Exploration of Text Collections with Hierarchical Feature Maps

Dieter Merkl *
Department of Computer Science
Royal Melbourne Institute of Technology
723 Swanston St., Carlton, VIC 3053, Australia
dieter@mds.rmit.edu.au

Abstract

Document classification is one of the central issues in information retrieval research. The aim is to uncover similarities between text documents. In other words, classification techniques are used to gain insight in the structure of the various data items contained in the text archive. In this paper we show the results from using a hierarchy of self-organizing maps to perform the text classification task. Each of the individual self-organizing maps is trained independently and gets specialized to a subset of the input data. As a consequence, the choice of this particular artificial neural network model enables the true establishment of a document taxonomy. The benefit of this approach is a straightforward representation of document similarities combined with dramatically reduced training time. In particular, the hierarchical representation of document collections is appealing because it is the underlying organizational principle in use by librarians providing the necessary familiarity for the user. The massive reduction in the time needed to train the artificial neural network together with its highly accurate clustering results makes it a challenging alternative to conventional approaches.

1 Introduction

During the last years we witnessed an ever increasing flood of miscellaneous digital information originating from very different sources. Powerful methods for organizing, exploring, and searching collections of textual documents are thus needed to deal with that information. Classical methods do exist for searching documents by means of keywords. These methods may be enhanced with proximity search functionality and keyword combination according to Boole's algebra. Other approaches rather rely on document similarity measures based on a vector representation of the various texts. What is still missing, however, are tools providing assistance for explorative search in text collections. Explorative search may be characterized with the lack of exact keywords which could guide the search process towards relevant information.

Exploration of document archives may be supported by organizing the various documents into taxonomies or hierarchies. In parentheses we should note that such an organization is in use by librarians for centuries. In order to reach such a classification a number of approaches are applicable. Among the oldest and most widely used ones we certainly have to mention statistics, especially cluster analysis [17]. The usage of cluster analysis for document classification has a long tradition in information retrieval research and its specific strength and weaknesses are well explored [49, 55].

The renaissance of widespread interest in artificial neural networks starting more than a decade ago is at least partly due to increased computing power available at reasonable prices and the development of a broad spectrum of highly effective learning rules. As a consequence, artificial neural networks are now widely used to uncover structure in a great variety of actual data [51]. Especially challenging in this context are high-dimensional input data. An almost perfect example of which is represented by text documents that are, by nature, described in a high-dimensional feature space spanned by the keywords extracted from the documents. There is already a substantial amount of work done in the application of neural networks to information retrieval. Consider as representatives the research reported in [10, 20, 29, 28, 31, 33, 39, 50].

In general, there is wide agreement that the application of artificial neural networks may be recommended in areas that are characterized by first noise, second poorly understood intrinsic structure, and third changing characteristics. Each of which is present in text classification. The noise is imposed due to the fact that no completely satisfying way to represent text documents has been found so far. Second, the poorly understood intrinsic structure is due to the non-existence of an authority knowing the contents of each and every document. Finally, the changing characteristics of document collections are due to the fact that the collections are regularly enlarged to comprise additional documents.

From the proposed architectures of artificial neural networks we regard the unsupervised models as especially well suited for text classification. This is due to the fact that in a supervised environment one would have to define proper input-output-mappings anew when the text archives changes; and such changes should be expected to happen quite frequently. By input-output-mapping we refer to the manual assignment of documents to classes which, obviously, is only possible when assuming the availability of considerable insight in the structure of the text archive. Contrary to that,

*Preferred address for correspondence: Institut für Softwaretechnik, Technische Universität Wien, Resselgasse 3/188, A-1040 Vienna, Austria, dieter@ifs.tuwien.ac.at.

20th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97), Philadelphia, PA, July 27–31, 1997.

in an unsupervised environment it remains the task of the artificial neural network to uncover the structure of the document archive. Hence, the unrealistic assumption of providing proper input-output-mappings is obsolete in an unsupervised environment. One of the most popular unsupervised neural network models certainly is the self-organizing map [22]. It is a general unsupervised tool for ordering high-dimensional statistical data in such a way that alike input items are mapped close to each other. In order to use the self-organizing map to cluster text documents, the various texts have to be represented as the histogram of its words. With this data, the artificial neural network performs the classification task in a completely unsupervised fashion.

In this paper we introduce the classification of text documents by means of hierarchically arranged self-organizing maps. A hierarchical arrangement has been chosen in order to enable the true establishment of a document taxonomy. Moreover, the hierarchical arrangement leads to remarkably faster training times of the neural network as compared to the basic self-organizing map algorithm.

The material contained in this paper is organized as follows. Section 2 is dedicated to an exposition of the neural network architecture we used for document classification. In Section 3 we provide an outline of our experimental setting. The discussion of the classification results and a comparison with alternative approaches is contained in Section 4. We give an account on related work in Section 5. Finally, we present our conclusions in Section 6.

2 Hierarchical Feature Maps

The key idea of hierarchical feature maps as outlined in [40, 42, 43] is to apply a hierarchical arrangement of several layers containing two-dimensional self-organizing maps [22, 23, 24]. The arrangement may be characterized as having the shape of a pyramid as shown in Figure 1. The hierarchy among the maps is established as follows. For each output unit in one layer of the hierarchy a two-dimensional self-organizing map is added to the next layer. The training of each single self-organizing map follows the basic self-organizing map learning rule.

