Repository Management in an Intelligent Indexing Approach for Multimedia Digital Libraries

B. Armani[1], E. Bertino[2], B. Catania[1], D. Laradi[3], B. Marin[3], G.P. Zarri[3]

[1]Dipartimento di Informatica e Scienze dell'Informazione, University of Genova, Italy
{armani,catania}@disi.unige.it
[2]Dipartimento di Scienze dell'Informazione, University of Milano, Italy
bertino@dsi.unimi.it
[3] Centre National de la Recherche Scientifique (CNRS), Paris, France
{laradi,marin,zarri}@ivry.cnrs.fr

Abstract. Metadata represent the vehicle by which digital documents can be efficiently indexed and retrieved. The need for such kind of information is particularly evident in multimedia digital libraries, which store documents dealing with different types of media (text, images, sound, video). In this context, a relevant metadata function consists in superimposing some sort of conceptual organization over the unstructured information space proper to these digital repositories, in order to facilitate the intelligent retrieval of the original documents. To this purpose, the usage of conceptual annotations seems quite promising. In this paper, we propose a two-steps annotation approach by which conceptual annotations, represented in NKRL [7], [8], are associated with multimedia documents and used during retrieval operations. We then discuss how documents and metadata can be stored and managed on persistent storage.

1 Introduction

It is today well recognized that an effective retrieval of information, from large bodies of multimedia documents contained in current digital libraries, requires, among other things, a characterization of such documents in terms of some *metadata*. A relevant metadata function consists in superimposing some sort of conceptual organization over the unstructured information space often typical of digital libraries, in order to facilitate the intelligent retrieval of the original documents. Querying or retrieving various types of digital media is executed directly at the metadata level.

Among the classes of metadata proposed by the scientific literature, only *content-specific metadata* "reflect the semantics of the media object in a given context" and provide a sufficient degree of generality [1]. Unfortunately, as well known, a veritable access by semantic content is particularly difficult to achieve, especially for non-textual material (images, video, audio). In those cases, content-based access is often supported by the use of simple keywords, or of features mainly related with the physical structure of multimedia documents (such as colour, shape, texture, etc.) [4]. In order to overcome the limitations of such approaches, *conceptual annotations* have been introduced for describing in some depth the context of digital objects [2], [3], [6]. However, the current approaches, often based on the use of simple ontologies in a

description logic style, have several limitations in terms of description of complex semantic contents (e.g., of complex events) events.

To alleviate these problems, we propose a different approach for building up conceptual annotations to be used for indexing documents stored in a thematic multimedia library. With thematic multimedia library we mean a library storing documents concerning a given application domain. Our approach is based on a *two steps* annotation process:

- During the first step, any interesting multimedia document is annotated with a simple Natural Language (NL) caption in the form of a short text, representing a general, neutral description of the content of the document. In the case of textual documents, the interesting parts of the text, or the text itself, could represent the NL caption. This approach corresponds to the typical process of annotating a paper document, by underlying the interesting parts or writing down remarks and personal opinions. In the case of other media documents, the NL caption may represent the semantic content of the document and additional observations associated with it.
- During the second phase, annotations represented by NL captions are (semi-automatically) converted into the final conceptual annotations. We propose to represent the final conceptual annotations in NKRL (*Narrative Knowledge Representation Language*) [7], [8]. In NKRL, the metaknowledge associated with a document consists not only in a set of concepts and instances of concepts (individuals) but also in a structured set of more complex structures (occurrences) obtained through the instantiations of general classes of events called *templates*. This approach is actually tested in the context of an European project, CONCERTO [9].

Note that the use of a two-steps annotation process guarantees a high level of flexibility in querying. First of all, this approach provides a general solution for the mixed media access. This means that a single metadata query can retrieve information from data that pertain to different media since the same mechanism is used to represent their content. Moreover, the first annotation step is quite useful in supporting a similarity-based indexing. Indeed, by associating similar captions to different documents we make them "similar" from the point of view of the content and therefore of the retrieval.

