Clustering of Document Collections using a Growing Self-Organizing Map

Andreas Nürnberger
University of California at Berkeley
EECS, Computer Science Division
Berkeley, CA 94720, USA
E-mail: anuernb@eecs.berkeley.edu

Abstract

Clustering methods are frequently used in data analysis to find groups in the data such that objects in the same group are similar to each other. Applied to document collections, clustering methods can be used to structure the collection based on the similarities of the contained documents and thus support a user in searching for similar documents. Furthermore, the discovered clusters can be automatically indexed by keywords. Therefore the user does not depend on manually defined index terms or a fixed hierarchy, which often did not reflect recent changes in the underlying document collections. In this article we present an approach that clusters a document collection using a growing self-organizing map. The presented method was implemented in a software tool, which combines keyword search methods with a visualization of the document collection.

1. Introduction

Index terms and (hierarchical) classification methods are frequently used to structure object collections, e.g. document archives or libraries, and thus simplify the access for a user who is searching for specific documents. One of the main drawbacks of this approach is that the maintenance of these indexes is very expensive. Furthermore, they are usually not applied consistently and – since they are not updated frequently – they usually did not reflect recent changes of the structure of the underlying document collection.

Clustering is a well-known approach to structure prior unknown and unclassified datasets. Applied to document collections, clustering methods can be used to structure the collection based on the similarities of the contained documents and thus support a user in searching for similar documents. Furthermore, the discovered clusters can be automatically indexed by keywords without the need to manually define index terms.

In the following we present an approach that clusters a document collections using a growing self-organizing map. The method was implemented in a software tool that combines conventional keyword search methods with a visualization of the document collection. With the approach presented in this paper we resolved some of the problems of a first prototype that was implemented using conventional self-organizing maps [9, 15]. The main disadvantage of this architecture was that the size and shape of the map had to be defined in advance. Therefore a map had to be trained several times to obtain an appropriate solution. Especially for huge collections of documents this process is usually very time-consuming.

Clustering a document collection using self-organizing maps requires a preprocessing of the document collection to obtain numerical data describing each document. Similar to most of the existing models for document retrieval our approach is based on the vector space model [18]. The vector space model represents terms and documents as vectors in k-dimensional space. The currently most popular models using this approach are Latent Semantic Indexing (LSI) [2], Random Projection [8], and Independent Component Analysis (ICA) [7].

The vector space model enables very efficient analysis of huge document collections due to its simple data structure without using any explicit semantic information. A document is described based on a 'statistical fingerprint' of word occurrences and the semantically information is considered based on statistical correlations in further processing steps, e.g. the so-called 'latent semantic' in the LSI approach [13, 14]. However, some approaches try to consider semantic information by a preprocessing step, see e.g., the analysis of three-word contexts discussed in [5]. In spite of the insufficiencies the vector space model enables the processing of large document collections efficiently. Furthermore, using self-organizing maps document collections and search results can be visualized in an intuitive way.

In the following section we will briefly review the concepts of self-organizing systems and the implemented growing self-organizing map approach. In Sect. 3 we describe the document pre-processing steps and the

methods used for grouping the text documents based on different similarity measures. In Sect. 4 we present the implementation of this approach.

2. Self-organizing maps

Self-organizing maps [10] are a special architecture of neural networks that cluster high-dimensional data vectors according to a similarity measure. The clusters are arranged in a low-dimensional topology that preserves the neighborhood relations in the high dimensional data. Thus, not only objects that are assigned to one cluster are similar to each other (as in every cluster analysis), but also objects of nearby clusters are expected to be more similar than objects in more distant clusters. Usually, two-dimensional grids of squares or hexagons are used. Although other topologies are possible, two-dimensional maps have the advantage of an intuitive visualization and thus good exploration possibilities.

