An Integrated Approach to Semantic Evaluation and Content-Based Retrieval of Multimedia Documents

A. Knoll[2], C. Altenschmidt[3], J. Biskup[3], H.-M. Blüthgen[1], I. Glöckner[2], S. Hartrumpf[4], H. Helbig[4], C. Henning[1], R. Lüling[5], B. Monien[5], T. Noll[1], and N. Sensen[5]

[1] University of Technology RWTH Aachen
[2] University of Bielefeld
[3] University of Dortmund
[4] University of Hagen
[5] University of Paderborn

Abstract. We present an overview of a large combined querying and retrieval system that performs content-based on-line searches in a large database of multimedia documents (currently text, tables and colour images). Queries are submitted as sentences in natural language and are transformed into the language of the target database. The documents are analyzed semantically for their information content; in a data fusion step the individual pieces of information extracted from these documents are aggregated into cognitively adequate result documents.
There is no pre-indexing necessary when new documents are stored into the system. This retains a high degree of flexibility with respect to the questions that may be asked. It implies, however, that both huge amounts of data must be evaluated rapidly and that intelligent caching strategies must be employed. It is therefore mandatory that the system be equipped with dedicated high-speed hardware processors.
The complete system is currently available as a prototype; the paper outlines its architecture and gives examples of some real sample queries in the knowledge domain of weather data documents.

1 Introduction

Due to the increasingly comfortable electronic authoring and publishing tools available today, knowledge is more and more represented in documents containing text, pictures and audiovisual information. While the automatic search for keywords in textual databases (e. g. HTML documents) is straightforward, the automatic indexing of pictures with respect to their semantic content is a very difficult issue. Furthermore, simple keyword search does not meet the average user's need for precise retrieval. Thus, the detection of information and its human-readable preparation becomes a central point of interest for multimedia documents. Tools are needed that analyze the content of a large number of documents and make possible

- the submission of content-related questions,
- the fast detection of interesting information,
- the association, interpretation and cognitively adequate presentation of the detected information.

The *HPQS* (High Performance Query Server) presented in this paper is a coherent architecture for such a tool. It implements the whole process from accepting queries in natural language to analyzing, summarizing and presentation of information.

There are a number of systems offering parts of the functionality of the HPQS. QBIC [10] and VIR [29] realize content-based retrieval in picture databases (on the *image signal level*, not, however, on a *symbolic level*). QBIC extracts information about color, texture and the shape of parts of the picture. For a given picture, the VIR systems performs a search through an image database and selects similar pictures in terms of color distribution and other parameters. Some other systems [1] [2] demand special input such as sketches. An overview of early systems is presented in [23]. IRIS [22] and MediaMiner [7] offer a content-based picture-search that is based on the input of keywords. All pictures are indexed when they are stored into the database. They can only be accessed using the given keywords.

The HPQS system goes far beyond the scope outlined above:

- *Natural language* (NL) *queries* are interpreted semantically using a number of problem independent databases and databases that model the target application domain;
- Semantic information extracted from various source documents is aggregated using *fuzzy data fusion algorithms*;
- A *mediator* caches meta data information that is extracted from mass data and uses it for further requests, if possible;
- A *parallel media server* stores and delivers mass data information and performs search operations on these mass data items;
- *Dedicated high speed hardware* processors perform selected search operations on mass data items that demand large computational power.

To realize these basic characteristics the system consists of the five main modules shown in Figure 1:

Natural Language Interface: Questions are posed in natural language. From the query an **I**ntermediate **S**emantic **R**epresentation (ISR) is constructed.

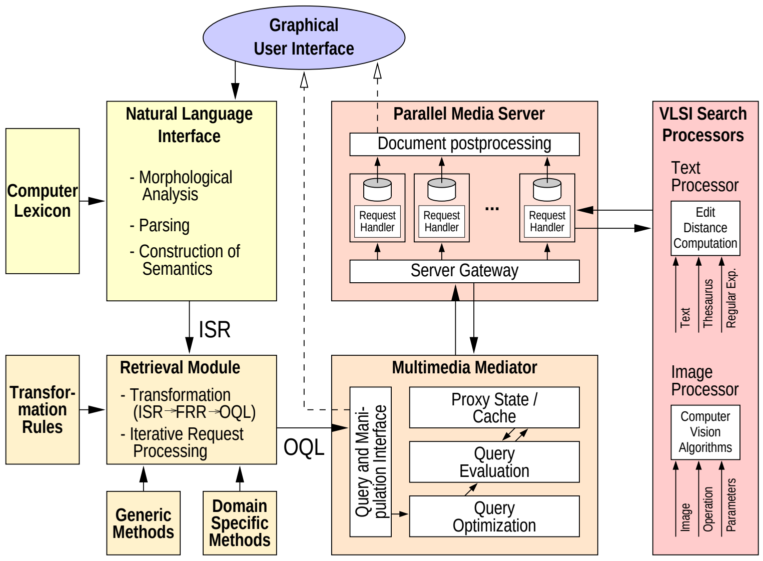


Fig. 1. Architecture of the High Performance Query Server

Retrieval Module: From the ISR representations an FRR (**F**ormal **R**etrieval **R**epresentation) is constructed. It makes use of transformation and interpretation knowledge. The FRR is transformed into ODMG-OQL requests. Information is fused using novel methods from fuzzy set theory.

Multimedia Mediator: The multimedia mediator structures and mediates the available mass data by storing, evaluating and creating additional *meta data*. OQL queries are processed with the help of the parallel server.

Parallel Server: The parallel server manages and delivers all available mass data. Additionally, time-consuming evaluation methods that work on the mass data information (e. g. image processing) can be initiated by the multimedia mediator and are performed by the parallel server.

Search Processors: Selected algorithms with high computational complexity for image processing and full-text retrieval are implemented as dedicated co-processors which are integrated into the parallel server using standard PCI interfaces.

