

# Externally Growing Self-Organizing Maps and Its Application to E-mail Database Visualization and Exploration

**Andreas Nürnberger**

University of Magdeburg, FIN IWS - IR Group  
Universitätsplatz 2  
39106 Magdeburg, Germany  
E-mail: nuernb@iws.cs.uni-magdeburg.de

**Marcin Detyniecki**

CNRS - Laboratoire d'Informatique de Paris 6  
8, Rue du Capitaine Scott  
75015 Paris, France  
E-mail: Marcin.Detyniecki@lip6.fr

**ABSTRACT:** In this paper we present an approach to organize and classify e-mails using self-organizing maps. The aim is on the one hand to provide an intuitive visual profile of the considered mailing lists and on the other hand to offer an intuitive navigation tool, where similar e-mails are located close to each other, so that the user can scan easily for e-mails similar in content. To be able to evaluate this approach we have developed a prototypical software tool that imports messages from a mailing list and arranges/groups these e-mails based on a similarity measure. The tool combines conventional keyword search methods with a visualization of the considered e-mail collection. The prototype was developed based on externally growing self-organizing maps, which solve some problems of conventional self-organizing maps and which are computationally viable. Besides the underlying algorithms we present and discuss some system evaluations in order to show the capabilities of the approach.

**KEYWORDS:** growing self-organizing maps, e-mail, document classification, visualization, information retrieval.

## 1. Introduction

Electronic mail has become one of the most widespread ways of communication in today's society. According to a study of Pitney Bowes, UK [26], the average British worker was sending or receiving around 190 messages every day in June 2000. A great part of these e-mails is coming from mailing lists. These lists can be viewed as a subcategory in e-mail. It is hard to determine the number of e-mail coming from mailing lists, but we can approximate it based on some available statistics. A study on mailing lists [20] has shown that 30% of them are managed using **LISTSERV**, which is one of the most frequently used mailing list managers. **LISTSERV** sends 30 million messages per day in approximately 190 thousand mailing lists [19]. Using this information, the authors [20] estimate the total number of mailing list messages at 36.5 billion per year. The pressure to handle this increased number of e-mails is becoming a real problem, in particular since the Internet is the fastest growing media in today's world. In fact, the Internet growth is still accelerating, which indicates that the Internet has not yet reached its highest expansion period.

In this context it is clear that some classification and filtering tools are needed to support the user in extracting relevant information. It has to be noticed that the problem of classifying e-mails is particularly difficult. E-mails and especially mailing list mails are usually very noisy, i.e. the mail contains irrelevant information in the form of signatures or parts of preceding e-mails that may partly cover different topics. At the same time mails may be extremely short and therefore very difficult to classify. Also e-mails are very rich in made-up words, slang, abbreviations or

combinations of characters that may be significant (as for instance e-mail smilies). So, the use of manually-made static dictionaries has to be avoided. Therefore, the approach we present in the following learns the vocabulary automatically from the e-mails provided. Furthermore, due to the performed preprocessing steps and the applied grouping approach, which does not focus on a crisp classification of e-mails in specific categories, but on arranging the documents based on their similarity, the applied method is quite tolerant to the type of noise encountered.

In the following section we briefly motivate the use of self-organizing maps for this specific task and discuss their architecture (Sect. 2). We show that there are some fundamental limitations on the classical approach, thus we introduce in Section 3 a growing algorithm, which is particularly computationally efficient, since the map is growing only by adding external cells. We also present an analysis of this type of growing by controlling the initial position of added data. In Section 4 we describe how this maps can be used in the case of text data (in particular e-mails). Furthermore, we briefly discuss the applied document pre-processing techniques and the methods used for grouping the text documents based on different similarity measures. Finally, we present a software implementation of this approach for searching and visualizing the contents of e-mail databases (Sect. 5).

## 2. Self-organizing maps

Self-organizing maps (SOMs) are a specific architecture of neural networks that cluster high-dimensional data vectors according to a similarity measure [13]. The clusters are arranged in a low-dimensional topology – usually a grid structure – that preserves the neighbourhood relations existing 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. Usually, two-dimensional arrangements of squares or hexagons are used for the definition of the neighborhood relations. Although other topologies are possible, two-dimensional maps have the advantage of intuitive visualization and thus good exploration possibilities. Therefore two-dimensional self-organization maps are frequently used to cluster, visualize and interpret large high-dimensional data sets coming from process reports, image or document features, as well as commercial or financial data [2].

In document retrieval, self-organizing maps can be used to arrange documents based on a similarity measure (e.g. [8, 17]). 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.

Until now, mainly conventional self-organizing maps are used (for previous work we refer to [2, 9, 12, 17, 25, 30]). The main disadvantage of this architecture is that the size and shape of the network structure has to be defined in advance. In other words the solution space is constraint in advance, making the projection into it more complicated. One of the consequences is that the map usually has to be trained several times to obtain an appropriate solution. This is particularly true for large collections of documents, for which this process is extremely time-consuming. Therefore, we developed an approach that tries to adapt the map to the distribution of the underlying data by a growing process. In the next paragraph we present first briefly the conventional self-organizing maps followed by the modifications we propose for the growing self-organizing maps.

## 2.1 Architecture of self-organizing maps

Self-organizing maps (SOMs) are a particular architecture of neural networks. The network structure has two layers (see Figure 1). The neurons of the input layer correspond to the input's vector dimensions. The output layer (map) contains as many neurons as clusters are 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 of the output layer represents a prototype. Using this particular architecture we train the network using a set of high-dimensional sample vectors. After this learning phase the network is ready to classify any vector (document) from this high-dimensional space.