Self-organizing maps are trained in an unsupervised manner (i.e. no class information is provided) from a set of high-dimensional sample vectors. The network structure has two layers (see Figure 1). The neurons in the input layer correspond to the input dimensions. The output layer (map) contains as many neurons as clusters needed. All neurons in the input layer are connected with all neurons in the output layer. The weights of the connection between input and output layer of the neural network encode positions in the high-dimensional data space. Thus, every unit in the output layer represents a prototype. Before the learning phase of the network, the two-dimensional structure of the output units is fixed and the weights are initialized randomly. During learning, the sample vectors are repeatedly propagated through the network. The weights of the most similar prototype w_s (*winner neuron*) are modified such that the prototype moves toward the input vector w_i. As similarity measure usually the scalar product is used. The weights w_s of the winner neuron are modified according to the following equation: $\forall i: w'_s = w_s + \delta \cdot (w_s - w_i)$, where δ is a learning rate.

To preserve the neighborhood relations, prototypes that are close to the winner neuron in the two-dimensional structure are also moved in the same direction. The weight change decreases with the distance from the winner neuron. Therefore, the adaptation method is extended by a neighborhood function v:

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

where δ is a learning rate. By this learning procedure, the structure in the high-dimensional sample data is non-linearly projected to the lower-dimensional topology. After learning, arbitrary vectors (i.e. vectors from the sample set or prior `unknown' vectors) can be propagated through the network and are mapped to the output units. For further details on self-organizing maps see [11].

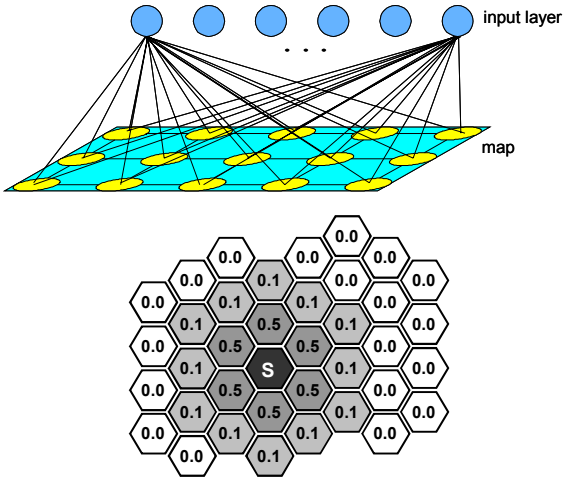


Figure 1. Structure of a rectangular self-organizing map (top) and possible neighborhood function for a structure based on hexagons (bottom).

Unfortunately, the standard model of self-organizing maps requires a predefined map structure. Therefore, the complete learning process has to be repeated if the size of the map was to small (the classification error which can be defined by $| w_s - w_i |$ for every pattern is usually very high and thus very dissimilar vectors are assigned to the same unit) or to large (similar vectors spread out on the map).

Growing self-organizing map approaches try to solve this problem by a learning method which modifies the size (and structure) of the map by adding new units to the map, e.g. if the accumulated error on a map unit increases a specified threshold. In the following the approach which is used in the presented application is briefly described.

2.1. A growing self-organizing map approach

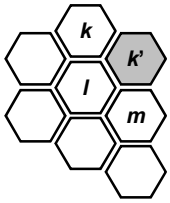
The proposed method is mainly motivated by the growing self-organizing map models presented in [1, 4]. In contrast to these approaches we use hexagonal map structure and restrict the algorithm to add new units to the external units if the accumulated error of a unit exceeds a specified threshold value.

The algorithm can be briefly described as follows:

1. Predefine the initial grid size (usually 2×2 units)
2. Initialize the assigned vectors with randomly selected values. Reset error values e_i for every unit i.

3. Train the map using all inputs patterns for a fixed number of iterations. During training increase the error values of a winner unit *s* by the current error value for pattern *i*.
4. Identify the unit with the largest accumulated error.
5. If the error does not exceed a threshold value stop training.
6. Identify the external unit *k* with the largest accumulated error.
7. Add a new unit to the unit *k*. If more than one free link is available select the unit at the nearest position to the neighboring unit which is most dissimilar to *k*. Initialize the weights of the new unit with respect to the vectors of the neighboring units so that the new vector is smoothly integrated into the existing vectors (see Figure 2).
8. Continue with step 3.
9. Continue training of the map for a fixed number of iterations. Reduce the learning rate during training.

