Università degli Studi Di Catania

Dipartimento di Ingegneria

Informatica e delle Telecomunicazioni

DOTTORATO DI RICERCA IN INGEGNERIA

INFORMATICA E DELLE TELECOMUNICAZIONI

XXIII CICLO

A Semantic-Based and Adaptive Architecture for Automatic Multimedia Retrieval Composition

Carmelo Pino

Il Coordinatore	Il Tutor
Prof. O. Mirabella	**Prof.ssa D. Giordano**

Abstract

In this thesis, we present a MMR (Multimedia Retrieval) system that is multi-domain and task independent. The use of MMR systems for different domains and tasks poses several limitations, mainly related to poor flexibility and adaptability to the different domains and user requirements. Therefore, it is desirable to have systems capable to automatically adapt to the domain they deal with. Another issue with MMR systems regards the relevance of the results that, as is known, depends strongly on the specific task. In order to address that, a retrieval system must be able to adapt automatically to a specific context and domain respecting the constraints imposed by the user and by the notion of relevance that one would like to apply. This, of course, implies that the system must understand what are the low-level features that fit a given purpose and which sequence of steps may produce satisfying results. In this thesis, a semantic-based system that uses ontologies for describing not only the application domain but also the algorithm’s steps to be performed for respecting user’s and domain requirements is proposed. More in detail, an ontological model, defined by the user, allows the system to adapt the retrieval mechanism to the application domain. Specifically, a user, by his/her requests, can choose from a set of predefined OWL ontology models and can add constraints/concepts to these models in order to formulate accurate and specific requests. According to the instances generated for each request, the system generates the interfaces (GUI) for the retrieval system specific for the domain (e.g. music, video, images) by a procedure self-guided by the defined ontology. Some examples of ontology-based GUI auto-composition and their related performance will be shown. Finally, the proposed architecture is scalable and flexible because of 1) a mechanism that allows an easy definition of new multimedia processing algorithms and 2) the use of semantic web-oriented technologies.

Acknowledgments

I would like to thank my supervisor, Prof. Daniela Giordano for her guidance while undertaking this challenging work, especially for introducing to me the problem to research and for the many ideas that she has given me to explore. Much gratitude goes to Giordano Buscemi and Giuseppe Santoro, respectively, for setting up the first prototype of the architecture and for the feature modeling part. Thanks are also due to my friends and collaborators Isaak and Concetto who have always supported me in realizing this thesis. A very special thank goes to my parents, my sister Danila, and my beloved Laura for having always believed in me.

Contents

Chapter 1

Introduction

Nowadays, the management of the large amounts of multimedia content generated every day in most of the companies has become not only an urgent need but also a success feature. Often the information is disorganized and not always the functionality offered by the software in use receive positive feedback from the user. The goal of every modern retrieval system is to provide a unique tool, simple, intuitive, able to meet the user's needs. Domain Specific Systems are often inflexible when the user request is more elaborated. Moreover, a query will produce different results in systems designed for different application domains. This thesis aims at developing a platform that is an innovation in the field of multimedia retrieval, combining the concepts of multimedia search with aspects of semantics and the benefits of indexing based on RDF. The system allows the execution of a personalized retrieval process. The user requests are mapped and interpreted and used to derive the sequence of steps for processing and retrieval of multimedia content. The user should be able to integrate into the system its reference model to conduct a search consistent with its expectations. In this work we have developed a system able to map the application domain requested by the user, based on ontological models built by the same user. In other words, each application domain can be seen as an ontology that the expert user can design. The proposed system will be able to map ontologies and algorithms for processing and matching features, adapting to user requests. Such architecture is composed of several elements whose integration allows the correct functioning. To achieve the final goal and the development of each element, a preliminary study was carried out to understand the issues that affect the architecture itself.

1.1 Research questions

The use of MMR systems turns to be problematic whenever the system has to meet the requirements dictated by the application domain. Searches of the same type of content return different results based on the types of constraint defined during the search stage that are in place and on the sequence of algorithms used for the extraction of features. As a starting point, several research questions were formulated for this thesis:

- Is there a tool that can adapt to the user’s requests?
- Is it possible to use the benefits of semantics to design a system that can adapt to the user interest domain?
- Can algorithms for features extraction designed for the elaboration of media in an application domain, be used for the same type of media but in a different application context?
- Can features extraction algorithms implemented for a given media, be adapted for the extraction of features from another media?
- Modeling the reality of interest, is it possible to integrate into the system a variety of application domains related to the same media?

These are talking about a system that is able to integrate different models and search for the media following the most appropriate model.

On the basis of this, other questions arise:

- In such system, can only one input interface and output satisfy user's needs?
- Is it possible to create automatically an interface for each model that is integrated into the system?

1.2 Requirements

Platform that can adapt to application domains need different requirements:

- Following the specification of constraints the system must be able to change the sequence of steps to perform processing of the media.

Also, the system can be used by:

- Expert users who can change its configuration in terms of algorithms and constraints.
- Novice users who will use the system from interfaces designed ad-hoc for a given application domain.

For the second type of users, we can derive the following requirements: rapid response, accuracy, simplicity in the formulation of the request, simple and intuitive interface, system reliability in terms of precision and recall, efficiency of the algorithms used for retrieval. In detail, the system requires more interfaces for:

- The creation and integration of ontology in the system.
- The creation of input interfaces that allow the user to express constraints based on the ontological model that the user wants to integrate.
- Output interface linked to the ontological model.

It also recognizes the following needs:

- Ontologies for each application domain.
- Repositories for user requests and for the results obtained from a given query.
- Web services for exposing the services available to components in the architecture.

1.3 Thesis Overview

The thesis is organized in chapters, providing first some background for the problem, then an overview of the implemented framework, followed by the descriptions of the components in more detail. Finally, experimental findings and future directions conclude the thesis. The outline of the thesis is as follows: *Chapter 2* contains the literature review and state-of-the-art. Starting with an overview of MMR systems, and then turning to MMR systems application specific, some existing architectures are surveyed. *Chapter3* presents information about the architecture of the developed system. It discusses in detail each section of the system, layer by layer. *Chapter 4* overviews the functionalities of the system, in particular, with emphasis on the functionality of interface auto composition. *Chapter 5* contains a detailed explanation of the layers namely the model developed for feature modeling, which is an important part of this project. *Chapter 6* exemplifies the system by discussing the case of the model developed for audio retrieval and the ontology developed. *Chapter 7* concludes the thesis and outlines possible future developments.

Chapter 2

Literature Review

Multimedia information retrieval deals with finding media other than text, i.e. music, pictures, and videos. With the explosion of digital media that is available on the Internet and present on users' computers techniques for quickly and accurately finding the desired media is important. Semantic-based information retrieval goes beyond classical information retrieval and uses semantic information to understand the documents and queries in order to aid retrieval. Semantic based information retrieval goes beyond standard surface information by using the concepts represented in documents and queries to improve retrieval performance. In this section we talking about MMR designed for Music, Images, Video retrieval. We will see the concepts and the use of semantic and ontologies in MMR. Moreover we will see the details of MMR architecture.

2.1 Multimedia Information Retrieval

Multimedia information retrieval (MIR) involves searching for a variety of media, such as video, music and images [49]. With the growing amount of music, video, and photos on users' computers and on the Internet the need for efficiently searching for desired media is rapidly growing. This section will take a look at the history of MIR and some of the more recent research. The earliest research on MIR was based on computer vision research [45].

Recently, researchers have been moving away from feature-based retrieval to content-based retrieval (in which try to give meaning to content mapping the LLF). There is also an increased effort to make the systems more human-centered, paying more attention to the user’s satisfaction. Many users have started using some type of MIR, through Google Video and Image Search, Altavista Audio search, etc. While not state-of-the-art, these systems are bringing MIR to the average user. There are numerous conferences and workshops on MIR. Some of the more prominent conferences include ACM SIGMM and the International Conference on Image and Video Retrieval. In addition, there are special tracks in multimedia conferences, computer vision conferences, etc. dealing with MIR. Lew et al. [45] pointed out two fundamental needs for MIR systems: searching and “browsing and summarizing a media collection”. The methods for achieving these needs fall mainly into two categories: feature-based and category-based. Recently, category-based methods have become increasingly popular, because they express the semantics of the media, which allows for better retrieval. With the two needs for MIR systems in mind, this section will continue as follows. First, we will give a look at current research in music retrieval. Next, we will look at the research on image retrieval. Then, we will look at research on video retrieval. Finally, we will talk about the future of MIR.

2.1.1 Music Retrieval

In the past 5 years there has been an explosion of music made available through services such as iTunes, Napster, eMusic, etc. Even the most casual user is quickly acquiring gigabytes of music data on their computers. And there is easily petabytes of available data on the Internet. Because of this, music retrieval is a hot topic. Downie [20] listed a number of challenges to music information retrieval including the interaction between features such as pitch and tempo. In addition, he pointed out that the representation scheme determines the computational costs, such as bandwidth. Byrd and Crawford [6] said that the same methods used in text IR, such as “conflating units of meaning”, are necessary for music IR They went on to say that music IR is much harder, because there is no agreed upon definition of what a unit of meaning is and segmentation is even much harder than segmenting Chinese [6]. What features (pitch, tempo, etc), how to represent them, and what is the basic unit of music are still in debate and being researched. Another problem is the method for querying a music database. One of the increasingly standard and popular querying methods is “query by humming.” This method allows users to find songs by humming a small portion of it. One of the earlier works done by Ghias et al. [29] focused on monophonic data and used pitch in the melodic track for representation. They converted user input data into a symbolic form based on pitch and used this form to search a database of MIDI music. Pickens et al. [60], then extended the querying technique to deal with polyphonic music data. They used a language model framework for retrieval of music performed by piano and used various methods of representation. One notable approach to music IR is to borrow from research in text IR. The previously mentioned research by Pickens et al. used the standard text IR approach of language modeling. Uitdenbogerd and Zobel [79], built an architecture using n-grams and approximate string matching. They found that using melody information was enough for practical systems and that each of the methods, n-grams and approximate string matching, worked well for certain types of music data.

Another active area of research is music filtering. This area deals with determining which music from a collection the user may enjoy. Research has been done on automatic playlist generation [62] and music recommendation [7]. Recently, work has been done by Hijikata et al. [37] on a content-based filtering system that has a user editable profile. They employed decision trees to learn profiles of users and then allow the users to edit the trees in an online environment. They used varying features such as tempo and tonality.

2.1.2 Image Retrieval

In the past few years digital photography has started to overtake traditional print photography. With the growing amount of digital images, it makes sense to have an easy and effective way to search for what is desired. Instead of looking through thousands or millions of photos it would be easier just to ask "Show me all the pictures of red cars" and get the desired set of images. Image retrieval really started in the 1970s with research done by researchers in computer vision and database management [66]. In these early days and up until the last 15 years or so, the predominant method for searching was to first annotate each image in the collection with text and then use standard text IR methods, such as [11]. Recently, as with the other areas in multimedia IR, content-based retrieval has been heavily researched. Smeulders et al. [73] broke image retrieval applications down into three categories of user views: search by association, targets the search, and category search. "Search by association" is when there is no real goal except for trying to find new interesting images. "Targets the search" is when the user has a specific image or object they are looking for. "Category search" is when users just want a picture, anyone, from a category of objects, i.e. “a car picture.” With these three categories in mind, the following paragraphs will take a look at some of the research done in the area in the last few years. Corridoni et al. [15] looked at retrieving images based on color semantics, such as warmth, accordance, contrast etc. The system allowed the users to specify certain color semantics and find images that match. Kato et al. [39] developed a system that takes a sketch done by the user and finds similar images. Bujis and Lew [4] developed the imagescape application that also allows the users to sketch in images and find images similar to it. Natsev et al. [54] used multiple signatures per image to help in computing the similarity between the given image and the images in the database. They found that this approach yielded more semantically accurate results than traditional methods. Chang et al. [10] showed that statistical learning methods help improve the performance of visual information retrieval systems They found that they needed to introduce new algorithms to deal with sparse training data and imbalance in the type of training data. Rui et al. [67] added relevance feedback to their MARS system to allow the user to guide the system in order to improve the search results Tieu and Viola [78] created a framework that uses many features and a boosting algorithm to learn queries in an online manner. They were able to achieve good results with only a small amount of training data, because they used selective features.

2.1.3 Video Retrieval

Recently, television shows, movies, documentaries, etc. have become available for download from a number of sites. In addition, digital video and home editing is becoming the norm. Video retrieval aims to help the user in finding the video they seek, whether it be a full video or just a scene. Like image retrieval some of the earliest approaches were to annotate video data and use standard IR techniques. This is still being used in modern day online video systems, such as YouTube and Google. However, with growing collections that are automatically collected from broadcast or other means, annotation is impossible. As such, automatic techniques are needed. Wactlar et al. [82] created a terabyte sized video library, and used automatically acquired descriptors for indexing and segmentation. Researchers have also tried to mimic text IR techniques in the video domain. Sivic and Zisserman [72] made analogies between text IR and video IR Their goal was to create a fast system that works on video as well as Google does on text. They pushed the analogy in every facet by doing such things as building a visual vocabulary and using stop list removal, and found that while there are still some problems the analogy to text IR worked well and appear to be promising. Video retrieval involves such tasks as content analysis and feature extraction; also, one of the most important parts of video retrieval is segmentation or partitioning [1]. Zhang et al. [89] used multiple thresholds on the same histogram to detect gradual transitions and camera breaks. Gunsel et al. looked at the use of syntactic and semantic features for unsupervised content-based video segmentation [34]. Sebe et al. list semantic video retrieval, learning and feedback strategies, and interactive retrieval as some of the new techniques used [69].

In the following some of the research done using these three techniques is covered. Naphide and Huang used a probabilistic framework to map low level features into semantic representations [53]. The semantic representations were then used for indexing, searching and retrieval. Snoek et al. developed a semantic value chain that extracts concepts from videos [74]. They used a 32 concept lexicon and were able to achieve very good performance in the 2004 TREC Video Track. Browne and Smeaton incorporated various relevance feedback methods and used object-based interaction and ranking [2]. Yan et al. used negative pseudo-relevance feedback for the 2002 TREC Video Track (TRECVID) [86]. They found that this approach increased performance over standard retrieval. Yan and Hauptman introduced a boosting algorithm called Co-Retrieval for determining the most useful features [85]. Gaughan et al. built a system that incorporates speech recognition and tested it in an interactive environment [28]. Girgensohn et al built a system focused on the user interface and used story segmentation with both text and visual search [31]. Their system was one of the best at TRECVID.

2.1.4 Semantic Based Information Retrieval

Semantic information retrieval tries to go beyond traditional methods by defining the concepts in documents and in queries to improve retrieval. In the previous section on multimedia information retrieval, we saw that there is a current trend toward content based, or semantic, retrieval. In a similar manner semantic based information retrieval is the next evolution of text IR. Some of the earliest work on semantic based IR was done by Raphael in 1964 [63]. He built the SIR system which broke down different queries/questions into different subroutines for processing. In a similar vein to Raphael, Li et al. looked at using semantic information for learning question classifiers [46]. Researchers have been bridging research done in semantic based IR and traditional natural language processing research fields. Li et al. used multiple information resources to help measure the semantic similarity between words

[47]. Varelas et al. looked at semantic similarity methods based on WordNet and how these apply to web based information retrieval [81]. The main methods for accomplishing semantic based IR are ontologies, semantic networks, and the semantic web. Ontologies and semantic networks can bring domain specific knowledge that allows for better performance. The semantic web, which has been a big buzz word for the past years, promises to bring semantic information in the form of standardized metadata. This section will continue as follows. First, we will take a look at how ontologies are being used in IR. Next, we will look at research that has used semantic maps or networks. Then, we will look at the semantic web. Finally, we will talk about the future of semantic based information retrieval.

2.1.5 Ontologies

One common form of semantic information used in information retrieval is ontology. Ontologies represent knowledge by linking concepts together and typically results in hierarchical classification. Khan et al. used an ontology model to generate metadata for audio and found an increase in performance over traditional keyword approaches [40]. Gomez-Perez et al. used an ontology for a legal oriented information retrieval system [32]. They found that the ontology helped guide the user in selecting better query terms. Soo et al. used an ontology as domain specific information to increase the performance of an image retrieval system [75]. Cesarano et al. used an ontology to help categorize web pages on the fly in their semantic IR system [9].

2.1.6 Semantic Maps and Networks

Semantic networks, which represent concepts as nodes and relations as edges in a directed graph, are a common method used for knowledge representation. They have many uses and have been used widely in semantic based IR. Cohen and Kjeldsen developed the GRANT system that used constrained spreading activation to help in the retrieval of funding sources [14]. They found that it gave a boost to recall and precision over previous systems and had a higher level of user satisfaction. Tang et al. examined self-organizing semantic overlay networks in peer-to-peer information retrieval [77]. Lin et al. examined self-organizing semantic maps [48]. They created a semantic map based on Kohonen’s self-organizing map algorithm and applied it to a set of documents. The information gained from the maps allowed for easy navigation of bibliographic data.

2.1.7 Semantic Web

The semantic web opens a realm of new possibilities for web oriented information retrieval. Shah et al. described an approach for retrieval using the semantic web [70]. They developed a prototype that allows the users to annotate their queries with semantic information from a couple of ontologies. Using this extra information they were able to significantly increase the precision of retrieval over standard text based methods. As with other semantic information, semantic web technology can help describe domain specific information that can help improve results. Mukherjea et al. used a semantic web for biomedical patents for an information retrieval and knowledge discovery system [52]. Yu et al. looked at bringing the power of the semantic web to personal information retrieval using web services [87]. One of the main problems with the semantic web is the need for annotation. However, research such as [41], [18] and [19] is working on automatic annotation methods. Dingli et al. looked at unsupervised information extraction techniques to create seed documents which are then used to bootstrap the learning process [19]. Dill et al. built the SemTag system that was designed to automatically tag large corpora with semantic information [18].

2.1.8 The Future

There are a two major problems facing semantic based IR. The first is the availability of semantic information sources. In English, this is not so much of a problem, but in other languages like Chinese, semantic resources are still scarce. The second problem is that, typically, algorithms dealing with semantics are much slower than the standard IR algorithms. In the future, as researchers in natural language processing progress in their own research on semantics these problems may not be so big. If the semantic web is able to reach its goal and automatic annotation methods are able to work precisely then in the future there should be no reason not to use semantic based IR, at least for the web.

2.2 MMR Systems: Applications and Design Methodology

Content-based retrieval has been proposed by different communities for various applications. These include:

- Medical Diagnosis: The amount of digital medical images used in hospitals has increased tremendously. As images with the similar pathology-bearing regions can be found and interpreted, those images can be applied to aid diagnosis for image-based reasoning. For example, Wei & Li (2004) proposed a general framework for content-based medical image retrieval and constructed a retrieval system for locating digital mammograms with similar pathological parts.
- Intellectual Property: Trademark image registration has applied content-based retrieval techniques to compare a new candidate mark with existing marks to ensure that there is no repetition. Copyright protection also can benefit from content-based retrieval, as copyright owners are able to search and identify unauthorized copies of images on the Internet. For example, Wang & Chen (2002) developed a content-based system using hit statistics to retrieve trademarks.
- Broadcasting Archives: Every day, broadcasting companies produce a lot of audiovisual data. To deal with these large archives, which can contain millions of hours of video and audio data, content-based retrieval techniques are used to annotate their contents and summarize the audiovisual data to drastically reduce the volume of raw footage.

For example, Yang et al. (2003) developed a content-based video retrieval system to support personalized news retrieval.

- Information Searching on the Internet: A large amount of media has been made available for retrieval on the Internet. Existing search engines mainly perform text-based retrieval. To access the various media on the Internet, content-based search engines can assist users in searching the information with the most similar contents based on queries. For example, Hong & Nah (2004) designed an XML scheme to enable content-based image retrieval on the Internet.

2.2.1 Design Of Content-Based Retrieval Systems

Before discussing design issues, a conceptual architecture for content-based retrieval is introduced and illustrated in Figure 1. Content-based retrieval uses the contents of multimedia to represent and index the dat. In typical content-based retrieval systems, the contents of the media in the database are extracted and described by multi-dimensional feature vectors, also called descriptors. The feature vectors of the media constitute a feature dataset. To retrieve desired data, users submit query examples to the retrieval system. The system then represents these examples with feature vectors. The distances (i.e., similarities) between the feature vectors of the query example and those of the media in the feature dataset are then computed and ranked. Retrieval is conducted by applying an indexing scheme to provide an efficient way to search the media database. For the design of content-based retrieval systems, a designer needs to consider four aspects: feature extraction and representation, dimension reduction of feature, indexing, and query specifications, which will be introduced in the following sections.

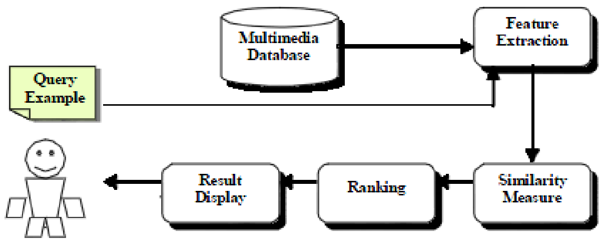


Figure 1 A conceptual architecture for content based retrieval

2.2.2 Feature Extraction And Representation

Representation of media needs to consider which features are most useful for representing the contents of media and which approaches can effectively code the attributes of the media. The features are typically extracted off-line so that efficient computation is not a significant issue, but large collections still need a long time to compute the features. Features of media content can be classified into low-level and high-level features.

- **Low-Level Features**

Low-level features such as object motion, color, shape, texture, loudness, power spectrum, bandwidth, and pitch are extracted directly from media in the database. Features at this level are objectively derived from the media rather than referring to any external semantics. Features extracted at this level can answer queries such as “finding images with more than 20% distribution in blue and green color," which might retrieve several images with blue sky and green grass. Many effective approaches to low-level feature extraction have been developed for various purposes.

- **High-Level Features**

High-level features are also called semantic features. Features such as timbre, rhythm, instruments, and events involve different degrees of semantics contained in the media. High-level features are supposed to deal with semantic queries (e.g., "finding a picture of water" or "searching for Mona Lisa Smile"). The latter query contains higher-degree semantics than the former. As water in images displays the homogeneous texture represented in low-level features, such a query is easier to process. To retrieve the latter query, the retrieval system requires prior knowledge that can identify that Mona Lisa is a woman, who is a specific character rather than any other woman in a painting. The difficulty in processing high-level queries arises from external knowledge with the description of low- level features, known as the semantic gap. The re-trieval process requires a translation mechanism that can convert the query of "Mona Lisa Smile" into low- level features. Two possible solutions have been proposed to minimize the semantic gap. The first is automatic metadata generation of the media. Automatic annotation still involves the semantic concept and requires different schemes for various media. The second uses relevance feedback to allow the retrieval system to learn and understand the semantic context of a query operation. Feedback relevance will be discussed in the Relevance Feedback section.

- **Dimension Reduction Of Feature Vector**

Many multimedia databases contain large numbers of features that are used to analyze and query the database. Such a feature-vector set is considered of high dimensionality. High dimensionality causes the “curse of dimension” problem, where the complexity and computational cost of the query increases exponentially with the number of dimensions (Egecioglu et al., 2004). Dimension reduction is a popular technique to overcome this problem and support efficient retrieval in large-scale databases. However, there is a tradeoff between the efficiency obtained through dimension reduction and the completeness obtained through the information extracted. If each data is represented by a smaller number of dimensions, the speed of retrieval is increased. However, some information may be lost. One of the most widely used techniques in multimedia retrieval is Principal Component Analysis (PCA). PCA is used to transform the original data of high dimensionality into a new coordinate system with low dimensionality by finding data with high discriminating power. The new coordinate system removes the redundant data and the new set of data may better represent the essential information.

2.2.3 Indexing

The retrieval system typically contains two mechanisms: similarity measurement and multi-dimensional indexing. Similarity measurement is used to find the most similar objects. Multi-dimensional indexing is used to accelerate the query performance in the search process.

- **Similarity Measurement**

To measure the similarity, the general approach is to represent the data features as multi-dimensional points and then to calculate the distances between the corresponding multi-dimensional points. Selection of metrics has a direct impact on the performance of a retrieval system. Euclidean distance is the most common metric used to measure the distance between two points in multi-dimensional space. However, for some applications, Euclidean distance is not compatible with the human perceived similarity. A number of metrics (e.g., Mahalanobis Distance, Minkowski-Form Distance, Earth Mover's Distance, and Proportional Transportation Distance) have been proposed for specific purposes.

