Self organization of a massive document collection

Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojärvi, Jukka Honkela, Vesa Paatero, and Antti Saarela

The authors are with the Neural Networks Research Centre, Helsinki University of Technology, Espoo, Finland. E-mail: websom@websom.hut.fi

Abstract

This article describes the implementation of a system that is able to organize vast document collections according to textual similarities. It is based on the Self-Organizing Map (SOM) algorithm. As the feature vectors for the documents we use statistical representations of their vocabularies. The main goal in our work has been to scale up the SOM algorithm to be able to deal with large amounts of high-dimensional data. In a practical experiment we mapped 6,840,568 patent abstracts onto a 1,002,240-node SOM. As the feature vectors we used 500-dimensional vectors of stochastic figures obtained as random projections of weighted word histograms.

Keywords

Data mining, exploratory data analysis, knowledge discovery, large databases, parallel implementation, random projection, Self-Organizing Map (SOM), textual documents.

I. Introduction

A. From simple searches to browsing of self-organized data collections

Locating documents on the basis of keywords and simple search expressions is a commonplace task nowadays. However, formulating effective search queries is rather difficult, and scanning through the lists of search results that are in no apparent meaningful order may be very tiresome.

A more user-friendly method for data exploration is exemplified in the so-called Hypertext approach where links are provided between a document and other related data. In the World Wide Web these links are often created manually by individual users, and the quality of linking varies greatly over documents. The quality can be improved with extensive human labor by constructing organized collections like the YAHOO, a hierarchical directory of Internet resources (http://www.yahoo.com/).

It would be of immense help if the document collections and databases could be *automatically* organized in some meaningful way. Especially interactive exploration of a document collection, where the user looks at individual documents one at a time, would be greatly aided by ordering of the documents according to their contents. The context of the other related documents residing nearby would then help in understanding the true meaning of the individual texts and in finding the information of highest interest. Furthermore, interpreting the results of searches would become easier if the results were already grouped according to the similarity of content, instead of returning matches as a list of hits. There exist many possibilities to organize a document collection, e.g. as a graph or a hierarchical structure.

The present article describes an organization in which the documents are represented as points on a two-dimensional plane, and the geometric relations of the image points of the documents represent their similarity relations. Such representations are called "maps."

Document maps add value to text retrieval by providing a meaningful visual background for portraying the results of searches. The background helps in making sense of the retrieved matches, as well as provides cues for selecting the most interesting ones. In addition, one may index the image points by information derived from the document collection, and the indexing can be utilized to perform further searches on the collection.

Organized collections of data also facilitate a new dimension in retrieval, namely, the possibility to locate pieces of relevant or similar information that the user was not explicitly looking for. Such tasks and methods constitute a field of their own called *exploratory data analysis* or *knowledge discovery in databases*, often colloquially called "data mining."

B. The scope of this work

There exist several classical methods in exploratory data analysis [1] and multivariate analysis that are able to form illustrative two-dimensional *projections* of distributions of items in high-dimensional data spaces. One of them is *multidimensional scaling (MDS)* [2], [3], [4], [5], [6], [7] and its frequently applied version is called *Sammon's projection* [8]. For large amounts of data items these mappings are computationally heavy. Therefore, considerable interest might be devoted to the neural-network methods, e.g., the *Self-Organizing Map (SOM)* [9], [10], [11] that approximate an unlimited number of input data items by a finite set of models. A further advantage achieved by the SOM mapping is then that unlike in, say, multidimensional scaling, the SOM can first be computed using any representative subset of old input data, and new input items can be mapped straight into the most similar models without re-computation of the whole mapping.

Any of the basic projection methods can be used to organize textual data items, too, such as documents, if their contents are described statistically as some kind of metric feature vectors. For instance, if the collection of words used in a document is described as a histogram, the latter can serve as the input feature vector on the basis of which the document collection can be organized.

We have developed a SOM-based methodology that can be used as a tool especially in exploring document collections but also in various searching tasks. In this method called the WEBSOM, a textual document collection is organized onto a graphical map display that provides an overview of the collection and facilitates interactive browsing. The browsing can be focused by locating first some interesting documents on the map using content addressing.

Since 1991, there have existed attempts to apply the SOM for the organization of texts, based on word histograms regarded as the input vectors [12], [13], [14]. In order to avoid their dimensionalities from growing too large, the vocabularies were limited manually. However, to classify masses of natural texts, it is unavoidable to refer to a rather large vocabulary, say, 50 000 words. There exist at least three possibilities to reduce the dimensionalities of the histogram vectors, without essentially losing accuracy in classification: 1. Representation of the histogram vectors by their eigenvectors (the latent semantic indexing described in Sec. III-B), 2. Clustering of words into semantic categories, as was done in our earlier WEBSOM publications [15], [16], [17], [18], [19], [20], [21], 3. Reduction of the dimensionality of the histogram vectors by a random projection method, as done in this work.

The present article describes the final phases of a major project that was launched in 1995. After several phases of development, it was decided in the summer of 1997 that one should make an experiment to demonstrate the up-scalability of the SOM method. While about 5 000 documents were still mapped in our first publications in 1996 [15], [16], [17], [18], we finally increased the database to about seven million documents [22]. There do not exist many sources of freely available information of this size. In order that our work would also lead to a useful application, we decided to use the corpus of all the patent abstracts that were available on CD ROMs or other electronic media. This corpus consisted of U.S., Japanese, and European patents.

It would have been interesting to compare the up-scalability of our method with other algorithms and variants of the SOM. However, as it took two years from our group to develop the final software for our method, it was not possible for us to construct alternative search methods of the same dimension for benchmarking. To make our system operate in real time and fit to medium-sized computers, we also had to develop several shortcut computing solutions that obviously cannot be used with other methods.

II. The Self-Organizing Map

The *Self-Organizing Map (SOM)* [11] is an unsupervised-learning neural-network method that produces a *similarity graph of input data.* It consists of a finite set of models that approximate the open set of input data, and the models are associated with nodes ("neurons") that are arranged as a regular, usually two-dimensional grid. The models are produced by a learning process that automatically orders them on the two-dimensional grid along with their mutual similarity.

The original SOM algorithm is a recursive regression process. Regression of an ordered set of model vectors $\mathbf{m}_i \in \mathbb{R}^n$ into the space of observation vectors $\mathbf{x} \in \mathbb{R}^n$ can be made recursively as

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{c(\mathbf{x}),i}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \ , \tag{1}$$

$$c(\mathbf{x}) = \arg\min_i \{\|\mathbf{x} - \mathbf{m}_i\|\} \ , \tag{2}$$

where t is the index of the regression step, and the regression is performed for each presentation of a sample of $\mathbf{x}$, denoted $\mathbf{x}(t)$. The scalar multiplier $h_{c(\mathbf{x}),i}(t)$ is called the *neighborhood function*, and it is like a smoothing kernel over the grid. Its first subscript $c(\mathbf{x})$ is defined by eqn (2), that is, $\mathbf{m}_{c(\mathbf{x})}(t)$ is the model (called the *"winner"*) that matches best with $\mathbf{x}(t)$. The comparison metric is usually selected as Euclidean; for other metrics, the form of (1) will change accordingly. If the samples $\mathbf{x}(t)$ are stochastic and have a continuous density function, the probability for having multiple minima in (2) is zero. With discrete-valued variables, however, multiple minima may occur; in such cases one of them can be selected at random for the winner.

The neighborhood function is often taken as the Gaussian

$$h_{c(\mathbf{x}),i}(t) = \alpha(t) \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{c(\mathbf{x})}\|^2}{2\sigma^2(t)}\right) \ , \tag{3}$$

where $0 < \alpha(t) < 1$ is the learning-rate factor which decreases monotonically with the regression steps, $\mathbf{r}_i \in \mathbb{R}^2$ and $\mathbf{r}_{c(\mathbf{x})} \in \mathbb{R}^2$ are the vectorial locations on the display grid, and $\sigma(t)$ corresponds to the width of the neighborhood function, which is also decreasing monotonically with the regression steps. In practice, for computational reasons, $h_{c(\mathbf{x}),i}(t)$ is truncated when $\|\mathbf{r}_i - \mathbf{r}_{c(\mathbf{x})}\|$ exceeds a certain limit.

It has been thought that the SOM algorithm might be derivable from some objective function that describes the average quantization error. In a recent study [23] it was shown that a different point density of the model vectors is thereby obtained. In this work we use the original SOM algorithm, which is computationally lightest of all variants. This aspect was most decisive in this very large implementation.

In an attempt to accelerate the computation of the SOM, the *Batch Map* principle [24] has turned out to be computationally very effective. The implementation of the present method is based on the Batch Map.

