
Adaptive Multimedia Retrieval: From Data to User Interaction

Andreas Nürnberger¹ and Marcin Detyniecki²

¹ University of Magdeburg, FIN IWS - IR Group, 39106 Magdeburg, Germany
`nuernb@iws.cs.uni-magdeburg.de`

² Laboratoire d'Informatique de Paris 6, CNRS 75015 Paris, France
`Marcin.Detyniecki@lip6.fr`

Summary. To improve today's multimedia retrieval tools and thus the overall satisfaction of a user, it is necessary to develop methods that are able to support the user in the retrieval process, e.g. by providing not only additional information about the search results as well as the data collection itself, but also by adapting the retrieval tool to the underlying data as well as to the user's needs and interests. In this chapter we give a brief overview of the state-of-the-art and current trends in research.

1 Introduction

During the last years several approaches have been developed that tackle specific problems of the multimedia retrieval process, e.g. feature extraction methods for multimedia data, problem specific similarity measures and interactive user interfaces. These methods enable the design of efficient retrieval tools if the user is able to provide an appropriate query. However, in most cases the user needs several steps in order to find the searched objects. The main reasons for this are on the one hand, the problem of users to specify their interests in the form of a well-defined query – which is partially caused by inappropriate user interfaces –, on the other hand, the problem of extracting relevant features from the multimedia objects. Furthermore, user specific interests and search context are usually neglected when objects are retrieved.

To improve today's retrieval tools and thus the overall satisfaction of a user, it is necessary to develop methods that are able to support the user in the search process, e.g. by providing additional information about the search results as well as the data collection itself and also by adapting the retrieval tool to the user's needs and interests.

In the following, we give an overview of methods that are used in retrieval systems for text and multimedia data. To give a guideline for the development of an integrated system, we describe methods that have been successfully used

by the authors to develop an adaptive retrieval system for multimedia data. Thus, we will all along this chapter follow the example of a system based on growing self organizing maps in order to illustrate the mentioned aspects. Furthermore, we give hints for improvement and point out related projects. The sections of this chapter, reflect the different steps necessary to develop a multimedia retrieval system, from initial feature extraction to interface design and user modelling.

The design of any retrieval system could and should be decomposed in three successive design steps (see Fig. 1): Preprocessing and indexing, querying and visualization, and finally interface design and user modelling.

In the first step features are extracted and indexes should be created. It is important to index, because on the one hand we work with large amounts of data and it is usually impossible to browse the whole data set in a reasonable time in order to locate what we are looking for. On the other hand, in case of non-textual-data, the indexing process aims to extract a semantic meaning of the visual or audio data available. This is particularly critical for the interaction with human users. The feature extraction is a crucial, but non-trivial process and therefore several research teams in the world focus on just this part. In Sect. 2 we present briefly the state of the art of automatic multimedia indexing tools and of optimized indexing techniques for text.

In the second step of the retrieval system design, we can start thinking on how to interact with the user. We distinguish three different ways of interaction: Querying, navigation through structures and visualization. In Sect. 3 we present the most simple and common way of interaction: Querying. This action can be decomposed in three successive steps. First the query formulation by the user, then how to find the information in the database and finally how to aggregate and rank the results. Each of these steps should be well thought-out and designed. Another common way to help a user in the retrieval task is to structure the data set. Again, because of the volume of data, we are looking for automatic techniques. One family of algorithms known as clustering techniques allow to find groups of similar objects in data. In Sect. 4, after a short introduction and an overview of clustering techniques, we focus on one particular method: The growing self-organizing map. The third way of supporting a user in his task is visualization. Sophisticated visual interaction can be very useful, unfortunately it is rarely exploited. In Sect. 5, we first present briefly the state of the art of visualization techniques and then we focus on self-organizing maps, which are particularly suited for visualization.

The final step of retrieval system design is the integration and coordination of the different tools. This step is usually considered as interface design. We dedicate Sect. 6 to present some important aspects of it.

Once we have designed the retrieval system based on the above mentioned criteria, we can focus on personalization. The idea is to adapt the system to an individual user or a group of users. This adjustment to the user can be considered for every type of interaction. All personalization is based on a model of the user, which can be manually configured or learned by analyzing

user behavior and user feedback. User feedback can be explicitly requested or – if the system is suitable designed – learned from the interaction between system and user. Finally, we discuss in Sect. 7 as exemplification, after a short state of the art of user modeling approaches, personalization methods for self-organizing maps.

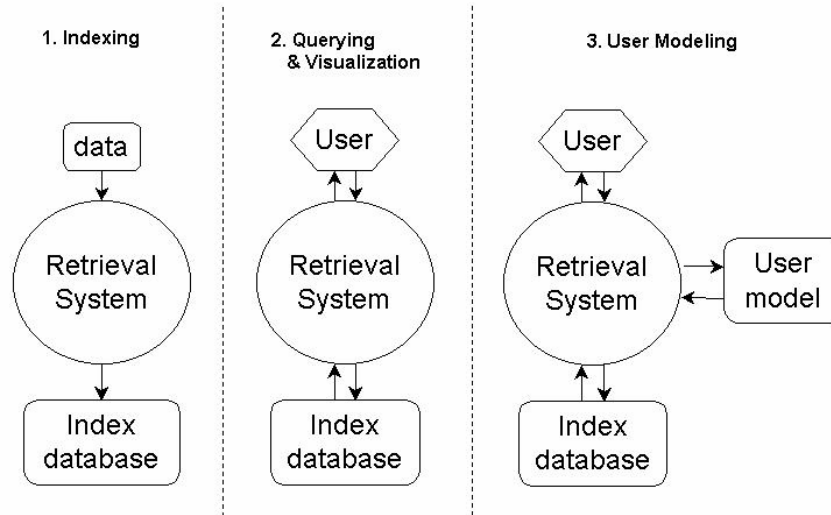


Fig. 1. Design Steps for Multimedia Retrieval Systems

2 Feature Extraction and Data Preprocessing

One of the major difficulties that multimedia retrieval systems face is that they have to deal with different forms of data (and data formats), as for instance text (e.g. text, html), hyperlinks (e.g. html, pdf), audio (e.g. wav, midi), images (e.g. jpeg, gif), video (e.g. mpeg, avi). Not only this polymorphism is a problem, but also the fact that the information is not directly available. In fact, it is not like a text file, where we can simply index by using the words appearing in the text. Usually, we have to index by annotating media data – manually or automatically –, in order to interact or work with them.

Formally, indexing is the process of attaching content-based labels to the media. For instance, existing literature on video indexing defines video indexing as the process of extracting the temporal location of a feature and its value from the video data. Currently indexing in the area of video is generally done manually. But since the indexing effort is directly proportional to the granularity of video access and since the number of videos available grows

and new applications demand fine grained access to video, automation of the indexing process becomes more and more essential. Presently it is reasonable to say that we can extract automatically the characteristics described in the following.

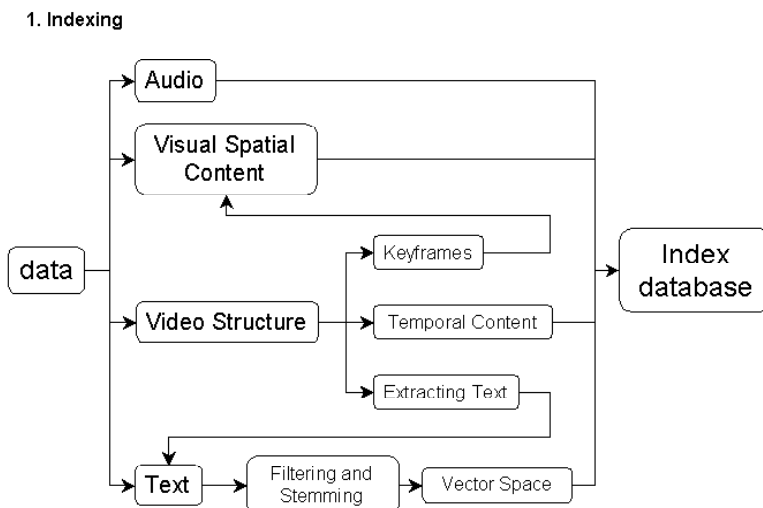


Fig. 2. The Process of Multimedia Indexing

2.1 Visual Spatial Content

From an image, we can extract color, texture, sketch, shape, objects and their spatial relationships. The color feature is typically represented by image histograms. The texture describes the contrast, the uniformity, the coarseness, the roughness, the frequency, and the directionality. In order to obtain this features either statistical techniques are used (autocorrelation, co-occurrence matrix) or spectral techniques as for instance detection of narrow peaks in the spectrum. The sketch gives an image containing only the object outlines and it is usually obtained by the combination of edge detection, thinning, shrinking and other transformations of this type. The shape describes global features as the circularity, eccentricity and major axis orientation, but also local ones such as for instance point of curvature, corner location, turning angles and algebraic moments. Note that the images can be already the result of an extraction, as for instance the key-frames, which are representative images from the visual stream (see video segmentation). Unfortunately, it is not yet possible to reliably extract a semantic content description of an image, even though several research groups are working on this problem.