- **Multi-Dimensional Indexing**

Retrieval of the media is usually based not only on the value of certain attributes, but also on the location of a feature vector in the feature space. In addition, a retrieval query on a database of multimedia with multi-dimensional feature vectors usually requires fast execution of search operations. To support such search operations, an appropriate multi-dimensional access method has to be used for indexing the reduced but still high dimensional feature vectors. Popular multi-dimensional indexing methods include R-tree and R*-tree. These multidimensional indexing methods perform well with a limit of up to 20 dimensions.

2.2.4 Query Specifications

Querying is used to search for a set of results with similar content to the specified examples. Based on the type of media, queries in content-based retrieval systems can be designed for several modes (e.g., query by sketch, query by painting [for video and image], query by singing [for audio], and query by example). In the querying process, users may be required to interact with the system in order to provide relevance feedback, a technique that allows users to grade the search results in terms of their relevance. This section describes the typical query by example mode and discusses relevance feedback.

- **Query by Example**

Queries in multimedia retrieval systems are performed, typically, by using an example or series of examples. The task of the system is to determine which candidates are the most similar to the given example. This design is generally termed Query By Example (QBE) mode. The interaction starts with an initial selection of candidates. The initial selection can be formed by randomly selected candidates or meaningful representatives selected according to specific rules. Subsequently, the user can select one of the candidates as an example, and the system will return those results that are most similar to the example. However, the success of the query in this approach heavily depends on the initial set of candidates. A problem exists in how to formulate the initial panel of candidates that contains at least one relevant candidate. This limitation has been defined as the page zero problem. To overcome this problem, various solutions have been proposed for specific applications.

- **Relevance Feedback**

Relevance feedback was originally developed for improving the effectiveness of information retrieval systems. The main idea of relevance feedback is for the system to understand the user’s information needs. For a given query, the retrieval system returns initial results based on predefined similarity metrics. Then, the user is required to identify the positive examples by labeling those that are relevant to the query. The system subsequently analyzes the user’s feedback using a learning algorithm and returns refined results. Among the learning algorithms frequently used to update iteratively the weights' estimation are the ones developed by Rocchio (1971) and Rui and Huang (2002). Although relevance feedback can contribute retrieval information to the system, two challenges still exist:

- the number of labeled elements obtained through relevance feedback is small when compared to the number of unlabeled elements in the database;
- relevance feedback iteratively updates the weight of high-level semantics but does not automatically modify the weight for the low-level features.

To solve these problems, Tian et al. (2000) proposed an approach for combining unlabeled data in supervised learning to achieve better classification.

2.3 Research Issues And Trends

Since the 1990s, remarkable progress has been made in theoretical research and system development for MMR. However, there are still many challenging research problems. This section identifies and addresses some issues in the future research agenda.

- **Automatic Metadata Generation**

Metadata (data about data) is the data associated with an information object for the purposes of description, administration, technical functionality, and so on. Metadata standards have been proposed to support the annotation of multimedia content. Automatic generation of annotations for multimedia involves high-level semantic representation and machine learning to ensure accuracy of annotation. Content-based retrieval techniques can be employed to generate the metadata, which can be used further by the text-based retrieval.

- **Establishment of Standard Evaluation Paradigm and Test-Bed**

The National Institute of Standards and Technology (NIST) has developed TREC (Text REtrieval Conference) as the standard test-bed and evaluation paradigm for the information retrieval community. In response to the research needs from the video retrieval community, the TREC released a video track in 2003, which became an independent evaluation (called TRECVID). In music information retrieval, a formal resolution expressing a similar need was passed in 2001, requesting a TREClike standard test-bed and evaluation paradigm. The image retrieval community still awaits the construction and implementation of a scientifically valid evaluation framework and standard test bed.

- **Embedding Relevance Feedback**

Multimedia contains large quantities of rich information and involves the subjectivity of human perception. The design of content-based retrieval systems has turned out to emphasize an interactive approach instead of a computer-centric approach. A user interaction approach requires human and computer to interact in refining the high-level queries. The research issue includes the design of the interface for relevance feedback with regard to usability and learning algorithms, which can dynamically update the weights embedded in the query object to model the high-level concepts and perceptual subjectivity.

- **Bridging the Semantic Gap**

One of the main challenges in multimedia retrieval is bridging the gap between low-level representations and high-level semantics. The semantic gap exists because low-level features are more easily computed in the system design process, but high-level queries are used at the starting point of the retrieval process. The semantic gap is not only the conversion between low-level features and high-level semantics, but it is also the understanding of contextual meaning of the query involving human knowledge and emotion. Current research intends to develop mechanisms or models that directly associate the high-level semantic objects and representation of low-level features.

2.4 MMR Architecture and Application domains

We have seen what the typical features of a MMR system are, considering that currently multimedia retrieval systems are either domain or task specific. In addition to the examples of application domains previously surveyed, other intelligent multimedia retrieval systems have been proposed recently for different applications, e.g. sport, medicine, law, etc. and for different types of multimedia contents (audio, video, 3D Model, etc..). Examples are: Zhang *et al* in [91] retrieves personalised sports video by integrating semantic annotation and user preference acquisition. MMR systems are therefore often domain specific, i.e., they are designed and developed for specific contexts. When a domain specific MMR is developed, the functionality is not based on the development of a set of features for a given media, but is based on a set of choices for indexing and searching of media in the given domain. Some example such as (**Adaptive content-based music retrieval system)** suggests how to make a special purpose leads to precise design choices. Many of the seen systems, (as functional and performance), report a lack of flexibility and adaptability for search in other contexts. Today we understand that the flexibility and scalability are important especially for such a system, both for systems performance and for exploit the hard work of planning behind the implementation of each MMR system.

Chapter 3

System Overview

In accordance with what was seen previously, we understand the need for systems capable of processing the media referring to a given application context. There are cases where some systems are designed for a given application domain (e.g. AUDIO retrieval) and for a given task e.g. audio speech retrieval. In this section we will see an overview of the developed system, explaining the layers functionality.

3.1 Requirements of a Multi-Domain MMR System

To derive the requirements for a multi-domain MMR system, let's consider the case of a MMR designed for audio speech processing. If we decide to use the same platform to process both audio from speech and audio from music, we have several problems:

- The platform designed for audio speech processing performs audio processing according to a precise sequence of steps. (fig 2).
- If we use the same platform to process music tracks, we see that the sequence required for the processing of a musical piece requires more steps, or requires a different sequence.
- Moreover, the platform handles a set of features that might be insufficient for the processing a segment of music track (fig 3).

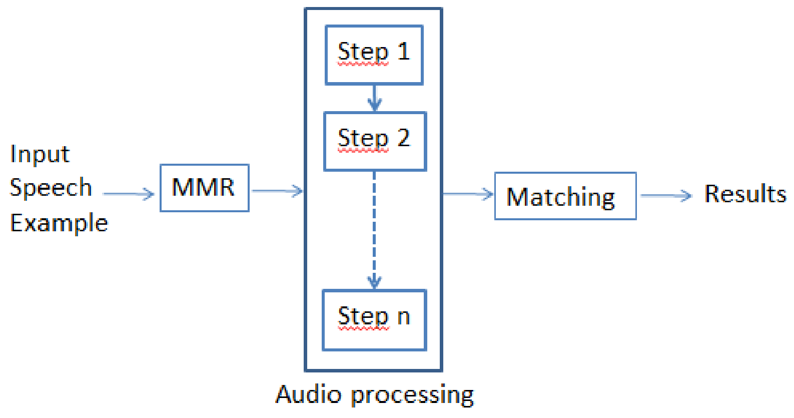


Figure 2 Example of MMR for Audio speech processing

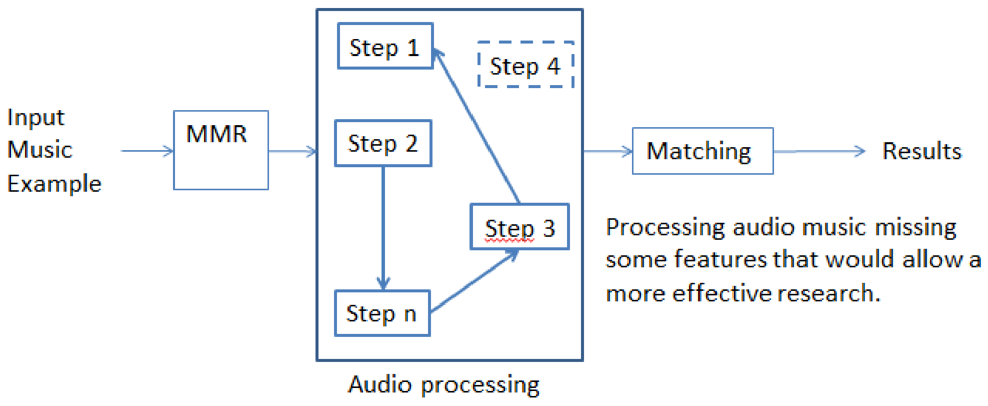

Step 1
Step 4
Step 2
Step 3
Step n
Input Music Example
MMR
Matching
Results
Processing audio music missing some features that would allow a more effective research.
Audio processing

Figure 3 Example of MMR for Audio music processing

Platform that can adapt to application domains need different requirements:

- the system must be able to change the sequence of steps to perform processing of the media, following constraints specified by the user.

Also the system can be used by:

- Expert users, who can change its structure in terms of algorithms and constraints.
- Novice or occasional users, who will use the system with interfaces designed ad-hoc for a given application domain.

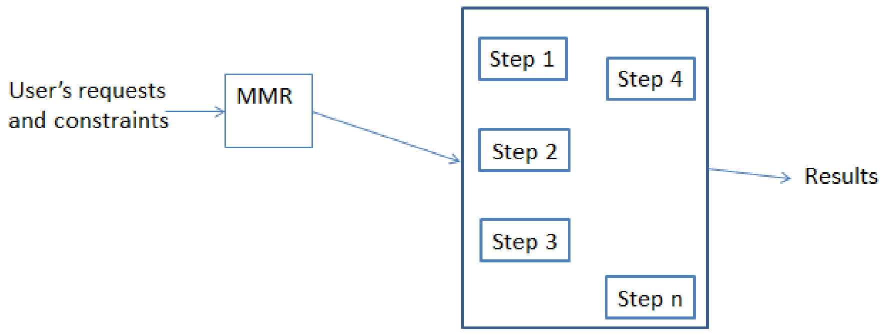


Figure 4 Example of processing with customizable process

The proposed system is a multi-domain MMR system. The functionalities of this system are targeted to the user that uses the system in order to search content, and to the developer that uses the system to create a model of MMR for its domain of interest.

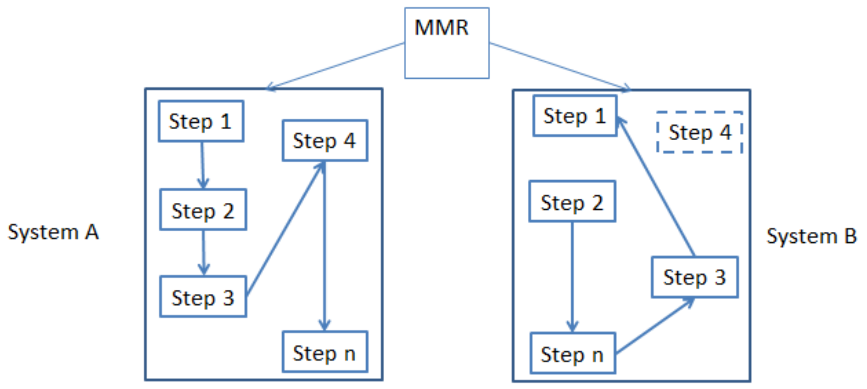


Figure 5 Example of processing with customizable process

The advantages of such system are:

- The user makes the requests and obtains a customized processing.
- The developer creates a model for its application domain.

The models developed within the systems are scalable: at any time the developer can add new features or new algorithms. The proposed system is designed to meet the needs and requirements of a MMR multi domain system. Broadly, we can identify the following components in the system architecture:

- User Layer
- Server Layer
- Repositories Layer
- Processing Layer
- Ontology Layer
- Okkam Layer

The user interacts with the system through the User Layer; this layer provides the necessary tools (interfaces and functionality) to formulate a request as accurately as possible. The User Layer forwards the requests to the Server Layer that makes them understandable to the Ontology layer. The Ontology layer contains various modules and related OWL ontological models uploaded by the users. It will interpret the constraints imposed by the user in order to return the most appropriate sequence of steps to perform the retrieval. The sequence obtained by the ontology layer will be returned to the server layer, this will proceed to the elaboration using the processing layer. The processing layer exposes a set of methods that are used for the processing of features. The Server Layer uses a sequence of these methods to implement the retrieval mechanism. The data returned from the processing will be formalized in RDF and stored in suitable repositories. Finally, taking into account the broader context of semantic web, there are some advantages that can be obtained by binding the entities with a unique identifier. In this way, the process information can be integrated with any other data source that identifies Web resources using the same globally recognized identifier. This is achieved by the Okkam Layer.

3.2 System Architecture

The proposed system architecture is sketched in figure 6.

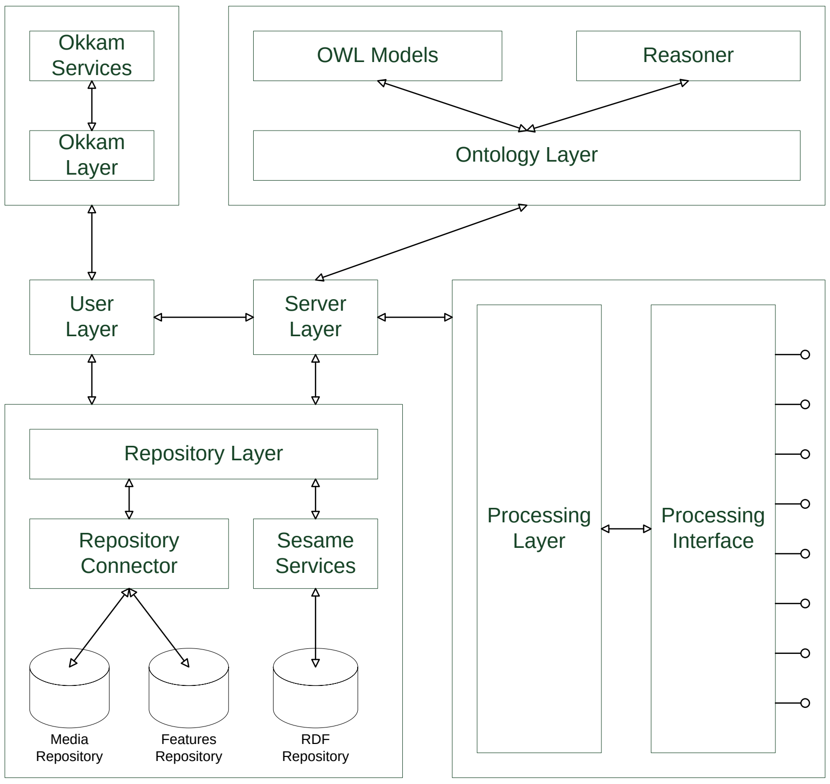


Figure 6 System architecture

In the following we point out the operation of each of these entities, and then a more detailed analysis will be performed.

- **User Layer**

  It's the system access point. The user can search content or insert it. This level should save the data in an understandable format at a global level, so it implements a function that creates an RDF file using the specific adopted syntax. These contents will be sent to the repository layer for storage. The user layer, also includes a module for creating user interfaces for domain specific MMR.

- **Server Layer**

  This Layer exposes a range of services for the User Layer. When one searches for a content of a given media type, the user layer refers to the server layer to perform the needed operations. This layer handles all the operations starting from the input of the media content. It uses Ontology layer (or features model) to derive the algorithm more suitable, it calls the processing layer for low level features elaboration and finally stores these in the repository data store (features repositories).

- **Okkam Layer**

  This level makes the entities universally identifiable. The information will be totally integrated with others placed in different contexts. A search is performed in the Okkam (see Appendix C) data store in order to establish if the media already has an Okkam Id. Then simply return the ID if it already exists, or allow the user to create a new entity. From figure 6 it can be seen that this level has only interactions with the user level. This is because the returned identifier is inserted in the RDF file created by User Layer.

- **Repository Layer**

  This layer is responsible for all the content stored in the system.

  There are three repositories:

  - *Repository of media file*

    This repository will allow direct access to the media file, if is needed to recalculate the features or just to preview the file. The content is identified using the RDF file that contains, among other things, the path of the media file. To get this property the system accesses the RDF repository files through an appropriate RDF query.

  - *Features repositories*

    This repository contains the computed features of each file. When new media content is inserted, its features will be included in this repository. As with the repository of media files, an RDF file is needed in order to access the features. Basically, the system will once again access the RDF repository; this happens whenever it is necessary to retrieve information quickly and correctly.

  - *RDF repository*

    When the user enters new media content, an RDF file is created.. contextually indexing the entities using unique identifiers derived by the Okkam Layer.

- **Ontology Layer**

  This layer is responsible to retrieve, using an ontology specified from time to time through a model, what are the steps involved for the media features elaboration. The user may wish that its requests are answered as quickly as possible, or as accurately as possible; this level will call each time the algorithm that best fits the user's wishes. Based on a set of constraints that can be chosen from one of the available (or generated) models, the list of relevant features will be returned to the server level. From this list of features that must be calculated, the server layer invokes the services exposed from Processing Layer. The result will be the computed values for the features, to be placed into the repository. This layer makes use of a basic module (Features Model) that is discussed explicitly in chap. 4.

- **Processing Layer**

  This level exposes the methods that correspond to the algorithms to be used for the features elaboration. These functions are often very similar, differing only by the precision with which elaboration is performed. The functions to be applied are as operational blocks that, starting from a set of input data, provide a set of output data. The server will only apply these functions recursively taking the data out of a block and placing them in the next entry. The functions to be used (and in what order) are dictated by the Ontology Layer. The processing layer provides only to the features elaboration.

3.3 Processing Layer

A processing algorithm can be seen as a "black box". Thus, from a set of input data, we get another set of data produced by the box; in our case: the features. The constituent elements of the processing algorithm are very important. Each algorithm in fact, whatever it is, consists of a set of basic steps. For preprocessing functions we intend all those functions that act before the features elaboration. In certain situations it may be necessary to emphasize distinctive elements of our media content, such as the signal at certain frequencies or specific time intervals, etc..; when one wants to improve the quality of content before process one must apply the special preprocessing functions. After pre-processing, it will be possible to execute the features elaboration. This phase influences the performance of the algorithm in terms of accuracy and speed, and usually consists of several sub-blocks (fig. 7). This is also true for the functions of preprocessing, in fact, it is possible to apply different functions according to a completely arbitrary flow: cascade, parallel or any combination thereof. At the end of the process, we get the features.

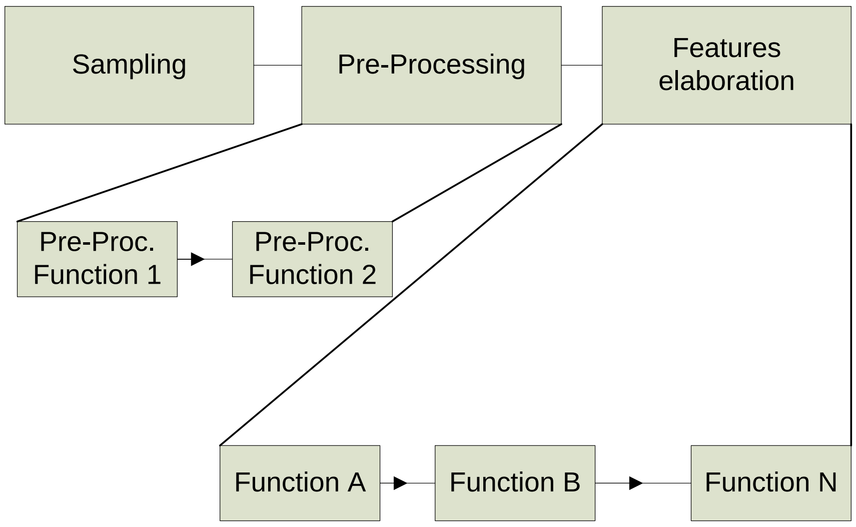


Figure 7 Processing steps

In the Processing Layer, there are operational blocks; the functions receiving a set of input data, apply it to one or more operations (dependent on the nature of the block) and then pass the result to the next block. In this system, we want to allow the dynamic creation of an algorithm thanks to the Ontology Layer. This Layer receives the user-specific information to create a feature diagram, i.e., a graph that links all the operations that we can do. From this graph, developed taking into account the constraints imposed, a list of functions is selected to be applied to the media content. This list is a set of steps, constituting the algorithm. This block can be reused in different applications altogether. Its capacity is only performing the operations that are required.

3.3.1 Using Processing Layer

The Server layer will require to the Features Model Ontology Layer to derive an algorithm and once receiving the list of functions to be applied, it will call the processing layer, which will perform the required elaborations on data sent. The features are the result of the last operation. The Server invokes a particular processing method indicating the input data, output data and the name of the method to be applied according to the following schema:

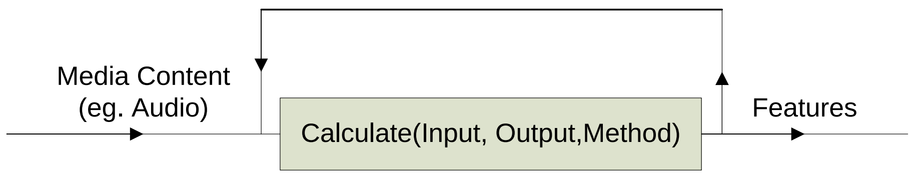


Figure 8 Processing flow

The output of a block will coincide with the input of the next. Finally, we will have in output the features. Figure 9 depicts the layer interaction.

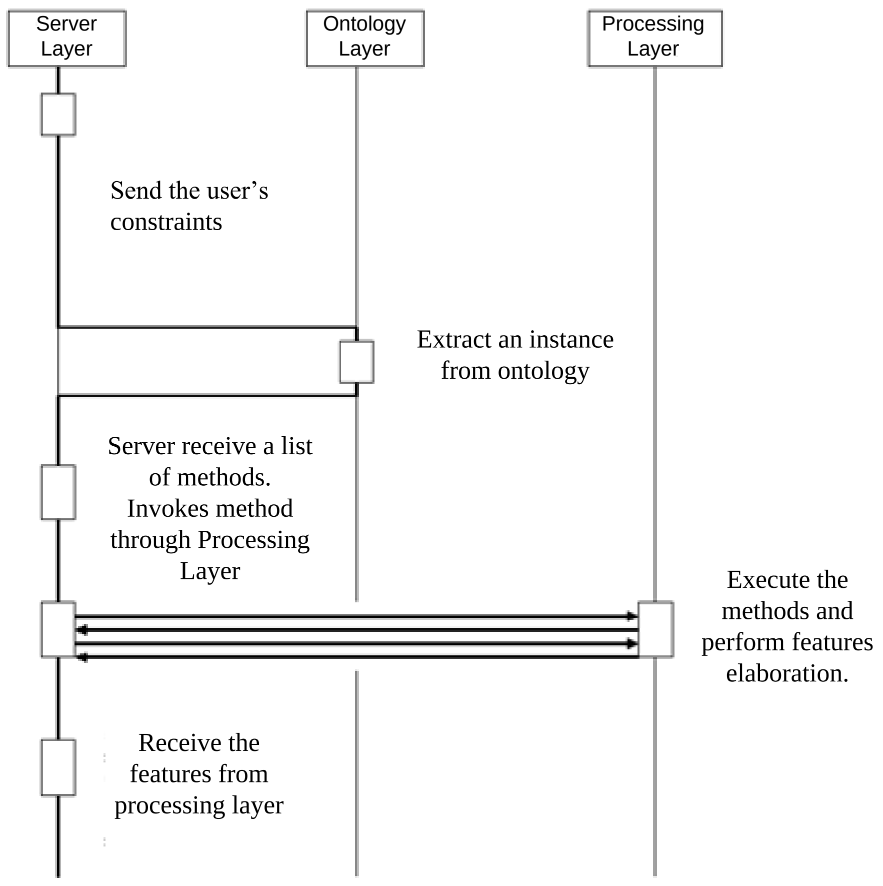

Server Layer
Ontology Layer
Processing Layer
Send the user's constraints
Extract an instance from ontology
Server receive a list of methods. Invokes method through Processing Layer
Execute the methods and perform features elaboration.
Receive the features from processing layer

Figure 9 Sequence diagram for the user's request elaboration

The ontology layer, according to the diagram created and the constraints imposed, will generate a list of different sequences of functions and, consequently, a different algorithm. This is a key feature of the system because instead of setting on a single, preferred algorithm, the user has freedom of choice and action.