The maps are trained in an unsupervised manner (i.e. there is no class information for any sample vector) using a competitive learning scheme. Before learning starts, the two-dimensional output structure is fixed (e.g. number of the units and global form) 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  $w_i$ . Usually the Euclidean distance or scalar product is used as similarity measure. The weights  $w_s$  of the winner neuron are modified according to the following equation:  $\forall i: w'_s = w_s + \delta \cdot (w_i - w_s)$ , where  $\delta$  is a learning rate.

To preserve the neighbourhood 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 neuron. Therefore, the adaptation method is extended by a neighbourhood function  $v$ :

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

Where  $\delta$  is a learning rate. By this learning procedure, the structure of the high-dimensional 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, e.g., [14].

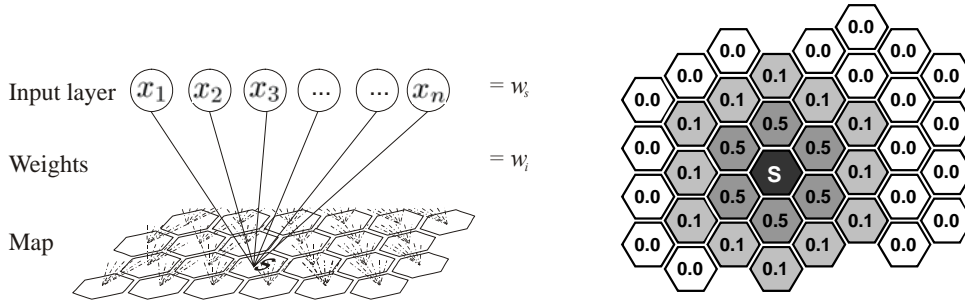


Figure 1. Structure of a rectangular self-organizing map based on hexagons (left) and possible neighborhood function (right).

Unfortunately, this standard learning model of self-organizing maps requires a predefined map structure. The main problem is that the complete learning process has to be repeated if the size of the map was too large or too small, since this leads either to maps where the documents are spread out or to maps with overcrowded cells containing dissimilar documents. Meanwhile some methods have been proposed to obtain the size of the map automatically by a growing process (see, e.g. [1, 6, 21]).

One method to perceive if the size of the map is appropriate is to compute the classification error for all documents. The classification error can be defined for a cell  $i$  by  $E_i = \frac{1}{|C_i|} \sum_{s \in C_i} |w_i - w_s|$ , where  $s$  is a document assigned to this cluster or by an error function computed based on the inner product, which is more appropriate in the case of document vectors (see, e.g. [32]). If the size of the map is too small, then the classification error for every cell is high since dissimilar vectors are assigned to the same unit. Notice that the error is low, if similar vectors are assigned to the same unit. Regrettably the error is also very low if the map is too large. However, this usually leads to empty cells, i.e. cells to which no document is assigned, and thus the growing process can be stopped.

Growing self-organizing map approaches try to exploit the observations described above. The idea is to modify the size (and structure) of the map by adding new units to the map, when the accumulated error on a map-unit exceeds a specified threshold. In the following, our approach is described in more detail.

## 2.2 An externally growing self-organizing map

Our growing self-organizing map approach is mainly motivated by the methods presented in [1, 6, 22]. In contrast to these learning methods we use a hexagonal map structure and *restrict* the algorithm to add new units around the *external units* of the map. This is computationally more efficient than changing the underlying data structure. One may think that this way of growing is restrictive and may limit the learning capabilities of the map, but as we will see in the following section this is not case.

The main principle of our learning method is to add a new unit *close* to the external unit, which achieved the highest error, if the accumulated error of one unit of the map exceeds a specified threshold value. The algorithm can be briefly described as shown in Table 1.

Table 1. Pseudocode description of a learning method for growing self-organizing maps

- |  |
|--|
| <ol style="list-style-type: none"> <li>1. Predefine the initial grid size (usually 2x2 units are used)</li> <li>2. Initialize the assigned vectors with randomly selected values. Reset error values <math>e_i</math> for every unit <math>i</math>.</li> <li>3. Train the map using all inputs patterns for a fixed number of iterations. During training increase the error values of a winner unit <math>s</math> by the current error value for pattern <math>i</math>.</li> <li>4. Identify the unit with the largest accumulated error.</li> <li>5. If the error does not exceed a threshold value stop training.</li> <li>6. Identify the external unit <math>k</math> with the largest accumulated error.</li> <li>7. Add a new unit to the unit <math>k</math>. If more than one free neighboring node is available select the unit at the nearest position to the neighboring unit which is most dissimilar to <math>k</math>. 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).</li> <li>8. Continue with step 3.</li> <li>9. Continue training of the map for a fixed number of iterations. Reduce the learning rate during training.</li> </ol> |
|--|

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. Notice that even if the considered neuron is an inner neuron, the new data pushes the prior assigned patterns to



