Exploration of Document Collections with Self-Organizing Maps: A Novel Approach to Similarity Representation

Dieter Merkl *

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

Abstract. Classification is one of the central issues in any system dealing with text data. The need for effective approaches is dramatically increased nowadays due to the advent of massive digital libraries containing free-form documents. What we are looking for are powerful methods for the exploration of such libraries whereby the detection of similarities between the various text documents is the overall goal. In other words, methods that may be used to gain insight in the inherent structure of the various items contained in a text archive are needed. In this paper we demonstrate the applicability of self-organizing maps, a neural network model adhering to the unsupervised learning paradigm, for the task of text document clustering. In order to improve the representation of the result we present an extension to the basic learning rule that captures the movement of the various weight vectors in a two-dimensional output space for convenient visual inspection. The result of the extended training algorithm allows intuitive analysis of the similarities inherent in the input data and most important, intuitive recognition of cluster boundaries.

1 Introduction

During the last years we witnessed an ever increasing flood of written information culminating in the advent of massive digital libraries. Powerful methods for organizing, exploring, and searching collections of textual documents are thus needed to deal with that information. The classical way of dealing with textual information as developed in the information retrieval community is defined by means of keyword-based document representations. 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,

* Address for correspondence:
Institut für Softwaretechnik, Technische Universität Wien, A-1040 Wien, Austria, dieter@ifs.tuwien.ac.at.

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 document organization a number of approaches are applicable. Among the oldest and most widely used ones we certainly have to mention statistics, especially cluster analysis. The usage of cluster analysis for document clustering has a long tradition in information retrieval research and its specific strength and weaknesses are well explored [18, 16, 21]. Generally, clustering is used to sort the documents into homogeneous groups. The rationale behind clustering approaches in information retrieval is the hypothesis that similar documents tend to be relevant with respect to the same queries, i.e. the cluster hypothesis [20].

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. In general, there is wide agreement that the application of artificial neural networks is suitable in areas that are characterized by first noise, second poorly understood intrinsic structure, and third changing characteristics. Each of which is present in text clustering. 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 wide range of proposed architectures of artificial neural networks we regard the unsupervised models as especially well suited for text clustering. 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, 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 having available proper input-output-mappings is obsolete in an unsupervised environment. A number of successful applications of unsupervised neural networks to information retrieval have already been reported in literature [5, 8, 9, 10, 11, 12].

One of the most popular unsupervised neural network models certainly is the self-organizing map [6]. 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 clustering task in a completely unsupervised fashion.

In the remainder of this paper we will describe the task of text clustering by means of self-organizing maps with an enhanced visual representation of the result. In particular, in Section 2 we will provide a brief outline of the necessary pre-processing operations in order to perform text clustering. We will cover the architecture and the learning rule of self-organizing maps in Section 3. Section 4 contains the results from clustering the documents contained in an experimental text archive. Finally, we will present our conclusions in Section 5.

2 Pre-Processing of Text Documents

Generally, the task of text clustering is to uncover the semantic similarities between various documents and to assign those documents to classes accordingly. An example for such classes consider the subject matter of the various documents. In a much broader sense we might regard the whole process of information retrieval as a classification task, namely the discrimination between a set of relevant and a set of non-relevant documents with respect to a particular query or interest profile. 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 relies on terms or keywords extracted directly from the documents. Of particular interest is the histogram of keywords with respect to a particular document. Roughly speaking, the documents are represented by feature vectors $x = (\xi_1, \xi_2, \ldots, \xi_n)$. Thereby ξ_i refers to a term extracted from the various documents contained in the archive. The specific value of ξ_i corresponds to some measure of importance of this particular feature for the document at hand. Quite a number of distinct functions to capture the importance of a term have been proposed in literature; see [17] for an overview of these so-called term weighting functions. In the perhaps most basic form of term weighting, i.e. binary single-term indexing, a value of one indicates that this specific feature was extracted from the document at hand. Contrary to that a value of zero means that the corresponding feature is not contained in that document.

Such a vector-based document representation schema is known as the vector space model of information retrieval [19]. Assuming a meaningful representation, the similarity between two components corresponds to the distance between their vector representations.

