

SOMLib - New Approaches for Information Presentation and Handling

by Andreas Rauber

While tools exist that allow us to search through vast amounts of text within seconds, most systems fail to assist the user in getting an overview of the information available or maintaining orientation within an information space, and fail also to convey meta-information in an intuitively graspable way. SOMLib is a digital library system addressing these issues by providing automatic content-based organisation and metaphor-graphics-based visualisation facilitating exploration and understanding of information spaces.

With the increasing availability of information in electronic form, be it online magazines, legal or medical document databases, project archives, or documentation on a company-internal intranet, advanced digital library systems that support users in interacting with large information repositories are gaining in importance. Yet, while databases and search engines help us in retrieving snippets of information, current tools fail to provide us with a feeling of 'where' information is available, and how different facts relate to each other. The ability to keep an overview of factors such as the information available, the topics covered by a given site and the amount of information available on a given topic is only poorly supported. In addition to the powerful search methods offered by modern information systems, it seems difficult to provide equally powerful means of organising and structuring the information.

What we would like are ways of information organisation and representation that allow us to make use of the concepts that we are using constantly, unconsciously, when handling and navigating

real-world information spaces. Libraries, bookstores, project documentation in binders, working material and paper collections are all conventionally organised (also) by thematic criteria. This allows us to immediately get an overview of which kind of information is available in which section of an archive, how many reports have been filed on a specific topic in a binder, and so on. Due to the spatial location it is also easier to find a paper, report etc for the second time, as it is easier to recall roughly where a given document was located than to remember sufficiently precise search criteria, or its relative position within a listing.

With the SOMLib digital library, we created a system providing content-based organisation of document repositories, facilitating intuitive browsing and exploration of the information space. It builds on and incorporates works in the fields of information retrieval, neural networks, information visualisation, and usability analysis.

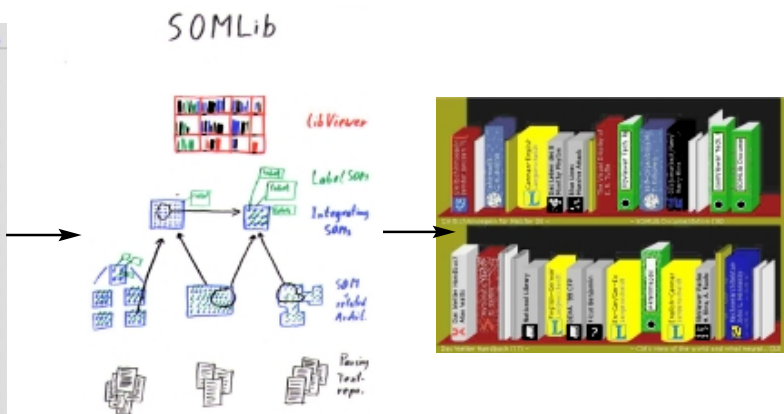
Low-level features based on word frequencies are extracted from the text to

provide a domain- and language-independent content representation of text documents in a high-dimensional vector space. The 'self-organising map' (SOM), a popular unsupervised neural network model, is further used to cluster the document feature vectors, performing a topology-preserving mapping of the documents from the high-dimensional vector space onto a two-dimensional map space. Documents are thus grouped according to their mutual similarity, having documents on similar topics mapped onto neighbouring locations on the map. Using the 'growing hierarchical self-organising map' (GHSOM), a novel extension of the SOM, we can further detect subject hierarchies in a document collection, with the neural network adapting its size and structure automatically during its unsupervised training process to reflect the topical hierarchy. Individual SOMs can further be integrated to form a network of referencing maps.

By mining the weight vector structure of the trained maps using the 'LabelSOM' technique, the system automatically extracts keywords describing the various

The SOMLib system: from text collections via content-based organisation, to metaphor-graphics-based representation of document repositories facilitating intuitive browsing and exploration.

