Content-based retrieval in multimedia databases based on feature models

Peter Apers[1] and Martin Kersten[2]

[1] University of Twente, Enschede the Netherlands,
apers@cs.utwente.nl,
WWW home page: http://www.utwente.nl/~apers
[2] CWI, Amsterdam, the Netherlands
mk@cwi.nl
WWW home page: http://www.cwi.nl/~mk

Abstract. With the increasing popularity of WWW, the main challenge in computer science has become content-based retrieval of multimedia objects. Until now access of multimedia objects in databases was done by means of keywords. Now, with the integration of feature-detection algorithms in database systems software, content-based retrieval can be fully integrated with query processing. In this invited paper, we describe our experimentation platform under development that fully integrates traditional query processing and content-based retrieval and that is based on feature databases, making database technology available to multimedia.

1 Introduction

Large scale multimedia information retrieval is one of the major scientific challenges of this decade. This focus of attention results from significant advances in technology to capture and store raw material in databases and in files on the world-wide web [?]. As a result the research field has entered a third stage.

First generation multimedia database systems focussed on *blobs* to efficiently store the sizeable objects. Since the early 90s, database vendors provide support for these non-interpreted byte streams in their core products, leaving timing, synchronization, and quality-of-service to specialized co-processors. Video-on-demand applications slowly make their way into the homes.

The second generation concerned techniques for annotation and linking media objects. Most of this activity found itself a breeding ground in user interface research and multi-media authoring system. The database merely contains the textual annotations and search accelerators using conventional information retrieval techniques.

The third wave of multi-media database retrieval research concerns itself with developing effective techniques for indexing and retrieval by content [?][?]. The ideal searched for are algorithms to automatically index objects according to a semantic framework.

Unfortunately, this vision is not feasible in the foreseen future. And probably in general not possible either, because semantic descriptions are too tightly coupled with the frame or reference (domain) of the intended audience.

At best what we can hope for is to make progress in derivation of syntactic features that aid pre-selection in a large multi-media database. Progress in this area is already demonstrated for retrieving still-images based on color distribution, directionality, texture etc.

In this paper we present ongoing research in the area of content-based retrieval using a novel view on the architecture of a multi-media database search-engine. The key scientific questions driving our research are:

content-based retrieval How to effectively process a user's query by mapping concept relationships onto the basic features of multimedia objects stored in a database.

feature databases How to efficiently derive simple and complex multi-media features for widely distributed sources of raw material and making this available as an index for query resolution.

The importance of these research challenges is illustrated by the abundance of funding available worldwide. Within the Netherlands alone, the authors are involved in the following large national programs:

AMIS This national project, bringing together researchers from image processing, computer graphics, database technology, and operating systems, focuses on indexing and searching of multimedia databases. [**?**]

Digital Media Warehouse This project, which runs under the umbrella of the Telematics Institute, brings together academia and industry to look at the usage of multimedia database in a cooperative environment [**?**][1].

Work presented in this invited paper is heavily based on [**?**] and [**?**]. It is organized as follows. In Section 2 a motivating example is given. This is followed by Section 3 in which we introduce the architecture of 3rd generation multi-media database system geared at content-based multi-media information retrieval. In the two following sections, these issues are elaborated on a bit more: Section 4 discusses our research line content-based retrieval and Section 5 the feature database model and processing scheme to construct a database of index data. In Section 6, we indicate challenges ahead and secondary roads to be explored.

2 An example

To set the stage for research, the following informal example illustrates the scope of problems to be dealt with.

Imagine a journalist looking for information for a TV news item on El Niño and its effect on weather. What he needs are some video fragments and background data to support his story. For his work he has access to a distributed, multimedia database containing news items. We will look at the way he searches depending on what is provided by the database.

[1] (http://www.cwi.nl/∼acoi/DMW)

In the simplest case all video fragments are labeled with relevant keywords. In this case he would search for fragments labeled with the keyword El Niño. This would give a perfect *precision*, but a very low *recall*. Meaning that he would of course find all fragments with the right label, but would skip all fragments that were not explicitly labeled El Niño. A more advanced database may have subtitles included. Many programs nowadays have subtitles for deaf people. In this case it would be possible to apply *Information Retrieval* techniques to search for El Niño among the subtitles. One step further would be that video fragments are searched in which an ocean is shown and that the corresponding audio track contains spoken text regarding worms stream before the coast of Peru. The holy grail is that the multimedia retrieval engine can conclude from a video that it concerns the effects of a warm ocean stream before the coast of Peru on the weather in other parts of the world.