Assuming that the convergence to some ordered state is true, we require that the expectation values of $\mathbf{m}_i(t+1)$ and $\mathbf{m}_i(t)$ for $t \to \infty$ must be equal. In other words, in the stationary state we must have

$$\forall i, \ \ \mathrm{E}_t\{h_{c(\mathbf{x}),i}(t)[\mathbf{x}(t) - \mathbf{m}_i^*(t)]\} = 0 \ , \tag{4}$$

where $\mathrm{E}_t\{\cdot\}$ means the expectation value over t.

For simplicity, consider that $h_{c(\mathbf{x}),i}$ can be regarded as time invariant (at the last steps of iteration (1) at least). In the special case where we have a finite number (batch) of the $\mathbf{x}(t)$ with respect to which (4) has to be solved for the $\mathbf{m}_i^*$, we can write (4) as [1]

$$\mathbf{m}_i^* = \frac{\sum_t h_{c(\mathbf{x}),i}\mathbf{x}(t)}{\sum_t h_{c(\mathbf{x}),i}} \ . \tag{5}$$

This is not yet an explicit solution for $\mathbf{m}_i^*$, because the subscript $c(\mathbf{x})$ on the right-hand side still depends on $\mathbf{x}(t)$ *and all the* $\mathbf{m}_i^*$. However, we can solve (5) iteratively. Starting with even coarse approximations for the $\mathbf{m}_i^*$, (2) is first utilized to find the indices $c(\mathbf{x})$ for all the $\mathbf{x}(t)$. On the basis of the approximate $h_{c(\mathbf{x}),i}$ values, the improved approximations for the $\mathbf{m}_i^*$ are computed from (5), which are then applied to (2), whereafter the computed $c(\mathbf{x})$ are substituted to (5), and so on. The optimal solutions $\mathbf{m}_i^*$ are usually obtained in

[1] With a small amount of input data in relation to the map size, it may happen that at some models the denominator of (5) becomes zero. This corresponds to the left side of (4) being zero, so (5) must not be applied for such models at such an iteration step, but the old value of $\mathbf{m}_i^*$ must be retained.

a few iteration cycles, after the discrete-valued indices $c(\mathbf{x})$ have settled down and are no longer changed in further iterations. (The convergence proof of a slightly different Batch Map has been presented in [25].)

In the following we formulate the Batch-Map SOM principle in such a way that it is reduced to two familiar computational steps, namely, that of the classical Vector Quantization [26], [27], [28] and that of smoothing of numerical values over a two-dimensional grid. The particular implementation in this work is based on these steps.

Let V_i be the set of all $\mathbf{x}(t)$ that have $\mathbf{m}_i^*$ as their closest model. This set is called the *Voronoi set.* The number of samples $\mathbf{x}(t)$ falling into V_i is called n_i.

Step 1. Initialize the $\mathbf{m}_i^*$ by any proper method [2].

Step 2. For a finite set $\{\mathbf{x}(t)\}$ of data samples, compute one step of Vector Quantization, where $\bar{\mathbf{x}}_i$ is regarded as the average of the $\mathbf{x}(t)$ over V_i:

$$\forall i \; , \quad \bar{\mathbf{x}}_i = \frac{\sum_{\mathbf{x}(t)\in V_i} \mathbf{x}(t)}{n_i} \tag{6}$$

Step 3. Carry out one smoothing step

$$\mathbf{m}_i^* = \frac{\sum_j n_j h_{ji} \bar{\mathbf{x}}_j}{\sum_j n_j h_{ji}} \; . \tag{7}$$

Alternate steps 2 and 3 until the $\mathbf{m}_i^*$ can be regarded as stationary.

III. Statistical models of documents

In automatic classification, the documents must be described by a set of features. If the purpose were to assign the documents into prescribed classes, the selection of the features could be optimized for maximum classification accuracy (cf. e.g. [30]). The goal in our work, however, was *unsupervised* classification, in which the classes are not known a priori[3]; the documents can only be clustered according to their detailed topical similarities.

It has been demonstrated that the SOM can map free-text natural-language documents, too, if their textual contents are describable by some statistical models such as word

[2]It has been shown that a random choice for the $\mathbf{m}_i^*$ is possible. However, faster convergence is obtained if the initial values of the $\mathbf{m}_i^*$ are even roughly ordered [29].

[3]We do, however, monitor the classification accuracy with respect to the major patent classes (subsections) in order to be able to compare the different algorithms. Such an accuracy, however, is only used as an indirect relative measure for comparing different algorithms.

histograms or their compressed forms [12], [13], [14], [31], [32], [33], [34], [35]. In a series of earlier works we replaced the word histograms by histograms formed over word clusters using "self-organizing semantic maps" [36]. This system was called the "WEBSOM." Its later phases have been explained, e.g., in Honkela et al. [37]. Certain reasons expounded below, however, recently led us to abandon the "semantic maps" and to encode the word histograms by the newly developed projection methods. The results reported in the present article are totally based on these new developments, which has given a reason for us to call the present system the "WEBSOM2." An overview of the WEBSOM2 system is depicted in Fig. 1.

Below we first review some attempts to describe the textual contents of documents statistically.

A. The primitive vector space model

In the basic *vector space model* [38] the stored documents are represented as real vectors in which each component corresponds to the frequency of occurrence of a particular word in the document.

Obviously one should provide the different words with weights that the information can be taken correspond to their significance or power of discrimination between the topics. For the weighting of a word one can use one of the well-known "inverse document frequency" (IDF) -based weighting schemes (IDF is the inverse of the number of documents in which the word occurs). If, however, the documents have some topical classification which contains relevant information, the words can also be weighted according to their Shannon entropy over the set of document classes (see Sec. V-B for a detailed description). The latter method was used in this work in order to utilize the additional information contained in the patent classification.

The weighted word histogram can be viewed as the feature vector describing the document. The main problem of the vector space model is the large vocabulary in any sizable collection of free-text documents, or the vast dimensionality of the model vectors that must be kept in main memory. The size of the vocabulary can, of course, be reduced by automatically or manually selecting a subset containing the most important words according to some criterion. It is, however, a difficult problem to find a suitable subset of words

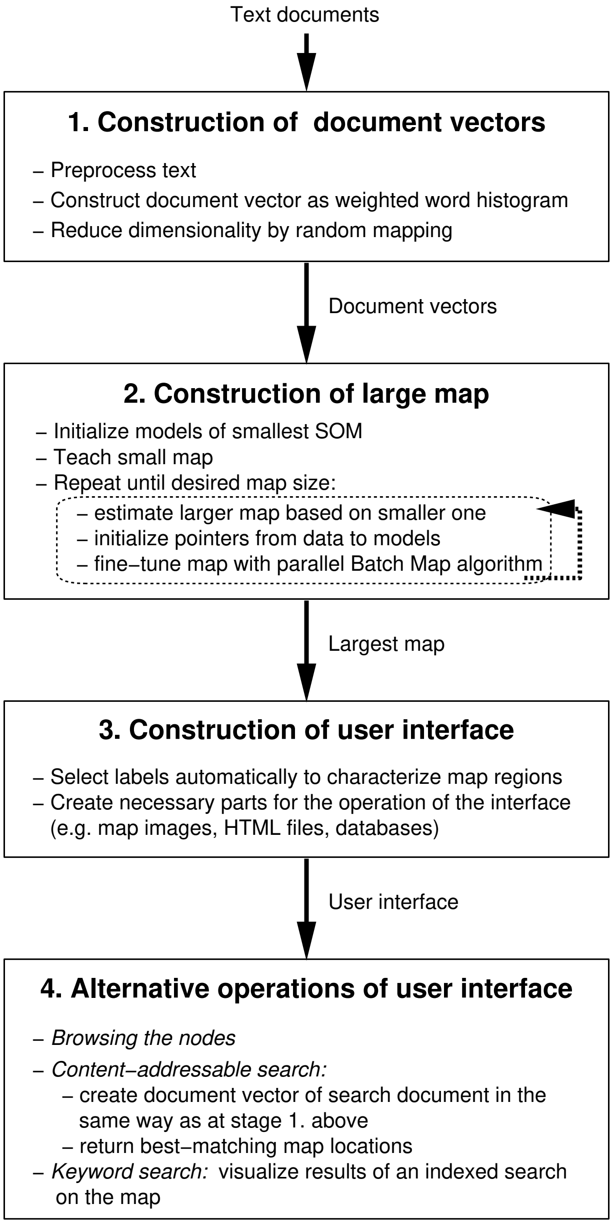

Fig. 1. Overview of the construction and operation of the WEBSOM2 system.

that still represents the essential characteristics of the documents.

