

# Exploration of Very Large Databases by Self-Organizing Maps

Teuvo Kohonen

Helsinki University of Technology

Neural Networks Research Centre

Rakentajanaukio 2 C, FIN-02150 Espoo, Finland

## ABSTRACT

This paper describes a data organization system and genuine content-addressable memory called the WEBSOM. It is a two-layer self-organizing map (SOM) architecture where documents become mapped as points on the upper map, in a geometric order that describes the similarity of their contents. By standard browsing tools one can select from the map subsets of documents that are most similar mutually. It is also possible to submit free-form queries about the wanted documents whereby the WEBSOM locates the best-matching documents. The document map exemplified in this paper has over 100,000 map nodes, with 315 inputs at each, and over 1,000,000 documents have been organized by it. The system has been implemented by software on a general-purpose computer.

## 1. Introduction

In the following we shall demonstrate *exploration of textual data bases*. The words in full free text always occur in the *context* of other words. It has turned out experimentally that the *crude conceptual meaning* of most words already transpires, statistically at least, from the *usage of the word in the frame of the closest words*. Many years ago [1] we performed experiments, which showed convincingly that the sequential dependencies within *triples* of successive words are already strong enough for an approximative conceptual clustering of the words. Such clustering has been found effective for the construction of invariant “signatures” or “fingerprints” of documents for their identification or statistical classification.

In order to describe the contextual usage of a word in as pure form as possible, the words can be encoded or represented by *vectors consisting of random-valued components*; we have found that a vector dimensionality of the order of ninety is suitable for such encoding purposes. A triple is then equivalent with a real vector  $x \in \mathcal{R}^{270}$ , and such *contextual pattern vectors* can be input to and analyzed by the neural-network architecture described here.

## 2. Word category histograms

We shall show in this section that if the contextual pattern vectors  $x$ , also named briefly *context vectors* that represent the successive word triples are clustered, *the distribution of such clusters of a document serves as a very good “fingerprint”* by which the document can be identified or classified. Computation and comparison of the distributions is not quite straightforward, however. One of the main problems is the vast number of possible word triples. Some kind of dimensionality reduction of their representation is necessary.

A new and effective framework for the statistical representation of the context vectors is the *self-organizing map (SOM)* [2, 3], which partitions the  $x$  space into a wanted finite set of “cells,” the so-called *Voronoi tessellation*.

If the SOM has now been computed for some data base and we take a document from the same data base (or a new document), we can map all its context vectors  $x$  onto the same SOM, whereby the number of hits in the various cells taken together can be regarded as a *histogram* of the document, formed over the cells or “bins” of the Voronoi tessellation. This histogram is a very effective reduced statistical representation

of the document. As there are thematic variations between the documents, the histograms can be used for the identification or classification of the latter. The method is usually optimized for one textual data base at a time; for a different data base the classification accuracy is expected to remain suboptimal.

*An important finding [1] is that the context vectors which have one or more similar members in respective positions of the triple will be mapped close to each other on the SOM.* This, of course, is naturally explained by the fact that triples in which all code vectors are different have a larger mutual distance in the vector space than triples which have similar code vectors in their respective positions, and similar items are in general mapped close to each other on the SOM.

This effect becomes even more explicit and enhanced if we compute the *conditional average context of each word* over the text corpus. Consider the successive code vectors (words)  $x_{i-1}$ ,  $x_i$ , and  $x_{i+1}$ , and form the *average context vector*  $X(i)$  of word  $x_i$

$$X(i) = \begin{bmatrix} \mathbf{E}\{x_{i-1}|x_i\} \\ \varepsilon x_i \\ \mathbf{E}\{x_{i+1}|x_i\} \end{bmatrix}, \quad (1)$$

where  $\mathbf{E}$  is the conditional average over the text corpus, and  $\varepsilon$  is a small number. The purpose of  $\varepsilon$  is to control the relative weight of the averages in determining the similarity of contexts. Now we only need to form  $X(i)$  once for each word of the vocabulary. The SOM is trained by the  $X(i)$  and when the latter, after training, are input once again, *the nodes (map units) are provided with labels of all those  $x_i$ , the  $X(i)$  of which have selected the corresponding nodes for winners.*

It may be obvious that *if two or more words  $x_i$  have been used frequently in similar contexts, even when those contexts are different from case to case, their average context vectors are mutually similar and their labels will be mapped close to each other on the SOM, maybe even on the same node.* It has been found that this happens if the words are synonyms, if they represent different values of the same attribute, or if they are otherwise semantically so similar that they can be regarded to form a category, roughly at least.