NAME	Last modified	Size
Parent Directory	34-Oct-1999 04:55	-
lib_110a00.txt	34-Oct-1999 04:55	1K
lib_110a01.txt	34-Oct-1999 04:55	1K
lib_110a02.txt	34-Oct-1999 04:55	1K
lib_110a03.txt	34-Oct-1999 04:55	1K
lib_110a04.txt	34-Oct-1999 04:55	1K
lib_110a05.txt	34-Oct-1999 04:55	1K
lib_110a06.txt	34-Oct-1999 04:55	1K
lib_110a07.txt	34-Oct-1999 04:55	1K
lib_110a08.txt	34-Oct-1999 04:55	1K
lib_110a09.txt	34-Oct-1999 04:55	1K
lib_110a10.txt	34-Oct-1999 04:55	1K
lib_110a11.txt	34-Oct-1999 04:55	1K
lib_110a12.txt	34-Oct-1999 04:55	1K
lib_110a13.txt	34-Oct-1999 04:55	1K
lib_110a14.txt	34-Oct-1999 04:55	1K
lib_110a15.txt	34-Oct-1999 04:55	1K
lib_110a16.txt	34-Oct-1999 04:55	1K
lib_110a17.txt	34-Oct-1999 04:55	1K
lib_110a18.txt	34-Oct-1999 04:55	1K
lib_110a19.txt	34-Oct-1999 04:55	1K
lib_110a20.txt	34-Oct-1999 04:55	1K
lib_110a21.txt	34-Oct-1999 04:55	1K
lib_110a22.txt	34-Oct-1999 04:55	1K
lib_110a23.txt	34-Oct-1999 04:55	1K
lib_110a24.txt	34-Oct-1999 04:55	1K
lib_110a25.txt	34-Oct-1999 04:55	1K
lib_110a26.txt	34-Oct-1999 04:55	1K
lib_110a27.txt	34-Oct-1999 04:55	1K
lib_110a28.txt	34-Oct-1999 04:55	1K
lib_110a29.txt	34-Oct-1999 04:55	1K
lib_110a30.txt	34-Oct-1999 04:55	1K
lib_110a31.txt	34-Oct-1999 04:55	1K
lib_110a32.txt	34-Oct-1999 04:55	1K
lib_110a33.txt	34-Oct-1999 04:55	1K
lib_110a34.txt	34-Oct-1999 04:55	1K
lib_110a35.txt	34-Oct-1999 04:55	1K
lib_110a36.txt	34-Oct-1999 04:55	1K
lib_110a37.txt	34-Oct-1999 04:55	1K
lib_110a38.txt	34-Oct-1999 04:55	1K
lib_110a39.txt	34-Oct-1999 04:55	1K
lib_110a40.txt	34-Oct-1999 04:55	1K
lib_110a41.txt	34-Oct-1999 04:55	1K
lib_110a42.txt	34-Oct-1999 04:55	1K
lib_110a43.txt	34-Oct-1999 04:55	1K
lib_110a44.txt	34-Oct-1999 04:55	1K
lib_110a45.txt	34-Oct-1999 04:55	1K
lib_110a46.txt	34-Oct-1999 04:55	1K
lib_110a47.txt	34-Oct-1999 04:55	1K
lib_110a48.txt	34-Oct-1999 04:55	1K
lib_110a49.txt	34-Oct-1999 04:55	1K
lib_110a50.txt	34-Oct-1999 04:55	1K
lib_110a51.txt	34-Oct-1999 04:55	1K
lib_110a52.txt	34-Oct-1999 04:55	1K
lib_110a53.txt	34-Oct-1999 04:55	1K
lib_110a54.txt	34-Oct-1999 04:55	1K
lib_110a55.txt	34-Oct-1999 04:55	1K
lib_110a56.txt	34-Oct-1999 04:55	1K
lib_110a57.txt	34-Oct-1999 04:55	1K
lib_110a58.txt	34-Oct-1999 04:55	1K
lib_110a59.txt	34-Oct-1999 04:55	1K
lib_110a60.txt	34-Oct-1999 04:55	1K
lib_110a61.txt	34-Oct-1999 04:55	1K
lib_110a62.txt	34-Oct-1999 04:55	1K
lib_110a63.txt	34-Oct-1999 04:55	1K
lib_110a64.txt	34-Oct-1999 04:55	1K
lib_110a65.txt	34-Oct-1999 04:55	1K
lib_110a66.txt	34-Oct-1999 04:55	1K
lib_110a67.txt	34-Oct-1999 04:55	1K
lib_110a68.txt	34-Oct-1999 04:55	1K
lib_110a69.txt	34-Oct-1999 04:55	1K
lib_110a70.txt	34-Oct-1999 04:55	1K
lib_110a71.txt	34-Oct-1999 04:55	1K
lib_110a72.txt	34-Oct-1999 04:55	1K
lib_110a73.txt	34-Oct-1999 04:55	1K
lib_110a74.txt	34-Oct-1999 04:55	1K
lib_110a75.txt	34-Oct-1999 04:55	1K
lib_110a76.txt	34-Oct-1999 04:55	1K
lib_110a77.txt	34-Oct-1999 04:55	1K
lib_110a78.txt	34-Oct-1999 04:55	1K
lib_110a79.txt	34-Oct-1999 04:55	1K
lib_110a80.txt	34-Oct-1999 04:55	1K
lib_110a81.txt	34-Oct-1999 04:55	1K
lib_110a82.txt	34-Oct-1999 04:55	1K
lib_110a83.txt	34-Oct-1999 04:55	1K
lib_110a84.txt	34-Oct-1999 04:55	1K
lib_110a85.txt	34-Oct-1999 04:55	1K
lib_110a86.txt	34-Oct-1999 04:55	1K
lib_110a87.txt	34-Oct-1999 04:55	1K
lib_110a88.txt	34-Oct-1999 04:55	1K
lib_110a89.txt	34-Oct-1999 04:55	1K
lib_110a90.txt	34-Oct-1999 04:55	1K
lib_110a91.txt	34-Oct-1999 04:55	1K
lib_110a92.txt	34-Oct-1999 04:55	1K
lib_110a93.txt	34-Oct-1999 04:55	1K
lib_110a94.txt	34-Oct-1999 04:55	1K
lib_110a95.txt	34-Oct-1999 04:55	1K
lib_110a96.txt	34-Oct-1999 04:55	1K
lib_110a97.txt	34-Oct-1999 04:55	1K
lib_110a98.txt	34-Oct-1999 04:55	1K
lib_110a99.txt	34-Oct-1999 04:55	1K
lib_110a100.txt	34-Oct-1999 04:55	1K