B. Latent semantic indexing (LSI)

In an attempt to reduce the dimensionality of the document vectors, without essentially losing information contained in the full vocabulary, one often first forms a matrix in which each column corresponds to the word histogram of a document, and there is one column for each document. After that the space spanned by the column vectors is decomposed into an ordered set of factors by a matrix-computation method called the singular-value decomposition (SVD). The decomposition has the property that the last factors have minimal influence on the matrix. When they are omitted the document vector formed from the histogram of the remaining factors has then a much smaller dimensionality while as much as possible is retained of the original histograms. This method is called *latent semantic indexing (LSI)* [39].

C. Randomly projected histograms

We have shown earlier that the dimensionality of the document vectors can be reduced radically by a much simpler method than the LSI, by the random projection method [20], [40], [41], without essentially losing the power of discrimination between the documents. Experimental results that prove this will be given below in Sec. III-E and in Table I. On the other hand, the computation of the random projections is orders of magnitude lighter than LSI as will be discussed in Sec. III-F[4]. Consider the original document vector (weighted histogram) $\mathbf{n}_i \in \mathbb{R}^n$ and a rectangular random matrix $\mathbf{R}$, the elements in each column of which are assumed as normally distributed vectors having unit length. Let us form the document vectors as the *projections* $\mathbf{x}_i \in \mathbb{R}^m$, where $m \ll n$:

$$\mathbf{x}_i = \mathbf{R}\mathbf{n}_i \ . \tag{8}$$

It can be shown that the pairwise similarity of projection vectors $(\mathbf{x}_i, \mathbf{x}_j)$, measured by the inner products, is the same on the average as the similarity of the corresponding original document vectors $(\mathbf{n}_i, \mathbf{n}_j)$, and the error thereby made is inversely proportional to m [41]. It was demonstrated experimentally [41] for dimensionalities of m exceeding 100 (cf. also Table I) that the classification accuracy is practically as good as with the vector space method, while with the decreasing dimensionality of the document vectors, the time needed to classify a document is decreased radically.

[4]It has recently been suggested that the random projection [42] or similar methods [43] could be used for reducing the computational complexity of the LSI as well.

D. Histograms on the word category map

In our original version of the WEBSOM [20], the reduction of the dimensionality of the document vectors was carried out by letting the words of free natural text be clustered onto neighboring grid points of another special SOM. The input to such a "word category map" [20] consisted of triplets of adjacent words in the text taken over a moving window, whereupon each word in the vocabulary was represented by a unique random vector.

Later we abandoned the word category map since we could eliminate the more or less haphazard process in which the words were categorized, and thus an even better accuracy of document classification was achieved by the straightforward random projection of the word histograms.

E. Validation of the random projection method by small-scale preliminary experiments

Before describing the new encoding of the documents [44] used in this work, some preliminary experimental results that motivate its idea must be presented. Table I compares the three projection methods discussed above in which the model vectors, except in the first case, were always 315-dimensional. For our final implementation we selected the dimensionality of 500 (cf. Sec. V-B) which was still computationally feasible, whereas in our preliminary experiments reported in this Section we still used 315-dimensional vectors for historical reasons, to be able to compare these results with our earlier works where 315-dimensional vectors were also used.

For the material in this smaller-scale preliminary experiment we used 13,742 patents from the whole corpus of 6,840,568 abstracts available to us. The patents were sampled at random but an equal number of patents from each of the 21 subsections of the patent classification system were included.

When the texts were preprocessed as will be explained later in Sec. V-A, the remaining (automatically extracted) vocabulary consisted of 1,814 words or word forms. Document maps consisting of 1,344 units were computed of the document collection, each document was mapped onto one of the grid points of each map, and all documents that represented a minority class at any grid point were counted as classification errors[5].

[5]In this experiment our goal was to organize a given data set into a structure from which the documents can be retrieved easily. Therefore the measured accuracies refer to classification of the original data. Note that this

TABLE I

CLASSIFICATION ACCURACIES OF DOCUMENTS, IN PER CENT, WITH DIFFERENT PROJECTION MATRICES **R**. THE FIGURES ON THE LAST ROW ARE AVERAGES FROM FIVE TEST RUNS WITH DIFFERENT RANDOM ELEMENTS OF THE MATRIX.

	Accuracy	Standard deviation due to different randomization of **R**
Vector space model	60.6	–
LSI	60.4	–
Normally distributed **R**	59.1	0.4

The classification accuracy of 60.6 per cent reported on the first row of Table I refers to a classification that was carried out with the classical vector-space model with full 1344-dimensional histograms as document vectors. In practice, this kind of classification would be orders of magnitude too slow for large text collections.

Dimensionality reduction with LSI resulted in almost the same accuracy as the full histograms. This is reported on the second row. The accuracy obtained with random projection, in which the factorial decomposition of LSI needs not be computed, was still close to the other two.

F. Construction of random projections of word histograms by pointers

Consider now that we want to simplify the projection matrix **R** in order to speed up computations. We can do this by thresholding the matrix elements, or using sparse matrices. Such experiments are reported next in Table II. The following rows have the following meaning: Second row, the originally random matrix elements were thresholded to $+1$ or -1; third row, exactly 5 randomly distributed ones were generated in each column, whereas the other elements were zeros; fourth row, the number of ones was 3; and fifth row, the number of ones was 2, respectively.

These results are now supposed to give us the idea that if we, upon formation of the

task is different from the pattern recognition task in which the goal is to classify new items, and after learning no attention is paid to the training data.

TABLE II

Classification accuracies of documents, in per cent, with different projection matrices $\mathbf{R}$. The figures are averages from five test runs with different random elements of the matrix.

	Accuracy	Standard deviation due to different randomization of $\mathbf{R}$
Normally distributed $\mathbf{R}$	59.1	0.4
Thresholding to $+1$ or -1	59.4	0.2
5 ones in each column	58.2	0.3
3 ones in each column	56.8	0.2
2 ones in each column	55.4	0.3

random projection, would reserve a memory array like an accumulator for the document vector $\mathbf{x}$, another array for the weighted histogram $\mathbf{n}$, and *permanent address pointers* from all the locations of the $\mathbf{n}$ array to all such locations of the $\mathbf{x}$ array for which the matrix element of $\mathbf{R}$ is equal to one, we could form the product very fast by following the pointers and summing up to $\mathbf{x}$ those components of the $\mathbf{n}$ vector that are indicated by the ones of $\mathbf{R}$.

In the method that was actually used we do not project ready histograms, but the pointers are already used with each word in the text in the construction of the low-dimensional document vectors. When scanning the text, the hash address for each word is formed, and if the word resides in the hash table, those elements of the $\mathbf{x}$ array that are found by the (say, five) address pointers stored at the corresponding hash table location are incremented by the weight value of that word. The weighted, randomly projected word histogram obtained in the above way may be optionally normalized.

The computing time needed to form the histograms in the above way in this small-scale experiments was about 20 per cent of that of the usual matrix-product method. This is due to the fact that the histograms, and also their projections, contain plenty of zero elements.

The computational complexity of the random projection with pointers is only $\mathcal{O}(Nl)+$

$\mathcal{O}(n)$, where N is the number of documents, l is the average number of different words in each document, and n is the original dimensionality. Here $\mathcal{O}(n)$ is due to the construction of the hash table and $\mathcal{O}(Nl)$ is the complexity of computing the actual projections, assuming that the hashing operation takes constant time. In contrast, the computational complexity of the LSI is known to be $\mathcal{O}(Nld)$, where d is the resulting dimensionality. Both estimates hold for short texts (sparse histogram vectors).

IV. Rapid construction of large document maps

The SOM algorithm is capable of organizing even a randomly initialized map. However, if the initialization is regular and closer to the final state [29], the asymptotic convergence of the map can be made at least an order of magnitude faster. Below we introduce several speed-up methods by which, first, a reasonable approximation for the initial state is formed and then, the stationary state of the SOM algorithm is reached effectively by a combination of various shortcut methods.

A. Fast distance computation

In word histograms there are plenty of zeros, and if the pointer method of random projection is used, the zeros are still predominant in the projected document vectors.

Since the document vectors are normalized, they can be mapped onto the SOM according to their inner products with the model vectors. Since the zero-valued components of the vectors do not contribute to inner products, it is possible to tabulate the indices of the non-zero components of each input vector, and thereafter consider only those components when computing the distances.

Related, more complex methods have been proposed for computing Euclidean distances between sparse vectors [45]. However, the model vectors must then be stored in the original high-dimensional format, for which we have no memory capacity; we must use low-dimensional models.

B. Estimation of larger maps based on carefully constructed smaller ones

Several suggestions for increasing the number of nodes of the SOM during its construction (cf., e.g. [46]) have been made. The new idea presented below is to *estimate* good initial values for the model vectors of a very large map on the basis of asymptotic values of the model vectors of a much smaller map.

