appeared in: R.J.Howlett, N.S.Ichalkaranje, L.C.Jain,G.Tonfoni (Eds): Internet-Based Intelligent Information Processing Systems, World Scientific, Singapore, pp.25-65, 2003

Adaptive Content Mapping for Internet Navigation

Rüdiger W. Brause, Markus Ueberall

J.W.G.-University, Frankfurt a.M., Germany,
{Brause,Markus}@Informatik.Uni-Frankfurt.de

1 Introduction

The Internet as the biggest human library ever assembled keeps on growing. Although all kinds of information carriers (e.g. audio/video/hybrid file formats) are available, text based documents dominate. It is estimated that about 80% of all information worldwide stored electronically exists in (or can be converted into) text form. More and more, all kinds of documents are generated by means of a text processing system and are therefore available electronically. Nowadays, many printed journals are also published online and may even discontinue to appear in print form tomorrow.
This development has many convincing advantages: the documents are both available faster (cf. prepress services) and cheaper, they can be searched more easily, the physical storage only needs a fraction of the space previously necessary and the medium will not age.
For most people, fast and easy access is the most interesting feature of the new age; computer-aided search for specific documents or Web pages becomes the basic tool for information-oriented work. But this tool has problems. The current keyword based search machines available on the Internet are not really appropriate for such a task; either there are (way) too many documents matching the specified keywords are presented or none at all. The problem lies in the fact that it is often very difficult to choose appropriate terms describing the desired topic in the first place.

This contribution discusses the current state-of-the-art techniques in content-based searching (along with common visualization/browsing approaches) and proposes a particular adaptive solution for intuitive Internet document navigation, which not only enables the user to provide full texts instead of manually selected keywords (if available), but also allows him/her to explore the whole database.

2 Standard information retrieval methods

The content based search within text documents has been established under the term *text retrieval*, which historically represents the first and most important branch within the *information retrieval* discipline, and is still subject to intensive research. Although we explicitly focus on *text retrieval* here, please note that al-

most all underlying concepts reviewed in this chapter can be applied to other information retrieval branches as well: just substitute “terms” and “words” by “features”.

In practice, the most obvious approach to characterize text documents by syntactic and semantic analysis quickly turns out to be intractable at least now. Therefore, almost all of the information retrieval mechanisms are based on condensed representations of the original documents like terms (i.e. keywords or catchwords) or meta information, if available.

2.1 Keyword search

The most simple approach to a content based search consists in scanning for one or several keywords. Although this is a straight and simple approach, a full text search takes too long for large databases. Therefore, all traditional Internet search engines like *Alta Vista* or *Fireball* parse the visited documents and only search within distilled term lists, which can also be accessed *much* faster [23].
This plain keyword search has the disadvantage of hit lists often being either too short or too long, because the user chose either wrong or inadequate keywords or very common terms. Limiting the result set to a reasonable size becomes an art *per se*.

2.2 Recall enhancers

The first improvement over plain keyword search (known as “aliasing”) consists in enlarging the user-provided list of keywords by similar words which can be restricted to cases where the original set of words resulted in too few hits. Common related techniques include:

- *word stemming*
  Here, all suffixes of words (e.g. in English [45] or German [14]) are discarded. Errors occur, if the same word stem is obtained for words of different semantics (*overstemming*) or if different stems are obtained for the same semantics (*understemming*). Note that all stemmers (unless they are dictionary-based) are language-dependent: an inappropriate use, e.g. in mixed language documents, leads to a drastic reduction in stemming quality.

- *dictionary-based identification of synonyms*
  A thesaurus is very useful if you want to cope with the problem of existing words having the same meaning (*synonymy*) or the same word having different meanings (*polysemy*), e.g. the word *bank* which has about a dozen different meanings in English.

- *synonym sets*
  Another idea consists in the construction of a set of all nouns, verbs, adjectives and adverbs, associated with a single semantic meaning (which can be

seen as some sort of super-thesaurus). A freely available realization of this concept is the WordNet lexical reference system [23]. It requires a much higher degree of manual input than a thesaurus and therefore cannot be generated semi-automatically as it is possible with the latter.

Additionally, all keywords (either user-provided or derived by aliasing) can be weighted or modified by means of a concept known as *relevance feedback* [29][38], whereby the user iteratively rates some documents in the list of results as being relevant or non-relevant. The system then tries to modify the keyword list accordingly, e.g. by discarding those query terms which only occur within non-relevant documents.

2.3 Using Meta information

A lot of meta information is contained in texts which contain semantic markup information, e.g. HTML/XML based Web pages with title or paragraph headers in subsequent order. The HTML standard also defines a "`meta`" tag which can (and should) be used to provide special information, e.g. the author´s name, manually identified keywords, or even an entire abstract.

Of course, these entries are of use only when they obey to a common standard (e.g. the Dublin Core Metadata Standard [20]), and cannot easily be maintained for an exponentially increasing number of documents which nowadays are often — at least partially — automatically generated by querying Web-based databases. Here, automatically derived meta information is needed.

One of the traditionally used sources of meta information are citations which can easily be used to build an automated ranking system, see e.g. *CiteSeer* [28]. Another feature of XML/HTML based documents, which distinguishes them from those in plain text format, is the existence of (true bi-directional) hyperlinks which can be taken into account for this task, too. Examples are the PageRank measure introduced by Google [6] or the more flexible HITS concept [33].

As of today, an automatic ranking of documents *solely* based on user-provided meta data greatly suffers from the overall imprecision and sponginess of the latter.

2.4 Classification trees

One important alternative for searching an unstructured set of documents is the manual classification of the document and the arrangement of the classes in a tree-like manner. These *classification trees* are very common in traditional library work and provide some advantages. They facilitate the search for new, unknown sources just by browsing through an appropriate subcategory of the classification tree. Even if the unknown document does not contain the critical keywords or terms, it can easily be found provided that the right paths to topics of interest are identifiable by the user.

The disadvantages are also well known:

- Since the classification is fixed, it is difficult to introduce changes in the classification system for evolving subjects, e.g. technology. It might become necessary to reorder whole subtrees which requires a reclassification of all the documents in the subtree.
- Documents often cannot be assigned to one single category. The common remedy for this problem is the duplication of the document reference which produces a multi-class membership. This implies other problems, see [32].
- The exploring and browsing is impeded by the fixed subclass boundaries and cannot automatically be redirected across the branches of the classification tree to another relevant branch.
- If many documents are located below one single subcategory, there is no feasibility to discriminate between them; in this case, the corresponding node can only be referred as a whole.
- The classification has to be performed manually by humans which is not affordable for huge collections. This is the crucial problem of Internet based documents: For the exponentially growing Internet resources, manual processing is prohibitive. Although there are initiatives like the *Open Directory Project* [43] which involves thousands of librarians, an automatic classification is necessary. Today, most of the automatic classification efforts rely on the automatic extraction of document features. This is done by a process called *indexing*.

2.5 Indexing

Standard information retrieval approaches use keywords, stemmers and thesauri only as preprocessing filters. They try to represent a document solely by all distinct terms which might characterize it. For this purpose, you have to preprocess and condense a document by several steps [4]:

(1) choice of appropriate terms

From the document choose appropriate terms and include them in a term collection. The meaning of "appropriate" depends on the chosen concept. For instance, the most simple strategy is to drop frequently occurring, uninteresting words (*stop terms*) like "and", "to", "in" [49]. This commonly used technique has the disadvantage that combinations of these simple words (phrases), e.g. "to be or not to be", might be important, and could not be found if the isolated words were discarded as stop terms. There are frequently used words which also should not be dropped like "time", "war", "home" [25]. Certainly, this makes the selection more subjective, or, at least, domain-dependent.

A more evolved strategy [52][53] only selects those terms that have a high *discrimination value* within a given set of documents. Here, as objective function to be minimized, the average document similarity based on the chosen subset of terms is used. Unfortunately, the computation of all interactions (similarities) between all documents is computationally expensive. This can be reduced by replacing the average similarity between all documents by the average simi-

larity between the documents and the term prototype, the average weight of all document terms. For each step of selecting the most discriminative terms, the value of the objective function is computed before and after dropping (or including) a term. If the objective function is increased by dropping a term, its discrimination value is positive and the term should be retained. Otherwise, it can be dropped.