2.2 Audio

From the audio stream, we can extract the following basic characteristics: The loudness as the strength of sound, determined by the amplitude of the sound wave. Using sound waves, loudness is dependant upon variations in pressure and the time rate at which sound energy flows through a defined area. The frequency translates what we perceive as pitch. The timbre is the characteristic distinguishing sounds from different sources. We can easily follow a particular timbre, without recognizing them.

More sophisticated characteristic can be obtained as for instance speaker tracking. To track a speaker means that we look for the person who speaks. Usually, at the beginning, a voice is learnt and then we recognize it during the whole audio stream. Most models for audio recognition use hidden Markov chains [1]. Another interesting feature is to recognize when noise, music, speech, or silence is predominant. Different approaches exist such as the use of expert systems [2], hidden Markov chains [3] or intelligent aggregation of specialized detectors.

2.3 Video Structure

We can simply extract all images and the sound track from a video or we can be more specific with respect to the video format and extract information regarding the structure by video segmentation and by extracting a representative image (called key-frame) from a segment. The purpose of video segmentation is to partition the video stream into basic units (shots) in order to facilitate indexing, browsing and to give some structure similar to paragraphs in a text document. Current techniques allow not only to find the shots, but also to describe the transition between them, as for instance fade in, fade out, dissolve or wipe.

Videos are stored either uncompressed or compressed, e.g. using MPEG or any other video compression method. The techniques for segmentation used in the uncompressed domain are based on pixel-wise or histogram comparison [4]. In the compressed domain [5] they are based either on coefficient manipulations as inner product or absolute difference or on the motion vectors. The key-frames are usually extracted to reduce the image processing to one image per shot. The idea is to find a representative frame (containing characteristic features) from the sequence of video frames. One simple method consists in extracting the first or the tenth frame [5]. More sophisticated methods look for local minima of motion or significant pauses [6]. Lately, expert systems having rules based on the camera motion have been proposed.

Video Visual Temporal Content

One important characteristic of the video is its temporal aspect. We can easily extract the following motions: The camera motion describes the real (translation and rotation) and factual movement of the camera (zoom in and out). It

is usually obtained either by studying the optical flow by dividing the video in several regions [7] or by studying the motion vector [8]. The object motion describes the trajectory obtained by tracking one object on the screen [9, 10].

Extracting Text

Since we are able to use text very efficiently to retrieve objects from a collection (see the following section 2.4), a lot of research is currently done in order to synchronize text with or to extract text from the video. One research direction is to synchronize the written script with the video [11]. Another one is to extract the text by doing a transcription of the audio channel [12]. This is usually done with complex speech recognition techniques on pre-processed data. Another interesting challenge is to extract the written information appearing on the screen [13, 14]. The idea is to locate the text on the screen and then to recognize it. These techniques are first attempts, but very often the synchronized text is available and can be exploited.

2.4 Text

For searching and navigation in large document collections it is necessary to pre-process the documents and store the information in a data structure, which is more appropriate for further processing than an unstructured text file. The currently predominant approaches are the vector space model [15], the probabilistic model [16], the logical model [17] and the Bayesian net model [18]. Despite of its simple data structure without using any explicit semantic information, the vector space model enables very efficient analysis of huge document collections and is therefore still used in most of the currently available document retrieval systems. The most popular retrieval methods that are based on this model are – besides the direct use of the vector description in an inverse index – Latent Semantic Indexing (LSI) [19], Random Projection [20] and Independent Component Analysis (ICA) [21]. The vector space description is also used for self-organizing maps. In the following we briefly describe the vector space model and appropriate document encoding techniques.

The Vector Space Model

The vector space model represents documents as vectors in t -dimensional space, i.e. each document i is described by a numerical feature vector $D_i = x_1, \dots, x_t$. Thus, documents can be compared by use of simple vector operations and even queries can be performed by encoding the query terms similar to the documents in a query vector. The query vector can then be compared to each document and a result list can be obtained by ordering the documents according to the computed similarity [22]. The main task of the vector space representation of documents is to find an appropriate encoding of the feature vector.

Each element of the vector usually represents a word (or a group of words) of the document collection, i.e. the size of the vector is defined by the number of words (or groups of words) of the complete document collection. The simplest way of document encoding is to use binary term vectors, i.e. a vector element is set to one if the corresponding word is used in the document and to zero if the word is not. This encoding will result in a simple Boolean search if a query is encoded in a vector. Using Boolean encoding the importance of all terms for a specific query or comparison is considered as similar. To improve the performance usually term weighting schemes are used, where the weights reflect the importance of a word in a specific document of the considered collection. Large weights are assigned to terms that are used frequently in relevant documents but rarely in the whole document collection [23]. Thus a weight w_{ik} for a term k in document i is computed by term frequency tf_{ik} times inverse document frequency idf_k , which describes the term specificity within the document collection. In [22] a weighting scheme was proposed that has meanwhile proven its usability in practice. Besides term frequency and inverse document frequency – defined as $idf_k := \log(N/n_k)$ –, a length normalization factor is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

$$w_{ik} = \frac{tf_{ik} \log(N/n_k)}{\sqrt{\sum_{j=1}^t (tf_{ij})^2 (\log(N/n_j))^2}}, \quad (1)$$

where N is the size of the document collection C and n_k the number of documents in C that contain term k .

Based on a weighting scheme a document i is defined by a vector of term weights $D_i = \{w_{i1}, \dots, w_{ik}\}$ and the similarity S of two documents (or the similarity of a document and a query vector) can be computed based on the inner product of the vectors (by which – if we assume normalized vectors – the cosine between the two document vectors is computed), i.e.

$$S(D_i, D_j) = \sum_{k=1}^m w_{ik} \cdot w_{jk}. \quad (2)$$

For a more detailed discussion of the vector space model and weighting schemes see, e.g. [24, 25, 23, 15].

Filtering and Stemming

To reduce the number of words and thus the dimensionality of the vector space description of the document collection, the size of the dictionary of words describing the documents can be reduced by filtering stop words and by stemming the words used. The idea of stop word filtering is to remove words that bear little or 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, and

also words that occur very seldom are likely to be of no particular statistical relevance and can be removed from the dictionary [26]. Stemming methods try 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. After the stemming process, every word is represented by its stem in the vector space description. Thus a feature of a document vector D_i now describes a group of words. A well-known stemming algorithm has been originally proposed by Porter [27]. He defined a set of production rules to iteratively transform (English) words into their stems.

Automatic Indexing

To further decrease the number of words that should be used in the vector description also indexing or keyword selection algorithms can be used (see, e.g. [19, 28]). In this case, only the selected keywords are used to describe the documents. A simple but efficient method for keyword selection is to extract keywords based on their entropy. E.g. in the approach discussed in [29], for each word a in the vocabulary the entropy as defined by [30] was calculated:

$$W(a) = 1 + \frac{1}{\ln(m)} \sum_{i=1}^m p_i(a) \cdot \ln(p_i(a)) \text{ with } p_i(a) = \frac{n_i(a)}{\sum_{j=1}^m n_j(a)}, \quad (3)$$

where $n_i(a)$ 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 the importance of a word in the given domain context. As index words a number of words that have a high entropy relative to their overall frequency can be chosen, i.e. from words occurring equally often those with the higher entropy can be preferred. This procedure has empirically been found to yield a set of relevant words that are suited to serve as index terms [29].