Throughout the remainder of the paper we will use the various manual-pages of the NIH Class Library [3], i.e. 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 clustering 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 `OIOout` and `OIOnihin`, 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 [4].

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. As a sample document consider Figure 1 showing a portion of the manual page related to class `Set`. 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 that will be described below. 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.

> `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.

Fig. 1. NIHCL manual entry of class `Set`

3 Self-Organizing Maps

3.1 The Basic Model

The self-organizing map as proposed in [6] and described thoroughly in [7] is one of the most distinguished artificial neural networks with unsupervised learning rule. Particularly remarkable is the model's ability to arrange input patterns within its output space such that alike input patterns are represented in neighboring regions.

Basically, the model consists of a layer of input units each of which is fully connected to a grid of output units. The only responsibility of the input units is to receive the input patterns x, $x \in \Re^n$, one at a time and propagate them as they are onto the output units. The output units are arranged in some topological order where the most common choice is represented by a two-dimensional grid which is henceforth referred to as the output space of the self-organizing map. Each of the output units i is assigned a weight vector m_i of the same dimension as the input data, i.e. $m_i \in \Re^n$.

During the learning process the unit c with the highest activity level with respect to a randomly selected input pattern x is adapted in a way to exhibit an even higher activity level with this very input in the future. We refer to this very unit as the 'winner' of the input presentation. For the computation of the activity level any metric of similarity or dissimilarity between two vectors may be chosen. A widely used metric is marked by the Euclidean distance between a unit's weight vector and the actual input pattern. When using the Euclidean distance to compute the activity level, the process of winner selection may be given as in Expression 1.

$$c : ||x - m_c|| = \min_i\{||x - m_i||\} \tag{1}$$

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, α, that is gradually decreasing in the course of time. As an extension to standard competitive learning as proposed in [15], units in a time-varying and gradually decreasing neighborhood around the winner are adapted, too. This neighborhood may be described by means of a so-called neighborhood function, $\phi_{c,i}$, taking into account the distance in terms of the output space between unit i currently under consideration and the winner c. Typically, the neighborhood function is symmetric around the location of the winner and monotonically decreasing with increasing distance from the winner. Hence, the adaptation of a weight vector according to learning step t may be written as given in Expression 2.

$$m_i(t + 1) = m_i(t) + \alpha(t) \cdot \phi_{c,i}(t) \cdot [x(t) - m_i(t)] \tag{2}$$

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.

3.2 Visualization by Adaptive Coordinates

We extended the basic learning rule in order to capture the movements of the various weight vectors within a two-dimensional 'virtual' output space for subsequent visualization of the clustering result. We will refer to this extension as the *adaptive coordinate* technique.

Initially, each of the output units of the self-organizing maps is shown in its initial position within the two-dimensional grid defined by the neural network architecture. In other words, the initial coordinates $\langle ax_i, ay_i \rangle$ of the unit i are identical to the unit's position within the grid of the map. In the beginning of each learning step the distance between the weight vectors and a randomly selected input pattern are stored in a table, i.e. $Dist(t)$. After the adaptation of weight vectors the new distance table is calculated, i.e. $Dist(t+1)$. The obvious next step is to compute the relative change in distance, i.e. Δ_{Dist}, according to this learning step for every unit i. This computation is given in Expression 3.

$$\Delta_{Dist_i}(t+1) = \frac{Dist_i(t) - Dist_i(t+1)}{Dist_i(t)} \tag{3}$$

The movement of the various weight vectors within the input space due to the adaptation process is performed analogously within the 'virtual' output space. Pragmatically speaking, the adaptive coordinates $\langle ax_i, ay_i \rangle$ are used in order to mimic the movement of this unit's weight vector during the training process. Since the presented input signal was mapped onto the winning unit c, no adaptation of the position of the winning unit is performed, with c now being representative of the selected input signal in terms of the 'virtual' output space. The adaptive coordinates of all units but the winner are now moved by the fraction given in $\Delta_{Dist_i}(t+1)$ towards the position of the winning unit c given by $\langle ax_c, ay_c \rangle$. In Expression 4 we provide the exact formulation of adaptation performed with the ax-coordinate. The calculation of the new ay-coordinates, ay_i, is performed analogously.