3.4 Okkam Layer

In the context of the Semantic Web it is essential that each entity is identified unambiguously. It will be necessary to use special identifier, uniquely defined and widely recognized. These identifiers will be included in the RDF file. Any external source can access to repository through query using the ID provided by Okkam, to be sure to get all the available data. For information about the internal structure of Okkam see appendix C.

3.4.1 Okkam Interaction

Within the system there is a Layer implemented as a Web Service (the Okkam Layer) that calls several external APIs. The exposed methods are used by the WSDL file at:

http://api.okkam.org/okkam-core/services/WebServices?wsdl

The operations that we provide in our system are relatively simple. When the user inserts a new content, it must be ensured that the identifiers used during the storage of the media are the Okkam id. In the example of music ontology treated in chapter 5, we see that the entity for which identification was used are the musical artists, however, the same approach can be applied to the music tracks, albums or other information.

When a new content is inserted, if an identifier already exists, then it will be used, else a new entity is created and then the new obtained identifier is used. Among the different system features, there is also a set that covers the possibility to execute queries on the Okkam data store. It is sufficient to invoke a particular method:

```
MatchingCandidate[] candidate = findQuery(query)
```

The method takes as input a string containing the query and returns an array (sorted by relevance) of candidates. For each candidate it is possible to trace the identifier and a coefficient *sim* indicating the element similarity. It is assumed that, if the search is successful, *candidate* structure is not empty and also the element in the first position is the one searched. We present the structure of the query used for the music ontology system, as example:

```
QUERY { name=singerName tag=singer }
METADATA{ entityType=person matchingModule=gl };
```

The user inserts the singer name (singerName), the system insert all the other parameters, valid for any query. The check is not performed only on the basis of this query; the author could be a band and not a solist, so the most appropriate entity Type is *organization* and not *person*. Also, the matching mechanism can be changed. Consequently, the comparison is made through a combination of queries. If a positive matching exists the id of the first candidate will be selected. The returned identifier is used during storage as resource id in the RDF file, otherwise the candidate structure value is null. Clearly, it is possible that the searched artist is not present, then we create a new entity and use the identifier returned. While research is a task for which you do not need special permission (do not change anything, read-only operations), the creation of a new entity is different. In that case we are altering the Okkam data store.

Adding a new entity to the system, means performing write operations. There are several ways to create an entity, including the use of particular API designed for this task. Since we are conducting operations so-called "protected", we must maintain a certain level of security. This is achieved through authentication with a certificate downloaded on purpose. Using the API means performing the following steps:

- Set Username, Password and location of the certificate for the credentials;
- Create an ENS (Entity Name System) client;
- Set the characteristics of the entity that we want to create, this implies attributes, alternative identifiers or equivalent and external references;
- Validate the entity (check if the entity already exists). In this case
  a merging is performed (merging the new inserted information with those already existing); this will not create a new entity but will return the id of the already existing entity;
- If the validation was successful, a new entity is created. In this case the id of the new entity is returned.

However, it must be taken into account that the Okkam project is evolving, the safety policies may change, as well other settings of the project. It’s possible to move part of processing on the server side. In this way our client work even in case of any changes to the server side. This approach is based on a particular page:

https://api.okkam.org/EnsWebToolKit/oec-wizard.jspx

This page contains an interface that provides all the tools to create a new entity. At the end of the development, a message is displayed that informs us that the new entity was created, adding the id. In our system this interface is called from the application itself; to ensure that this happens we must follow some steps. The application that allows us to create a new entity is actually a stateful service. In fact we have implemented an hash table that, by session identifiers, allow us to "remember" the received requests. First a session id must be retrieved. To do this the following page must be invoked:

http://api.okkam.org/EnsWebProxy/getSession

In this way we obtain an id session. Now we call the creation wizard adding two parameters: sid and create; sid is associated with the session id previously obtained, create should be set to true. In this way the creation interface realizes that it was called in the context of another application. At this point the user interface can be accessed and, once creation has been completed, the server associates the session ID to the created entity Okkam ID. Now, to get the id okkam the following call must be executed:

http://api.okkam.org/EnsWebProxy/getEnsId

passing, as a parameter, sid (the session identifier values).

The Server returns the Okkam Id required. All steps are shown in figure 10 below.

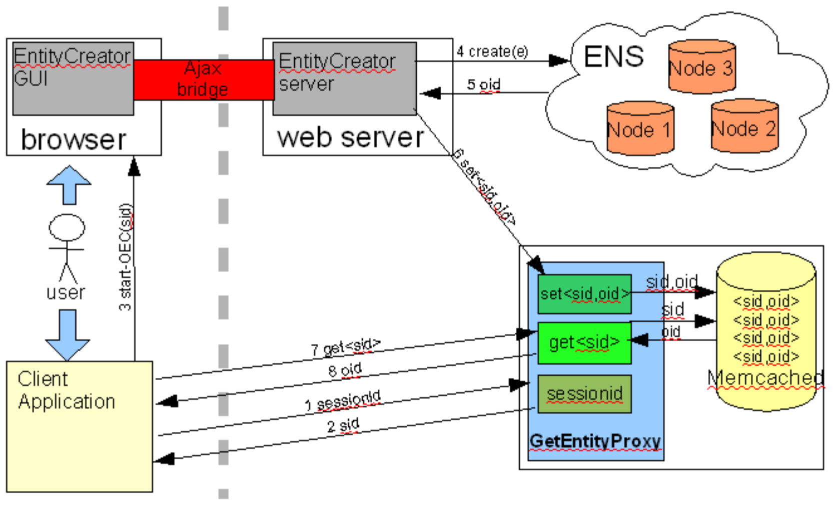


Figure 10 Okkam and ENS

In the first stage, the user indicates the author, the system, using the query mechanism discussed above to check if the artist is present. If the author is present, the first value of the structure candidates is returned. At this point the creation page is opened. Through the technique described above the client will know the Okkam id.

3.5 Repository Layer

Once obtained, the RDF file that describes the audio content must be inserted into the system store. What is needed is a "container" of RDF files: in other word a repository. The repository is in general, an element with high capacity of digital storage that can handle any changes made on the data. Inside the proposed system there is a layer dedicated exclusively to perform this task. There are many frameworks available for this purpose, developed and designed in a different way. We use the SESAME RDF repository. This is a Java framework for storing and querying RDF data type.

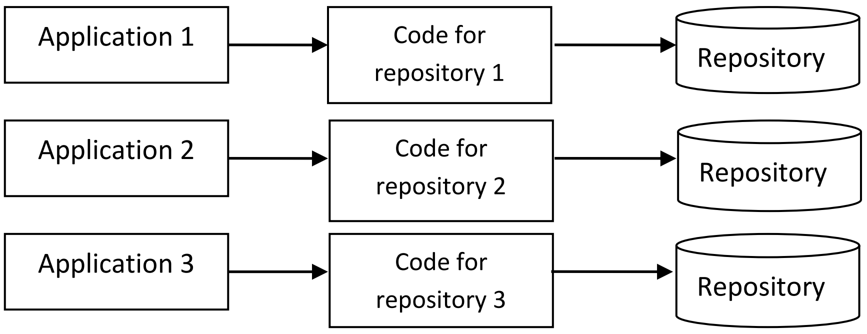


Figure 11 Interaction between some applications and repositories

A service that exposes several methods that allows interacting with different repositories was created.

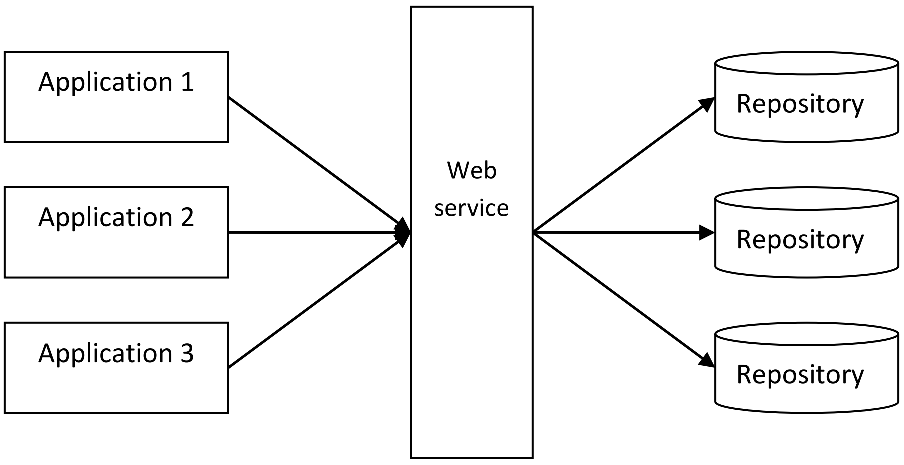


Figure 12 Interaction between some applications and repositories trough web-service

By this approach we get a service usable from any application written in any language. Furthermore, the user will not create explicit connections, to handle exceptions or write queries to interrogate the repository; the web service will take care of all these aspects.

3.6 Technical Details

Almost all layers have been implemented using the .NET framework. Some levels are applications implemented in Java, but the application uses the .NET interoperability, guaranteed through the use of web services. Several additional libraries have been used, each of which with a different and specific scope. In the example exposed in chapter 5, we will see a music retrieval system created by our architecture. The user layer receives an input audio content, makes some processing operations, and finally sends it to Server. To allow the processing, preview, and content filtering, two libraries designed for managing audio files have been developed. *AudioLab* is a set of components for fast audio processing. This library allows to capture, reproduce, display, and perform audio mix of audio track. For the audio track processing another low-level library was used: *irrKlang.* In summary, an audio content will be presented and processed through AudioLab thanks to irrKlang. After processing the audio content, the user level creates the RDF file, using a special instrument: *SemWeb.NET*. This library, written in C#, allows the reading and writing of RDF files, and has also defined mechanisms for querying. The API defined in SemWeb.NET are very simple and flexible, they also allow a greater understanding of the file structure. One could simply use the methods that allow the writing of XML files, without additional tools, but this approach does not provide a full and comprehensive view of the created file. Each element of an RDF file is a triplet: Subject Properties Object, and when an item to include in our file is created, the API defined in *SemWeb.NET* receive in input these three parameters defined as an entity. Once the RDF file has been created, it is sent to a RDF repository. A set of functions will be used to perform the query. *Sesame* is an open source Java framework for storing and querying RDF data. The language of the library is different from (C#), so the approach based on Web services was chosen. The user layer calls the exposed service, that are based on the Sesame API to store the created RDF file. The level Server will need to relate to user services that are exposed as methods for querying the repository. The approach based on Web services enables high reusability of the code, since different applications can uses the system and external services. Another advantage of this approach is the considerable simplification of operations used from the application.

Chapter 4

Features Model Layer

In this section we describe a functionality of an important section of our system the Features Model Layer. Our system uses a module for the creation and modeling of ontological models. This layer exposes a set of services to interact with ontologies created to represent the reality of interest. Note that it is not a module for modeling LLF (Low Level Features), as the name might suggest, in this case for features we intends the an important property of a concept of the domain of interest. Generally features models are used to describe common and variable properties of families of related software systems referred as SPL (Software Product Line). Every program in an SPL is identified by a unique and legal combination of features called feature configuration. There is no formal semantic for describing a feature model and no standard tool for building and validate a feature configuration. In this section we present an OWL-based approach for building and editing feature models together with an OWL-based inferential engine for creating a feature configuration and check its consistency.

4.1 Feature models

A feature is defined as "an important property of a concept" [16] where by concepts, we mean anything in the domain of interest. "Feature modeling is the activity of modelling the common and the variable properties of concepts and their interdependencies and organizing them into a coherent model referred to as a feature model.”[16]. A feature model consists of a tree diagram called feature diagram with some additional textual information, such as a semantic description. A feature diagram is made up of nodes, directed edges and edge decorations, where the root node represents the subject we want to describe, formally called concept, and the remaining nodes are its features. Features are further connected by edges to sub-features in a hierarchical structure. Formally an instance of a feature diagram is known as a concept description or a feature configuration and is defined valid if does not break feature constraints and inclusion rules The rest of this section is organized as follows. In section II-A we explain feature models through a simple example. Section II-B explains each feature diagram entity and inclusion rules for building a valid feature configuration. Section II-C introduces our formalism.

4.1.1 Feature diagram example

The example we are going to discuss has been taken from [16] and describes commonalities and differences among instances of a Car (see Fig. 13).

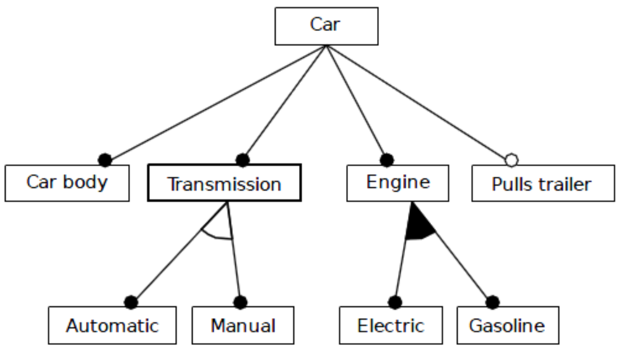


Figure 13 Feature Model example (taken from [16])

In order to represent this example, we have chosen to use Czarnecki’s notation with decorated edges, because it is the best know formalism in the literature. The concept Car is described by features Car body, Transmission, Engine and Pulls trailer. The Transmission feature is described further by the sub-features Automatic and Manual; an Engine is described by the sub-features Electric and Gasoline. Without a way to add some restrictions on the feature model this diagram also describes invalid cars like one with a transmission that is both automatic and manual. Semantic restrictions like those are provided by different edge decorations. A valid feature configuration of this example can be described by the features: Car body, Transmission, Automatic, Engine, Electric. Such a configuration do not violate model restrictions and so a car with a car body, an automatic transmission and an electric engine can be manufactured. Besides this one, such diagram allows cars that pull a trailer and hybrids cars, i.e., cars with an engine that is both electric and gasoline.

4.1.2 Feature diagram entities

Many variations to the original feature model notation FODA (Feature Oriented Reuse Method) [38] have been proposed, such as FORM (Feature Oriented Reuse Method) and FeatuRSEB [68], but none of them has been accepted as a standard. In this module we have chosen to use Czarnecki’s notation without edges decoration as a starting point and we introduced some new feature constraints. We give here a brief description of each feature type and their selection rules with Czarnecki’s notation with decorated edges. In fact, feature types and inclusion rules are the same for both two notations. Besides with this formalism we have also the occasion to explain why we preferred this one over the others. As in FODA, Czarnecki distinguishes between mandatory, alternative and optional features, but he introduces also or features. A Mandatory Feature (see Fig. 13 at features Engine or Car body) is included in a feature configuration if and only if its parent is included as well. It is represented graphically by a simple edge without decorations ending with a filled circle. An Optional Feature (see Fig. 13 at feature Pulls Engine) may be included in a feature configuration if and only if its parent is included. It is represented by a simple edge without decorations ending with an empty circle. Only one feature in a set of Alternative Features (see Fig. 13 at features Automatic and Manual) can be included in a configuration. Alternative Features are represented by edges connected by an arc. In a set of Or Features (see Fig. 13 at features Electric and Gasoline) any non-empty subset of features can be included in a configuration. Or Features are represented by edges connected by a filled arc. Besides these features, we have also Optional Alternative Features, when there is at least one optional Feature in a set of Alternative Features (see the left side of normalization at Fig. 14), and Optional Or Features, when there is at least one optional feature in a set of Or Features (see the left side of normalization at Fig. 15).

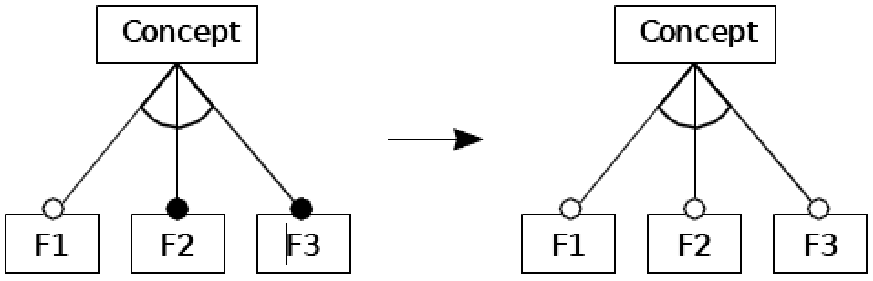


Figure 14 Optional Alternative Features normalisation

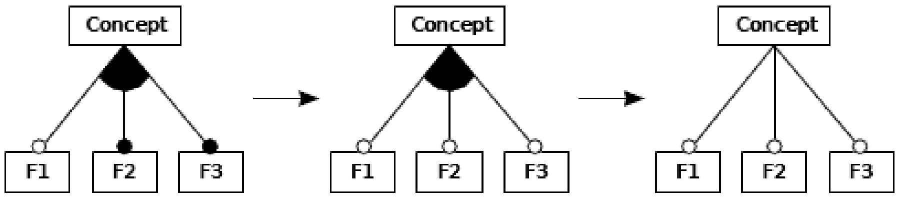


Figure 15 Optional Or Features normalisation

A feature diagram with one or more Optional Alternative Feature is normalized into a diagram with all Optional Alternative features (see Fig. 14). A feature diagram with one or more Optional Or features is normalized into a diagram with all Optional Or features which is equivalent to have all features optional (see Fig. 15). So the category of Optional Or features equivalent to the category of Optional features.

4.1.3 Feature diagram notation

In this section we introduce our new formalism with a representation of the running example (see Fig. 16).

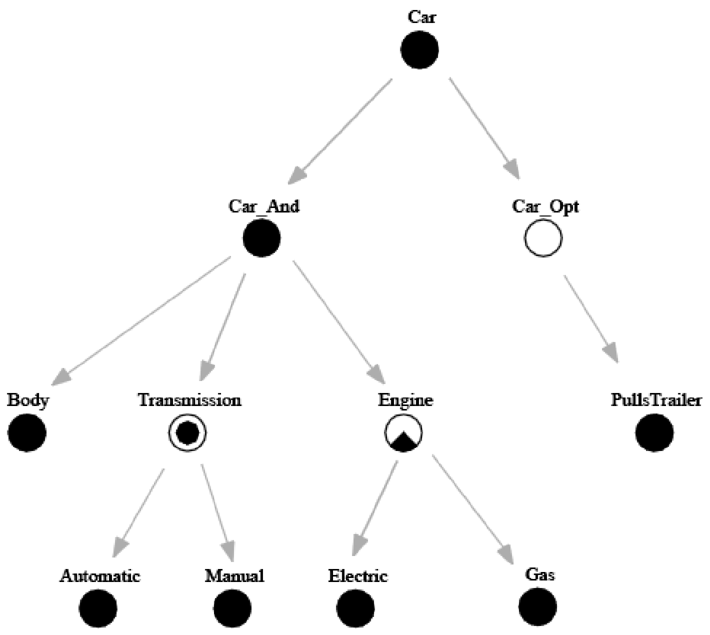


Figure 16 Car example without edge decorations

This notation comes from Czarnecki's [16] feature diagram notation without edge decorations but contains also some new features constrains. Every feature model created with the previous formalism can be converted to an equivalent feature diagram without edge decorations. We used this formalism, even if it is less concise than the other one, because of its simpler structure and a simpler analysis required. The simpler structure is due to type information not stored in a feature itself but in its parent node, so that every such node has an homogeneous set of subfeatures. As consequence of this structure, if one starts using it from the beginning, no normalisation is ever required. In this notation concepts, parent nodes of mandatory features and leaf features (features without sub-features) are represented like a filled circle (see Fig. 16 at features

Car, Car And and Body); parent nodes of optional features with an empty circle (see Fig. 16 at feature Car Opt); parent nodes of alternative features with two concentric circle, where the internal one is filled (see Fig. 16 at feature Transmission); parent nodes of optional alternative features are represented like alternative features but with the internal circle empty; parent nodes of or-features are represented with a more complex figure (see Fig. 16 at feature Engine). In this new diagram there are two new nodes Car And and Car Opt that were not there in the previous one. These nodes do not represent an entity of the world but are only used for grouping features under a common feature type. This kind of node is called feature group and like every other node with child features (also called node feature) has a particular representation depending on the type of sub-features. Selection rules with this notation are equivalent to those introduced in section II-B, but now rules on a parent node influences selection of its children. Besides a concept node and all its direct features are always selected in a feature configuration. Feature type rules are not the only restrictions that influences feature configuration construction. Constraints can exists between features in different branches of a diagram tree. Czarnecki [16] enriched the original FODA notation with two kind of feature constraints: mutual-exclusion constraints and requires constraints, renamed here respectively excludes constraints and implies constraints. Those two types of constraints are modified to be unidirectional but maintains same semantic. We introduced in our formalism two other kind of constraints avoid and default. Implies and excludes constraints are binary and unidirectional ones whereas avoid and default are unary. In a feature model A implies B means that the existence of a feature A in a feature configuration implies the existence of a feature B; whereas A excludes B means that if a feature A is included in the configuration feature B should be not included; Avoid A means that the feature A should not be in the feature configuration and default A means that the feature A should be in a configuration by default.

4.2 Feature Models in Owl

In this section we want to illustrate the OWL-based approach we developed to represent and manage feature models. OWL stands for Web Ontology Language and is the de facto standard for the semantic web. “Its expressive power and formal semantics made it usable in many other domains” [88]. It consists of three increasingly expressive sublanguages: OWL Lite, DL and Full. We use the OWL DL dialect because we want to infer a valid feature configuration using a DL reasoner. In the literature we found two main approaches for representing a feature model through an OWL ontology and checking its consistency. The first approach [83] represents features in a diagram like an OWL class and every feature relations like an object property. For example, in order to represent a car with a transmission (see and Fig. 13) you should create two OWL classes Car and Transmission and an object property hasTransmission for representing this relation. This approach does not represent features in a configuration as instances of classes (OWL individual) as intuitively one would think. OWL classes are used to simulate features in order to use TBox (terminological box or class-level) reasoning for checking consistency. This solution was justified by limitations of the OWL reasoner RACER [35] in 2005. At that time RACER was only able to detect ABox (assertional box or instance level) inconsistencies but not which classes has caused them. The second approach [88] represents feature models in a descriptive way, mapping every feature in a model or in a configuration with an instance of class Feature or one of its subclasses. This solution was proposed in 2008 with another reasoner called Pellet [71], [14]. This reasoner was already capable of both ABox and TBox reasoning and debugging, overtaking every limitation of the previous approach. We have chosen to use the latter approach because it simplify the representation of feature hierarchies and more important because it make possible to express SWRL consistency rules on OWL classes and infer a valid feature configuration through SWRL rules.

4.2.1 Feature model ontology

Every feature model or configuration is an instance of a schema defined in an OWL file. Such schema describes vocabulary, structure and type restrictions of feature models (see Fig. 17), i.e. features cannot be added as children of a diagram node.

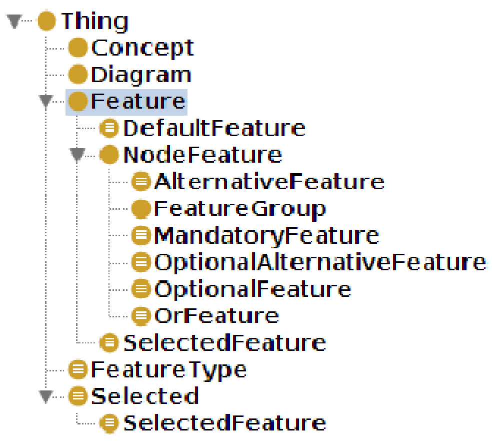


Figure 17 Feature model ontology classes

The main ontology class of this schema is Feature, which represents all kind of nodes in a feature diagram except for the root node which is a concept. In this ontology we distinguish between leaf and node feature. Because of the Open World Assumption only node features are described in our ontology (see Fig. 17 at the NodeFeature class). Every feature that is not a node feature is considered a leaf feature. Our formalism without edge decoration enables to represent edges like a normal parent/child relation without any type information through object properties hasChild and hasParent. In order to ensure that a feature diagram is a tree and not a graph we introduced some restrictions on these properties: hasParent is defined as a functional property, meaning that an individual of its domain can only have one parent; hasChild is defined as an inverse functional property, specifying that two different parents cannot share the same child node; both these properties are declared irreflexive, which avoids that an individual has itself as a child or a parent. Feature type information are stored on diagram node with the object property hasFeatureType. It is defined functional because it can connect a Feature instance with only one of the individuals of class FeatureType (see Fig. 17 at the FeatureType class): Mandatory, Optional, Alternative, Or and OptionalAlternative. The main subclass of Feature is NodeFeature, which represents a feature with at least one subfeature. A NodeFeature instance can be further classified according to feature types in: MandatoryFeature, OptionalFeature, AlternativeFeature, OrFeature and OptionalAlternativeFeature. This new formalism introduces a new feature diagram entity called FeatureGroup (see Fig. 17 at the FeatureGroup class), this does not represents a real diagram entity but only a way to group features under a common feature type. A feature group inherits from the super class NodeFeature and must have at least a child feature.