In the first example the search is a *boolean search* with which we are very familiar in databases. The query is formulated in terms of what is available in the database. The second example requires the integration of database query processing with *Information Retrieval techniques*. Both the query and the contents of the database (the subtitles) require some processing, e.g. words are replaced by their stems. In the third example the gap between the query formulated by the user and what is in the database becomes larger. In this case features are used. *Features* are complex functions that are applied to the raw representation of a multimedia object. Examples of features are: color distribution, directionality, circularity, texture etc. Of course a user will never be able to phrase his query in terms of feature values. For this type of search it is more common to search for a key frame from a video fragment that comes close to what one is looking for and ask the system to find other videos that have feature values close to the example provided. Via *relevance feedback* the user can indicate which of the returned video fragments are better than other ones, in this way changing the relative weight of each of the features.

The final example is of course something for the future. It requires the attachment of concepts (semantics) to video fragments and of relationships among video fragments.

3 MIR Architecture

The structure of the multi-media information retrieval (MIR) architecture pursued is illustrated in Figure **??**. From the top level the system consists of four components: a (web-based) query interface, a multi-media query processor, a feature detector engine, and an extensible database server. Their role within the overall architecture is summarized as follows:

Query interface

The query interface for multimedia databases differ considerable from the traditional straight line approach encountered in OQL or SQL. The query formulation

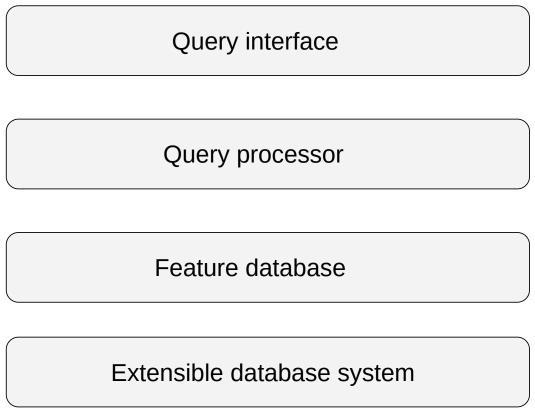

Fig. 1. Multi-media information retrieval architecture

is composed of a mix of textual descriptions, component clipping, and expression of temporal /spatial /topological relationships.

We have taken a pragmatic approach to assume that such interfaces are largely built on Java with an identifiable clean interface to a database. Except for occasional demonstrators we do not invest in the area of user-interfaces per se.

Multimedia query processing

In [?] it was stated that the three fundamental issues in probabilistic information retrieval are: representation of documents, query formulation, and a ranking function. These three issues come back in the three layers (see Figure **??**): *concept space*, which manages the basic concepts in documents, *evidential reasoning*, which implements the ranking function, and *relevance feedback*, which implements the query facility.

The basis for these layers is a database extended with proper metrics to relate objects within feature space. *Feature clustering* is used as a first step from points in a feature space to concepts. For text retrieval the features are the words, or their stem, in a text document. These words very much correspond to concepts in real life. For multimedia documents, the computation of a feature is a point in an abstract feature space, often without semantic meaning in real life. Feature clustering algorithms have been proposed as the prime means to cluster documents, such that similarity queries are easily (and efficiently) answered. The hypothesis underlying this approach is that clustering leads to kind of concept - not necessarily semantic meaningful- for query processing.

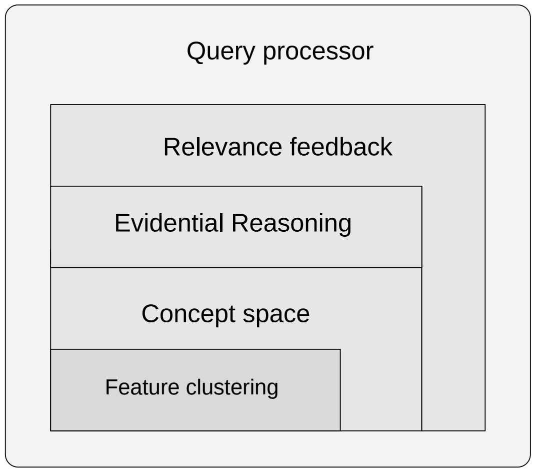

Fig. 2. Architecture multimedia query processor

Feature databases

The query processing layer is supported by a feature database, i.e. an instantiation of a feature space model. Such models differ from traditional storage models - object-oriented and relational - in their rich provision for partial- and multi-view descriptions of the underlying object.