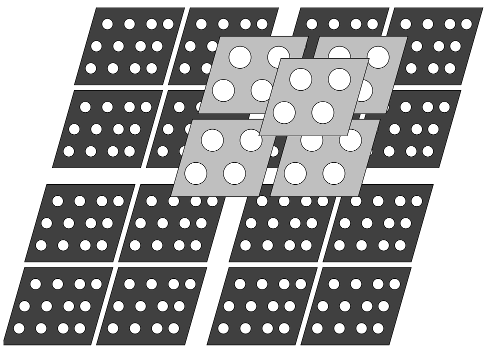

Figure 1: Hierarchical feature map

2.1 A self-organizing learning process

The learning process of self-organizing maps can be seen as a generalization of competitive learning [48]. The key idea of competitive learning is to adapt the unit c with the highest activity level with respect to a randomly selected input pattern x, $x = (\xi_1, \xi_2, \ldots, \xi_n)^T$, in a way to exhibit an even higher activity level with this very input in future. Commonly, the activity level of an output unit is computed as the Euclidean distance between the unit's weight vector m_c, $m_c = (\mu_{c_1}, \mu_{c_2}, \ldots, \mu_{c_n})^T$, and the actual input pattern. Given such an activation function, one is obviously interested in the unit exhibiting the minimal activity level. Hence, the so-called winning unit, i.e. the winner in short, is the output unit with the smallest distance between the two vectors in terms of the Euclidean vector norm. Adaptation takes place at each learning step and is performed as a gradual reduction of the difference between the respective components of input and weight vector. The degree of adaptation is guided by a so-called learning-rate, gradually decreasing in the course of time.

As an extension to competitive learning, units in a time-varying and gradually decreasing neighborhood around the winner are adapted, too. Pragmatically speaking, during the learning steps of self-organizing maps a set of units around the actual winner is tuned towards the currently presented input pattern. This learning rule leads to a clustering of highly similar input patterns in closely neighboring parts of the grid of output units. Thus, the learning process ends up with a topological ordering of the input patterns. One might say that self-organizing maps represent a spatially smooth neural version of k-means clustering [44] where k is equal to the number of output units.

More precisely, the four steps of the learning process may be outlined as given below.

1. Random selection of an input pattern $x(t)$.

2. Calculation of the distances between weight vectors and the input vector according to Expression 1.

$$D_i(t) = ||x(t) - m_i(t)|| = \sqrt{\sum_{p=1}^{n} (\xi_p - \mu_{i_p})^2} \qquad (1)$$

   In this expression m_i refers to the weight vector of unit i and $|| \cdot ||$ represents the Euclidean vector norm. As usual in discrete time notation, t represents the time-stamp of the current learning iteration.

3. Determination of the winning unit c as given in Expression 2.

$$c : ||x(t) - m_c(t)|| = \min_i (D_i(t)) \qquad (2)$$

4. Adaptation of weight vectors in the neighborhood of the winner. In particular we use a learning rule as outlined in Expression 3.

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{ci}(t) \cdot [x(t) - m_i(t)] \qquad (3)$$

   In this formula, $\alpha(t)$ represents a time-varying gain term, i.e. learning-rate, decreasing in the course of time. $h_{ci}(t)$ is a time-varying neighborhood function taking into account the distance between the winner c and unit i within the output space, i.e. $||r_c - r_i||$ with r_i denoting the two-dimensional vector pointing to the location of unit i within the grid.

Typically, the neighborhood function is designed to be symmetric around the location of the winner and monotonically decreasing with increasing distance from the winner. The task of the neighborhood function is to impose a spatial structure on the amount of weight vector adaptation in that the function computes values in the range of [0, 1], depending on the distance between the unit in question and the winner. By means of this function, units in close vicinity to the winner are provided with a larger value of the neighborhood function and thus, are adapted more strongly than units that are further away from the winner. In the experiments presented thereafter we use a simple Gaussian neighborhood function as given by Expression 4.

$$h_{ci}(t) = e^{-\frac{||r_c - r_i||}{2\cdot\delta(t)^2}} \tag{4}$$

In this realization δ is gradually decreasing with increasing learning iterations t. The purpose of this parameter is to determine the spatial width of the neighborhood function in terms of affected units.

Please note that by reducing both the learning-rate and the neighborhood-range, the learning process will converge towards a stable state. The stable state is reached when no further changes to the various weight vectors are observed. In practice, however, the learning process may be terminated earlier, namely at the time when no further variation within the process of winner selection is detected. In other words, the training process may be terminated when each input data is mapped repeatedly onto the same unit.

2.2 Properties of hierarchical feature map training

Generally, the training of a hierarchical feature map is performed sequentially from the first layer formed by one self-organizing map, downwards along the hierarchy. The maps on each layer are trained according to the standard self-organizing map learning process. As soon as the first layer map has reached a stable state, training continues with the maps of the second layer. Within the second layer, each map is trained with only that input data assigned to the corresponding unit of the first layer map. Moreover, the length of the input vectors may be reduced by omitting those vector components that are equal in the original input vectors. Due to this reduction in size of the input vectors the time needed to train the maps is reduced as well. The training of the second layer is completed when every map has reached a stable state. Analogously, the same training procedure is utilized to train the third and any subsequent layers of self-organizing maps.