Features can be classified in SelectedFeature and Default-Feature respectively, according to the properties isSelected and default. To the first group belong features that have been selected by the user to be in a feature configuration. To the second group belong features that the user want to be selected automatically by the inferential engine during the feature diagram construction. Besides the feature class, we introduce two other classes: Concept and Diagram. A concept must contain at least a child node of type FeatureGroup, while a diagram must contain at least a concept. In fact, a feature model can be described by multiple diagrams related by inter-relation feature constraints. In order to support feature constraints, we introduced the following properties: implies, excludes, default and avoid. Implies and excludes are object properties and they connect two instances of the Feature class. They are declared irreflexive, so that a feature cannot imply or exclude itself. Avoid and default are boolean data properties and they do apply only to feature instances. Both these properties are declared functional, so that they cannot be declared twice on the same feature instance. Avoid property is used to define which feature we want to deselect from a feature model. It differs from setting the property is Selected to false, because it can be used also during model construction. Default properties are used to define a feature that will be automatically selected by the inferential engine. This property is very useful for or, alternatives and optional alternative features in order to select automatically one of the sub-features. It is useless upon mandatory features where every feature is selected anyway. We provide also two other object properties: next and previous, which enable sorting features in a diagram. They are defined irreflexive so that a feature cannot be in such relation with itself.

4.3 Feature Model Ontology Framework

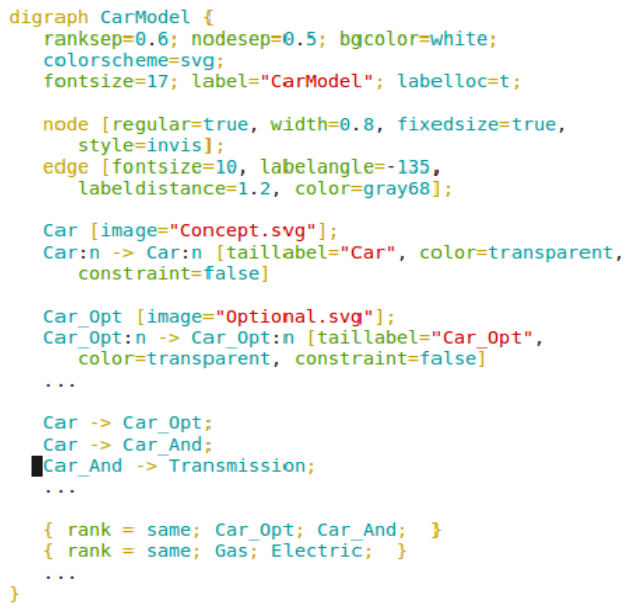
Our feature model framework consists of five modules:
ModelManager, ModelBuilder, InconsistencyChecker, SelectionEngine and SVGDiagramBuilder. ModelManager is the main module of the entire application and it is used by every other module in order to load and save local or remote feature models. This module allows also to obtain an OWL DL reasoner used for building the inferential engine or to serialise a feature model to a string. ModelBuilder is used for building a new feature ontology model or editing an existing one. Some common allowed operations are:

- creating/deleting diagram nodes, i.e. diagram, concept, leaf node, feature group.
- adding/removing a parent/child relation between two nodes, i.e. add a concept to a diagram or remove a leaf feature to a node feature.
- setting/changing the feature type of a feature node.
- adding/removing feature constraints between two features, i.e. implies, excludes, avoid or default constraints.
- selecting/deselecting a feature in a feature configuration, i.e. set the data property isSelected to true/false.

Every operation is followed by a inconsistency check, in order to detect model or OWL conflicts, i.e., adding to a diagram a feature instead of a concept or adding an irreflexive feature constraint between a feature and itself. Such conflicts are induced by OWL restrictions in the feature model schema. If a conflict is detected, an exception is launched with the OWL individual, which cause the inconsistency, and the SWRL (Semantic Web Rule Language) rule violated. Such exceptions are raised also for additional user defined inconsistency (see InconsistencyChecker) and selection rules inconsistency (see Section V). The InconsistencyChecker module is used to check additional SWRL inconsistencies rules on a feature model. SWRL rules files can be loaded/removed dynamically at runtime and inconsistency is checked through a OWL DL reasoner, which you can get with the ModelManager module.

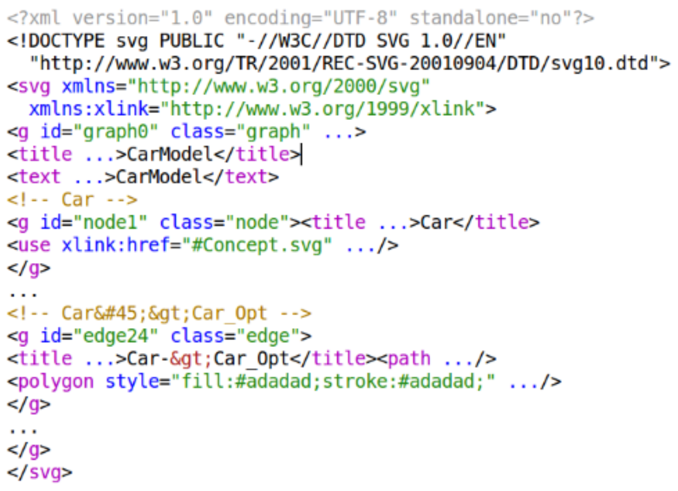
4.3.1 SVGDiagramBuilder

This module is optional and it is used only to get a graphic representation of a feature diagram. It takes as input the URI of a feature diagram within an OWL ontology file and produces an SVG vectorial image. Building such graphic representation requires three steps: loading the feature model to which the diagram belongs; building with the language Graphviz/DOT [33] a textual representation of the diagram (see Fig. 18); compiling the DOT representation into an SVG image (see Fig. 19 for the textual representation and Fig. 16 for the graphical representation).



```
digraph CarModel {
   ranksep=0.6; nodesep=0.5; bgcolor=white;
   colorscheme=svg;
   fontsize=17; label="CarModel"; labelloc=t;

   node [regular=true, width=0.8, fixedsize=true,
      style=invis];
   edge [fontsize=10, labelangle=-135,
      labeldistance=1.2, color=gray68];

   Car [image="Concept.svg"];
   Car:n -> Car:n [taillabel="Car", color=transparent,
      constraint=false]

   Car_Opt [image="Optional.svg"];
   Car_Opt:n -> Car_Opt:n [taillabel="Car_Opt",
      color=transparent, constraint=false]
   ...

   Car -> Car_Opt;
   Car -> Car_And;
   Car_And -> Transmission;
   ...

   { rank = same; Car_Opt; Car_And;  }
   { rank = same; Gas; Electric;  }
   ...
}
```