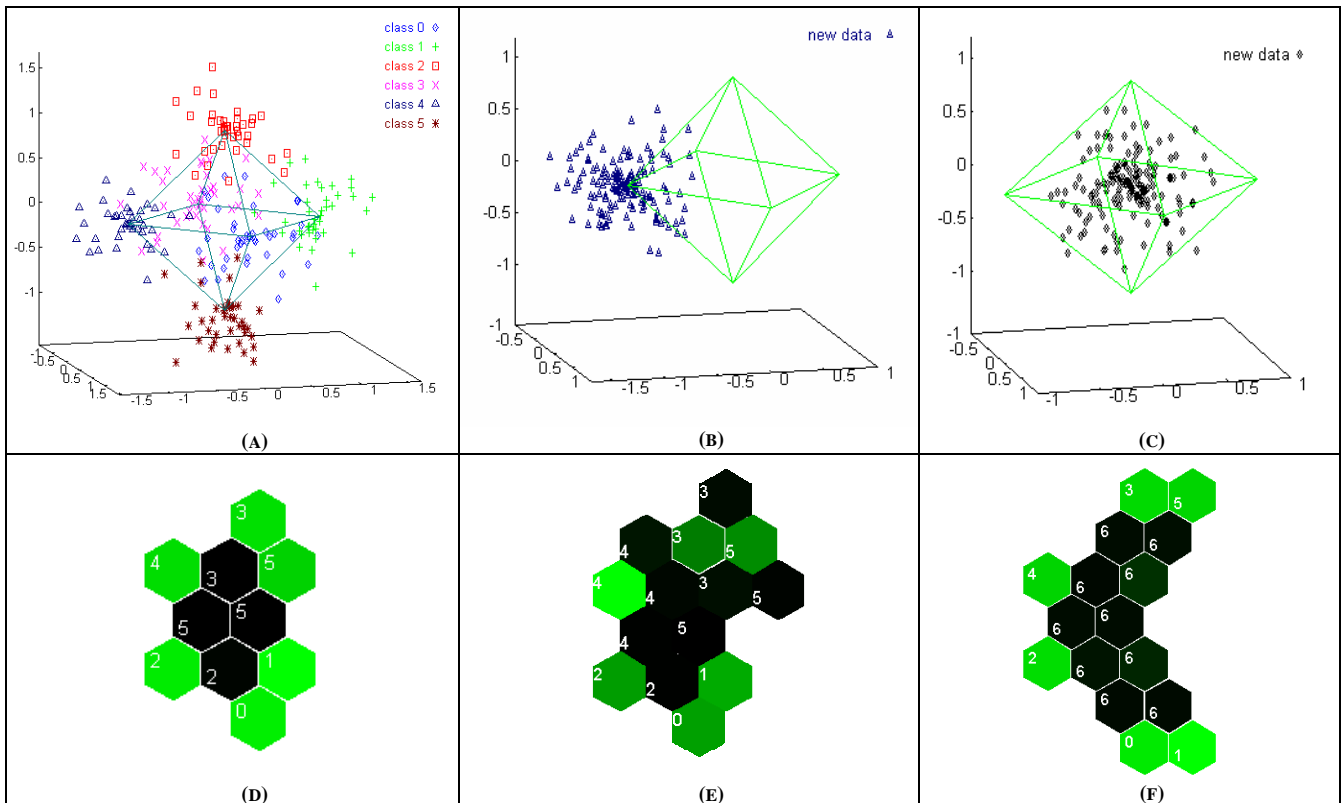


Figure 3. Randomly Generated Training data and resulting Maps. Upper row: Initial Data used for training (A), increased number of elements for Class 4 (B) and introduced class in the center of the training data clouds (C). Lower Row: Resulting Maps (numbers depict classes assigned by majority labelling; colors depict amount of data assigned to a cell, brighter colors depict more data)

### 2.3.2. Introducing a new cluster

One of the specificities of our growing algorithm is that the adding of new cells is done only around the external units of the map. We may expect that problems may arise if we have new data that should be projected in the middle of the map. Therefore, we designed a new experiment in which we introduced a new class in the middle of the data set. For this example we used once more the map obtained with the initial data set (6 classes; Figure 3-D). We create a new class of 150 data points at the center (in-between of the 6 classes). 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 existing classes and pushed them away by creating new cells at the outside of the map (see Figure 3-F).

We obtain the "correct" projection, even if the new cells were added at the borders of the map. Again a user is able to correctly interpret the changes in the underlying data by visual comparison of the resulting map and the initial one.

## 2.4 Introducing new topics (text data)

One of the main differences of the previous experiments and a traditional text data application is the meaning of the data dimension. In fact for any text application every dimension of the space is one word of a dictionary. Therefore we designed a more specific experiment, but first we describe how

the maps in general can be used with text data in the following section. For details on the text data experiment see Section 3.4.

### 3. Building maps for text e-mails collections

As we stated in the introduction the quantity of electronic mail to be managed by any of us is quite impressive. Therefore we need appealing document retrieval and structuring systems. From our perspective, self-organizing maps are an interesting solution since they propose an arrangement of the documents based on their similarity (e.g. [17, 8]). We also notice that this approach opens up several appealing navigation possibilities (see Section 4.1).

For searching and navigating in large document collections (except for 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 files. The currently predominant approaches are the vector space model [34], the probabilistic model [31], the logical model [29] and the Bayesian net model [36]. The vector space model enables very efficient analysis of massive document collections and is therefore 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) [5, 16], Random Projection [11] and Independent Component Analysis (ICA) [10]. The vector space model can also be used for self-organizing maps. In the following we briefly describe the vector space model and corresponding document encoding techniques.

#### 3.1 The vector space model

The vector space model makes possible the analysis of large document collections due to its simple data structure. A document is described based on a ‘statistical fingerprint’ of word occurrences (word frequencies). More precisely, in the vector space model documents are represented as vectors in a  $t$ -dimensional space, i.e. each document  $i$  is described by a numerical feature vector  $D_i = \{x_1, \dots, x_t\}$ , where 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. This is particularly interesting in the case of e-mail text documents, since the vector can be built automatically, so that new words (not in a dictionary) or fancy expressions (as smiles) can be taken into account.

Documents can be compared by use of simple vector operations. For instance queries can be encoded in a query vector of the same 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 [32].