An interesting property of hierarchical feature maps is the tremendous speed-up as compared to the self-organizing map. An explanation that goes beyond the obvious dimension reduction of the input data from one level to the next may be found by an investigation of the general properties of the self-organizing learning process. In self-organizing maps, the units that are subject to adaptation are selected by using a neighborhood function. It is common practice that in the beginning of the learning process almost the whole map is affected by the presentation of an input vector. Thus, the map is forced to establish large clusters of similar input data in the beginning of learning. The neighborhood size decreases gradually during the learning iterations leading to ever finer distinctions within the clusters whereas the overall topology of the cluster arrangement is maintained. However, in the single-level architecture the self-organizing process of each cluster interferes with the self-organization of its topologically neighboring clusters. Especially units along the boundaries tend to be occasionally modified as belonging to one or another cluster. This interference is one reason for the rather time-consuming self-organizing process. Contrary to that, such an interference is dramatically reduced due to the architecture of hierarchical feature maps. The topology of the high-level categories is depicted in the first layer of the hierarchy. Each of its sub-categories are then independently organized on separate maps at lower layers within the hierarchy. These maps in turn are free from maintaining the overall structure since this structure is already determined by the architecture of the hierarchical feature map. To conclude, much computational effort is saved due to the fact that the overall structure of the clusters is maintained in terms of the architecture rather than in terms of the learning process. However, the decision on the best size of the various maps as well as on the depth of the hierarchy remains a non-trivial problem requiring some insight in the structure of the input data.

3 The Experimental Document Archive

Throughout the remainder of the paper we will use the various manual pages of the NIH Class Library [13], NIHCL, as a sample document archive. The selection of the NIHCL is motivated by the fact that software libraries represent a convenient application arena for information retrieval systems. The reason is that much of the information about a particular software component is available as its textual description organized in manual pages. Moreover, the results from the classification process may easily be evaluated because the semantic, i.e. the functionality of the software component, is well-known.

The NIHCL is a collection of classes developed in the C++ programming language. The class library covers classes for storing and retrieving arbitrarily complex data structures on disk like OIOifd and OIOostream, generally useful data types such as String, Time, and Date, and finally a number of container classes as, for example, Set, Dictionary, and OrderedCltn. A more complete description of the class library may be found in [14].

Generally, the task of text classification may be paraphrased as uncovering the semantic similarities between various text documents. However, since natural language processing is still far from understanding arbitrarily complex document structures, the various documents have, in a first step, to be mapped onto some representation language in order to be comparable. Still one of the most widely used representation languages is single-term full-text indexing [49]. Roughly speaking, the documents are represented by the set of words they are built of. As a result, the various text documents are represented by vectors of equal dimension. Each vector component corresponds to a keyword from the representation language, and each vector entry in a particular component relates to the importance of that very keyword in describing the component. In the basic form, i.e. binary single-term indexing, an entry of one indicates that this specific keyword was extracted from the description of the component at hand. Contrary to that an entry of zero means that the corresponding keyword is not contained in that component's description [52].

Assuming a suitable representation, the similarity between two documents corresponds to the distance between their vector representations. The benefit of such a vector-based representation is the ease of implementing a best-match retrieval strategy, where the retrieved documents may be ranked according to decreasing similarity between the vector representing the actual query, i.e. the description of the needed documents, and the vectors representing the various text documents stored in the archive.

In order to obtain the final document representation, we accessed the full-text of the various manual pages describing NIHCL classes. As an illustrative example we refer to Figure 2 containing a portion of the textual description of class Set as provided in [13].

`Set` – Unordered Collection of Non-Duplicate Objects

Base Class: `Collection`

Derived Classes: `Dictionary`, `IdentSet`

Related Classes: `Iterator`

A `Set` is an unordered collection of objects. The objects cannot be accessed by a key as can the objects in a `Dictionary`, for example. Unlike a `Bag`, in which equal objects may occur more than once, a `Set` ignores attempts to add any object that duplicates one already in the `Set`. A `Set` considers two objects to be duplicates if they are `isEqual()` to one another.
Class `Set` is implemented using a hash table with open addressing. `Set::add()` calls the virtual member function `hash()` and uses the number returned to compute an index to the hash table at which to begin searching, and `isEqual()` is called to check for equality.

Figure 2: NIHCL manual entry of class Set

The full-text of the various manual pages describing the classes are accessed and full-text indexed in order to generate a binary vector-space representation of the documents. Just to provide the exact figure, the indexing process identified 489 distinct content terms and thus, each component is represented by a 489-dimensional feature vector. These vectors are subsequently used as the input data to the artificial neural network. Please note that we do not make use of the meta-information contained in the manual entries, like the reference to base class, derived classes, and related classes. Such an information is not generally available in document archives and thus, for the sake of wide applicability, we omitted the usage of this type of meta-information here. This type of meta-information, however, might be of importance when the focus of interest is concentrated on an application fine-tuned to this particular type of document archives, i.e. software libraries.

4 Experimental Results

4.1 A taxonomy of documents

Based on the document representation as presented above, we trained a hierarchical feature map to classify the document space, i.e. the descriptions of various software components. A typical result of the training process is presented below. The actual setup for the hierarchical feature map was exactly the same as depicted in Figure 1. Hence, we had four units in the top layer. Each of these units is expanded to a 2×2 self-organizing map in the second layer. The third layer consists of nine units in each single map.

Consider first Figure 3 containing the top-level map. In this figure we refrained from typing the names of the various classes for the sake of readability. Instead, we give the general purpose of the classes as meta-information concerning the mapping process. This meta-information, obviously, is derived manually by means of an inspection of the various documents represented by the respective units.