In designing an architecture supporting the approach described above, the component dealing with the storage and the management of all the types of knowledge (documents, templates, concepts, and conceptual annotations) on secondary storage plays a fundamental role, since its implementation strongly influences the performance of the overall system. The aim of this paper is that of presenting a proposal for designing and implementing such component, that we call *Knowledge Manager*. For this task, we have followed a Web-Based approach. In particular, the Knowledge Manager has been implemented as a true server manager that can be hosted on a generic machine connected over Intranet/Internet networks to the clients requiring such services. The advantage of this approach is that the software component we have designed can be easily used by other architectures, based on the use of NKRL or similar languages for encoding conceptual annotations.

The paper is organized as follows. Section 2 introduced NKRL whereas Section 3 introduces an approach for the internal representation of such language. The

Knowledge Manager architecture is then presented in Section 4. Finally, Section 5 presents some concluding remarks.

2 NKRL as a metalanguage for document annotations

In the following, we briefly review the basic characteristics of NKRL (*Narrative Knowledge Representation Language*) (we refer the reader to [7], [8], [10] for additional details).

The core of NKRL consists of a set of general representation tools that are structured into four integrated components, described in the following.

Definitional and enumerative components. The *definitional component* of NKRL supplies the tools for representing the important notions (*concepts*) of a given domain. In NKRL, a concept is, therefore, a definitional data structure associated with a symbolic label like `human_being`, `city_`, etc. Concepts (*definitional component*) and individuals (*enumerative component*) are represented essentially as frame-based structures. All NKRL concepts are inserted into a generalization/specialization hierarchy that corresponds to the usual ontology of terms and is called H_CLASS(es).

The *enumerative component* of NKRL concerns the formal representation of the instances (individuals) (`lucy_`, `wardrobe_23`) of the concepts of H_CLASS. Throughout this paper, we will use the italic type style to represent a `concept_` and the roman style to represent an `individual_`.

Descriptive and factual components. The dynamic processes describing the interactions among the concepts and individuals in a given domain are represented by making use of the *descriptive* and *factual* components. The *descriptive component* concerns the tools used to produce the formal representations (*predicative templates* or simply *templates*) of general classes of narrative events, like 'moving a generic object', 'formulate a need', 'be present somewhere'. In contrast to the binary structure used for concepts and individuals, templates are characterized by a threefold format where the central piece is a predicate, i.e., a named relation that exists among one or more arguments introduced by means of roles. The general format of a predicative template is therefore the following:

$$(P_i\ (R_1\ a_1)\ (R_2\ a_2)...\ (R_n\ a_n))$$

In the previous expression, P_i denotes the symbolic label identifying the predicative template, R_k, k = 1,...,n, denote generic roles, and a_k, k = 1,...,n, denote the role arguments. The predicates pertain to the set BEHAVE, EXIST, EXPERIENCE, MOVE, OWN, PRODUCE, RECEIVE, and the roles to the set SUBJ(ect), OBJ(ect), SOURCE, BEN(e)F(iciary), MODAL(ity), TOPIC, CONTEXT. Templates are structured into an inheritance hierarchy, H_TEMP(lates), which corresponds to a taxonomy (ontology) of events. The instances (*predicative occurrences*) of the predicative templates, i.e., the representation of single, specific events like "Tomorrow, I will move the wardrobe" or "Lucy was looking for a taxi" are in the domain of the *factual component*.