Alternatively, instead of trying to identify (good) index *words* in the first place, simple *substrings* of fixed size *N* (so called *N-grams* [18]) can be extracted from a given text. If hash tables are used to store the counters needed to calculate the relative frequency of these substrings within the document after parsing, this approach is very fast [16]. Aside from this, the concept of N-grams is clearly language-independent, but, on the other hand, also not very descriptive for humans.

(2) weighting of the terms

Long documents naturally contain the same terms more frequently than short ones. In order to get rid of this peculiarity, the term frequencies have to be normalized [61]. Also, long documents contain a higher number of distinct terms which might better match a given request. Therefore, the length of a document has to be taken into account when weighting the terms.

(3) choice of appropriate indexing data structures

There are two popular data structures for index management: *signature files* and *inverted index files*. For each document, a signature file is created which consists of hash-encoded bit patterns of text parts within the corresponding document. This drastically reduces the search time, because instead of the document itself, only the much shorter signature file is searched for the hash-encoded search terms [21].

Alternatively, we might invert our term lists, one for each document, by building global lists, one for each term of the "global vocabulary", i.e. all distinct terms within the collection. Each list (*posting list*) contains the pointers to documents that include the specific term. This method has a lot of advantages over the use of signature files (e.g. false matches cannot occur) and should be preferred, see [66].

How many different terms do we have occurring f_1 times? To evaluate this, let us order all terms according to their occurrence frequency and assign them an index. The index one is for the most frequent term, the index two to the next frequent one and so on. Then we will notice an interesting fact: the product of index r and frequency f is approximately constant: $r \times f = \text{const} = K$. This observation is known as "Zipf's law" [65]. The number of different terms with frequency f_1 is reflected by the number of indices which have the same number f_1 of terms, the difference $Dr = r_2 - r_1 = K(\frac{1}{f_1+1} - \frac{1}{f_1}) = K\frac{1}{f_1(f_1+1)}$. The constant K can be observed for the rank r_m with frequency $f=1$: $K = r_m \times f = r_m$.

In conclusion, Zipf's law says that the number of different terms increases non-linearly with decreasing term occurrence and importance. Therefore, an important fraction of terms can be dropped if we introduce an occurrence threshold for the list of terms.

2.6 Vector space models

One of the classical methods of encoding the information space of a given set of documents is the approach of applying the well-known mathematical tool of linear algebra. Regarding the entries d_{ij} as the components of a document vector relative to "term vectors", the vector space model (VSM) [51] describes the documents as a linear combination of orthogonal base vectors, representing the basic terms.
Given a static vocabulary consisting of n distinct terms, each document can be represented as a vector of length n. Therefore, the documents as rows of terms form a document-term matrix **D** with the terms as columns. Each entry d_{ij} in the matrix represents the number of occurrences of term j within document i.

The "formally unclean" assumption of orthogonal base vectors was remedied by the later-proposed generalized vector space model (GVSM) [60]. Here, the orthogonality of boolean minimal conjunctive expressions (the dual space) is exploited to generate orthogonal base vectors.

Later on, the GVSM approach was modified to only represent the most relevant linear combinations of document features by Latent Semantic Indexing (LSI) [19] and to drop "unimportant" correlations. The term correlations between documents are treated by their statistical properties: the document-term matrix **D** is analyzed by a singular value decomposition in order to reduce the number of descriptive dimensions and to get the principal directions as intrinsic latent semantic structures.

2.7 Similarity measures

Within the mentioned vector space models, a query can be regarded as the problem of finding the most similar document to a given *pseudo-document* (e.g. consisting of a user-provided list of keywords or a *real* document). The similarity measures employed here are often derived from standard linear algebra measures, for instance the scalar product or the cosine between the vector representations of the documents to be compared [50][66].

Here, for our purpose a less-known (but not less-compelling) measure, the *cover coefficient concept* (CCC) [9][10][11], shall be sketched. Defining the importance of the j-th term relative to all terms of document $\mathbf{d}_i$, the i-th row of **D**, by

$$s_{ij} = \frac{d_{ij}}{\sum_k d_{ik}} \tag{1}$$

and the importance of the j-th term in document $\mathbf{d}_i$ relative to all documents in the collection containing the term (j-th column of $\mathbf{D}$) by

$$\tilde{s}_{ij} = \frac{d_{ij}}{\sum_k d_{kj}} \tag{2}$$

we get the degree c_{ij} of document coverage (the cover coefficient matrix $\mathbf{C}$) from the cross-correlation between two documents

$$c_{ij} = \sum_k s_{ik} \tilde{s}_{jk} \tag{3}$$

One major problem of all similarity measures discussed in this section is the situation where new documents with unknown terms have to be inserted in and compared with an existing collection of documents. Here, for huge collections the length of the vectors (list of terms) usually become very long and both the comparison and the weighting process becomes intractable. One approach to deal with this problem consists in passing over from a global document description into a local one which is only valid within a certain context or cluster of documents discussed in the next section.

3 Adaptive content mapping

The most interesting alternative to the manual classification task is the automatic, content based classification which maps the documents into different classes. In general, the topic oriented associative relationship maps have been standardized by the international norm ISO 13250, see e.g. [54], but there is no standard adaptive approach. Although the actual adaptive methods still have problems, the rapid growing Internet content produced by non-librarians allows no other approach in the near future. Based on the methods introduced in the previous sections, we will briefly review current adaptive content mapping methods.

3.1 Automatic classification by clustering

The similarity measures defined so far can be used to group documents into clusters. These semantic clusters represent a natural classification. In contrast to the static classification performed according to fixed criteria in section 2.4, the adaptive classification reflects the statistical properties of the document collection and will change according to the specific document collection.

There are two kinds of clustering algorithms: the non-hierarchical ones which, given a neighborhood criterion and a distance metric, divide the document space into a set of clusters, and the hierarchical ones which find clusters composed of smaller clusters on several levels. An overview can be found in [47] and [59].

Here, we take a closer look at the non-hierarchical cluster algorithms using the cover coefficient concept. With the expected number n_c of clusters

$$n_c = \left\lceil \sum_i c_{ii} \right\rceil \tag{4}$$

and the "cluster seed power" measure, which basically tries to capture the extend with which terms are distributed within a set of documents [12] and can be used to derive the term discrimination value of individual terms as well as to identify documents which contain a high number of "good" terms. The algorithm can be summarized as follows:

1. N_c:=0; WHILE $N_c<n_c$ DO
   Choose (n_c–N_c) the next documents of maximum cluster seed power as new cluster seeds
   Let N_c be the number of equivalence classes within this (sub)set of documents (two documents i and j belong to the same class if they have nearly identical c_{ii}, c_{iij} c_{ji} and c_{ii})
   ENDWHILE
2. With the N_c cluster documents obtained, assign each document *i* of the collection not being a cluster seed to the cluster document *k* of maximal coverage c_{ik}.
3. Documents which were not assigned to any cluster during the last step form a cluster by themselves.

This cluster algorithm has several advantages:

- It is stable; small variations in the term-document representation only lead to small changes in clustering
- If there is no similarity between documents, they will not share the same cluster as opposed to standard algorithms
- Given *m* documents and *n* terms, $n>>m$, this algorithm will cluster the documents by a computation complexity of $O(m*n)$
- The input sequence of the documents does not influence the clustering results

3.2 Adaptive hierarchical classification

The non-hierarchical cluster methods produce a set of clusters without any structure. For huge sets, the navigation is greatly facilitated if the set can be structured in a hierarchical manner. An automatic hierarchical classification, adapted to a document collection, can be performed by two different approaches, either bottom up or top down:

- *Agglomerative approach*
  The agglomerative strategy tries to fuse small entities in order to get bigger ones on the next higher level. The clustering fuses the *m* documents by *m-1* operations into a tree structured cluster set. A common used algorithm for this is the *nearest neighbor* approach [47] [59].