Figure 18 DOT representation of a feature diagram



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<g id="graph0" class="graph" ...>
<title ...>CarModel</title>
<text ...>CarModel</text>
<!-- Car -->
<g id="node1" class="node"><title ...>Car</title>
<use xlink:href="#Concept.svg" .../>
</g>
...
<!-- Car&#45;&gt;Car_Opt -->
<g id="edge24" class="edge">
<title ...>Car-&gt;Car_Opt</title><path .../>
<polygon style="fill:#adadad;stroke:#adadad;" .../>
</g>
...
</g>
</svg>
```

Figure 19 SVG representation of a feature diagram

The first step is performed by the module ModelManager, and it requires the URI of the OWL file containing the feature diagram. Building a DOT representation of the diagram implies mapping each node in feature diagram into a node with a label, representing the name of the feature, and an image, representing the type of node. In a feature configuration, two different colours, i.e. grey and black, represent deselected/selected features or concepts (see Fig. 20, 21, 22 and 23). The last step is performed with the command dot-Tsvg. One can use this command specifying an input DOT file and an output SVG file, but we preferred not to create intermediate DOT files. Thus we use this command writing our DOT diagram to the standard input and getting the SVG representation from the standard output. SVG images are vectorial images serialised in an XML dialect. We have chosen to create SVG images instead of raster images for two main reason. First of all, vector graphics allow scaling images indefinitely without degrading quality, so that big SVG feature diagrams can be displayed in small screen piece by piece and labels yet be readable. The second reason is that vector graphics represents images with geometrical primitives such as points, lines, curves, and shapes or polygon(s) in a textual format. This representation allows editing a vector image through its textual representation. In particular you can change an object colour, shape or position in an SVG image with an XSLT transformation.

4.4 Selectionengine

This is the main module of the application and it consists of three parts: a general algorithm, an OWL DL reasoner, which is available from within the ModelManager module, and some SWRL rules. The SelectionEngine has the objective of creating automatically a valid feature configuration from some users defined features. In order to be defined valid, such feature configuration should not violates SWRL selection rules and OWL model. A feature configuration is created selecting/deselecting some features in a feature model, thus model consistency is verified during the feature model construction.

4.4.1 Selection Algorithm

The algorithm used by the SelectionEngine requires as input an inconsistency free feature model optionally with some feature constraints. In fact, every model inconsistency does not allow the OWL reasoner to be used. This algorithm is made of the following steps:

1) The user requires to select/deselect a feature or a concept (see Fig. 20 and 22) from the feature model.
2) The SelectionEngine selects/deselects the required node (see Fig. 21 and 23) according to the SWRL rules, verifying that it does not contain inconsistencies caused by selection rules or feature constraints (i.e. more than a feature selected in a set of alternative features or the user attempts to select a feature with an avoid constraint)).
3) Steps 1 and 2 are repeated until every features/concepts required by the user are selected/deselected or an inconsistency is detected. In this latter case, the selection engine cannot go further and it launches an exception. Such exception describes the OWL individual that has generated the inconsistency, and the SWRL rule not verified. The feature model has to be modified by the user in order to solve all the conflicts.
4) The user asks to end the selection procedure.

5) The SelectionEngine selects all the default features not already selected, removes all the features that do not have their concept selected and calculates a set of selected features.
6) The user can get only the selected leaf features, all the selected features (leaf + node features) or all the selected entities (leaf feature + node feature + concepts). This last option can be used to obtain an SVG diagram from a feature configuration.

4.4.2 SWRL rules

An SWRL (Semantic Web Rule Language) [57] rule is a rule that has an antecedent part defining a condition to check and a consequent part that declares a classification or a property to set upon individuals of the antecedent part. SWRL rules are used through the entire module in order to calculate derived OWL properties or for checking OWL model consistency. Here we describe the rules we used to create a valid feature configuration and to check its consistency. All the SWRL rules can be divided in two groups, selection rules and consistency rules. To the first group belong all the rules that implement feature type selection restrictions, i.e. no more than a selected feature in a set of alternative features or a feature with an avoid constraint should not be selected. To the second group belong rules that check for model consistency, i.e. a mandatory feature should be selected. Here we give an ordered list of SWRL rules and describe their meaning. Rules 1 to 9 are selection rules whereas rules 10 to 14 are consistency rules.

1) Concept(?y), SelectedFeature(?x), hasParent(?x, ?y)→ Selected(?y)
2) Feature(?y), SelectedFeature(?x), hasParent(?x, ?y) → Selected(?y)

3) Concept(?x), MandatoryFeature(?y), Selected(?x), hasChild(?x, ?y) → Selected(?y)
4) Feature(?y), MandatoryFeature(?x), SelectedFeature(? x), hasChild(?x, ?y) → Selected(?y)
5) DefaultFeature(?y), OrFeature(?x), SelectedFeature(?x), hasChild(?x, ?y) → Selected(?y)
6) AlternativeFeature(?x), DefaultFeature(?y), Selected-Feature(?x), hasChild(?x, ?y) → Selected(?y)
7) SelectedFeature(?x), implies(?x, ?y) → SelectedFeature(?y)
8) SelectedFeature(?x), excludes(?x, ?y) → isSelected(?y, false)
9) avoid(?x, true) → isSelected(?x, false)
10) Feature(?y), MandatoryFeature(?x), SelectedFeature(?x), hasChild(?x,?y), isSelected(?y, false) →Nothing(?y)
11) Concept(?x), isSelected(?x, false) → Nothing(?x)
12) AlternativeFeature(?x), SelectedFeature(?x), SelectedFeature(?y), SelectedFeature(?z), hasChild(?x,?y), hasChild(?x,?z), DifferentFrom(?y, ?z) → Nothing(?x)
13) OptionalAlternativeFeature(?x), SelectedFeature(?x),SelectedFeature(?y), SelectedFeature(?z), hasChild(?x,?y), hasChild(?x,?z), DifferentFrom(?y, ?z) →Nothing(?x)
14) Feature(?x), MandatoryFeature(?y), hasParent(?x, ?y) avoid(?x, true) → Nothing(?x)

In the previous list, rules 1 to 9 are selection rules whereas rules 10 to 14 are consistency rules. Rules 1 and 2 declare that if a feature X is selected its parent node (a feature or a concept) should be selected as well. The effect of this rule is that a concept in a feature configuration must be always selected. Thus, for example, if a leaf feature is selected (see Fig. 20) its parent node will be selected, as well as every node till the concept node (see Fig. 21 at nodes Engine, Car And and Car). During these steps every rule in this list can be triggered selecting other features (see Fig. 21 at Body feature selected as the result of the rule 4).

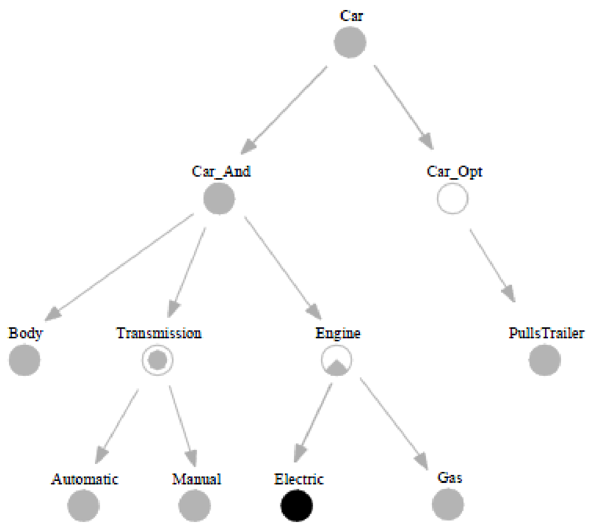


Figure 20 Electric feature selected by user

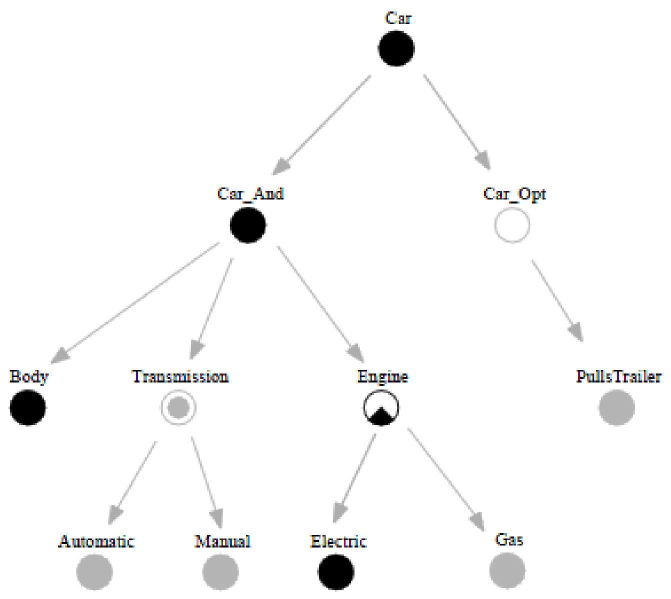


Figure 21 Inferred features by inferential engine

Rule 3 says that if a concept is selected every child feature of type mandatory should be selected as well. The user during the selection can select only the concept. Thus every mandatory feature group will be selected and thanks to the rule 4 also their child features (see Fig. 22 and 23). Rule 4 declares that if a mandatory feature is selected, every child feature should be selected (see Fig. 21 and 23 at Car And and Body nodes).

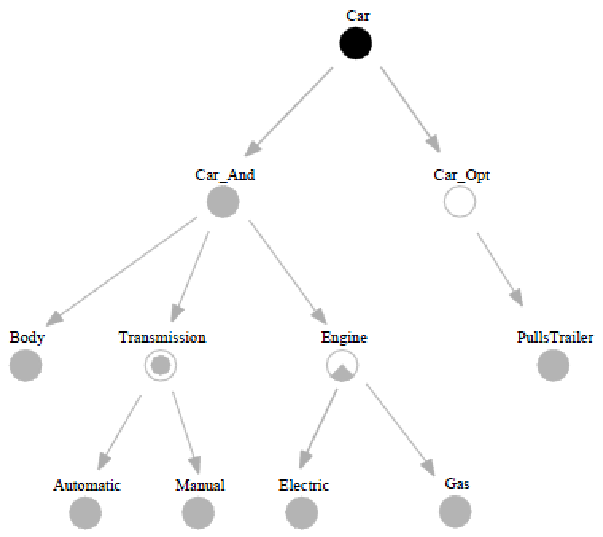


Figure 22 Concept node selected

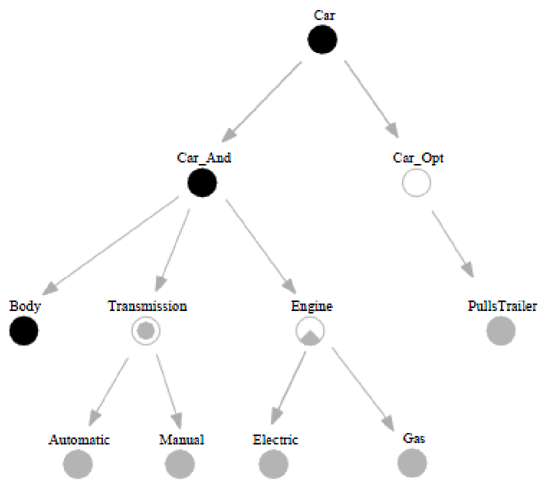


Figure 23 Diagram after rules application

Rules 5 and 6 say that if a node X is selected and it has a child feature Y of type Or or Alternative with a subfeature Z, which has a default constraint, that feature Z should be selected. Rules 7 and 8 are used to select/deselect a feature according to the feature constraints implies and excludes. If a feature X is selected and it has an implies/excludes constraint with a feature Y, Y should be selected/deselected. Rule 9 says that an avoid constraint on a feature implies that feature to be deselected. Rule 10 says that if a mandatory features is selected and at least one of its child feature is not selected an inconsistency is detected. Rule 11 says that a concept cannot be deselected otherwise an inconsistency is detected. Rules 12 and 13 state that no more that a feature can be selected in set of Alternative/OptionalAlternative features. Rule 14 says that if an avoid constraint is defined on a feature X and this feature has a mandatory feature parent an inconsistency is detected.

4.5 Implementation

In this module we created a Java framework for creating and editing feature models using an OWL ontology. We used the latest version of OWL (i.e., OWL 2) because it affords a better expressiveness through some new properties restrictions like irreflexive and asymmetric. For creating, parsing and serialising an OWL ontology we used the library OWLAPI, a Java implementation of an OWL/XML parser. This library supplies also Reasoner interfaces for working with reasoners such as FaCT++, HermiT, Pellet [71], [14] and Racer [35]. Pellet is an open source Java implementation of an OWL DL reasoner and it is capable of both ABox (instance level) and TBox (class level) reasoning and debugging. This module uses three OWL ontologies: fm-schema.owl, fmrules. owl and fm-select.owl. We have chosen to use three different OWL ontologies to make this subsytem modular and allow any optional OWL ontology to be replaced dynamically. The fm-schema ontology contains the description of the feature model ontology schema and it is required whenever a feature model or a configuration has to be created or modified. The fm-rules ontology is used by the InconsistencyChecker for additional inconsistency rules. The fm-select ontology is used to store selection rules for the SelectionEngine module. Both fm-rules and fm-select can be modified dynamically at runtime to support new SWRL rules. Every ontology in this module has been designed with Protege 4.1 alpha [61]. The developed framework is distributed as a Java library in a single jar file except for the SVGDiagramBuilder that is in another jar. In fact this module is optional and in its current implementation uses the command line program dot [33].

Chapter 5

Case study of MMR composition with the framework: a music retrieval system

The proposed MMR composition framework can be used with any type of media. An example of use has been performed for music retrieval, so the use cases presented in this section address the interaction with this type of data (audio file). A features model has been designed to define all the steps basically involved in music audio processing. An interface related to defined model has been developed using some audio library. This is only an example of use of this system, however it's possible to use images of audio or video, simply adding the appropriate features and create a suitable features model. The defined operations are insert and search. By inserting the user enriches the data store system and, through search, the user searches for a music content. Logically the two operations may be considered very different, but they are not different at the operational level. In both cases, we need to calculate the features, and while in the insert they will be added to the data store, in the case of search they will be compared with the other stored in the system. We will examine first the case of a user searching for a song, then the insertion of a track in the data store system.

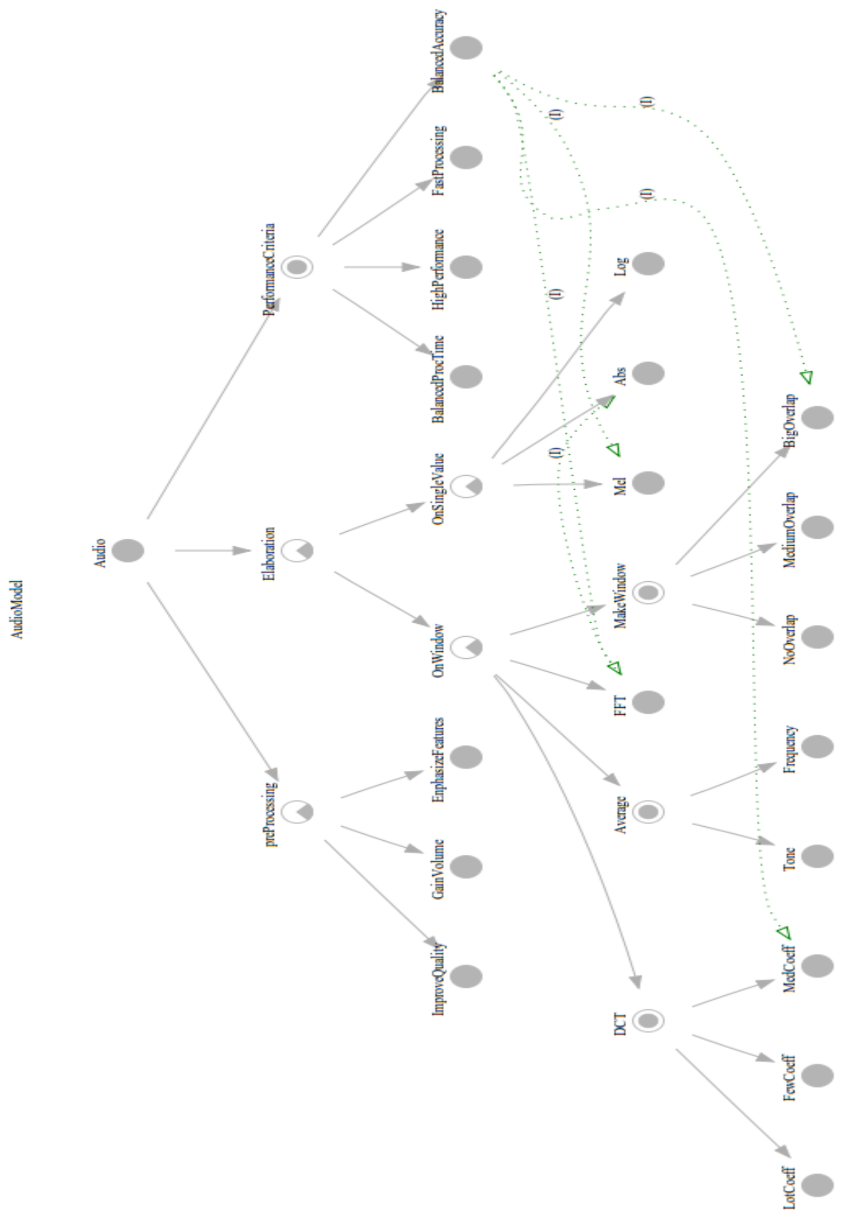


Figure 24 Ontology designed for music processing

5.1 The adopted features model

The Ontology layer extracts features as an instance from the feature diagram defined, starting from the constraints imposed by the user interface. As can be seen from figure 24 there are different types of nodes. The root defines the main concept, i.e. Audio. The main concept's children have the task of grouping different types of features, such as pre-processing and processing (see chapter on Processing Layer). There is also another group called the performance criteria, it represents the type of processing required by the user. Through this group, the user can specify the constraints to obtain a custom processing. In accordance with the input parameters and other factors, the user can choose an elaboration oriented to accuracy (High Performance), execution time (Fast Processing) or a middle ground between the two (Balanced accuracy and processing time). Moreover, the Performance Criteria group is an alternative type, this implies that exactly one of the features below must be selected. Pre-processing and Elaboration are OrFeatures. With OrFeatures one can select none, one or more features below. The leaf node represents the features to extract, the algorithm used for a specific low level features processing. We must also enter some constraints that, in this context, are expressed through the directed edges. In the reported example "Balanced Accuracy implies “Big Overlap”, “mediumCoeff” and other features to extract; in the previous figure we can see an arrow from node “Balanced Accuracy” to all the features included. The letter ‘I’ denotes the inclusion relationship. Clearly, it would have beeen possible to have other types of constraints: exclusions, default and Avoid, however in this context they were not needed. The presented diagram here is simple: the operation executable on a music file are hundreds, grouped by many types of features. In this work, we use a few features, since the focus has been placed in creation of the entire architecture. In the example, the selection Engine will start by first selecting the "Balanced Accuracy", this will include several features, each of which will be returned according to the group. For example, from the group "On Window" the valid features will be considered "FFT" and one between "MakeWindow" and "DCT". Which one to choose depends on the constraints imposed by the performance criteria selected. The features “FFT” include “Abs”, then “Abs” will be considered in the returned configuration. Ultimately, applying all the rules displayed in the graph, the sequence of features is: BigOverlap, FFT, Abs, Mel and MedCoeff.

5.2 Use Case: Search

At this stage the user logs on to the system through an interface that allows different settings. The interface created for music retrieval includes a set of functionality that allows to customize a search task and to define the system performance. The source of our audio file can be either memory or the microphone. Accordingly with the specific selection will be chosen a more or less fast function (and therefore more or less accurate), that will compose the algorithm for audio processing. At this point the user starts the search sending the files and the criteria to the server layer.

- **Server Layer: Receiving data**

  At this point the system proceeds to calculate the features from the selected audio file, using the user’s constraint. The server will collect the data sent from the user level, then it will change a lot in order to interact with the Ontology Layer. The selection of the features set is made from an instance of the features model, calculated from the server layer through the invocation of the methods exposed by the Ontology Layer.

- **Server Layer: received algorithm**

  Upon receiving the list from the Ontology Layer, the server layer may calculate the features. The low level features algorithms are physically present on the processing layer that exposes a methods for data elaboration. The Server Layer invokes the processing layer methods.

- **Processing Layer: features elaboration**

  The processing layer receives the server requests, invokes methods and sends the results. This operation is done through a dynamic libraries linking.

- **Server Layer: Matching**

  At this point the Server will be able to compare the features calculated with the features contained in the data store. The comparison is usually based on a given distance, and compares the different values of this distance. We can use different distances, such as the Euclidean or the Mahalanobis ones. It is also possible to add new algorithms for matching.

The Server Layer, before comparing the features, accesses the repository of RDF files to know what algorithm has been used for the stored features. If the features calculated correspond to the features stored in repository, they can be compared. If the features do not correspond with the features stored in repository, then the server Layer has to elaborate a request to the processing layer to perform a features extraction . The features obtained will be added to repository.

5.3 Use Case: Insert

Once the musical track has been uploaded, the system will be able to automatically acquire basic information regarding the audio content, such as duration, resolution and sampling frequency. Now the user adds other information such as name, song title and album. A newly added content must have a unique, globally recognized identifier. For this reason, after inserting the files and information, the RDF file will be created. During this phase, we will invoke the methods exposed by the Okkam Layer. Once created, the RDF file will be included in the Sesame repository.

5.4 Music Ontology

A description of content has to follow certain rules: for example, the type and the correlations with internal and external content must be defined unambiguously. There is therefore a need for a common vocabulary that can identify and link together all the resources for audio content, i.e, a need for an ontology. We use and extended music ontology. The Music Ontology is an attempt to provide a vocabulary for linking wide range music-related information, and to provide a democratic mechanism for doing so. Anybody can publish Music Ontology data and link it with existing data, in order to help create a music-related web of data. For example, John Doe may publish some information about a performance he saw last night (like the fact that he was there, and a review). Mary Doe may publish the fact that she attended the same performance, that she recorded it using her cell-phone, and that the corresponding item is available in her podcast. The Music Ontology provides a vocabulary to express information ranging from this example to the following:

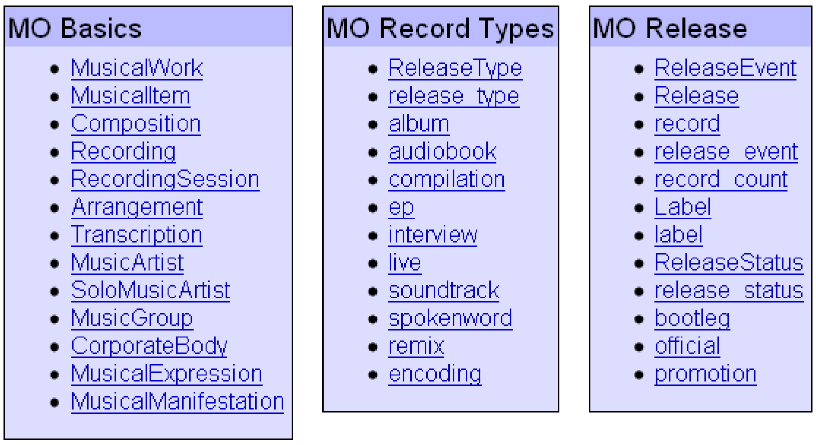
> In this performance a particular arrangement of the Quintet by Franz Schubert was interpreted.
>
> This work was performed ten times, but only two of these performances were recorded.
>
> Ten takes of this particular track have been recorded, each of which with a particular microphone location.
>
> "Come as You Are" by Nirvana was released on a single and the "Nevermind" album.
>
> During this gig, the band played ten songs. During the last one (a cover of "Eight days a week"), the drummer from the support band joined them to play with them.

The Music Ontology is divided in three levels of expressiveness - from the simplest one to the more complex one. Everything is clustered around the following categories:

- Level 1: aims at providing a vocabulary for simple editorial information (tracks/artists/releases, etc.)

- Level 2: aims at providing a vocabulary for expressing the music creation workflow (composition, arrangement, performance, recording, etc.)
- Level 3: aims at providing a vocabulary for complex event decomposition, to express, for example, what happened during a particular performance, what is the melody line of a particular work, etc.

The Music Ontology definitions presented here are written using a computer language (RDF/OWL) that makes it easy for software to process some basic facts about the terms in the Music Ontology, and consequently about the things described in Music Ontology documents. A Music Ontology document, unlike a traditional Web page, can be combined with other Music Ontology documents to create a unified database of information.


MO Basics

- MusicalWork
- MusicalItem
- Composition
- Recording
- RecordingSession
- Arrangement
- Transcription
- MusicArtist
- SoloMusicArtist
- MusicGroup
- CorporateBody
- MusicalExpression
- MusicalManifestation

MO Record Types

- ReleaseType
- release_type
- album
- audiobook
- compilation
- ep
- interview
- live
- soundtrack
- spokenword
- remix
- encoding

MO Release

- ReleaseEvent
- Release
- record
- release_event
- record_count
- Label
- label
- ReleaseStatus
- release_status
- bootleg
- official
- promotion

This specification serves as the Music Ontology "namespace document". As such it describes the Music Ontology and the terms (RDF classes and properties) that constitute it, so that Semantic Web applications can use those terms in a variety of RDF-compatible document formats and applications. This document presents the Music Ontology as a Semantic Web vocabulary or Ontology. The Music Ontology is straightforward, pragmatic and designed to allow simultaneous deployment and extension, and is therefore intended for widescale use. The Music Ontology is identified by the namespace URI 'http://purl.org/ontology/mo/'. Revisions and extensions of Music Ontology are conducted through edits to the namespace document, which by convention is published in the Web at the namespace URI. The properties and types defined here provide some basic concepts for use in Music Ontology descriptions. Other vocabularies (e.g. the Dublin Core metadata elements for simple bibliographic description, FOAF, etc.) can also be mixed in with the Music Ontology terms, as can local extensions. The Music Ontology is designed to be extended, and modules may be added at a later date.

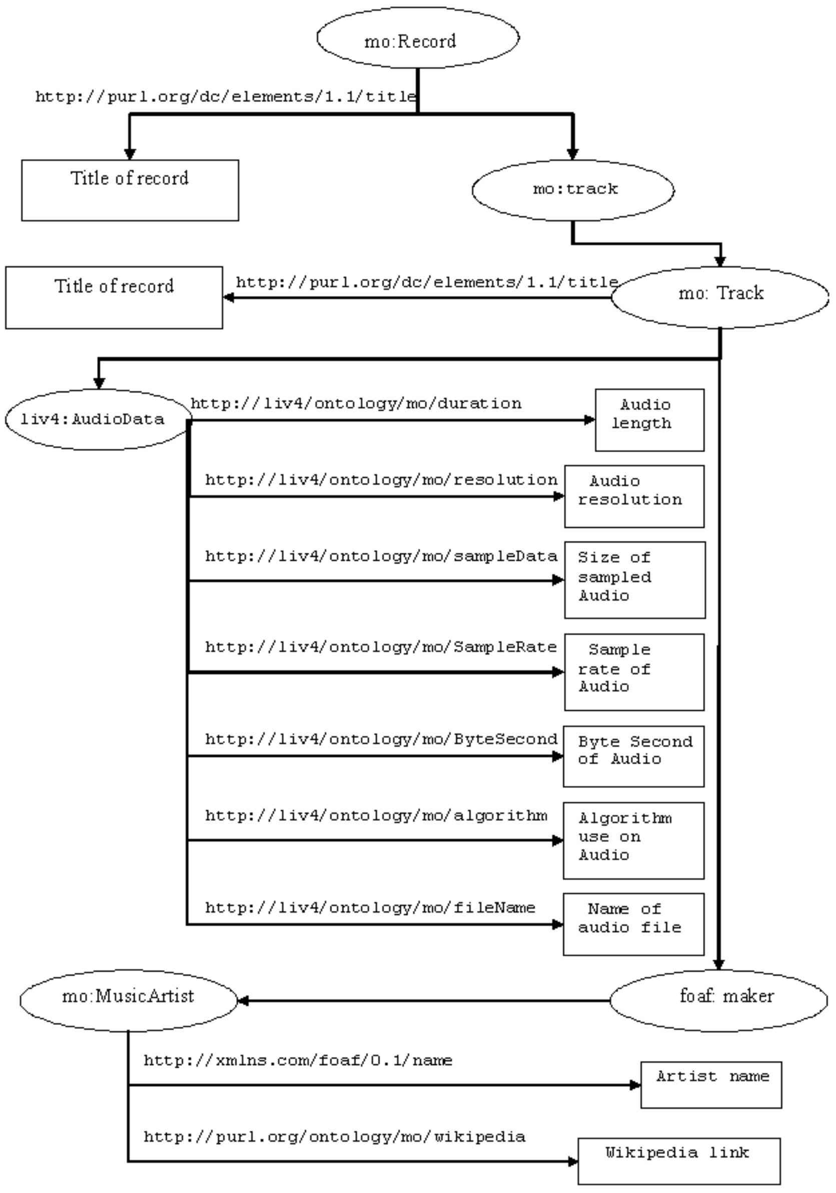

mo:Record
http://purl.org/dc/elements/1.1/title
Title of record
mo:track
Title of record
http://purl.org/dc/elements/1.1/title
mo: Track
liv4:AudioData
http://liv4/ontology/mo/duration
Audio length
http://liv4/ontology/mo/resolution
Audio resolution
