Multimedia Queries in Digital Libraries

Ilaria Bartolini and Marco Patella

Abstract The intrinsic complexity and diversity of data in Multimedia Digital Libraries (MDLs) require devising techniques and solutions that are inherently different from those usually adopted in traditional Information Retrieval and Database systems. Moreover, the size and the dynamicity of MDLs force researchers to strive for efficiency, so as to guarantee real time results to the users. Finally, semantics should be also brought into context, in order to facilitate users' experience in querying, browsing, and consuming multimedia information. This chapter will present an approach towards the efficient, effective, and semantically rich data retrieval in MDLs. With respect to the commonly used holistic approach, where the multimedia datum is considered as an atomic entity, our reductionist strategy considers the multimedia information as a complex combination of component subparts, and eases the fulfillment of the three above properties of efficiency, effectiveness, and semantic richness. Indeed, by decomposing multimedia information into simpler and smaller component objects, we are able to index such components without giving up the ability of querying the original information as a whole.

1 Peculiarities of Querying Multimedia Data

As already described in [1], multimedia queries are utterly different with respect to those commonly used in traditional information retrieval and/or database systems. Indeed, the diverse content and complex structure of multimedia documents makes traditional (boolean) queries hardly effective. As an example, in a digital image

Ilaria Bartolini
Department of Computer Science and Engineering – DISI, University of Bologna.
e-mail: i.bartolini@unibo.it

Marco Patella
Department of Computer Science and Engineering – DISI, University of Bologna.
e-mail: marco.patella@unibo.it

library we may want to retrieve pictures which are similar, in content, to a user-provided image: The result we expect from the system is probably a set of library images, sorted for decreasing values of visual similarity with respect to the query; this is very different to the result of a regular SELECT query.

The processing of queries in multimedia digital libraries should also take into account the distinctive features of multimedia data, so that one should consider aspects of:

- Efficiency, due to the size and number of multimedia data which exceed those of textual data by several orders of magnitude.
- Effectiveness, since the representation of the information "local" to the system may differ from what the user issuing the query has in mind.
- Semantics, due to the fact that "concepts" can be attached (implicitly or as metadata) to each multimedia datum.
- Dynamicity, because digital content is continuously added to/removed from the library.
- Complexity, since a multimedia document is usually composed of several parts, each of a particular medium (e.g., a web page contains text, images, videos, and so on).

The usual approach pursued by digital libraries is to represent information about documents as metadata, that are harvested when each single document enters the system. Then, such metadata are "indexed" to allow the retrieval of query results: This can be carried out by exploiting inverted files. In this way, the digital library is not accessed during the search, making the whole process very efficient. The above approach, however, fails to address the problems of effectiveness, since the metadata harvesting process is hardly tailored to the needs of a specific user, and complexity, mainly because the library datum is commonly seen as an atomic entity, which cannot be divided into sub-parts.

The typical query paradigms that are offered by multimedia digital libraries are of two types:

Boolean search: The user specifies a (complex) boolean predicate on metadata and only those documents whose metadata satisfy the predicate are returned. The result of such a query is a (unordered) set of objects.

Ranked search: The user provides a multimedia object, and library data are returned in descending order of similarity with respect to the query. The result of this query is a (ordered) list of objects. The user has three ways of limiting the result size:

  Range query: The user specifies a minimum similarity threshold θ, so that objects whose similarity with the query is lower than θ are not returned (note that this requires a knowledge of the distribution of similarity values between objects).

  k-NN query: The user provides a maximum number of results k, and only the k objects more similar to the query are returned.

Sorted access: Library data are returned to the user in descending order of similarity with respect to the query, until the user is satisfied with the result (similar to a common iterator).

If one wants to support more complex query types, the main challenge is how the system can assess whether a datum is relevant for the query or not.

Consider, for example, a digital library offering access to documentaries. Every movie also contains metadata like year of production, title, name of the director, etc. The system provides the users visiting the library with two different search paths:

1. A search-by-content paradigm, where the user can provide a sample video and library movies are returned in descending order of visual similarity with respect to the provided video.
2. A search-by-metadata paradigm, expressed by means of a boolean predicate on metadata, e.g., director = 'Riefenstahl' AND year BETWEEN 1920 AND 1929.