Example 1. The NKRL sentence presented in Figure 1 codes an information like: "On April 5th, 1982, Gordon Pym is appointed Foreign Secretary by Margaret Thatcher", that can be directly found in a textual document, contained in an historical digital library. The subject of this event is Gordon Pym, represented as a particular instance (`gordon_pym`) of the concept *`individual_person`*. The object of this event is the position Gordon Pym is appointed to, represented by the concept *`foreign_secretary_pos`*. Finally, the source of this event is Margaret Thatcher (represented by the instance `margaret_thatcher`) since she is responsible for the event. In the predicative occurrence, temporal information is represented through two temporal attributes, `date-1` and `date-2`. They define the time interval in which the meaning represented by the predicative occurrence holds. In `c1`, this interval is reduced to a point on the time axis, as indicated by the single value, the timestamp `5-april-82`, associated with the temporal attribute `date-1`; this point represents the beginning of an event because of the presence of `begin` (a *temporal modulator*).

```
c1) OWN  SUBJ gordon_pym
         OBJ foreign_secretary_pos
         SOURCE margaret_thatcher
         [begin]
         date-1: (5-april-82)
         date-2:
```

Fig. 1. Annotation of a WWW textual document

In the previous example, the arguments associated with roles are simple. However, NKRL also provides a specialized sublanguage, AECS, supporting the construction of *structured arguments* by using four operators: the disjunctive (ALTERNative) operator, the distributive (ENUMerative) operator, the collective (COORDination) operator, and the attributive (SPECIFication) operator.

Predicative occurrences can also be combined together, through the use of specific second order structures, called *binding occurrences*. Each binding occurrence is composed of a binding operator and a list of predicative or binding occurrences, representing its arguments. Each document (NL caption, in the considered framework) is then associated with a single *conceptual annotation*, corresponding to the binding occurrence representing its semantic content.

In order to query NKRL occurrences, *search patterns* have to be used. Search patterns are NKRL data structures representing the general framework of information to be searched for, within the overall set of conceptual annotations. A search pattern is a data structure including, at least, a predicate, a predicative role with its associated argument, where it is possible to make use of explicit variables, and, possibly, the indication of the temporal interval where the unification holds. As an example, the conceptual annotation in Figure 1 can be successfully unified with a search pattern like: "When was Gordon Pym appointed Foreign Secretary?", presented in Figure 2. The variable `?x` means that we want to know the instant when the event happened.

We refer the reader to [7], [8] for additional details on these topics.

```
(?w IS-PRED-OCCURRENCE
:predicate OWN
:SUBJ gordon_pym
:date-1 ?x
```

Fig. 2. A simple example of an NKRL search pattern

3 A representation language for NKRL

The usual way of implementing NKRL has been, until recently, that of making use of a three-layered approach: Common Lisp + a frame/object oriented environment (e.g., CRL, Carnegie Representation Language, in the NOMOS project) + NKRL. In order to ensure a high level of standardization, we are now realizing, in the context of the CONCERTO project [9], a new version of NKRL, implemented in Java and RDF-compliant (RDF = Resource Description Format) [5].

RDF is a proposal for defining and processing WWW metadata that is developed by a specific W3C Working Group (W3C = World Wide Web Consortium). The model, implemented in XML (eXtensible Markup Language), makes use of Directed Labeled Graphs (DLGs) where the nodes, that represent any possible Web resource (documents, parts of documents, collections of documents, etc.) are described basically by using attributes that give the named properties of the resources. No predefined 'vocabulary' (ontologies, keywords, etc.) is in itself a part of the proposal. The values of the attributes may be text strings, numbers, or other resources. In the last versions of the RDF Model and Syntax Specifications new, very interesting, constructs have been added [5]. Among them, of particular interest are the 'containers', i.e., tools for describing collections of resources. In an NKRL context the containers are used to represent the structured arguments created by making use of the operators of the AECS sublanguage (see Section 2).

A first, general problem to be solved to set up an RDF-compliant version of NKRL has concerned the very different nature of the RDF and NKRL data structures. The first are 'binary' ones, i.e., based on the usual organization into 'attribute – value' pairs. The second are 'tripartite', i.e., are organized around a 'predicate', whose 'arguments' are introduced through a third, functional element, the 'role'. To provide the conversion into RDF format, the NKRL data structures have been represented as intertwined binary 'tables' that describes the RDF-compliant, general structure of an NKRL template.