2.5 Hints and Tips

- Feature extraction and data preprocessing are essential and fundamental steps in the design of a retrieval system. The feasibility of the following steps and the quality of the retrieval system directly depend on it.
- In general the difficulty and the time required to obtain good indexes is often underestimated. In particular, for real multimedia objects this task is even more complex, because we have not only very sophisticated indexing processes per media, but also we have to coordinate the cross-indexes. Therefore, very often multimedia retrieval systems are only focused on a particular kind of data, either on image or sound or video.

- Whenever text is available do not hesitate to use it. Query results using text are particularly good compared to other features.
- Even if it is possible to use not preprocessed text, the results are highly improved if preprocessing is done.

Given the current state of the art, reliable and efficient automatic indexing is only possible for the presented low-level characteristics. But is clear that any intelligent interaction with the multimedia data should be based on a higher level of description. For instance, in the case of a video we are more interested in finding a dialog, than in finding a series of alternated shots. In case of an image you can always query for a specific color and texture, but is this really what you want to do?

The current intelligent systems use high level indexing as for instance a predefined term index or even ontological categories. Unfortunately, the high level indexing techniques are based on manual annotation. So, these approaches can only be used for small quantities of new video and do not exploit intelligently the automatic extracted information.

In addition, the characteristics extracted by the automatic techniques are clearly not crisp attributes (color, texture) or they are defined with a degree of truth (camera motion: fade-in, zoom out) or imprecise (30% noise, 50% speech). Therefore, fuzzy techniques seem to be a promising approach to deal with this kind of data (see, for example, [31]).

3 Querying

In information retrieval it is usually distinguished between searching by example (or content based searching) and feature based querying. Content based searching means that the user has a sample and is looking for similar objects in a document collection. Content based searching is usually performed by preprocessing the provided sample object to extract the relevant features. The features are then used to perform a query. Therefore, content based and feature based searching differ only in the way the features for searching are defined: In feature based searching the user has to provide the features, in content based searching the features are automatically extracted from the sample given.

Searching based on given features is performed by retrieving the objects in the collection that have similar features. In case of Boolean queries the features of the objects have to match exactly the criteria defined in the query. Therefore, the result set cannot be sorted by similarity (since all objects match the criteria exactly), but by other criteria like date, size or simply by alphabetical order. If we have indexed a text document collection using the vector space model as described in the previous section, we can search for similar documents by encoding the query terms in form of a query vector similar

to the documents. Then we can use the cosine measure (see Eq. 3) to compare the documents with the query vector and compute the similarity of each document to the query.

3.1 Ranking

Ranking defines the process of ordering a result set with respect to given criteria. Usually a similarity measure that allows to compute a numerical similarity value for a document to a given query is used. For text document collections ranking based on the vector space model and the $tf \times idf$ weighting scheme (Eq. 1) discussed in the previous section has proven to provide good results. Here the query is considered as a document and the similarity is computed based on the scalar product (cosine; see Eq. 2) of the query and each document (see also [23]).

If fuzzy modifiers should be used to define vague queries, e.g. using quantifiers like '*most of (term1,term2, ...)*' or '*term1 and at least two terms of (term2,term3, ...)*', ranking methods based on ordered weighted averaging operators (OWA) can be used [32]. If the document collection also provides information about cross-references or links between the documents, these information can be used in order to compute the 'importance' of a document within a collection. A link is in this case considered as a vote for a document. A good example is the PageRank algorithm [33] used as part of the ranking procedure by the web search engine Google. It has proven to provide rankings that sorts heavily linked documents for given search criteria in high list ranks.

3.2 Hints and Tips

- Users are used to querying systems. Therefore any retrieval system should have a simple keyword query interface.
- Very often the raw results of the similarity (or distance) computation are directly used to rank from high to low. However, a short analysis of the ranking process can highly improve the quality of the system, especially if diverse features are combined.

4 Structuring Document Collections

A structure can significantly simplify the access to a document collection for a user. Well known access structures are library catalogues or book indexes. However, the problem of manual designed indexes is the time required to maintain them. Therefore, they are very often not up-to-date and thus not usable for recent publications or frequently changing information sources like the World Wide Web.

The existing methods to automatically structure document collections either try to find groups of similar documents (clustering methods) or assign

keywords to documents based on a given keyword set (classification or categorization methods). Examples of clustering methods that are frequently used in information retrieval systems are k-means and agglomerative clustering [34, 35], self-organizing maps [36, 37] and fuzzy clustering methods [38]. To classify (or categorize) documents, we frequently find decision trees [34], probabilistic models [39] and neural networks [40, 41]. A recent softcomputing based approach to classify web documents was presented in [42]. For clustering and for classification non-hierarchical and hierarchical approaches exist, which enable the creation of a deep structure of the collection.

Hierarchical categorization methods can be used to organize documents in a hierarchical classification scheme or folder structure. To create a categorization model – if we neglect manual designed filters – a training set of already (e.g. manually) classified documents is required. If this information is not available we can use clustering methods to pre-structure the collection. Clustering methods group the documents only by considering their distribution in a document space (for example, a n -dimensional space if we use the vector space model for text documents). To each discovered group we can then assign a class label. However, this can be a tedious and time consuming task.

Meanwhile, methods are available that are able to use classified and unclassified documents to create a categorization scheme. The idea is to exploit the classification information of the pre-classified documents and the distribution of the document collections in document space. A recent approach – of these so-called semi-supervised clustering methods – for text categorization was presented in [43].

The approaches described above consider the documents itself as unstructured, i.e. they use only a feature vector created from each document. However, especially in web pages we could exploit structural information that marks headlines and paragraphs or even refers to other documents. The same holds for scientific contributions where usually references to different (additional or related) publications are given. These information can be provided to the user as additional information for navigation and search. An approach that exploits this idea was presented in [44].

The main problem of all these approaches is that they only consider information extracted from the underlying document collection and are not able to include user interests, e.g. based on keywords that define areas the user is interested in and that should be considered more important in the clustering process. We get back to this aspect in Sect. 7 when we discuss techniques for user modelling. In the following, we briefly describe self-organizing maps as an example for clustering methods.

4.1 An Example: Self-organizing Maps

Self-organizing maps (SOMs) are trained in an unsupervised manner, i.e. no class information is provided, using a set of high-dimensional sample vectors. The network structure has two layers (see Fig. 3). 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.

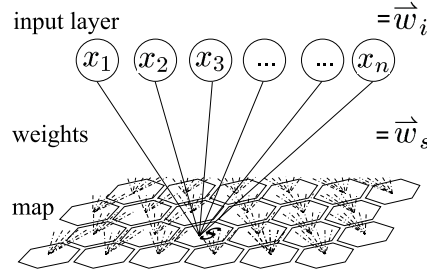


Fig. 3. Structure of a Self-Organizing Map Based on Hexagons

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 Euclidean distance or 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 strength of the modification 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 d 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. Finally, arbitrary vectors (i.e. vectors from the sample set or prior 'unknown' vectors) can be propagated through the trained network and are mapped to the output units. For further details on self-organizing maps see [45].

Unfortunately, the standard model of self-organizing maps requires a manual predefined map structure. Therefore, the size of the map is usually too small or too large to map the underlying data appropriately. If the size of the map was too small (the classification error for every pattern is usually very

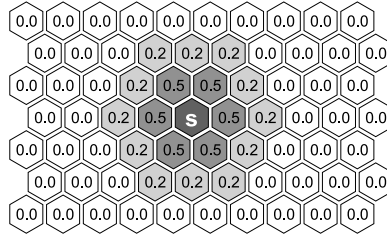


Fig. 4. Possible Neighborhood Function for a Self-Organizing Map

high and thus very dissimilar vectors are assigned to the same unit) or to large (similar vectors spread out on the map). Therefore, the complete learning process has to be repeated several times until an appropriate size is found. Growing self-organizing map approaches try to solve this problem by modifying the size (and structure) of the map by adding new units to the map, if the accumulated error on a map unit increases a specified threshold. Thus they adapt themselves to the structure of the underlying data collection. In the following we briefly describe the approach that we used in a prototypical retrieval system [46].