In many cases it is insufficient to derive a single feature value, e.g. color distribution, for a complete object, but we should detect and store discriminative features for object components. Including retention of temporal and spatial relationships. The state of the art further impose a constraint to deal with an ever-increasing set of such feature types and their costly detection.

Given the distribution and volatility of the underlying database, construction of a multi-media feature index is a continuing activity. At any time, the query processor should be aware of the fact that the feature index is incomplete to answers all requests immediately.

Database support

Realization of a feature database calls upon the facilities offered by modern extensible databases. They provide both the facilities to deal with multi-media data items and the mechanisms to implement the necessary search accelerators for boosting the metric search in feature spaces.

The underlying database system deployed is Monet, a novel and powerful extensible DBMS. Monet provides the concept of modular extension, a technique in line with *data cartridges* and *data blades*, which encapsulate the routines and data structures for a particular data type. In particular, the system provides modules to support GIS, images, and videos. Results on their functionality and performance have been reported elsewhere [?].

In the following subsections we discuss the motivations and design considerations of the two core layers in our MIR architecture.

4 Content-based retrieval

The goal of our work on content-based retrieval is to take the strong points of Information Retrieval and adapt them to multimedia retrieval.

Concept space

As mentioned, in multimedia retrieval there is an enormous gap between the real world concepts from the user perspective and the feature spaces the system is using. On the one hand a sunset and on the other hand a RGB value; the former is a concept that can be used in different contexts, the latter is a probably unique point in a large feature space.

Before using these feature points as indexing term they have to be clustered. Two techniques are used: supervised and unsupervised feature clustering. In *supervised* clustering the system is trained to learn the differences between two concepts. Similar to the way neural networks work. At GMD in Darmstadt experiments have been done to learn the system to see the difference between "type of light" (artificial or natural) [?]. The result is that from then on a user can refer to the concept articial light and the system knows how to translate that to feature values. One of the objections against this method is that manually it has to be decided which concepts are relevant.

In *unsupervised* clustering documents are clustered based on the proximity of their feature points in the feature space. Very likely the significance of this proximity reveals an underlying concept. However this concept may not be well understood in real life, and is therefore not exposed to the user. We expect that these detected concepts may be of great help in query processing. This is based on the results of the FourEyes learning agent for the Photobook image database [?]. Experiments will have to show whether we are right.

Evidential reasoning

This part of the system has to determine which documents are relevant for a particular query. In Information Retrieval this is of course done by means of a ranking function. The relevance of a document for a particular query is based on the evidence found in the representation of a document. Matching documents to a query is based on a theory called *evidential reasoning.* The evidence is based on the presence or absence of concepts derived in the concept layer, very similar to traditional Information Retrieval.

Many models exist to implement evidential reasoning, for example, probability theory, Demster-Shafer theory, fuzzy logic [?], and Bayesian belief networks

[?]. The architecture is such that any of these can be used. In the miRRor project [?] we have chosen for Bayesian networks. In spite of the fact that Bayesian inferencing is NP-hard [?], it becomes tractable for more restricted networks. INQUERY, a text retrieval system, is based on the inference network model [?]. The additional advantage of using Bayesian networks is to handle several features, possibly coming from different media of one document. For example, evidence that a document is the right document for a particular query can come from subtitles, keyframes from a video, and the corresponding audio track. Research shows that evidence obtained from different representations is a better support and gives better results.

Relevance feedback

In [?] we argued that the best way to handle multimedia queries is by means of a *dialogue* between the user and the database system. The idea is that during this dialogue the low-level concepts are identified that are relevant for the user.

Basically, there are two approaches to relevance feedback: query-space modification and document-space modification. In *query-space modification* [?] the relative importance of terms is adjusted. For example, a set of picture is presented to a user. Based on positive and negative feedback the weight of the various low-level concepts or features is adjusted, resulting in a follow-up query.