The main problem of the vector space representation of documents is to find an appropriate encoding of the feature vector. The simplest way to do it 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 equal.

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 [33]. 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 [32] 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 (the denominator) is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

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

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.

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

For a more detailed discussion of the vector space model and weighting schemes see, e.g. [3, 7, 33, 34].

In order to improve the representation of text documents with the vector space model, not every term is encoded using an additional dimension. For example, usually plural forms of nouns are mapped to its singular form or terms are reduced to their stems. Thus the dimensionality of the data space can be reduced and semantically similar terms can be represented by only one weight and are thus implicitly encoded as being similar. In the following, we briefly describe the document pre-processing methods we have applied.

### 3.2 Filtering, stemming and index term selection

The idea of stop word *filtering* is to remove words that bear no content information, like articles, conjunctions, prepositions, etc. Furthermore, words that occur extremely often can be said to be of little information content to distinguish between documents. Also, words that occur very seldom are likely to be of no particular statistical relevance.

*Stemming* tries to build the basic forms of words, i.e. strip the plural ‘s’ from nouns, the ‘ing’ from verbs, or other affixes. A stem is a natural group of words with equal (or very similar) meaning. We currently used the stemming algorithm of [27], which uses a set of production rules to iteratively transform (English) words into their stems.

To further reduce the number of words in the vector description we applied a simple method for keyword selection by extracting keywords based on their entropy. In the approach discussed in [12], for each word  $k$  in the vocabulary the entropy as defined by [18] was computed:

$$W_k = 1 + \frac{1}{\log_2 n} \sum_{j=1}^n p_{jk} \log_2 p_{jk} \quad \text{with} \quad p_{jk} = \frac{tf_{jk}}{\sum_{l=1}^n tf_{lk}}$$

where  $tf_{jk}$  is the frequency of word  $k$  in document  $j$ , and  $n$  is the number of documents in the collection. Here the entropy gives a measure of how well a word is suited to separate documents by keyword search. For instance, words that occur in many documents will have low entropy. The entropy can be seen as a measure of the importance of a word in the given domain context. As index-words we choose a number of words that have a high entropy relative to their overall frequency, i.e. from words occurring equally often we prefer those with the higher entropy. This procedure has empirically been found to yield a set of relevant words that are suited as index terms [12].

In order to obtain a fixed number of index terms that appropriately covers the documents, a greedy strategy was applied: From the first document in the set of documents select the term with the highest relative entropy as an index term. Then mark this document and all other documents containing this term. From the first document of the remaining unmarked documents select again the term with the highest relative entropy as an index term. Then mark again this document and all other documents containing this term. Repeat this process until all documents are marked, then unmark all of them and start again. The process can be terminated when the desired number of index terms has been selected. A more detailed discussion of the benefits of this approach for clustering – with respect to reduction of words required in order to obtain a good clustering performance – can be found in [4].

### 3.3 Generating the document map

After the considered document collection (in our case an e-mail database, e.g., a newsgroup) has been pre-processed as described above, i.e. the documents are split into words, then stop words are filtered and the word stems are generated, the considered vocabulary is reduced to a number of groups or *buckets*. These buckets are the selected index words. The words of every document are then sorted into the buckets, i.e. the occurrences of the word stems associated with the buckets are counted. Thus, the document is described based on the word/bucket frequencies. Each of the  $n$  buckets builds a component in a  $n$ -dimensional vector that characterizes the document. These vectors can be seen as the *fingerprints* of each document. An overview of the indexing process is given in Figure 4.

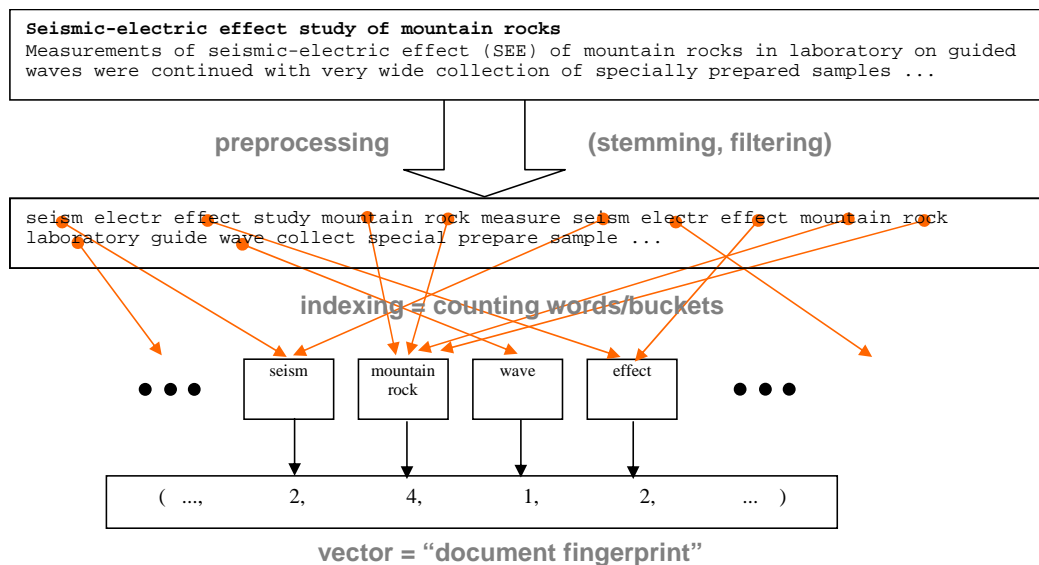


Figure 4. Overview of the indexing process

For every document in the collection such a fingerprint is generated. In our self-organizing map approach, these document vectors are then clustered and arranged into a hexagonal grid – the so-called *document map* – using the SOM learning process as described above. In Figure 5 an example of the growing process is shown. The initial map consisted of 3x3 units. In Figure 5 (a) the first unit was added to the neuron at the top right corner of the map. This unit had achieved the highest error after the map has been initialized and trained.