When a document picked up from the data base for which the SOM has been computed, or a new document shall be analyzed, one could input all its successive (overlapping) context vectors  $[x_{i-1}^T, x_i^T, x_{i+1}^T]^T$  to the SOM, find the corresponding winners, and form the histogram of the hits on the SOM. However, it has turned out almost equally accurate but computationally much more effective to use input vectors of the form  $[\emptyset, x_i^T, \emptyset]^T$  where  $\emptyset$  means "don't care." Since the map units of the SOM were already labeled according to the  $x_i$ , it is equivalent but much faster to locate the labels on the map and to form the histograms by a fast associative software method called the *hash coding* [3, 4].

### 3. The WEBSOM architecture

#### 3.1. The general scheme

An important benefit of the above encoding scheme of documents is that, e.g., the histograms formed on the SOM are to a great extent *invariant to the selection of alternative synonyms*. For instance, if we want to search for documents that match best with a short manually given query, in principle there need not even exist identical words in the query and the documents, as long as the texts are semantically similar (described by similar synonyms)!

The complete architecture for the encoding and searching of documents is shown in Fig. 1 and called the *WEBSOM*. It consists of *two layers*, of which the lower one is a SOM that forms the histograms, and the second one is another SOM that studies the histogram of a new document and maps it into a point on the upper SOM.

In addition the WEBSOM system contains programs that link the nodes of the upper SOM to a document database and facilitate browsing and searching of the documents.

#### 3.2. Preprocessing of text

Before applying the SOM to a document collection (in this case 1,124,134 documents from 85 Usenet newsgroups, with a total of 245,592,634 words) we automatically removed some non-textual information from

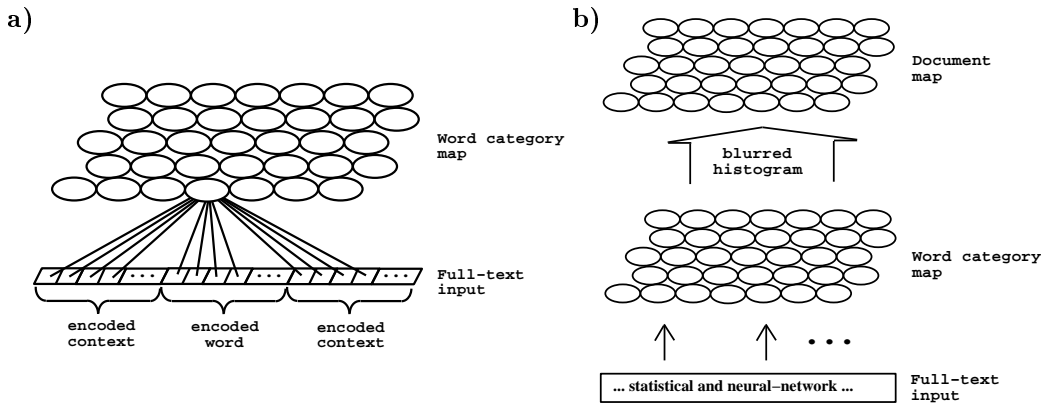


Fig. 1: The basic two-level WEBSOM architecture. a) The word category map first learns to represent relations of words based on their averaged contexts. The word category map is used to form a word histogram of the text to be analyzed. b) The histogram, a “fingerprint” of the document, is then used as input to the second SOM, the document map.

the documents. Also the articles (“a”, “an”, “the”), which would disturb the triples while not conveying much information, were removed. In the remaining text, the numerical expressions and several kinds of common code words were categorized with heuristic rules into a few classes of special symbols. To reduce the computational load the words that occurred only a few times (in this experiment less than 50 times) in the whole data base were neglected and treated as empty slots.