This process creates an incremental growing map and it also allows training the map incrementally by adding new documents, since the training algorithms affects mainly the winning units to which new documents are assigned. If these units accumulate high errors, which means that the assigned documents cannot be classified appropriately, this part of the map starts to grow. Even if the consider neuron is an inner neuron, than the additional documents pushes the prior assigned documents to outer areas to which new neurons had been created. This can be interpreted, e.g. as an increase in publications concerning a specific topic. Therefore also dynamic changes can be visualized by comparing maps, which were incrementally trained by newly published documents.



k: winner unit
l, m: neighboring units
k': new unit
Computation of new weight vector for *k'*:
$w_{k'} = w_k + w_m - w_l$

Figure 2. Example of a computation of a vector for a new inserted unit

3. Building a map of a document collection

For the training of self-organizing maps, the documents must be encoded in form of numerical vectors. To be suited for the learning process of the map, to similar documents similar vectors have to be assigned, i.e. the vectors have to represent the document content. After training of the map, documents with similar contents should be close to each other, and possibly assigned to the same neuron. So, when a user has discovered a document of interest on the map, he or she can search the surrounding area.

The presented approach is based on statistical evaluations of word occurrences. We do not use any information on the *meaning* of the words since in domains like scientific research we are confronted with a wide and (often rapidly) changing vocabulary, which is hard to catch in fixed structures like manually defined thesauri or keyword lists. However, it is important to be able to calculate significant statistics. Therefore, the number of considered words must be kept reasonably small, and the occurrences of words sufficiently high. This can be done by either removing words or by grouping words with equal or similar meaning. A possible way to do so is to filter so-called *stop words* and to build the stems of the words (see e.g. [3]).

Although these document preprocessing steps are well known, they are still rarely used in commercially available document retrieval approaches or search engines. An overview of document pre-processing and encoding is given in Figure 3.

3.1. Stemming and filtering

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

Stemming tries to build the basic forms of words, i.e. strip the plural 's' from nouns, the 'ing' from verbs, or other affixes. A stem is a natural group of words with equal (or very similar) meaning. We currently used the stemming algorithm of [16], which uses a set of production rules to iteratively transform (English) words into their stems.

For the further reduction of relevant words we use two alternative approaches. The first reduces the vocabulary to a set of index words. These words are not selected manually, but automatically chosen by an information theoretic measure. The second approach is based on the work discussed in [17] and [6]. It uses a self-organizing map to build clusters of similar words, where *similarity* is defined based on a statistical measure over the word's context.

3.2. Selection of index words based on their entropy

For each word *a* in the vocabulary we calculate the entropy as defined by [14]:

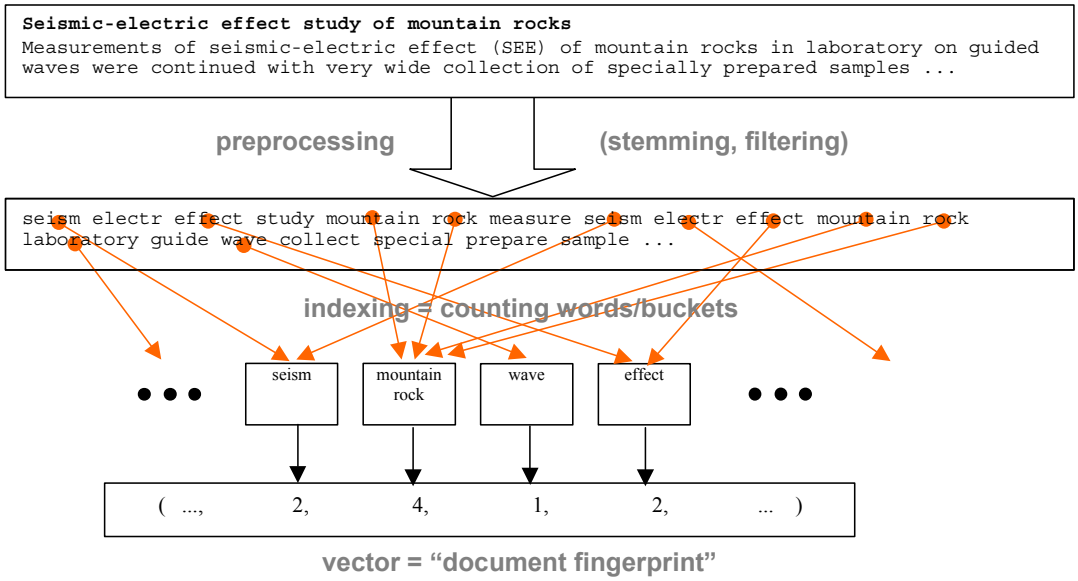


Figure 3. Document pre-processing and encoding

$$W(a) = 1 + \frac{1}{\ln(m)} \sum_{i=1}^{m} p_i(a) \cdot \ln(p_i(a)) \quad \text{with}$$

$$p_i(a) = \frac{n_i(a)}{\sum_{j=1}^{m} n_j(a)},$$

$n_i(a)$ is the frequency of word a in document i and m is the number of documents in the document collection. Here, the entropy gives a measure how well a word is suited to separate documents by keyword search. E.g. words that occur in many documents will have low entropy. The entropy can be seen as a measure of importance of words in the given domain context. We choose a number of words that have a high entropy relative to their overall frequency (i.e. from words occurring equally often we prefer those with the higher entropy). This procedure has empirically been found to yield a set of relevant words that are suited to serve as index terms.

3.3. The word category map

This approach does not reduce the number of words by removing irrelevant words from the vocabulary, but by building groups of words which are frequently used in similar (three-word-)contexts. A self-organizing map is used to find appropriate clusters of words. To be able to use words for training of a self-organizing map, the words have to be encoded. Therefore, to every word a random vector $\vec{w}$ with 90 dimensions is assigned (discussions concerning the number of dimensions can be found in [5]). This encoding does not imply any word ordering, as random vectors of dimensionalities that high can be shown to be 'quasi-orthogonal': the scalar product for nearly every pair of words is approximately zero. Then, the three-word-context of a word a is encoded by calculating the element-wise mean vectors of the words before $\vec{w}_{before}$ and after $\vec{w}_{after}$ the considered word over all documents and all occurrences of a. These mean (or expectation value) vectors $< \vec{w}_{before} >$ and $< \vec{w}_{after} >$ over the random vectors of enclosing words are used to define the context vector $\vec{w}_c$ of the considered word: $\vec{w}_c = (< \vec{w}_{before} >, \vec{w}, < \vec{w}_{after} >)$.
The obtained context vectors have 270 (=3·90) dimensions. Words a, b that often occur in similar contexts have similar expectation values and therefore similar context vectors $\vec{w}_c^{(a)}$, $\vec{w}_c^{(b)}$. The vectors $\vec{w}_c$ are finally clustered on a two-dimensional hexagonal grid using a self-organizing map. Words that are used in similar contexts are expected to be mapped to the same or to nearby neurons on this so-called word category map. Thus, the words in the vocabulary are reduced to the number of clusters given by the size of the word category map. Instead of index terms, the word categories *buckets* are used for the document indexing.
The most apparent advantage of this approach over the index term approach is that no words are removed from the vocabulary. Thus, all words are considered in the document clustering step. Furthermore, the word category map can be used as an expedient for the visual exploration of the document collection, because one often finds related words clustered together in the same or adjacent neurons of the word category map. From these clusters the user may choose related keywords which are appropriate for a (new) keyword search to reduce (or increase) the number of considered documents. However, due to the statistical peculi-

arities of the approach and the rather weak semantic clues the context vectors give, there are often additional words in the clusters that stand in no understandable relation to the others. The main drawback of this approach is that the words in one cluster become indistinguishable for the document indexing.