A Growing Self-Organizing Map Approach

The proposed method is mainly motivated by the growing self-organizing map models presented in [47, 48]. 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 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 Fig. 5).
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. Furthermore, it allows training the map incrementally by adding new data, since the training algorithm affects mainly the winning units to which new data vectors are assigned. If these units accumulate high errors, which means that the assigned patterns cannot be classified appropriately, this part of the map starts to grow. Even if the considered neuron is an inner neuron, then the additional data pushes the prior assigned patterns to outer areas to which new neurons had been created. This can be interpreted as an increase of the number of data items belonging to a specific area or cluster in data space, or if text documents are assigned to the map as an increased number of publications concerning a specific topic. Therefore also dynamic changes can be visualized by comparing maps, which were incrementally trained by, e.g. newly published documents [49].

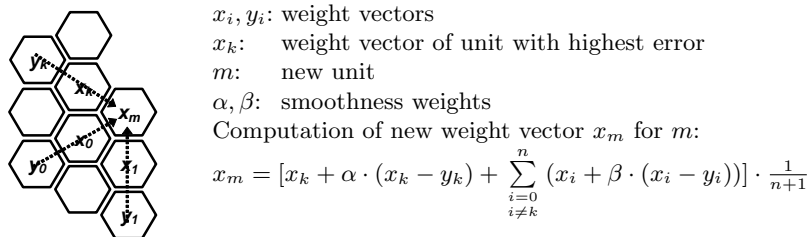


Fig. 5. Insertion of a New Unit

4.2 Hints and Tips

- Structuring document collections can be very useful for a user. The question is which clustering algorithm to use. In order to choose an appropriate algorithm we have to consider the complexity of the algorithm. This depends on the amount of data that has to be processed and the time available for computation, which might be limited in interactive applications.
- Hierarchical methods are usually more appropriate to support a user in navigation, but its sometimes hard to use them in order to visualize structure or provide an overview of a set of documents.
- We have chosen self-organizing maps, because of their good visualization capabilities, good usability in interactive application and the simplicity of the algorithm (only a few parameters have to be defined). Nevertheless, the time required for the computation of a map usually prevents learning of SOMs online. Thus, a SOM has to be computed before it is used in an interactive application.

5 Visualization Techniques

Graphical visualization of information frequently provides more comprehensive and better and faster understandable information than it is possible by pure text based descriptions. Therefore, in information retrieval systems additional visual information can improve the usability to a great extent. Information that allow a visual representation comprises aspects of the document collection or result sets, keyword relations, ontologies or aspects of the search process itself, e.g. the search or navigation path in hyperlinked collections.

However, especially for text collections we have the problem of finding an appropriate visualization for abstract textual information. Furthermore, an *interactive* visual interface is usually desirable, e.g. to zoom in local areas or to select or mark parts for further processing. This results in great demands on the user interface and the hardware. In the following we give a brief overview of visualization methods that have been realized for information retrieval systems.

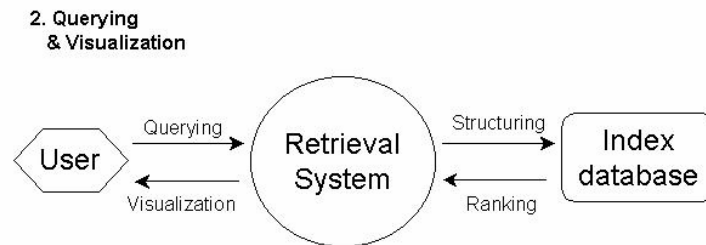


Fig. 6. Using Querying, Structuring and Visualization Techniques in a Multimedia Retrieval System

5.1 Visualizing Relations and Result Sets

Interesting approaches to visualize keyword-document relations are, e.g., the Cat-a-Cone model [50], which visualizes in a three dimensional representation hierarchies of categories that can be interactively used to refine a search. The InfoCrystal [51] visualizes a (weighted) boolean query and the belonging result set in a crystal structure. The lyberworld model [52] and the visualization components of the SENTINEL Model [53] are representing documents in an abstract keyword space. This idea was applied slightly modified to image databases in [54].

An approach to visualize the results of a set of queries was presented in [55]. Here, retrieved documents are arranged according to their similarity to a query on straight lines. These lines are arranged in a circle around a common

center, i.e. every query is represented by a single line. If several documents are placed on the same (discrete) position, they are arranged in the same distance to the circle, but with a slight offset. Thus, clusters occur that represent the distribution of documents for the belonging query.

5.2 Visualizing Document Collections

For the visualization of document collections usually two-dimensional projections are used, i.e. the high dimensional document space is mapped on a two-dimensional surface. In order to depict individual documents or groups of documents usually text flags are used, which represent either a keyword or the document category. Colors are frequently used to visualize the density, e.g. the number of documents in this area, or the difference to neighboring documents, e.g. in order to emphasize borders between different categories. If three-dimensional projections are used, for example, the number of documents assigned to a specific area can be represented by the z-coordinate.

An Example: Visualization using Self-Organizing Maps

Visualization of document collections requires methods that are able to group documents based on their similarity and furthermore that visualize the similarity between discovered groups of documents. Clustering approaches that are frequently used to find groups of documents with similar content [35] – see also Sect. 4 – usually do not consider the neighborhood relations between the obtained cluster centers. Self-organizing maps, as discussed above, are an alternative approach which is frequently used in data analysis to cluster high dimensional data. They cluster high-dimensional data vectors according to a similarity measure. The resulting clusters are arranged in a low-dimensional topology that preserves the neighborhood relations of the corresponding high dimensional data vectors. Thus, not only objects that are assigned to one cluster are similar to each other, but also objects of nearby clusters are expected to be more similar than objects in more distant clusters. Usually, two-dimensional arrangements of squares or hexagons are used for the definition of the neighborhood relations. Although other topologies are possible for self-organizing maps, two-dimensional maps have the advantage of intuitive visualization and thus good exploration possibilities. In document retrieval, self-organizing maps can be used to arrange documents based on their similarity. This approach 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, the distribution of keyword search results can be visualized by coloring the grid cells of the map with respect to the number of hits. 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.

A first application of self-organizing maps in information retrieval was presented in [56]. It provided a simple two-dimensional cluster representation (categorization) of a small document collection. A refined model, the WEBSOM approach, extended this idea to a web based interface applied to newsgroup data that provides simple zooming techniques and coloring methods [57, 58, 59]. Further extensions introduced hierarchies [60], supported the visualization of search results [37] and combined search, navigation and visualization techniques in an integrated tool [46]. A screenshot of the prototype discussed in [46] is depicted in Fig. 7.