An example knowledge area was selected to demonstrate the performance of the HPQS: the domain of weather data documents, i.e. satellite images, tables in numerous different formats and textual weather reports. This domain can in principle be replaced with every other domain. Examples of questions that can be handled are the following:

- *Which was the warmest town in Germany yesterday?*
- *There were how many sunny days in Berlin in the last month?*
- *Show me pictures of cloud formation over Bavaria during the first week of August!*

In the remainder of this paper, the most important aspects of the five main modules of the HPQS are described. As a running example the the processing of the last of the above questions is explained in each section.

2 The natural language interface (NLI)

2.1 The case for an NL interface

The acceptance of a multimedia system depends crucially on the design of its user interface. Ideally, the user interface should hide the complexity of the program, thus providing the view of an easy-to-use "information assistant". Furthermore, the interface must be well adapted to the needs of average users who may be competent specialists in their field but who do not know or do not want to learn the peculiarities of retrieval techniques or languages. The usual way of querying information systems in terms of keywords and Boolean connectives does not meet these requirements. For the following reasons it is not well suited to querying in a multimedia domain either:

- *Absence of keywords.* First of all, there are obviously no *words in the images* being searched for and hence keyword matching is not possible. In our framework, descriptors are assigned automatically (on demand) by means of application-dependent methods operating on the multimedia documents. These image analysis methods typically provide *much more* than keywords; e. g. local relations between regions of interest can be computed along with a description of the regions themselves. This relational information plays an important role in image interpretation. It may make a difference to users planning their vacation in Italy whether "there are clouds" in the current weather image or whether "there are clouds in Italy". Keywords and boolean connectives, however, are obviously not sufficient for expressing such structured, model-based information.
- *Semantic querying.* Keywords with Boolean connectives are a rather cumbersome way of specifying a user's search request. Even in the textual case, users are seldom interested in documents in which only some search keywords happen to occur. Put succinctly, texts are more to humans than a set or distribution of word occurrences; they are pieces of natural language with an associated *meaning* and informational content which may or may not fulfill the user's information needs. Ideally, users should be permitted to query an information system on this level of meaning.
- *Intuitive interface.* Another problem with the usual query interfaces is their lack of user-friendliness. Although some of these interfaces (e. g. GLIMPSE, see [26]) offer powerful querying mechanisms like approximative search, regular expression search, adjacency operators, and search in specific document fields, most users are presumably not capable of taking full advantage of these features because they do not know when to apply them. By contrast,

natural language provides an intuitive interface because everybody knows how to use his native language without effort. Therefore, providing an NL front-end not only relieves the user from learning yet another querying language, but also removes technical barriers in accessing the more advanced features of an information retrieval system.

2.2 The NatLink interface

The NL interface NatLink (**nat**ural **l**anguage **in**terface to **k**nowledge) aims at constructing of adequate semantic representations for a broad class of acceptable queries and texts in general. In contrast to many other linguistic formalisms, emphasis is laid on the semantic acceptability of NL input, not on its grammaticality. In particular, the robustness issue plays an important role. This includes the proper treatment of unknown words, elliptic sentences, and slightly ungrammatical sentences.

Robustness of NL analysis conflicts to some extent with the goal of generating expressive and deep semantic representations. For example, if a word is not found in the computational lexicon, morphologic analysis and syntactic context will usually provide very shallow information about the semantics of the given word. In the approach taken here, the depth of semantic analysis dynamically adapts to the syntactic and semantic information being available, resulting in a trade-off between robustness and depth of semantic representations.

A prerequisite of this approach is that the target formalism supports semantic representations on different levels of granularity or specificity. MESNET (see [16, 20]), a multilayered extension of semantic networks, has been designed to fulfill these requirements. Due to its multidimensional structure of classificatory knowledge, it is also possible to handle generic and individual concepts in MESNET as well as intensional vs. extensional aspects of meaning. MESNET has shown to be useful as

- the semantic representation for computational lexica ([27, 28]);
- the target language for NL analysis ([18, 19]);
- the basis for a translation into formal queries to information retrieval systems ([17]).

The natural language input[1] is analyzed by NatLink (see Figure 2) according to the principles of the *word-class controlled functional analysis* (WCFA, see [18, 19, 15]). Like other word-oriented approaches (e. g. [9], [6]), WCFA supports incremental parsing which improves the system's robustness.

The *word class functions* (WCFs), which WCFA is based on, roughly correspond to the traditional parts of speech but are usually more fine-grained. They

[1] Currently, the natural language used is German. The examples in this paper are translated.