In *document-space modification* [?] concepts of a document are added or dropped based on a large set of queries for which this document is found relevant. The representation of a document (its attached low-level concepts) is adjusted based on the fact that this document should or should not be included in a result set of queries.

Although we regard both types of modification as important, currently we focus on query-space modification. One of the major advantages is the fact that this can be done during query execution.

5 Feature database

Query processing a multi-media database presupposes a rich feature database. The designer of a feature database is challenged with finding a balance between flexibility, storage, and performance. Flexibility to support a broad spectrum of possibly proprietary feature detectors, to store their multi-dimensional results in a database with ease of access, and high performance to permit index construction. In this section we discuss the ingredients to built them focusing on the model requirements and construction of the feature database.

Feature models

The model proposed is based on the observation that indexing an arbitrary multimedia object leads to a hierarchical structure that describes the components of interest for the search. Such hierarchical structures are concisely described with formal grammars. Our interpretation of parsing, however, slightly differs from conventional techniques in language processing.

To recall, we describe a language of properties using a grammar $G = (V, T, P, S)$ where V is a collection of variables, T a set of terminals, P productions of the form $V \rightarrow (V \cup T)^*$, and S the start symbol taken from V. A sentential form α is a string of terminals and variables, such that $S \overset{*}{\rightarrow} \alpha$. The collection of parse trees is denoted by PT.

A sub-language $L(G_w)$ is described with the sub-grammar $G_w = (V_w, T_w, P_w, w)$, taking a consistent subset of the corresponding components of G. It describes the structure of sub-sentences in the language $L(G)$.

The terminals T are ordinary typed lexicals. The built-in set of types encompasses the traditional programming types `int` $\cdots$ `str`. Furthermore, type extensibility of Monet provides for more complex types, such as `image`. The terminals are collected into token sequences or sentences $TS = [t_0(v_0), \cdots t_k(v_k)]$ where $t_i \in T$ is an atomary type name, and v_i a value in $domain(t_i)$. A token sequence ts belongs the language $L(G)$, i.e. ts is parsed against grammar G, if there exists a sequence of productions such that $S \overset{*}{\rightarrow} ts$.

Turning back to our main objective, we consider a feature database a collection of sentences with indexing values. Their parse tree denotes a hierarchical structure and provides a name space to access and manipulate components. Actually, there exists a natural mapping from sententials to complex objects. In particular, the (non-)terminals are mapped into object attributes; repetition into a list constructor; and alternatives as elements in abstract classes. When a class description is needed for application interfacing, it can readily be derived and refined with application specific behavior. This leads to the following observation:

Definition 1. For $v \in V \cup T$ the class C_v denotes the class of complex objects equivalent to the sub-language G_v.

Feature detectors fit in this framework as operations associated with non-terminals, which massage a token sequence to steer correct parsing of the corresponding sub-language. For this they may inspect the parse tree under construction (its sentential form).

Definition 2. A *feature detector* $d \in D \subset V$ is a function that maps a token sequence $w \in TS$ into $w' \in TS$ using its parse tree d_t, such that the head of w' is a sentence in the sub-language G_d.

A feature detector may involve user interaction to identify the element in F or even extend F in the classification process. For example, the detector could ask the user explicitly for the classification information using a dialogue initialized with a set of choices *ask("car","house",...)* or to let the user draw geometric structures on the screen to identify the portions of interest, e.g. *faces.*

Ordinary functions differ from the feature detectors in that the information derived is not kept permanently in the database for recall. As such, they are also total functions instead of partial functions (over the database extent).