- *Divisive approach*
  The division of each cluster into smaller clusters is based on the similarity measure between the documents.
  One of the algorithms for successively dividing clusters and grouping them in a tree is the Principal Direction Divisive Partitioning algorithm (PDDP) introduced by Boley [7]. Like the LSI algorithm of section 2.6, it uses the dominant eigenvectors of the appropriate cross-correlation matrix. It transforms the document descriptions of the most scattered cluster in the *eigenspace*, and, based on the principal eigenvector, then decides for each document of the cluster whether to shift it in either the left or the right leafs of a binary tree.
  The algorithm was developed in the context of the WebACE project [8], where an user agent automatically retrieved potentially relevant documents from the web, based on a single user profile (namely, bookmarks and visited pages).

One of the newer search engines using hierarchical clustering is the Vivísimo project [56]. It uses conventional search engines for keyword search and then clusters the results dynamically, notably without parsing the referenced documents in its entirety by itself.

3.3 Local adaptation

Once the document collection has been transferred into a hierarchical classification, it becomes very expensive to add new documents. In the extreme, by statistical deviations, the whole adaptive classification tree becomes unstable and has to be reorganized. How do we handle such a situation?

In principle, this cannot be avoided if we want the classification to properly reflect the data-induced configuration. Nevertheless, we can try to make unstability less probable by several means:

- During initial adaptive clustering and classification, the documents with the "broadest" set of features should be chosen in order to build up a very general framework.
- For subsequent insertions of new documents, the structure should be kept stable as it is in the case of manual classification of section 2.4.
- The whole process of adaptive classification might be resumed if the number of new documents exceeds a predefined threshold.

This approach has one major drawback: the high computational costs of reorganizing the whole document collection, even if it occurs only periodically. As a compromise between stability and plasticity, only local adaptations can be made. This kind of continuous re-adaptation avoids the complexity of adapting the whole collection and supports the correct local document relations. Nevertheless, in the case of huge local changes in document statistics also changes in the global class hierarchy have to be considered.

4 Intuitive navigation

User interfaces for smart ("intelligent") systems have to face many demands. One of the most popular is described by the term "intuitive" which is not well defined. Raskin [46] references it as "familiar" which means that the guessing in bad user interfaces is replaced by knowledge. In this sense, we want to implement a user interface which is based on already existing knowledge.

4.1 Hierarchical navigation and the zoomable user interface

The search in huge databases is often facilitated by the approach of successively splitting the search space into smaller parts. This divide-and-conquer approach only needs logarithmic time, in contrast to an exhaustive scan of the entire database. It can be backed up and exploited by the user interface design. For browsing through a huge database, you might structure the data in a hierarchical manner and use the hierarchy in the user interface. Each hierarchy level might be presented visually, in a way appropriate to its content. On each hierarchy level, the user decides where to go next and selects the next level until he/she reaches the underlying document(s).

This idea of a level oriented top-down (and vice versa also bottom up) user interface can be extended to a continuous version: the zoomable interface [1][2][46]. This interface propagates the idea that the metaphor of flying, approaching a place by zooming in and leaving a place by zooming out, is sufficient to navigate within huge databases.

The zooming interface only has two modes: shifting and zooming. When you shift within an hierarchical level you only see abstract quantities mapped into a 2D plane. You move within these entities and select an interesting region. Then, you switch to zooming and approach the spot (a document) while the context (the other documents) becomes clearer, and you might even deviate to a more appropriate document.

The zoomable interface can also be used for other purposes than database navigation. Raskin [46] claims that it is even capable of replacing the traditional user interface completely, thereby rendering mouse devices and windows superfluous.

4.2 Similarity based visualization

There are already systems for intuitive navigation in documents by means of graphical user interfaces. One of the most straightforward implementations of content based navigation consists of placing all documents as symbols (small rectangles or circles) on a 2-D plane. The location on the plane is chosen according to their similarity value based on index terms, see section 2.7. There are several approaches for determining the position of a document (or document cluster) within the plane.

The first approach is given by the vector space model: each document is described by an index term vector of length *n* which gives the absolute coordinates or, alternatively, the difference i.e. the relative position between the documents to set up the 2-D display. This approach also needs a mapping stage where the *n*-dim. document space is mapped on the 2-D display.

One of the classical algorithms for doing this mapping is the nonmetric algorithm for multidimensional scaling (MDS) [35][36], which is computational expensive. A fast heuristic can be found in [22]. Here, the objective function "stress" (this term really represents a family of functions, cf. [16]), a measure of difference between the original distance matrix and the 2-D distance matrix, is minimized. One of the most compelling definitions for stress within this family is the so-called "proportional stress" which punishes deviations at long distances proportionally more than those at small distances. This can be interpreted as proportional to the energy of a system of particles joined by springs whose equilibrium configuration corresponds to a local energy minimum. Therefore, a system of "force-directed placements" like this used for visualizing a graph is often called a *spring embedder* and very popular in graph visualization. As application example, the *Lighthouse* system display of 50 documents matching the query "Samuel Adams" is shown in Fig. 1. The 50 matching documents are visualized as pseudo 3-D balls, the best matching ones marked by thick circles. Since the original text references are included, the whole window quickly becomes overloaded. Since these algorithms do not consider absolute coordinates, the resulting picture has no preferred orientation; it can arbitrarily be rotated.

This interface suffers from the several drawbacks:

- Only documents are displayed which match a certain search criterion, all other documents are ignored
- The configuration of displayed documents change after each modification of the search criterion, making it impossible to remember a certain area of the document space.
- The number of documents in the display is limited to approximately 100

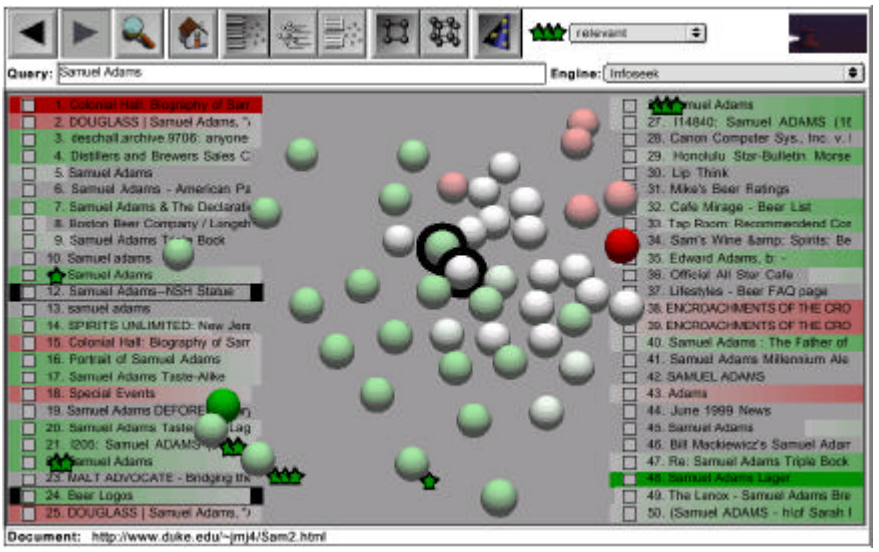

Fig. 1. A query display of the *Lighthouse* system [37]

Therefore, huge document collections can hardly be explored. As a remedy, hierarchical maps may be defined. One of the most famous examples is the WEBSOM approach [31] where an adaptive Kohonen map is used for mapping the document space onto a regular 2-D grid. The contents of the nodes in the fixed display has to be evaluated afterwards. In Fig. 2, a couple of windows representing several hierarchical levels are shown.

This absolute coordinate approach has several disadvantages:

- If you introduce new documents and / or new terms, the whole system has to be retrained which takes a long time in huge databases – often prohibitively long.
- Another disadvantage is that the layout will change afterwards. Since the cluster display changes, the user has to habituate to the new scene even when he/she already knows the majority of documents.