In order to emphasize the subject matter of the articles and to reduce variations caused by the different discussion styles, which were not of interest in this experiment, a group of common words that were not supposed to discriminate any discussion topics were discarded from the vocabulary. In the actual experiment we had originally 1,127,184 different words or symbols. After discarding the rare words, 67,220 words or symbols were left. From these we removed 3,447 common words,

### 3.3. Formation of the word category map and histogram

The first SOM, with 270 inputs and 315 map units, was formed and labeled using the whole preprocessed text material as training data.

The nodes of the SOM were labeled by inputting the feature vectors once again and finding the winner node for each. A node was thus labeled by *all the words* the corresponding feature vector of which selected this node for a winner. Fig. 2 illustrates the word category map.

In the encoding of *documents*, the text of each document separately was preprocessed as described in Sec. 3.2. When the encoded string of its words was scanned, the occurrence of each word was counted and recorded at that node of the first SOM which was labeled according to this word. A *word category histogram* is thus obtained.

If the documents belong to different groups, such as the newsgroups in the Internet, the counts can be further *weighted* by the information-theoretic *entropies* (Shannon entropies) of the words, defined in the following way. Denote by  $n_g(w)$  the frequency of occurrence of word  $w$  in group  $g$  ( $g = 1, \dots, 85$ ), normalized by the total size of the group, and by  $P_g(w)$  the share of the occurrences of word  $w$  belonging to group  $g$ . The entropy  $H$  of this word is defined as:

$$H(w) = - \sum_g P_g(w) \log P_g(w) \approx - \sum_g \frac{n_g(w)}{\sum_{g'} n_{g'}(w)} \log \frac{n_g(w)}{\sum_{g'} n_{g'}(w)}, \quad (2)$$