Since detectors may be introduced long after the database has been created, the indexing process necessarily is incremental, because the source may not be available at all times. This leads to two sub-classes for any class C as follows:

Definition 3. The object class indexed by feature detector d is denoted by $\vec{C}_d$. Those not yet indexed are denoted by $\overset{\rightharpoonup}{C}_d$. At any time class $C_d = \vec{C}_d \cup \overset{\rightharpoonup}{C}_d$.

```
%ATOM       image;
%ATOM       str server, directory;
%ATOM       str basename, extension
%ATOM       int width, height;

%DETECTOR url;
%DETECTOR picture;
%DETECTOR icon(image);
%DETECTOR avatar SELECT thumnail WHERE picture.width=48
            AND picture.height=64;

mmo:        url category;
url:        server directory* basename extension;
category:   thumbnail | avatar;
thumbnail:  picture icon;
picture:    image width height;
icon:       picture;
```

Fig. 3. A Feature Grammar Example

To illustrate, consider the feature grammar defined in Figure **??**. The top part defines atoms (typed terminals) and feature detectors. Detector `avatar` is a *white*-box detector; its behavior is defined by an expression understood by the feature detection tool kit. The other detectors are black box detectors, known by their name only. It is up to the user to supply an implementation. Their body may inspect parse tree - it provides access to contextual information- and change the token sequence to assure proper continued parsing.

The bottom part contains a grammar for a hierarchical structured feature space. An object o that is known to obey this grammar has an implied syntax tree where the edges are labeled with the names of the corresponding production rules. Components of this parse tree can be accessed with regular (path) expressions.

Unlike traditional grammars, alternation between `thumbnail` and `avatar` is not exclusive. Both productions describe alternate views on the same underlying object. The `category` rule succeeds when for all succesfull alternatives produce the same remaining token pool for continuation. An alternative that fails is further ignored.

Observe that semi-structured databases follow the same pattern, a document is a hierarchical composition whose structure is conveniently described by a grammar (e.g. SGML, HTML, XML, Hytime). However, in Acoi we expect an a priori geven grammar and do not derive the schema on the fly from the documents in the database.

Feature engines

Feature extraction is a time consuming operation, because the multi-megabyte source is often stored remotely and the detection algorithm is often compute intensive. With our focus on the volatile web as the primary source for retrieving multi-media objects calls for a mechanism to schedule feature detection activities. In an ideal situation, all relevant features for a given user query have been pre-computed and consolidated in the query result. More often, though, we may have to calculate on the fly feature properties, such that the user can be satisfied with (at least) a partial answer. Unlike traditional database queries it is out of the question to wait for all objects of interest to be processed.

An informal description of how the feature grammar is used to obtain the index runs as follows (using the feature grammar in Figure **??**). At some point in time, a string (e.g. "http://www.cwi.nl/∼ monet/lady.gif") is inserted in the token pool from which the grammatical structure is parsed. The start node `mmo` creates a parsing context that ultimately leads to acceptance or rejection of the object as a `mmo` object. This proof is attempted by proving the right hand side of the `mmo` rule, which starts with calling the `url` detector. It searches the pool for a string and breaks it into components as follows:
`[server(www.cwi.nl), directory(∼ mk), basename(lady), extension(gif)]`
and the detector returns SUCCEED. The modified token pool is consumed by the parser looking for a valid url. The `mmo` rule can then proceed with the `category` proof with two alternatives, `thumbnail` and `avatar`, both are valid continuations.

The `thumbnail` rule triggers the detector `picture`. Its body has access to the complete parse tree built so far. It uses this information to access the file being referenced and determine its type from the extension component. Upon success (it is a `gif` file) it opens the corresponding file and generates atoms `[image(cache/lady.gif), width(85), height(250)]` pushed in front of the token queue.

Subsequently the `icon` detector is called with the most recent `image` object as parameter. It derives a small icon, leaving it behind in the token stream for consumption as `[image(cache/lady.icon.gif), width(75), height(75)]`. When `thumbnail` proof has ended successfully, the category proof proceeds with the next alternative, `avatar`.

The `avatar` is an example of a predicate-based detector. The `thumbnail` argument sets the context. But there are two pictures available in the parse tree (thumbnail and icon). Therefore, the path should explicate the context to locate the correct `width` and `height`.

The `category` rule succeeds if at least `thumbnail` or `avatar` reports success. When the complete `mmo` rule has been proven the original string object has been parsed into a hierarchical structure containing classification and feature information.

This execution model gives a systematic parsing method to classify a new object. The feature detector engine uses this method to steer feature detector behavior. Basically classification is based upon the success or failure of parsing the token sequence. The detectors merely assure that the proper classification information is available just in time.

6 Conclusions

For multimedia retrieval one of the main challenges is to translate concepts in the real-world environment of users to features that can be computed from the raw data of multimedia objects. And, furthermore, fully integrate multimedia retrieval with tradational query processing thereby showing the full power of database technology.