3.4. Generating characteristic document vectors

Figure 3 shows the principle of the proposed document encoding. At first, the original documents are preprocessed, i.e. they are split into words, then stop words are filtered and the word stems are generated (Sect. 3.1). Afterwards the considered vocabulary is reduced to a number of groups or *buckets*. These buckets are the index words from Sect. 3.2 or the word category maps from Sect. 3.3. The words of every document are then sorted into the buckets, i.e. the occurrences of the word stems associated with the buckets are counted. Each of the *n* buckets builds a component in a *n*-dimensional vector that characterizes the document. These vectors can be seen as the *fingerprints* of each document.

For every document in the collection such a fingerprint is generated. Using a self-organizing map, these document vectors are then clustered and arranged into a hexagonal grid, the so-called *document* map. Furthermore, each grid cell is labeled by a specific keyword that describes the content of the assigned documents.

The labeling method we used is based on methods proposed in [12]. It focuses on the distribution of words used in the documents assigned to the considered grid cell compared to the whole document database. The labeled map can then be used in visual exploration of the document collection, as shown in the following section.

4. Using the maps to explore document collections

To assess the usability of this approach a software prototype has been developed. The interactive user interface has been implemented in Java. The tool processes the documents as described above and stores the indexes and maps in a simple database. So we finally have a document map, where similar documents are grouped, and a word category map (if this approach is chosen) where the grouping of words is shown. In Figure 4 a screenshot of the software tool is shown.

The document map opens up several appealing navigation possibilities. Most important, the surrounding grid cells of documents known to be interesting can be scanned for further (similar) documents. Furthermore, a keyword search method has been implemented that – besides providing an ordered result set – visualizes the distribution of keyword search results by coloring the grid cells of the document map with respect to the number of hits for specific keywords or combinations of keywords. This allows a user to judge e.g. whether the search results are assigned to a small number of (neighboring) grid cells of the map, or whether the search hits are spread widely over the map and thus the search was – most likely – too unspecific and should be further refined.

If the highlighted nodes build clusters on the map we can suppose that the corresponding search term was relevant for the neighborhood relations in the learning of the self-organizing map. In this case the probability to find documents with similar topics in adjacent nodes can be expected to be higher.

Furthermore, the labels (index terms) assigned to the grid cells can be used to search for specific topics and thus supports the user in navigating through the document collection.

4.1. Using the word category map

The word category map can be used e.g. to look for related keywords for searching. If, for example, the number of search hits seems to be very small, so we would like to broaden our search. On the other hand, we would like the query to be still specific. In the word category map we can visualize the fingerprints of the matching documents. The highlighted nodes give us visual hints on which important keywords the document contains in addition to those keywords we have been searching for. Furthermore, we may find groups of documents with visually similar fingerprints (i.e. similar highlighted regions) and thus similar content. Therefore, we are supported in finding some keywords which describe the document content and which can then be used to refine the search by adding (or prohibiting) these keywords.

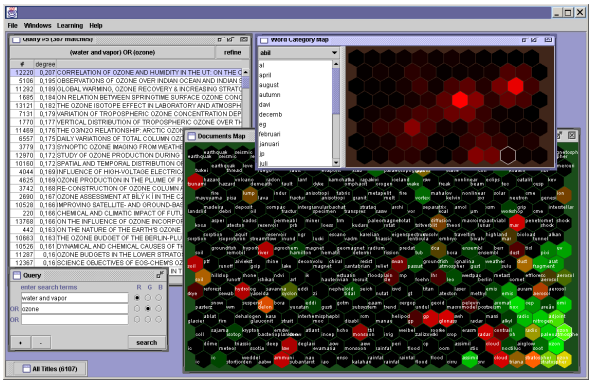


Figure 4. Screenshot of the software tool: Ranked search results (top left), word category map (top right), query window (bottom left), and document map with colored grid cells according to search results (bottom right)

5. Conclusions

The presented approach enables a user to search for specific documents, but also to enlarge obtained result sets (without the need to redefine search terms) by navigating through groups of documents with similar contents surrounding the search hits. Furthermore, the user is supported in finding appropriate search keywords to reduce or increase the documents under consideration by using a word category map, which groups together words used in similar contexts.