File I/O Classes	Data Types
Data Structures allowing access via key-attributes	Data Structures

Figure 3: Top level map

It is exactly the general structure that is mirrored in the arrangement of classes within the top-level map. In other words, the top-level neural network arranges the documents within four classes which show the variety of classes contained in the NIHCL with respect to their intended functionality, namely file I/O, data types, and data structures.

With the documents assigned to the various classes in the top-layer map, the training process continues with the second layer. Here, only the subset of documents that is matched onto the corresponding top-layer unit is used for training. The result of the training process of the second layer maps is presented in Figure 4. There is already highly interesting information presented concerning the relationship between the various classes assigned to the same general, or top-level, group.

As a first example, consider the map that is assigned the file I/O classes, i.e. the upper left map of Figure 4. In this map, pairs of classes performing "complementary" operations are assigned to the same unit. In fact, the "direction" of the operation is (in virtually every case) signified by the fourth character of the class name, i.e. 'i' for input and 'o' for output. In order to provide validation of the training process, we have split the input data in a set of training and a set of validation data. The validation data is printed by using bold face letters in the figure. One element of the validation data set was the class OIOostream which is perfectly correct mapped onto the same second layer unit as its pendant class OIOistream.

As a second example consider the separation of the documents describing data structures into a subset of data structures allowing access via a key attribute and in a subset with "non-keyed-access." This classification is already imposed within the top-layer map. In the middle-layer map, however, a highly useful arrangement of classes unfolds. Consider the right upper unit in the middle-level map containing the data structures with keyed-access. This unit comprises not only the data structures themselves, i.e. Dictionary and IdentDict, but also the classes that actually implement the keyed-access, namely Assoc, AssocInt, and LookupKey. The validation set contains a data structure with keyed-access as well, i.e. KeySortCltn, which is classified perfectly correct.

There is certainly a lot of other interesting information concerning the text collection contained in the maps depicted in Figure 4. We will, however, pick just one more for

File I/O Classes

OIOistream **OIOostream**	OIOnihout OIOnihin
OIOin OIOout	OIOifd OIOofd

Data Types

ReadFromTbl StoreOnTbl	Class, Object, Exception Link, FDSet, Range Point, Vector, Regex Nil, Time, Date
String	Float, Integer **Random**

Data Structures with access via key-attributes

	Assoc, AssocInt LookupKey Dictionary, IdentDict **KeySortCltn**
IdentSet	Set

Data Structures

Bitset	Arraychar Arrayob
Collection, Iterator OrderedCltn, Stack, Heap Bag, SortedCltn LinkedList	LinkOb

Figure 4: Middle level maps

detailed discussion here. It is the lower right unit within the "data types" map. This unit represents the basic numerical data types Float and Integer. The class Random, contained in the validation set, was mapped onto this very unit as well. This again corresponds to the functionality of the respective data types since Random implements a random number generator producing (pseudo-) random numbers of Float data type.

From the third layer maps we just present one of the "data types" maps in this paper. More precisely, the third-layer representation of the highly crowded right upper unit of the second layer "data types" map is depicted in Figure 5. The names of the various classes are mostly self-contained, so visual inspection of the classification is readily possible. The large number of entries to that particular unit is due to the fact that the class library's general base classes, i.e. Object, Class, and Exception, are regarded as "data types" by the unsupervised learning process, which, obviously, is not as confusing as one might expect on first sight. These classes, however, are nicely separated in the third layer of the hierarchy. A similar observation may be drawn from the lower left unit in the second layer of the "data structures" branch of the hierarchy. In this particular case, the base class for all container classes, i.e. Collection, and the class providing the means for uniform access to all container classes, i.e. Iterator, are mapped onto this very unit, too. Again, these classes are clearly separated within the next layer of the hierarchy. We refrain, however, from presenting a graphic representation of this and the other third-layer maps simply because they are highly similar in spirit to Figure 5.

Data Types

Object Class	FDSet	Range Regex Vector
Exception	Link	Point
Nil	Date	Time

Figure 5: Bottom level map

4.2 Comparison of the results

In order to provide the means for convenient comparison, we present the results from two alternative approaches to document classification in this subsection. One of these approaches is statistical, i.e. cluster analysis, the other is neural, i.e. self-organizing maps.

The statistical result presented in Table 1 was obtained with hierarchical agglomerative clustering by using complete linkage as the fusion algorithm. The results from other fusion algorithms like centroid and Ward are quite similar. Regarding the information on the overall structure of the document archive we may conclude that cluster analysis tends to produce on the one hand large groups of rather unrelated clusters, cf. classes 1 and 2 in the result presented in Table 1. On the other hand, some classes are highly specific in the sense of containing often just one document. Obviously, such a result is not very useful in an environment where one is interested in obtaining insight concerning the inherent structure of a particular document archive. In this sense, the distinction between the various groups of documents as contained in the archive, i.e. the various software components is by no means apparent from the results of cluster analysis. For a more detailed exposition of the shortcomings of cluster analysis in comparison with unsupervised neural networks we refer to [31].