and the weight  $W(w)$  of word  $w$  is defined as

$$W(w) = H_{\max} - H(w), \quad H_{\max} = \log 85. \quad (3)$$

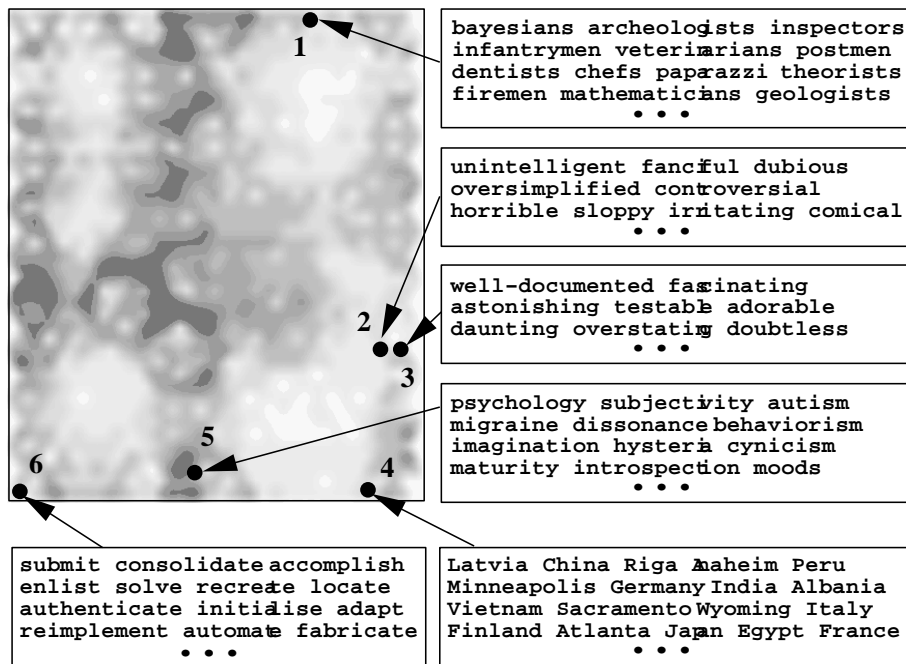


Fig. 2: Word category map, constructed from 1,124,134 Usenet articles, after discarding 3,447 most common words and words that occurred less than 50 times. The final size of the vocabulary used to make this map was 63,773 words or symbols. The degree of clustering is shown by the shade of gray (white: high density, dark: low density). Excerpts of the contents of six nodes are shown. Node 1: Mostly professions and roles (in total 109 words). Node 2: Mostly belittling attributes (in total 395 words). Node 3: Mostly praising attributes (in total 212 words). Node 4: Mostly names of countries and cities (in total 185 words). Node 5: Mostly psychological terms (in total 400 words). Node 6: Mostly verbs that describe constructive actions (in total 486 words).

### 3.4. Formation of the document map

The weighted histograms obtained in Sec. 3.3 were further *blurred* using a convolution with a symmetric Gaussian kernel, the full width at half maximum of which was two lattice units. The blurring increases invariance in classification. The *document map*, with 315 inputs and 104,040 map units, was then computed using the blurred histograms as input vectors.

With large maps, both winner search and updating are time-consuming tasks. Recently [3] (second edition), [5] we have been able to compute very large SOMs by two solutions: 1. Good initial values for a much larger map can be *estimated* on the basis of the asymptotic values of a smaller map. 2. In order to accelerate computations, the winner search can be speeded up by storing with each training vector an *address pointer to the old winner location*. During the next updating cycle, the approximate location of the winner can be found directly with the pointer, and only a *local search* around it needs to be performed. The pointer is then updated. In order to guarantee that the asymptotic state is not affected by this approximation, updating with a full winner search can be performed intermittently, say, after every 30 training cycles.

First we computed a SOM of the size of 18 by 45 = 810 nodes, letting it to converge carefully. After that the SOM was enlarged to the size of 204 by 510 = 104,040 nodes by the estimation procedure, and the final learning phase with 1,000,000 cycles was executed. With such a big size of the SOM, the neighborhood radius should be larger than with small maps, in order to keep the distribution of weight vectors smooth. The radius during the last fine-tuning phase varied from 11 to 10 lattice spacings. Each map vector became updated on the average 350 times during this last phase, which may be regarded as statistically sufficient.

### 3.5. Browsing interface

The document space in the WEBSOM software is presented at three basic levels of the system hierarchy: the map, the titles stored at the nodes, and the individual documents. Any subarea of the map can be selected and zoomed by “clicking.” One may explore the collection by following the links from hierarchy level to another. It is also possible to move to neighboring areas of the map, or to neighbors at the node level directly. This hierarchical system has been implemented as a set of WWW pages. They can be explored using any standard graphical browsing tool. Demonstrations are accessible in the Internet at the address <http://websom.hut.fi/websom/>.

### 3.6. Content-addressable search of documents or starting points for exploration

If the user wishes to explore information related to a specific theme, the most natural way to express what kind of information is desired is to describe it in natural language. The user may also wish to find and explore information related to a specific document.

In the system a new document, or any key text file may be mapped onto the document map. The map nodes in a nearby position of the new document then most likely contain related information, since they represent similar documents. Thus, the position of the new document on the document map provides an ideal starting point for exploring related documents and related information.

The facility of mapping a new document onto the document map enables the system to act as a genuine *content-addressable memory*, i.e. a memory where information may be searched by its content and not by the exact form in which it was input.

It should be noticed that the searched document and the key text file can use different synonyms for the same concept.

In Fig. 3 the user has written a text describing the desired information. On the document map, twenty best-matching map nodes are visualized by circles. (These circles are confused into a dot in Fig. 3.) The size of the circles, only visible after zooming, is supposed to reflect the goodness of match, i.e. the inverse distance between the normalized histogram vectors of the query and the node. By clicking one of the encircled nodes, the titles of the nodes are seen. The documents can then be read out.

The content-addressable searching demo is not yet available in the Internet.