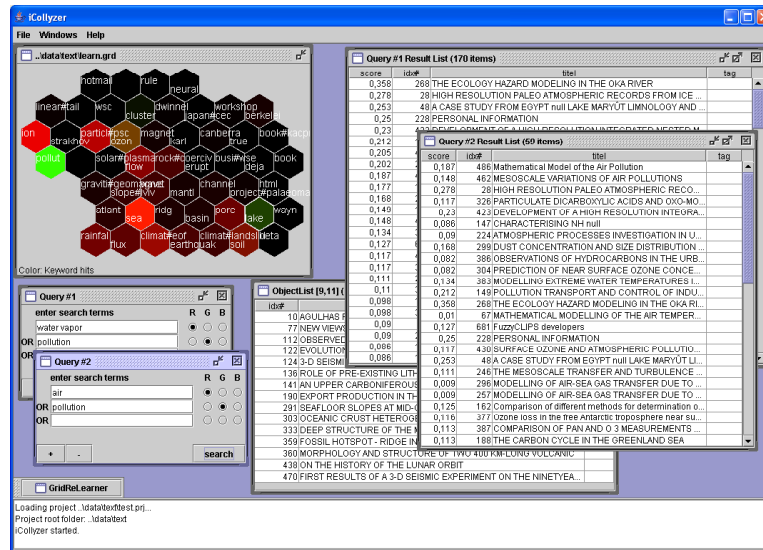


Fig. 7. A Prototypical Retrieval System Based on Self-Organizing Maps

Other Techniques

Besides methods based on self-organizing maps several other techniques have been successfully applied to visualize document collections. For example, the tool VxInsight [61] realizes a partially interactive mapping by an energy minimization approach similar to simulated annealing to construct a three dimensional landscape of the document collection. As input either a vector space description of the documents or a list of directional edges, e.g. defined based on citations of links, can be used. The tool SPIRE [62] applies a three step approach: It first clusters documents in document space, then projects the discovered cluster centers onto a two dimensional surface and finally maps the documents relative to the projected cluster centers. SPIRE offers a scatter

plot like projection as well as a three dimensional visualization. The visualization tool SCI-Map [63] applies an iterative clustering approach to create a network using, e.g., references of scientific publications. The tools visualizes the structure by a map hierarchy with an increasing number of details.

One major problem of most existing visualization approaches is that they create their output only by use of data inherent information, i.e. the distribution of the documents in document space. User specific information can not be integrated in order to obtain, e.g., an improved separation of the documents with respect to user defined criteria like keywords or phrases. Furthermore, the possibilities for a user to interact with the system in order to navigate or search are usually very limited, e.g., to boolean keyword searches and simple result lists. Therefore, we describe in Sect. 7 fundamental methods to include user preferences and feedback in the visualization interface.

5.3 Hints and Tips

- Visualization Techniques can be extremely useful for the user.
- There is no visualization technique that is appropriate for all kinds of users. Some users are used to simple lists or trees and reject to use complex visual output. Others prefer complex interactive navigation capabilities. So far, no general design rule can be given, but to keep the output as simple and intuitive as possible.
- The coding of theoretical ideas for visualization into a running system is often complicated and time consuming. Therefore we recommend not to underestimate the workload of this step.

6 General Hints and Tips for Interface Design

In information retrieval it is usually distinguished between three types of input interfaces: Command language oriented (like SQL), form or menu based and (pseudo) natural language based. In most current information retrieval systems a combination of form and natural language based interface is used, which is usually extended by a set of operators to combine or refine queries (e.g. Boolean operators). This combination is accepted by users and has proven to be reasonably efficient [64, 65].

The search results are usually listed in a ranked list that is sorted according to the similarity to the query. Depending on the type of the query and the underlying data collection different ranking methods can be applied (see Sect. 3.1). Besides a document title and direct access to the document itself, usually some additional information about the documents content is automatically extracted and shown. For example, if the underlying document is a personal homepage it might be helpful to list besides the name of the person, the affiliation and phone number in order to avoid that the user needs to perform an additional interaction with the system to get the desired information. This

is especially important, if the user can filter out documents based on the additional information without being required to scan the document and then going back to the result list, which is usually a time consuming task.

Furthermore, additional information on how a search can be refined can be given by providing additional keywords. This additional keywords can be derived based on thesauri or ontologies. Here research with respect to the semantic web might lead to further more refined techniques.

In order to improve the selection of the proposed keywords information about user interests can be used (for details see the following section). This can be done, for example, by selecting from a list of candidate keywords for refinement the words that have a high and words that have a very low importance in the user profile. The user profile information should also be used to change the order of the documents in the result list. Thus, the rank of a document should ideally be computed based on the degree of match with the query and information extracted from the user profile. Current web search engines usually sort hits only with respect to the given query and further information extracted from the web, e.g. based on a link analysis like PageRank (see [33]), a popularity weighting of the site or specific keywords. This results usually in an appropriate ordering of the result sets for popular topics and well linked sites. Unfortunately, documents dealing with special (sub) topics are ranked very low and a user has to provide very specific keywords in order to find information provided by these documents. Especially for this problem a user specific ranking could be very beneficial.

Another method to support a user in navigation is to structure result sets as well as the whole data collection using, e.g., clustering methods as discussed above. Thus, groups of documents with similar topics can be obtained (see, for example, [37]). If result sets are grouped according to specific categories, the user gets a better overview of the result set and can refine his search more easily based on the proposed classes.

7 User Modelling

The main objective of user modelling in the area of information retrieval is to extract and store information about a user in order to improve the retrieval performance. A model of the user usually consist of at least a list of keywords to which relevance degrees are assigned. More complex models distinguish between different search contexts, e.g. private and business, or store additionally relations between keywords in order to model a more expressive search context. A simple profile of user interests can be obtained, e.g., by extracting keywords from the queries performed or documents read by the user in the past.

User models are currently extensively used in the area of intelligent agents. For example, agents that adapt the content of a website by changing links or contents based on the classification of a user in a specific user group like

'fashion-conscious' or 'pragmatic'. The belonging agents that analyze the user behavior based on the web pages the user visits, are called profiling agents. By analyzing the content of the pages and the sequence in which the user visited the pages (click stream analysis), it is very often possible to identify the interests of a user. A good overview of methods in this area is given, for example, at <http://www.dfki.de/AgentSurvey/> and in [66].

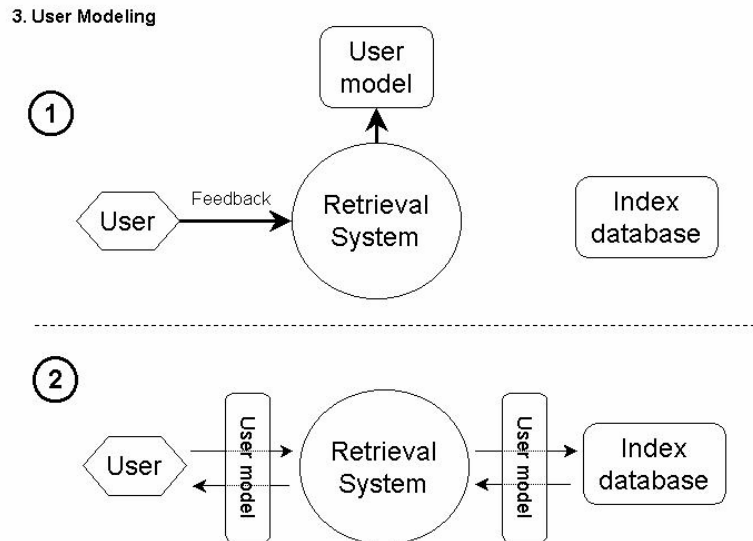


Fig. 8. User Profiling and Application of User Models in Retrieval Systems

7.1 User Feedback

User feedback is any response of a user to an output of the system. In information retrieval we are especially interested in feedback that can be used to learn something about the user (user profiling as described above) or feedback that can directly be used to improve the retrieval performance. For example, we can provide a user the possibility to mark relevant and not relevant objects of a result set. This is called *relevance feedback*. Based on this information a modified query can be derived automatically that makes use of information extracted from the tagged objects. This can be done by increasing the importance of keywords that are used in documents marked as relevant and decreasing the importance of keywords used in documents that are marked as not relevant. Thus, the result set can be iteratively refined. For details see Rochio [67] for the vector space model or [68] for the probabilistic model.

Almost all current approaches for relevance feedback are based on the ideas presented in these papers.

A model that utilizes user feedback to obtain a user profile was presented in [69]. This approach is also able to distinguish between long term interests of a user and ad hoc queries. Thus it avoids the undesired modification of queries that can not appropriately be described by the information stored in the user profile. The model was specifically designed for a Newsreader (NewsDude) that filters relevant information from different news sources. A Bayesian network is used in order to compute the user specific relevance of documents. The approach discussed in [70] follows a similar motivation in order to derive interest categories for users by an incremental clustering approach.

The tool WebWatcher [71] was designed to support a user by highlighting potentially relevant links on a web site. This approach uses a reinforcement learning approach to learn a profile based on the link usage of a group of users.

