Article

The VISIONE Video Search System: Exploiting Off-the-Shelf Text Search Engines for Large-Scale Video Retrieval

Giuseppe Amato [1], Paolo Bolettieri [1], Fabio Carrara [1], Franca Debole [1], Fabrizio Falchi [1], Claudio Gennaro [1], Lucia Vadicamo [1,*] and Claudio Vairo [1]

[1] Institute of Information Science and Technologies (ISTI), Italian National Research Council (CNR), Via G. Moruzzi 1, 56124, Pisa, Italy; firstname.lastname@isti.cnr.it

* Correspondence: lucia.vadicamo@isti.cnr.it

Abstract: This paper describes in detail VISIONE, a video search system that allows users to search for videos using textual keywords, the occurrence of objects and their spatial relationships, the occurrence of colors and their spatial relationships, and image similarity. These modalities can be combined together to express complex queries and meet users' needs. The peculiarity of our approach is that we encode all information extracted from the keyframes, such as visual deep features, tags, color and object locations, using a convenient textual encoding that is indexed in a single text retrieval engine. This offers great flexibility when results corresponding to various parts of the query (visual, text and locations) need to be merged. In addition, we report an extensive analysis of the retrieval performance of the system, using the query logs generated during the Video Browser Showdown (VBS) 2019 competition. This allowed us to fine-tune the system by choosing the optimal parameters and strategies from those we tested.

Keywords: Content-based Video Retrieval; Surrogate Text Representation; Known Item Search; Ad-hoc Video Search; Multimedia and multimodal retrieval; Multimedia information systems; Information systems applications; Video search; Image search; Users and interactive retrieval; Retrieval models and ranking; Users and interactive retrieval.

Citation: Amato G.; Bolettieri P.; Carrara F.; Debole F.; Falchi F.; Gennaro c.; Vadicamo L.; Vairo C. The VISIONE Video Search System. *J. Imaging* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

With the pervasive use of digital cameras and social media platforms, we witness a massive daily production of multimedia content, especially videos and photos. This phenomenon poses several challenges for the management and retrieval of visual archives. On one hand, the use of content-based retrieval systems and automatic data analysis is crucial to deal with visual data that typically are poorly-annotated (think for instance of user-generated content). On the other hand, there is an increasing need for scalable systems and algorithms to handle ever-larger collections of data.

In this work, we present a video search system, named VISIONE, which provides users with various functionalities to easily search for targeted videos. It relies on artificial intelligence techniques to automatically analyze and annotate visual content and employs an efficient and scalable search engine to index and search for videos. A demo of VISIONE running on the V3C1 dataset, described in the following, is publicly available at http://visione.isti.cnr.it/.

VISIONE participated in the Video Browser Showdown (VBS) 2019 challenge [1]. VBS is an international video search competition [1–3] that evaluates the performance of interactive video retrieval systems. Performed annually since 2012, it is becoming increasingly challenging as its video archive grows and new query tasks are introduced in the competition. The V3C1 dataset [4] used in the competition since 2019 consists of 7,475 videos gathered from the web, for a total of about 1,000 hours. The V3C1 dataset is segmented into 1,082,657 non-overlapping video segments, based on the visual content of the videos [4]. The shot segmentation for each video as well as the keyframes

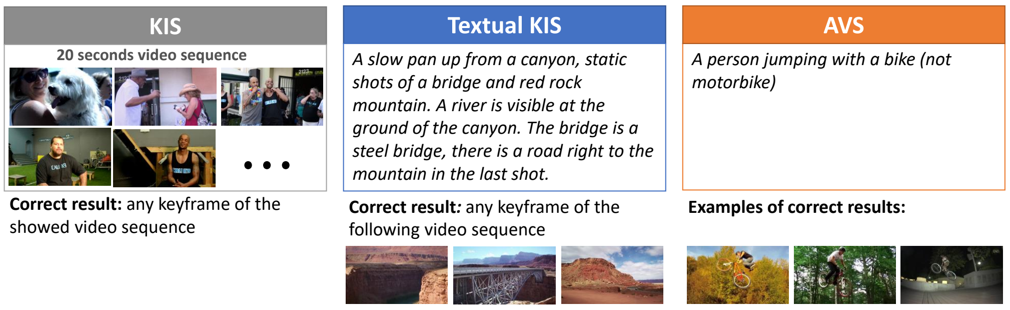

KIS
20 seconds video sequence
Correct result: any keyframe of the showed video sequence
Textual KIS
A slow pan up from a canyon, static shots of a bridge and red rock mountain. A river is visible at the ground of the canyon. The bridge is a steel bridge, there is a road right to the mountain in the last shot.
Correct result: any keyframe of the following video sequence
AVS
A person jumping with a bike (not motorbike)
Examples of correct results:

Figure 1. Examples of KIS, textual KIS and AVS tasks.

and thumbnails per video segment are available within the dataset[1]. In our work, we used the video segmentation and the keyframes provided with the V3C1 dataset. The tasks evaluated during the competition are: *Known-Item-Search (KIS)*, *textual KIS* and *Ad-hoc Video Search (AVS)*. Figure 1 gives an example of each task. The KIS task models the situation in which a user wants to find a particular video clip that he or she has previously seen, assuming that it is contained in a specific data collection. The textual KIS is a variation of the KIS task, where the target video clip is no longer visually presented to the participants of the challenge, but it is rather described in detail by some text. This task simulates situations in which a user wants to find a particular video clip, without having seen it before, but knowing exactly its content. For the AVS task, instead, a general textual description is provided and participants have to find as many correct examples as possible, i.e. video shots that match the given description.

VISIONE can be used to solve both It integrates several content-based data analysis and retrieval modules, including a keyword search, a spatial object-based search, a spatial color-based search, and a visual similarity search. The main novelty of our system is that it employs text encodings that we specifically designed for indexing and searching video content. This aspect of our system is crucial: we can exploit the latest text search engine technologies, which nowadays are characterized by high efficiency and scalability, without the need to define a dedicated data structure or even worry about implementation issues like software maintenance or updates to new hardware technologies, etc.

In [5] we initially introduced VISIONE by only listing its functionalities and briefly outlining the techniques it employs. In this work, instead, we have two main goals: first, to provide a more detailed description of all the functionalities included in VISIONE and how each of them are implemented and combined together; second, to present an analysis of the system retrieval performance by examining the logs acquired during the VBS2019 challenge. Therefore, this manuscript primarily presents how all the aforementioned search functionalities are implemented and integrated into a unified framework that is based on a full-text search engine, such as Apache Lucene[2]; secondly, it presents an an experimental analysis for identifying the most suitable text scoring function (ranker) for the proposed textual encoding in the context of video search.

The rest of the paper is organized as follows. The next section reviews related works. Section 3 gives an overview of our system and its functionalities. Key notions on our proposed textual encoding and other aspects regarding the indexing and search phases are presented in Section 4. Section 5 presents an experimental evaluation to determine which text scoring function is the best in the context of a . Section 6 draws the conclusions.

[1] https://www-nlpir.nist.gov/projects/tv2019/data.html
[2] https://lucene.apache.org/

2. Related Work

Video search is a challenging problem of great interest in the multimedia retrieval community. It employs various information retrieval and extraction techniques, such as content-based image and text retrieval, computer vision, speech and sound recognition, and so on.

In this context, several approaches for cross-modal retrieval between visual data and text description have been proposed, , to name but a few. Many of them are image-text retrieval methods that make use of a projection of the image features and the text features into the same space (visual, textual or a joint space) so that the retrieval is then performed by searching in this latent space (e.g., [6–8]). Other approaches are referred as video-text retrieval methods as they learn embeddings of video and text in the same space by using different multi-modal features (like visual cues, video dynamics, audio inputs, and text) [9–14]. For example, [11] simultaneously utilizes multi-modal features to learn two joint video-text embedding networks: one learns a joint space between text features and visual appearance features, the other learns a joint space between text features and a combination of activity and audio features.

Many video retrieval systems are designed in order to support complex human generated queries that may include but are not limited to keywords or natural language sentences. Most of them are interactive tools where the users can dynamically refine their queries in order to better specify their search intent during the search process. The contest provides a live and fair performance assessment of interactive video retrieval systems and therefore in recent years has become a reference point for comparing state-of-the-art video search tools. During the competition, the participants have to perform various KIS and AVS tasks in a limited amount of time (generally within 5-8 minutes for each task). To evaluate the interactive search performance of each video retrieval system, several search sessions are performed by involving both expert and novice users[3].