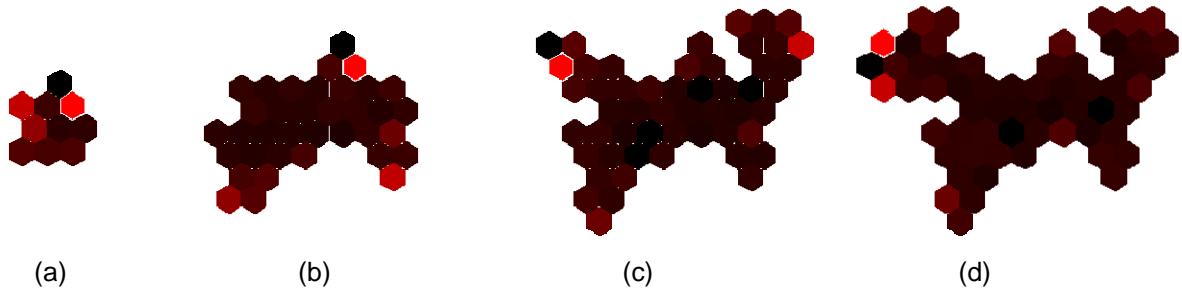


Figure 5. Example of cell growing during learning. (The shading represents the error of a specific unit – lighter colors depict higher errors. The black cells close to the red ones had been added.)

After the training of the map each grid cell is labeled by a specific keyword, which describes the content of the assigned documents. The labeling method we used is to choose for a cell a keyword describing the majority of documents assigned to this cell. For document collections more refined labelling methods have been proposed that also consider the distribution of documents in neighbouring cells, see [15, 28].

An example of a labeled map is given in Figure 6, where also the distribution of the documents in the map is visualized (i.e. the shading represents the number of documents assigned to a specific unit – lighter colors depict less documents).

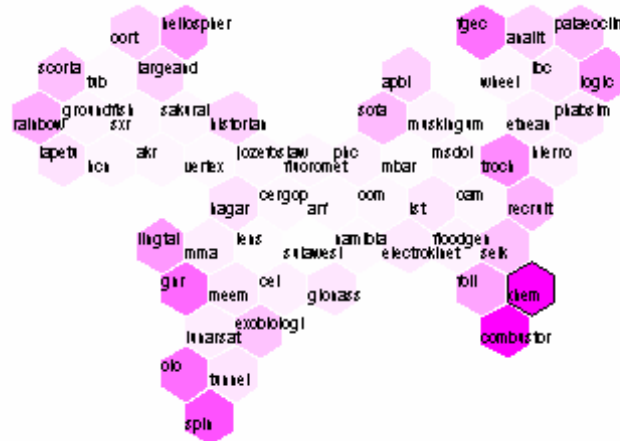


Figure 6. Example of the document distribution after learning from [21]. (The shading represents the number of documents assigned to a specific unit – lighter colors depict less documents.)

At this point the labeled map can be used for the visual exploration of the e-mail collection, as described in more detail in Section 4. But before, we present some text-specific experiments applied to our growing algorithm in order to show how changes in the document collections are represented in the document map.

### 3.4 Introducing a new topic

As we mention before, the structure of the data when using a vector space model for text documents is particular. In this framework, a document is represented simply by a numerical vector. Every dimension represents one word (or a group of words having the same meaning) and therefore the dimensionality of the space is usually very large (it equals the cardinality of the indexing dictionary). Furthermore, frequently some keywords appear in one text and not in the others, leaving the data vector with a lot of zeros (sparse vectors).

In order to show the behaviour of our growing self-organizing map algorithm on this type of data (text), data-distribution and space (vector model), we created two simple artificial data sets. We especially address the problem if new data that is naturally “in the middle” of a document set (in a semantic sense) will be projected still on the middle of the map, since we add the new cells on the borders of the map. Therefore, 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 contain 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 2.

Table 2. Distribution of keywords in artificial text data sample

	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

A self-organizing map was learned using the first data set (A). The resulting map is shown in Figure 7 (left). Since we just want to describe the document distribution, we used the majority class as labelling method.

In a second run we added to the training data set A, the data set B, which contains documents with keyword-combinations as described above. Then we retrained the map. The result is shown in Figure 7 (right). The documents using keyword combinations were placed in-between the ‘pure’ classes, which were 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.

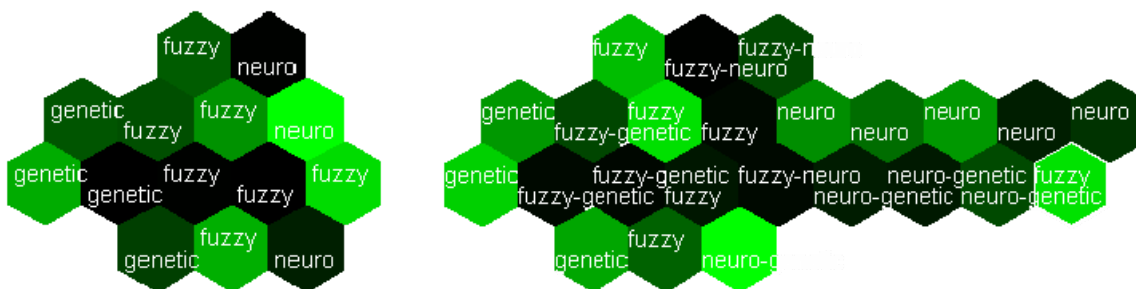


Figure 7. Learned Maps for text data.