http://liv4/ontology/mo/sampleData
Size of sampled Audio
http://liv4/ontology/mo/SampleRate
Sample rate of Audio
http://liv4/ontology/mo/ByteSecond
Byte Second of Audio
http://liv4/ontology/mo/algorithm
Algorithm use on Audio
http://liv4/ontology/mo/fileName
Name of audio file
mo:MusicArtist
foaf: maker
http://xmlns.com/foaf/0.1/name
Artist name
http://purl.org/ontology/mo/wikipedia
Wikipedia link

Figure 25 Extended music ontology

Chapter 6

Interface Autocomposition

The architecture of the proposed system has many advantages, but to be usable, it must be equipped with appropriate interfaces to ensure optimal interaction.

In design phase, we needed to develop interfaces for:

- The creation and integration of ontology in the system.
- The creation of input interfaces that allow expressing constraints based on the ontological model that the user wants to integrate.

As mentioned in the beginning, we recognize two types of users of the system:

- A first category comprises expert users able to model their ontology and domain of interest.
- A second category comprises people who use the functionality provided by the system to get results without making the integration of new elements.

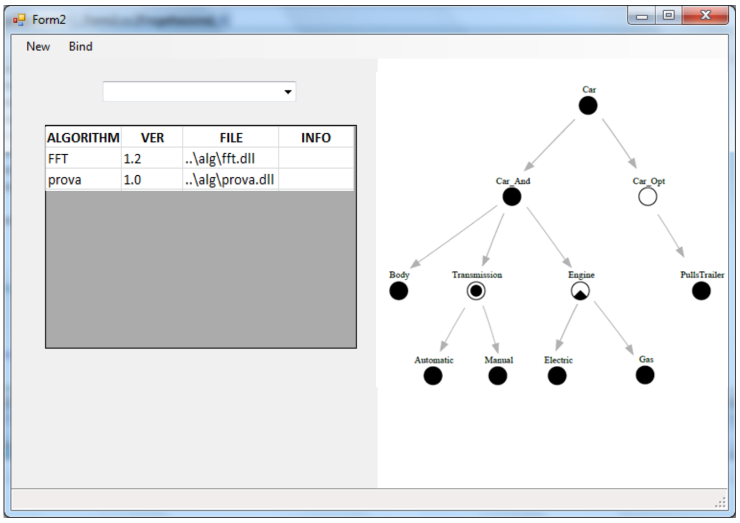
Let's see in detail how these two types of users can interact with the system.

5.2 Interface for Expert user

The expert user can use the system to model an MMR system based on the application domain. Our system allows creating a model of MMR and its interface. The processing layer will be able to adapt the sequence of steps to perform by referring to the interface choosen by the user and to the imposed constraints. The user can use the system to design a model and to design the interface for its system. We must ensure the follows functionality to design a MMR system and its related interface:

- Design suitably the related ontology.
- If the ontology already exists, specify for extensions.
- Develop the required algorithms and add them to the processing layer (if they are not already included).
- Design the GUI for user queries.

First step is modeling the domain application using an ontology in OWL. The ontology will have to explain all the possible processing sequences that the model allows. The ontology must be validated by a reasoner to check the absence of ambiguity in the constraints. In the ontological model (that represents the application domain modeling) in addition to the possible sequence of steps, the algorithms used by the model created must be specifed. A first interface allows loading the ontological model and to associate to it algorithms for processing features (Fig. 26).



ALGORITHM	VER	FILE	INFO
FFT	1.2	..\alg\fft.dll	
prova	1.0	..\alg\prova.dll	

Figure 26 Interface for associating algorithms to ontological model by drag & drop

However, before the association we must create the algorithms for processing in dll. The dll files, represents the set of algorithms stored in our system. The files are stored in a repository and are used from processing layer to perform the elaboration required. The next step will be the interface for the retrieval used a wizard procedure. A wizard procedure, assist the user for the creation of the interface for a specific domain.

6.1.1 Wizard Interface

During this phase one of the most important things is the possibility to create links that allows the formulation of the search criteria. How is it possible to associate the constraints with the wizard interface? The problem is solved by mapping properly the constraints selected from the ontological model creator.

The model should be designed in order to have the option branches where the choice is dependent from the constraints that the reasoner will accept. The reasoner returns as output an instance of the ontology that represents the sequence of steps to get the retrieval according with the imposed constraints. During the creation of wizard interface, we need a tool that allows us to map the constraints that the ontological model proposes onto interface objects.

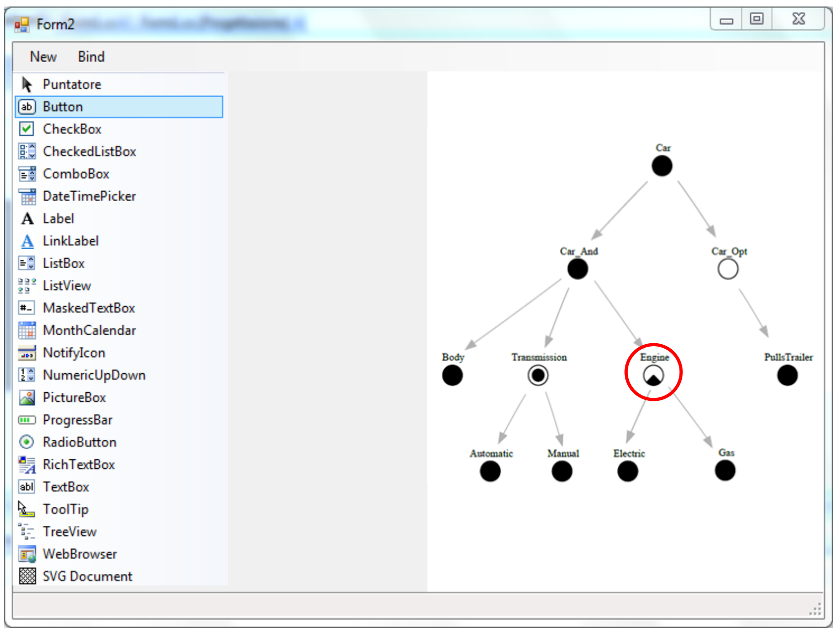


Figure 27 Binding between model OWL constraint and interface object

In summary, through the user interface, the expert can:

- Upload a model described in OWL.
- Bind the low level features processing algorithm with the OWL model.
- Create an interface for the proposed model by:

- Adding graphical objects.
- Mapping the objects with possible constraints presents in the ontological model.

6.2 Interface for simple user

The created interface, will enable searching the MMR system through query by example. The user chooses from a window the type of media to search and type of task related to that media. In other words, the user chooses among the models that have been uploaded by the expert. Then a dedicated interface allows the users to perform the search.

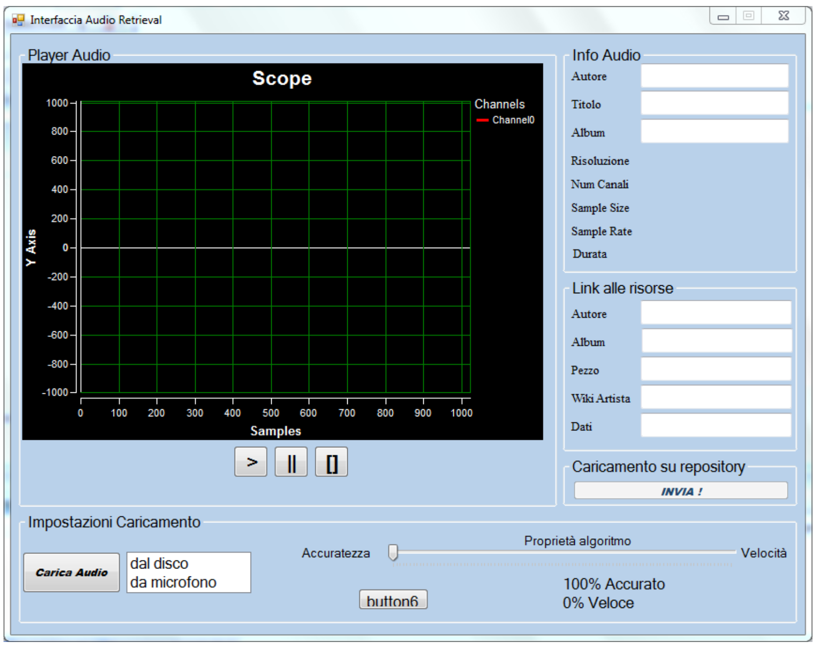


Figure 28 Example of interface created for music retrieval

Chapter 7

Conclusion and Future Work

This thesis has demonstrated how multimedia retrieval can be performed efficiently and adaptively according to the domain's features and users requirements. In detail, a novel architecture for general purpose automatic multimedia retrieval, based on some key technologies, i.e. ontology, features modeling and interface auto-composition, was devised and implemented. The hybrid architecture (in that it combines both modeling and retrieval features) was introduced in Chapter 3, followed by the features modeling sub-system (Chapter 4), an use case on audio retrieval enhanced with a specific domain ontology (Chapter 5) and, finally, by the interface autocomposition functionalities (Chapter 6). The integrated architecture, herein presented, results in a semantic-rich and flexible mechanism for generating automatically any type of multimedia retrieval system. Of course, this is beneficial for the scientific, research and application communities.

The main strengths of the proposed architecture are its flexibility, adaptability and integration capabilities in creating multimedia retrieval application according to the user needs and to the domain constrains. In fact, one of the main peculiarities of the proposed architecture is its adaptability to 1) a specific domain, simply through the design of a proper ontology, as, for instance, it has been demonstrated in the audio retrieval use case previously presented and 2) the user needs, by creating on-line instances of the developed ontologies for the specific domain, wherein the reasoner follows the user's specified constraints. The flexibility of the system was achieved by the Interface auto-composition. The user interfaces are created according to the instances of the ontology that describes the domain the user is dealing with. Moreover, for each auto-composed interface, the media processing algorithms that better adapt to the user requirements and domain constrains are chosen. The system ensures integration with existing applications by adopting the philosophy of the global and fully interconnected web of data: in fact, the entities are structured by ontologies universally recognized, the identifiers used are unique and the databases are usable from any external application.

Finally, in the thesis an OWL-based approach for building and editing ontology feature models together with an OWL-based inferential engine for creating a feature configuration and checking ontology consistency has been presented.

While the approach that has been used to design and implement this MMR architecture has several prominent features, at the current stage, it has some limitations. First, the computational capability of the system depends only on the performance of the machine that hosts the implemented services. However, the architecture may adapt easily to a distributed context e.g. in a web, Grid or Cloud services since it relies on semantic web technologies. Indeed, one next step will be to test it in a distributed environment, such as the one described in Appendix E, especially with respect to storage and processing issues. Currently, the system does not contain enough processing algorithms for creating different use cases (e.g. video, image, 3D models…), although the built-in flexibility allows the users to include easily new ones (e.g., the relevance feedback algorithm in Appendix E). Once the collection of available processing algorithms will be enriched, it will be possible to evaluate more systematically the added value of the proposed adaptive compositional capabilities from the user's perspective and system's performance point of view.

Appendix A

RDF (Resource Description Framework)

The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web. This specification is one of several [RDF-PRIMER] [RDF-SYNTAX] [RDF-CONCEPTS] [RDF-SEMANTICS] [RDF-TESTS] related to RDF. The reader is referred to the RDF schema chapter in the RDF Primer [RDF-PRIMER] for an informal introduction and examples of the use of the concepts specified in this document. This specification introduces RDF's vocabulary description language, RDF Schema. It is complemented by several companion documents which describe RDF's XML encoding [RDF-SYNTAX], mathematical foundations [RDF-SEMANTICS] and Resource Description Framework (RDF): Concepts and Abstract Syntax [RDF-CONCEPTS]. The RDF Primer [RDF-PRIMER] provides an informal introduction and examples of the use of the concepts specified in this document. First, let us describe the RDF Schema Specification, based on [W3C 1999b], in order to discuss and point out some unconventional design decisions taken in this specification. We will try to make this chapter self-contained, but a working knowledge of [W3C 1999b] will help to understand the discussion in this section. The prefixes rdf: and rdfs: indicate, whether a resource is part of the RDF Data Model [W3C 1999a] or the RDF Schema Specification [W3C 1999b]. RDF schemas are used to define the structure of the metadata that are used to describe WWW resources (i.e. WWW pages or parts of WWW pages, referenced by an URL). The RDF Schema Specification consists of some basic classes and properties, and can be extended by others to fit possibly any given domain. Classes are arranged hierarchically, and the use of properties can be constrained to members of certain classes. The root of the class hierarchy is rdfs:Resource, rdfs:Class is subclass of rdfs:Resource.

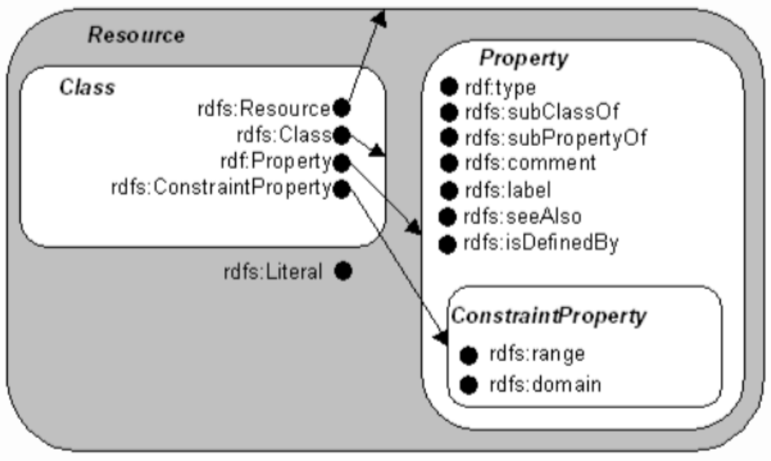


Figure 29 RDF Classes and Resources as Sets and Elements

Properties are defined by the rdf:Property class and can be seen as attributes, that are used to describe resources by assigning values to them. Properties are resources themselves. The RDF Schema Specification defines four specific properties (rdfs:subClassOf, rdf:type, rdfs:range, rdfs:domain) that have, unlike other predefined or self-defined properties, certain constraints. These four properties are both used to define the other RDF schema constructs and also as constructs defined in the RDF schema. Additional predefined properties such as rdfs:seeAlso and rdfs:comment are used to specify resources with related subjects, or to give a human readable description of a resource. The fact, that these properties are predefined can be seen as a convenience, they are not needed for the definition of other properties. Figure 43, 44 and 45 (which we have reproduced from [W3C 1999b]) show the RDF schema specification as a set of pictures. We will use an abbreviated description of these pictures based on the text in [W3C 1999b] and discuss the design issues we want to address in our alternative RDF schema specification model. Figure 1 shows RDF classes, subclasses and resources as sets, subsets and elements. A class is depicted by a rounded rectangle, a resource is depicted by a large dot. Arrows are drawn from a resource to the class it defines. A sub-class is shown by having a rounded rectangle (the sub-class) completely enclosed by another (the super-class). If a resource is inside a class, then there exists either an explicit or implicit rdf:type property of that resource whose value is the resource defining the containing class. The constraint properties rdfs:range and rdfs:domain are distinguished from the other predefined properties. The property rdf:type is present both as a specific property and depicted as an arrow, rdfs:subClassOf both as a specific property and depicted as set containment. Figure 44 shows the same information about the class hierarchy as in figure Figure 43, but does so using a „nodes and arcs“ graph representation of the RDF data model. If a class is a subset of another, then there is an rdfs:subClassOf arc from the node representing the first class to the node representing the second. Similarly, if a Resource is an instance of a Class, then there is an rdf:type arc from the resource to the node representing the class.

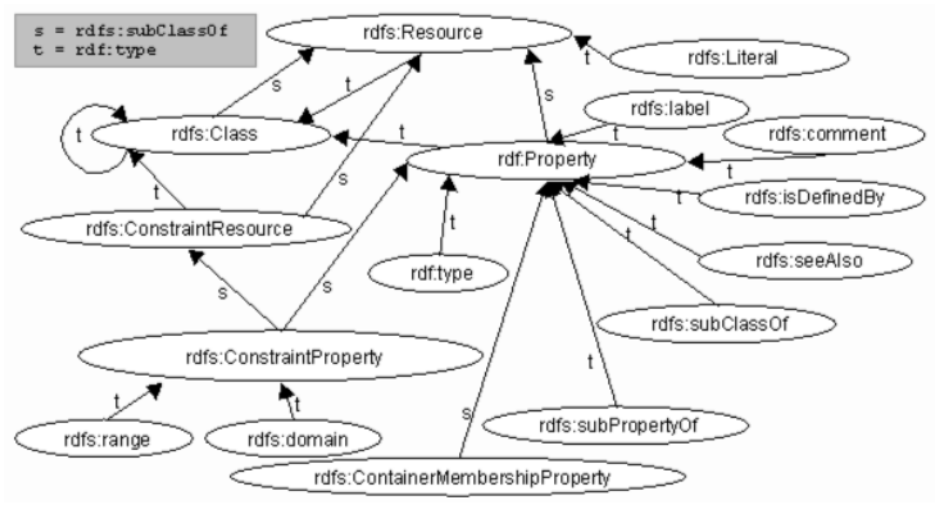


Figure 30 Class Hierarchy for the RDF Schema

Again, rdfs:subClassOf is present both as a specific property and a primitive construct (an arrow labelled with „s“), rdf:type as specific instance of property and as primitive construct (an arrow labelled with „t“).

Appendix B

OWL (Web Ontology Language)

The expressivity of RDF and RDF Schema is deliberately very limited: RDF is (roughly) limited to binary ground predicates, and RDF Schema is (again roughly) limited to a subclass hierarchy and a property hierarchy, with domain and range definitions of these properties. However, the Web Ontology Working Group of W3C3 identified a number of characteristic use-cases for Ontologies on the Web which would require much more expressiveness than RDF and RDF Schema. A number of research groups in both America and Europe had already identified the need for a more powerful ontology modeling language. This lead to a joint initiative to define a richer language, called DAML+OIL4 (the name is the join of the names of the American proposal DAML-ONT5, and the European language OIL6). DAML+OIL in turn was taken as the starting point for the W3C Web Ontology Working Group in defining OWL, the language that is aimed to be the standardized and broadly accepted ontology language of the Semantic Web. In this chapter, we first describe the motivation for OWL in terms of its requirements, and the resulting non-trivial relation with RDF Schema. We then describe the various language elements of OWL in some detail.

Requirements for ontology languages

Ontology languages allow users to write explicit, formal conceptualizations of domains models. The main requirements are:

1. a well-defined syntax

2. a well-defined semantics

3. efficient reasoning support

4. sufficient expressive power

5. convenience of expression.

The importance of a well-defined syntax is clear, and known from the area of programming languages; it is a necessary condition for machine-processing of information. All the languages we have presented so far have a well-defined syntax. DAML+OIL and OWL build upon RDF and RDFS and have the same kind of syntax. Of course it is questionable whether the XML-based RDF syntax is very user-friendly, there are alternatives better suitable for humans (for example, see the OIL syntax). However this drawback is not very significant, because ultimately users will be developing their ontologies using authoring tools, or more generally ontology development tools, instead of writing them directly in DAML+OIL or OWL. Formal semantics describes precisely the meaning of knowledge. \Precisely" here means that the semantics does not refer to subjective intuitions, nor is it open to different interpretations by different persons (or machines). The importance of formal semantics is well-established in the domain of mathematical logic, among others. One use of formal semantics is to allow humans to reason about the knowl- edge. For ontological knowledge we may reason about:

- Class membership: If x is an instance of a class C, and C is a subclass of D, then we can infer that x is an instance of D.

- Equivalence of classes: If class A is equivalent to class B, and class B equivalent to class C, then A is equivalent to C, too.
- Consistency: Suppose we have declared x to be an instance of the class A. Further suppose that - A is a subclass of B \ C - A is a subclass of D - B and D are disjoint Then we have an inconsistency because A should be empty, but has the instance x. This is an indication of an error in the ontology.
- Classification: If we have declared that certain property-value pairs are sufficient condition for membership of a class A, then if an individual x satisfies such conditions, we can conclude that x must be an instance of A.

Semantics is a prerequisite for reasoning support: Derivations such as the above can be made mechanically, instead of being made by hand. Reasoning support is important because it allows one to

- check the consistency of the ontology and the knowledge; Web Ontology Language: OWL 3
- check for unintended relationships between classes.
- automatically classify instances in classes

Automated reasoning support allows one to check many more cases than what can be done manually. Checks like the above are valuable for

- designing large ontologies, where multiple authors are involved;
- integrating and sharing ontologies from various sources.

Formal semantics and reasoning support is usually provided by mapping an ontology language to a known logical formalism, and by using automated reasoners that already exist for those formalisms. We will see that OWL is (partially) mapped on description logic, and makes use of existing reasoners such as FaCT and RACER. Description logics are a subset of predicate logic for which efficient reasoning support is possible.

Limitations of the expressive power of RDF Schema

RDF and RDFS allow the representation of some ontological knowledge. The main modeling primitives of RDF/RDFS concern the organization of vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions, and instances of classes. However a number of other features are missing. Here we list a few:

- Local scope of properties: rdfs:range defines the range of a property, say eats, for all classes. Thus in RDF Schema we cannot declare range restrictions that apply to some classes only. For example, we cannot say that cows eat only plants, while other animals may eat meat, too.
- Disjointness of classes: Sometimes we wish to say that classes are disjoint. For example, male and female are disjoint. But in RDF Schema we can only state subclass relationships, e.g. female is a subclass of person.
- Boolean combinations of classes: Sometimes we wish to build new classes by combining other classes using union, intersection and complement. For example, we may wish to define the class person to be the disjoint union of the classes male and female. RDF Schema does not allow such definitions.

- Cardinality restrictions: Sometimes we wish to place restrictions on how many distinct values a property may or must take. For example, we would like to say that a person has exactly two parents, and that a course is taught by at least one lecturer. Again such restrictions are impossible to express in RDF Schema.
- Special characteristics of properties: Sometimes it is useful to say that a property is transitive (like \greater than"), unique (like \is mother of"), or the inverse of another property (like \eats" and \is eaten by").

So we need an ontology language that is richer than RDF Schema, a language that offers these features and more. In designing such a language one should be aware of the tradeof between expressive power and efficient reasoning sup- port. Generally speaking, the richer the language is, the more inefficient the reasoning support becomes, often crossing the border of non-computability. Thus we need a compromise, a language that can be supported by reasonably efficient reasoners, while being sufficiently expressive to express large classes of ontologies and knowledge.

Compatibility of OWL with RDF/RDFS

Ideally, OWL would be an extension of RDF Schema, in the sense that OWL would use the RDF meaning of classes and properties (rdfs:Class, rdfs:subClassOf, etc), and would add language primitives to support the richer expressiveness identified above. Unfortunately, the desire to simply extend RDF Schema clashes with the trade-of between expressive power and efficient reasoning mentioned be- fore. RDF Schema has some very powerful modelling primitives, such as the rdfs:Class (the class of all classes) and rdf:Property (the class of all properties). These primitives are very expressive, and will lead to uncontrollable computational properties if the logic is extended with the expressive primitives identified above.

Three species of OWL

All this as lead to a set of requirements that may seem incompatible: efficient reasoning support and convenience of expression for a language as powerful as a combination of RDF Schema with a full logic. Indeed, these requirements have prompted W3C's Web Ontology Working Group to define OWL as three different sublanguages, each of which is geared towards fulfilling different aspects of these incompatible full set of requirements:

- OWL Full: The entire language is called OWL Full, and uses all the OWL languages primitives (which we will discuss later in this chapter). It also allows combining these primitives in arbitrary ways with RDF and RDF Schema. This includes the possibility (also present in RDF) to change the meaning of the pre-defined (RDF or OWL) primitives, by applying the language primitives to each other. For example, in OWL Full we could impose a cardinality constraint on the class of all classes, essentially limiting the number of classes that can be described in any ontology. The advantage of OWL Full is that it is fully upward compatible with RDF, both syntactically and semantically: any legal RDF document is also a legal OWL Full document, and any valid RDF/RDF Schema conclusion is also a valid OWL Full conclusion. The disadvantage of OWL Full is the language has become so powerful as to be undecidable, dashing any hope of complete (let alone efficient) reasoning support.