Example 3. Consider the predicative occurrence presented in Figure 1. The RDF/XML description of `c1` is presented in Figure 3. In general, the RDF text associated with each predicative occurrence is composed of several tags, all nested inside the `<CONCEPTUAL_ANNOTATION>` tag and belonging to two different namespaces: `rdf` and `ca`. The first namespace describes the standard environment under which RDF tags are interpreted. The second namespace describes specific tags defined in the context of our specific application. More precisely, the tag `<ca:Template_i>` is used to specify that the predicative occurrence is an instance

of the template identified by `Template_i`. The identifier of the occurrence is an attribute of such tag (`occ11824` in our example). The other tags specify the various roles of the predicative occurrence, together with the associated arguments. Additional tags are used to represent temporal information and modulators.

4 The Knowledge Manager architecture

Four main modules compose the architecture supporting our approach:

```
<?xml version="1.0" ?>
<!DOCTYPE DOCUMENTS SYSTEM "CA_RDF.dtd">
<CONCEPTUAL_ANNOTATION>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:ca="http://projects.pira.co.uk/concerto#">
    <rdf:Description about="occ11824">
    <rdf:type resource="ca:Occurrence"/>
       <ca:instanceOf>Template43</ca:instanceOf>
       <ca:predicateName>Own</ca:predicateName>
       <ca:subject rdf:ID="Subj43" rdf:parseType="Resource">
          <ca:filler>gordon_pym</ca:filler>
       </ca:subject>
       <ca:object rdf:ID="Obj43" rdf:parseType="Resource">
          <ca:filler>foreign_secretary_pos<ca:filler>
       </ca:object>
       <ca:source rdf:ID="Source43" rdf:parseType="Resource">
          <ca:filler>margaret_thatcher</ca:filler>
       </ca:source>
       <ca:listOfModulators>
          <rdf:Seq><rdf:li>begin</rdf:li></rdf:Seq>
       </ca:listOfModulators>
       <ca:date1>05/04/1982</ca:date1>
       </rdf:Description>
   </rdf:RDF>
  </CONCEPTUAL_ANNOTATION>
```

Fig. 3. The RDF format of a predicative occurrence

- *Acquisition module*, providing a user-friendly interface by which the user can insert documents and associate with them some short NL captions.
- *Annotation module*, that is in charge of the translation of the NL captions into the NKRL format.
- *Knowledge Manager module*, implementing the basic features for storing and managing NKRL concepts, templates, original documents, and the associated conceptual annotations on persistent storage.
- *Query module*, applying sophisticated mechanisms to retrieve all documents satisfying certain user criteria, by using conceptual annotations.

In the context of the proposed architecture, the Knowledge Manager plays a fundamental role. Indeed, since it manages the repositories on secondary storage, its implementation strongly influences the performance of the overall system. In the current architecture, the Knowledge Manager has been implemented as a server, following a Web-based approach, by using Internet derived technologies for the

communication protocol and metadata representation. In particular, the Knowledge Manager is organized according to a three-tier architecture, represented in Figure 4. The first level corresponds to the repository management on persistent storage, through the use of a specific database management system (IBM DB2 in our case); the second level is an application level, providing an easy programming interface (through a Java API) to the repository. Finally, the third level consists of a specific interface language (called KMIL) to provide access to the Knowledge Manager through a Web-Based approach. In the following, the repositories and their management as well as the communication protocol are described in more details.