Class 1	Arraychar, Arrayob, Bag, Bitset, Class, Heap, IdentDict, IdentSet, KeySortCltn, LinkedList, LinkOb, OIOifd, OIOin, OIOistream, OIOnihin, OIOnihout, OIOofd, OIOostream, OIOout, OrderedCltn, ReadFromTbl, Regex, SeqCltn, Set, SortedCltn, Stack, StoreOnTbl
Class 2	Assoc, AssocInt, Date, Dictionary, Exception, FDSet, Float, Integer, Link, LookupKey, Nil, Point, Random, Range
Class 3	Collection, Iterator
Class 4	Object
Class 5	String
Class 6	Vector

Table 1: Cluster analysis using complete linkage

Our previous work on document classification was mainly concerned with the application of self-organizing maps. A comparison to these results is necessary, even more since hierarchical feature maps are built up from a number of independent self-organizing maps. Moreover, the self-organizing map was used in [29] for text classification. In this paper, however, the authors use just a fairly small and thus unrealistic document representation, made up from 25 distinct terms taken from the titles of scientific papers.

In Figure 6 we provide a typical result from training a self-organizing map with the NIHCL data. For this experiment we used our own implementation of self-organizing maps as most thoroughly described in [30]. We want to point out that a highly effective public-domain implementation exists, the so-called SOM-PAK [25]. In the graphical representation of the self-organizing map each unit is signified by a dot in the figure. If a unit represents a particular document, or in other words, if a unit is the winner for a document, that document's name, i.e. the class name, is shown at the respective position in the figure.

Figure 6 shows in fact a highly similar classification result, in that the various documents are arranged within the two-dimensional output space of the self-organizing map in concordance with their mutual functional similarity. An exact positioning of the borderline between the various groups of similar documents, however, is not as intuitively to determine as with hierarchical feature maps that are presented above. Pragmatically speaking, an erroneous conclusion might be that the documents Object and OIOnihout are as similar to each other as the documents OIOnihout and OIOistream are. This because both pairs are arranged with the same distance on the top of the final map. Apparently, such a conclusion results in a highly erroneous perception of the underlying document collection. We should note at this point, however, that this drawback of self-organizing maps is subject to intensive research in the neural network community [6, 27, 35, 36, 37, 53].

Finally, we want to compare the classification capability of both neural network models according to the time needed for training. As already indicated in Section 2 one of the striking arguments in favor of hierarchical feature maps is their tremendous speed-up compared to self-organizing maps. The timing was done on an otherwise idle SUN SPARC-20 workstation with neural networks of exactly the same setup as above. The exact figures are presented in Table 2. In a nutshell, the hierarchical feature map performs training in just a fraction of the time required for the self-organizing map even though the number of units in the hierarchical feature map was 164 as compared to 100 units of the self-organizing map. These performance figures represent a convincing argument of the suitability of hierarchical feature maps especially for a task such as text clustering. The reason for the speed-up of the hierarchical model is related to the correlation within the feature space, on the one hand. On the other hand, the contents of a document archive as such is inherently structured in a hierarchical fashion.

	Training Time in Minutes
Hierarchical Feature Map	09:73
Self-Organizing Map	59:08

Table 2: Time needed for training on a SPARC-20

We have to note, however, that the detection of inter-cluster similarity is by no means as straightforward as in self-organizing maps. Consider for example the separation in the data structure classes. The criterion for the separation is the distinction in whether or not a direct access to the various elements via a key-attribute is provided. The information that both groups are container classes and in this sense related is invisible in the final arrangement simply because the respective documents are represented within different branches of the hierarchy. As another slight deficiency we have to admit that the determination of the hierarchical feature map's actual setup with respect to the size of the various self-organizing maps that make up the different layers is a non-trivial task. In general, one has to have some insight in the structure of the input data prior to neural network training to select an appropriate setup. The fact, however, that the hierarchical feature map has the benefit of a comparatively fast training time makes some experimentation with different network setups feasible which should presumably compensate for any inconvenience of the design step that requires some experience with the underlying input data.

5 Related Work

Document classification by using artificial neural networks has already gained some attention in the information retrieval community. Among the first and most influential papers we certainly have to mention [2]. In this work the author argues in favor of using feed-forward neural networks for query expansion. The neural network's role within the overall system is to perform spreading-activation during retrieval in order to describe the relation between terms, on the one hand, and documents and queries, on the other hand. This

.	OIOofd	OIOifd	OIOin	OIOistream	.	OIOnihout	.	Object	.
ReadFromTbl		.	OIOout	.	OIOnihin	.	.	.	Date
StoreOnTbl	.	LinkOb	.	OIOostream	.	FDSet	.	Point	.
.	Assoc	.	Class	.	Exception	.	Nil	.	Time
AssocInt	.	LookupKey	.	Link	.	Iterator	.	Random	.
.	Dictionary	.	.	LinkedList	.	.	Collection	.	Integer
.	KeySortCltn	.	OrderedCltn	.	SeqCltn	.	.	Float	.
IdentDict	.	SortedCltn	.	Stack	.	.	Arraychar	.	.
.	Set	.	Heap	.	.	Arrayob	.	String	.
IdentSet	.	Bag	.	Bitset	Vector	.	Range	.	Regex

Figure 6: A 10×10 self-organizing map

line of research is continued in [47, 46]. Comparable work is described in [20, 54]. Another approach relying on feed-forward networks is reported in [9]. In this paper the author describes an experiment where the weights of the neural networks are computed by using a supervised learning strategy rather than set directly using the term frequency histograms as it is done in most of the other studies.

A different line of research is performed using unsupervised neural networks. The paper of [29] perhaps marks the first attempt to utilize unsupervised neural networks for an information retrieval task. Similar to our approach, the authors rely on self-organizing maps. In this paper, however, the document representation is made up from 25 manually selected index terms and is thus not really realistic. In [31] this line of research is continued, yet this time with full-text indexed documents. In the area of legal information processing, the self-organizing map has been used in [38, 50] for exploratory analysis of judicial concept and document spaces.