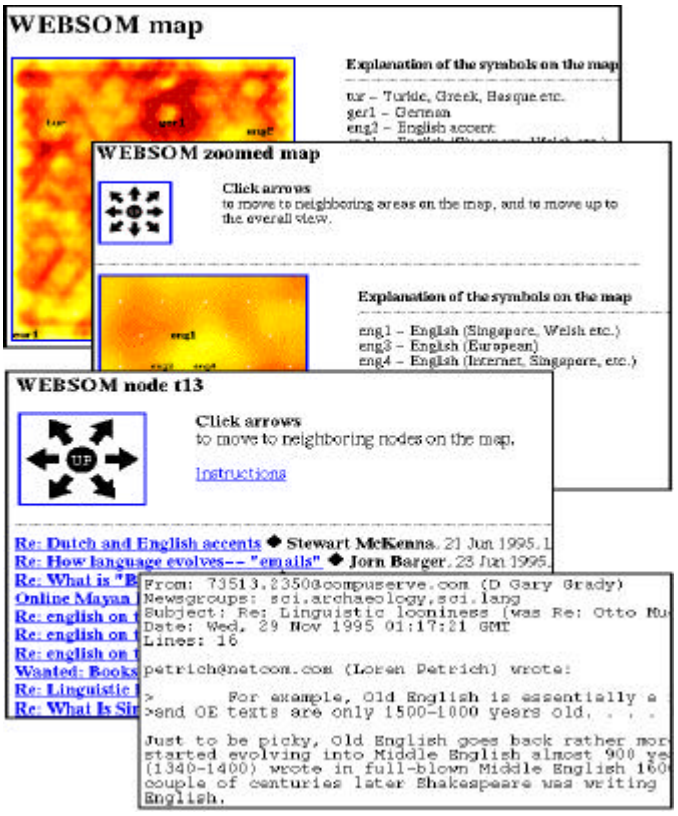

Fig. 2 The WEBSOM adaptive map and its hierarchical windows [31]

Here, too, some properties hinder an intuitive navigation:

- The high number of windows of several search process "levels" makes it difficult to maintain an overview of the search process
- The *document content distance* between the regular spaced clusters in the map display are expressed by different color shades. However, this makes a quick orientation rather difficult.

An interesting alternative visualization is demonstrated by the *WebMap* system [58], which allows for the (manual) assignment of icons to clusters and single documents.

5 The HADES System

In this section we will present a new adaptive system for intuitive navigation called HADES (Hierarchical Adaptive Document Exploration System). Its under-

lying concepts are based on the review results presented in the previous sections, integrating the most advanced and our new concepts into one concise design and adding often neglected but important features like portability and intelligent load balancing [55].

5.1 Specifications

The system has to meet the following criteria:

- *Adaptive classification*
  The classification structure must not be constant but always should reflect the characteristics of the growing document collection.
- *Intuitive navigation*
  The user interface should reflect the underlying hierarchical structure. It should be possible to explore the classification tree intuitively (i.e. without special training).
- *Modularity*
  The system has to be designed such that it contains functionally distinctive modules which enables local updates of functions or even the complete replacement of them by similar functional software.
- *Portability*
  The program code should not depend on a specific machine type or operating system but should be easily portable to new architectures.
- *Load balancing*
  For large document collections, the interaction speed and therefore the user acceptance of the system depends on the ability to automatically distribute the workload within a cluster of servers. This feature can hardly be implemented afterwards – it has to be taken into account at specification time.

5.2 The adaptation mechanisms

There are several mechanisms which are designed to reflect the specification of adaptive classification.

- *Adaptive clustering*
  When a new document is processed by the system, at first all terms are extracted by the parser. This reduced representation is then merged into a central data structure, consisting of a number of inverse index tables [4][42][66], see section 2.5. This enables the inclusion of new distinct terms of new documents. Then, the document is routed, starting with the root node of the classification tree, until the lowest hierarchy level (leaf) is reached and inserted in the last node visited. The similarity measure for routing is the so-

called cosine coefficient [51][42], combined with the cover coefficient concept [11].
If the node cluster size limit is reached, the cluster must be split into several parts. During this operation, the involved node has to be locked and the reorganization takes place.
The more interesting case, which involves the fusing of nodes, is computationally much more expensive: If the node contains references to other nodes, instead of fusing the whole collection and completely reorganizing it, we use the following heuristic: We do not lock all document representations but use *copies* of a *subset* of them when re-clustering. Afterwards, the resulting clusters are split. If the new clusters contain too many representatives of different clusters, a shifting and reorganization is not favorable. Instead, the same algorithm is recursively tried on lower levels until it either reaches a smaller diversity in a cluster or the lowest node of the tree. This kind of heuristic assists the demand for structure conservation and *recognition support* for the user [25].
Note that the original representatives (and subtrees) of the hierarchy are not touched until the re-clustering was successful in which case a quick node substitution takes place. Otherwise, large parts of the hierarchy would permanently be inaccessible to the users while updating the database.

The dynamic, adaptive hierarchy depends on the sequence ordering of the incoming documents. This might result in the paradox that the same document is assigned to two different leafs of the classification tree, depending on the time when it has been classified. Reclassifying the whole collection after classification changes may be prohibitive for large collections and will necessitate unwanted reorientation efforts of the user.
Here, a compromise between stability and plasticity has to be designed. The kind and degree of adaptation has to reflect the users' needs for stable, known classification regions. This is done by the introduction of a *cohesion* and an *adhesion* parameter which depend on the position and depth of the nodes within the hierarchy tree. An additional *affinity* parameter controls the local readapting in regular intervals depending on the workload. All three parameters are controlled by the user habits and adapt to the users' needs.

- *Recognition of structures*
  The goal of intuitive navigation in the context of adaptive clustering demands stable document cluster structures which can be recognized by the user. This avoids confusion at the user interface level and supports the feeling of familiarity with the system.
  Since we have different words which have the same meaning a thesaurus can help to cluster similar documents with different terms into a content based neighborhood. The small document specific thesaurus is automatically generated on the base of a general thesaurus and is treated like an abstract of the document.

- *Meta information*
  Meta information (references of all kind) is preserved during the indexing process and flows into the affinity values for document pairs and clusters. There are still some questions open: What should we do with documents which are referenced by other documents? Should the information be propagated to other levels and if so, how should it be considered there? Should we allow the user to jump back-and-forth between hierarchy nodes?

5.3 Intuitive Navigation

For the exploration of the document database we chose the content based, zoomable user interface as interaction paradigm. It consists of the following elements:

- *content based similarity mapping*
  The documents are represented on a 2D screen window by symbols: sheets for real documents and directory symbols for cluster representatives. The display of the symbols uses the computational feasible FastMap [21] algorithm. The distances between the symbols reflect the similarity in document content. As similarity measure we use the well-known *cosine coefficient* [11] in (indirect) combination with the *cover coefficient concept* cf. section 2.7).
  Additionally, in the extended view the document file information is also shown in a list, ordered by the search request similarity criterion. Fig. 3 shows a sample window.

- *zoomable content*
  The zoomable interface permits the display of details if you zoom into a document. In order to implement this, we chose not to present the document text directly in physically different resolutions to the user, but to successively show the document details in several stages: In the first stage only a main term, then a term list, then an abstract and afterwards the whole text are shown. The abstracts or relevant text fragments (*gists*) are generated automatically, see [15][39].