- OWL DL: In order to regain computational efficiency, OWL DL (short for: Description Logic) is a sublanguage of OWL Full which restricts the way in which the constructors from OWL and RDF can be used. We will give details later, but roughly this amounts to disallowing application of OWL's constructor's to each other, and thus ensuring that the language corresponds to a well studied description logic. The advantage of this is that it permits efficient reasoning support. The disadvantage is that we loose full compatibility with RDF: an RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document. Conversely, every legal OWL DL document is still a legal RDF document.
- OWL Lite: An ever further restriction limits OWL DL to a subset of the language constructors. For example, OWL Lite excludes enumerated classes, disjointness statements and arbitrary cardinality (among others). The advantage of this is a language that is both easier to grasp (for users) and easier to implement (for tool builders). The disadvantage is of course a restricted expressivity.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL and OWL Full. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations will be impossible.

There are strict notions of upward compatibility between these three sub-languages:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL still uses RDF and RDF Schema to a large extent:

- all varieties of OWL use RDF for their syntax
- instances are declared as in RDF, using RDF descriptions and typing in- formation
- OWL constructors like owl:Class, owl:DatatypeProperty and owl:ObjectProperty are all specialisations of their RDF counterparts. Figure 1 shows the subclass relationships between some modelling primitives of OWL and RDF/RDFS. 6 Grigoris Antoniou and Frank van Harmelen rdfs:Class owl:Class owl:ObjectProperty owl:DatatypeProperty rdf:Property rdfs:Resource

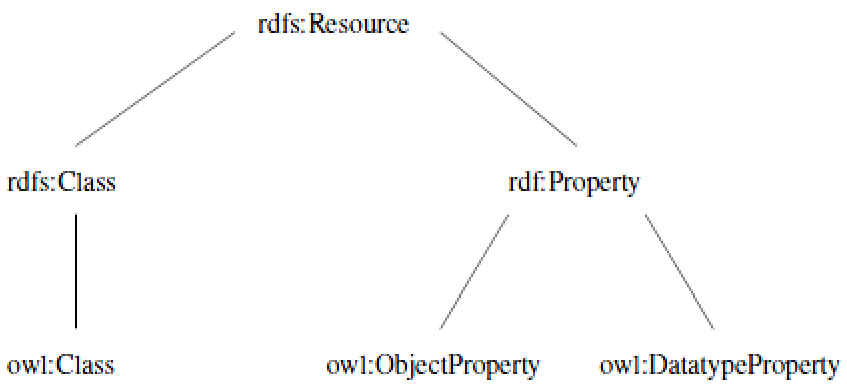


Figure 31 Subclass relationships between OWL and RDF/RDFS

The original hope in the design of OWL was that there would be a down- ward compatibility with corresponding re-use of software across the various layers. However, the advantage of full downward compatibility for OWL (that any OWL aware processor will also provide correct interpretations of any RDF Schema document) is only achieved for OWL Full, at the cost of computational intractability.

Appendix C
Okkam

The OKKAM project aims at enabling the Web of Entities, namely a virtual space where any collection of data and information about any type of entities (e.g. people, locations, organizations, events, products, ...) published on the Web can be integrated into a single virtual, decentralized, open knowledge base (like the Web did for hypertexts, readhere what Tim Berners-Lee says on this parallel). OKKAM will contribute to this vision by supporting the convergence towards the use of a single and globally unique identifier for any entity which is named on the Web. The intuition of the project is that the concrete realization of the Web of Entities requires that we enable tools and practices for cutting to the root the proliferation of unnecessary new identifierss for naming the entities which already have a public identifier (the OKKAM's razor). Therefore, OKKAM will make available to content creators, editors and developers a global infrastructure and a collection of new tools and plugins which support them to easily find public identifiers for the entities named in their contents/services, use them for creating annotations, build new network-based services which make essential use of these identifiers in an open environment (like the Web or large Intranets). To realize this vision, OKKAM proposes the following roadmap:

providing a scalable and sustainable infrastructure, called the Entity Name System (ENS), for making the systematic reuse of global and unique entity identifiers not only possible, but easy and straightforward. The ENS will be a distributed service which permanently stores identifiers for entities and provides a collection of core services (e.g. entity matching, ID mapping and resolution) needed to support their pervasive reuse;

bootstrapping and enabling the fast growth of Web of Entities by fostering the creation of OKKAMized content (i.e. content where entities are named or annotated with OKKAM IDs) in OKKAM-empowered applications (i.e. applications which can interact with the ENS for getting and reusing identifiers); showcasing the benefits of enabling the Web of Entities and, more in general, of an entity-oriented approach to content and knowledge management by building relevant applications on top of the new infrastructure in three important areas: information retrieval and semantic search, content authoring (more specifically, in scientific publishing andnews production) and organizational knowledge management.

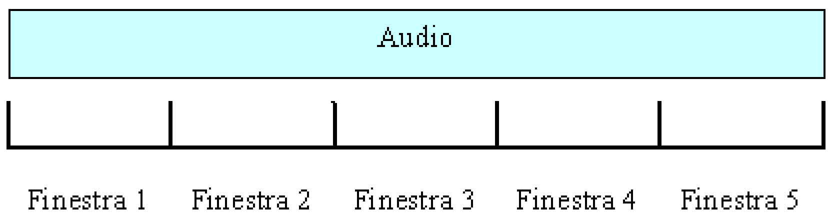
The impact of the proposed infrastructure cannot be easily overestimated. Not only it will provide a general service for entity-level integration of virtually any type of data and service into the global Web of Entities; but it will also provide the solid foundation for a whole generation of new applications and services which will benefit from the use of global identifiers in large collections of OKKAMized content and data.

Appendix D
Implemented Methods

Now we're going to expose some of the functions implemented. For some of the features proposed exists different versions, to differentiate the performance levels. It’s possible to study audio content according to different criteria:

Windows (overlap)

Some operations, especially preprocessing operations, are usually performed on the entire audio, while others using the intervals of the incoming content. It’s necessary that the input file is divided into sections called windows. When a window is created it’s possible to have overlapping. Suppose we make a simple division of an audio content:



The signal is partitioned into fixed-length windows. According to this approach the problem is given by the extremes of the windows. In the extremes it’s possible to lose some important elements of audio, especially in subsequent processing. In this case, can be useful to have some overlap between windows (overlapping). It’s possible to variate the overlapping windows in accordance with the requirements of accuracy and speed of execution.

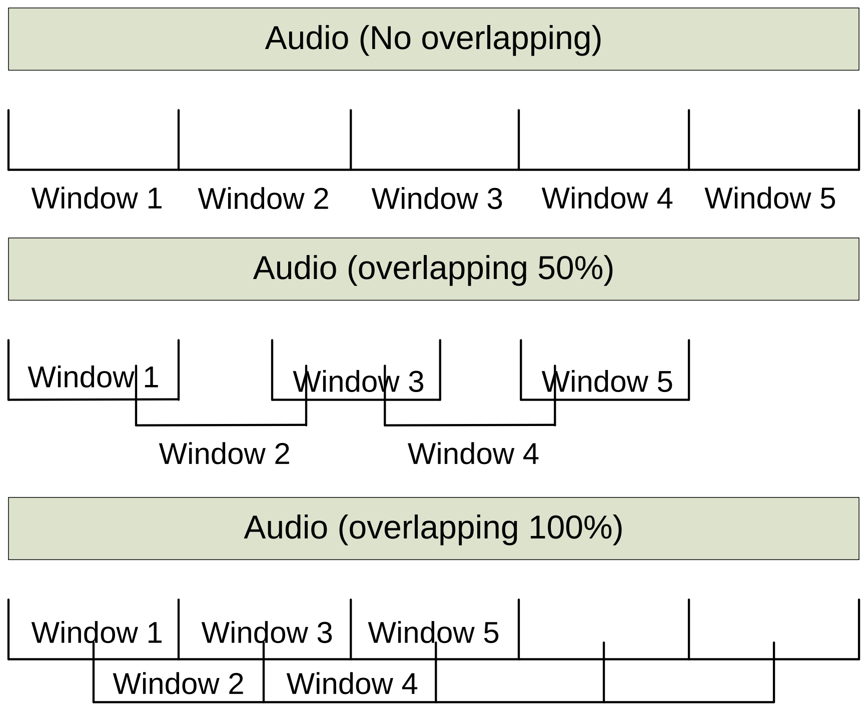


Figure 32 Overlapping Examples

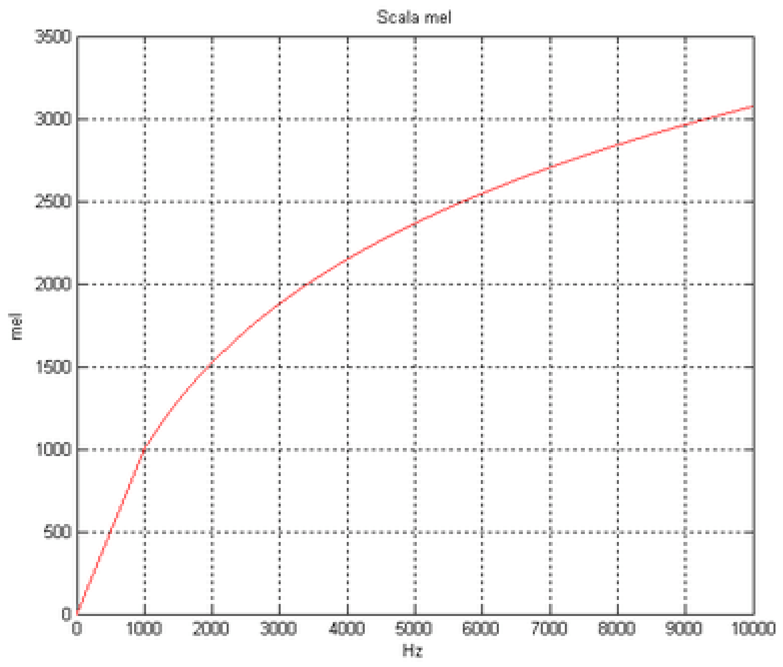
The percentage of overlapping indicated refers to the amount of each window subject to overlap. In the last section we see that every window is subject to overlap in all samples, so the value will be 100%. Have overlapping, as mentioned above, provides a reasonable assurance of not losing important data. Inside our system there are several overlapping modes that coincide with those see in the picture. The processing layer implements the operation of three possible overlapping in different ways. The Ontology Layer select which of those overlapping oerations, best adapted to the user requests.

FFT

A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory; this article gives an overview of the available techniques and some of their general properties, while the specific algorithms are described in subsidiary articles linked below. A DFT decomposes a sequence of values into components of different frequencies. This operation is useful in many fields (see discrete Fourier transform for properties and applications of the transform) but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly: computing a DFT of N points in the naive way, using the definition, takes O(N2) arithmetical operations, while an FFT can compute the same result in only O(N log N) operations. The difference in speed can be substantial, especially for long data sets where N may be in the thousands or millions—in practice, the computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to N / log(N). This huge improvement made many DFT-based algorithms practical; FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

Conversion in Mel scale

The mel scale, proposed by Stevens, Volkman and Newman in 1937 is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels.



Above about 500 Hz, larger and larger intervals are judged by listeners to produce equal pitch increments. As a result, four octaves on the hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale. The name mel comes from the word melody to indicate that the scale is based on pitch comparisons. A popular formula to convert f hertz into m mel is:

$$mel(f) = \begin{cases} f & \text{se } f \leq 1kHz \\ 2595 \cdot Log(1 + \frac{f}{700}) & \text{se } f > 1kHz \end{cases}$$

Discrete Cosine Transform

A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio and images (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications: for compression, it turns out that cosine functions are much more efficient (as explained below, fewer are needed to approximate a typical signal), whereas for differential equations the cosines express a particular choice of boundary conditions.

Appendix E

Related Technologies

Visual Attention for Implicit Relevance Feedback in CBIR

In this section we propose an implicit relevance feedback method with the aim to improve the performance of known Content Based Image Retrieval (CBIR) systems by re-ranking the retrieved images according to users’ eye gaze data. This represents a new mechanism for implicit relevance feedback, in fact usually the sources taken into account for image retrieval are based on the natural behavior of the user in his/her environment estimated by analyzing mouse and keyboard interactions. In detail, after the retrieval of the images by querying CBIRs with a keyword, our system computes the most salient regions (where users look with a greater interest) of the retrieved images by gathering data from an unobtrusive eye tracker, such as Tobii T60. According to the features, in terms of color, texture, of these relevant regions our system is able to re-rank the images, initially, retrieved by the CBIR. Performance evaluation, carried out on a set of 30 users by using Google Images and “pyramid” like keyword, shows that about the 87% of the users is more satisfied of the output images when the re-raking is applied.

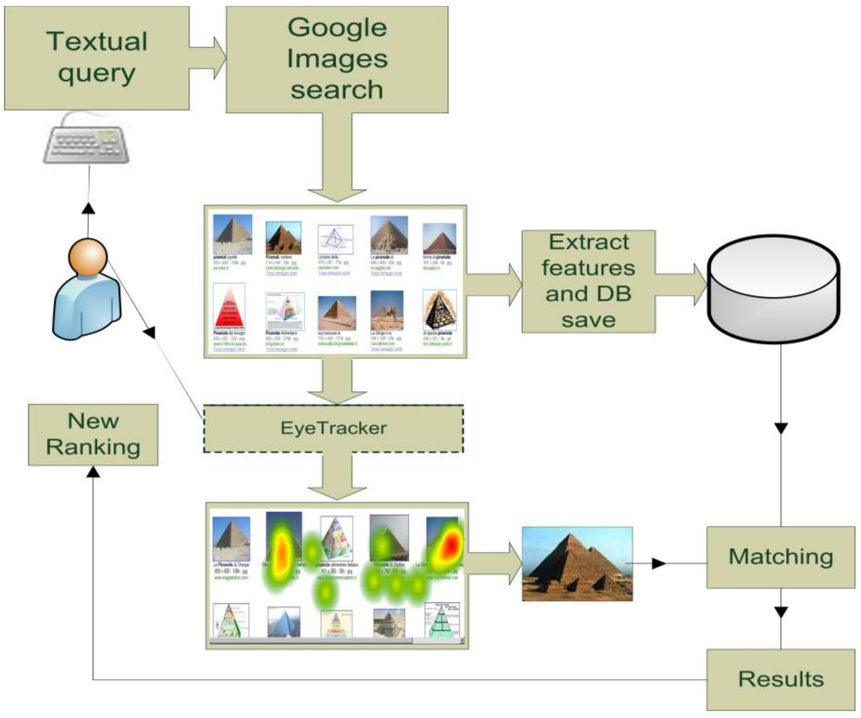


Figure 33 Implicit Relevance Feedback for the new Ranking Method in web-based CBIR.

The Proposed System

In order to improve the ranking provided by the search on a CBIR environment, a system that uses an eye tracker to capture an implicit relevance feedback and to classify the images in a different order of relevance has been created. The aim of this is to capture, by an eye tracker, the user's gaze fixations in order to identify the characteristics of the images s/he declares to be of her/his interest. This will allow the tool to retrieve automatically further relevant images. The tool may be also able to discover in an unsupervised way the characteristics of the images of potential user interest. Indeed, it is able to derive the characteristics of the images of user interest by considering the images, which mainly captured the user attention, e.g., by taking into account the user visual activity over the analyzed images. In the former case the tool learns how to select further relevant images, whereas in the latter case it could be also able to reclassify the images already examined by the user suggesting to her/him of reconsidering more deeply some potentially relevant images. Although the system proposed has been only tested on Google images to improve the precision of the retrieval, it may be applied to improve the precision of the retrieval of any document on the basis of the images featuring the documents. Figure 29 shows the general architecture of proposed implicit relevance feedback, where we point out the system ability of rearranging the images initially retrieved from a web-based CBIR (e.g. Google Images) without any user supervision, i.e., only on the basis of the user gaze fixations. A fine tuning of the characteristics to be possessed by the images may be carried out by the system on the basis of the user agreement for a better rearrangement of the images or for extracting relevant images from other datasets. In detail, the re-ranking mechanism is composed of the following steps:

- **First Image Retrieval**. The user enters some keywords on the used CBIR and observes the results. During this phase, the eye tracker stores gaze fixations on the thumbnails of the retrieved images, which most captured the user attention and her/his eye movements;
- **Features Extraction**. One of the crucial point in CBIR is the choice of low-level features, to be used to compare the image under test with the queried image. The features combination determines the

effectiveness of research. The extracted features can be related to the entire image, so we are talking about global features, or to its portion, then we are talking about local features. The local features extraction is more complex, because it requires a first step for the detection of the important regions of the image, such as clustering algorithms and object recognition, but it permits a considerable reduction of computational complexity of search algorithms. In our case the detection is simplified by the eye tracker, which allows us to identify the regions of major interest. The local features, considered for describing image content, are:

  - Brightness;
  - Smoothness;
  - Contrast;
  - Correlation;
  - Energy;
  - Homogeneity;
  - Gabor filters.

- Therefore, in the proposed system, the images returned by the CBIR and the file containing the data taken by the eye tracker are processed in order to identify the most relevant images and their features.
- Re-Ranking. The values of the extracted features, which should be possessed by the images to best fit the user interest, are then processed to produce a ranking of the images initially retrieved. In detail, we compute similarity scores (which represents a sort of implicit relevance feedback) between the most relevant images, detected at the previous step, and the images retrieved at the first step (see fig. 30).

The metrics to evaluate the similarity is based on the concept of distance,

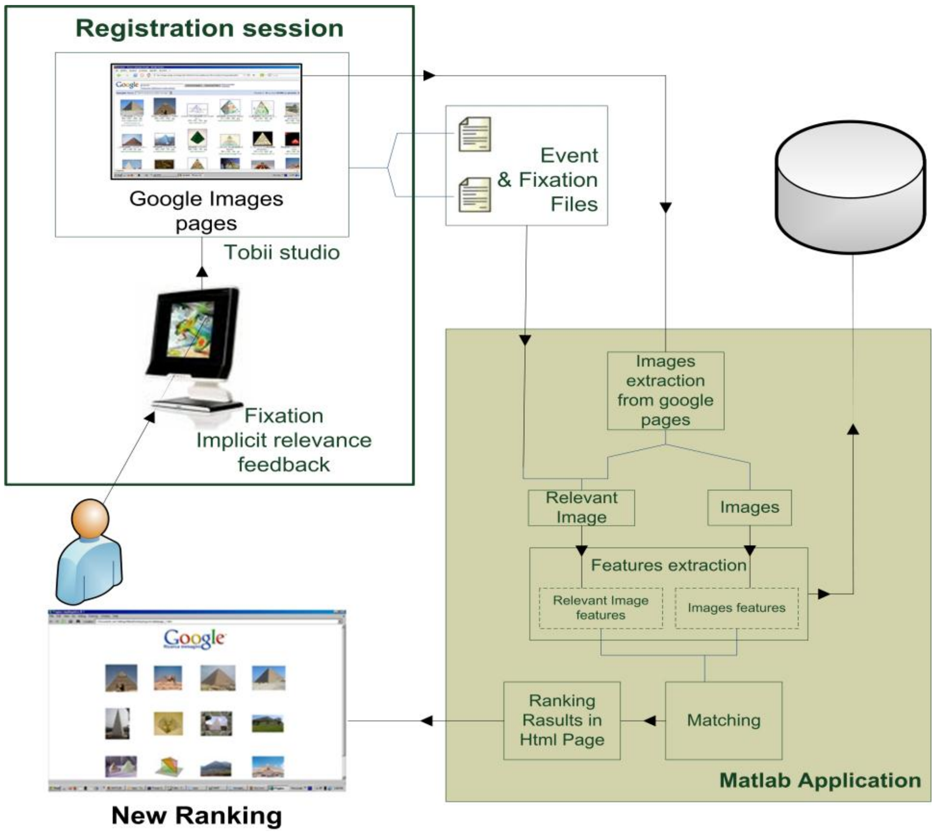


Figure 34 System Architecture.

measured between the features of the most salient images (extracted at the previous step) and the features of the images initially retrieved (at step 1). The images are re-ranked by using these similarity scores.

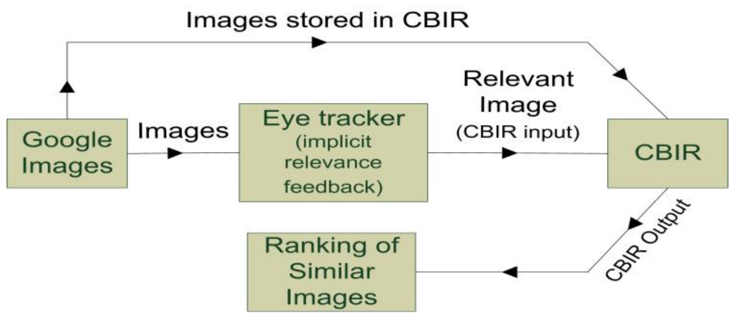

Images stored in CBIR
Google Images
Images
Eye tracker (implicit relevance feedback)
Relevant Image (CBIR input)
CBIR
CBIR Output
Ranking of Similar Images

Figure 35 Eye Tracker with the implicit relevance feedback produces an image input for CBIR system.

The relevance feedback detected by the eye tracker could be improved by taking into account the ranking carried out by other methods, e.g., by the ones, which model the user behavior during the phase of image analysis from how the user operates on the mouse and keyboard.

User Interface and Experimental Results

The system has been implemented by integrating the functionality of the Tobii Studio to Matlab 7.5 responsible for processing the output provided from the eye tracker. The Tobii studio makes possible to register a web browsing, setting appropriate parameters such as the URL and the initial size of the window on the web browser. By default the web browsing is set to http://images.google.com/ as homepage, whereas the window size and resolution are put equal to the entire screen and the maximum resolution allowed by the monitor. After a proper training phase of the instrument, the user is authorized to start regular recording sessions that terminate by pressing the F10 key on the keyboard. At the end of the session the user should confirm the export in textual form of the two files related to fixations and events needed for the computation of the relevance feedback. Thus, the information representing the gaze fixations and the one related to the images, which are merged in the same picture, are actually separated into two files. To evaluate the effectiveness of the proposed system for increasing the precision of the information retrieval carried out by Google Images, we will show below how the system rearranges significantly the collection of images proposed by Google in response to the word "pyramid" and we will evaluate the performance increase as perceived by a set of 30 users. Indeed, such collection is proposed without any knowledge of the user interest by merging images of pyramid where the subject is either a monument or a geometric solid (see fig. 32). With the eye tracker we may go insight the user interests, by discovering, for example that s/he is more interested in the pyramids as monuments since the more fixed images are related to the Egyptian pyramids (see fig. 33). With this information at hand it is relatively easy for the system to discover, after the recording session, the images relevant for the user following the processing procedure pointed out in the previous section. Fig. 34 shows the collection of the images as re-proposed by our system. The new ranking correctly suggests a sequence that favors the pyramids more similar to those observed and then

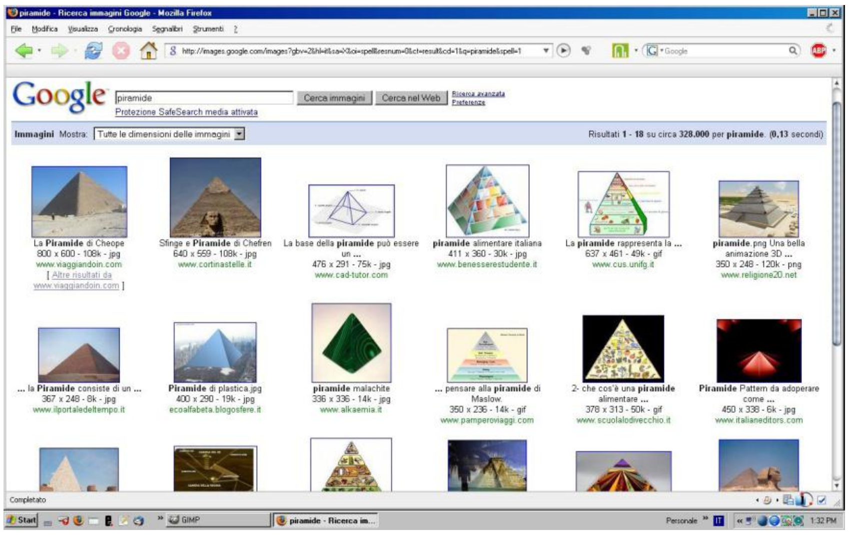

piramide - Ricerca immagini Google - Mozilla Firefox
Google
piramide
Cerca immagini
Cerca nel Web
Protezione SafeSearch media attivata
Immagini Mostra: Tutte le dimensioni delle immagini
Risultati 1 - 18 su circa 328.000 per piramide. (0,13 secondi)
La Piramide di Cheope
Sfinge e Piramide di Chefren
La base della piramide può essere un ...
piramide alimentare italiana
La piramide rappresenta la ...
piramide.png Una bella animazione 3D ...
... la Piramide consiste di un ...
Piramide di plastica.jpg
piramide malachite
... pensare alla piramide di Maslow.
2- che cos'è una piramide alimentare ...
Piramide Pattern da adoperare come ...
Completato

Figure 36 Google Ranking for "Pyramid" Keyword.

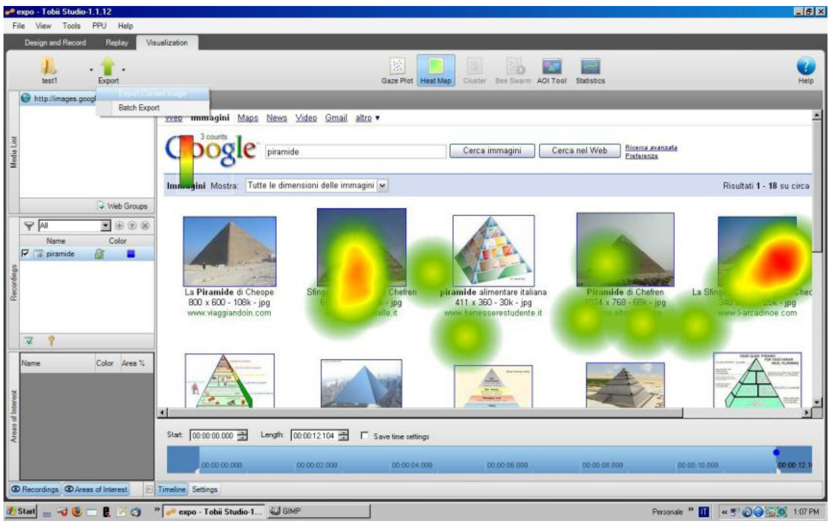


Figure 37 Gaze Fixations on the Images retrieved by Google using the "Pyramid" keyword.

requested by the user. The users will was caught with an implicit relevance feedback by taking into account that s/he was particularly attracted by a picture with the Sphinx in the foreground and the pyramid in the background. The proposed system was then able to discover meaningful information from how the perception process has been carried out by the user. Indeed, by the new re-proposed ranking, at the top two places there are images with the pyramid and the Sphinx.

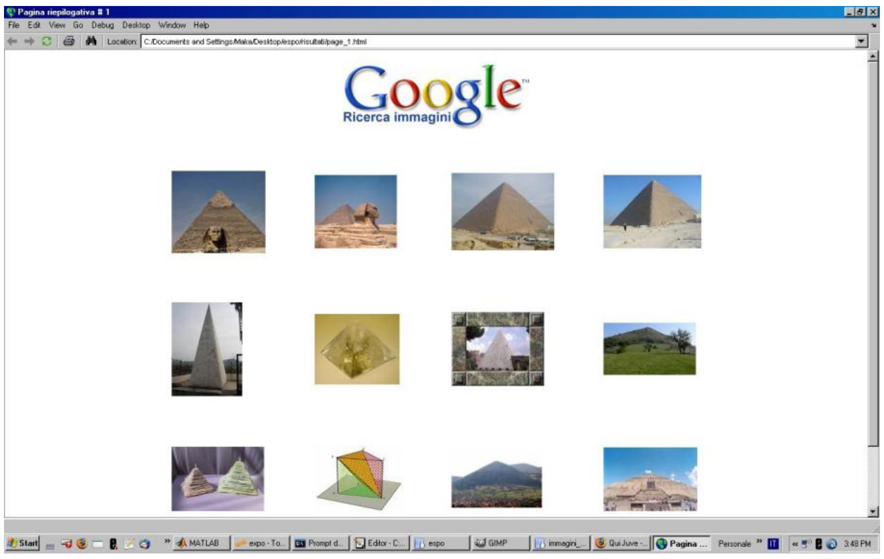

Pagina riepilogativa # 1
File Edit View Go Debug Desktop Window Help
Location: C:/Documents and Settings/Maria/Desktop/espo/risultati/page_1.html
Google
Ricerca immagini
Start
MATLAB
GIMP
Pagina ...
3:48 PM

Figure 38 New Images Pyramid Ranking according to the Eye Tracker feedback given by the user.

Distributed architecture for sharing and retrieving medical images

Large amounts of images (SPECT, PET, etc…) in nuclear medicine field have been routinely produced in the last years. In this section we propose an image management system that allows nuclear medicine physicians to share the acquired images and the associated metadata both locally (i.e. within the same medical institute) and globally with other physicians located in any part of the world by using GRID services for data (LFC) and metadata (AMGA) storage. The proposed system guarantees medical data protection by anonymization that aims at removing most sensitive data for unauthorized users and encryption that guarantees data protection when it is stored at remote sites. Another important issue is that often nuclear medicine data is associated with other medical data (e.g. neurological data) for diagnosis and therapy follow-up. In order to correlate images with other clinical information, the common metadata are enriched by developing a controlled vocabulary, which integrates known standards such as FOAF, CCR and GeneOntology. All the metadata are stored in an RDF (Resource Description Framework) repository in order to make the system fully compatible with existing metadata storage systems following the semantic web's philosophy.

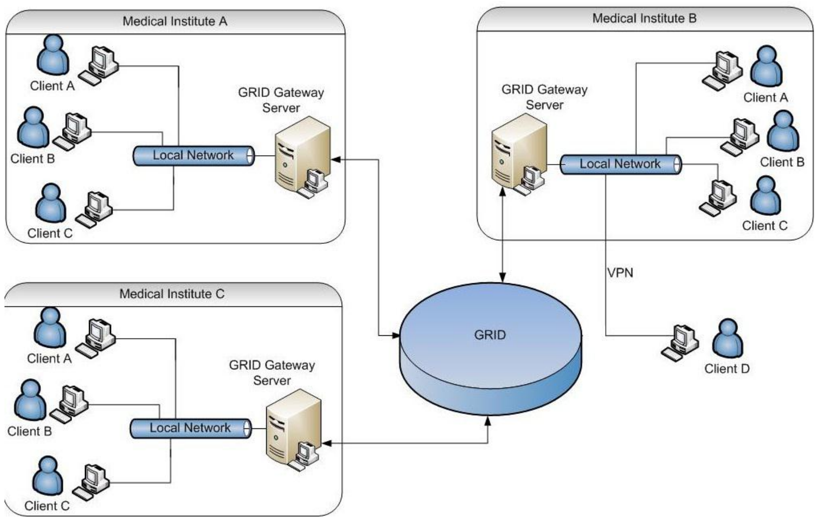


Figure 39 Architecture for Local and Global Data Storage and Sharing

Storage and sharing System Overview

In order to develop a distributed environment for image and information sharing to support the diagnosis, the treatment of patients and for statistical evaluation the system is provided with two levels of storage and sharing: the first is locally managed by a client-server architecture, deployed in the medical institute nuclear medicine physicians belong to, whereas the second one is on Grid and allows global data sharing, i.e. data may be shared among researchers within the same medical institute by using a client-server architecture or among different institutes using the services offered by the GRID computing. Fig. 35 shows the local and global data sharing. The typical use case is the following: a user, using a suitable interface, can store the images and the metadata of a performed examination in its own local database (located in his/her computer). Afterwards, the client creates an anonymous version of the data removing all the confidential information so they can be sent to the main server avoiding privacy issues. Additionally, the client allows users to define the set of metadata he/she wants to share both in GRID and in his/her medical institute. The data transmission between client and server runs asynchronously in order 1) to make the system robust because if no internet connection is available, data are locally stored and subsequently sent to the main server and to GRID when the connection will be available again and 2) to avoid doctors to have the perception of the actual time needed for the data transfer. The server contains the data and metadata repositories where

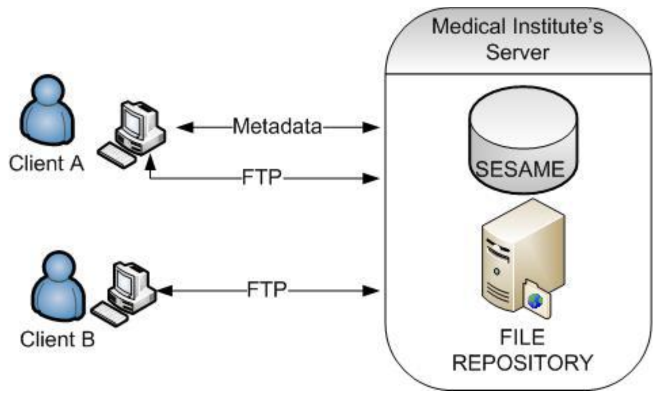


Figure 40 Local Data Storage and Sharing