Consider first a rectangular two-dimensional SOM array with two-dimensional input vectors. If the probability density function (pdf) of the input were selected as *uniform* in a rectangular domain and zero outside it, there is a characteristic "shrink" of the distribution of the model vectors with respect to the borders of the support of the pdf, whereas inside the array the model vectors can be assumed as uniformly distributed (cf., e.g., [11], Fig. 3.3 (a)). For an arbitrary number of grid points in the SOM array, rectangular or hexagonal, the amount of this "shrinkage" can easily be estimated.

Consider then that the input has an arbitrary higher dimensionality and an arbitrary pdf, which, however, is continuous and smooth. Even then, the relative "shrinkage" and the relative *local* differences of the new model vectors are similar as in the uniform case (cf., e.g., [11], Fig. 3.7).

Consider again a pdf that is uniform over a two-dimensional rectangular area. This same area is now approximated by either the set of vectors $\{\mathbf{m}'^{(d)}_i \in \mathbb{R}^2\}$, or by $\{\mathbf{m}'^{(s)}_i \in \mathbb{R}^2\}$, where the superscript d refers to the "dense" lattice, and s to the "sparse" lattice, respectively. If the three "sparse" vectors $\mathbf{m}'^{(s)}_i, \mathbf{m}'^{(s)}_j$, and $\mathbf{m}'^{(s)}_k$ do not lie on the same straight line, then in the two-dimensional signal plane any "dense" vector $\mathbf{m}'^{(d)}_h$ can be approximated by the linear expression

$$\mathbf{m}'^{(d)}_h = \alpha_h \mathbf{m}'^{(s)}_i + \beta_h \mathbf{m}'^{(s)}_j + (1 - \alpha_h - \beta_h)\mathbf{m}'^{(s)}_k \;, \tag{9}$$

where α_h and β_h are the interpolation-extrapolation coefficients. This is a two-dimensional vector equation from which the two unknown scalars α_h and β_h can be solved. For illustration of the relations of the codebook vectors, see Fig. 2.

Consider then another, nonuniform but still smooth pdf in a space of arbitrary dimensionality and the two SOM lattices with the same topology but with different density as in the ideal example. When the true pdf is arbitrary, we may not assume the lattices of true codebook vectors to be planar. Nonetheless we can perform a *local linear estimation* of the true codebook vectors $\mathbf{m}^{(d)}_h \in \mathbb{R}^n$ of the "dense" lattice on the basis of the true codebook vectors $\mathbf{m}^{(s)}_i, \mathbf{m}^{(s)}_j$, and $\mathbf{m}^{(s)}_k \in \mathbb{R}^n$ of the "sparse" lattice, using the same interpolation-extrapolation coefficients as in (9).

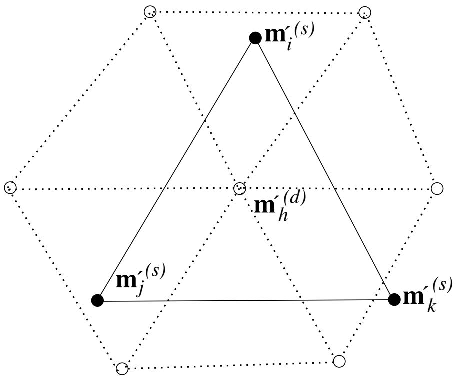

Fig. 2. Illustration of the relations of the model vectors in a sparse (s, solid lines) and dense (d, dashed lines) grid. Only partial grids are shown in the figure. Here $\mathbf{m}'^{(d)}_h$ shall be interpolated in terms of the three closest "sparse" models $\mathbf{m}'^{(s)}_i, \mathbf{m}'^{(s)}_j$, and $\mathbf{m}'^{(s)}_k$, respectively.

In practice, in order that the linear estimate be most accurate, the respective indices h, i, j, and k should be such that $\mathbf{m}'^{(s)}_i$, $\mathbf{m}'^{(s)}_j$, and $\mathbf{m}'^{(s)}_k$ are the three codebook vectors *closest* to $\mathbf{m}'^{(d)}_h$ in the signal space (but not on the same line). With α_h and β_h solved from (9) for each node h separately we obtain the wanted interpolation-extrapolation formula as

$$\hat{\mathbf{m}}^{(d)}_h = \alpha_h \mathbf{m}^{(s)}_i + \beta_h \mathbf{m}^{(s)}_j + (1 - \alpha_h - \beta_h)\mathbf{m}^{(s)}_k \; . \tag{10}$$

Notice that the indices h, i, j, and k refer to *topologically identical* lattice points in (9) and (10). The interpolation-extrapolation coefficients for two-dimensional lattices depend on their topology and the neighborhood function used in the last phase of learning. For best results the "stiffness" of both the "sparse" and the "dense" map should be the same, i.e. the relative width of the final neighborhoods, when referred to the diameter of the array, should be equal.

C. Rapid fine-tuning of the large maps

C.1 Addressing old winners.

Assume that we are somewhere in the middle of the training process, whereupon the SOM is already smoothly ordered although not yet asymptotically stable. Assume that the model vectors are not changed much during one iteration of training. When the same training input is used again some time later, it may be clear that the new winner is found at or in the vicinity of the old one. When the training vectors are then expressed as a linear table, with a *pointer* to the corresponding *old winner location* stored with each training vector, the map unit corresponding to the associated pointer is searched for first, and then a local search for the new winner in the neighborhood around the located unit will suffice (Fig. 3). After the new winner location has been identified, the associated pointer in the input table is replaced by the pointer to the new winner location. This will be a significantly faster operation than an exhaustive winner search over the whole SOM. The search can first be made in the immediate surrounding of the said location, and only if the best match is found at its edge, searching is continued in the surrounding of the preliminary best match, until the winner is one of the middle units in the search domain.

In order to ensure that the matches are globally best, a full search for the winner over the whole SOM can be performed intermittently.

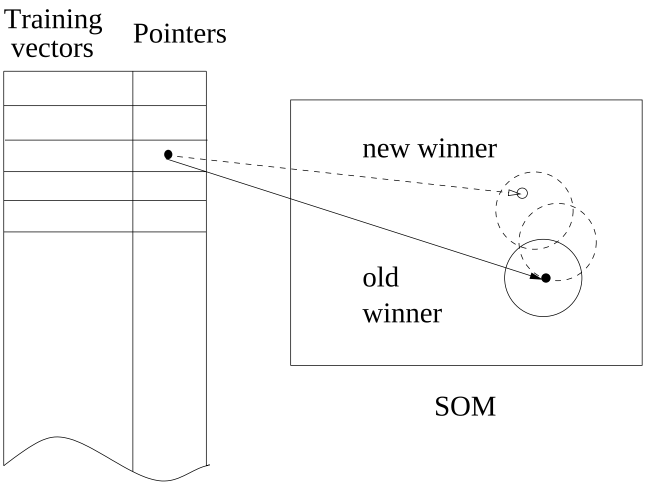

Fig. 3. Finding the new winner in the vicinity of the old one, whereby the old winner is directly located by a pointer. The pointer is then updated.

Koikkalainen [47], [48] has suggested a similar speedup method for a search-tree structure.

C.2 Initialization of the pointers

When the size (number of grid nodes) of the maps is increased stepwise during learning using the estimation procedure discussed in Section IV-B, the initial pointers for all data vectors after each increase can be estimated quickly by utilizing the formula that was used in increasing the map size, equation (10). The winner is the map unit for which the inner product with the data vector is the largest, and so the inner products can be computed rapidly using the expression

$$\mathbf{x}^T\mathbf{m}_h^{(d)} = \alpha_h \mathbf{x}^T\mathbf{m}_i^{(s)} + \beta_h \mathbf{x}^T\mathbf{m}_j^{(s)} + (1-\alpha_h-\beta_h)\mathbf{x}^T\mathbf{m}_k^{(s)} \; . \tag{11}$$

Here d refers to model vectors of the large map and s of the sparse map, respectively. Expression (11) can be interpreted as the inner product between two *three-dimensional* vectors, $[\alpha_h; \beta_h; (1-\alpha_h-\beta_h)]^T$ and $[\mathbf{x}^T\mathbf{m}_i^{(s)}; \mathbf{x}^T\mathbf{m}_j^{(s)}; \mathbf{x}^T\mathbf{m}_k^{(s)}]^T$, *irrespective of the dimensionality of* $\mathbf{x}$. If necessary, the winner search can still be speeded up by restricting the winner search to the area of the dense map that corresponds to the neighborhood of the winner on the sparse map. This is especially fast if only a subset (albeit a subset that covers the whole map) of all the possible triplets (i, j, k) is allowed in (10) and (11).