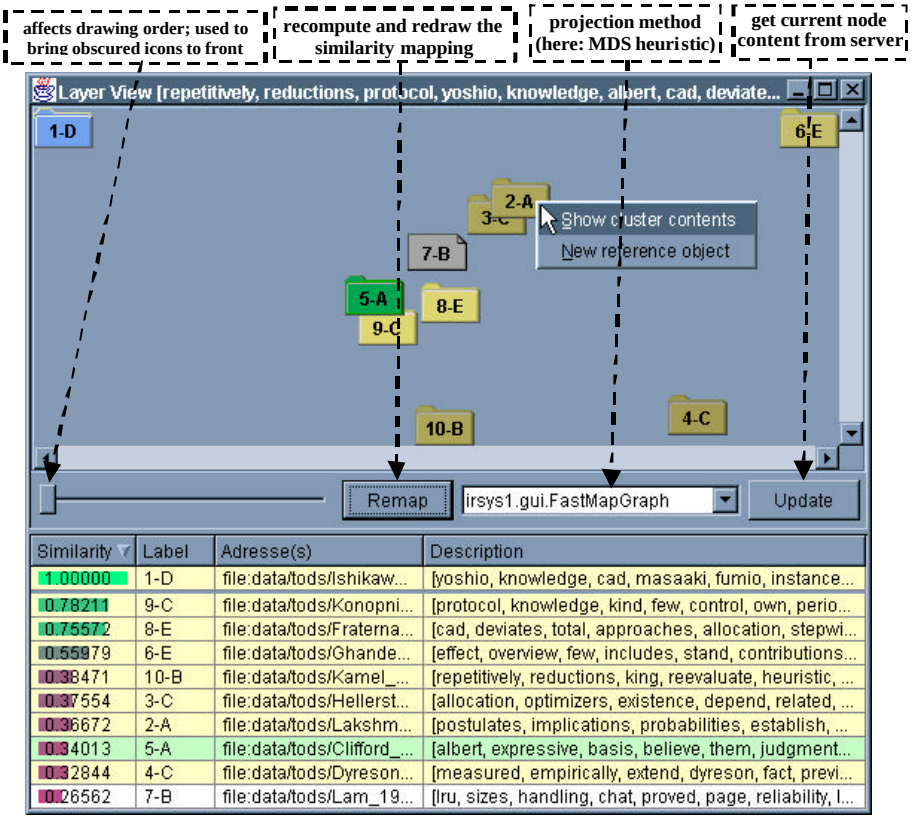

Fig. 3. Visualization of single documents (sheets) and clusters (directories)

- *zoomable hierarchy*
  For the representative documents, we have to distinguish between the document itself and its representation function. In the latter case we switch to the next level of hierarchy and its associated similarity mapping. In Fig. 4 this is shown for the example of Fig. 3 where the cluster representative "2-A" has been selected. Please note that the node description has adaptively changed due to the new context showing the new discrimination terms.

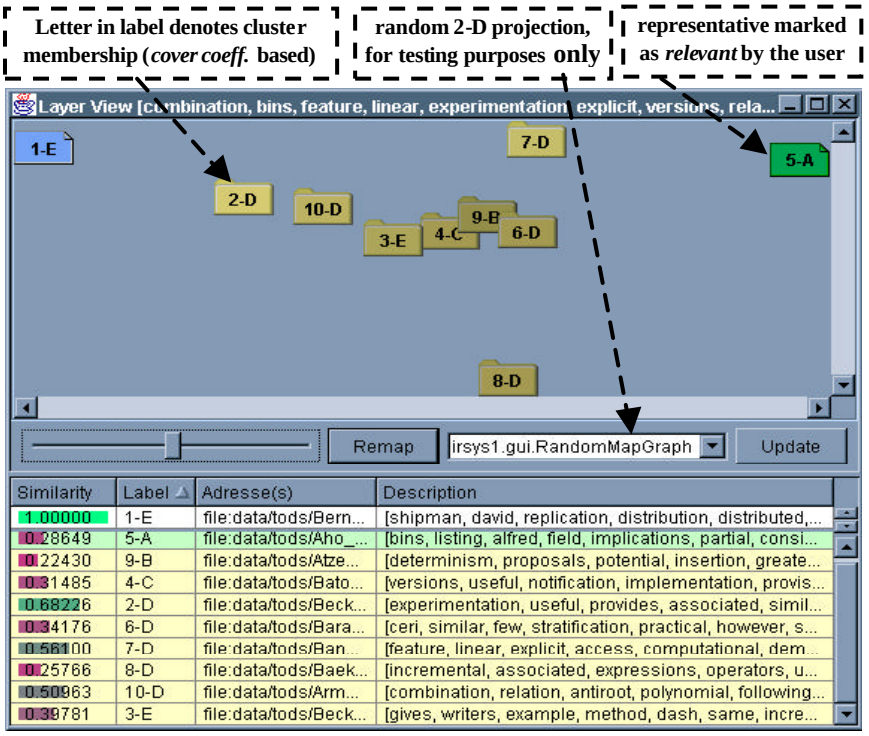

Fig. 4. The next hierarchy level

- *context display*
  For navigation, it is often very helpful to orient oneself along the context map in order to plan the next moves. Here, we chose the classification tree as context. A sample display is shown in Fig. 5.

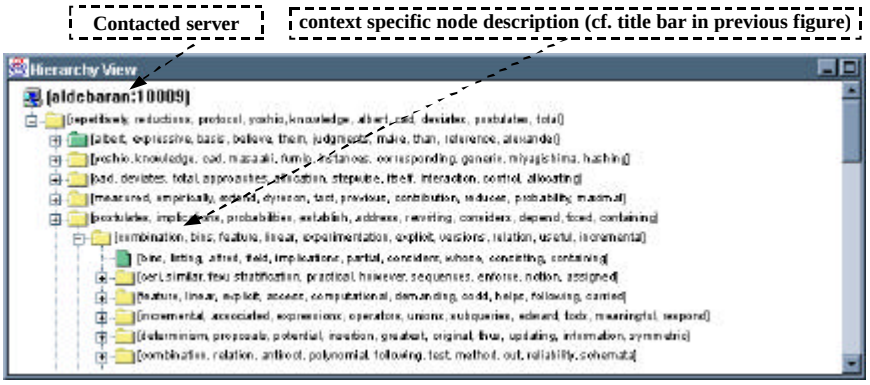

Fig. 5. The hierarchy display window for the example

- *search history display*
  An additional help is the display of the search history. It shows all documents marked as relevant in a compact form, see Fig. 6. Only three levels of hierarchy are displayed: The initial search document, all visited (visualized) nodes and all documents marked as "relevant" within these nodes.

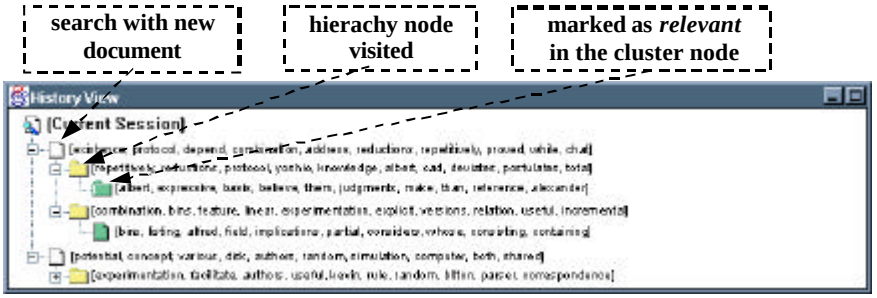

Fig. 6. The search history display window

Additional navigation possibilities evolve if hyperlinks can be exploited to jump back-and-forth between documents. It is not clear if this feature is helpful or confusing and has therefore to be evaluated.

5.4 Implementation issues

There are several practical implementation features of our system which should be mentioned here also.

5.4.1 Modularity

The code of the system is based on a client-server structure, each one divided into several, independent parts, see Fig. 7. Each part can be replaced by an equivalent functional entity implementing another algorithm. Therefore, changes in the database or user interface are easily tolerated.
The communication between the following listed modules is based entirely on message passing.

- `UserInterface`: This module implements the graphic user interface on the client side.
- `MainControl`: The main module initializes all services on the server side. The server may be part of a cluster.
- `ClusterControl`: This module is responsible for the generation and adaptive modification of the hierarchy of the local computer.

- `Parser`: The parser (on server side) scans documents, transforms them into the internal representation and generates the index terms. This might also be migrated to client side.
- `Gatherer`: This module is responsible for the setup of the internal data structures and preprocesses all document input (scanning for further references).
- `Repository`: this module is responsible for the persistent storage of the internal data structures (documents, meta data and so on.).