What if the user wants only the documentaries shot by Leni Riefenstahl during the 1920s, but sorted for visual similarity with respect to a provided video? How can such a query be processed by the system? Should a documentary with very similar content but very different metadata be included in the result? In case how such documentary should be ranked with respect to another movie with different visual content but satisfying the boolean predicate?

In the following we will discuss how modern systems for the management of digital libraries can be extended to tackle the above described problems. First, we present a model able to capture the complexity and diversity of multimedia documents and show how such model can also support efficient query resolution (Section 2). Then, we face the problem of enriching multimedia documents with semantic annotations (Section 3). It is widely known, in fact, that content-based multimedia retrieval using low-level features only does not provide sufficiently accurate results. This is due to the so-called *semantic gap* existing between the user subjective notion of similarity and the one implemented by the system. The use of semantic annotations, therefore, allows querying digital libraries using high-level semantic concepts, helping to bridge the semantic gap.

2 The Windsurf Model

In the Windsurf model [9][1], each multimedia document D is composed of n_D elements, $D = \{R_1, \ldots, R_{n_D}\}$. Each element can be recursively defined in terms of

[1] www-db.disi.unibo.it/Windsurf/

other components and so on. Simple (base) elements are finally described by way of metadata, including features that allow the comparison of elements. Note that the subdivision of a multimedia document into its component parts can be either explicit (i.e., specified in the very same description of the multimedia document, like in a web page) or implicit. In the latter case, it is the system that automatically breaks the multimedia object into component parts; examples of this automatic subdivision include:

- a video segmented into scenes of homogeneous visual content [8],
- an image divided into regions of pixels sharing common color and/or texture [2, 9]
- an image represented by way of its salient points [19, 10].

When compared to a query objects Q, also composed of query elements $Q = \{Q_1, \ldots, Q_n\}$, the components of D are compared to those of Q by way of a specific similarity function and the overall similarity $s(Q,D)$ (recursively) depends on the similarity between components. This modular approach, which we borrowed from the PANDA model [5] for describing mining patterns, allows for a great generality and flexibility. Indeed, although, for comparing complex multimedia documents one could devise arbitrary models, it is useful and, at the same time, sufficient for practical purposes, to consider solutions that decompose the "difficult" problem of comparing complex documents into simpler subproblems like those of comparing simple components, and then "smartly" aggregate the so-obtained partial solutions into an overall similarity value. Comparing complex documents is obtained by combining two basic elements:

1. a matching type, which is used to establish how the component elements of the two multimedia documents can be matched, and
2. the aggregation logic, which is used to combine the similarity scores of the matched component elements into a single value representing the total similarity between the complex documents.

The matching type specifies the constraints that should be used when comparing components: For example, it is obvious that only components of the same type are to be compared (i.e., one cannot compare a video with a text); otherwise one can specify that only one-to-one matchings are allowed (e.g., a same image in a multimedia document cannot be matched to two different images in another document). The aggregation logic, on the other hand, states how scores of the matched component pairs should be aggregated so as to yield the overall similarity score. We note here that, among all the valid matchings (as specified by the matching type), the rationale is to pick the "best" one, i.e., the one that maximizes the aggregated similarity [5].

The choice of which matching type and aggregation logic are better suited for the query at hand can be left to the user or assumed by default, with the understanding that this choice has a clear impact on both the effectiveness and the efficiency of the result.

We finally note that an appropriate choice of matching type and/or aggregation logic can also support other query types, like those denoted in [6] as *partial*, *zoom-in*, *containment*, and *part-of* queries, i.e., queries where only a fraction of components of the query and/or of the library document are considered.

The flexibility and generality of the proposed model are indeed required to deal with the possible complexity of queries in multimedia digital libraries.

As another example, we consider a digital library storing web pages about music composers. Every page includes a textual part containing structured data, like the composer name, his nationality and date of birth, and so on, and an unstructured part containing other documents, like pictures, other texts, and audio fragments, all of them possibly tagged with metadata.