## 4. Discussion

The main purpose of this paper was to demonstrate that an extremely large WEBSOM can be computed on a general-purpose computer (Silicon Graphics Power Challenge or Origin 2000) in a tolerable time, about eight or nine weeks; this time could still be reduced by optimization of the program code. Searching takes place in real time, on-line. Most of the computing time is needed for preprocessing of text, calibration of the word category SOM, and linking of the documents to the map.

The documents were not as well clustered in this experiment as in our smaller maps reported earlier [5]. The main reason is the heterogeneity of the contents of the 85 very different newsgroups involved in this experiment. The nodes of the document map contain, except the wanted information, also documents the topic of which is slightly different, although their *structure* is similar. This is due to the “too good” generalizing ability of the word category map: words that have a similar role in sentences while describing different themes have become mapped to the same nodes. It seems that for the size of the document map we had in this experiment, the size of the present word category map (315 nodes) was much too small. We are studying various methods to enlarge the word category map by an order of magnitude, whereby theme-specific terms would be mapped to different nodes.

Another handicap in this experiment was that the document map array was enlarged 128-fold in one step, as described in Sec. 3.4. This radical expansion of the array was made in order to produce the results for this paper by the deadline. Although after final tuning the SOM now seems to be rather well organized, its fine structure might have been smoother if we had enlarged the map in several smaller steps, letting the intermediate maps converge at each step.

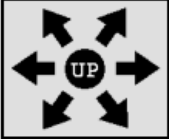
**Position your own document on the map**

Write here a description of what kind of information you

I would appreciate any references covering the use of fuzzy neural nets. As I have experience in NNs, and somewhat in fuzzy logic systems, but not the two used together, I don't require anything simplistic, but I probably would not benefit much from a highly advanced treatment of the topic. Thanks

Give the number of nodes :

**WEBSOM Node 37646**



**Click arrows**  
to move to neighboring nodes on the map.

[Instructions](#)

**Fuzzy logic and diagnostic decision making** ♦ Peter Hamilton, Tue, 07 Mar 1996, Lines: 1  
**Re: Fuzzy Class Lib for VisualWorks** ♦ Nicola Zordan, 12 Jun 1996, Lines: 1  
**BIOSCI Newsgroups Information** ♦ David Kristofferson, 1 Apr 93 9:2: C  
**comp.ai.neural-nets FAQ: Neurofuzzy** ♦ Warren Sarle, Wed, 12 Jun 1996, Lines: 1  
**A Letter from The Management** ♦ Mark D. Vanderbilt, Wed, 6 Mar 96, Lines: 1  
**Re: Why is action AI so poor?** ♦ Marty Stoneman, 20 Feb 1996, Lines: 1  
**Re: Critics on Fuzzy and Neural Net. Control** ♦ Peter J Turner, 12 Jul 1996, Lines: 1  
**Special High Intensity Training** ♦ M.S. Chan, 9 Apr 1996, Lines: 57.  
**fuzzy logic & constraint programming** ♦ Lim Swee Kiew, 18 Oct 1995, Lines: 1

Fig. 3: The 104,040-node map that contains 1,124,134 documents is shown in the background. A submitted query, shown on the top, has selected 20 best-matching nodes, and the titles of the documents mapped at one node are shown on the bottom.

## References

- [1] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biol. Cybern.*, vol. 61, pp. 241-254, 1989.
- [2] T. Kohonen, *Self-Organization and Associative Memory*, Series in Information Sciences, vol. 8, Springer-Verlag, Heidelberg, 1984.
- [3] T. Kohonen, *Self-Organizing Maps*, Series in Information Sciences, vol. 31, Springer-Verlag, Heidelberg, 1995; second edition, 1997.
- [4] T. Kohonen, *Content-Addressable Memories*, Series in Information Sciences, vol. 1, Springer-Verlag, Heidelberg, 1980.
- [5] T. Kohonen, S. Kaski, K. Lagus, and T. Honkela, "Very-large two-level SOM for the browsing of newsgroups," *Proc. Int. Conf. on Artificial Neural Networks (ICANN96)*, C. von der Malsburg et al. (eds.), Lecture Notes in Computer Science 1112, Springer, 1996.