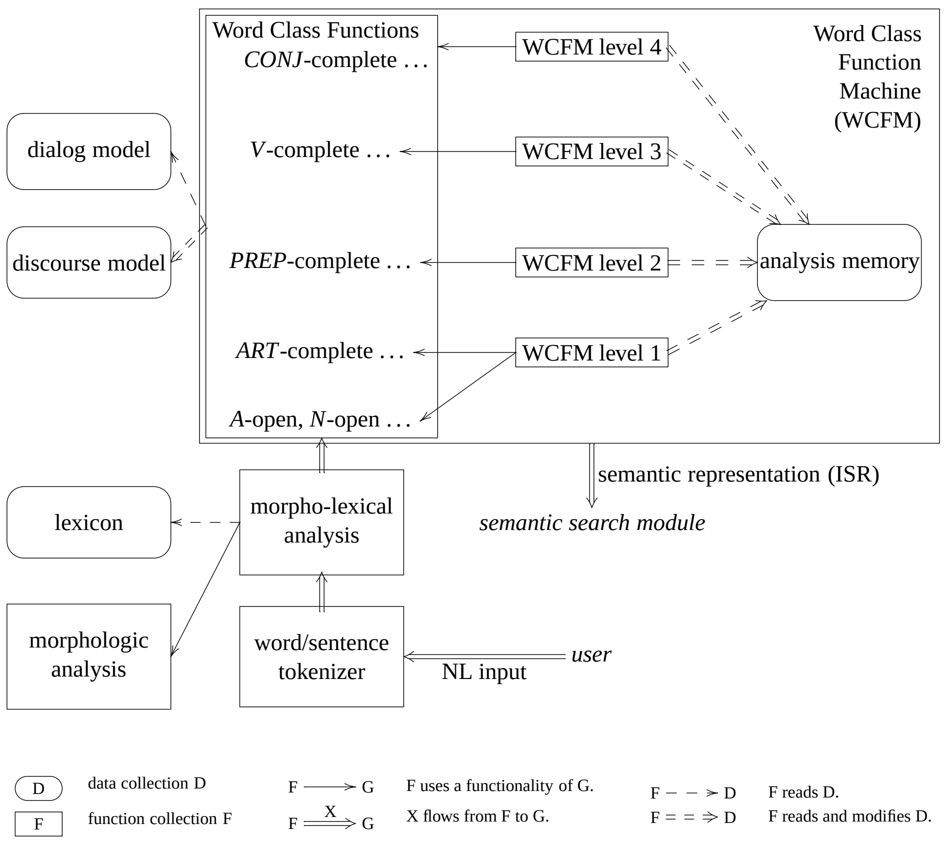


Fig. 2. Architecture of NatLink's WCFA implementation

comprise common nouns, proper nouns, interrogative determiners, interrogative pronouns, etc. All morpho-syntactic and semantic word information relevant to the analysis process is stored in the computational lexicon using typed feature structures. The lexicon is hierarchically structured, which ensures reusability, consistency, and extensibility of information by means of multiple inheritance with defaults (see [13, 14]).

WCFA parsing is *expectation-oriented.* After morphologic and lexical processing of a word, an "opening act" is performed which generates syntactic and semantic expectations (valency, agreement, etc.). The WCF also specifies "completing acts", which perform the saturation or refinement of the expectations stipulated during the opening act. When performing a "closing act", the analysis of the current constituent is completed and may then be used by the WCFA parser to fill other expectations.

Semantic representations of so-called *semantic kernels* (e. g. for noun phrases, prepositional phrases) are constructed as soon as possible during analysis in order to resolve syntactic ambiguities in an early processing phase. Semantic kernels constitute the minimal units which a semantic representation can be assigned to in the course of incremental parsing. An example of an ISR expression, which is passed on to the next component (the semantic search module) as NatLink's analysis result, is shown in Figure 3.

One central problem in constructing semantic representations for NL sentences is the disambiguation of different kinds of ambiguities as the system should infer the true meaning the user had in mind. NatLink uses hybrid disambiguation methods (see for example [30]) that use symbolic knowledge (e. g. interpretation rules for prepositions) and subsymbolic knowledge (e. g. frequency statistics semi-automatically generated from NL corpora).

The WCFA parser is *modularly structured* in that the process of NL analysis is decomposed into four different levels, roughly corresponding to elementary nominal phrases, complex nominal phrases, elementary propositions, and complex propositions. It covers a large fragment of German syntax relevant to natural language access to databases. The lexicon contains a considerable percentage of German lexemes that occur frequently in the given application domain. In the next project phase, lexical and grammatical coverage will be extended and NatLink will be equipped with an appropriate dialog model.

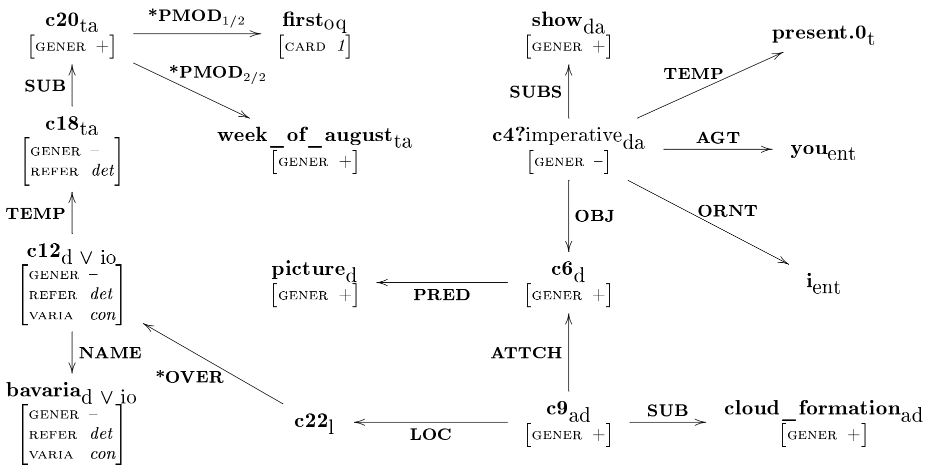

c20ta [GENER +]
*PMOD1/2
firstoq [CARD 1]
*PMOD2/2
week_of_augustta [GENER +]
SUB
c18ta [GENER − REFER det]
TEMP
c12d ∨ io [GENER − REFER det VARIA con]
NAME
bavariad ∨ io [GENER − REFER det VARIA con]
*OVER
c22l
LOC
pictured [GENER +]
PRED
showda [GENER +]
SUBS
c4?imperativeda [GENER −]
TEMP
present.0t
AGT
youent
ORNT
ient
OBJ
c6d [GENER +]
ATTCH
c9ad [GENER +]
SUB
cloud_formationad [GENER +]

Fig. 3. Simplified ISR expression (in graphical form) generated by NatLink for the query: *Show me pictures of cloud formation over Bavaria in the first week of August!* (*Zeige mir Aufnahmen von Wolkenbildung über Bayern in der ersten Augustwoche!*)

3 The Retrieval-Module

Information retrieval (IR) differs from database querying not only in that the objects involved are far less structured, but also by the fact that with IR the relevance of an object with respect to a given query must be treated as an inherently gradual relation. With NL input, there are additional sources of gradual information, i.e. the inherent fuzziness or vagueness of natural language concepts has to be dealt with on the query side. On the interpretation side, knowledge about the constituting features of the concepts in the query is often incomplete. This results in a partial mismatch between the semantic representation of the question and the target concept. Instead of postulating "crisp" methods using Boolean evaluations, it seems more natural to represent such knowledge by fuzzy rules and judge the degree of applicability of these rules by means of gradual evaluations.