C.3 Parallelized Batch Map algorithm

The Batch Map algorithm introduced in Section II facilitates a very efficient parallel implementation. At each iteration we first compute the pointer $c(t)$ to the best-matching unit for each input $\mathbf{x}(t)$. If the old value of the pointer can be assumed as being close to the final value, as is the case if the pointer has been initialized properly or obtained in the previous iteration of a relatively well-organized map, we need not perform an exhaustive winner search as discussed above. Moreover, since the model vectors do not change at this stage, the winner search can be easily implemented in parallel by dividing the data into the different processors in a shared-memory computer.

After the pointers have been computed, the previous values of the model vectors are not needed any longer. The means $\bar{\mathbf{x}}_j$ as defined by (6) can be computed as recursive expressions at nodes defined by the pointers $c(t)$ associated with the $\mathbf{x}(t)$, and therefore extra memory is not needed to keep the old values of the $\mathbf{m}_i^*$ when computing their new values.

Finally, the new values of the model vectors can be computed based on (7). This computation can also be implemented in parallel and done within the memory reserved for the model vectors if a subset of the new values of the model vectors is held in a suitably defined buffer.

C.4 Saving memory by reducing representation accuracy

The memory requirements can be reduced significantly by using a coarser quantization of the vectors. We have used a common adaptive scale for all of the components of a model vector, representing each component with eight bits only. If the dimensionality of the data vectors is large, the statistical accuracy of the distance computations is still sufficient as shown in earlier studies [49]. The sufficient accuracy can be maintained during the computation if a suitable amount of noise is added to each new value of a model vector before quantizing it.

D. Performance evaluation of the new methods

D.1 Numerical comparison with the traditional SOM algorithm

In this section we have introduced several methods for speeding up the computation of large SOMs. We will next verify that the quality of the resulting maps is comparable to the maps constructed with the traditional SOM algorithm.

The smaller-scale tests were carried out on the same collection of 13,742 patent abstracts that was used already in Sec. III-E. We shall use two performance indices to measure the quality of the maps: the average distance of each input from the closest model vector called the *average quantization error*, and the separability of different classes of patents on the resulting map called the *classification accuracy*. The classes were the 21 subsections of the patent classification system.

We computed two sets of maps, one with the traditional SOM algorithm, and the other using the new methods, respectively, and compared their quality. In computing both sets we used parameter values that in preliminary experiments had been found to guarantee good results.

The model vectors of the maps in the first, traditionally computed set were initialized by values spaced evenly on the subspace spanned by the two dominant eigenvectors of the data set [29]. The map was then computed using the SOM algorithm, eqns (2) and (1). The total number of iterations was about 150 per map unit, and both the width and the height of the neighborhood kernel decreased more rapidly at first, and more slowly towards the end of learning.

For the second set of maps, small maps consisting of 84 units were first computed with the SOM algorithm, again using about 150 iterations per map unit. The final large maps were then estimated based on these small ones, the pointers to the winning units from each input sample were initialized, and 5 iterations of the Batch Map algorithm were carried out.

As can be seen from Table III, the quality of the resulting maps is comparable, but the time needed for the shortcut methods is only about one tenth of that of the traditional algorithm. The time has been measured with a SGI O2000 computer without parallelization of any programs.

TABLE III

Comparison of the new shortcut methods with the traditional SOM algorithm. The figures are averages from five test runs with different random matrices used in the encoding of the documents, and the error margins are standard deviations.

	Classification accuracy (%)	Quantization error	Time (s)
Traditional SOM	58.2 ± 0.2	0.799 ± 0.001	2550 ± 40
Shortcut methods	58.0 ± 0.2	0.798 ± 0.002	241 ± 3.5

D.2 Comparison of the computational complexity

For very large maps the difference in the computation times is even more marked than in Table III, but can only be deduced from the computational complexities given in Table IV.

The complexity of computation of the traditional SOM algorithm is $\mathcal{O}(dN^2)$, since the complexity of each full winner search is $\mathcal{O}(dN)$, and the number of iterations should be a multiple of the number of map units to guarantee sufficient statistical accuracy of the resulting map.

In the complexity of the new method, the first term, $\mathcal{O}(dM^2)$, stems from the computation of the small map. The second term, $\mathcal{O}(dN)$, results from the VQ step (eqn. 6) of the Batch Map algorithm, in which the winners are sought only in the vicinity of the old winners as described in Sec. IV-C.1. Here it has been assumed that a search in a neighborhood having a size independent of N is sufficient. The last term in the computational complexity, $\mathcal{O}(N^2)$, refers to the estimation of the pointers, cf. Sec. IV-C.2, and the smoothing step, eqn (7), of the Batch Map computation. The initialization of the pointers can actually be carried out in $\mathcal{O}(N^2/M)$ time, since about N/M units of the larger map need to be searched for each input. In the smoothing step an average over the neighbors of each map unit is computed, and if it is desired to keep the "stiffness" of the map approximately constant when the number of map units is increased, the size of the neighborhood should always be the same fraction of the number of map units.

It may thus be estimated, taking into account the speedup obtained already in the random projection, that the total speedup factor in construction of large maps is of the order of the dimensionality of the original input vectors, which in our largest experiment was about 50,000[6]

TABLE IV

COMPUTATIONAL COMPLEXITY OF THE METHODS. HERE N DENOTES THE NUMBER OF DATA SAMPLES, M THE NUMBER OF MAP UNITS IN THE SMALL MAP, AND d THE DIMENSIONALITY OF THE INPUT VECTORS. IT HAS BEEN ASSUMED THAT THE NUMBER OF MAP UNITS IN THE FINAL MAP IS CHOSEN TO BE PROPORTIONAL TO THE NUMBER OF DATA SAMPLES.

	Computational complexity
Traditional SOM	$\mathcal{O}(dN^2)$
Shortcut methods	$\mathcal{O}(dM^2) + \mathcal{O}(dN) + \mathcal{O}(N^2)$

V. THE DOCUMENT MAP OF ALL ELECTRONIC PATENT ABSTRACTS

For the largest WEBSOM map made so far we selected a data base of 6,840,568 patent abstracts available in electronic form and written in English. These patents were granted by the U.S., European, and Japan patent offices and stored in two databases: the "First Page" database (1970-1997), and "Patent Abstracts of Japan" (1976-1997). The average length of each text was 132 words. The size of the SOM was 1,002,240 models (neurons).

[6]In Sec. V we computed the largest map in several stages. The complexity of each stage is, however, only $\mathcal{O}(dN) + \mathcal{O}(N^2)$.

A. Preprocessing

From the raw patent abstracts we first extracted the titles and the texts for further processing. We then removed non-textual information. Mathematical symbols and numbers were converted into special "dummy" symbols. The whole vocabulary contained 733,179 different words (base forms). All words were converted to their base form using a stemmer [50]. The words occurring less than 50 times in the whole corpus, as well as a set of common words in a stopword list of 1,335 words were removed. The remaining vocabulary consisted of 43,222 words. Finally, we omitted the 122,524 abstracts in which less than 5 words remained.

B. Formation of statistical models

To reduce the dimensionality of 43,222 of the histograms we used the random projection method (Sec. III-C). For the final dimensionality we selected 500, and 5 random pointers were used for each word (in the columns of the projection matrix $\mathbf{R}$). The words were weighted using the Shannon entropy of their distribution of occurrence among the subsections of the patent classification system. There are 21 subsections in the patent classification system in total; examples of such subsections are agriculture, transportation, chemistry, building, engines, and electricity (cf. Fig. 4).

The weight is a measure of the unevenness of the distribution of the word in the subsections. The weights were calculated as follows: Let $P_g(w)$ be the probability of a randomly chosen instance of the word w occurring in subsection g, and N_g the number of subsections. The Shannon entropy thus becomes $H(w) = -\sum_g P_g(w) \log P_g(w)$, and the weight $W(w)$ of word w is defined to be $W(w) = H_{\max} - H(w)$, where $H_{\max} = \log N_g$.

C. Formation of the document map

The final map was constructed in four successively enlarged stages, at all of which the same 500-dimensional document vectors were used as input. The map was increased twice sixteen-fold and once nine-fold. The smallest, 435-unit map was constructed using the original SOM algorithm and 300 000 learning steps. Each of the enlarged, estimated maps (cf. Sec. IV-B) was then fine-tuned by five Batch Map iteration cycles. In order that the asymptotic form of the map would be smooth and regular enough, we had to use the final neighborhood size where the radius was nine grid spacings.