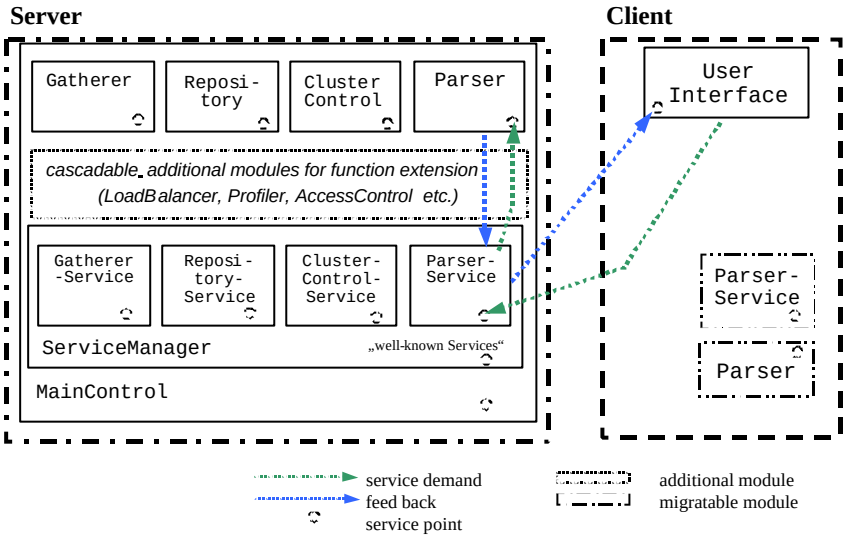

Fig. 7. Schematic overview of the system architecture

The advantages of such a modular concept are obvious:

- The *user interface* can be coupled with a diversity of search engines. It provides an uniform interface based on a 2-D similarity display if possible, or a (conventional) list representation otherwise.
- The replacement of the cluster component of the `ClusterControl`-module allows the use of statistical classifiers.
- The parser can be extended for other data formats independently of the rest of the system.
- The encapsulation of the communication within an abstract message passing-based subsystem favors a restructuring or redistribution of the software within the client-server model. Single JAVA classes can even be substituted at run time.

5.4.2 Portability and Integration ability

The implementation of the system in the programming language JAVA basically enables us to use our software on a diversity of computer systems. This makes load balancing possible even within a cluster of non-uniform machines.
Additionally, by implementing the client in the form of a signed applet, the user interface can rely on the functionality of standard browsers.

The client-server architecture not only does support load balancing, but also provides means for confidently (pre)processing the sample document provided by the user without the necessity to transfer its contents to the unsecure/untrusted server side.

Another important feature is the ability to integrate already existing search systems within our framework. Text based systems can easily use our user interface for text output. Also, existing word stemmers, clustering and indexing mechanisms can be used alternatively. This integration possibility facilitates the user acceptance and helps migrating from already existing older systems to the new one.

5.4.3 Load balancing

Aside from the already discussed possibilities of migrating some modules between client and server there are a couple of other load distribution possibilities for the modules:

- *parser*

  Beside the possibility of shifting heavy workload of user input processing (e.g. huge PostScript or PDF files) to the client also simple round robin schemes can easily be implemented in a cluster.

- *cluster control*

  The clustering of the index terms (fusion and division of clusters) is one of the most critical workloads. Especially the root node of the classification tree is a heavily frequented data structure which should be mirrored by other computers of the cluster. Low level nodes, e.g. leafs, are not frequently visited and can therefore be moved to the machines with low workload. Between these extremes, common strategies can be used to decide whether or not to move nodes.

- *repository*

  The most simple load balancing strategy consists of the use of several independent repositories for partial hierarchies. Shifting the representatives from one level to another or between nodes on the same hierarchy level might require additional data transfer efforts between the computers who store the affected nodes, i.e. directory and context information/links have to be adjusted. Similar to the parser, a *round robin* load balancing scheme can be implemented, but must be backed up by corresponding data replication.

6 Discussion and outlook

After the review of state-of-the-art information retrieval concepts and related algorithms we focused on the case of document search on the Internet. We introduced a system architecture and the components of a new adaptive, intuitive Internet document navigation system.

Although our system builds upon the experiences of existing systems, there are many open questions left for our special design. Especially those related to the user interface have to be evaluated in practice.

- *adaptive classification*
  Usage of standard search engines has already been evaluated [34], even for Web visualization [30], but truly searchable adaptive directory structures are new. Are users willing to accept structural changes? What are the optimal stability/plasticity parameters?
- *user interface*
  The zoomable interface paradigm is quite new and has not yet been evaluated within an adaptive setting. Does it help the user or does it rather hinder the information retrieval process? This "intuitive" approach might be misleading; perhaps zooming is the wrong metaphor for such a task.

In order to answer these questions, an evaluation stage is planned in cooperation with the German National Library.

Beside these basic questions, there is still much work left:

- *linguistic analysis*
  The "semantic" meaning of identified clusters might be greatly improved if commercial (language-dependent) dictionaries/thesauri can be used to support the classification of terms [39]. Aside from arising licensing problems (client-side preprocessing of documents would certainly be hindered), it is not clear how to match the dictionaries´ underlying *static* classifications with the dynamic ones which are generated by our system.
- *content management*
  A content management system deals with the task of management of data formats, data conversion, version control, protocols for web publishing and so on. Our system does not contain these features (yet) as our current focus really is on *content-oriented navigation*. Nevertheless, as stated above, thanks to the modular architecture, this kind of functionality could be added later.
- *coupling of independent systems*
  If there are two (ore more) independent systems, each one using its own document database, a combination of the hierarchy trees might result in better search results and exploration possibilities compared with contacting each one of them separately. On the other hand, each system administrator may want to

maintain his/her own database and may not be willing to fuse the document collections. What should we do? The answer is a time-limited coupling of the systems. The obvious approach would consist in the usage of some sort of *meta crawler*, but in order to enable the user to truly navigate within the resulting classification forest even across system boundaries, additional information needs to be exchanged on the server side.

The research topic is: Which (sub)modules within the different systems should communicate with each other? The required coordination has to take place at the same time ordinary search tasks are processed by the coupled systems – a difficult task.

In conclusion, adaptive internet navigation provides a lot of new and user friendly topics for content oriented document search. However, the adaptive plasticity in the data structures also implies new challenges for data consistency and user orientation within the information retrieval process. Our approach will provide new insights into balancing stability vs. plasticity of data structures and visualization.

References

For all URL the date of a valid access is given in brackets [..]

[1] Bederson, B., Hollan, J.D. (1994) Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics, *Proc. ACM UIST'94*, ACM Press.

[2] Bederson, B., Meyer, J. (1998). Implementing a Zooming User Interface: Experience Building Pad++, *Software: Practice and Experience* **28**(10): 1101-1135, ISSN 1097-024X.
URL: http://www.cs.umd.edu/hcil/pad++/papers/spe-98-padimplementation/spe-98-padimplementation.pdf [2001-08-18]

[3] Belkin, N. J. and Croft, W. B. (1992). Information Filtering and Information Retrieval: Two Sides of the Same Coin?, *Communications of the ACM* **35**(12): 29–38, ISSN 0001-0782.

[4] Blelloch, G. (1998).*Algorithms in the Real World*, Berkeley University. Class notes 1997/98.
URL: http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/294/class-notes/all/AlgsInRealWorld.ps.gz [2001-03-27]

[5] Boley, D. L. (1998). Hierarchical Taxonomies using Divisive Partitioning, *Technical Report TR-98-012*, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455.
URL:ftp://ftp.cs.unm.edu/dept/users/boley/reports/taxonomy.ps.gz [2001-03-27]

[6] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine, *Proceedings of the 7th International World Wide Web Conference (WWW7)*, Brisbane, Australia.
URL: http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm [2001-06-12]

[7] Boley, D. L. (1997). Principal Direction Divisive Partitioning, *Technical Report TR-97-056*, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455.

URL: ftp://ftp.cs.umn.edu/dept/users/boley/reports/PDDP.ps.gz [2001-03-27]

[8] Boley, D., Gini, M., Gross, R., Han, E.-H., Hastings, K., Karypis, G., Kumar,V., Mobasher, B. and Moore, H. (1999). Document Categorization and Query Generation on the World Wide Web using WebACE, *Artificial Intelligence Review* **13**(5-6): 365–391.
URL: http://www-users.cs.umn.edu/~gross/papers/aij.agent.ps [2001-03-27]

[9] Can, F. and Ozkarahan, E. A. (1983). A Clustering Scheme, *in* J. J. Kuehn [Eds.], *Proceedings of the 6th Annual Int. ACM/SIGIR Conf. on Research and Development in Information Retrieval*, Vol. 17, No. 4, ACM Press, Bethesda, Maryland, USA, S. 115–121, ISBN 0-89791-107-5.