Several video retrieval systems participated at the VBS in the last years [1,3,15,16]. Most of them, including our system, support multimodal search with interactive query formulation. The various systems differ mainly on (i) the search functionalities supported (e.g. query-by-keyword, query-by-example, query-by-sketch, etc.), (ii) the data indexing and search mechanisms used at the core of the system, (iii) the techniques employed during video preprocessing to automatically annotate selected keyframes and extract image features, (iv) the functionalities integrated into the user interface, including advanced visualization and relevance feedback. Among all the systems that participated in VBS, we recall VIRET [17], vitrivr [18], and SOM-Hunter [19], which won the competition in 2018, 2019, and 2020, respectively.

VIRET [17,20] is an interactive frame-based video retrieval system that currently provides four main retrieval modules (query by keyword, query by free-form text, queries by color sketch, and query by example). The keyword search relies on automatic annotation of video keyframes. In the latest versions of the system, the annotation is performed using a retrained deep (NasNet [21]) with a custom set of 1243 class labels. A retrained NasNet is also used to extract deep features of the images, which are then employed for similarity search. The free-form text search is implemented by using a variant of the W2VV++ model [22]. An interesting functionality supported by VIRET is the temporal sequence search, which allows a user to describe more than one frame of a target video sequence by also specifying the expected temporal ordering of the searched frames.

Vitrivr [23] is an open-source multimedia retrieval system that supports content-based retrieval of several media types (images, audio, 3D data, and video). For video retrieval, it offers different query modes, including query by sketch (both visual and semantic), query by keywords (concept labels), object instance search, speech transcription

[3] Expert users are the developers of the in race retrieval system or people that already know and use the system before the competition. Novices are users who interact with the search system for the first time during the competition.

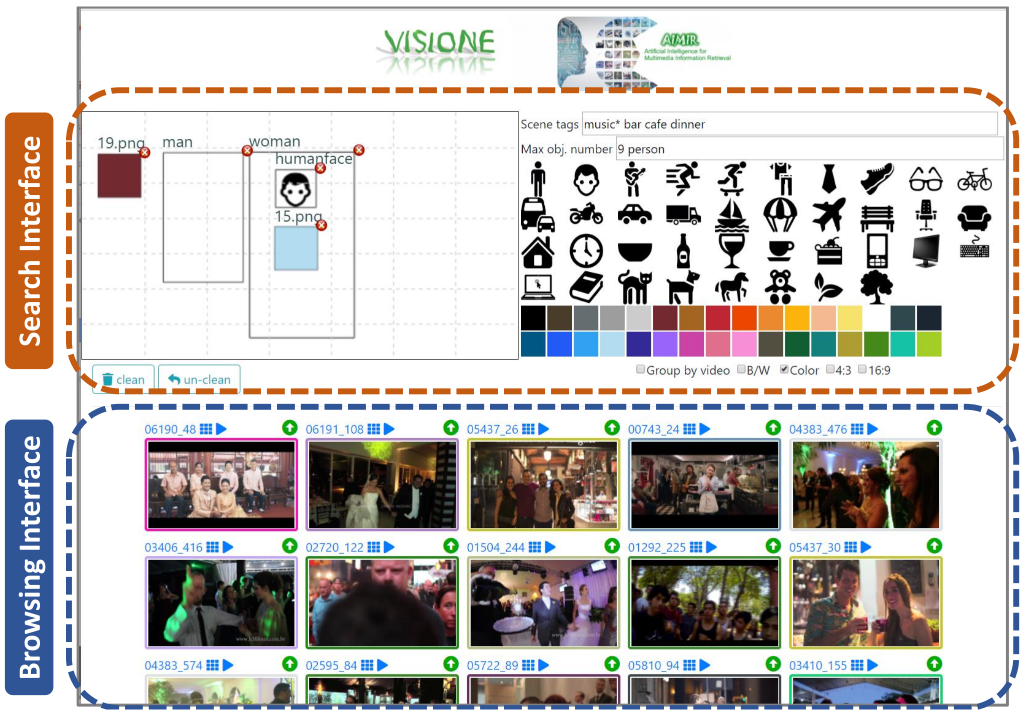
search, and similarity search. For the query by sketch and query by example, vitrivr uses several low-level image features and a pixel-wise semantic annotator [24]. The textual search is based on scene-wise descriptions, structured metadata, OCR, and ASR data extracted from the videos. Faster-RCNN [25] (pre-trained on the Openimages V4 dataset) and a ResNet-50 [7] (pre-trained on ImageNet) are used to support object instance search. The latest version of vitrivr also supports temporal queries.

SOM-Hunter [19] is an open-source video retrieval system that supports keyword search, free-text search, and temporal search functionalities, which are implemented as in the VIRET system. The main novelty of SOM-Hunter is that it relies on the user's relevance feedback to dynamically update the search results displayed using self-organizing maps (SOMs).

Our system, like almost all current video retrieval systems, relies on artificial intelligence techniques for automatic video content analysis (including automatic annotation and object recognition). Nowadays, content-based image retrieval systems (CBIR) are possible solution to the problem of retrieving and exploring a large volume of images resulting from the exponential growth of accessible image data. Many of these systems use both visual and textual features of the images, but often most of the images are not annotated or only partially annotated. Since manual annotation for a large volume of images is impractical, Automatic Image Annotation (AIA) techniques aim to bridge this gap. For the most part, AIA approaches are based solely on the visual features of the image using different techniques: one of the most common approaches consists in training a classifier for each concept and obtaining the annotation results by ranking the class probability [26,27]. There are other AIA approaches that aim to improve the quality of image annotation by using the knowledge implicit in a large collection of unstructured text describing images, and are able to label images without having to train a model (Unsupervised Image Annotation approach [28–30]). In particular, the image annotation technique we exploited is an Unsupervised Image Annotation technique originally introduced in [31].

Recently, image features built upon Convolutional Neural Networks (CNN) have been used as an effective alternative to descriptors built using image local features, like SIFT, ORB and BRIEF, to name but a few. CNNs have been used to perform several tasks, including image classification, as well as image retrieval [32–34] and object detection [35]. Moreover, it has been proved that the representations learned by CNNs on specific tasks (typically supervised) can be transferred successfully across tasks [32,36]. The activation of neurons of specific layers, in particular the last ones, can be used as features to semantically describe the visual content of an image. Tolias et al. [37] proposed the Regional Maximum Activations of Convolutions (R-MAC) feature representation, which encodes and aggregates several regions of the image in a dense and compact global image representation. Gordo et al. [38] inserted the R-MAC feature extractor in an end-to-end differentiable pipeline in order to learn a representation optimized for visual instance retrieval through back-propagation. The whole pipeline is composed by a fully convolutional neural network, a region proposal network, the R-MAC extractor and PCA-like dimensionality reduction layers, and it is trained using a ranking loss based on image triplets. In our work, as a feature extractor for video frames, we used a version of R-MAC that uses the ResNet-101 trained model provided by [39] as the core. This model has proven to perform best on standard benchmarks.

Object detection and recognition techniques also provide valuable information for semantic understanding of images and videos. In [40] the authors proposed a model for object detection and classification, which integrates Tensor features. The latter are invariant under spatial transformation and together with SIFT features (which are invariant to scaling and rotation) allow improving the classification accuracy of detected objects using a Deep Neural Network. In [41,42], the authors presented a cloud based system that analyses video streams for object detection and classification. The system is based on a scalable and robust cloud computing platform for performing automated



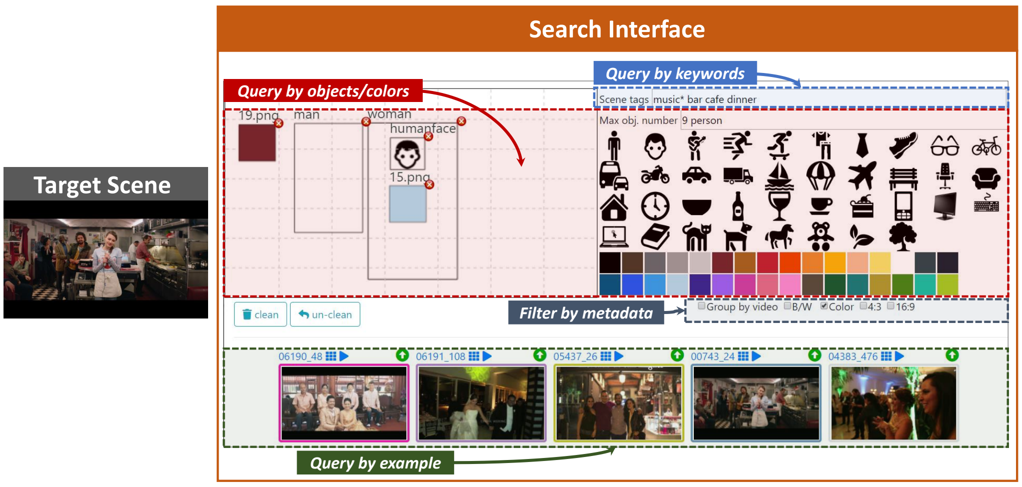
Scene tags: music* bar cafe dinner
Max obj. number: 9 person
Group by video ☐ B/W ☐ Color ☑ 4:3 ☐ 16:9 ☐

Figure 2.

analysis of thousands of recorded video streams. The framework requires a human operator to specify the analysis criteria and the duration of video streams to analyze. The streams are then fetched from a cloud storage, decoded and analyzed on the cloud. The framework executes intensive parts of the analysis on GPU-based servers in the cloud. Recently, in [43], the authors proposed an approach that combines Deep and SIFT. In particular, they extract features from the analyzed images with both approaches, they fuse the features by using a serial-based method that produces a matrix that is fed to ensemble classifier for recognition.