Several choices for the parameter values (map sizes, training lengths etc.) during the training process had to be made. These were based on earlier experiences and experiments made using smaller subsets of the same document collection. We also monitored the classification accuracy (cf. Sec. IV-D) after each stage of computation: several trials and variations for the smaller maps were made, whereas the fine-tuning of the largest map could be carried out only once, because it took several weeks. The final classification accuracy was compatible with the results obtained with the smaller maps, whereas the fine structures of clustering manifested themselves best in the largest map.

With the newest versions of our programs the whole process of computation of the document map takes about six weeks on a six-processor SGI O2000 computer. At the moment we cannot provide exact values of the real processing time since we have all the time developed the programs while carrying out the computations.

The amount of main memory required was about 800MB.

Forming the user interface automatically took an additional week of computation. This time includes finding the keywords to label the map, forming the WWW-pages that are used in exploring the map, and indexing the map units for keyword searches.

D. Results

In order to get an idea of the quality of the organization of the final map we measured how the different subsections of the patent classification system were separated on the map. When each map node was labeled according to the majority of the subsections in the node and the abstracts belonging to the other subsections were considered as misclassifications, the resulting "accuracy" (actually, the "purity" of the nodes) was **64%**. It should be noted that the subsections overlap partially—the same patent may have subclasses which belong to different subsections. The result corresponded well with the accuracies we have obtained in several different runs with smaller maps computed on subsets of the same document collection. The distribution of patents on the final map has been visualized in Fig. 4.

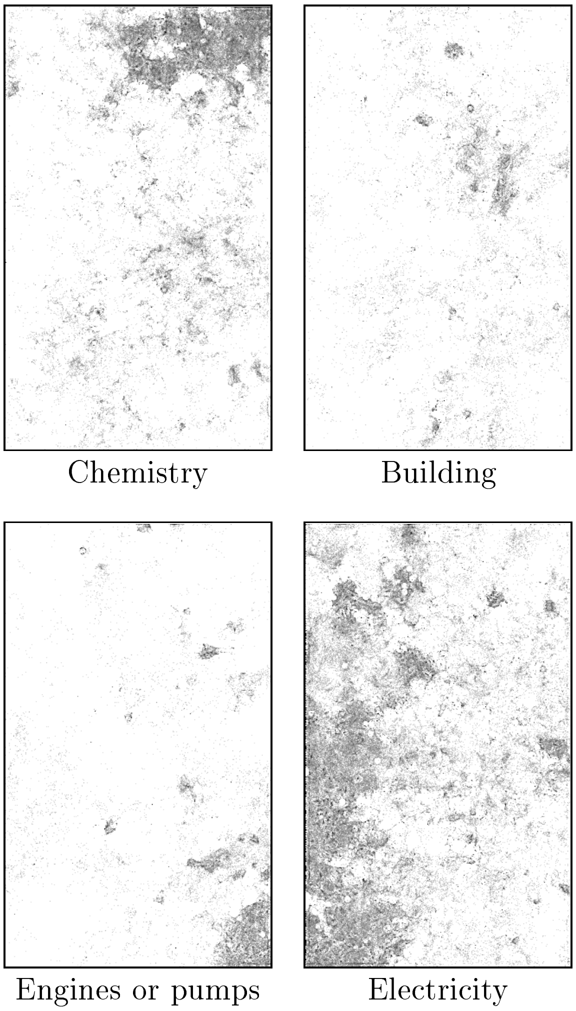

Fig. 4. Distribution of four sample subsections of the patent classification system on the document map. The gray level indicates the logarithm of the number of patents in each node.

E. Exploration of the document map

The document map is presented to the user as a series of HTML pages that enable the exploration of the map: when clicking a point on the map display with a mouse, links to the document database enable reading the contents of the documents. If the map is large, subsets of it can first be viewed by zooming. With the largest maps we have used three zooming levels before reaching the documents. To provide guidance to the exploration, an automatic method has been utilized for selecting keywords to characterize map regions [51]. These keywords, to be regarded only as some kind of *landmarks* on the map display, serve as navigation cues during the exploration of the map, as well as provide information on the topics discussed in the documents on the respective map area.

E.1 Content addressable search: example

The interface to the map has been provided with a form field into which the user can type a query, or a description of interest, in the form of a short "document." This query is preprocessed and a document vector is formed in the exactly same manner as for the stored documents prior to construction of the map. The resulting vector is then compared with the model vectors of all map units, and the best-matching points are marked with circles on the map display: the better the match, the larger the circle. These locations provide good starting points for browsing. An example of utilizing the content-addressable search on the map of seven million patent abstracts is shown in Fig. 5. With the map of all patent abstracts performing the search takes only a few seconds in total.

E.2 Keyword search: example

A more conventional *keyword search mode* has also been provided for finding good starting-points for browsing. After building the map, for each word we indexed the map units that contain the word. Given a search description, the matching units are found from the index and the best matches are returned and displayed as circles on the map, all within a few seconds of starting the search. An example of performing a keyword search is depicted in Fig. 6. The example shows an advantage of the visual map over more traditional searching. When performing a search, the best matches often contain different kinds of material, or different aspects that may be relevant to the query. In a more traditional search engine these matches might be returned as a list organized by relevance. On the map the relevance information can be portrayed as the size of the circle or other symbol marking a match, and, furthermore, the different aspects of the topic may be found within different clusters or areas of the map. If the user is already familiar with the map, the display may immediately help in selecting the most interesting subset of matches.

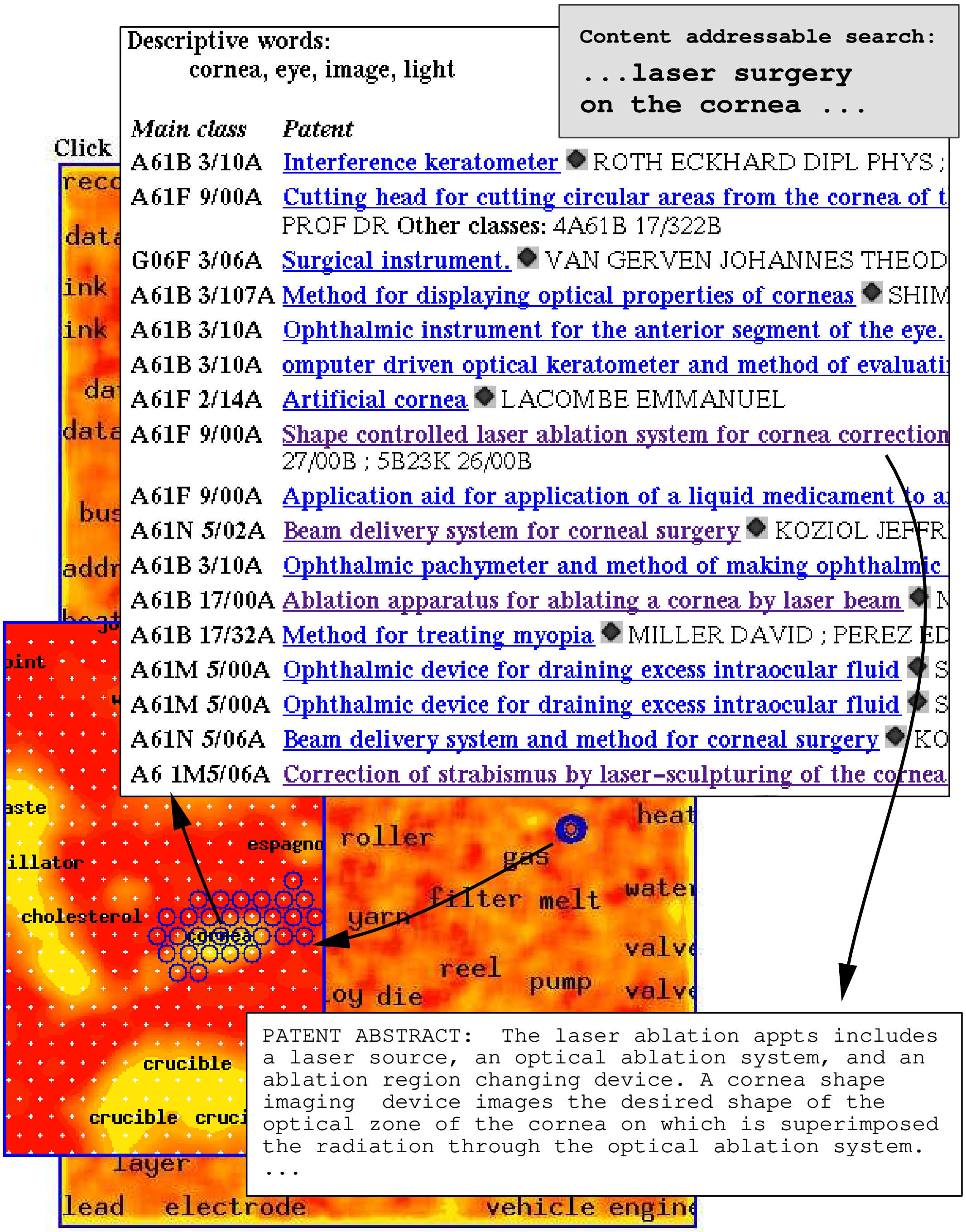

Fig. 5. Content addressable search was utilized to find information on laser surgery on the cornea of the eye. The best-matching locations are marked with circles. Zooming on the area reveals a small cluster of map units that contains patent abstracts mostly about the cornea of the eye, and of surgical operations on it. Several abstracts concerned with the description of interest, i.e. laser surgery on the cornea, are found in the best-matching units.