Among the shortcomings of self-organizing maps one certainly has to mention the remarkable computational demands of the learning rule. Possibilities to increase the speed of learning may be found in the learning rule itself by using the biologically motivated concept of lateral inhibition [19]. Two different realizations of this principle are described in [34, 41]. Pragmatically speaking, a learning function incorporating lateral inhibition pushes the weight vector of distant units to the winner slightly away from the current input pattern. The effect of such a learning function is that the phase of rough input clustering is considerably accelerated in terms of the number of learning iterations that are needed to reach a stable state of the self-organizing process. Another convenient behaviour of such a learning rule is a remarkably increased accurracy of pattern representation in terms of the remaining quantization error after completion of the training process [34].

Another means for increasing the speed of the learning process is obviously related to the representation of the documents. In general, the feature space is not free from correlations due to the inexact mapping of free-form natural language text onto lexical entities. As a consequence, one might be interested in the transformation of the original document representation into a (much) lower dimensional space. In fact, this is the underlying principle of the *latent semantic indexing* technique [11]. Comparable results might be achieved by using *principal component analysis* [18] in order to reduce the dimensionality of the feature space. We refer to [1] for a recent achieving in the utilization of principal component analysis in the area of document processing. An approximation to the principal components may be gained by utilizing auto-associative feedforward neural networks, i.e. feedforward networks trained to reproduce the input at their output layer via a smaller layer of hidden units. The smaller hidden layer is further used to represent the input patterns. The effect of such a dimension reduction in keyword-based document representation and subsequent self-organizing map training with the compressed input patterns is described in [32]. To summarize the results, the experiments indicated that basically the same cluster results can be achieved by spending only a fraction of time for the training process.

Only recently, a number of papers have been published on the utilization of the self-organizing map for text representation [16] based on the seminal work of [45] and subsequent interactive exploration [15, 26, 28].

Apart from the self-organizing map just a limited number of other unsupervised models have been evaluated for their usability in information retrieval applications. In [21] the authors report on an application of *growing cell structures*, a network with adaptive architecture [12]. The learning process of this artificial neural network is highly similar to self-organizing maps in that during each training cycle the weight vector of the winner and those of a number of units in the neighborhood of the winner are adapted. A slight variation concerns the definition of the neighborhood where only direct neighbors of the winner are taken into account. The fundamental difference, however, is that ever after a fixed number of training cycles a new unit is added to the network at the position of the highest quantization error, i.e. the position of the largest deviation between the weight vector and the input patterns that are represented by that very unit. Additionally, a unit that serves the least often as winner may be deleted from the network. As an effect of this learning strategy the network structure itself is adapted to the particular requirements of the input space as opposed to self-organizing maps where the network structure in terms of the number of units and their topology has to be defined prior to the training process. This model, however, requires much more learning parameters, related to network structure adaptation, to be adjusted in advance. Moreover, it is much more susceptible to minor variations in these parameters than the self-organizing map.

A report on the applicability of *adaptive resonance theory* networks [5] to document clustering is provided in [33]. The major advantage of this type of network is its fast learning speed combined with continuous plasticity, i.e. the network is capable to add new data items without the need of re-training. In its most rudimentary form an adaptive resonance theory network consists of two layers, the one representing the input pattern and the other representing the various clusters in terms of a number of competitive units. The distinguished characteristic of that type of artificial neural networks, i.e. the continuous plasticity, is achieved by adding a new competitive unit in case non of the existing ones represent the actual input pattern with satisfying accuracy. In this sense, we might regard adaptive resonance theory networks as one of the earliest artificial neural network models with both adaptive weights and adaptive architecture. Information concerning intercluster similarity, however, cannot be deduced from the results.

There are still a number of apparently usable artificial neural network models unexplored as far as their applicability to document clustering is concerned. In particular, the so-called *generative topographic mapping* as only recently suggested in [3, 4] is developed as a substitute to the widely used self-organizing maps. In the words of the authors of [3], the generative topographic mapping is developed as a principled alternative to self-organizing maps. Basically, the generative topographic mapping is a latent variable density model with an apparently sound statistical foundation which is claimed to have several advantageous properties when compared to self-organizing maps, but no significant disadvantages. Probably one of the more important advantages is that generative topographic mapping should be open for rigorous mathematical treatment, an area where the self-organizing map has a remarkable tradition in effective resistance [7, 8]. We can imagine that an alternative model that lends itself to thorough mathematical treatment might reduce the highly time-consuming need for large numbers of empirical tests in order to realize a successful artificial neural network application. As just one example consider the question of how many neural units shall be considered to represent a particular set of input patterns with satisfactory accuracy. An (approximate) answer may only be found empirically with self-organizing maps and hierarchical feature maps.

On balance, unsupervised neural networks have proven to be remarkably successful as tools for explorative analysis of text archives as a number of studies have demonstrated that unsupervised neural networks are highly capable in uncovering similarities between text documents.

6 Conclusion

In this paper we reported on the successful application of a non-standard neural network model to the task of document classification. As the experimental setting we used the manual pages describing various C++ classes contained in a real-world class library where the inherent structure is known and thus, available for evaluation. A document representation according to the vector space model was further used as the input data during the learning process of the neural network.