topical clusters. This is based on the analysis of the feature distributions within each cluster. It helps users in identifying which topics are present in a given document collection and where they are located on the map.

Finally, the 'libViewer' provides an intuitive representation of the documents in a repository by using real-world metaphors such as different document types, spine widths, dust etc to convey metadata in an intuitively graspable way.

Using the 'SOMLib' system, users can browse a document collection in the form of bookshelves and find clusters of documents on similar topics located in neighbouring boxes, with the topic of

each box being described by a set of automatically extracted keywords, and metadata being depicted in the form of different document representations. In combination with conventional approaches for searching and dynamically sorting text archives we thus have a powerful tool at our disposal, which allows us to obtain and maintain an overview of the amount and type of information available, and to detect relationships between different documents. This means we can better handle, interact with, and use the available information.

The 'SOMLib' system has been applied in numerous different domains in a variety of languages, such as the organisation of legal databases, newspaper

archives, scientific document collections and Web search results. Recently, the principles of this work have been expanded for use in digital music archives as part of the 'SOM-enhanced Jukebox' (SOMeJB) system. By analysing frequency spectra of audio files and transforming them to time-invariant representations while incorporating psycho-acoustic models, organisation and exploration following musical genres is facilitated.

Andreas Rauber is currently an ERCIM Research Fellow at INRIA

Please contact:

Andreas Rauber
Vienna University of Technology
E-mail: rauber@ifs.tuwien.ac.at
<http://www.ifs.tuwien.ac.at/~andi>

Hybrid Caching of Search Engine Results

by Tiziano Fagni and Fabrizio Silvestri

The High Performance Computing Lab of ISTI-CNR in Pisa, has developed an efficient caching system aimed at exploiting the locality present in the queries submitted to a Web search engine. The cache adopts a novel hybrid strategy, according to which the results of the most frequently submitted queries are maintained in a static cache of fixed size, and only the queries that cannot be satisfied by the static cache compete for the use of a dynamic cache.

Caching is a very effective technique to make a service that distributes data/information to a multitude of clients scalable. As suggested by many researchers, caching can be used to improve the efficiency of a Web Search Engine (WSE). This is motivated by the high locality present in the stream of queries processed by a WSE, and by the relatively infrequent updates of WSE indexes that allow us to think of them as mostly read-only data.

The architecture of a modern WSE may be very complex and includes several machines, since a query may require various sub-tasks to be executed. For example, a large and scalable WSE, which is typically placed behind an http server, is usually composed of several searcher modules, each of which preferably runs on a distinct machine and is responsible for searching the index relative to one specific sub-collection of

documents. Of course, in front of these searcher machines there is a mediator/broker, which collects and reorders the results of the various Searchers, and produces a ranked vector of the most relevant document identifiers (DocIDs), eg a vector usually composed by 10 DocIDs. These DocIDs are then used to get the associated URLs and page snippets included in the html page returned to the user through the http server (see Figure 1).

On the basis of an accurate analysis of the locality present in three large query logs, we designed and implemented a novel hybrid caching strategy where the results of the most frequently accessed queries are maintained in a static cache of fixed size, which is completely rebuilt at fixed time intervals. Only the queries that cannot be satisfied by the static cache compete for the use of a dynamic cache.

The superiority of our hybrid strategy over purely static or dynamic caching policies was demonstrated by several experimental tests, conducted to measure the cache hit-rate as a function of the size of the cache, the percentage of static cache entries (f_{static}), and the replacement policy (LRU, LRU2, LRU2S, 2Q, FBR) used for managing dynamic cache entries. Figure 2 shows the cache hit rate obtained on the largest of our query logs by using different replacement policies as a function of the ratio between static and dynamic cache entries. As it can be seen, our hybrid caching strategy always outperforms purely static and dynamic policies.

Moreover, we showed that WSE query logs exhibit not only temporal locality, but also a limited spatial locality, due to the presence of requests for subsequent pages of results. While most user searches are satisfied by the first page of