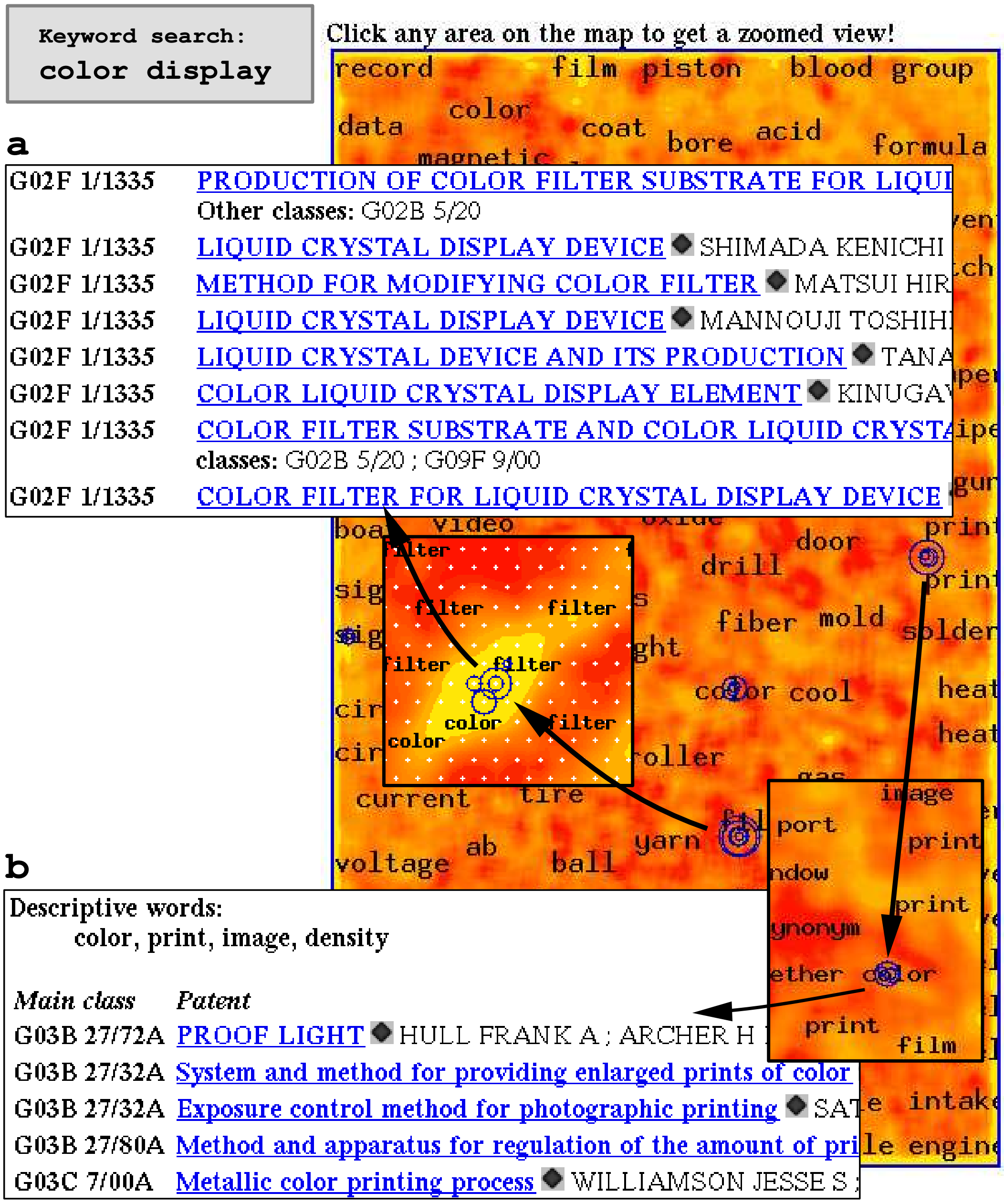

Fig. 6. The keyword search mode was utilized to find information on color displays. 30 best-matching units were marked on the display with circles the size of which indicates the goodness of the match. As seen from the map display, the matches are distributed into several tight "clusters" found in different regions of the map. From two of these clusters the partial contents of a sample matching unit are shown in the insets. Closer inspection of the units reveals different aspects of color and displays. Unit **a** features a considerable number of abstracts about color filters used in building LCD displays, whereas in **b** one finds technology related to displaying colors when printing documents (the "Descriptive words" lists were found for each map unit using the automatic keyword selection method introduced in [51]). The user who probably did not have printing in mind when formulating the query can then concentrate on the other clusters.

VI. Conclusions

It has been demonstrated that similarity graphs of very large free-form (even defective) collections of English texts can be produced by contemporary computers.

The similarity graphs appear to be especially suitable for interactive data mining or exploration tasks in which the user either does not know the domain very well or has only a vague idea of the contents of the full-text database being examined.

In this paper the emphasis has been on the up-scalability of the methods relating to very large text collections. The novel contributions of the present article are: 1. A new application that is an order of magnitude larger than our previous one, 2. a new method of forming statistical models of documents, and 3. several new fast computing methods ("shortcuts"). In the present experiments the computational complexity was reduced by a factor of over four orders of magnitude compared with the straightforward solution.

Acknowledgements

We wish to thank the European Patent Office and the National Board of Patents and Registration of Finland for their help with the patent collection, and the Academy of Finland for financial support.

References

[1] J. W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, Reading, MA, 1977.

[2] G. Young and A. S. Householder, "Discussion of a set of points in terms of their mutual distances," *Psychometrica*, vol. 3, pp. 19–22, 1938.

[3] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrica*, vol. 17, pp. 401–419, 1952.

[4] J. B. Kruskal and M. Wish, "Multidimensional scaling," Tech. Rep. 07-011, Sage University Paper Series on Quantitive Applications in the Social Sciences, Newbury Park, CA, 1978.

[5] J. de Leeuw and W. Heiser, "Theory of multidimensional scaling," in *Handbook of Statistics*, P. R. Krishnaiah and L. N. Kanal, Eds., vol. 2, pp. 285–316. North-Holland, Amsterdam, 1982.

[6] M. Wish and J. D. Carrol, "Multidimensional scaling and its applications," in *Handbook of Statistics*, P. R. Krishnaiah and L. N. Kanal, Eds., vol. 2, pp. 317–345. North-Holland, Amsterdam, 1982.

[7] F. W. Young, "Multidimensional scaling," in *Encyclopedia of Statistical Sciences*, S. Kotz, N. L. Johnson, and C. B. Reads, Eds., vol. 5, pp. 649–659. Wiley, New York, 1985.

[8] J. W. Sammon Jr., "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, pp. 401–409, 1969.

[9] T. Kohonen, "Self-organizing formation of topologically correct feature maps," *Biol. Cyb.*, vol. 43, no. 1, pp. 59–69, 1982.

[10] T. Kohonen, "Clustering, taxonomy, and topological maps of patterns," in *Sixth Int. Conf. on Pattern Recognition, Munich, Germany, October 19-22, 1982*, 1982, pp. 114–128.

[11] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, 1995, (Second, extended edition 1997).

[12] X. Lin, D. Soergel, and G. Marchionini, "A self-organizing semantic map for information retrieval," in *Proc. 14th. Ann. Int. ACM/SIGIR Conf. on R & D In Information Retrieval*, 1991, pp. 262–269.

[13] J. C. Scholtes, "Unsupervised Learning and the Information Retrieval Problem," in *Proc. IJCNN'91, Int. Joint Conf. on Neural Networks*, Singapore, 1991, vol. I, pp. 95–100, IEEE Service Center, Piscataway, NJ.

[14] D. Merkl and A. M. Tjoa, "The representation of semantic similarity between documents by using maps: Application of an artificial neural network to organize software libraries," in *Proc. FID'94, General Assembly Conf. and Congress of the Int. Federation for Information and Documentation*, 1994.