$$ax_i(t+1) = ax_i(t) + \Delta_{Dist_i}(t+1) \cdot (ax_c(t) - ax_i(t)) \tag{4}$$

Thus, the clustering of units around the winning unit resembles the clustering of the units' weight vectors around the presented input signal after the current training cycle. After convergence of the training process the clusters learned by the self-organizing map can be visualized by using the coordinates $\langle ax_i, ay_i \rangle$ to plot the unit i in the 'virtual' output space. We have to note that we usually do not start with adaptive coordinate alteration right from the beginning of the learning process. We rather wait until the first stage of rough clustering is finished when the neighborhood function returns values of, say, half the initial one.

A more detailed discussion of the visualization technique together with results achieved on some small yet classic benchmark data sets may be found in [13, 14].

4 Visualization of Document Clusters

4.1 An Experiment in Document Clustering

A representative result from a 10×10 self-organizing map trained to represent the NIH documents is depicted in Figure 2. Each unit is designated by a small circle in the figure. In case the unit is winner of a C++ class, i.e. the weight vector of the unit has the smallest distance to that very input vector, we provide the name of the C++ class as well.

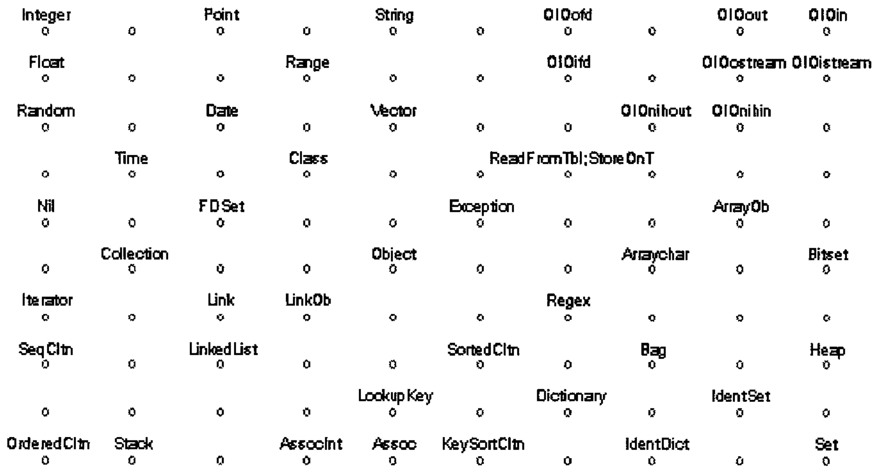

Fig. 2. Arrangement of documents in a 10 × 10 self-organizing map

As an interesting area of the final result consider the upper left part of the map containing the data types. In this area the functionally similar classes such as `Integer` and `Float` or `Time` and `Date`, just to name two examples, are assigned neighboring units. Moreover, the class `Random`, i.e. random number generator producing floats, is assigned to a unit neighboring class `Float`.

The right upper part of the final map contains all classes that implement file input/output operations. These classes are identifiable by the '`OIO`'-part in their class name. Additionally, the class performing an input operation is assigned neighboring the class that performs the analogue output operation, e.g. `OIOifd` and `OIOofd`.

Finally, we want to direct the attention to the bottom part of the map containing the classes implementing basic data structures. More specifically, we refer to the container classes allowing access to their data elements via a specific key attribute, i.e. `KeySortCltn`, `Dictionary`, and `IdentDict`. These classes are assigned to units in a closely neighboring region in the bottom middle of the map. In direct vicinity we find the NIHCL classes that actually implement such a key attribute based access, i.e. `Assoc`, `AssocInt`, and `LookupKey`.

However, in case one is not familiar with the NIHCL as such it is still problematic to identify regions of functionally similar C++ classes. From the pure two-dimensional representation as provided in Figure 2 a casual user might conclude that the classes `Point` and `String` or `String` and `OIOofd` are of comparable similarity since they are located at units of equal distance in the upper middle of the final map. The similarity of the former two classes is evident in that both represent data types. The latter pair, however, is formed from two unrelated classes, the one being a data type, the other a file I/O class. Such a representation might be misleading for the user of the software library, to say the least.