Consequently, the well-structured view of the multimedia system provided by the mediator component (see below) must be complemented by powerful means for representing vague queries and gradual search results. In order to meet these requirements, a formal retrieval representation (FRR) was developed, which extends the object query language by connectives from fuzzy logic and operators for information aggregation and data fusion.

3.1 The formal retrieval representation (FRR)

Domain independent FRR methods —The generic part of FRR comprises:

- *A sophisticated text-search module*, which builds on the very fast implementation of edit-distance matching provided by the VLSI text search processors.
- Standard *image processing primitives* which are also partly implemented in VLSI hardware, e.g. two-dimensional convolution, edge detection, median filter, histogramming and others;
- *Discrete and parameterized fuzzy sets* and corresponding *fuzzy connectives*;
- *Interpolation methods* which estimate a continuous-time parameter curve from a discrete time-series;
- *Information fusion methods* which combine related information taken from different sources (e.g. weighted average, fuzzy integral, and other data fusion operators [25, 5]).
- *Fuzzy quantifiers* [12] to evaluate quantifying expressions in NL queries (*almost everywhere*, *never*,...);

FRR hence provides domain-independent search methods for all considered multimedia formats, as well as fuzzy information combination operators. These are the focus of our work because they are a prerequisite of

- The evaluation of complex NL queries involving expressions of quantification, comparison, or other types of aggregation;

– The processing of NL queries which refer to an underlying *domain model*.

In the meteorological domain, for example, users are typically interested in certain weather conditions in a specified local and temporal region. Weather conditions, however, are *not fully specified by any single document* in the database; for example, satellite images provide the required data on cloudiness, while ground temperature readings can be obtained from temperature maps. In addition, more than one document may describe the same aspect of a weather situation. Therefore, operators for information combination are required which establish content-related links between the available documents and allow for the fusion of (possibly conflicting) information extracted from several document sources. These tasks are crucial to the management of the various levels of abstraction within our system.

Special emphasis has been laid on an adequate treatment of fuzzy quantification. This is because the "modes of combination" of natural language, i. e. the various ways in which concepts might be interrelated, are by no means restricted to the Boolean connectives *and*, *or* and *not*. Unlike those, NL offers a variety of (more subtle) aggregation operators, among which *fuzzy quantifiers* (like *many*, *few*,. . .) are of particular relevance due to their abundance in NL queries. The meaning of NL queries depends heavily on these quantifying expressions, as witnessed for example, by the different meaning of *there are few clouds over Italy* vs. *there are lots of clouds over Italy*, which both could be part of queries submitted to our system. In addition, fuzzy quantifiers are an important research topic because they form a class of powerful yet human-understandable operators for information aggregation.
In order to adequately cover the meaning of these fuzzy quantifiers, an axiomatic approach to fuzzy quantification has been developed [12] which builds on the novel concept of a Determiner Fuzzification Scheme (DFS). DFS theory overcomes several limitations of existing approaches to fuzzy quantification (e.g. [33, 8]). It is the first theory to model fuzzy quantifiers of *arbitrary dimension*, it is not limited to absolute (*at least n*,. . .) and proportional (*most*,. . .) quantifiers; it is not limited to finite universes of discourse; and, most importantly, it is *based on a rigid axiomatic foundation*. The theory has been fully implemented and is used in the HPQS system for interpreting quantifiers in NL queries, and for various purposes of information aggregation. An example will be given below.

Domain dependent methods —In addition to the generic part of FRR, domain-specific methods must be implemented which provide for an operational interpretation of natural language domain concepts based on the raw document data (or intermediate search results). These domain methods can build on generic FRR methods for text and image search provided by the FRR.

The prototypical implementation in the meteorological domain serves to validate the fuzzy search mechanism. These sample evaluation methods include:

– Cartographic projections of the specific image classes under consideration;
– Access to temperature readings in false-colour ground-temperature maps;
– Objective ("more than 20° Celsius") and subjective ("warm") classification of temperatures;
– Estimation of cloud-top height and cloud density in satellite images;
– Estimation of degrees of cloudiness.

Combined with the generic FRR methods described above, temperatures and degrees of cloudiness can be interpolated; average, minimal and maximal values can be determined; evaluations can be aggregated across time and space using fuzzy quantifiers etc. As the user queries are matched against a weather model which combines information extracted from related documents, selection or rejection of a document under a given query may depend on the information provided by other documents. By considering these content-related links between associated documents, a much finer-grained and purposive search can be achieved than is possible with standard methods.

3.2 Query translation step

The retrieval module translates ISR query representations generated by the NL interface into FRR queries, which are then passed to the mediator module for evaluation. The translation process includes

– *Normalization*: mapping of terms to their database correlates (e.g. of names of cities to geographic identifiers);
– *Anchoring in discourse situation* (e.g. resolution of temporal deictic expressions like "today");
– *Default assumptions* whenever needed, e.g. in order to limit the scope of a region search to Europe;
– *User interaction* to validate decisions on alternative interpretations.

The transformation is accomplished by application-specific transformation rules which construct the query FRR from the ISR graph. The premise part of each transformation rule specifies the structure of subgraphs to which the rule applies (e.g. temporal or local specifications, domain concepts). The consequence parts of the rules provide the corresponding FRR expressions from which the query FRR is constructed.

3.3 Query execution step

For performance reasons, the FRR methods are implemented and actually executed in the lower layers of the system, namely on the parallel media server and (whenever possible) on VLSI hardware. In order to enable remote access to these methods, they are wrapped into corresponding classes and methods of the mediators object-oriented database schema. The retrieval module is shielded from the intricacies of remote query execution: in order to invoke an FRR operation, it simply invokes an OQL query to the mediator module, which then caters for the remote (and possibly parallel) execution of the corresponding FRR operation. NL queries are usually decomposed into a sequence of such mediator queries.