The essentials of the chosen non-standard neural network model may be characterized as follows. First, the model adheres to the unsupervised learning paradigm which makes it well suited in an application with frequently changing underlying data. Text classification is certainly a perfect example of such an application. Second, the model imposes a hierarchical structure on the input data because of its architecture made up from small, hierarchically arranged unsupervised neural networks. In particular, the various layers comprise a number of mutually independent self-organizing maps. This hierarchical organization enables the true establishment of a document taxonomy. Such a hierarchical organization seems to fit perfectly for an area such as text archive exploration where a corresponding organization according to the various subject matters may be found. Additionally, a user may benefit from a rich source of experience not at least because hierarchical organizations are in use by librarians for centuries in order to arrange the contents of (conventional) libraries. Third, a specific benefit of the model is the remarkably fast training time making it a challenging alternative to standard approaches to text classification. We have identified this reduction of the time needed for training as compared with self-orgainizing maps being related to first the reduction of those components of the input patterns, i.e. the feature vectors describing a particular document, that are equal amoung the patterns mapped onto the same neural unit. It is obvious, that this alone should have considerable impact on the learning speed especially in an application area such as document clustering. Second and even more important, a substantial portion of the time needed to train a self-organizing map is dedicated to maintaining the overall arrangement of clusters within the output space. In the case of hierarchical feature maps this task is moved to the architecture of the artificial neural network and thus away from the training process.

As far as the classification process as such is concerned, we have demonstrated that the neural network has uncovered the semantic similarity of the various text documents successfully and structured them accordingly. The exploration of the document space is facilitated thanks to the establishment of a true document taxonomy. This fact together with the high learning speed makes hierarchical feature maps a promising alternative to standard approaches for text classification in large libraries.

References

[1] T. Bayer, I. Renz, M. Stein, and U. Kressel. Domain and language independent feature extraction for statistical text categorization. In *Proc of the Workshop on Language Engineering for Document Analysis and Recognition*, Sussex, United Kingdom, 1996.

[2] R. Belew. A connectionist approach to conceptual information retrieval. In *Proc of the Int'l Conference on Artificial Intelligence and Law (ICAIL'87)*, Boston, MA, 1987.

[3] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: A a principled alternative to the self-organizing map. In *Proc of the Int'l Conf on Artificial Neural Networks (ICANN'96)*, Bochum, Germany, 1996.

[4] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. Technical Report NCRG/96/015, Aston University, Neural Computing Research Group, http://www.ncrg.aston.ac.uk, Birmingham, United Kingdom, 1996.

[5] G. A. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, 21(3), 1988.

[6] M. Cottrell and E. de Bodt. A Kohonen map representation to avoid misleading interpretations. In *Proc of the European Symposium on Artificial Neural Networks (ESANN'96)*, Brugge, Belgium, 1996.

[7] M. Cottrell and J.-C. Fort. Etude d'un processus d'auto-organisation. *Annales de l'Institut Henri Poincaré*, 23(1), 1987.

[8] M. Cottrell, J.-C. Fort, and G. Pagès. Two or three things that we know about the Kohonen algorithm. In *Proc of the European Symposium on Artificial Neural Networks (ESANN'94)*, Bruxelles, Belgium, 1994.

[9] F. Crestiani. Learning strategies for an adaptive information retrieval system using neural networks. In *Proc of the IEEE Int'l Conf on Neural Networks (ICNN'93)*, San Francisco, California, 1993.

[10] C. J. Crouch, D. B. Crouch, and K. Nareddy. A connectionist model for information retrieval based on the vector space model. *Int'l Journal of Expert Systems*, 7(2), 1994.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Hashman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 1990.

[12] B. Fritzke. Growing Cell Structures: A self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9), 1994.

[13] K. E. Gorlen. *NIH class library reference manual.* National Institutes of Health, Bethesda, Maryland, 1990.

[14] K. E. Gorlen, S. Orlow, and P. Plexico. *Abstraction and Object-Oriented Programming in C++.* John Wiley, New York, 1990.

[15] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. Newsgroup exploration with WEBSOM method and browsing interface. Technical Report A32, Helsinki University of Technology, Laboratory of Computer and Information Science, http://websom.hut.fi, Espoo, Finland, 1996.

[16] T. Honkela, V. Pulkki, and T. Kohonen. Contextual relations of words in Grimm tales analyzed by self-organizing maps. In *Proc of the Int'l Conf on Artificial Neural Networks (ICANN'95)*, Paris, France, 1995.

[17] A. K. Jain and R. D. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, 1988.

[18] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, Berlin, 1986.

[19] E. R. Kandel, S. A. Siegelbaum, and J. H. Schwartz. Synaptic transmission. In E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors, *Principles of Neural Science.* Elsevier, New York, 1991.

[20] S. Keane, V. Ratnaike, and R. Wilkinson. Hierarchical news filtering. In *Proc of the Int'l Conf on Practical Aspects of Knowledge Management*, Basel, Switzerland, 1996.

[21] M. Köhle and D. Merkl. Visualizing similarities in high dimensional input spaces with a growing and splitting neural network. In *Proc of the Int'l Conf on Artificial Neural Networks (ICANN'96)*, Bochum, Germany, 1996.

[22] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 1982.

[23] T. Kohonen. Generalizations of the self-organizing map. In *Proc of the Int'l Joint Conf on Neural Networks (IJCNN'93)*, Nagoya, Japan, 1993.