Our extension of the learning process addresses exactly this type of misleading representation. Figure 3 depicts the result of a training process with the adaptive coordinate technique. The other parameters of the learning process, i.e. initial learning-rate and neighborhood function, are the same for both results as presented in Figures 2 and 3.

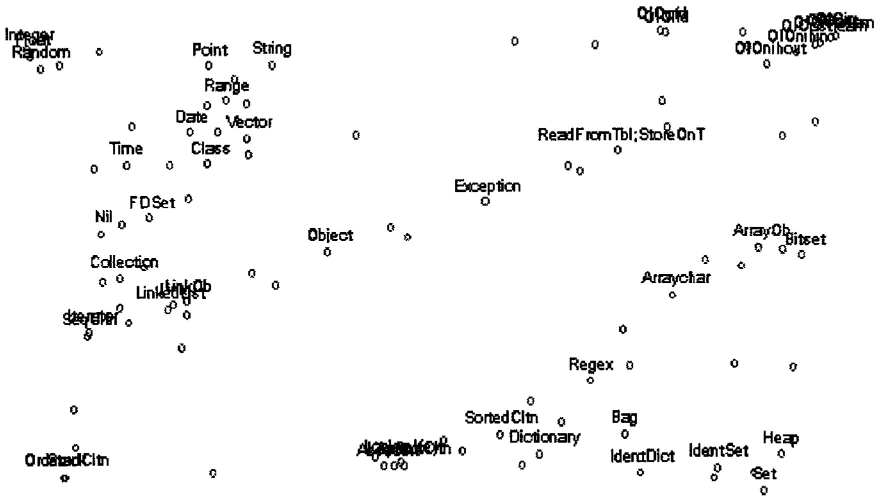

Fig. 3. Adaptive coordinate representation of a 10 × 10 self-organizing map

The most obvious observation from the result as depicted in Figure 3 is that the regions we described for Figure 2 are readily recognizable thanks to the fact that they are better separated graphically. Consider for example the pairs `Point` and `String` or `String` and `OIOofd`. These pairs have been identified as a sample area of misleading visualization in the conventional output space representation. With the adaptive coordinate technique they are now separated substantially in the output space. There is certainly no longer the risk that these classes might be considered as being comparably similar.

Moreover, some parts of the final map contain such highly similar classes that the learning process allocates a fairly condensed region in the map for them. As an example consider the file I/O classes in the right upper part of Figure 3. A similar observation holds true for the left upper part of the map containing the various classes implementing data types. The fact that these documents belong to a fairly homogeneous group is even stressed by using the adaptive coordinate technique for output space visualization.

4.2 Discussion of the Visualization Technique

The adaptive coordinate technique represents a novel approach for similarity visualization in the context of self-organizing maps. The major benefit is the representation of input pattern similarity by means of the distances between their respective locations in the output space of the self-organizing map. The various clusters themselves are organized according to the same principle, i.e. similar clusters are located more closely to each other than distinct ones. Thus, the structure of the input data is readily visible as areas of strong concentration within the output space. As a consequence, the uniformity of output space visualization provided in the standard representation is damaged and is substituted by a more cluttered visualization. The inherent structure of the input data, however, is more intuitively observable, i.e. less prior knowledge concerning the input patterns is needed to realize which patterns are more and which patterns are less similar to each other. The final interpretation of the clustering result, however, is left to the user. The artificial neural network and the novel visualization technique provide merely an enumeration of similar documents.

The adaptive coordinate technique does not reveal 'new' clusters as compared to the standard representation because the same mapping is represented. The intention of the novel visualization technique is to enable an easier uncovering of the inherent structure of the input data while leaving less room for erroneous perceptions. In other words, the overall topology of the pattern arrangement by using adaptive coordinates is the same as with the standard representation. The adaptive coordinate technique provides the possibility of explicit cluster boundary visualization by means of separation in terms of the output space. This is, obviously, not to the same degree achievable with the standard representation due to the fixed grid and the limited number of output units.