## 4. Using the maps to explore e-mail collections

After processing the e-mail database we finally have a document map, where similar mails are grouped. To assess the usability of this approach a software prototype was developed in Java. Once pre-processed and learned, the indexes and maps are stored in a simple database. The tool provides an interface for simple keyword-search and content-based retrieval. Furthermore, the document map can be used to search for similar mails and coloring methods for the visualization of the search results in the document map are provided. In Figure 8 a screenshot of the software is shown.

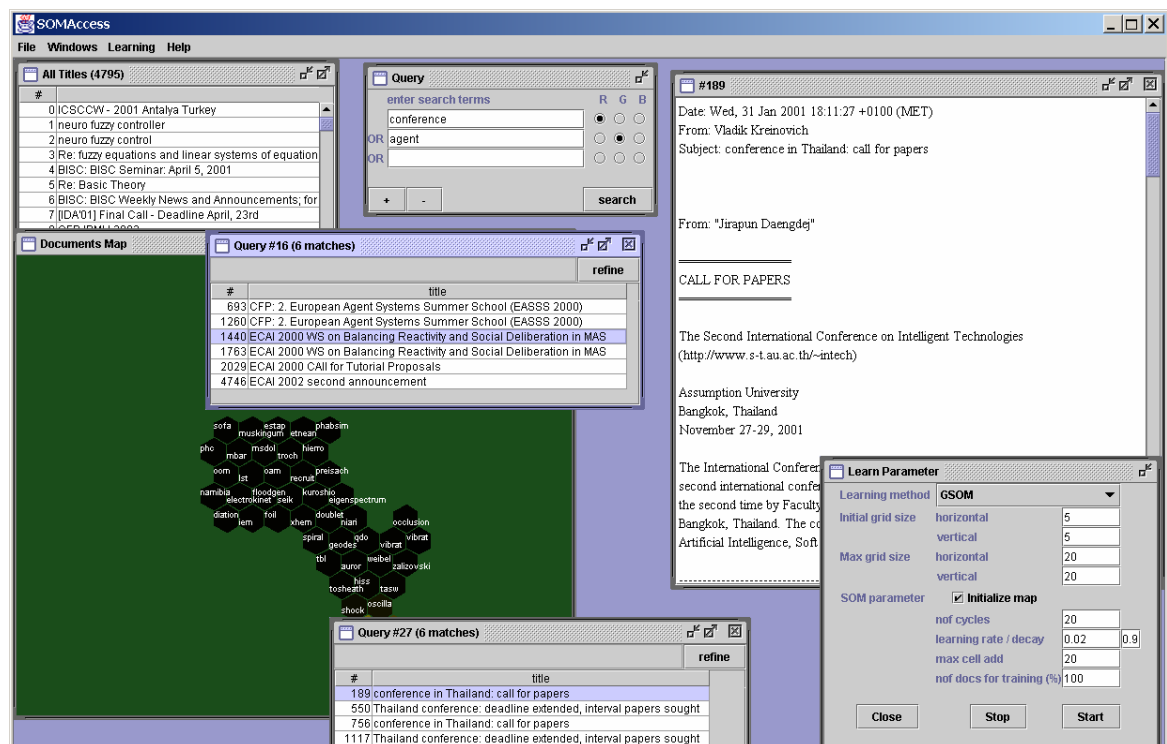


Figure 8. Screenshot of the software tool: Document map (left bottom), query window (top center); search result lists and retrieved document (right), learning dialog window (right bottom). (The used e-mail database represented in this screenshot consists of e-mails from the fuzzy mailing list <fuzzy-mail@dbai.tuwien.ac.at>.)

The complete system can be built with very little user intervention and directly from the e-mail database. It can always be completely reconstructed from scratch with an updated document database. However, this brute force approach is not always necessary, and more reasonable methods are possible:

- For small changes we can keep the learned maps and just add the new documents to the map.
- If we expect extensive changes, we could re-train the document map. The learning algorithm of the growing self-organizing map is incremental. Thus, we can use the old map as an initialization and just slightly rearrange the nodes to better fit the new collection. This way, the maps can also be used to visualize changes in the topics (see also Sect. 3.4).
- Alternatively, we can also relearn a new document map from scratch. However, we can still use the old document encoding, i.e. the automatically chosen set of index words.

When the changes in the collection are more severe, then there is no other solution than rebuilding everything even the set of index words. However, in this case an analysis of the changes on the set of index words might yield interesting hints on current developments of research topics, e.g. upcoming buzzwords.

After training, the user can directly use the e-mail document map to get an overview of the database (e.g. newsgroup) contents by scanning the labels on the map and analysing the e-mails that are assigned to the specific nodes. However, if he is searching for specific information more refined methods are possible. The capabilities of the tool are briefly described in the following.

## 4.1 Visualization and searching

The visualization of e-mails collections provided by self-organizing maps is based on the similarity between the different documents (i.e. similar document should be close to each other). Based on the same idea other clustering approaches are frequently used to find groups of documents with similar content [35]. However, these methods usually do not consider the neighbourhood relations between the obtained cluster centres. Favourably, self-organizing maps were designed to cluster high-dimensional data vectors according to a similarity measure in a low-dimensional topology. Thus, not only the e-mails that are assigned to one cluster (cell) are similar to each other, but also objects of nearby clusters (cells) are expected to be more similar than objects in more distant clusters. This architecture opens up several appealing navigation possibilities. Most important, the surrounding grid cells of documents known to be interesting can be scanned for further similar e-mails. Furthermore, the distribution of keyword search results can be visualized by colouring the grid cells of the map with respect to the number of hits, i.e. the brightness of the colour of a cell depends on the ratio of the total number of documents assigned to a cell to the number of documents assigned to this cell that matches the search criteria. The tool currently supports the possibility to assign three Boolean queries to the colours red, green and blue (see also Figure 8).