3.4 Example of semantic search in HPQS

In order to show how a semantic search based on interrelated documents is actually performed in HPQS, we consider the sample query: *Show me pictures of cloud formation over Bavaria during the first week of August!* This query requests weather images C from the database with an associated date $C.date$ in the first week of August 1997 subject to the condition $R(C)$

C witnesses a weather situation of cloud formation over Bavaria.

While the user simply requests *documents* of the database (in opposition to question-answer systems and expert systems), the condition R refers to an underlying *domain model* (in this case, of meteorology) in which the concept of "cloud formation" over some specified region may be given a procedural interpretation.

Considering a single image C is not sufficient to compute $R(C)$ because "formation" is essentially a matter of change and thus can only be judged relative to other images (in this case, images of relevance to the weather situation *before* $C.date$). Therefore, at least the weather image immediately preceding C must be taken into account. The task of detecting the relevant documents is decomposed as follows: "cloud formation" in a given local region is interpreted as a strong increase in cloudiness in that region in two consecutive weather maps. The system thus scans through the sequence of images in the specified time interval (first week of August 1997). For each image C under consideration, it firstly determines the predecessor image P. The following operations are then applied to C and P :

a) Compute estimates $\mathtt{C_1 = C.cloudiness()}$, $\mathtt{P_1 = P.cloudiness()}$ of the density of low clouds in C and P;
b) Compute the fuzzy evaluations (grey-value images)
$\mathtt{C_2 = C_1.sunny().negation(), P_2 = P_1.sunny();}$
c) Transform the results into the cartographic projection of the region B under consideration – in this case, `B = Bavaria`, yielding
$\mathtt{C_3 = C_2.germanyProjection(), P_3 = P_2.germanyProjection();}$
d) Combine C_3 and P_3 *relative to the given region* B by applying a suitable fuzzy quantifier $\mathcal{M}(\geq 70\,\%)$ to form the gradual result
$$R = \min\{\, \mathcal{M}(\geq 70\,\%)(B,\, C_3),\ \mathcal{M}(\geq 70\,\%)(B,\, P_3)\,\}.$$

R is the fuzzy conjunction of the conditions that at least 70 % of Bavaria are sunny in the predecessor image P and that more than 70 % of Bavaria are cloudy in the current image C (for details on the underlying theory of fuzzy quantification and the definition of $\mathcal{M}(\geq 70\,\%)$, see Glöckner [12]).

Intermediate results of the search process are shown in Figure 4. As indicated by the search results obtained from real data in Figure 5, the system detects images in which there is a clearly visible increase in cloudiness.

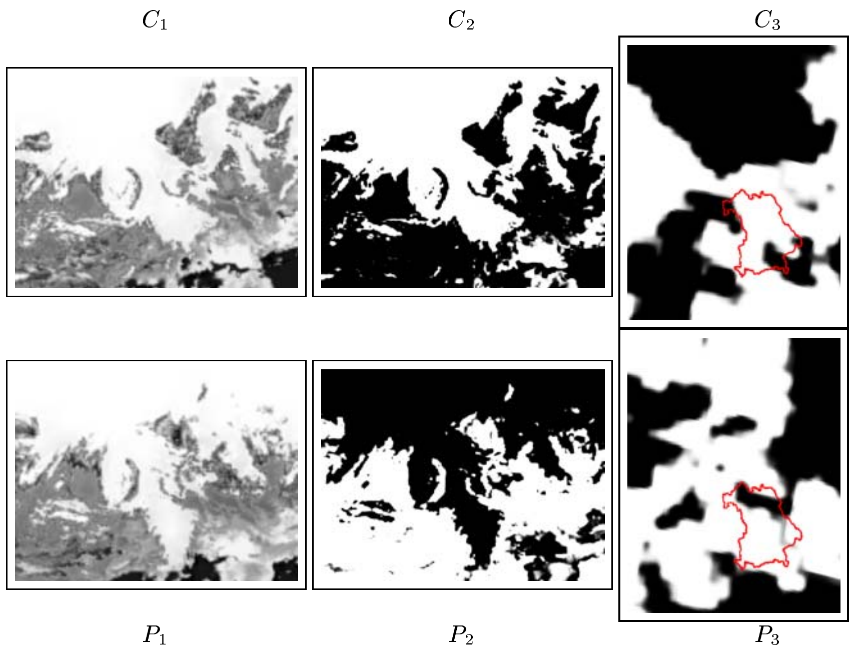
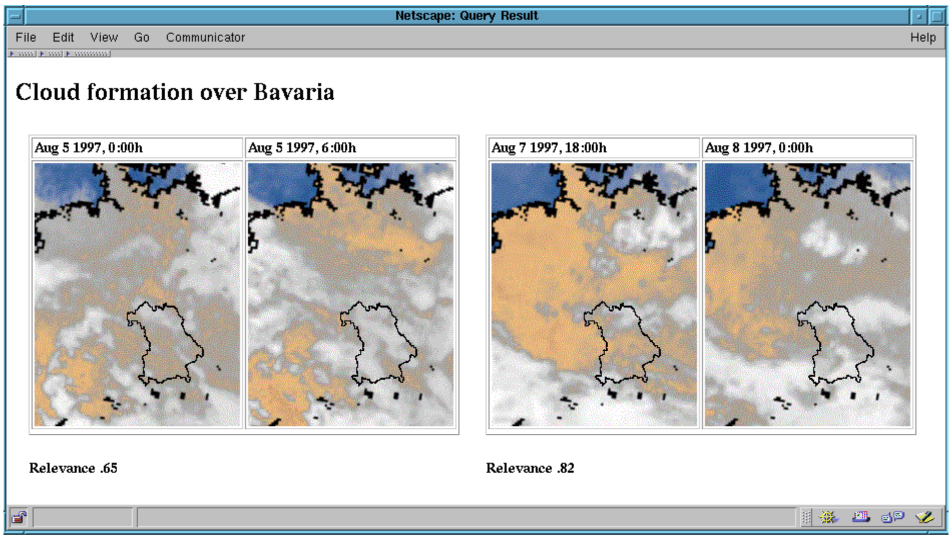


Fig. 4. Intermediate search results (C image under consideration, P predecessor of C)

4 Multimedia Mediator

The paradigm of a *mediator* was first advocated by G. Wiederhold [31]. The HPQS' **M**ulti**m**edia **M**ediator, or MMM for short, distributes an OQL query to the external sources making the best possible use of their parallelism and translates return values to the format specified in the query for a uniform presentation. Within the HPQS we have only one external source, namely the Parallel Server (see below).