The methods proposed in this article combine (iterative) keyword search with grouping of documents based on a similarity measure in an interactive environment without the need to manually define lists of index terms or a classification hierarchy, which usually require expensive maintenance. Especially in rapidly changing document collections – like collections of publications of scientific research – classification systems that are not frequently updated are usually not accepted by the users, for whom especially new topics are of high importance.

Acknowledgements

The work presented in this article was partially supported by BTexact Technologies, Adastral Park, Martlesham, UK.

References

[1] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery, *IEEE Transactions on Neural Networks,* 11(3), pp. 601-614, 2000.

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

[3] W. B. Frakes, and R. Baeza-Yates, *Information Retrieval: Data Structures & Algorithms,* Prentice Hall, New Jersey, 1992.

[4] B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, *Neural Networks,* 7(9), pp. 1441-1460, 1994.

[5] T. Honkela, Self-Organizing Maps in Natural Language Processing, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 1997.

[6] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, Newsgroup Exploration with the WEBSOM Method and Browsing Interface, Technical Report, In: Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 1996.

[7] C. L. Isbell, and P. Viola, Restructuring sparse high dimensional data for effective retrieval, In: *Proc. of the Conference on Neural Information Processing (NIPS'98),* pp. 480-486, 1998.

[8] S. Kaski, Dimensionality reduction by random mapping: Fast similarity computation for clustering, In: *Proc. Of the International Joint Conference on Artificial Neural Networks (IJCNN'98),* pp. 413-418, IEEE, 1998.

[9] A. Klose, A. Nürnberger, R. Kruse, G. K. Hartmann, and M. Richards, Interactive Text Retrieval Based on Document Similarities, *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy,* 25(8), pp. 649-654, Elsevier Science, Amsterdam, 2000.

[10] T. Kohonen, Self-Organized Formation of Topologically Correct Feature Maps, *Biological Cybernetics,* 43, pp. 59-69, 1982.

[11] T. Kohonen, *Self-Organization and Associative Memory,* Springer-Verlag, Berlin, 1984.

[12] K. Lagus, and S. Kaski, Keyword selection method for characterizing text document maps, In: *Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks,* pp. 371-376, IEEE, 1999.

[13] T. K. Landauer, P. W. Foltz, and D. Laham, An Introduction to Latent Semantic Analysis, *Discourse Processes,* 25, pp. 259-284, 1998.

[14] K. E. Lochbaum, and L. A. Streeter, Combining and comparing the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval, *Information Processing and Management,* 25(6), pp. 665-676, 1989.

[15] A. Nürnberger, A. Klose, R. Kruse, G. Hartmann, and M. Richards, Interactive Text Retrieval Based on Document Similarities, In: G. Hartmann, A. Nölle, M. Richards, and R. Leitinger (eds.), *Data Utilization Software Tools 2 (DUST-2 CD-ROM),* Max-Planck-Institut für Aeronomie, Katlenburg-Lindau, Germany, 2000.

[16] M. Porter, An algorithm for suffix stripping, *Program,* pp. 130-137, 1980.

[17] H. Ritter, and T. Kohonen, Self-organizing semantic maps, *Biological Cybernetics,* 61(4), 1989.

[18] G. Salton, A. Wong, and C. S. Yang, A vector space model for automatic indexing, *Communications of the ACM,* 18(11), pp. 613-620, (see also TR74-218, Cornell University, NY, USA), 1975.