More general ideas concerning user and context modelling can be found in [72, 66]. As a more detailed example for the use of user feedback we discuss in the following an approach that has been integrated in a clustering and visualization tool for multimedia object collections.

7.2 Incorporating Weighting and User Feedback: An Exemplification Using Self-Organizing Maps

Self-organizing maps perform clustering of the objects in an unsupervised manner. Therefore it depends on the choice of describing feature vectors and the definition of similarity between them whether the resulting groups meet the users' expectations. This is especially true for multimedia data, where a general definition of significant features is rather hard and strongly depends on the application. However, the user has often a good intuition of a 'correct' clustering. Therefore it seems to be very important to incorporate user feedback information to optimize the classification behavior by gathering information about the desired similarity measure and hereby modifying, e.g., the importance of features using weights. To allow the user to give feedback information, a user can be allowed to drag one or several objects from one node on the map to another that in his opinion is more appropriate for the considered objects. Furthermore, the user can mark objects that should remain at a specific node, thus preventing the algorithm from moving them together with the moved object after re-computing the groups on the map. In the following, three user-feedback models are described that have been designed to solve specific clustering problems. All approaches implement some kind of semi-supervised learning, as they use classification information about some of the objects as given by the user. The first two approaches modify the underlying similarity measure by increasing or decreasing the importance of individual features. The idea of the third approach is to fine-tune or re-compute parts of the self-organizing map to guide the learning process in how

the high-dimensional feature space should be folded (i.e. non-linearly projected) into the two dimensions of the map. The methods are described in more detail in the following (see also [73, 74, 75]).

Learning a Global Feature Weighting

For the implementation of a global feature weighting scheme, we replaced the Euclidean similarity function used for the computation of the winner nodes by a weighted similarity measure. Therefore, the distance of a given feature vector to the feature vectors of the prototypes is computed by

$$e_s = \left(\sum_i w^i \cdot (x_s^i - y_k^i)^2 \right)^{\frac{1}{2}} \quad (4)$$

where w is a weight vector, y_k the feature vector of an document k and x_s the prototypical feature vector assigned to a node s .

We update the global-weight vector w based on the differences of the feature vectors of the moved document and the vectors of the origin node and of the target node. The goal is to increase the weights of similar features between the document and target node and to decrease the weights of similar features between the document and its current node. And symmetrically decrease and increase the weights of dissimilar features.

Let y_i be the feature vector of an document i , s be the source and t the target node, x_s and x_t be the corresponding prototypes, then w is computed as described in the following. First we compute an error vector e for each object based on the distance to the prototypes

$$e_{ji}^k = |d_{ji}^k| \forall k, \text{ where } d_{ji}^k = \frac{y_i - x_j}{\|y_i - x_j\|}. \quad (5)$$

If we want to ensure that an object is moved from the source node to the target node using feature weights, we have to assign higher weights to features that are more similar to the target than to the source node. Thus for each object we compute the difference of the distance vectors

$$f_i = e_{si} - e_{ti}. \quad (6)$$

The global weight vector is finally computed iteratively. For the initial weight vector we choose $w^{(0)} = w_1$, where w_1 is a vector where all elements are equal one. Then we compute a new global weight vector $w^{(t+1)}$ by doing a by element multiplication:

$$w^{k(t+1)} = w^{k(t)} \cdot w_i, \forall k \text{ with } w_i = (w_1 + \eta \cdot f_i), \quad (7)$$

where η is a learning rate. The global weight is modified until – if possible – all moved objects are finally mapped to the target node. A pseudocode description of this approach is given in Fig. 1.

Obviously, this weighting approach also affects the assignments of all other documents. The idea is to interactively find a feature weighting scheme that improves the overall classification performance of the map. Without a feature weighting approach the map considers all features equally important.

```

Compute the weight vectors  $w_i$ ;
If the global weight vector  $w$  is undefined
  create a vector and initialize all elements to one;
 $cnt = 0$ ;
Repeat until all documents are moved or  $cnt > max$ 
   $cnt + +$ ;
  For all documents  $i$  to be moved do
    Compute the winning node  $n$  for  $i$ ;
    if  $N \neq t_i$  (target for  $i$ ) then
       $w^k := w^k \cdot w_i^k, \forall k$ ;
      normalize  $w$ ;
    end if;
  end for;
end repeat;

```

Table 1. Pseudocode Description of the Computation of a Global Weight

The global weights can also be used to identify features that the user considers important or less important. If, for example, text documents are used where the features represents terms, then we might get some information about the keywords that the user seems to consider important for the classification of the documents.

Learning a Local Weighting Scheme

The global weighting scheme emphasizes on general characteristics, which support a good overall grouping of the data collection. Unfortunately, this may lead to large groups of cells with quite similar documents. In this case some features – which are of less importance on a global scope – might be useful for distinguishing between local characteristics. Thus, modifying locally the weights assigned to these features might improve the assignment of the documents to more specific *local* classes.

The proposed learning method used is quite similar to the method described above. However, instead of modifying the global weight w , we modify local weights assigned to the source and the target nodes (noted here w_s and w_t). As before, we first compute an error vector e for each document based on the distance to the prototypes, as defined in equation (2). Then we set all elements of the weight vectors w_s and w_t to one and compute local document weights w_{si} and w_{ti} by adding (subtracting) the error terms from the neutral

weighting scheme w_1 . Then we compute the local weights iteratively similar to the global weighting approach:

$$w_s^{k(t+1)} = w_s^{k(t)} \cdot w_{si} \forall k, \text{ with } w_{si} = w_1 + \eta \cdot e_{si} \quad (8)$$

and

$$w_t^{k(t+1)} = w_t^{k(t)} \cdot w_{ti} \forall k, \text{ with } w_{ti} = w_1 - \eta \cdot e_{ti}, \quad (9)$$

where η is a learning rate. The weights assigned to the target and source node are finally normalized such that the sum over all elements equals the number of features in the vector, i.e.

$$\sum_k w_s^k = \sum_k w_t^k = \sum_k 1. \quad (10)$$

In this way the weights assigned to features that achieved a higher (lower) error are decreased (increased) for the target node and vice versa for the source node.

A Generalized Learning Model

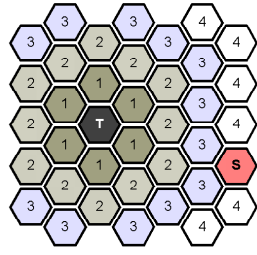
With the local approach we just modified weighting vectors of the source and target nodes. However, as adjacent map nodes should ideally contain similar documents, one could demand that the weights should not change abruptly between nodes. Thus, it is a natural extension of this approach to modify the weight vectors of the neighboring map units accordingly with a similar mechanism as in the learning of the map. Depending on the radius r of the neighborhood function, the result would lie between the local approach ($r = 0$) and the global approach ($r = \infty$). In the following, we present such an extension.

As for the local approach we have a weighting vector per node. Then – as before – we start by computing an error vector e for each object based on the distance to the prototypes, as defined in equation (2). Based on the error vectors e weight vectors of each node n are computed iteratively. For the initial weight vector $w_n^{k(0)}$ we choose vectors where all elements are equal to one. We then compute a new local weight vector for each node by an elementwise multiplication:

$$w_n^{k(t+1)} = w_n^{k(t)} \cdot w_{ni}, \forall k, \text{ with } w_{ni} = w_1 + \eta \cdot (g_{sn}^r \cdot e_{si} - g_{tn}^r \cdot e_{ti}) \quad (11)$$

where η is a learning rate and where g_{sn}^r and g_{tn}^r are weighting values calculated using a neighborhood function.

Because two similar prototypes can be projected in distant cells of the map, the neighborhood function should be based on the actual topology of the map. Therefore a linear decreasing function for g_{sn}^r , which equals one for the source node and equals zero at the hull defined by the radius r can be



$$g_{tn}^r = \begin{cases} 1 - \frac{\text{dist}(n,t)}{r} & \text{if } \text{dist}(n,t) < r, \\ 0 & \text{otherwise} \end{cases},$$

where $\text{dist}(x,y)$ is the radial distance in nodes between nodes x and y

Fig. 9. Neighborhood Function Centered on Target Node (decreasing to zero for r)

used. The same holds for the target node and r (see also Fig. 2). Notice that more refined functions can be used as for instance Gaussian-like functions.