Feature database can concisely be described with a formal grammar, which captures both the inter-component structure and provides the semantic basis for evidential reasoning. A direct mapping to either a relational or object-relational scheme makes is attractive intermediate model. As such, the approach is in line with XML, where the grammatical structure and annotations form the underlying model for the information modelled.

The novel way to look at parsing as a concerted action of multiple feature detectors, make the approach emendable for wide scale (and parallel) deployment against multi-media indexing on the web.

Currently, a large scale experimentation platform is under construction to demonstrate the technology against a database of 1M images and other multimedia objects gathered from the web.

7 Acknowledgements

We would like to thank our researchers working in our multimedia projects. They are the ones that have to realize our ambitious goals. A special word of thanks goes to Arjen de Vries, Niels Nes, and Menzo Windhouwer for their research contribution to this invited paper.

References

1. AMIS, Advanced Multimedia Indexing and Searching, http://www.cwi.nl/ acoi/Amis/index.html
2. P.M.G. Apers, H.M.Blanken, M.A.W. Houtsma (eds), Multimedia Databases in Perspective, Springer Verlag, ISBN 3540761098, June 1997.
3. P. Boncz and M.L. Kersten, Flattening an object algebra to provide performance, ICDE, 1998.

4. G.F. Cooper, The computational complexity of probablistic inference using Bayesian belief networks, Advances in Knowledge Discovery and Data Mining, AAAI Press, 1995.
5. DMW, Digital Media Warehouses, http://www.cwi.nl/ acoi/DMW/index.html
6. C. Faloutsos, Searching multimedia databases by content, Kluwer Academic Publishers, 1996.
7. R. Ferber, Accessing documents to knowledge discovery methods and intelligent retrieval, ERCIM-97-W001, pp 17-22, 1996.
8. N. Fuhr, and C, Buckley, A probabilistic learning approach for document indexing, ACM Transactions on Office Information Systems, Vol 9, No 3, pp. 223-248, July 1991.
9. M.L. Kersten, M.A. Windhouwer, and N.J. Nes, A Feature Database for Multimedia Objects, Proc. workshop ERCIM DBRG, May 1998, Schloss Birlinghoven, Germany.
10. MiRRor, Multimedia Information Retrieval Reducing information OveRload, http://wwwis.cs.utwente.nl:8080/DOLLS/.
11. T.P. Minka and R.W. Picard, Interactive learning using a "society of models", technical report TR-349, MIT Media Laboratory Perceptual Computing Section, 1997.
12. S. Parsons, Current approaches to handling imperfect information in data and knowledge bases, IEEE Transactions on Knowledge and Data Engineering, Vol 8. No 3, pp. 353-372, June 1996.
13. A.P. de Vries and Henk Blanken, The Relationship between IR and Multimedia Databases, accepted for publication at IRSG'98.
14. S.E. Robertson, On term selection for query expansion, Journal of documentation, Vol 46, No 4, pp. 359-364, 1990.
15. H.R. Turtle, Inference networks for document retrieval, PhD Thesis, University of Massachusetts, 1991.
16. H. Turtle and W.B. Croft, Evaluation of an inference network-based retrieval model, ACM Transactions of Information Systems, Vol 9, No 3, 1991.
17. A.P. de Vries, G.C. van der Veer, and H.M. Blanken, Let's talk about it: Dialogues with multimedia databases. Database support for human activity, Displays, 1998, 18, 4, pp. 215-220.
18. A.P. de Vries, B. Eberman, and D.E. Kovalcin, The design and implementation of an infrastructure for multimedia digital libraries, Proc 1998 Int Database Engineering & Applications Symposium, 1998, Cardiff, UK, July, pp. 103-110.
19. S.K.M. Wong and Y.Y. Yao, On modeling information retrieval with probabilistic inference, ACM Transactions on Information Systems, Vol 13, No 1, pp. 38-68, January 1995.
20. J.K. Wu, A.Desei Narasimhalu, B.M. Mehtre, C.P. Lam, and Y.J. Gao, CORE: a content-based retrieval engine for multimedia information systems, Multimedia Systems, Vol 3, pp 25-41, 1995.