An user might be interested in all documents related to German composers of the 19th century that contain a picture sufficiently similar to a user-specified image and an audio fragment at least 30 seconds long. Such a query contains both predicates on metadata (nationality, date of birth, length of the audio fragment) and similarity predicates (similar image). Moreover, some of the predicates are related to the whole document (nationality, date of birth) while others only concern component parts (length of the audio fragment, similar image). It is clear that with a reductionist approach, that considers a document as composed by parts, the complexity of such query is easily dealt with, as opposed to an holistic strategy that considers the multimedia datum as an atomic entity.

2.1 Efficient Processing of Similarity Queries

Having defined the ingredients needed to assess the similarity between two multimedia objects, it is immediate to derive a sequential algorithm able to retrieve the best results for a given query. Clearly, any sequential algorithm will incur prohibitively high costs even for moderately large digital libraries. In order to guarantee scalability, the query processor should exploit the presence of indices able to efficiently retrieve relevant results. Our arguments will be developed independently of the specific index; rather, we will refer to a generic distance-based index, i.e., any index that relies on the computation of distances to return back objects. Distance-based indices include both multi-dimensional [15] and metric [11] indices, relevant examples of which are the R-tree [16] and the M-tree [12], respectively.

The resolution of a similarity query is depicted in Fig. 1. After the query document has been decomposed in its component parts, each query part is processed separately by a distance-based index using a sorted access. The results of each index, a list of objects ranked for decreasing similarity with respect to the input query component, is then fed to the middleware algorithm implementing the chosen matching/aggregation strategy. With the help of a *random access* on the database of

documents' components, the middleware algorithm is able to compute the similarity value for all those objects for which at least a component has been returned by at least an index scan. As soon as the algorithm is able to assess that yet-unseen documents cannot lead to a similarity score which is higher than the "best" document seen so far, then such document can be returned to the user, and so on for all subsequent documents in the list. The above can be guaranteed [6, 14] if the aggregation logic is monotone, i.e., a lower similarity score for any component cannot increase the overall similarity score, a property which is sound and is shared by all the most used aggregation logics.

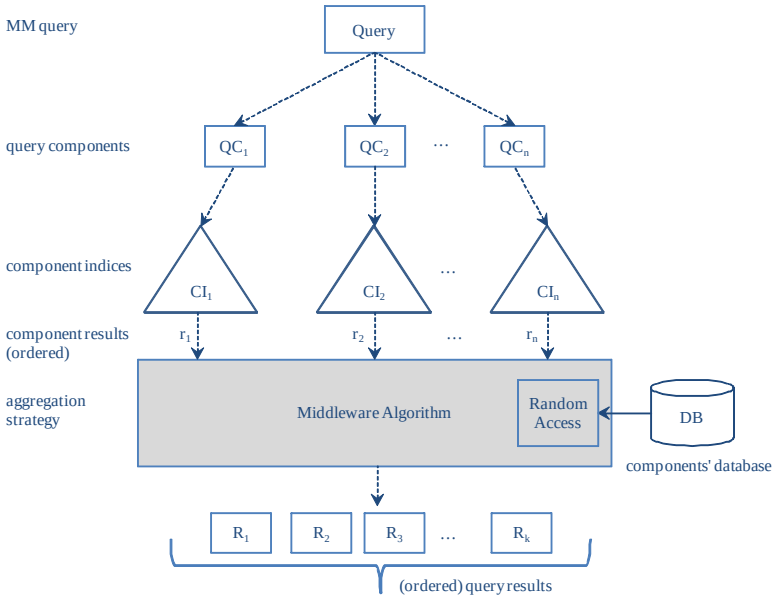

Fig. 1 Processing a similarity query: The middleware algorithm combines, through the matching/aggregation strategy, sorted accesses to the component indices and random accesses to the component database. The result is a sorted list of library documents, ranked for decreasing values of similarity to the query.

When considered as a whole, Fig. 1 can be seen as a (complex) index which is able to support sorted access, since results are provided in a ranked order. Therefore, this can be recursively used to obtain the arbitrary complexity required by the model: Distance-based indices are used on simple (base) components, while the complex index depicted in Fig. 1 is used on complex multimedia objects.