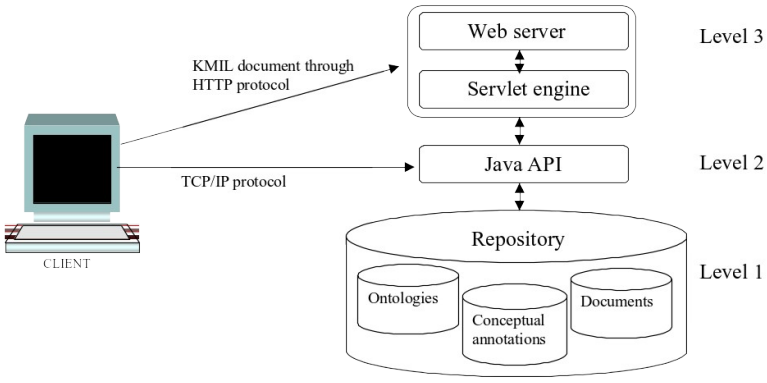

Fig. 4. General architecture of the Knowledge Manager

4.1 The repositories and their management

In order to deal with NKRL data structures, we designed three distinct but interrelated repositories. The first repository is the *Document Repository*, storing the original documents, together with the corresponding NL captions. In order to deal with conceptual annotations, the H_TEMP and H_CLASS ontologies are stored in the *Ontology Repository*. The concrete conceptual annotations, generated by the Annotation Module, are then stored in the *Conceptual Annotation Repository*.

The Conceptual Annotation Repository is certainly the most critical one since user queries are executed against it. It contains two main types of data: predicative occurrences and binding occurrences. Each predicative occurrence is characterized, among the others, by its XML/RDF text and the identifier of the template it is an instance of. For each template, we also maintain the set of predicative occurrences representing the leaves of the subtree rooted by it in the H_TEMP. The use of this information optimizes query processing since a search pattern always selects a set of predicative occurrences that are instances of a single template. Each binding occurrence is internally characterized, among the others, by the binding operator and the identifiers of its arguments (i.e., binding or predicative occurrences).

Each document is then associated with a single conceptual annotation, arbitrarily complex, describing atomic information, through the use of predicative occurrences, and combined information, through the use of binding occurrences. The repository maintains the relationship between documents and the associated conceptual annotations. It is important to note that, to guarantee a high level of flexibility, we

assume that each occurrence can be associated with different documents. This corresponds to the situation in which different documents refer similar or equal events or contain similar or equal images or sound.

Since RDF can be implemented by using XML, in order to store conceptual annotations and templates, we choose *IBM DB2 Universal Database* together with the XML extender, recently released by IBM. The repositories are then managed through the use of a Java API, implementing specific operation to be executed against the repositories. Each operation, before execution, is translated into some SQL commands to be executed by DB2. The use of a Java API provides a high level of portability for the system we have developed. Moreover, since several packages for implementing an XML parser in Java are currently available, this choice fits well in the overall system architecture. Among the supported operations, queries against the Conceptual Annotation Repository intensively use the functionalities supported by IBM DB2 and IBM DB2 XML Extender to retrieve predicative occurrences starting from given selection conditions.

4.2 The communication protocol and the interface language

The Knowledge Manager services can be executed under two different modalities (see Figure 4). In a local environment, the Java API operations are directly called and executed. In a remote environment, communication is performed through the HTTP protocol. The use of HTTP guarantees an efficient access to the Knowledge Manager from any software module located at any site on the Internet. In order to guarantee a standard communication between modules, services have to be expressed by means of an XML document. Such document has to be constructed according to a specific XML language, called *Knowledge Manager Interface Language* (KMIL). KMIL requests can be sent by using an HTTP post action to a Knowledge Manager front-end Servlet running under a specific HTTP servlet engine. This solution has the advantage that the Knowledge Manager can be hosted on a generic machine, becoming strongly independent from other modules of the architecture. All requests sent to the Knowledge Manager are then captured by a Web Server that activates a specific Java Servlet for the execution of the requested services, through the use of the Java API, on the underlying DBMS. As a result, an XML document containing the result of the computation is returned to the calling module.