In our system, we used YOLOv3 [44] as CNN architecture to recognize and locate objects in the video frames. The architecture of YOLOv3 jointly performs a regression of the bounding box coordinates and classification for every proposed region. Unlike other techniques, YOLOv3 performs these tasks in an optimized fully-convolutional pipeline that takes pixels as input and outputs both the bounding boxes and their respective proposed categories. This CNN has the great advantage of being particularly fast and at the same time exhibiting remarkable accuracy. To increase the number of categories of recognizable objects, we used three different variants of the same network trained on different data sets, namely, YOLOv3, YOLO9000 [45], and YOLOv3 OpenImages [46].

One of the main peculiarities of our system, compared to others participating in VBS, is that we decided to employ a full-text search engine to index and search video content, both for the visual and textual parts. Since nowadays text search technologies have achieved impressive performance in terms of scalability and efficiency VISIONE turns out to be scalable. To take full advantage from these stable search engine technologies, we specifically designed various text encodings for all the features and descriptors extracted from the video keyframes and the user query, and we decided to use the Apache Lucene project. In previous papers, we already exploited the idea of using text encoding, named Surrogate Text Representation [47], to index and search image for deep features [47–50]. In VISIONE, we extend this idea to index also information regarding the position of objects and colors that appear in the images.



Scene tags: music* bar cafe dinner

Max obj. number: 9 person

Group by video, B/W, ✓ Color, 4:3, 16:9

Figure 3.

3. The VISIONE video search tool

VISIONE is a visual content-based retrieval system designed to support large scale video search. It allows a user to search for a video describing the content of a scene by formulating textual or visual queries (see Figure 2).

VISIONE, in fact, integrates several search functionalities and exploits deep learning technologies to mitigate the semantic gap between text and image. Specifically it supports:

- *query by keywords*: the user can specify keywords including scenes, places or concepts (e.g. outdoor, building, sport) to search for video scenes;
- *query by object location*: the user can draw on a canvas some simple diagrams to specify the objects that appear in a target scene and their spatial locations;
- *query by color location*: the user can specify some colors present in a target scene and their spatial locations (similarly to object location above);
- *query by visual example*: an image can be used as a query to retrieve video scenes that are visually similar to it.

Moreover, the search results can be filtered by indicating whether the keyframes are in color or in b/w, or by specifying its aspect ratio.

3.1. The User Interface

The VISIONE user interface is designed to be simple, intuitive and easy to use also for users who interact with it for the first time. As shown in Figure 2, it integrates the *searching* and the *browsing* functionalities in the same window.

The searching part of the interface (Figure 3) provides:

- a *text box*, named *"Scene tags"*, where the user can type keywords describing the target scene (e.g. "park sunset tree walk");
- a *color palette* and an *object palette* that can be used to easily drag & drop a desired color or object on the canvas (see below);
- a *canvas*, where the user can sketch objects and colors that appear in the target scene simply by drawing bounding-boxes that approximately indicate the positions of the desired objects and colors (both selected from the palettes above) in the scene;
- a *text box*, named *"Max obj. number"*, where the user can specify the maximum number of instances of the objects appearing in the target scene (e.g.: two glasses);
- two *checkboxes* where the user can filter the type of keyframes to be retrieved (B/W or color images, 4:3 or 16:9 aspect ratio).

The canvas is split into a grid of 7×7 cells, where the user can draw the boxes and then move, enlarge, reduce or delete them to refine the search. The user can select the

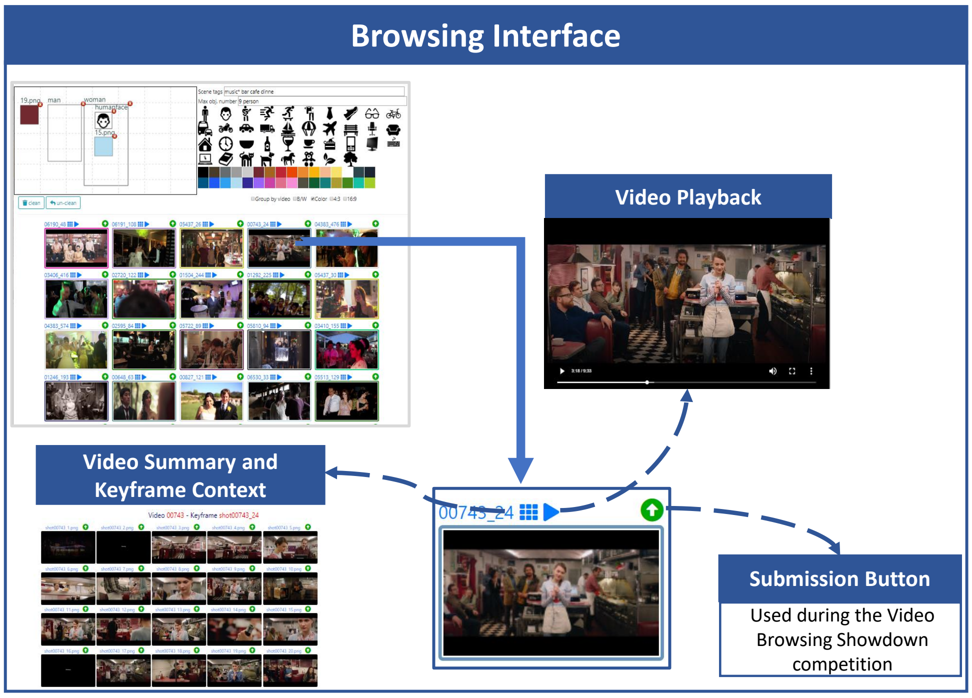

Browsing Interface
Video Playback
Video Summary and Keyframe Context
00743_24
Submission Button
Used during the Video Browsing Showdown competition

Figure 4.

desired color from the palette, drag & drop it on the canvas and then resize or move the corresponding box as desired. There are two options to insert objects in the canvas: (i) directly draw a box in the canvas using the mouse and then type the name of the object in a dialog box (auto-complete suggestions are shown to the user), (ii) drag & drop one of the object icon appearing in the object palette on the canvas. For the user's convenience, a selection of 38 common (frequently used) objects are included in the object palette.

Note that when objects are inserted in the canvas (e.g. a "person" and a "car"), then the system filters out all the images not containing the specified objects (e.g. all the scenes without a person or without a car). However, images with multiple instances of those objects can be returned in the search results (e.g. images with two or three people and one or more cars). The user can use the *"Max obj. number"* text box to specify the maximum number of instances of an object appearing in the target scene. For example by typing *"1 person 3 car 0 dog"* the system returns only images containing at most one person, three cars and no dog.

The *"Scene tags"* text box provides auto-complete suggestions to the users and for each tag also indicates the number of keyframes in the databases that are annotated with it. For example, by typing *"music"* the system suggests *"music (204775); musician (1374); music hall (290); ..."*, where the numbers indicates how many images in the database are annotated with the corresponding text (e.g. 204775 images for *"music"*, 1374 images for *"musician"*, etcetera). This information can be exploited by the user when formulating the queries. Moreover, the keyword-based search supports wildcard matching. For example, with *"music∗"* the system searches for any tag that starts with *"music"*.

Every time the user interacts with the search interface (e.g type some text or add/move/delete a bounding box) the system automatically updates the list of search results, which are displayed in the browsing interface, immediately below the search panel. In this way the user can interact with the system and gradually compose his query by also taking into account the search results obtained so far to refine the query itself.