2.2 Processing of Mixed Queries

Although the strategy described in the previous section allows us to efficiently process similarity queries, it does not help when mixed queries, i.e., queries combining metadata and similarity predicates, are issued. For such queries, we have to somehow combine the result of a similarity query (an ordered list of objects) with that of a metadata query (a set of objects). The possibilities here are those listed in [4]: Join with order, union with order, and difference with order; we focus here on the most complex operator, join with order, since the implementation of other operators does not differ much from those commonly used for relational DBs.

To exemplify our arguments, we use again the example on documentaries. The metadata predicate (director = 'Riefenstahl' AND year BETWEEN 1920 AND 1929) would return a set of documentaries satisfying both boolean conditions, while for the similarity query three possibilities exist:

1. If the similarity predicate is a range query, the user has also specified a minimum similarity threshold θ; the similarity predicate is now analogous to a boolean predicate, $s(Q,D) \geq \theta$. The query result should therefore include only those documentaries that satisfy all three conditions, sorted for decreasing values of similarity.
2. If the similarity predicate is a k-NN query, a first possibility is to consider the maximum number of results k specified by the user as an a-priori condition; the similarity predicate is again analogous to a boolean predicate, which is satisfied only by the k movies most similar to the provided one. Again, the query result includes only those documentaries that satisfy all three conditions, sorted for decreasing values of similarity (note that this will retrieve a number of results which is not higher than k, but can also produce an empty result if the k best movies do not satisfy the metadata predicate).
3. Finally, we can consider k as an a-posteriori condition, i.e., the user wants the k documentaries shot by Leni Riefenstahl during the 1920s ranked for visual similarity with respect to the provided video (assuming that k of such objects exist).

Fig. 2 depicts the components of the mixed query processor: The two query predicates (MQC, multimedia query component, and SQC, semantic query component) are fed to the relative indices (MI, multimedia/similarity index, and SI, semantic/metadata index); the so-obtained results have to be combined by the matcher to produce the result for the overall query.

It is obvious that processing cases 1. and 2. above is quite simple, since one can compute the intersection of results provided by the similarity index and by the semantic index. For case 3. we have two alternative strategies:

1. The first strategy retrieves all documentaries satisfying the metadata predicate (by using SI) and sorts them by way of the similarity predicate. Then, only the first k results are returned to the user.

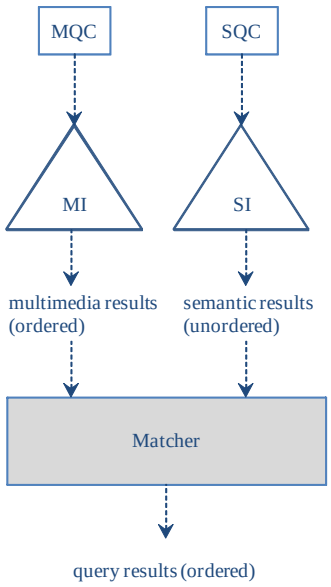

Fig. 2 Processing a mixed query: The results obtained by the similarity index (MI) and the semantic index (SI) are combined by the matcher to produce the result.

2. As an alternative, we can perform a sorted access on MI and check, for all retrieved documentaries, the metadata predicate, returning objects satisfying it until k results have been output.

Above strategies provide the same result, but have different efficiency. In particular, this heavily depends on the selectivity f of the metadata predicate and on the cost of evaluating the similarity predicate. Since we will assume that the latter is always (much) higher than evaluating the metadata predicate, due to the complexity of comparing multimedia features [1], for high selective metadata predicates, the first strategy is to be preferred: In this case only a few documents ($f \times N$, where N is the number of documents in the library) are compared against the query. On the other hand, for low selective predicates, the second strategy attains the best efficiency, because only k/f sorted accesses are required to solve the query (every retrieved document has a probability f to satisfy the metadata predicate). We conclude this section by highlighting the analogy with the optimizer component of a DBMS, that has to choose the optimal query plan according to the (estimated) selectivity of query predicates.