[24] T. Kohonen. *Self-organizing maps.* Springer-Verlag, Berlin, 1995.

[25] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. *SOM-PAK: The self-organizing map program package, Version 3.1.* Helsinki University of Technology, Laboratory of Computer and Information Science, http://nucleus.hut.fi, Espoo, Finland, 1995.

[26] T. Kohonen, S. Kaski, K. Lagus, and T. Honkela. Very large two-level SOM for the browsing of newsgroups. In *Proc of the Int'l Conf on Artificial Neural Networks (ICANN'96)*, Bochum, Germany, 1996.

[27] M. A. Kraaijveld, J. Mao, and A. K. Jain. A nonlinear projection method based on Kohonen's topology preserving maps. In *Proc of the Int'l Conference on Pattern Recognition (ICPR'92)*, 1992.

[28] 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 Int'l Conf on Knowledge Discovery and Data Mining (KDD-96)*, Portland, OR, 1996.

[29] X. Lin, D. Soergel, and G. Marchionini. A self-organizing semantic map for information retrieval. In *Proc of the ACM SIGIR Int'l Conf on Research and Development in Information Retrieval (SIGIR'91)*, Chicago, IL, 1991.

[30] D. Merkl. *Self-Organization of Software Libraries: An Artificial Neural Network Approach.* PhD thesis, Institut für Angewandte Informatik und Informationssysteme, Universität Wien, 1994.

[31] D. Merkl. A connectionist view on document classification. In *Proc of the Australasian Database Conf (ADC'95)*, Adelaide, SA, 1995.

[32] D. Merkl. Content-based document classification with highly compressed input data. In *Proc of the Int'l Conf on Artificial Neural Networks (ICANN'95)*, Paris, France, 1995.

[33] D. Merkl. Content-based software classification by self-organization. In *Proc of the IEEE Int'l Conf on Neural Networks (ICNN'95)*, Perth, WA, 1995.

[34] D. Merkl. The effect of lateral inhibition on learning speed and precision of a self-organizing map. In *Proc of the Australian Conf on Neural Networks*, Sydney, NSW, 1995.

[35] D. Merkl. Exploration of document collections with self-organizing maps: A novel approach to similarity representation. In *Proc of the European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97)*, Trondheim, Norway, 1997.

[36] D. Merkl and A. Rauber. Alternative ways for cluster visualization in self-organizing maps. In *Proc of the Workshop on Self-Organizing Maps*, Espoo, Finland, 1997.

[37] D. Merkl and A. Rauber. On the similarity of eagles, hawks, and cows: Visualization of similarity in self-organizing maps. In *Proc of the Int'l Workshop Fuzzy-Neuro-Systems'97*, Soest, Germany, 1997.

[38] D. Merkl, E. Schweighofer, and W. Winiwarter. CONCAT: Connotation analysis of thesauri based on the interpretation of context meaning. In *Proc of the Int'l Conference on Database and Expert Systems Applications (DEXA'94)*, Athens, Greece, 1994.

[39] E. Merlo, I. McAdam, and R. De Mori. Source code informal information analysis using connectionist models. In *Proc of the Int'l Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambèry, France, 1993.

[40] R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2, 1990.

[41] R. Miikkulainen. Self-organizing process based on lateral inhibition and synaptic resource redistribution. In *Proc of the Int'l Conf on Artificial Neural Networks (ICANN'91)*, Espoo, Finland, 1991.

[42] R. Miikkulainen. Trace feature map: A model of episodic associative memory. *Biological Cybernetics*, 66, 1992.

[43] R. Miikkulainen. *Subsymbolic Natural Language Processing: An integrated model of scripts, lexicon, and memory.* MIT-Press, Cambridge, MA, 1993.

[44] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, United Kingdom, 1996.

[45] H. Ritter and T. Kohonen. Self-organizing semantic maps. *Biological Cybernetics*, 61, 1989.

[46] D. E. Rose. *A Symbolic and Connectionist Approach to Legal Information Retrieval.* Lawrence Erlbaum, Hillsdale, 1994.

[47] D. E. Rose and R. K. Belew. Legal information retrieval: A hybrid approach. In *Proc of the Int'l Conference on Artificial Intelligence and Law (ICAIL'89)*, Vancouver, Canada, 1989.

[48] D. E. Rumelhart and D. Zipser. Feature discovery by competitive learning. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. I. Foundations.* MIT Press, Cambridge, MA, 1986.

[49] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.

[50] E. Schweighofer, W. Winiwarter, and D. Merkl. Information filtering: The computation of similarities in large corpora of legal text. In *Proc of the Int'l Conf on Artificial Intelligence and Law (ICAIL'95)*, College Park, MD, 1995.

[51] K. Swingler. *Applying Neural Networks: A Practical Guide.* Academic Press, London, 1996.

[52] H. R. Turtle and W. B. Croft. A comparison of text retrieval models. *Computer Journal*, 35(3), 1992.

[53] A. Ultsch. Self-organizing neural networks for visualization and classification. In O. Opitz, B. Lausen, and R. Klar, editors, *Information and Classification – Concepts, Methods, and Applications.* Springer-Verlag, Berlin, 1993.

[54] R. Wilkinson and P. Hingston. Incorporating the vector space model in a neural network used for information retrieval. In *Proc of the ACM SIGIR Int'l Conf on Research and Development in Information Retrieval (SIGIR'91)*, Chicago, IL, 1991.

[55] P. Willet. Recend trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24, 1988.