The browsing part of the user interface (Figure 4) allows accessing the information associated with the video, every displayed keyframe belongs to it, a keyframe-based

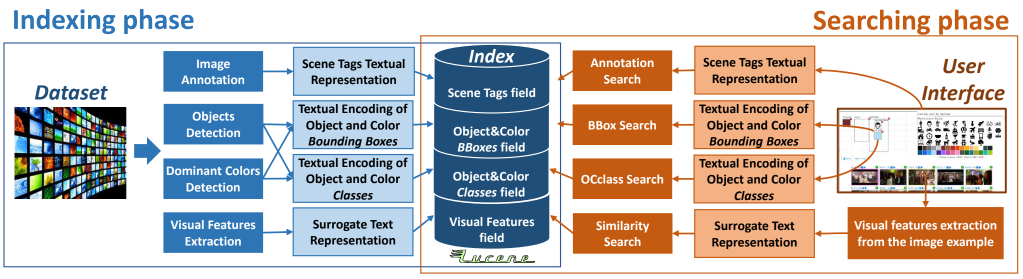

Indexing phase
Searching phase
Dataset
Image Annotation
Objects Detection
Dominant Colors Detection
Visual Features Extraction
Scene Tags Textual Representation
Textual Encoding of Object and Color Bounding Boxes
Textual Encoding of Object and Color Classes
Surrogate Text Representation
Index
Scene Tags field
Object&Color BBoxes field
Object&Color Classes field
Visual Features field
Lucene
Annotation Search
BBox Search
OClass Search
Similarity Search
Scene Tags Textual Representation
Textual Encoding of Object and Color Bounding Boxes
Textual Encoding of Object and Color Classes
Surrogate Text Representation
User Interface
Visual features extraction from the image example

Figure 5. System Architecture: a general overview of the components of the two main phases of the system, the indexing and the browsing.

video summary and playing the video starting from the selected keyframe. In this way, the user can easily check if the selected image belongs to the searched video. The search results can also be grouped together according to the fact that the keyframes belong to the same video. This visualization option can be enabled/disabled by clicking on the *"Group by video"* checkbox. Moreover, while browsing the results, the user can use one of the displayed images to perform an image Similarity Search and retrieve frames visually similar to the one selected. A Similarity Search is executed by double clicking on an image displayed in the search results.

3.2. System Architecture Overview

The general architecture of our system is illustrated in Figure 5. Each component of the system will be described in detail in the following sections; here we give an overview of how it works. To support the search functionalities introduced above, our system exploits deep learning technologies to understand and represent the visual content of the database videos. Specifically, it employs:

- an image annotation engine, to extract scene tags (see Sec. 4.1);
- state-of-the-art object detectors, like YOLO [4], to identify and localize objects in the video keyframes (see Sec 4.2);
- spatial colors histograms, to identify dominant colors and their locations (see Sec 4.2);
- the R-MAC deep visual descriptors, to support the Similarity Search functionality (see Sec. 4.3)

The peculiarity of the approach used in VISIONE is to represent all the different types of descriptors extracted from the keyframes (visual features, scene tags, colors/object locations) with a textual encoding that is indexed in a single text search engine. This choice allows us to exploit mature and scalable full-text search technologies and platforms for indexing and searching video repository. In particular, VISIONE relies on the Apache Lucene full-text search engine. The text encoding used to represent the various types of information, associated with every keyframe, is discussed in Section 4.

Also the queries formulated by the user through the search interface (e.g. the keywords describing the target scene and/or the diagrams depicting objects and the colors locations) are transformed into textual encoding, in order to process them. We designed a specific textual encoding for each typology of data descriptor as well as for the user queries.

In the full-text search engine, the information extracted from every keyframe is composed of four textual fields, as shown in Figure 5:

- *Scene Tags*, containing automatically associated tags;
- *Object&Color BBoxes*, containing text encoding of colors and objects locations;
- *Object&Color Classes*, containing global information on objects and colors in the keyframe;

[4] https://pjreddie.com/darknet/yolo/

- *Visual Features*, containing text encoding of extracted visual features.

These four fields are used to serve the four main search operations of our system:

- *Annotation Search*, search for keyframes associated with specified annotations;
- *BBox Search*, search for keyframes having specific spatial relationships among objects/colors;
- *OClass Search*, search for keyframes containing specified objects/colors;
- *Similarity Search*, search for keyframes visually similar to a query image

The user query is broken down into In the next section, we will describe the four search operations and further details on the indexing and searching phases.

4. Indexing and Searching Implementation

In VISIONE, as already anticipated, content of keyframes is represented and indexed using automatically generated annotations, positions of occurring objects, positions of colors, and deep visual features. In the following we describe how these descriptors are extracted, indexed, and searched.

4.1. Image Annotation

One of the most natural ways of searching in a large multimedia data set is using a keyword-based query. To support such kind of queries, we employed our automatic annotation system[5] that is introduced in [31]. This system is based on an unsupervised image annotation approach that exploits the knowledge implicitly existing in a huge collection of unstructured texts describing images, allowing us to annotate the images without using a specified trained model. The advantage is that the target vocabulary we used for the annotation reflects well the way people actually describe their pictures. Specifically, our system uses the tags and the descriptions contained in the metadata of a large set of media selected from the Yahoo Flickr Creative Commons 100 Million (YFCC100M) dataset [51]. Those tags are validated using WordNet [52], cleaned and then used as the knowledge base for the automatic annotation.

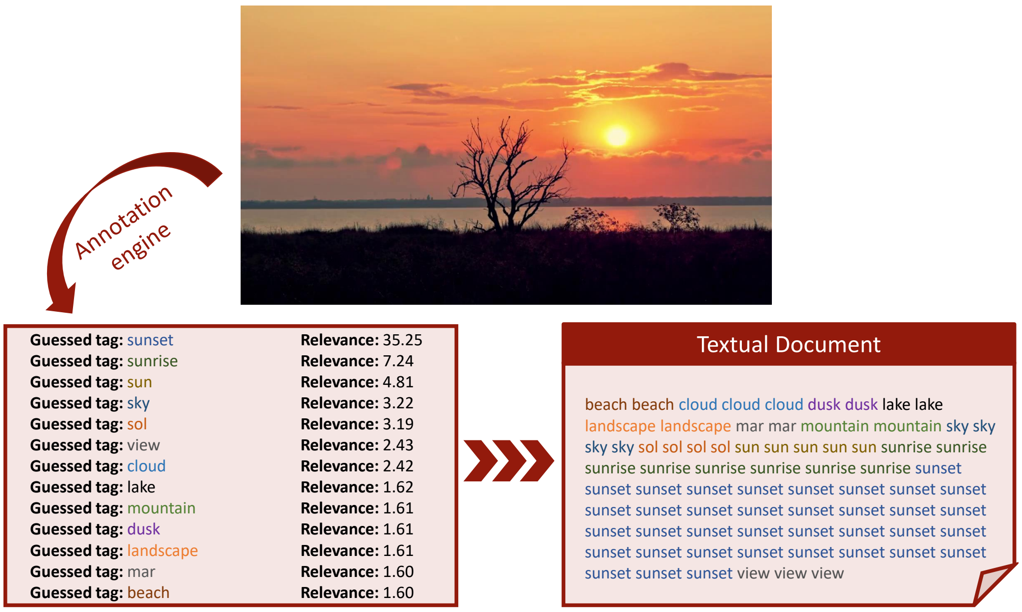
The subset of the YFCC100M dataset that we used for building the knowledge base was selected by identifying images with relevant textual descriptions and tags. To this scope, we used a metadata cleaning algorithm that leverages on the semantic similarities between images. Its core idea is that if a tag is contained in the metadata of a group of very similar images, then that tag is likely to be relevant for all these images. The similarity between images was measured by means of visual deep features; specifically, we used the output of the sixth layer of the neural network Hybrid-CNN [6] as visual descriptors. [7].

As a result of our metadata cleaning algorithm we selected about 16 thousands terms associated with about one million images. The set of deep features extracted from those images were then indexed using the MI-file index [53] in order to allow us to access the data and perform similarity search in a very efficient way.

The annotation engine is based on a k-NN classification algorithm. An image is annotated with the most frequent tags associated with the most similar images in the YFCC100M cleaned subset. The specific definition of the annotation algorithm is out of the scope of this paper and we refer to [31] for further details.