3 Semantic Enrichment of Multimedia Data

Although content-based techniques presented above, possibly assisted with user relevance feedback from the user [23, 7], can indeed attain a very good effectiveness, in several cases they still stay below the optimal 100% precision value, in particular

when the user is looking for documents matching some high-level concept, which is hardly representable by means of low-level features only. In such cases, a possible way to fill the semantic gap is to assign meaningful terms to documents, so as to indeed allow a high-level, concept-based, retrieval.

Terms associated to documents can, indeed, be considered as a different type of metadata, one which is not included in the multimedia datum, but which is associated with it by way of a manual or automatic process. Clearly, such terms can be seamlessly used into predicates for boolean search, as traditional metadata are.

For instance, assuming that the two images in Fig. 3 are labeled as shown, it would be possible to discriminate among them if, say, one is looking for horses and, at the same time, to consider both relevant if one is looking for mammals on grass.

`horse,grass`

`buffalo,grass`

Fig. 3 Two images with associated tags.

Associating semantic labels to multimedia objects is usually performed in one of the following ways:

Tagging by an expert: This is the solution commonly used in libraries, where an expert provides labels for every datum. Such labels are expected to be of great quality, but the process is lengthy and expensive [1]; moreover, the problem of subjectivity can plague the whole data collection, because the user searching the library can have a different view on the data with respect to the user providing labels.

Social tagging: Using this approach, which is the one exploited by "social" libraries, like YouTube[2] and Flickr[3], users accessing a multimedia document can also provide labels for it. The so-obtained annotation is usually of low quality, due to problems of ambiguity (because a label could carry different meanings due to polysemy or homonymy), lack of information (because a document could never have been labeled), and problems of synonymy/mistyping.

Automatic tagging: These techniques exploit the similarity among multimedia documents (computed by way of low-level features) to extract labels relevant for a non-annotated object, with the assumption that objects sharing similar features also convey the same semantic content, and can thus be tagged with the same labels.

[2] `www.youtube.com`

[3] `www.flickr.com`

Several techniques for semi-automatic annotation of multimedia objects [3, 8] have been proposed in recent years and the first prototypes for annotation are now available on the Internet (e.g., ALIPR[4] and Behold[5] for images). We can group state-of-the-art solutions into two main classes, namely semantic propagation and statistical inference. In both cases, the problem to be solved is the same: Given a training set of annotated multimedia objects, discover affinities between low-level features and terms that describe the object content, with the aim of predicting "good" terms to annotate a new document. With propagation models [20], a supervised learning technique that compares content similarity at a low-level and then annotates objects by propagating terms over the most similar objects is adopted. Working with statistical inference models [22], an unsupervised learning approach tries to capture correspondences between low-level features and terms by estimating their joint probability distribution. Both approaches improve the annotation process and the retrieval on large multimedia libraries.

The typical approach for annotating multimedia objects exploits user-defined textual labels [3, 18, 24]. However, this is commonly performed by drawing tags from a unstructured set, thus not taking into account their intended meaning (i.e., the meaning such tags convey in the context where their associated multimedia data are found: This gives a sort of meaning vagueness to labels [21]). To deal with this, in order to connect each tag with its intended meaning, the coexistence of multiple, independent classification criteria [13] can be exploited. According to this "multi-dimensional" approach, labels belonging to different dimensions may have separate meanings, while each dimension will represent the meaning of high-level concepts contained therein, providing a disambiguation of their semantics.[6] Moreover, each dimension takes the shape of a tree, where each concept is represented within a taxonomy node and terms are linked with a parent/child relationship. More precisely, each concept is denoted as a *semantic tag*, represented as a path in a tree. To each tree node is therefore associated a single label; the label of the root node corresponds to the name of the dimension itself.

The above model of hierarchical faceted categories has been successfully used in a variety of applications to provide a coherent and complete description of data [17, 8]. For instance, Fig. 4 shows some dimensions for a real-world scenario: These include "animal", "landscape", "geographic location", and so on. Each node in a tree path corresponds to a more specialized concept with respect to its parent node, so that moving up/down within a tree a user encounters more/less abstract semantic concepts. This means that if a document is tagged with a given semantic concept t, it is also associated to all ancestors of t. In the example of Fig. 4, the label "landscape/sky/rainbow" also includes the semantic concept "landscape/sky".