Cloud formation over Bavaria

Aug 5 1997, 0:00h | Aug 5 1997, 6:00h | Aug 7 1997, 18:00h | Aug 8 1997, 0:00h

Relevance .65

Relevance .82

Fig. 5. Detected documents (relevant segments)

4.1 Architecture

Figure 6 illustrates the MMM's overall architecture. The core of the mediator is based upon an object-oriented database. The database consists of a *Multimedia Schema* defined by a mediator administrator and a *Proxy State* for external multimedia items conforming to the schema.

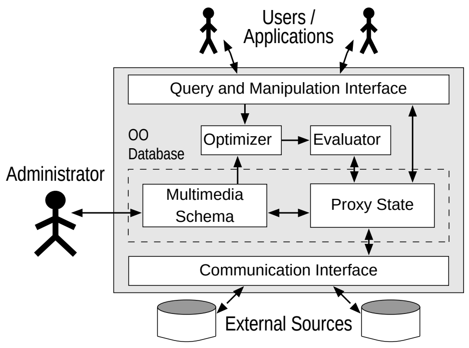


Fig. 6. Architecture of the MMM

Users or applications can query the proxy state as well as manipulate data by communicating with a *Query and Manipulation Interface*. Such requests are passed to the database, possibly after having been improved by an *Optimizer*.

Most operations on proxy objects have to be reflected to the multimedia items they represent. This reflection is performed by a *Communication Interface*, which contains specialized communication methods such that it can contact the external sources containing the multimedia items.

4.2 Optimizing queries

The optimizer is based on query transformations. In general this transformation process translates a single query into two passes with several subqueries. When the subqueries of the first pass are evaluated, they produce all necessary parameters for retrieving the data under consideration from the external sources. After actually retrieving these data, another query is evaluated in the second pass, which combines the subqueries' intermediate results for answering the original query. We demonstrate the evaluation of one of the subqueries generated by the retrieval module from the sample question:

```
select ImageAndRelevance(
  image: I,
  relevance: BAY.rateGreaterEqual(0.7,
          I.cloudiness().sunny().negation().germanyProjection()))
from I in q_138
```

where the objects in the set `q_138` represent image files stored on the parallel server together with the required operations `cloudiness`, `sunny`, `negation`, and `germanyProjection`. Each of these specific operations performs some image processing and returns the processed image. The query is thus translated into a series of subqueries:

```
R1: select I.P_cloudiness() from I in q_0138
R2: select I.P_sunny() from I in R1
R3: select I.P_negation() from I in R2
R4: select I.P_germanyProjection() from I in R3
```

These queries are evaluated sequentially and after each evaluation the resulting parameter lists are sent to the parallel server in order to retrieve the required attributes. `rateGreaterEqual` is a method defined locally in the mediator. Thus, we can finally evaluate the combining query, which is identical to the original query in our example. If an external source is capable of parallel processing, like the HPQS' Parallel Server, the parameter lists returned by the subqueries are sent to the source as a set, so the source can execute one method on several lists of parameters in parallel.

In order to reduce communication overhead and avoid expensive external computations the mediator can materialize query results and intermediate results for future use, i.e. for each retrieved multimedia item a *proxy object* is created which contains or points to the retrieved value. Then this proxy object can be used for evaluation of further queries referencing it.

When materializing values, we have to deal with several problems, including the choice of a *materialization strategy*, i.e. under which conditions to materialize an object, assumptions on *incomplete knowledge*, i.e. in case of set values whether to evaluate under the closed world assumption or a partial open world assumption, and the choice of *refreshment strategies.* A more detailed discussion of these topics is beyond the scope of this paper but can be found in [3] and [4].

5 The parallel media server

The main task of the parallel server is firstly to manage the mass data that is stored in the HPQS, and secondly to perform search operations on these mass data items. As the execution of search operations is the most time-consuming operation of the request processing, the introduction of parallelism promises a substantial cut in the overall response time of the system. Furthermore, the server is scalable and can be expanded if necessary.

The parallel server stores objects on distributed disks, delivers objects, inserts new objects, and deletes objects. Since these are also the typical tasks of a WWW server, its communication channels are based on the HTTP protocol. The parallel server can apply search methods and other procedures *directly* and simultaneously on different mass data items. There are two main interfaces with the other modules in the HPQS:

- The parallel server interacts with the mediator. The latter can put, delete and receive data items from the parallel server. Additionally, the interface enables the mediator to start the parallel execution of methods on the parallel server.
- The parallel server also interacts with the hardware search processors. The server feeds data into the search processors and receives the results of the operations that are forwarded to the mediator. Thus, the search processors act as clients of the parallel server, which itself is a client of the mediator.

5.1 Structure of the server

The server is mainly constructed as a parallel WWW-Server that also executes methods on data items. Figure 7 shows the architecture of the server, which is composed of the data management unit and the execution unit. Each incoming request is examined and then processed by the appropriate unit, both of which share the same object database.