Example 4. Suppose that the conceptual annotation of Figure 1 has to be inserted into the Conceptual Annotation Repository. This can be specified by using the KMIL document presented in Figure 5. Such document contains a <KMIL-ACTION> tag for the document and the predicative occurrence that have to be inserted, respectively, together with all the required information. This information is then used to consistently update the content of the Conceptual Annotation and Document repositories.

```
<?xml version="1.0"?>
   <!DOCTYPE KMIL-SESSION SYSTEM "KmilIn.dtd">
     <KMIL-SESSION>
       <KMIL-ACTION serial_number="1">
         <KMIL-INSERT-Document IdDoc="doc132" >
            <TEXT>
```

```
        On April 5th, 1982, Gordon Pym is appointed Foreign
        Secretary by Margaret Thatcher
      </TEXT>
    </KMIL-INSERT-Document>
  </KMIL-ACTION>
 <KMIL-ACTION serial_number="2">
    <KMIL-INSERT-PredOcc IdPO="occ11824" Doc="doc132">
      <TEXT> RDF Text </TEXT>
    </KMIL-INSERT-PredOcc>
 </KMIL-ACTION>
</KMIL-SESSION>
```

Fig. 5. Example of a KMIL request

5 Concluding remarks

In this paper we have presented an approach for indexing and retrieving multimedia digital documents through the use of conceptual annotations, describing in details the component entrusted with the management of documents and conceptual annotations in secondary storage. The techniques presented in this paper are now being exploited in the framework of the Esprit project CONCERTO (*CONCEptual indexing, querying and ReTrieval Of digital documents*, Esprit 29159) [9]. The aim of such project is to improve current techniques for indexing, querying and retrieving textual documents, mainly concerning the socio-economical and the biotechnology contexts. Future work includes the definition of specialized techniques for storing and indexing conceptual annotations. In particular, disk placement and caching techniques for conceptual annotations are currently under investigation in order to improve the performance of the system.

Acknowledgements. We would like to thank Pietro Leo, from IBM, for several useful comments and suggestions about the design of the proposed architecture.

References

1. S. Boll, W. Klas, and A. Sheth. Overview on Using Metadata to Manage Multimedia Data. In *Multimedia Data Management*, pages 1-24, 1998. McGraw Hill, New York.
2. D. Fensel, S. Decker, M. Erdmann, M., and R. Studer. Ontobrocker: Or How to Enable Intelligent Access to the WWW. In *Proc. of the 11th Banff Knowledge Acquisition for KBSs Workshop, KAW'98*. University of Calgary, 1998.
3. J. Heflin, J. Hendler, and S. Luke. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078, Univ. of Maryland, College Park (MA), 1999.
4. IEEE Computer - Special Issue on Content-Based Image Retrieval Systems. *IEEE Computer*, 28(9), 1995.
5. O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Technical report, W3C, 1999.
6. A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Sources Descriptions. In *Proc. of the 22nd Int. Conf. on Very Large Databases* (VLDB-96), pages 251-262, 1996.
7. G. Zarri. NKRL, a Knowledge Representation Tool for Encoding the 'Meaning' of Complex Narrative Texts. *Natural Language Engineering* - Special Issue on Knowledge Representation for Natural Language Processing in Implemented Systems, 3:231-253, 1997.

8. G. Zarri. Representation of Temporal Knowledge in Events: The Formalism, and Its Potential for Legal Narratives. *Information & Communications Technology Law* - Special Issue on Models of Time, Action, and Situations, 7:213-241, 1998.
9. G. Zarri et al. CONCERTO, An Environment for the `Intelligent' Indexing, Querying and Retrieval of Digital Documents. In *LNCS 1609: Proc. of the 11th Int. Symp. on Methodologies for Intelligent Systems*, pages 226-234, Varsavia, Poland, 1999. Springer Verlag.
10. G. Zarri and L. Gilardoni. Structuring and Retrieval of the Complex Predicate Arguments Proper to the NKRL Conceptual Language. In *LNCS 1079: Proc. of the 9th Int. Symp. on Methodologies for Intelligent Systems*, pages 398-417, Zakopane, Poland, 1996. Springer Verlag.