In other words the idea is to combine visualisation, navigation and searching for better document retrieval. In the following we present different ways of interaction offered by our model.

### 4.1.1. Keyword search and visualization using the maps

The traditional way to find an e-mail is to use keyword queries. This querying method is integrated in the tool, combined with visual and navigation support. We present a possible search process in order to illustrate our ideas:

- We can start in the traditional way by querying the e-mail database (keyword search) to find initial interesting subsets of the document collection.
- Then we can visually inspect the document map based on the coloring (see next paragraph), i.e. to see how wide the matching documents are spread over the map. We can use this coloring to discover new region of the map to which documents are assigned that are related to the subject, but which might be unknown to the user before. We can obtain new keywords (for query modification or refinement) directly from the labels or by reading the assigned documents.
- We can then refine (or reorient) the query with further keywords, and for potentially relevant documents, inspect documents that lie on the same or adjacent nodes of the document map.

#### 4.1.2. Navigating in the document map

Another way of start the navigation is to assign different search terms to specific colours. The nodes of the document map are then highlighted with blends of these colours to indicate how well the documents assigned to a node match the search terms. This allows a user to judge e.g. whether the search results are assigned to a small number of (neighbouring) 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. The user can then open a cell and scan the documents that are assigned to this cell. This may be interesting since all documents in a cell are similar; furthermore documents in the neighbouring cells should also be similar.

We can also notice, that if the highlighted nodes build clusters on the map, we can suppose that the corresponding search term is relevant for the neighbourhood relations of the map. In this case the probability to find documents with similar topics in adjacent nodes can be expected to be higher.

#### 4.1.3. Content based searching

If a user is searching for e-mails that are similar to an e-mail he already found elsewhere, i.e. he is searching for a document that is similar in content, then this e-mail can be mapped directly to the document map after its fingerprint is computed. In this case the map is colored with respect to the similarity of the considered document to the vector prototypes assigned to each cell. An example is given in Figure 9. The most similar documents – with respect to the used error measure – are assigned to the winning unit (marked by a black frame around the darkest unit in Figure 9). The user can obtain a list of documents assigned to this unit or navigate through surrounding (similar) units.

This approach can also be used to map incoming e-mails and classify them automatically for the user, either by providing the colored map to the user or by using each node (or a group of nodes) as a separate e-mail folder. In this case the map is used as e-mail classification system and the map represents the profile of the e-mail traffic of a specific user.

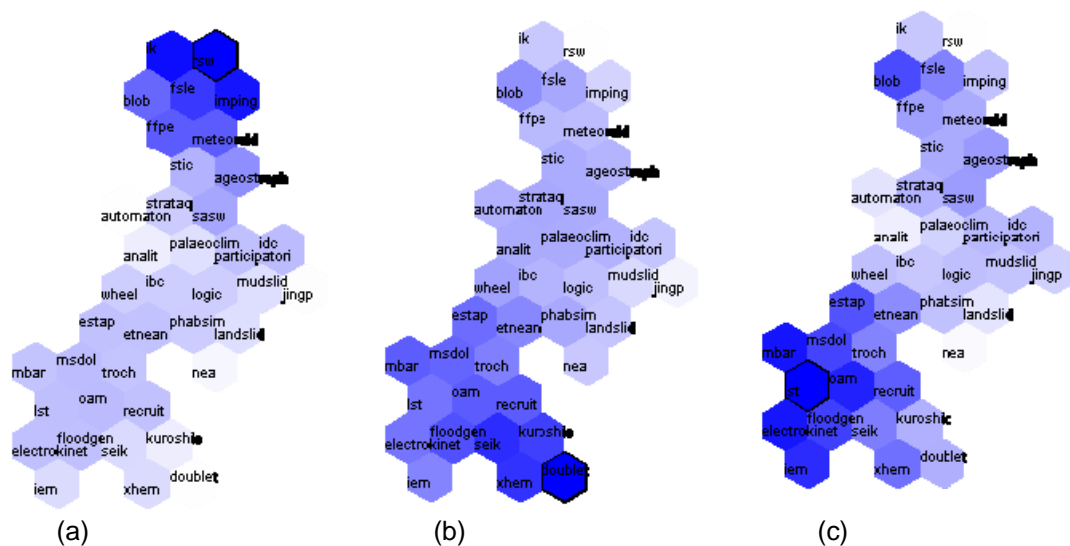


Figure 9. Example of content based searching using a sample document: A black frame marks the winner unit. Each unit is colored according to the similarity of its prototype to the sample documents (dark color means high degree of similarity). In (a) and (b) only local areas of the map are active, while in (c) two areas of the map depicts a high degree of similarity. (The labels depicted in this maps result from the applied stemming algorithm.)

#### 4.1.4. Global and user profile visualization

When visiting for the first time a newsgroup (or any other e-mail database) the user may just want to see what the newsgroup is about. In this case the user has only to look at the labelled map to discover the different topics. Since one label is chosen per cell, we obtain a uniform representation of the clustered space. Again, the user can deepen its search by opening and looking at the documents title or even at the documents itself.

Another interesting visualization, specific to e-mails, is to colour the map based on the author of the e-mails. The idea is to use all e-mails sent by a specific user to characterize his interest. In this way we will be able to visualize on the map any newsgroup participants profile.