In Figure 6, we show an example of annotation obtained with our system. Please note that our system also provides a relevance score to each tag associated with the image. The bigger the score the more relevant the tag. We used our annotation system to label the video keyframes of the V3C1 dataset. For each keyframe we produce a “tag textual encoding" by concatenating all the tags associated with the images. In order to represent the relevance of the associated tag, each tag is repeated a number of times

[5] Demo available at http://mifile.deepfeatures.org
[6] Publicly available in the Caffe Model Zoo, http://github.com/BVLC/caffe/wiki/Model-Zoo
[7]



Guessed tag: sunset	**Relevance:** 35.25
Guessed tag: sunrise	**Relevance:** 7.24
Guessed tag: sun	**Relevance:** 4.81
Guessed tag: sky	**Relevance:** 3.22
Guessed tag: sol	**Relevance:** 3.19
Guessed tag: view	**Relevance:** 2.43
Guessed tag: cloud	**Relevance:** 2.42
Guessed tag: lake	**Relevance:** 1.62
Guessed tag: mountain	**Relevance:** 1.61
Guessed tag: dusk	**Relevance:** 1.61
Guessed tag: landscape	**Relevance:** 1.61
Guessed tag: mar	**Relevance:** 1.60
Guessed tag: beach	**Relevance:** 1.60

beach beach cloud cloud cloud dusk dusk lake lake landscape landscape mar mar mountain mountain sky sky sky sky sol sol sol sol sun sun sun sun sun sunrise sunrise sunrise sunrise sunrise sunrise sunrise sunrise sunset view view view

Figure 6. Example of our image annotation and its representation as single textual document. In the textual document, each tag is repeated a number of times equal to the least integer greater than or equal to the tag relevance

equal to the relevance score of the tag itself (the relevance of each tag is approximated to an integer using the ceiling function). The ordering of the tags in the concatenation is not important because what matters are the tag frequencies. In Figure 6 the box named *Textual Document* shows an example of concatenation associated with a keyframe.

Annotation Search.

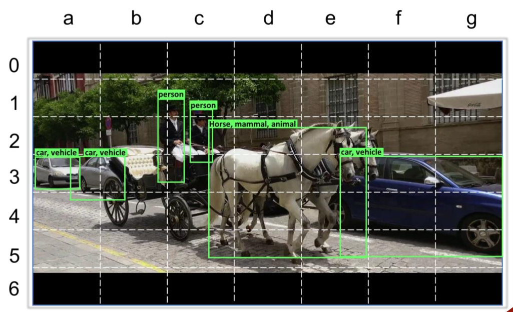
The annotations, generated as described above, can be used to retrieve videos, by typing keywords in the *"Scene tags"* text box of the user interface (see Figure 3). As already anticipated in Section 3.2, we call *Annotation Search* this searching option. The Annotation Search is executed performing a full-text search. As described in Section 5, during the VBS competition the similarity was used as a text scoring function.

4.2. Objects and Colors

Information related to objects and colors in a keyframe are treated in a similar way in our system. Given a keyframe we store both local and global information about objects and colors contained in it. As we discussed in Section 3.2, the positions where objects and colors occur are stored in the *Object&Color BBoxes* field; all objects and colors occurring in a frame are stored in the *Object&Color Classes* field.

4.2.1. Objects

We used a combination of three different versions of YOLO to perform object detection: YOLOv3 [44], YOLO9000 [45], and YOLOv3 OpenImages [46], to extend the number of detected objects. The idea of using YOLO to detect objects within video has already been exploited in VBS, e.g. by Truong et al. [54]. The peculiarity of our approach is that we combine and encode the spatial position of the detected objects in a single textual description of the image. The textual encoding of this information is created as follows. For each image, we have a space-separated concatenation of *ENC*s, one for all the cells (cod_{loc}) in the grid that contains the object (cod_{class}): for example, for the image in Figure 7 the rightmost car is indexed with the sequence $\{e3car\ f3car\ ...\ g5car\}$ where "car" is the cod_{class} of the object *car*, located in cells $e3, f3, g3, e4, f4, g4, e5, f5, g5$. This information is stored in the *Object&Color BBoxes* field of the record associated with the



	a	b	c	d	e	f	g
0							
1		person	person, person				
2		person	person, person, horse, mammal, animal	horse, mammal, animal	horse, mammal, animal		
3	car, car, vehicle, vehicle	person, car, vehicle	person, person, horse, mammal, animal	horse, mammal, animal	horse, mammal, animal, car, vehicle	car, vehicle	car, vehicle
4	car, vehicle	car, vehicle	horse, mammal, animal	horse, mammal, animal	horse, mammal, animal, car, vehicle	car, vehicle	car, vehicle
5			horse, mammal, animal	horse, mammal, animal	horse, mammal, animal, car, vehicle	car, vehicle	car, vehicle
6							

Textual Encoding of Object Classes

person1 person2 vehicle1 vehicle2 vehicle3 car1 car2 car3 mammal1 horse1 animal1

Textual Encoding of Object Bounding Boxes

a3car a3car a3vehicle a3vehicle a4car a4vehicle b1person b2person b3person b3car b3vehicle b4car b4vehicle c1person c1person c2person c2person c2horse c2mammal c2animal c3person c3person c3horse c3mammal c3animal c4horse c4mammal c4animal c5horse c5mammal c5animal d2horse d2mammal d2animal d3horse d3mammal d3animal d4horse d4mammal d4animal d5horse d5mammal d5animal e2horse e2mammal e2animal e3horse e3mammal e3animal e3car e3vehicle e4horse e4mammal e4animal e4car e4vehicle e5horse e5mammal e5animal e5car e5vehicle f3car f3vehicle f4car f4vehicle f5car f5vehicle g3car g3vehicle g4car g4vehicle g5car g5vehicle

Figure 7.

keyframe. In addition to the position of objects, we also maintain global information about the objects contained in a keyframe, in terms of number of occurrences of each object detected in the image (see Figure 7). Occurrences of objects in a keyframe are encoded by repeating the object (cod_{class}) as many times as the number of the occurrences (cod_{occ}) of the object itself. This information is stored using an encoding that composes the classes with their occurrences in the image: ($cod_{class}cod_{occ}$). For example, in Figure 7, YOLO detected 2 persons, 3 cars, which are also classified as vehicle by the detector, and 1 horse, also classified as animal and mammal, and this results in the Object Classes encoding as "*person1 person2 vehicle1 vehicle2 vehicle3 car1 car2 car3 mammal1 horse1 animal1*". This information is stored in the *Object&Color Classes* field of the record associated with the keyframe.

4.2.2. Colors

To represent colors, we use a palette of 32 colors[8] which represents a good trade-off between the huge miscellany of colors and simplicity of choice for the user at search time. For the creation of the color textual encoding we used the same approach employed to encode the object classes and locations, using the same grid of 7×7 cells. To assign the colors to each cell of the grid we used the following approach. We first evaluate the color of each pixel by using the CIELAB color space. Then, we map the evaluated color of the pixel to our 32-colors palette. To do so, we perform a k-NN similarity search between the evaluated color and our 32 colors to find the colors in our palette that most match the color of the current pixel. The metric used for this search is the Earth Mover's Distance [55]. We take into consideration the first two colors in k-NN results. The first color is assigned to that pixel. We then compute the ratio between the scores of the two colors and if it is greater than 0.5 then we also assign the second color to that pixel. This is done to allow matching of very similar colors during searching. We repeat this for each pixel of a cell in the grid and then we sum the occurrences of each color of our palette for all the pixels in the cell. Finally, we assign to that cell all the colors whose occurrence is greater than 7% of the number of pixels contained in the cell. So more than one color may be assigned to a single cell. This redundancy helps reduce misclassified colors from what they appear to the human eye.

[8] https://lospec.com/palette-list

The colors assigned to all the 7×7 cells are then encoded into two textual documents, one for the color locations and one for the global color information, using the same approach employed to encode object classes and locations, and discussed in section 4.2.1. Specifically, the textual document associated to the color location is obtained by concatenating textual encodings of the form $cod_{loc}cod_{class}$, where cod_{loc} is an identifier of a cell and cod_{class} is the identifier of a color assigned to the cell. This information is stored in the *Object&Color BBoxes* field. The textual document for the color classes is obtained by concatenating the text identifiers (cod_{class}) of all the colors assigned to the image. This information is stored in the *Object&Color Classes* field of the record associated with the keyframe.

Object and Color Location Search.

At run-time phase, the search functionalities for both the query by object and color location are implemented using two search operations: the bounding box search (*BBox Search*) and the object/color-class search (*OClass Search*).