As above, all weights vectors are modified until – if possible – all moved objects are finally mapped to the target node.

This weighting approach affects the assignments of documents of neighboring cells. The influence of the modification is controlled by the neighborhood function. The idea is that a local modification has a more global repercussion on the map. In this way we can interactively find a feature weighting scheme that improves the classification performance of the map.

We note here that the general approach has by construction the local and global approach as limiting models. In fact, depending on the radius r of the neighborhood function, we obtain the local approach for $r \rightarrow 0$ and the global approach for $r \rightarrow \infty$.

The methods described above have been integrated into a prototype for image retrieval shown in Fig. 10. Further details can be found in [74].

7.3 Hints and Tips

- The user modelling step should be integrated at the very end of the design process. All other functionalities of the system should be tested and evaluated before.
- A user model should contain at least a list of features or keywords describing the user interests.
- The user should have access to his profile in order to increase his confidence in the system.
- Particular care should be given on how to obtain user feedback. Should the user be requested directly or should we 'spy' on him? One hint to answer this question is to take into account that most users do not like to be disturbed too much by questions. This particular aspect has been the reason of several failures of besides carefully designed systems.

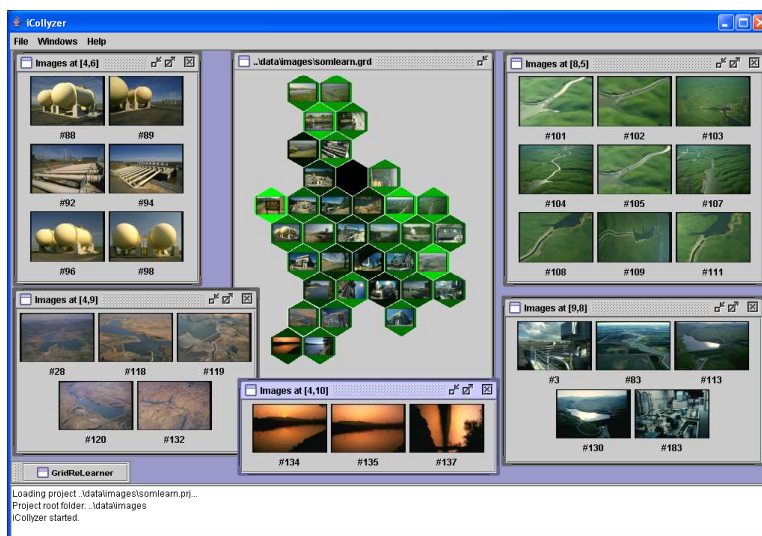


Fig. 10. A Prototypical Image Retrieval System: Overview of the Image Collection by a Map and Selected Clusters of Images

8 Concluding Remarks

The creation of a multimedia retrieval system is a difficult process that requires considerable efforts. As we pointed out all along this chapter, these efforts are often underestimated and this is particularly true for some crucial steps. It is also important to note that a well thought out design of the retrieval system, for instance following the methodology proposed here, is the key of a successful system. We recommend in particular considering all the possible interactions presented in this chapter. Because of these hidden difficulties, very often a multimedia retrieval system focuses on just one media like image or audio, eventually combined with text. But it is clear that in the future the use of several media types in one single retrieval system will show its synergies, and the joined use of several media types is anyway required for the design of improved video retrieval systems.

Although the construction of a multimedia retrieval system is a difficult task, it represents a fascinating challenge. The results are always gratifying, because the new designed tools help to search through data that is richer and its meaning subtler than that of pure text.

References

1. Bonastre, J.F., Delacourt, P., Fredouille, C., Merlin, T., Wellekens, C.J.: A speaker tracking system based on speaker turn detection for nist evaluation. In: Proc. of ICASSP 2000, Istanbul (2000)

2. Santo, M.D., Percannella, G., Sansone, C., Vento, M.: Classifying audio streams of movies by a multi-expert system. In: Proc. of Int. Conf. on Image Analysis and Processing (ICIAP01), Palermo, Italy (2001)
3. Montacié, C., Caraty, M.J.: A silence/noise/music/speech splitting algorithm. In: Proc. of ICSLP, Sydney, Australia (1998)
4. Idris, F., Panchanathan, S.: Review of image and video indexing techniques. *Journal of Visual Communication and Image Representation* **8** (1997) 146–166
5. Zhang, H.J., Low, C.Y., Smoliar, S.W., Wu, J.H.: Video parsing, retrieval and browsing: an integrated and content-based solution. In: Proc. of ACM Multimedia 95 - electronic proc., San Francisco, CA (1995)
6. Yu, H.H., Wolf, W.: A hierarchical multiresolution video shot transition detection scheme. *Computer Vision and Image Understanding* **75** (1999) 196–213
7. Sudhir, G., Lee, J.C.M.: Video annotation by motion interpretation using optical flow streams. *Journal of Visual Communication and Image Representation* **7** (1996) 354–368
8. Pilu, M.: On using raw mepg motion vectors to determine global camera motion. Technical report, Digital Media Dept. of HP Lab., Bristol (1997)
9. Lee, S.Y., Kao, H.M.: Video indexing - an approach based on moving object and track. *SPIE* **1908** (1993) 81–92
10. Sahouria, E.: Video indexing based on object motion. Master's thesis, UC Berkeley, CA (1997)
11. Ronfard, R., Thuong, T.T.: A framework for aligning and indexing movies with their script. In: Proc. of IEEE International Conference on Multimedia & Expo (ICME 2003), IEEE (2003)
12. Potamianos, G., Neti, C., Luettin, J., Matthews, I.: Audio-visual automatic speech recognition: An overview. In Bailly, G., Vatikiotis-Bateson, E., Perrier, P., eds.: *Issues in Visual and Audio-Visual Speech Processing*. MIT Press (2004)
13. Jain, A.K., Yu, B.: Automatic text location in images and video frames. *Pattern Recognition* **31** (1998) 2055–2076
14. Wu, V., Manmatha, R., Riseman, E.M.: Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21** (1999) 1224–1229
15. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. *Communications of the ACM* **18** (1975) 613–620 (see also TR74-218, Cornell University, NY, USA).
16. Robertson, S.E.: The probability ranking principle. *Journal of Documentation* **33** (1977) 294–304
17. van Rijsbergen, C.J.: A non-classical logic for information retrieval. *The Computer Journal* **29** (1986) 481–485
18. Turtle, H., Croft, W.B.: Inference networks for document retrieval. In: Proc. of the 13th Int. Conf. on Research and Development in Information Retrieval, New York, ACM (1990) 1–24
19. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K.: Indexing by latent semantic analysis. *Journal of the American Society for Information Sciences* **41** (1990) 391–407
20. Kaski, S.: Dimensionality reduction by random mapping: Fast similarity computation for clustering. In: Proc. of the International Joint Conference on Artificial Neural Networks (IJCNN'98). Volume 1., IEEE (1998) 413–418