#### 4.1.5. Visualizing changes

One limitation of the standard self-organizing maps is its static nature. On the one hand changes in data collections cannot 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 find a similar distribution of the data in the new map. Using our growing approach the user will not only keep the initial structure, but also he will be able to visualize the changes of the document collections.

In fact, if just few documents are introduced, we can colour the map so that we see where they are introduced. If a lot of new e-mails on a new subject are brought in, then new cells will appear, illustrating the changes and the region where they are affected. All this is done without destroying the structure of the initial map.

## 5. Conclusion and future work

Self-organizing maps can be a useful method for clustering and exploration of e-mail collections. Applied to text documents, especially the combination of iterative keyword search and self-organizing maps in an interactive environment provides a valuable tool for document retrieval purposes.

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. Our approach is computationally efficient, since we add the new cells around the growing map. In this paper we empirically proved that this way of growing does not affect the ability of the map to learn any type of data and in particular text when using the vector space representation.

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. This is especially important for rapidly changing document collections like newsgroups, where a lot of new terms, abbreviations or spellings are introduced, as for instance smilies and abbreviations like "LOL" that stands for "laughing out loud" or "<g>" for "grin", or in specialized user newsgroups where users are naturally highly interested in current and upcoming topics.

So far, the results are encouraging. E-mails on a particular subject are classified in the same cells and sequences of reply messages tend also to be in the same cell, which indicates that the system detects that these chains keep the discussion on the same subject. Nevertheless, the present approach has still some insufficiencies. One problem is that very short e-mails are often not correctly classified. In fact, the text appearing in short e-mails provides very often insufficient information to obtain an appropriate classification. Further work should focus on the exploitation of intrinsic information, for instance by giving a heavier weight to the subject field in case of chained e-mails. However, the presented tool already gives a good intuition of what can be achieved using self-organizing maps combined with conventional search methods.

Future work is also directed towards the use of non-text documents (e.g. images) and the integration of user feedback, so that the underlying similarity measure can adapt to user specific grouping criteria. Some results of first studies in this direction can be found in [23, 24].

## 6. Acknowledgements

The work presented in this article was partially supported by BText Technologies, Adastral Park, Martlesham, UK, and the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE).

## 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] C. Borgelt, A. Nürnberger, Fast Fuzzy Clustering of Web Page Collections, *In: Proc. of PKDD Workshop on Statistical Approaches for Web Mining (SAWM)*, Pisa, Italy, 2004.
- [5] 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.
- [6] B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, *Neural Networks*, 7(9), pp. 1441-1460, 1994.
- [7] 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.
- [8] T. Honkela, *Self-Organizing Maps in Natural Language Processing*, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 1997.
- [9] 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.
- [10] 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.

- [11] 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.
- [12] 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.
- [13] T. Kohonen, Self-Organized Formation of Topologically Correct Feature Maps, *Biological Cybernetics*, 43, pp. 59-69, 1982.
- [14] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.
- [15] 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.
- [16] T. K. Landauer, P. W. Foltz, and D. Laham, An Introduction to Latent Semantic Analysis, *Discourse Processes*, 25, pp. 259-284, 1998.
- [17] 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.
- [18] K. E. Lochbaum, and L. A. Streeter, Combining and comparing the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval, *Information Processing and Management*, 25(6), pp. 665-676, 1989.
- [19] L-Soft, Listserv statistics (<http://www.lsoft.com/products/default.asp?item=listserv>), 2001.
- [20] P. Lyman, and H. R. Varian, How Much Information?, Project Report, School of Information Management and Systems, University of California, Berkeley, 2000.
- [21] 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.
- [22] A. Nürnberger and M. Detyniecki. Visualizing Changes in Data Collections Using Growing Self-Organizing Maps, In *Proc. of International Joint Conference on Neural Networks (IJCNN 2002)*, pp. 1912-1917, Honolulu, Hawaii, 2002.
- [23] A. Nürnberger and M. Detyniecki, Weighted Self-Organizing Maps: Incorporating User Feedback, In: *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Proc. of the joined 13<sup>th</sup> International Conference*, LNCS Series Vol. 2714, pp. 883-890, Springer-Verlag, 2003.
- [24] 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.
- [25] A. Nürnberger, A. Klose, R. Kruse, G. Hartmann, and M. Richards, Interactive Text Retrieval Based on Document Similarities, In: G. Hartmann, A. Nölle, M. Richards, and R. Leitinger (eds.), *Data Utilization Software Tools 2 (DUST-2 CD-ROM)*, Max-Planck-Institut für Aeronomie, Katlenburg-Lindau, Germany, 2000.
- [26] Pitney Bowes, Messaging Practices in the Knowledge Economy (Study; <http://www.pitneybowes.co.uk/news/article.asp?article=103>), UK, 2000.

- [27]M. Porter, An algorithm for suffix stripping, *Program*, pp. 130-137, 1980.
- [28]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.
- [29]C. J. van Rijsbergen, A non-classical logic for Information Retrieval, *The Computer Journal*, 29(6), pp. 481-485,1986.
- [30]H. Ritter, and T. Kohonen, Self-organizing semantic maps, *Biological Cybernetics*, 61(4), 1989.
- [31]S. E. Robertson, The probability ranking principle, *Journal of Documentation*, 33, pp. 294-304, 1977.
- [32]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.
- [33]G. Salton, and C. Buckley, Term Weighting Approaches in Automatic Text Retrieval, *Information Processing & Management*, 24(5), pp. 513-523, 1988.
- [34]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).
- [35]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).
- [36]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.