The user can draw a bounding box in a specific position of the canvas and specify which object/color wants to found in that position, or he/she can drag & drop a particular object/color from the palette in the user interface and resize the corresponding bounding box as desired (as shown in the "Query by object/colors" of Figure 3). All the bounding boxes present in the canvas, both related to objects and colors, are then converted into the two textual encoding described respectively

For the actual search phase, first an instance of the OClass Search operator is executed. This operator tries to find a match between all the objects represented in the canvas and the frames stored in the index that contains these objects. This produces a result set containing a subset of the dataset with all the frames that match the objects in the canvas. After this, the BBox Search operator performs a rescoring of the result set by matching the textual encoding of the Object and Color Bounding Boxes encoding of the query with all the corresponding encodings in the index. The metric used in this case during the VBS competition was BM25. After the execution of these two search operators, the frames that satisfied these two searches ordered by descending score are shown in the browsing part of the user interface.

4.3. Deep Visual Features

VISIONE also supports content-based visual search functionality, i.e., it allows users to retrieve keyframes visually similar to a query image given by example. In order to represent and compare the visual content of the images, we use the Regional Maximum Activations of Convolutions (R-MAC) [37], which is a state-of-art descriptor for image retrieval. The R-MAC descriptor effectively aggregates several local convolutional features (extracted at multiple positions and scales) into a dense and compact global image representation. We use the ResNet-101 trained model provided by Gordo et al. [38] as an R-MAC feature extractor since it achieved the best performance on standard benchmarks. The used descriptors are 2048-dimensional real-valued vectors.

To efficiently index the R-MAC descriptor, we transform the deep features into a textual encoding suitable for being indexed by a standard full-text search engine. We used the *Scalar Quantization-based Surrogate Text representation* to transform the deep features into a textual encoding, which was proposed in [49]. The idea behind this approach is to map the real-valued vector components of the R-MAC descriptor into a (sparse) integer vector that acts as the term frequencies vector of a synthetic codebook. Then the integer vector is transformed into a text document by simply concatenating some synthetic codewords so that the term frequency of the i-th codeword is exactly the i-th element of the integer vector. For example, the four-dimensional integer vector $[2, 1, 0, 1]$ is encoded with the text "τ_1 τ_1 τ_2 τ_4", where $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ is a codebook of four synthetic alphanumeric terms.

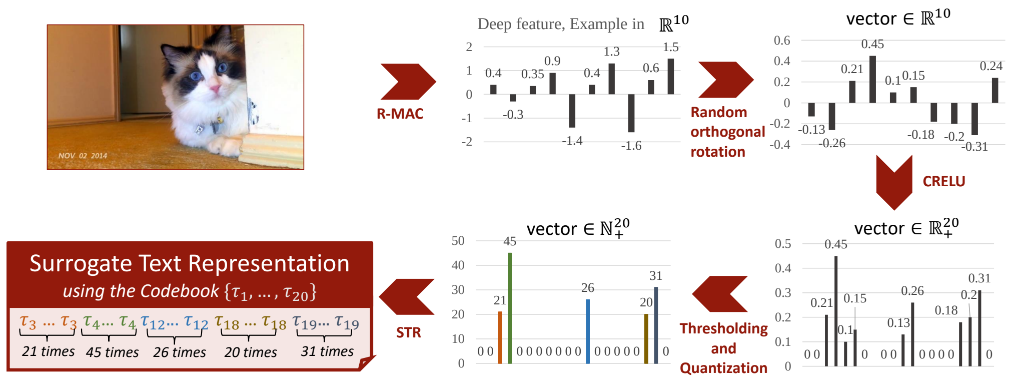

R-MAC
Deep feature, Example in $\mathbb{R}^{10}$
Random orthogonal rotation
vector ∈ $\mathbb{R}^{10}$
CRELU
vector ∈ $\mathbb{R}^{20}_{+}$
Thresholding and Quantization
vector ∈ $\mathbb{N}^{20}_{+}$
STR
Surrogate Text Representation
using the Codebook $\{\tau_1, \ldots, \tau_{20}\}$
$\tau_3 \ldots \tau_3$ 21 times
$\tau_4 \ldots \tau_4$ 45 times
$\tau_{12} \ldots \tau_{12}$ 26 times
$\tau_{18} \ldots \tau_{18}$ 20 times
$\tau_{19} \ldots \tau_{19}$ 31 times

Figure 8.

The overall process used to transform an R-MAC descriptors into a textual encoding is summarized in Figure 8 (for simplicity, the R-MAC descriptor is depicted as a 10-dimensional vector). The mapping of the deep features into the term frequencies vectors is designed (i) to preserve as much as possible the rankings, i.e. similar features should be mapped into similar term frequencies vectors (for effectiveness) and (ii) to produce sparse vectors, since each data object will be stored in as many posting lists as the non-zero elements in its term frequencies vector (for efficiency). To this end, the deep features are first centered using their mean and then rotated using a random orthogonal transformation. The random orthogonal transformation is particularly useful to distribute the variance over all the dimensions of the vector as it provides good balancing for high dimensional vectors without the need to search for an optimal balancing transformation. In this way, we try to increase the cases where the dimensional components of the features vectors have the same mean and variance, with mean equal to zero. Moreover the used roto-traslation preserves the rankings according to the dot-product (see [49] for more details). Since search engines, like the one we used, use an inverted file to store the data, as a second step, we have to sparsify the features. Sparsification guarantees the efficiency of these indexes. To achieve this, Scalar Quantization approach maintains components above a certain threshold by zeroing all the others and quantizing the non-zero elements to integer values. To deal with negative values the Concatenated Rectified Linear Unit (CReLU) transformation [56] is applied before the thresholding. Note that the CReLU simply makes an identical copy of vector elements, negates it, concatenates both original vector and its negation, and then zeros out all the negative values.

In VISIONE the Surrogate Text Representation of a dataset image is stored in the *"Visual Features"* field of our index (Figure 5).

Similarity Search.

VISIONE relies on the Surrogate text encodings of images to perform the Similarity Search. When the user starts a Similarity Search by selecting a keyframe in the browsing interface, the system retrieves all the indexed keyframes whose Surrogate Text Representation are similar to the Surrogate Text Representation of the selected keyframe. We used the dot product over the frequency terms vectors (TF ranker) as text similarity function since it achieved very good performance for large-scale image retrieval task [49].

4.4. Overview of the Search Process

As we described so far, our system relies on four *search operations*: an Annotation Search, a BBox Search, an OClass Search, and a Similarity Search. Every time a user interacts with the VISIONE interface (add/remove/update a bounding box, add/remove a keyword, click on an image, etc...), a new query Q is executed, where Q is the sequence of the instances of search operations currently active in the interface. The query is then