We finally note that, in line of principle, a same label could appear in several different trees: This allows to distinguish between the different uses and/or

[4] `www.alipr.com`

[5] `www.behold.cc`

[6] Such dimensions are therefore quite different with respect to those used in other semantic indexing approaches, like bag-of-words or Latent Semantic Indexing (LSI), where each dimension corresponds to a single, basic concept.

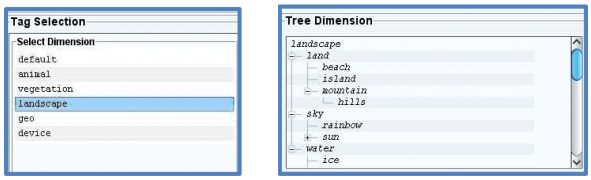

Fig. 4 A 6-dimensional scenario: The "landscape" dimension is selected on the left and (part of) its concept tree is shown on the right-hand side.

meanings that different occurrences of a same label could convey. For example, it is possible that the label "turkey" will appear as a node within both the dimension "geographic location" (e.g., used to describe documents according to the location they were created) and the dimension "animal", associated to documents about the gallinaceous bird. Extending the example, we could conceive the existence of a "sports" dimension (associated to documents related to sport events), where the same "turkey" label could appear in several places, for example, "sports/soccer/turkey" and "sports/basketball/turkey", representing, respectively, the soccer and basketball Turkish national teams. Clearly the fact that each semantic tag corresponds to a single complete path within a tree allows the disambiguation of the different uses of a same label, as illustrated by the previous example.

In order to ensure compatibility with unstructured ("flat") dimensions, such as those used in systems like YouTube or Flickr, we can support 2-level taxonomies, with all tags appearing as children of the single tree root node (in Fig. 4, this is represented by the "default" dimension). This fact allows us to also exploit another technique to solve label ambiguity, i.e., label co-occurrence. On the other hand, label co-occurrence is not able, alone, to solve problems of homonymy/polysemy. For example, suppose the user wants to retrieve documents about the animals of the Eurasian country of Turkey: It is quite likely that querying the system using the flat concepts "animal" and "turkey" would primarily return documents concerning the gallinaceous bird, due to the polysemy of the term "turkey". The system is therefore not able to satisfy this particular information need of the user by using label co-occurence only.

According to the presented model, each document can be assigned a variable number of semantic tags. If a dimension is not relevant for a document, then no semantic tag from such dimension is used to characterize content. On the other hand, a document could be characterized by multiple semantic tags from the same dimension, if this is appropriate. For instance, an image depicting a dog and a cat might be assigned the two semantic tags "animal/dog" and "animal/cat", both from the "animal" dimension. Thus, although each dimension provides a means to classify documents, this classification is not exclusive at the instance level, a fact that provides the necessary flexibility to organize documents.

3.1 Efficient Annotation of Complex Multimedia Documents

For tagging purposes, we advocate the technique exploited in the Imagination system [3], using a set of pre-annotated documents as a knowledge base. Such documents are used by the system as an example of the semantic concepts attached to them; in this way, all and only concepts included in the knowledge base could possibly be suggested as relevant for a given (non-labeled) document. When provided with a document to be labeled, the systems retrieves (by way of the multimedia index MI) documents having a similar content and proposes semantic concepts depending on the similarity of the document with the documents in the knowledge base. The use of MI is clearly the key to attain efficiency in the annotation process. Every time a new document is processed and tagged, its information is also inserted into the semantic index SI, hence improving the system accuracy and quality.

Although the above algorithm has been proved effective for simple multimedia objects, e.g., for images, its accuracy on complex documents is questionable. Indeed, the model presented in Section 2 allows for an arbitrary complexity for a document D that can have components, which have content similar to components of other documents, but still is not very similar to any of the other documents in the library. Exploiting the above technique on the document D is likely to lead to a imprecise labeling of D. To overcome this problem for complex documents, we propose the use of hierarchical tagging [8].