Due to the fact that the adaptive coordinate technique is an add-on in the sense that the standard learning rule is left unchanged we are not faced with the need for defining a number of additional learning parameters as it is necessary with other models, e.g. [1, 2, 5], that address cluster boundary visualization, too. Thus, we may still rely on the results and experiences obtained from the application of self-organizing maps. This is even more desirable when taking into account the fact that the self-organizing map is remarkably robust in the sense of being far less susceptible to variations in the learning parameters than other neural network models.

We feel that our extension to the standard learning process of self-organizing maps yields results that are even easier interpretable by simple visual inspection. This benefit certainly compensates for slightly increased computational demands of the adaptive coordinates technique due to distance table calculation.

5 Conclusions

In this paper we described the adaptive coordinate technique, an extension to the self-organizing map learning process, aiming at an improved visual representation of high-dimensional input structures in the map's two-dimensional output space. With this extension we adapt the output unit's coordinates, too, during the training process. The key idea may be paraphrased as moving the units proportional to the amount of adaptation performed with the unit's weight vector at each learning iteration. The effects of this extension have been described by using an application from the area of text clustering where the various documents are represented by feature vectors compiled from keywords extracted from the full-text of the documents. We pointed out that with the adaptive coordinate technique the resulting maps are more easily interpretable in the sense that regions containing similar input data are mapped onto regions of the map that are recognizable by simple visual inspection without the need of profound prior knowledge concerning the underlying input data.

Acknowledgments

Thanks go to Andreas Rauber for long and fruitful discussions on cluster visualization and the implementation of the adaptive coordinate visualization technique.

References

1. J. Blackmore and R. Miikkulainen. Incremental Grid Growing: Encoding high-dimensional structure into a two-dimensional feature map. In *Proc IEEE Int'l Conf on Neural Networks (ICNN'93)*, San Francisco, CA, 1993.
2. B. Fritzke. Growing Cell Structures: A self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9), 1994.
3. K. E. Gorlen. *NIH class library reference manual.* National Institutes of Health, Bethesda, MD, 1990.
4. K. E. Gorlen, S. Orlow, and P. Plexico. *Abstraction and Object-Oriented Programming in C++.* John Wiley, New York, 1990.
5. M. Köhle and D. Merkl. Visualizing similarities in high dimensional input spaces with a growing and splitting neural network. In *Proc Int'l Conf on Artificial Neural Networks (ICANN'96)*, Bochum, Germany, 1996.
6. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 1982.
7. T. Kohonen. *Self-organizing maps.* Springer-Verlag, Berlin, 1995.
8. K. Lagus, T. Honkela, S. Kaski, and T. Kohonen. Self-organizing maps of document collections: A new approach to interactive exploration. In *Proc Int'l Conf on Knowledge Discovery and Data Mining (KDD'96)*, Portland, OR, 1996.
9. X. Lin, D. Soergel, and G. Marchionini. A self-organizing semantic map for information retrieval. In *Proc Int'l ACM SIGIR Conf on Research and Development in Information Retrieval (SIGIR'91)*, Chicago, IL, 1991.

10. D. Merkl. A connectionist view on document classification. In *Proc Australasian Database Conf (ADC'95)*, Adelaide, Australia, 1995.
11. D. Merkl. Content-based software classification by self-organization. In *Proc IEEE Int'l Conf on Neural Networks (ICNN'95)*, Perth, Australia, 1995.
12. D. Merkl. Exploration of text collections with hierarchical feature maps. In *Proc Int'l ACM SIGIR Conf on Research and Development in Information Retieval (SIGIR'97)*, Philadelphia, PA, 1997.
13. D. Merkl and A. Rauber. On the similarity of eagles, hawks, and cows – Visualization of similarity in self-organizing maps. In *Proc Int'l Workshop Fuzzy-Neuro-Systems*, Soest, Germany, 1997.
14. A. Rauber. *Cluster Visualization in Unsupervised Neural Networks.* Master Thesis, Technische Universität Wien, Wien, Austria, 1996.
15. 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.
16. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.
17. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 1988.
18. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.
19. H. R. Turtle and W. B. Croft. A comparison of text retrieval models. *Computer Journal*, 35(3), 1992.
20. C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1975.
21. P. Willet. Recend trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24, 1988.