[10] Can, F. und Ozkarahan, E. A. (1984). Two Partitioning Type Clustering Algorithms, *Journal of the American Society for Information Science* **35**(5): 268–276, John Wiley & Sons, Inc., ISSN 0002-8231.

[11] Can, F. and Ozkarahan, E. A. (1985).Similarity and Stability Analysis of the Two Partitioning Type Clustering Algorithms, *Journal of the American Society for Information Science* **36**(1): 3–14, John Wiley & Sons, Inc., ISSN 0002-8231.

[12] Can, F. and Ozkarahan, E. A. (1989). Dynamic Cluster Maintenance, *Information Processing & Management* **25**(3): 275–291, Pergamon Press Ltd., ISSN 0306-4573.

[13] CARMEN: Content Analysis, Retrieval and Metadata: Effective Networking (1999).
URL: http://www.mathematik.uni-osnabrueck.de/projects/carmen/ [2001-08-17]

[14] Caumanns, J. (1998).*A Fast and Simple Stemming Algorithm*, Freie Universität Berlin, CeDiS.
URL: http://www.wiwiss.fu-berlin.de/~caumanns/i4/ papers/se/ stemming.ps [2001-02-01]

[15] Cohen, J. D. (1995). Highlights: Language- and Domain-Independent Automatic Indexing Terms for Abstracting, *Journal of the American Society for Information Science* **46**(3): 162–174, John Wiley & Sons, Inc., ISSN 0002-8231. (Erratum in *JASIS* **47**(3): 260.)
URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=10050162&PLACEBO=IE.pdf [2001-06-15]
URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=57719&PLACEBO=IE.pdf [2001-06-15]

[16] Cohen, J. D. (1997). Drawing Graphs to Convey Proximity: An Incremental Arrangement Method, *ACM Transactions On Computer-Human Interaction*, Vol. 4, No. 3, ACM Press, pp. 197–229.
URL: http://www.acm.org/pubs/citations/journals/tochi/1997-4-3/p197-cohen/ [2001-06-08]

[17] Cohen, J. D. (1997). Recursive Hashing Functions for N-Grams, *ACM Transactions On Information Systems*, Vol. 15, No. 3, ACM Press, pp. 291–320.
URL: http://www.acm.org/pubs/citations/journals/tois/1997-15-3/p291-cohen/ [2001-06-08]

[18] Damashek, M. (1995). Gauging Similarity via N-Grams: Language-Independent Sorting, Categorization, and Retrieval of Text, *Science* **267**: 843–848, American Association for the Advancement of Science.
URL: http://gnowledge.sourceforge.net/damashek-ngrams.pdf [2001-05-11]

[19] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. (1990). Indexing by Latent Semantic Analysis, *Journal of the American Society for Information Science* **41**(6): 391–407, John Wiley & Sons, Inc., ISSN 0002-8231. URL:http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=10049585&PLACEBO=IE.pdf [2001-03-29]

[20] Dublin Core Metadata Initiative (1995). URL: http://www.dublincore.org [2001-08-21]

[21] Faloutsos, C. (1992). *Signature Files*, in [24], Chapter 4, pp. 44-65.

[22] Faloutsos, C. und Lin, K.-I. (1995). FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets, *Proceedings of the 1995 Int. ACM/SIGMOD Conf. on Management of Data*, Vol. 24, No. 2, ACM Press, pp. 163–174. URL: http://www.acm.org/pubs/citations/ proceedings/mod/223784/p163-faloutsos/ [2001-04-05

[23] Fellbaum, C. [Hrsg.] (1998). *WordNet: An electronic lexical database*, MIT Press, Cambridge, Massachusetts [u.a.], ISBN 0-262-06197-X. URL: http://www.cogsci.princeton.edu/~wn [2001-03-01]

[24] Frakes, W. B. and Baeza-Yates, R. S. [Eds.] (1992). *Information Retrieval: Data Structures and Algorithms*, first Ed., PTR Prentice-Hall, Inc., Eaglewood Cliffs, New Jersey 07632, ISBN 0-13-463837-9.

[25] Frakes, W. B. (1992b). *Stemming Algorithms*, in [24], Chapter 8, pp. 131-160.

[26] Jones, K. S. und Willett, P. [Hrsg.] (1997). *Readings in Information Retrieval*, The Morgan Kaufmann Series in Multimedia Information and Systems, erste Aufl., Morgan Kaufmann Publishers, San Francisco, CA 94104-3205, ISBN 1-55860-454-5.

[27] Koenemann, J. und Belkin, N. J. (1996). A case for interaction: a study of interactive information retrieval behaviour and effectiveness, in R. Bilger, S. Guest und M. J. Tauber [eds.], Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems, ACM Press, Vancouver, BC, Canada, pp. 205–212, ISBN 0-89791-777-4. URL: http://www.acm.org/pubs/citations/ proceedings/chi/238386/p205-koenemann/ [2001-03-29]

[28] Giles, C. L., Bollacker, K. D. and Lawrence, S. (1998). CiteSeer: An Automatic Citation Indexing System, *in* I. Witten, R. Akscyn and F. M. Shipman III [Eds.], *Third ACM Conference on Digital* Libraries, ACM Press, New York, pp. 89–98, ISBN 0-8979-1965-3. URL: http://www.neci.nj.nec.com/homepages/lawrence/papers/cs-dl98/cs-dl98-letter.pdf [2000-12-15]

[29] Harman, D. (1992). Relevance Feedback and Other Query Modification Techniques, in [24], Chapter 11, pp. 241–263.

[30] Heo, M. and Hirtle, S. (2001). An Empirical Comparison of Visualization Tools to Assist Information Retrieval on the Web, *Journal of the American Society for Information Science and Technology* **52**(8): 666–675, John Wiley & Sons, Inc., ISSN 1532-2882. URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=80002501&PLACEBO=IE.pdf [2001-06-08]

[31] Honkela, T., Kaski, S., Lagus, K. and Kohonen, T. (1996). Newsgroup Exploration with WEBSOM Method and Browsing Interface, *Report A32*, Helsinki Uni-

versity of Technology, Faculty of Information Technology, Laboratory of Computer and Information Science, Rakentajanaukio 2C, SF-02150 Espoo, Finland, ISBN 951-22-2949-8.
URL: http://websom.hut.fi/websom/doc/ps/honkela96tr.ps.gz [2001-04-05]

[32] Kaszkiel, M. and Zobel, J. (2001). Effective Ranking with Arbitrary Passages, *Journal of the American Society for Information Science and Technology* **52**(4): 344–364, John Wiley & Sons, Inc., ISSN 1532-2882.
URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=76508338&PLACEBO=IE.pdf [2001-04-30]

[33] Kleinberg, J. M. (1998). Authoritative Sources in a Hyperlinked Environment, *Proceedings of the 9th ACM/SIAM Symposium on Discrete Algorithms*, ACM Press, pp. 668–677. (Extended version appeared in *Journal of the ACM* **46**(5): 604-632.) URL:http://www.acm.org/pubs/articles/journals/jacm/1999-46-5/p604-kleinberg/p604-kleinberg.pdf [2001-06-16]

[34] Koenemann, J. and Belkin, N. J. (1996). A case for interaction: a study of interactive information retrieval behaviour and effectiveness, *in* R. Bilger, S. Guest and M. J. Tauber [Eds.], *Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, Vancouver, BC, Canada, pp. 205–212, ISBN 0-89791-777-4.
URL: http://www.acm.org/pubs/citations/proceedings/chi/238386/p205-koenemann/ [2001-03-29]

[35] Kruskal J.B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, vol. 29, pp. 1-27

[36] Kruskal J.B. (1964). Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, vol. 29, pp. 115-129

[37] Leuski, A. (2000). Details of Lighthouse, *Technical Report IR-212*, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts, Amherst, MA 01003. (Extended version of [38].)
URL: http://ciir.cs.umass.edu/pubfiles/ir-212.pdf [2001-03-26]

[38] Leuski, A. (2000). Relevance and Reinforcement in Interactive Browsing, *Proceedings of the Ninth* International *Conference on Information and Knowledge Management (CIKM)*, Washington, DC. (Also available as CIIR Technical Report IR-208, Department of Computer Science, University of Massachusetts, Amherst, MA 01003.)
URL: http://ciir.cs.umass.edu/pubfiles/ir-208.pdf [2001-03-26]

[39] Liddy, E. D. (1994). Text Categorization for Multiple Users Based on Semantic Features from a Machine-Readable Dictionary, ACM Transactions on Information Systems, 12(3):278-295, ACM Press.
URL: http://www.acm.org/pubs/citations/journals/tois/1994-12-3/p278-liddy/ [2001-03-26]

[40] Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts, *IBM Journal of Research and Development* **2**: 159–165, International Business Machines Corporation, ISSN 0018-8646.