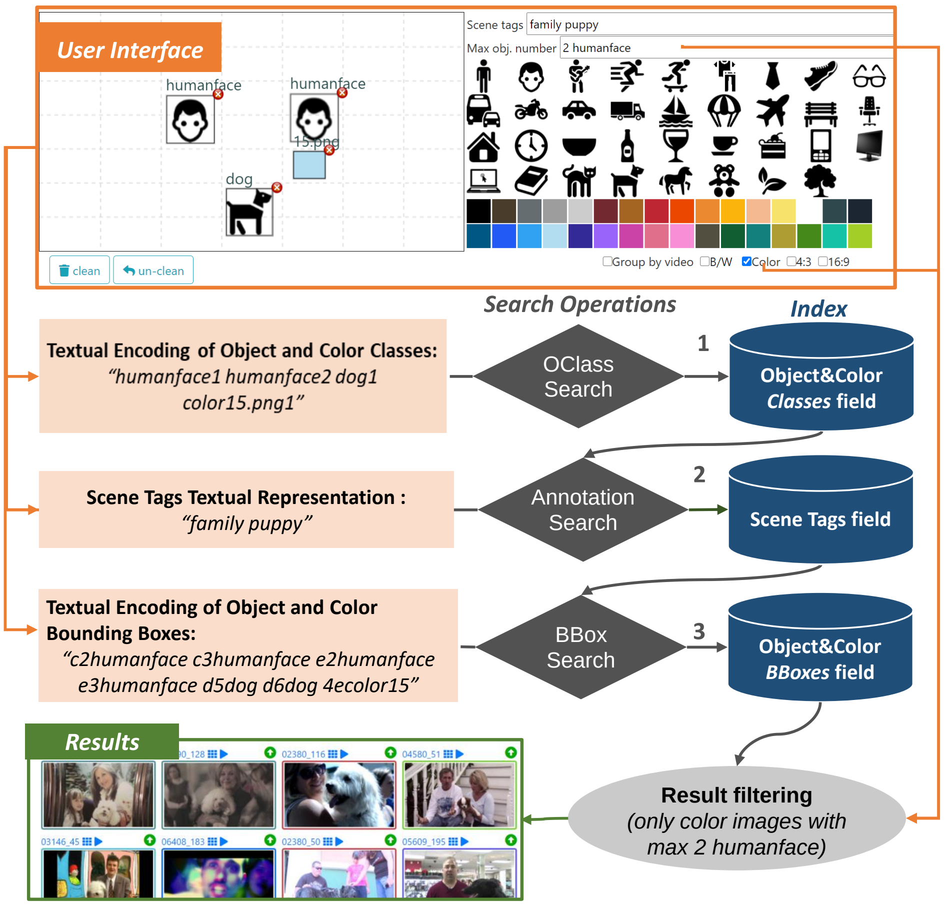

User Interface
humanface
humanface
dog
15.png
clean
un-clean
Scene tags family puppy
Max obj. number 2 humanface
Group by video
B/W
Color
4:3
16:9
Search Operations
Index
Textual Encoding of Object and Color Classes: "humanface1 humanface2 dog1 color15.png1"
OClass Search
1
Object&Color Classes field
Scene Tags Textual Representation : "family puppy"
Annotation Search
2
Scene Tags field
Textual Encoding of Object and Color Bounding Boxes: "c2humanface c3humanface e2humanface e3humanface d5dog d6dog 4ecolor15"
BBox Search
3
Object&Color BBoxes field
Results
Result filtering (only color images with max 2 humanface)

Figure 9.

split into subqueries, where a *subquery* contains instances of a single search operation. In a nutshell, the system runs all the subqueries using the appropriate search operation and then combines the search results using a sequence of reordering. In particular, we designed the system so the OClass Search operation has the priority: the result set contains all the images which match the given query with taking into account the classes drawn in the canvas (both object and colors), and not their spatial location. If the query includes also some scene tags (text box of the user interface), then the Annotation Search is performed but only on the result set generated by the first OClass Search. So in this case the Annotation Search actually produces only a rescoring of the results obtained at the previous step. Finally, another rescore is performed using the BBox Search. If the user does not issue any annotation keyword in the interface, only the OClass Search and BBox Search are used. If, on the other hand, only one or more keywords are put in the interface, only the Annotation Search is used to find the results.

However, we that in future versions of VISIONE it may be interesting to also include the possibility of using Similarity Search to reorder the results obtained from other search operations.

5. Evaluation

As already discussed in Sections 3 and 4, a user query is executed as a combination of search operations (Annotation Search, BBox Search, OClass Search, and Similarity Search). The final result set returned to the user highly depends on the results returned by each executed search operation. Each search operation is implemented in Apache Lucene using a specific *ranker* that determines how the textual encoding of the database items are compared with the textual encoding of the query in order to provide the ranked list of results.

In our first implementation of the system, used at the VBS competition in 2019, we tested for each search operation various rankers, and we estimated the performance of the system using our personal experience and feeling. Specifically, we tested a set of queries with different rankers and we select the ranker that provided us with good results in the top positions of the returned items. However, given the lack of a ground truth, this qualitative analysis was based on a subjective feedback provided by a member of our team who explicitly looked at the top-returned images obtained with the various tested scenarios, and judged how good the results were.

After the competition, we decided to have a more accurate approach to estimate the performance of the system, and the results of this analysis are discussed in this section. As the choice of the rankers strongly influences the performance of the system, we decided to have a more in-depth and objective analysis based on this part of the system. The final scope of this analysis is finding for our system the best rankers combination. Intuitively, the best combination of rankers is the one that, on average, puts more often good results (that is target results for the search challenge) at the top of the result list. Specifically, we used the query logs acquired during the participation at the challenge. The logs store all the sequences of search operations that were executed as consequence of users interacting with the system. By using these query logs, we were able to re-execute the same user sessions using different rankers. In this way we objectively measured the performance of the system, obtained when the same sequence of operation was executed with different rankers.

We focus mainly on the rankers for the BBox Search, OClass Search, and Annotation Search. We do not consider the Similarity Search as it is an independent search operation in our system, and previous work [49] already proved that the dot product (TF ranker) works well with the surrogate text encodings of the R-MAC descriptors, which are the features adopted in our system for the Similarity Search.

5.1. Experiment Design and Evaluation Methodology

As anticipated before, our analysis makes use of the log of queries executed during the 2019 VBS competition. The competition was divided in three content search *tasks*: *visual KIS*, *textual KIS* and *AVS*, already described in Section 1. For each task, a series of *runs* is executed. In each run, the users are requested to find one or more target videos. When the user believes that he/she has found the target video, he/she submits the result to the organization team that evaluates the submission.

After the competition, the organizers of VBS provided us with the VBS2019 server dataset that contains all the tasks issued at the competition (target video/textual description, start/end time of target video for KIS tasks, and ground-truth segments for KIS tasks), the client logs for all the systems participating to the competition, and the submissions made by the various teams. We used the ground-truth segments and the log of the queries submitted to our system to evaluate the performance of our system under different settings. We restricted the analysis only to the logs related to textual and visual KIS tasks since ground-truths for AVS tasks were not available[9].

During the VBS competition a total of four users (two experts and two novices) interacted with our system to solve 23 tasks (15 visual KIS and 8 textual KIS). The total number of queries executed on our system for those tasks was 1600[10].

In our analysis, we considered four different rankers to sort the results obtained by each search operation of our system. Specifically we tested the rankers based on the following text scoring function:

[9] Please note that for the AVS tasks the evaluation of the correctness of the results submitted by each team during the competition was made on site by members of a jury who evaluated the submitted images one by one. For these tasks, in fact, a predefined ground-truth is not available.

[10] We recall that, in our system, a new query is executed at each interaction of a user with the search interface.

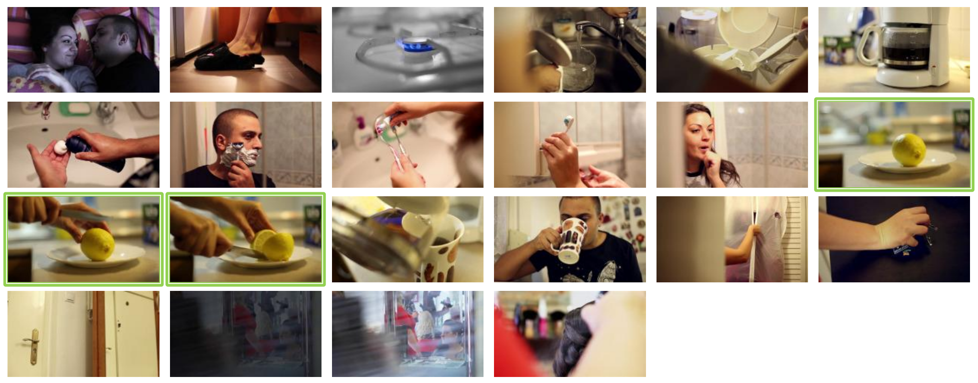


Figure 10. Example of the ground-truth keyframes for a 20 second video clip used as a KIS task at VBS2019. During the competition, our team correctly found the target video by formulating a query describing one of the keyframes depicting a lemon. However, note that most of the keyframes in the ground-truth were not relevant for the specific query submitted to our system .

- *BM25*: Lucene's implementation of the well-known similarity function BM25 introduced in [57];
- *TFIDF*: Lucene's implementation of the weighing scheme known as introduced in [58];
- *TF*: implementation of dot product similarity over the frequency terms vector;
- *NormTF*: implementation of cosine similarity (the normalized dot product of the two weight vectors) over the frequency terms vectors.

Since we three search operations and four rankers, we have a total of 64 possible combinations. We denote each combination with a triplet R_{BB}-R_{AN}-R_{OC} where R_{BB} is the ranker used for the BBox Search, R_{AN} is the ranker used for the Annotation Search, and R_{OC} is the ranker used for the OClass Search. In the implementation of VISIONE used at the 2019 VBS competition, we employed the combination BM25-BM25-TF. With the analysis reported in this section, we compare all the different combinations in order to find the one that is most suited for the video search task.

For the analysis reported in this section we went through the logs and automatically re-executed all the queries using the 64 different combinations of rankers in order to find the one that, with the highest probability, finds a relevant result (i.e. a keyframe in the ground-truth) in the top returned results. Each combination was obtained by selecting a specific ranker (among BM25, NormTF, TF, and TFIDF) for each search operation (BBox Search, Annotation Search, and OClass Search).