The *data-management unit* consists of a dispatcher and a couple of worker processes on each processor node. The dispatcher examines the incoming requests

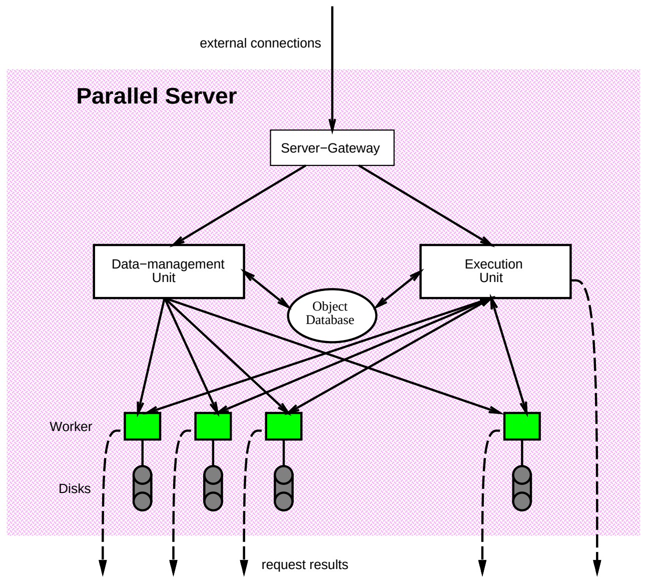


Fig. 7. The structure of the parallel server system

and decides by which node the request is to be processed. The appropriate workers then parse the request and perform the necessary disk accesses. Since there are several workers on each node, requests can be processed in parallel on each node. The *execution unit* first schedules the jobs which are necessary for processing the request. The jobs are then executed on the appropriate nodes; the results are collected and sent back.

There are several problems with designing such a combined data storage and processing server as efficiently as possible. One problem arises from the question of data placement: which data should be stored on the disk of which node? The disks are the slowest part in the parallel server and each disk can only provide one access per time unit. It is therefore important to distribute the data over the disks in such a way to allow as many accesses to different disks in parallel as possible.

The main idea of the data distribution is to place data items that are often accessed simultaneously on different disks. We achieve this goal by analyzing the access pattern of the past and by using a graph-theoretic model in order to determine the distribution [24].

Another problem arises from the scheduling of the methods. These jobs should be distributed over the nodes as evenly as possible to make sure that they can be finished as early as possible. Nonetheless, each job needs special data and should be executed on the node which stores most of these data. This problem can be formalized into a new scheduling problem not examined so far. It turns out that even the special case with only two possible processing nodes is NP-hard. We have therefore developed heuristics which solve the scheduling problem in a fast and efficient way.

5.2 Processing of the example request

The example requests resulting from the sample question leads to a couple of method invocations on the parallel server. On each picture that has to be investigated for the selected week a number of operations have to be executed. Table 1 shows the times that are needed for one execution of each of these functions.

Method	**Avg. time** [s]
`mefExtFilteredFindLowClouds`	14.2
`ociExtSunny`	0.3
`fiExtNegation`	0.2
`germanyProjection`	4.0

Table 1. Execution times of methods invoked by the sample question

The total sequential processing time of the methods for this request is $(14.2 + 0.3 + 0.2 + 4.0) \cdot 56$ s $= 1047.2$ s ($\approx$ 18 minutes), assuming that there are 56 relevant images which have to be examined. Using the parallel server with n nodes, each of the methods can be executed on all 56 images in parallel. So the total processing time would be $\lceil \frac{56}{n} \rceil \cdot 18.7$s. Assuming that there are 8 computing nodes, this results in a total time of $\approx$ 2.2 minutes. These times apply to the software solution without the VLSI search processors.

6 Search Processors

To ensure acceptable response times for the HPQS, basic and frequently used algorithms for image processing and full text retrieval will be implemented in dedicated processors. The image processing algorithms contain 2-dimensional and weighted median filtering as well as histogramming. The text algorithm allows approximate (fuzzy) full text search based on a dynamic programming procedure.

The dedicated processors are located on an add-on card. Integrated into the parallel server they communicate with the host CPUs via a PCI interface. The host CPUs perform the pre- and postprocessing functionality with low computational complexity but with a high grade of flexibility for the special processors.

The image processor component for 2-dimensional filtering provides different window sizes. Through the application of new resource-sharing concepts [32] a flexible trade-off between throughput rate and chip real estate is possible. The solution for the weighted median filtering component is based on the odd/even-transposition algorithm [21]. Beyond flexible window sizes it features efficient implementations for different throughput rates through the use of a bit-serial approach. The histogramming component achieves a high throughput rate through the use of histogram value partitioning and a dedicated synchronous full-custom memory based on the 3-transistor DRAM cell.

The text processor will be a solution optimized for approximate (i.e. fault tolerant) full text search. Therefore, the underlying basic algorithm was extended to enable the processing of language specific and special characters as well as wildcards and multi token words. The main part of the processor is a systolic array of processing elements with a throughput of more than 130 Mill. characters per second and a computational power of nearly 17 Bill. operations per second.

The work on the dedicated processors follows a special design style. Components critical to time, area, and especially power consumption are assembled as highly optimized full-custom macros by a datapath generator [11]. Starting from a VHDL description of the component's signal flow graphs, this datapath generator assembles the macro layouts from abutment cells. These abutment cells are derived automatically from a library of a very few handcrafted and optimized leaf cells. This exploits the inherent regularity and locality typical for signal flow graphs of digital signal processing for the optimization of chip area, throughput rate and especially power consumption. Moreover, the automation offers the possibility for effectively optimizing the signal flow graph by simply modifying the VHDL description. By contrast, irregular control logic is implemented using a common semi-custom design style.

7 Conclusions

We have presented the prototypical implementation of a complex digital library system whose search and presentation capabilities reach beyond current systems. The user input is assigned a semantic interpretation. This enables a fine grained and precise search if methods are available in the specific knowledge domain that analyze documents for their semantic contents and produce result documents utilizing data fusion.

