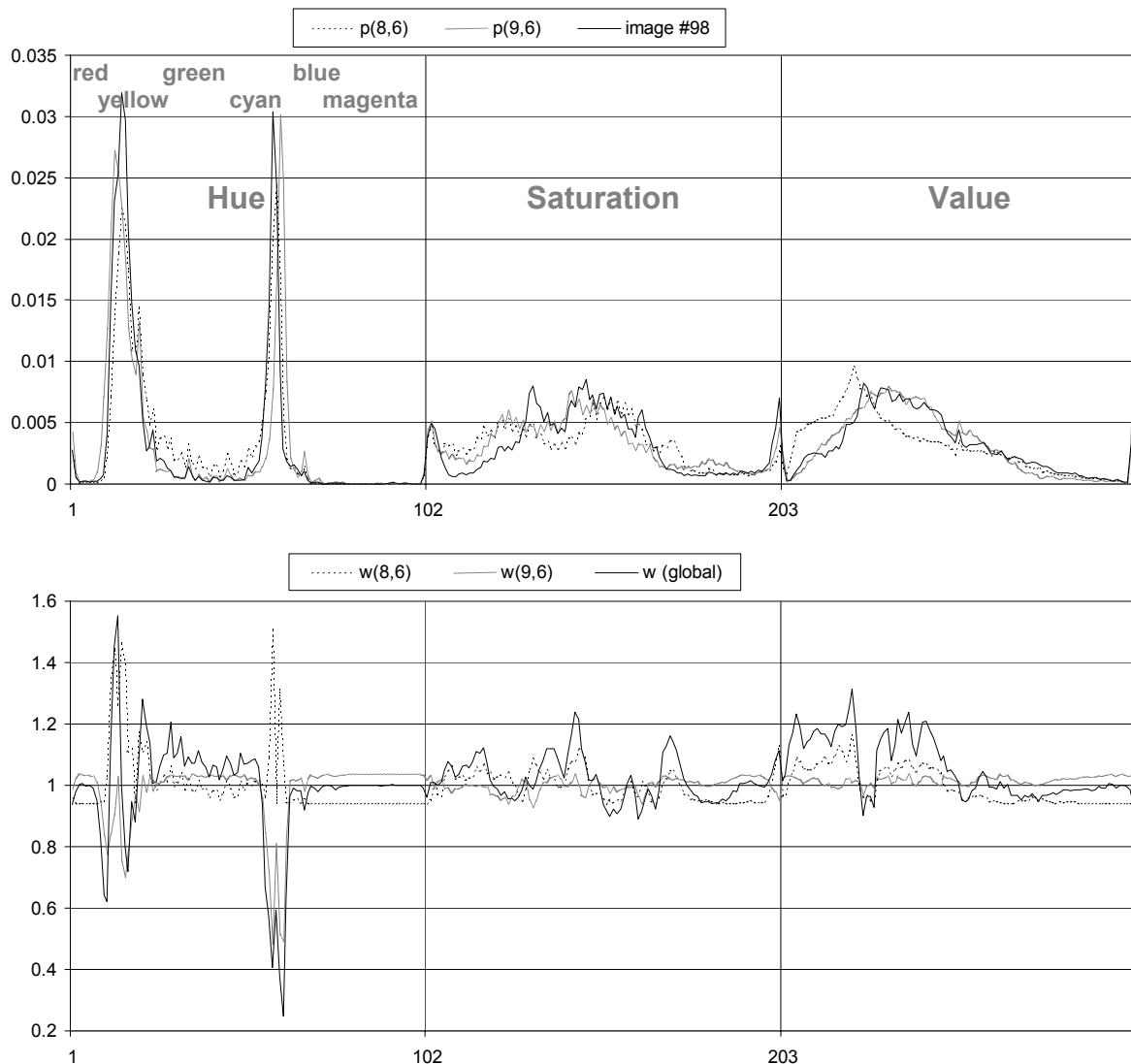
adjusted by gathered information about user specific grouping criteria. Thus, a ‘user desired’ similarity measure is obtained by modification of the importance of certain features using weights.

Using this learning approaches, an interactive retrieval tool could be developed that adapts its visualization and clustering to user specific needs. Since the obtained clustering reflects the expectancies of a user, these techniques can increase the user acceptance as well as the retrieval performance.

## Acknowledgements

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

We like to thank the California Department of Water Resources and the Digital Library Project, UC Berkeley, for providing the photos.



**Figure 9. Top: Features (histograms) for prototypes of node (8,6), (9,6) and the moved image (#98) – Bottom: Learned weighting vectors.**

## References

- [1] 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-Verlag, Berlin, 2001.
- [2] V. Gudivada, and J. V. Raghavan, *Special issue on content-based image retrieval systems*, *IEEE Computer Mag.* 28(9), IEEE, 1995.
- [3] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, *Newsgroup Exploration with the WEBSOM Method and Browsing Interface*, *Technical Report*, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 1996.
- [4] 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: Solid Earth and Geodesy*, 25(8), pp. 649-654, Elsevier Science, Amsterdam, 2000.
- [5] T. Kohonen, *Self-Organized Formation of Topologically Correct Feature Maps*, *Biological Cybernetics*, 43, pp. 59-69, 1982.
- [6] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.
- [7] M. Kurimo, *Indexing Audio Documents by using Latent Semantic Analysis and SOM*, In: S. Oja, and E. Kaski (eds.), *Kohonen Maps*, pp. 363-374, Elsevier, Amsterdam, 1999.
- [8] J. Laaksonen, M. Koskela, and E. Oja, *PicSOM: Self-Organizing Maps for Content-Based Image Retrieval*, In: *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN'99)*, IEEE, Washington, DC, 1999.
- [9] A. Narasimhalu, *Special issue on content-based retrieval*, *ACM Multimedia Systems*, 3(1), 1995.
- [10] A. Nürnberger, *Interactive Text Retrieval Supported by Growing Self-Organizing Maps*, In: T. Ojala (edt.), *Proc. of the International Workshop on Information Retrieval (IR 2001)*, pp. 61-70, Infotech, Oulu, Finland, 2001.
- [11] A. Nürnberger, and M. Detyniecki, *Content Based Analysis of Email Databases Using Self-Organizing Maps*, In: *Proc. of the European Symposium on Intelligent Technologies (EUNITE 2001)*, Verlag Mainz, Aachen, 2001.
- [12] 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.
- [13] A. Nürnberger, and A. Klose, *Improving Clustering and Visualization of Multimedia Data Using Interactive User Feedback*, In: *Proc. of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*, pp. 993-999, 2002.
- [14] 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.