21. Isbell, C.L., Viola, P.: Restructuring sparse high dimensional data for effective retrieval. In: Proc. of the Conference on Neural Information Processing (NIPS'98). (1998) 480–486
22. Salton, G., Allan, J., Buckley, C.: Automatic structuring and retrieval of large text files. *Communications of the ACM* **37** (1994) 97–108
23. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. *Information Processing & Management* **24** (1988) 513–523
24. Baeza-Yates, R., Ribeiro-Neto, B.: *Modern Information Retrieval*. Addison Wesley Longman (1999)
25. Greiff, W.R.: A theory of term weighting based on exploratory data analysis. In: 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, ACM (1998)
26. Frakes, W.B., Baeza-Yates, R.: *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, New Jersey (1992)
27. Porter, M.: An algorithm for suffix stripping. *Program* (1980) 130–137
28. Witten, I.H., Moffat, A., Bell, T.C.: *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, San Francisco (1999)
29. Klose, A., Nürnberger, A., Kruse, R., Hartmann, G.K., Richards, M.: Interactive text retrieval based on document similarities. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy* **25** (2000) 649–654
30. Lochbaum, K.E., Streeter, L.A.: Combining and comparing the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. *Information Processing and Management* **25** (1989) 665–676
31. Detyniecki, M.: Browsing a video with simple constrained queries over fuzzy annotations. In: *Flexible Query Answering Systems FQAS'2000*, Warsaw (2000) 282–287
32. Yager, R.R.: A hierarchical document retrieval language. *Information Retrieval* **3** (2000) 357–377
33. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proc. of the 7th International World Wide Web Conference, Brisbane, Australia (1998) 107–117
34. Manning, C.D., Schütze, H.: *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA (2001)
35. Steinbach, M., Karypis, G., Kumara, V.: A comparison of document clustering techniques. In: *KDD Workshop on Text Mining*. (2000) (see also TR 00-034, University of Minnesota, MN).
36. Nürnberger, A.: Clustering of document collections using a growing self-organizing map. In: Proc. of BISC International Workshop on Fuzzy Logic and the Internet (FLINT 2001), Berkeley, USA, ERL, College of Engineering, University of California (2001) 136–141
37. Roussinov, D.G., Chen, H.: Information navigation on the web by clustering and summarizing query results. *Information Processing & Management* **37** (2001) 789–816
38. Mendes, M.E., Sacks, L.: Dynamic knowledge representation for e-learning applications. In: Proc. of BISC International Workshop on Fuzzy Logic and the Internet (FLINT 2001), Berkeley, USA, ERL, College of Engineering, University of California (2001) 176–181
39. Weigend, A.S., Wiener, E.D., Pedersen, J.O.: Exploiting hierarchy in text categorization. *Information Retrieval* **1** (1999) 193–216

40. Ruiz, M.E., Srinivasan, P.: Hierarchical text categorization using neural networks. *Information Retrieval* **5** (2002) 87–118
41. Wermter, S.: Neural network agents for learning semantic text classification. *Information Retrieval* **3** (2000) 87–103
42. Teuteberg, F.: Agentenbasierte informationserschließung im www unter ein-satz von künstlichen neuronalen netzen und fuzzy-logik. *Künstliche Intelligenz* **03/02** (2002) 69–70
43. Benkhalifa, M., Mouradi, A., Bouyakhf, H.: Integrating external knowledge to supplement training data in semi-supervised learning for text categorization. *Information Retrieval* **4** (2001) 91–113
44. Vegas, J., de la Fuente, P., Crestani, F.: A graphical user interface for structured document retrieval. In Crestani, F., Girolami, M., van Rijsbergen, C.J., eds.: *Advances in Information Retrieval, Proc. of 24th BCS-IRSG European Colloquium on IR Research*, Berlin, Springer (2002) 268–283
45. Kohonen, T.: *Self-Organization and Associative Memory*. Springer-Verlag, Berlin (1984)
46. Nürnberger, A.: Interactive text retrieval supported by growing self-organizing maps. In Ojala, T., ed.: *Proc. of the International Workshop on Information Retrieval (IR 2001)*, Oulu, Finland, Infotech (2001) 61–70
47. Fritzke, B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks* **7** (1994) 1441–1460
48. Alahakoon, D., Halgamuge, S.K., Srinivasan, B.: Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks* **11** (2000) 601–614
49. Nürnberger, A., Detyniecki, M.: Visualizing changes in data collections using growing self-organizing maps. In: *Proc. of International Joint Conference on Neural Networks (IJCNN 2002)*, Piscataway, IEEE (2002) 1912–1917
50. Hearst, M.A., Karadi, C.: Cat-a-cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchie. In: *Proc. of the 20th Annual International ACM SIGIR Conference*, ACM (1997) 246–255
51. Spoerri, A.: *InfoCrystal: A Visual Tool for Information Retrieval*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA (1995)
52. Hemmje, M., Kunkel, C., Willett, A.: Lyberworld - a visualization user interface supporting fulltext retrieval. In: *Proc. of ACM SIGIR 94*, ACM (1994) 254–259
53. Fox, K.L., Frieder, O., Knepper, M.M., Snowberg, E.J.: Sentinel: A multiple engine information retrieval and visualization system. *Journal of the American Society of Information Science* **50** (1999) 616–625
54. Pu, P., Pecenovic, Z.: Dynamic overview technique for image retrieval. In: *Proc. of Data Visualization 2000*, Wien, Springer (2000) 43–52
55. Havre, S., Hetzler, E., Perrine, K., Jurrus, E., Miller, N.: Interactive visualization of multiple query result. In: *Proc. of IEEE Symposium on Information Visualization 2001*, IEEE (2001) 105–112
56. Lin, X., Marchionini, G., Soergel, D.: A selforganizing semantic map for information retrieval. In: *Proc. of the 14th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, New York, ACM Press (1991) 262–269
57. Honkela, T., Kaski, S., Lagus, K., Kohonen, T.: Newsgroup exploration with the websom method and browsing interface. Technical report, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland (1996)

58. Honkela, T.: Self-Organizing Maps in Natural Language Processing. PhD thesis, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland (1997)
59. Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paattero, V., Saarela, A.: Self organization of a massive document collection. *IEEE Transactions on Neural Networks* **11** (2000) 574–585
60. Merkl, D.: Text classification with self-organizing maps: Some lessons learned. *Neurocomputing* **21** (1998) 61–77
61. Boyack, K.W., Wylie, B.N., Davidson, G.S.: Domain visualization using vxinsight for science and technology management. *Journal of the American Society for Information Science and Technologie* **53** (2002) 764–774
62. Wise, J.A., Thomas, J.J., Pennock, K., Lantrip, D., Pottier, M., Schur, A., Crow, V.: Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In: *Proc. of IEEE Symposium on Information Visualization '95*, IEEE Computer Society Press (1995) 51–58
63. Small, H.: Visualizing science by citation mapping. *Journal of the American Society for Information Science* **50** (1999) 799–813
64. Nielsen, J.: *Usability Engineering*. Morgan Kaufmann Publishers (1994)
65. Shneiderman, B., Byrd, D., Croft, W.B.: Sorting out searching: A user-interface framework for text searches. *Communications of the ACM* **41** (1998) 95–98
66. Klusch, M., ed.: *Intelligent Information Agents*. Springer Verlag, Berlin (1999)
67. Rochio, J.J.: Relevance feedback in information retrieval. In Salton, G., ed.: *The SMART Retrieval System*. Prentice Hall, Englewood Cliffs, NJ (1971) 313–323
68. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. *Journal of the American Society for Information Science* **27** (1976) 129–146
69. Wong, S.K.M., Butz, C.J.: A bayesian approach to user profiling in information. *Technology Letters* **4** (2000) 50–56
70. Somlo, G.S., Howe, A.E.: Incremental clustering for profile maintenance in information gathering web agents. In: *Proc. of the 5th International Conference on Autonomous Agents (AGENTS '01)*, ACM Press (2001) 262–269
71. Joachims, T., Freitag, D., Mitchell, T.M.: Webwatcher: A tour guide for the world wide web. In: *Proc. of the International Joint Conferences on Artificial Intelligence (IJCAI 97)*, San Francisco, USA, Morgan Kaufmann Publishers (1997) 770–777
72. Jameson, A.: Modeling both the context and the user. *Personal and Ubiquitous Computing* **5** (2001) 29–33
73. Nürnberger, A., Klose, A., Kruse, R.: Self-organising maps for interactive search in document databases. In Szczepaniak, P.S., Segovia, J., Kacprzyk, J., Zadeh, L.A., eds.: *Intelligent Exploration of the Web*. Physica-Verlag, Heidelberg (2002) 119–135
74. Nürnberger, A., Klose, A.: Improving clustering and visualization of multimedia data using interactive user feedback. In: *Proc. of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*. (2002) 993–999
75. Nürnberger, A., Detyniecki, M.: User adaptive methods for interactive analysis of document databases. In: *Proc. of the European Symposium on Intelligent Technologies (EUNITE 2002)*, Aachen, Verlag Mainz (2002)