[41] Meghabghab, G. (2001). Google's web page ranking applied to different topological web graph structures, *Journal of the American Society for Information Science and Technology* **52**, John Wiley & Sons, Inc., ISSN 1532-2882.
URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=82002488&PLACEBO=IE.pdf [2001-06-15]

[42] Moffat, A. und Zobel, J. (1996). Self-indexing Inverted Files for Fast Text Retrieval, ACM Transactions On Information Systems, Vol. 14, No. 4, ACM Press, pp. 349–379.
URL: http://www.acm.org/pubs/citations/journals/tois/1996-14-4/p349-moffat/ [2001-04-05]

[43] Open Directory Project. URL: http://dmoz.org [2001-09-01]

[44] Perlin, K., Fox, D. (1993). Pad - An Alternative Approach to the Computer Interface, *Proceedings of the ACM SIGGRAPH Conference*, Vol. 28, ACM Press, Anaheim, USA, pp. 57–64.
URL: http://www.cs.umd.edu/hcil/pad++/papers/siggraph-93-origpad/siggraph-93-origpad.ps.gz [2001-08-18]

[45] Porter, M. F. (1980). An algorithm for suffix stripping, *Program* **14**(3): 130–137, reprinted in [25].
URL: http://www.tartarus.org/~martin/PorterStemmer/ [2001-06-15]

[46] Raskin, J. (2000). *The* Humane *Interface: New Directions for Designing Interactive Systems*, first Ed., Addison Welsley Longman, Inc., Reading, Massachusetts 01867, ISBN 0-2-1-37937-6.

[47] Rasmussen, E. (1992). *Clustering Algorithms*, in [24], Chapter 16, pp. 419–442.

[48] van Rijsbergen, C. J. (1979). *Information Retrieval*, second Ed., Butterworths, London, ISBN 0-408-70929-4.
URL: http://www.dcs.gla.ac.uk/Keith/Preface.html [2001-07-12]

[49] Sahami, M. (1998). Using *Machine Learning to improve Information Access*, PhD thesis, Stanford University, Department of Computer Science.
URL: http://robotics.stanford.edu/users/sahami/papers-dir/thesis.ps [2001-07-09]

[50] Salton, G. and Buckley, C. (1988).Term Weighting Approaches in Automatic Text Retrieval, *Information Processing & Management* **24**(5): 513–523, Pergamon Press Ltd., ISSN 0306-4573. (Also available as Technical Report 87-881, Cornell University, Department of Computer Science, Ithaca, New York 14853.)
URL: http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR87-881/ [2001-04-05]

[51] Salton, G., Wong, A. and Yang, C. S. (1975). A Vector Space Model for Automatic Indexing, *Communications of the ACM* **18**(11): 613–620, ISSN 0001-0782. (Also available as Technical Report 74-218, Cornell University, Department of Computer Science, Ithaca, New York 14853.)
URL: http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR74-218/ [2001-04-05]

[52] Salton, G., Wong, A. and Yu, C. T. (1976). Automatic Indexing Using Term Discrimination and Term Precision Measurements, *Information Processing & Management* **12**: 43–51, Pergamon Press Ltd., ISSN 0306-4573.

[53] Salton, G., Wu, H. and Yu, C. T. (1981). The Measurement of Term Importance in Automatic Indexing, *Journal of the American Society for Information Science* **32**(3): 175–186, John Wiley & Sons, Inc., ISSN 0002-8231.

[54] Topic Maps: XML schema of ISO 13250
URL: http://www.diffuse.org/TopicMaps/schema.html [2001-09-29]

[55] Ueberall, M. (2001). Ein adaptives, hierarchisch organisiertes, verteiltes Navigationssystem zur inhaltsbasierten Dokumentensuche, Diplomarbeit am Institut für Informatik der Johann Wolfgang Goethe-Universität, Frankfurt/Main.

[56] Vivísimo. URL: http://www.vivisimo.com [2001-09-17]

[57] Wätjen, H.-J., Diekmann, B., Möller, G. and Carstensen, K.-U.(1998). Bericht zum DFG-Projekt GERHARD: German Harvest Automated Retrieval and Directory, *Technical Report*, Bibliotheks- und Informationssystem (BIS), Carl von Ossietzky Universität Oldenburg.
URL :http://www.gerhard.de/info/dokumente/dokumentation/gerhard/bericht.pdf [2001-07-23]

[58] WebMap URL: http://www.webmap.com [2001-09-17]

[59] Willett, P. (1988). Recent Trends in Hierarchic Document Clustering: A critical Review, *Information Processing & Management* **24**(5): 577–597, Pergamon Press Ltd., ISSN 0306-4573.

[60] Wong, S. K. M., Ziarko, W. and Wong, P. C. N. (1985). Generalized Vector Space Model in Information Retrieval, *Proceedings of the 8th Annual Int. ACM/SIGIR Conf. on Research and Development in Information Retrieval*, ACM Press, Montreal, Canada, pp. 18–25.
URL: http://www.acm.org/pubs/citations/proceedings/ir/253495/p18-wong/ [2001-04-13]

[61] Wulfekuhler, M. R. and Punch, W. F. (1997). Finding Salient Features for Personal Web Page Categories, *Proceedings of the Sixth International World Wide Web Conference*.
URL: http://www.cps.msu.edu/~wulfekuh/research/ PAPER118.ps [2001-07-23]

[62] Xu, J. and Croft, W. B. (1998). Corpus-Based Stemming using Co-occurence of Word Variants, *ACM Transactions On Information Systems*, Vol. 16, No. 1. (Also available as CIIR Technical Report IR-95, Department of Computer Science, University of Massachusetts, Amherst, MA 01003.)
URL: http://www.acm.org/pubs/citations/journals/tois/1998-16-1/p61-xu/ [2001-06-08]

[63] Zhang, J. and Korfhage, R. R. (1999). A Distance and Angle Similarity Measure Method, *Journal of the American Society for Information Science* **50**(9): 772–778, John Wiley & Sons, Inc., ISSN 0002-8231.
URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=10050162&PLACEBO=IE.pdf [2001-06-12]

[64] Zhang, J. and Wolfram, D. (2001). Visualization of term discrimination analysis, *Journal of the American Society for Information Science and Technology* **52**(8): 615–627, John Wiley &Sons, Inc., ISSN 1532-2882.
URL: http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=79502836&PLACEBO=IE.pdf [2001-06-08]

[65] Zipf, G. K. (1949). *Human behaviour and the principle of least effort – An introduction to human ecology*, Addison-Wesley, Cambridge, Massachusetts, ISBN 0-012-78978-1. (Facsimile: Hafner Publishing Company, New York, 1965.)

[66] Zobel, J. and Moffat, A. (1998).Exploring the Similarity Space, *ACM SIGIR Forum* **32**(1): 18–32, ACM Press.
URL: http://www.cs.mu.oz.au/~alistair/abstracts/zm98:forum.html [2001-06-08]