all the data/metadata produced within the same institute are stored. The communication with GRID is delegated to it, thus optimizing the bandwidth's use. A. Local Data Storage and Sharing Inside a medical institute, data are stored and shared using a standard client-server architecture, as shown in fig. 36. The client and the server are connected by a local network or a VPN (Virtual Private Network). The client contains the user interface and implements the logic communication with the GRID infrastructure. It also contains a file repository (for image storage) and a SESAME server 1 (for RDF metadata storage), in order to save patient's data locally. The server also includes a file and a SESAME metadata repository for the data produced by all the physicians in the institute. Data are sent from the client to the server using FTP, whereas metadata is transmitted using SOAP requests since we implement a webservice for metadata storage in SESAME, as shown in fig. 36. By using the client interface a nuclear medicine physician can record and manage patients, add information to patient's clinical history (according to the schema shown in the next section), include any relevant documents (textual reports, generic images, DICOM images, etc..), run queries locally or on GRID data, associate the metadata deriving from the queries to the data locally stored and perfom statistical analysis on set of data and virtual data (i.e. coming from the main institute center or from GRID).

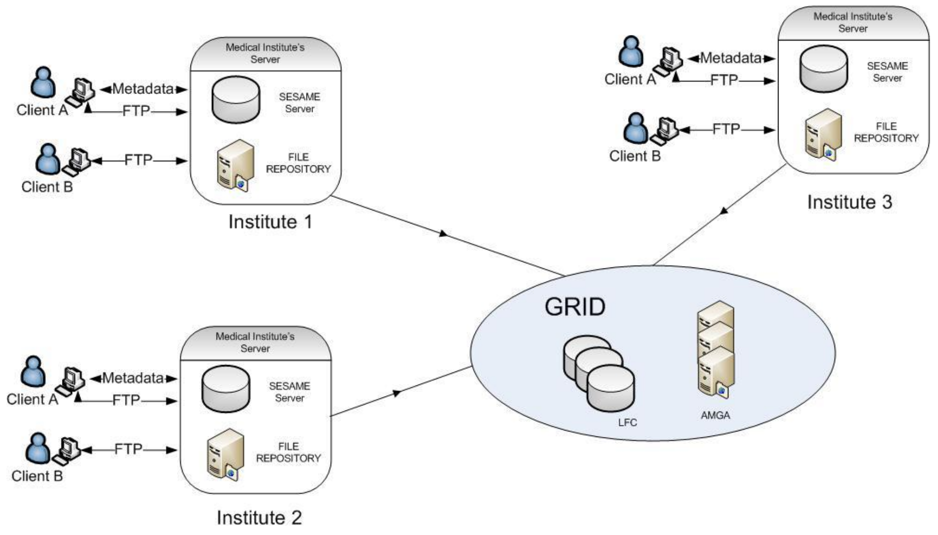


Figure 41 Global Data Storage and Sharing middleware2.

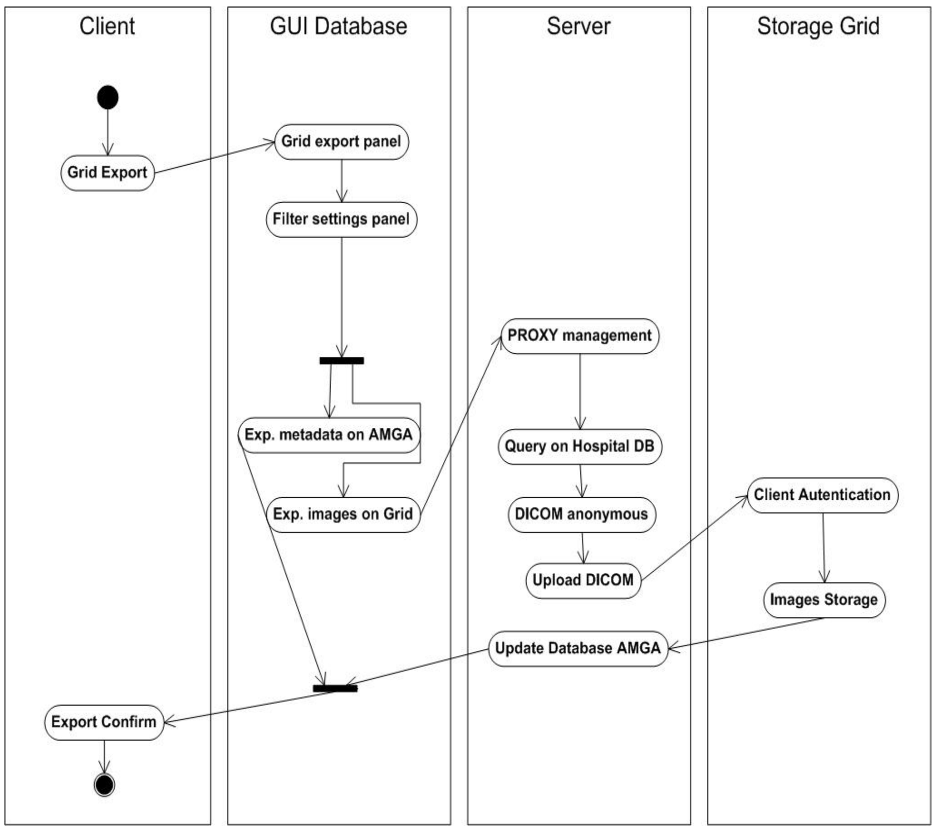


Figure 42 Interaction with Medical Institute - GRID Infrastructure server.

High Level Features

In order to provide useful information about the stored images and to make available them and the related metadata to the nuclear medicine community, the system is provided with high level features. More in detail, the system contains three processing levels:

- A semantic layer which aims at enriching patient metadata by constructing a controlled vocabulary using RDF/XML standard. This level guarantees the interoperability existing frameworks;
- A image processing layer that aims at analyzing the stored images. This is an important layer, since sometime is very useful to share only the processing results and not the entire image. This level performs the image analysis and interacts with the semantic layer for processing results storage in RDF/XML;
- Query Composition for performing complex queries both locally and on GRID. This module allows users to search useful information by processing only the metadata available locally or in GRID.

The interaction between the three levels and the system's architecture is shown in fig. 38. A. Semantic Layer Usually nuclear medicine images (PET, SPECT,..) are stored in DICOM format, containing the metadata provided with the standard. These metadata are not sufficient for describing the clinical history of patients. For this reason we enrich the information available in order to give the nuclear medicine physician the possibility to better figure out a specific disease by developing a model that represents the medical data so that it can be analyzed by semantic tools. In detail, the system stores concepts, specifies typed relationships between these concepts using RDF (Resource Description Framework) with XML syntax format. More in detail, we enrich the DICOM metadata by developing a controlled vocabulary that includes:

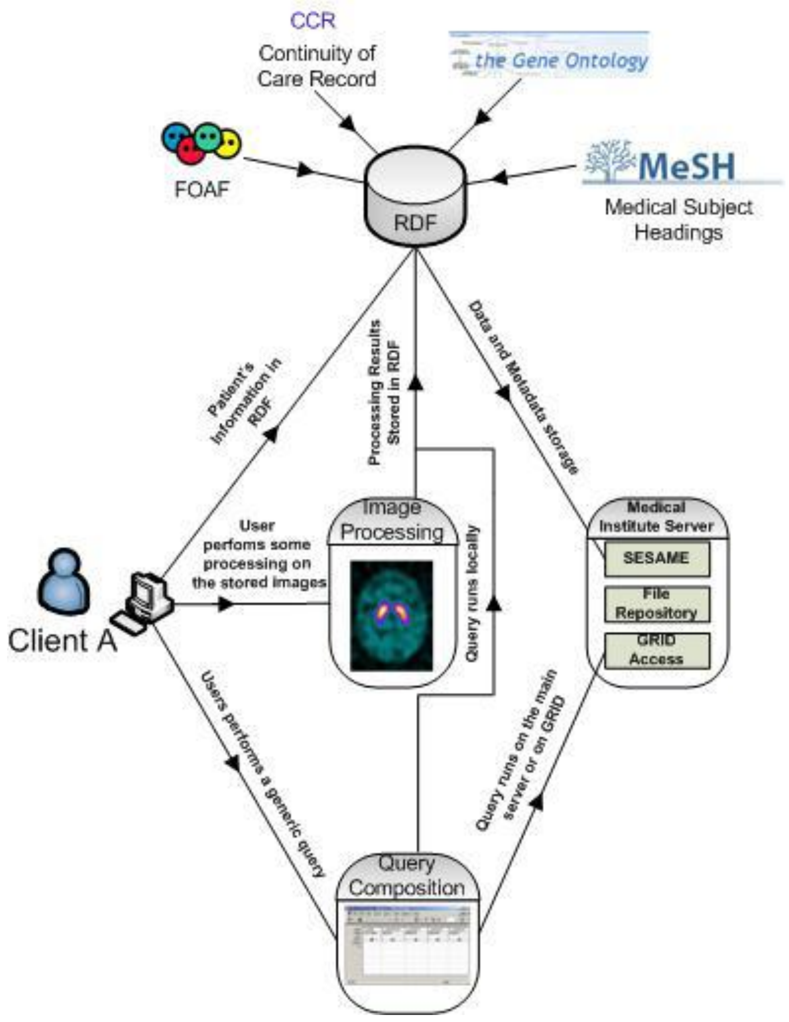

CCR
Continuity of Care Record
the Gene Ontology
FOAF
MeSH
Medical Subject Headings
RDF
Patient's Information in RDF
Processing Results Stored in RDF
Data and Metadata storage
Image Processing
Query runs locally
Medical Institute Server
SESAME
File Repository
GRID Access
User perfoms some processing on the stored images
Client A
Users performs a generic query
Query runs on the main server or on GRID
Query Composition

Figure 43 High Level Features

- Personal data by using FOAF ontology;
- Generic Health Information according to the CCR standard such as: Problems/diagnoses, Allergies, Medication list, Immunizations,

Family history, Social history, Vital signs, Procedures, Symptoms, Plan of care, Functional status, Biosignals (EEG, ECG, etc...);
- Genetic Information using GeneOntology;
- Neurological detailed information by using Mesh;
- Image Processing information that represent the output of the implemented image processing algorithm and which introduces a new semantic level to the stored metadata.

It is notable that the above information are inserted by the users, but they can be easily obtained by querying systems that share data using RDF. For instance, personal data in FOAF can be derived from a generic social network or by using a vcard; generic health information can be obtained by the user's Google Health Account3 or other systems that aim at storing online health care data. Metadata storage has been carried out by using SESAME server so that these information may be available also for other purporses. The sensitive data, such as Name, Surname, SSN, etc ... must be available only for the physician who carries out the examination, and are not exported in RDF in order to ensure data privacy. 3https://www.google.com/health/ B. Image Processing Layer This level is provided with a set of image processing utilities for SPECT and PET image analisys. The output of this processing is stored according to the semantic layer and is related to the specific processed image. This allows users to also share the results of the processing avoiding to send the original images when it is not required, resulting in less bandwidth occupation. The functions present for image analysis are:

- Measurement of distances, angles and some parameters within the images;

- The contrast absorption curve over time;
- Image Texture and Image Contour Analysis for specific organs;
- Pattern recognition for identifying brain structures.

Therefore when a user performs one of the above methods, the output will be treated as metadata and stored in the SESAME server. C. Query Composition The query composition level aims at building complex query both locally and on GRID. The queries are performed only on the metadata (stored in the SESAME server and in the AMGA server) since content based image retrieval module is not present. This level receives users query (by using a controlled GUI) and interacts both with the local storage performing SPARQL query on the SESAME server and with the GRID.

Bibliography

[1] Aslandogan, Y. A. and C. T. Yu, *Techniques and systems for image and video retrieval*, Knowledge and Data Engineering **11** (1999), pp. 56–63.

[2] Browne, P. and A. F. Smeaton, *Video information retrieval using objects and ostensive relevance feedback*, in: *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing* (2004), pp.1084–1090.

[3] Buckley, C., M. Mitra, J. A. Walz and C. Cardie, *Using clustering and superconcepts within SMART: TREC 6*, Information Processing and Management **36** (2000), pp. 109–131.

[4] Buijs, J. M. and M. S. Lew, *Visual learning of simple semantics in imagescape*, in: *Proceedings of the Third International Conference on Visual Information and Information Systems*, 2003.

[5] Bush, V., *As we may think*, Atlantic Monthly **176** (1945), pp. 101–108.

[6] Byrd, D. and T. Crawford, *Problems of music information retrieval in the real world*, Information Processing and Management: an International Journal **38** (2002), pp. 249–272.

[7] Cano, P., M. Koppenberger and N. Wack, *An industrial-strength content-based music recommendation system*, in: *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (2005), pp. 673–673.

[8] Carbonell, J., Y. Yang, R. Frederking, R. Brown, Y. Geng and D. Lee, *Translingual information retrieval: A comparative evaluation*, in: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997, pp. 708–715.

[9] Cesarano, C., A. d'Acierno and A. Picariello, *An intelligent search agent system for semantic information retrieval on the internet*, in: *WIDM '03: Proceedings of the 5th ACM international workshop on Web information and data management* (2003), pp. 111–117.

[10] Chang, E., L. Beitao, G. Wu and K. Goh, *Statistical learning for effective visual information retrieval*, in: *IEEE International Conference on Image Processing*, 2003.

[11] Chang, S. and T. Kunii, *Pictorial database systems*, IEEE Computer **14** (1981), pp. 13–21.

[12] Chen, A. and F. C. Gey, *Combining query translation and document translation in cross-language retrieval*, in: *4th Workshop of the Cross-Language Evaluation Forum*, 2004, pp. 108–121.

[13] Clark and Parsia, "Pellet: Owl 2 reasoner for java," http://clarkparsia.

com/pellet/, June 2009.

[14] Cohen, P. R. and R. Kjeldsen, *Information retrieval by constrained spreading activation in semantic networks*, Inf. Process. Manage. **23** (1987), pp. 255–268.

[15] Corridoni, J., A. D. Bimbo and P. Pala, *Image retrieval by color semantics*, Multimedia Systems **7** (1999), pp. 175–183.

[16] K. Czarnecki and U. Eisenecker, Generative Programming: Methods,

Tools, and Applications. Addison-Wesley Professional, June 2000,

ch. 5. [Online]. Available: http://www.worldcat.org/isbn/0201309777

[17] Dalrymple, P. W. and D. L. Zweizig, *Users' experience of information retrieval systems: An exploration of the relationship between search experience and affective measures*, Library and Information Science Research **14** (1992), pp. 167–81.

[18] Dill, S., N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin and J. Y. Zien, *Semtag and seeker:*

bootstrapping the semantic web via automated semantic annotation, in: *WWW '03: Proceedings of the 12th international conference on World Wide Web* (2003), pp. 178–186.

[19] Dingli, A., F. Ciravegna and Y. Wilks, *Automatic semantic annotation using unsupervised information extraction and integration*, in: *Proceedings of the K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation*, 2003.

[20] Downie, J., *Music information retrieval*, Annual Review of Information Science and Technology **37** (2003), pp. 295–340.

[21] Dumais, S. T., T. A. Letsche, M. L. Littman and T. K. Landauer, *Automatic cross-language retrieval using latent semantic indexing*, in: *AAAI Spring Symposium on Cross-Language Text and Speech Retrieval*, 1997.

[22] A. Faro, D. Giordano, C. Pino, C. Spampinato, *"Visual Attention for Implicit Relevance Feedback in a Content Based Image Retrieval"*, ETRA 2010

[23] Flynn, R., editor, "Computer Sciences: Macmillan Science Library," Macmillan Reference USA, 2002.

[24] Fuji Ren, David B. Bracewell,"*Advanced Information Retrieval"*, Electronic Notes in Theoretical Computer Science 225 (2009) 303–317

[25] Fukui, M., S. Higuchi, Y. Nakatani, M. Tanaka, A. Fuji and T. Ishikawa, *Applying a hybrid querytranslation method to japanese/english cross-language patent retrieval*, in: *ACM SIGIR 2000 Workshop on Patent Retrieval*, 2000.

[26] Gao, J. and J.-Y. Nie, *A study of statistical models for query translation: finding a good unit of translation*, in: *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), pp. 194–201.

[27] Gao, J., M. Zhou, J.-Y. Nie, H. He and W. Chen, *Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations*, in: *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 183–190.

[28] Gaughan, G., A. F. Smeaton, C. Gurrin, H. Lee and K. McDonald, *Design, implementation and testing of an interactive video retrieval system*, in: *MIR '03: Proceedings of the 5th ACM SIGMM international* workshop on *Multimedia information retrieval* (2003), pp. 23–30.

[29] Ghias, A., J. Logan, D. Chamberlin and B. C. Smith, *Query by humming: musical information retrieval in an audio database*, in: *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia* (1995), pp. 231–236.

[30] D. Giordano, C. Pino, C. Spampinato, M. Fargetta, D. Distefano, *"Nuclear Medicine Image Management System for Storage and Sharing by using GRID services and Semantic Web"*, BIOSTEC 2010

[31] Girgensohn, A., J. Adcock, M. D. Cooper and L. Wilcox, *A synergistic approach to efficient interactive video retrieval*, in: *INTERACT 2005*, 2005, pp. 781–794.

[32] Gomez-Perez, A., F. Ortiz-Rodriguez and B. Villazon-Terrazas, *Ontology-based legal information retrieval to improve the information access in e-government*, in: *WWW '06: Proceedings of the 15th international conference on World Wide Web* (2006), pp. 1007–1008.

[33] "Graphviz - graph visualization software," http://www.graphviz.org/,

2010.

[34] Gunsel, B., A. Ferman and A. Tekalp, *Temporal video segmentation using unsupervised clustering and semantic object tracking*, Journal of Electronic Imaging **7** (1998), pp. 592–604.

[35] P. D. V. Haarslev, P. D. R. M¨oller, K. Hidde, and M. Wessel, "Racer

pro," http://www.sts.tu-harburg.de/r.f.moeller/racer/, June 2009.

[36] Hedlund, T., E. Airio, H. Keskustalo, R. Lehtokangas, A. Pirkola and K. Jrvelin, *Dictionary-based cross-language information retrieval: Learning experiences from clef 2000-2002,* Information Retrieval **7** (2004), pp. 99–119.

[37] Hijikata, Y., K. Iwahama and S. Nishida, *Content-based music filtering system with editable user profile*, in: *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing* (2006), pp. 1050–1057.

[38] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, *Feature-*

Oriented Domain Analysis (FODA) Feasibility Study, 1990.

[39] Kato, T., T. Kurita, N. Otsu and K. Hirata, *A sketch retrieval method for full color image databasequery by visual example*, in: *Proceedings of the 11th IAPR International Conference on Computer Vision and Applications*, 1992.

[40] Khan, L., D. McLeod and E. Hovy, *Retrieval effectiveness of an ontology-based model for information selection*, The VLDB Journal The International Journal on Very Large Data Bases **13** (2004), pp. 71–85.

[41] Kiryakov, A., B. Popov, I. Terziev, D. Manov and D. Ognyanoff, *Semantic annotation, indexing, and retrieval.*, J. Web Sem. **2** (2004), pp. 49-79.

[42] Korfhage, R. R., "Information Storage and Retrieval," John Wiley and Sons, 1997.

[43] Lavrenko, V., M. Choquette and W. B. Croft, *Cross-lingual relevance models*, in: *SIGIR '02: Proceedingsof the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 175–182.

[44] Levow, G.-A., D. W. Oard and P. Resnik, *Dictionary-based techniques for cross-language information retrieval*, Information Processing and Management: an International Journal **41** (2005), pp. 523–547.

[45] Lew, M. S., N. Sebe, C. Djeraba and R. Jain, *Content-based multimedia information retrieval: State of the art and challenges*, ACM Trans. Multimedia Comput. Commun. Appl. **2** (2006), pp. 1–19.

[46] Li, X., D. Roth and K. Small, *The role of semantic information in learning question classifiers*, in: *The First International Joint Conference on Natural Language Processing*, 2004.

[47] Li, Y., Z. A. Bandar and D. McLean, *An approach for measuring semantic similarity between wordsusing multiple information sources*, IEEE Transactions on Knowledge and Data Engineering **15** (2003), pp. 871–882.

[48] Lin, X., D. Soergel and G. Marchionini, *A self-organizing semantic map for information retrieval*, in: *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval* (1991), pp. 262–269.

[49] Maybury, M. T., editor, “Intelligent Multimedia Information Retrieval,” AAAI Press, 1997.

[50] McCarley, J. S., *Should we translate the documents or the queries in cross-language information retrieval?*, in: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics* (1999), pp. 208–214.

[51] McNamee, P. and J. Mayfield, *Comparing cross-language query expansion techniques by degrading translation resources*, in: *SIGIR ’02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 159–166.

[52] Mukherjea, S., B. Bamba and P. Kankar, *Information retrieval and knowledge discovery utilizing a biomedical patent semantic web*, IEEE Transactions on Knowledge and Data Engineering **17** (2005), pp. 1099–1110.

[53] Naphide, H. and T. Huang, *A probabilistic framework for semantic video indexing, filtering, and retrieval*, IEEE Transactions on Multimedia **3** (2001), pp. 141–151.

[54] Natsev, A., R. Rastogi and K. Shim, *Walrus: a similarity retrieval algorithm for image databases*, IEEE Transactions on Knowledge and Data Engineering **16** (2004), pp. 301–316.

[55] Nie, J.-Y., M. Simard, P. Isabelle and R. Durand, *Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web*, in: *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), pp. 74–81.

[56] Oard, D. W., *A comparative study of query and document translation for cross-language information retrieval*, in: *AMTA '98: Proceedings of the Third Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup* (1998), pp. 472–483.

[57] Och, F. J. and H. Ney, *A systematic comparison of various statistical alignment models*, Computational Linguistics **29** (2003), pp. 19–51.

[58] Parsia and Sirin, "Cautiously approaching swrl," http://www.mindswap.

org/papers/CautiousSWRL.pdf, 2005.

[59] Picard, R. W., "Affective Computing," MIT Press, 2000.

[60] Pickens, J., J. P. Bello, G. Mont, M. Sandler, T. Crawford, M. Dovey and D. Byrd, *Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach*, Journal of New Music Research **32** (2003), pp. 223 – 236.

[61] "Protegè", http://protege.stanford.edu/, 2010.

[62] Ragno, R., C. J. C. Burges and C. Herley, *Inferring similarity between music objects with application to playlist generation*, in: *MIR ’05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval* (2005), pp. 73–80.

[63] Raphael, B., *Sir: A computer program for semantic information retrieval*, Technical report, Cambridge, MA, USA (1964).

[64] Resnik, P. and N. A. Smith, *The web as a parallel corpus*, Computational Linguistics **29** (2003), pp. 349–380.

[65] Rogati, M. and Y. Yang, *Multilingual information retrieval using open, transparent resources in clef 2003*, in: *CLEF 2003*, 2003, pp. 133–139.

[66] Rui, Y., T. Huang and S. Chang, *Image retrieval: current techniques, promising directions and open issues*, Journal of Visual Communication and Image Representation **10** (1999), pp. 39–62.

[67] Rui, Y., T. S. Huang, S. Mehrotra and M. Ortega, *A relevance feedback architecture for content-based multimedia information retrieval systems*, cbaivl **00** (1997), p. 82.

[68] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps, “*Generic*

semantics of feature diagrams,” Comput. Netw., vol. 51, no. 2, pp. 456–479, 2007.

[69] Sebe, N., M. S. Lew, X. Zhou, T. S. Huang and E. M. Bakker, *The state of the art in image and video retrieval*, in: *Image and Video Retrieval*, 2003.

[70] Shah, U., T. Finin and A. Joshi, *Information retrieval on the semantic web*, in: *CIKM ’02: Proceedings of the eleventh international conference on Information and knowledge management* (2002), pp. 461–468.

[71] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical owl-dl reasoner,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, no. 2, pp. 51–53, June 2007.

[72] Sivic, J. and A. Zisserman, *Video google: a text retrieval approach to object matching in videos*, in: *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[73] Smeulders, A. W. M., M. Worring, S. Santini, A. Gupta and R. Jain, *Content-based image retrieval at the end of the early years*, IEEE Trans. Pattern Anal. Mach. Intell. **22** (2000), pp. 1349–1380.

[74] Snoek, C. G., M. Worring, J.-M. Geusebroek, D. C. Koelma and F. J. Seinstra, *The mediamill TRECVID 2004 semantic video search engine*, in: *Proceedings of the 2th TRECVID Workshop*, 2004.

[75] Soo, V.-W., C.-Y. Lee, C.-C. Li, S. L. Chen and C. chih Chen, *Automated semantic annotation and retrieval based on sharable ontology and case-based learning techniques*, in: *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries* (2003), pp. 61–72.

[76] Systran, *Systran language translation technology*, [Online], http://www.systransoft.com.

[77] Tang, C., Z. Xu and S. Dwarkadas, *Peer-to-peer information retrieval using self-organizing semantic overlay networks*, in: *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (2003), pp. 175–186.

[78] Tieu, K. and P. Viola, *Boosting image retrieval*, International Journal of Computer Vision **56** (2004), pp. 17–36.

[79] Uitdenbogerd, A. L. and J. Zobel, *An architecture for effective music information retrieval*, Journal of the American Society for Information Science and Technology **55** (2004), pp. 1053–1057.

[80] Utsuro, T., K. Hino, M. Kida, S. Nakagawa and S. Sato, *Integrating cross-lingually relevant news articles and monolingual web documents in bilingual lexicon acquisition*, in: *Proceedings of Coling 2004* (2004), pp. 1036–1042.

[81] Varelas, G., E. Voutsakis, P. Raftopoulou, E. G. Petrakis and E. E. Milios, *Semantic similarity methods in wordnet and their application to information retrieval on the web*, in: *WIDM ’05: Proceedings of the 7th annual ACM international workshop on Web information and data management* (2005), pp. 10–16.

[82] Wactlar, H. D., M. G. Christel, Y. Gong and A. G. Hauptmann, *Lessons learned from building a terabyte digital video library*, Computer **32** (1999), pp. 66–73.

[83] H. Wang, Y. F. Li, J. Sun, H. Zhang, and J. Pan, “*A semantic web approach to feature modeling and verification*” In Workshop on Semantic Web Enabled Software Engineering (SWESE’05, 2005).

[84] Chia-Hung Wei, Chang-Tsun Li, “*Content-Based Multimedia Retrieval”*

[85] Yan, R. and A. Hauptman, *Co-retrieval: a boosted reranking approach for video retrieval*, Vision, Image and Signal Processing **152** (2005), pp. 888– 895.

[86] Yan, R., A. G. Hauptmann and R. Jin, *Negative pseudo-relevance feedback in content-based video retrieval*, in: *MULTIMEDIA ’03: Proceedings of the eleventh ACM international conference on Multimedia* (2003), pp. 343–346.

[87] Yu, H., T. Mine and M. Amamiya, *An architecture for personal semantic web information retrieval system–integrating web services and web contents*, in: *IEEE International Conference on Web Services*, 2005, pp. 329–336.

[88] L. A. Zaid, G. Houben, O. D. Troyer, and F. Kleinermann, "*An owl-based approach for integration in collaborative feature modelling*," in SWESE 2008, 4th Workshop on Semantic Web Enabled Software Engineering, workshop at ISWC 2008, the International Semantic Web Conference 2008, Karslruhe, Germany, Oct. 2008, pp. 93–100.

[89] Zhang, H., A. Kankanhalli and S. W. Smoliar, *Automatic partitioning of full-motion video*, Multimedia Syst. **1** (1993), pp. 10–28.

[90] Zhang, Y. and P. Vines, *Using the web for automated translation extraction in cross-language information retrieval*, in: *SIGIR '04:* Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (2004), pp. 162–169.

[91] Zhang,Y., Xu, C., Zhang, X., Lu, X., *Personalized retrieval of sports video based on multi-modal analysis and user preference acquisition*, Multimedial Tools and Applications, vol. 44, pp. 305-33, 2009.