The system is fully operational with some limits on the classes of questions that it can be asked; the data sets it works on are already on the order of several Gigabytes of text and image data. The response time for answering questions covering up to 100 colour images is currently on the order of a few minutes. The latter will drop to fractions of a second once the hardware processors currently under development become available.

Lines of future research are the analysis of video and audio documents so as to include them in the search process, the optimization of the mediator and the database as well as the generation and presentation of the result documents.

Acknowledgement —The authors wish to acknowledge the valuable contribution Volker Jentsch made by establishing this research group.

References

1. A. Del Bimbo, M. Campanai, and P. Nesi. A three-dimensional iconic environment for image database querying. *IEEE Trans. on Software Eng.*, 19(10):997–1011, October 1993.
2. A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 19(2):121–132, February 1997.
3. J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. A mediator for multimedia systems. In *Proceedings 3rd International Workshop on Multimedia Information Systems*, Como, Italia, Sept. 1997.
4. J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. Query evaluation in an object-oriented multimedia mediator. In *Proceedings 4th International Conference on Object-Oriented Information Systems, Brisbane, Australia*, Berlin, Nov. 1997. Springer.
5. I. Bloch. Information combination operators for data fusion: a comparative review with classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(1):52–67, 1996.
6. Norbert Bröker, Udo Hahn, and Susanne Schacht. Concurrent lexicalized dependency parsing: The ParseTalk model. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*, 1994.
7. IBM Corp. http://www.software.ibm.com/data/mediaminer/immn0b15.html.
8. D. Dubois and H. Prade. Fuzzy cardinality and the modelling of imprecise quantification. *Fuzzy Sets and Systems*, 16:199–230, 1985.
9. Michael Eimermacher. *Wortorientiertes Parsen.* PhD thesis, TU Berlin, Berlin, 1988.
10. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkhani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9), September 1995.
11. Michael Gansen, Frank Richter, Oliver Weiß, and Tobias G. Noll. A datapath generator for full custom macros of iterative logic arrays. *Proceedings of the IEEE 1997 International Conference on Application Specific Systems, Architectures, and Processors*, pages 438–447, July 1997.
12. Ingo Glöckner. DFS – an axiomatic approach to fuzzy quantification. Technical Report TR97-06, Technische Fakultät, Universität Bielefeld, 1997.
13. Sven Hartrumpf. Redundanzarme Lexika durch Vererbung. Master's thesis, Universität Koblenz-Landau, Koblenz, June 1996.
14. Sven Hartrumpf. Partial evaluation for efficient access to inheritance lexicons. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP-97)*, pages 43–50, Tzigov Chark, Bulgaria, September 1997.
15. Hermann Helbig. Syntactic-semantic analysis of natural language by a new word-class controlled functional analysis. *Computers and Artificial Intelligence*, 5(1):53–59, 1986.

16. Hermann Helbig. Der MESNET Primer – Die Darstellungsmittel der Mehrschichtigen Erweiterten Semantischen Netze. Technische Dokumentation, FernUniversität Hagen, Hagen, Germany, January 1997.
17. Hermann Helbig, Carsten Gnörlich, and Dirk Menke. Realization of a user-friendly access to networked information retrieval systems. In *Proceedings of the AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, pages 62–71, Stanford, CA, 1997.
18. Hermann Helbig and Sven Hartrumpf. Word class functions for syntactic-semantic analysis. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP-97)*, pages 312–317, Tzigov Chark, Bulgaria, September 1997.
19. Hermann Helbig and Andreas Mertens. Word Agent Based Natural Language Processing. In Loe Boves and Anton Nijholt, editors, *Proceedings of the 8th Twente Workshop on Language Technology – Speech and Language Engineering, Twente, 1 and 2 December 1994*, pages 65–74, Enschede, 1994. Universiteit Twente, Fakulteit Informatica.
20. Hermann Helbig and Marion Schulz. Knowledge representation with MESNET: A multilayered extended semantic network. In *Proceedings of the AAAI Spring Symposium on Ontological Engineering*, pages 64–72, Stanford, CA, 1997.
21. Christiane Henning and Tobias G. Noll. Architecture and implementation of a bitserial sorter for weighted median filtering. *Proceedings of the 1998 Custom Integrated Circuits Conference, Santa Clatra, CA*, May 1998.
22. T. Hermes, C. Klauck, J.Kreyß, and J. Zhang. Image retrieval for information systems. In *Proc. SPIE's Symp. on Electronic Imaging*, San Jose, February 1995.
23. S. Iyengar and R. Kashyap. Special section on image databases. *IEEE Trans. on Software Eng.*, 14(5):608–688, May 1988.
24. Jörg Jensch, Reinhard Lüling, and Norbert Sensen. A data layout strategy for parallel web servers. In *Proceedings of EuroPar '98*, 1998.
25. A. Knoll, R. Schröder, and A. Wolfram. Fusion of data from fuzzy integral-based active and passive colour stereo vision systems for correspondence identification. In *Proceedings of the VIII European Signal Processing Conference (EUSIPCO-96)*, Trieste, Italy, Sept. 10-13 1996.
26. U. Manber and S. Wu. GLIMPSE: A tool to search through entire file systems. Tr 93-34, Department of Computer Science, University of Arizona, Tucson, Arizona, 1993.
27. Marion Schulz. *Eine Werkbank zur interaktiven Erstellung semantikbasierter Computerlexika.* PhD thesis, FernUniversität Hagen, Hagen, 1998.
28. Marion Schulz and Hermann Helbig. COLEX: Ein Computerlexikon für die automatische Sprachverarbeitung. Informatik-Bericht 210, FernUniversität Hagen, Hagen, Germany, December 1996.
29. Virage Inc. http://www.virage.com.
30. Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors. *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040 of *LNAI*. Springer, Berlin, 1996.
31. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.
32. Wolfgang Wilhelm and Tobias G. Noll. A new mapping technique for automated design of highly efficient multiplexed fir digital filters. *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2252–2255, June 1997.
33. L.A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Comput. and Math.*, 9:149–184, 1983.