[15] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, "Newsgroup exploration with WEBSOM method and browsing interface," Tech. Rep. A32, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[16] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, "Creating an order in digital libraries with self-organizing maps," in *Proc. WCNN'96, World Congress on Neural Networks, September 15-18, San Diego, California*, pp. 814–817. Lawrence Erlbaum and INNS Press, Mahwah, NJ, 1996.

[17] T. Kohonen, S. Kaski, K. Lagus, and T. Honkela, "Very large two-level SOM for the browsing of newsgroups," in *Proc. ICANN96, Int. Conf. on Artificial Neural Networks, Bochum, Germany, July 16-19, 1996*, C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, Eds., Lecture Notes in Computer Science, vol. 1112, pp. 269–274. Springer, Berlin, 1996.

[18] K. Lagus, T. Honkela, S. Kaski, and T. Kohonen, "Self-organizing maps of document collections: A new approach to interactive exploration," in *Proc. of the Second Int. Conf. on Knowledge Discovery and Data Mining*, E. Simoudis, J. Han, and U. Fayyad, Eds., pp. 238–243. AAAI Press, Menlo Park, California, 1996.

[19] T. Kohonen, "Exploration of very large databases by self-organizing maps," in *Proc. of ICNN'97, Int. Conf. on Neural Networks*, pp. PL1–PL6. IEEE Service Center, Piscataway, NJ, 1997.

[20] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, "WEBSOM—self-organizing maps of document collections," *Neurocomputing*, vol. 21, pp. 101–117, 1998.

[21] K. Lagus, T. Honkela, S. Kaski, and T. Kohonen, "WEBSOM for textual data mining," *Artificial Intelligence Review*, 1999, In press.

[22] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela, "Self organization of a massive text document collection," in *Kohonen Maps*, E. Oja and S. Kaski, Eds., pp. 171–182. Elsevier, Amsterdam, 1999.

[23] T. Kohonen, "Comparison of SOM point densities based on different criteria," *Neural Computation*, vol. 11, no. 8, pp. 2171–2185, 1999.

[24] T. Kohonen, "New developments of Learning vector Quantization and the Self-Organizing map," in *Symp. on Neural Networks; Alliances and Perspectives in Senri*, Osaka, Japan, 1992, Senri Int. Information Institute.

[25] Y. Cheng, "Convergence and ordering of Kohonen's batch map," *Neural Computation*, vol. 9, no. 8, pp. 1667–76, 1997.

[26] A. Gersho, "Asymptotically optimal block quantization," *IEEE Transactions on Information Theory*, vol. 25, pp. 373–380, 1979.

[27] R. M. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4–29, April 1984.

[28] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proceedings of the IEEE*, vol. 73, pp. 1551–1588, 1985.

[29] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, "SOM_PAK: The Self-Organizing Map program package," Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Jan. 1996.

[30] D. Koller and M. Sahami, "Toward optimal feature selection," in *Machine Learning: Proc. of the Thirteenth Int. Conf. (ICML'96)*, L. Saitta, Ed. 1996, pp. 284–292, Morgan Kaufmann.

[31] H. Chen, C. Schuffels, and R. Orwig, "Internet categorization and search: a self-organizing approach," *Journal of Visual Communication and Image Representation*, vol. 7, no. 1, pp. 88–102, 1996.

[32] S. Lesteven, P. Ponçot, and F. Murtagh, "Neural networks and information extraction in astronomical information retrieval," *Vistas in Astronomy*, vol. 40, pp. 395, 1996.

[33] X. Lin, "Map displays for information retrieval," *Journal of the American Society for Information Science*, vol. 48, pp. 40–54, 1997.

[34] D. Merkl, "Text classification with self-organizing maps: Some lessons learned," *Neurocomputing*, vol. 21, pp. 61–77, 1998.

[35] H. Chen, J. Nunamaker Jr., R. Orwig, and O. Titkova, "Information visualization for collaborative computing," *IEEE Computer*, pp. 75–82, August 1998.

[36] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biol. Cyb.*, vol. 61, no. 4, pp. 241–254, 1989.

[37] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, "WEBSOM—self-organizing maps of document collections," in *Proc. of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pp. 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[38] G. Salton and M. J. McGill, *Introduction to modern information retrieval*, McGraw-Hill, New York, 1983.

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

[40] S. Kaski, "Data exploration using self-organizing maps," *Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82*, March 1997, D.Sc. (Tech) Thesis, Helsinki University of Technology, Finland.

[41] S. Kaski, "Dimensionality reduction by random mapping: Fast similarity computation for clustering," in *Proc. of IJCNN'98, Int. Joint Conf. on Neural Networks*, vol. 1, pp. 413–418. IEEE Service Center, Piscataway, NJ, 1998.

[42] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, "Clustering in large graphs and matrices," in *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA*, pp. 291–299. ACM, 1999.

[43] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proc. of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems June 1 - 4, 1998, Seattle, WA*, pp. 159–168. ACM Press, 1998.

[44] T. Kohonen, "Self-organization of very large document collections: State of the art," in *Proc. ICANN98, the 8th Int. Conf. on Artificial Neural Networks*, L. Niklasson, M. Bodén, and T. Ziemke, Eds., vol. 1, pp. 65–74. Springer, London, 1998.

[45] D. Roussinov and H. Chen, "A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation," *CC-AI—Communication, Cognition and Artificial Intelligence*, vol. 15, no. 1-2, pp. 81–111, 1998.

[46] J. S. Rodrigues and L. B. Almeida, "Improving the learning speed in topological maps of patterns," in *Proc. INNC'90, Int. Neural Networks Conference*, Dordrecht, Netherlands, 1990, pp. 813–816, Kluwer.

[47] P. Koikkalainen, "Progress with the tree-structured self-organizing map," in *Proc. ECAI'94, 11th European Conf. on Artificial Intelligence*, A. G. Cohn, Ed., New York, 1994, pp. 211–215, John Wiley & Sons.

[48] P. Koikkalainen, "Fast deterministic self-organizing maps," in *Proc. ICANN'95, Int. Conf. on Artificial Neural Networks*, F. Fogelman-Soulié and P. Gallinari, Eds., Nanterre, France, 1995, vol. II, pp. 63–68, EC2.

[49] T. Kohonen, "Things you haven't heard about the Self-Organizing Map," in *Proc. ICNN'93, Int. Conf. on Neural Networks*, 1993, pp. 1147–1156, IEEE Service Center, Piscataway, NJ.

[50] K. Koskenniemi, *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*, Ph.D. thesis, University of Helsinki, Department of General Linguistics, 1983.

[51] K. Lagus and S. Kaski, "Keyword selection method for characterizing text document maps," in *Proc. ICANN99, Ninth Int. Conf. on Artificial Neural Networks*, vol. 1, pp. 371–376. IEE Press, London, 1999.

Dr. Teuvo Kohonen is Professor of the Academy of Finland, and he is affiliated with the Helsinki University of Technology. He was the founding President of the European Neural Network Society. His most recent research area is self-organization, in which he has introduced the widely known unsupervised learning algorithm called the Self-Organizing Map. He is author of five textbooks or monographs, of which "Self-Organizing Maps" (Springer, 1995, 2nd ed. 1997) is the most recent one, and co-author of several edited books.

Samuel Kaski received his D.Sc. (Tech) degree in Computer Science from Helsinki University of Technology, Finland, in 1997. He is currently Professor of Computer Science at the Laboratory of Computer and Information Science (Neural Networks Research Centre), Helsinki University of Technology. His main research areas are neural computation and data mining.

Krista Lagus received her M.Sc. degree in Computer Science from Helsinki University of Technology, Finland, in 1996. She is a research associate at the Neural Networks Research Centre, Helsinki University of Technology since 1995. Her main research interests are related to neural networks, especially Self-Organizing Maps, and their application to natural language processing and data mining.

Jarkko Salojärvi received his M.Sc. degree in Technical Physics from Helsinki University of Technology in 1998. His main research interests are related to neural networks, the emphasis being on self-organizing maps and their application to data mining. Mr. Salojärvi is a research associate at the Neural Networks Research Centre, Helsinki University of Technology since 1997.

Jukka Honkela is an undergraduate student at the University of Oulu, Finland, where he studies computer engineering. The research was done while he was a research assistant at the Neural Networks Research Centre, Helsinki University of Technology. Currently he works for GuruSoft Oy utilizing Self-Organizing Maps in textual data mining.

Vesa Paatero is an undergraduate student at Helsinki University of Technology, where he studies information technology. He is working as a research assistant at the Neural Networks Research Centre, Helsinki University of Technology.

Antti Saarela is an ungergraduate student at Helsinki University of Technology, where he studies information technology. He is currently doing his Master's Thesis at ABB Industry Oy using neural nets and image analysis for paper defect classification.