The idea at the base of *hierarchical tagging* is quite simple: Labels associated to components of D are propagated to D using a frequency analysis (only most frequent labels are propagated to D). Propagating tags from components to documents is therefore an activity of summarization, i.e., the description of the document is a compact sum of the tags associated with its components.

Given a document D and a semantic tag t, the relevance of t for D is computed by combining the relevance of all components s in D (e.g., this could be the relative length of a scene in a video or of an audio track in a compilation, or the size of an object in an image) and the relevance of t in s, which can vary from 0 to 1. More precisely, we denote the relevance of component s for D as $W(s,D)$ and the relevance of tag t in s as $A(t,s)$. Note that, in the default case, it is $W(s,D) = 1/n_D$, where n_D is the number of components of D, and $A(t,s) = 1$ for all tags t in s, otherwise $A(t,s) = 0$. Then, the relevance of t in D can be defined as $R(t,D) = \sum_s A(t,s) \times W(s,D)$. Semantic tags can then be ranked for decreasing values of $R(t,D)$ and only the more relevant tags are selected for D. The same process can then be recursively applied upward in the hierarchy of multimedia documents.

As an example, we consider again the case of documentaries. Each video is (automatically or manually) divided into scenes and a number of keyframes are automatically extracted from each scene, representing its visual content. Assume that each scene contained in the library has been annotated (either manually or automatically) and that a new documentary D is inserted in the library. D is then segmented into scenes and the system extracts, for each scene,

the relevant keyframes. Each keyframe f is then used to query the multimedia index and, say, the k most similar keyframes in the library are retrieved. The set of tags for f is then built using the tags of the scenes represented by the keyframes returned by the index.

Labels automatically suggested for frames are then propagated at the scene level: Here, $W(s,D)$ and $A(t,s)$ assume the default values of $1/n_D$ (with n_D denoting the number of keyframes representing a given scene) and 0/1. Finally, scene labels are summarized into tags for the whole video D, with $W(s,D)$ equal to the relative length of scene s with respect to video D while $A(t,s)$ is obtained from the previous step. The whole process is depicted in Fig. 5, where more relevant labels are shown in boldface, for the sake of clarity.

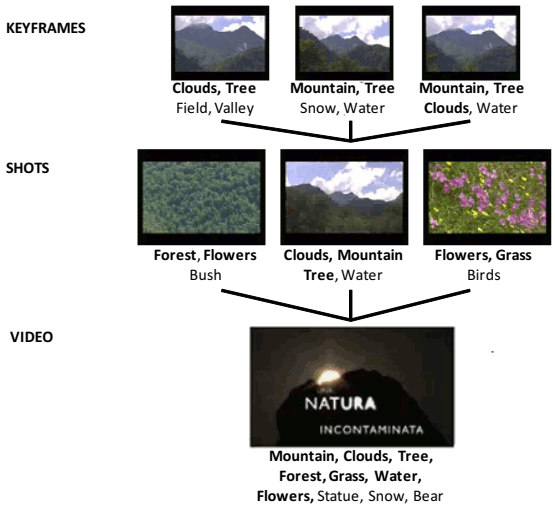

Fig. 5 Hierarchical annotation: Labels suggested for keyframes are summarized into scene and video tags.

References

1. Flora Amato, Luca Greco, and Fabio Persia. Content-Based Multimedia Information Retrieval - Chapter 14 in this book. Springer, Berlin, 2015.
2. Stefania Ardizzoni, Ilaria Bartolini, and Marco Patella. Windsurf: Region-Based Image Retrieval Using Wavelets. In 1st Int'l Workshop on Similarity Search (IWOSS 1999 - DEXA 1999), pages 167–173, Florence, Italy, 1999.

3. Ilaria Bartolini and Paolo Ciaccia. Imagination: Exploiting Link Analysis for Accurate Image Annotation. In 5th Int'l Workshop on Adaptive Multimedia Retrieval (AMR 2007), (Lecture Notes in Computer Science), vol. 4918/2008, pages 32–44, Paris, France, 2007.
4. Ilaria Bartolini, Paolo Ciaccia, Lei Chen, and Vincent Oria. A Meta-Index to Integrate Specific Indexes: Application to Multimedia. In 12th Int'l Conf. on Distributed Multimedia Systems (DMS 2006), pages 29–36, Grand Canyon, AZ, 2006.
5. Ilaria Bartolini, Paolo Ciaccia, Irene Ntoutsi, Marco Patella, and Yannis Theodoridis. The Panda Framework for Comparing Patterns. In Data and Knowledge Engineering, 68(2), pages 244–260, 2009.
6. Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. Query Processing Issues in Region-Based Image Databases. In Knowledge and Information Systems (KAIS), 25(2), pages 389–420, 2010.
7. Ilaria Bartolini, Paolo Ciaccia, and Florian Waas. FeedbackBypass: A New Approach to Interactive Similarity Query Processing. In 27th Int'l Conf. on Very Large Data Bases (VLDB 2001), pages 201–210, Rome, Italy, 2001.
8. Ilaria Bartolini, Marco Patella, and Corrado Romani. SHIATSU: Tagging and Retrieving Videos without Worries. In Multimedia Tools and Applications Journal, 63(2), pages 357–385, 2013.
9. Ilaria Bartolini, Marco Patella, and Guido Stromei. Efficiently Managing Multimedia Hierarchical Data with the Windsurf Library. In Communications in Computer and Information Science (Lecture Notes in Computer Science), vol. 314/2012, pages 347–361, 2012.
10. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Computer Vision and Image Understanding (CVIU), 110(3), pages 346-359, 2008.
11. Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Proximity Searching in Metric Spaces. In ACM Computing Surveys, 33(3), pages 273-321, 2001.
12. Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In 23rd Int'l Conf. on Very Large Data Bases (VLDB'97), pages 426-435, Athens, Greece, 1997.
13. Ronald Fagin, R. Guha, Ravi Kumar, Jasmine Novak, D. Sivakumar, and Andrew Tomkins. Multi-Structural Databases. In 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 184-195, Baltimore, MD, 2005.
14. Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal Aggregation Algorithms for Middleware. In 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 102–113, Santa Barbara, CA, 2001.
15. Volker Gaede and Oliver Günther. Multidimensional Access Methods. In ACM Computing Surveys, 30(2), pages 170-231, 1998.
16. Antonin Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In 984 ACM SIGMOD Int'l Conf. on Management of Data, pages 47-57, Boston, MA, 1984.
17. Marti A. Hearst. Clustering versus Faceted Categories for Information Exploration. In Communication of the ACM, 49(4), pages 59-61, 2006.
18. Jim Kleban, Emily Moxley, Jiejun Xu, B.S. Manjunath. Global Annotation of Georeferenced Photographs. In 8th ACM Int'l Conf. on Image and Video Retrieval (CIVR 2009), article 12, Santorini Island, Greece, 2009.
19. David G. Lowe. Object Recognition from Local Scale-Invariant Features. In 7th IEEE Int'l Conf. on Computer Vision. Vol. 2, pages 1150-1157, Kerkyra, Greece, 1999.
20. Oded Maron and Aparna Lakshmi Ratan. Multiple-Instance Learning for Natural Scene Classification. In 15th Int'l Conf. on Machine Learning (ICML 1998), pages 341–349, San Francisco, CA, USA, 1998.
21. Roberto Navigli. Word Sense Disambiguation: A Survey. In ACM Computing Surveys, 41(2), article 10, 2009.
22. Jia-Yu Pan, HyungJeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic Multimedia Cross-Modal Correlation Discovery. In 10th ACM SIGKD Int'l Conf. on Knowledge Discovery and Data Mining, pages 653–658, Seattle, WA, 2004.

23. Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrotra. Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval. In IEEE Transactions on Circuits and Systems for Video Technology, 8(5), pages 644–655, 1998.
24. Lei Wang and Latifur Khan. Automatic Image Annotation and Retrieval using Weighted Feature Selection. In Multimedia Tools and Applications, 29(1), pages 55-71, 2006.