5.1.1. Evaluation Metrics

During the competition the user has to retrieve a video segment from the database using the functionalities of the system. A video segment is composed of various keyframes, which can be significantly different from one another, see Figure 10 as an example.

In our analysis, we assume that the user stops examining the ranked result list as soon as he/she finds one relevant result, that is one of the keyframes belonging to the target video. Therefore, given that relevant keyframes can be significantly different one from the other, we do not take into account the rank position of *all* the keyframes composing the ground-truth of a query, as required for performance measures like *Mean Average Precision* or *Discounted Cumulative Gain*. We want to measure how the system is good at proposing in the top position at least one of the target keyframes.

In this respect, we use the *Mean Reciprocal Rank-MRR* (Equation (5.1.1)) as a quality measure, since it allows us to evaluate how good is the system in returning at least one relevant result (one of the keyframes of the target video) in top position of the result set.

Formally, given a set Q of queries, for each $q \in Q$ let $\{I_1^{(q)}, \ldots, I_{n_q}^{(q)}\}$ the ground-truth, i.e., the set of n_q keyframes of the target video-clip searched using the query q; we define:

- $rank(I_j^{(q)})$ as the rank of the image $I_j^{(q)}$ in the ranked results returned by our system after executing the query q
- $r_q = \min_{j=1,\ldots n_q} rank(I_j^{(q)})$ as the rank of the first correct result in the ranked result list for the query q.

The Mean Reciprocal Rank for the query set Q is given by

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} RR(q), \tag{1}$$

where the Reciprocal Rank (RR) for a single query q is defined as

$$RR(q) = \begin{cases} 0 & \text{no relevant results} \\ 1/r_q & \text{otherwise} \end{cases} \tag{2}$$

We evaluated the MRR for each different combination of rankers. Moreover, as we expect that a user inspects just a small portion of the results returned in the browsing interface, we also evaluate the performance of each combination in finding at least one correct result in the top k positions of the result list (k can be interpreted as the maximum number of images inspected by a user). To this scope we computed the MRR at position k (MRR@k):

$$MRR@k = \frac{1}{|Q|} \sum_{q \in Q} RR@k(q) \tag{3}$$

where

$$RR@k(q) = \begin{cases} 0 & r_q > k \text{ OR no relevant results} \\ 1/r_q & \text{otherwise} \end{cases} \tag{4}$$

In the experiments we consider values of k smaller than 1000, with a focus on values between 1 and 100 as we expect cases where a user inspects more than 100 results to be less realistic.

5.2. Results

In our analysis, we used $|Q| = 521$ queries (out of 1600 above mentioned) to calculate both MRR and $MRR@k$. In fact the rest of the queries executed on our system during the VBS2019 competition are not eligible for our analysis since they are not informative to choose the best ranker configuration:

- about 200 queries involved the execution of a Similarity Search, a video summary or a filtering, whose results are independent of the rankers used in the three search operations considered in our analysis;
- the search result sets of about 800 queries do not contain any correct result due to the lack of alignment between the text associated with the query and the text associated with images relevant to the target video. For those cases, the system is not able to display the relevant images in the result set regardless of the ranker used. In fact, the effect of using a specific ranker only affects the ordering of the results and not the actual selection of them.

Note that the combination that we used at VBS2019 (indicated with diagonal lines in the graph), and that was chosen according to subjective feelings, has a good performance, but it is not the best. In fact, we noticed that there exist some patterns in the combinations of the rankers used for the OClass Search and the Annotation Search which are particularly effective and some which, instead, provide us with very poor results. For example, the combinations that use *TF* for the OClass Search and *BM25* for the Annotation Search gave

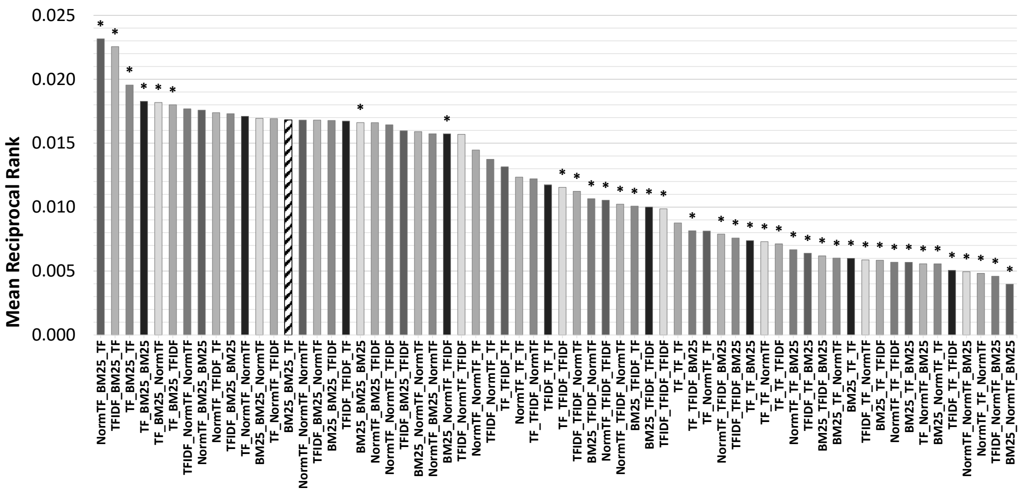


Figure 11.

Table 1: MRR@k for eight combinations of the rankers (the four best, the four worst and thesetting used at VBS2019) varying k.Statistically significant results with two-sided p value lower than 0.05 over the baseline *BM25-BM25-TF* are marked with $*$.

	$k = 1$	$k = 5$	$k = 10$	$k = 50$	$k = 100$	$k = 500$	$k = 1000$
NormTF-BM25-TF	0.015	0.017	0.019 $*$	0.022 $*$	0.022 $*$	0.023 $*$	0.023 $*$
TFIDF-BM25-TF	0.013	0.016	0.018 $*$	0.021 $*$	0.022 $*$	0.022 $*$	0.022 $*$
TF-BM25-TF	0.013	0.016	0.017	0.018 $*$	0.019 $*$	0.019 $*$	0.019 $*$
TF-BM25-BM25	0.013	0.015	0.016	0.017	0.017 $*$	0.018 $*$	0.018 $*$
TF-BM25-NormTF	0.013	0.015	0.016	0.017 $*$	0.017 $*$	0.018 $*$	0.018 $*$
BM25-BM25-TF (VBS 2019)	0.013	0.014	0.015	0.016	0.016	0.016	0.017
NormTF-TF-NormTF	0.000 $*$	0.001 $*$	0.003 $*$	0.004 $*$	0.004 $*$	0.005 $*$	0.005 $*$
NormTF-NormTF-BM25	0.000 $*$	0.001 $*$	0.002 $*$	0.004 $*$	0.004 $*$	0.005 $*$	0.005 $*$
BM25-NormTF-BM25	0.002 $*$	0.002 $*$	0.002 $*$	0.003 $*$	0.003 $*$	0.004 $*$	0.004 $*$
TFIDF-NormTF-BM25	0.000 $*$	0.001 $*$	0.001 $*$	0.003 $*$	0.004 $*$	0.004 $*$	0.004 $*$

us the overall best results. While the combinations that use *BM25* for the OClass Search and the *NormTF* for the Annotation Search have the worse performance. Specifically, we have a MRR of 0.023 for the best (NormTF-BM25-TF) and 0.004 for the worst (BM25-NormTF-BM25) a further analysis of the MRR results, it turned out quite clearly that for the Annotation Search the ranker *BM25* is particularly effective, while the use of the *TF* ranker highly degrades the performance.

Furthermore, to complete the analysis on the performance of the rankers, we analyze the MMR@k, where k is the parameter that controls how many results are shown to the user in the results set.

In conclusion, we identified the combination *NormTF-BM25-TF* as the best one, a relative improvement of 38% in *MRR* and 40% in *MRR*@100 with respect to the setting previously used at the VBS competition.

5.3. Efficiency and Scalability Issues

As we stated in the introduction, the fact that the retrieval system proposed in this article is built on top of a text search engine guarantees in principle efficiency and scalability of queries. This has been practically verified by obtaining average response times of less than a second for all types of queries (even more complex ones). On the scalability of the system, we can make some optimistic assumptions because we have not conducted experiments on it. This optimistic assumption is based on the observation that if the "synthetic" documents generated for visual search by similarity, and for the localization of objects and colors behave as textual documents then the scalability of our system is comparable to that of commercial Web search engines. To this end, with regard

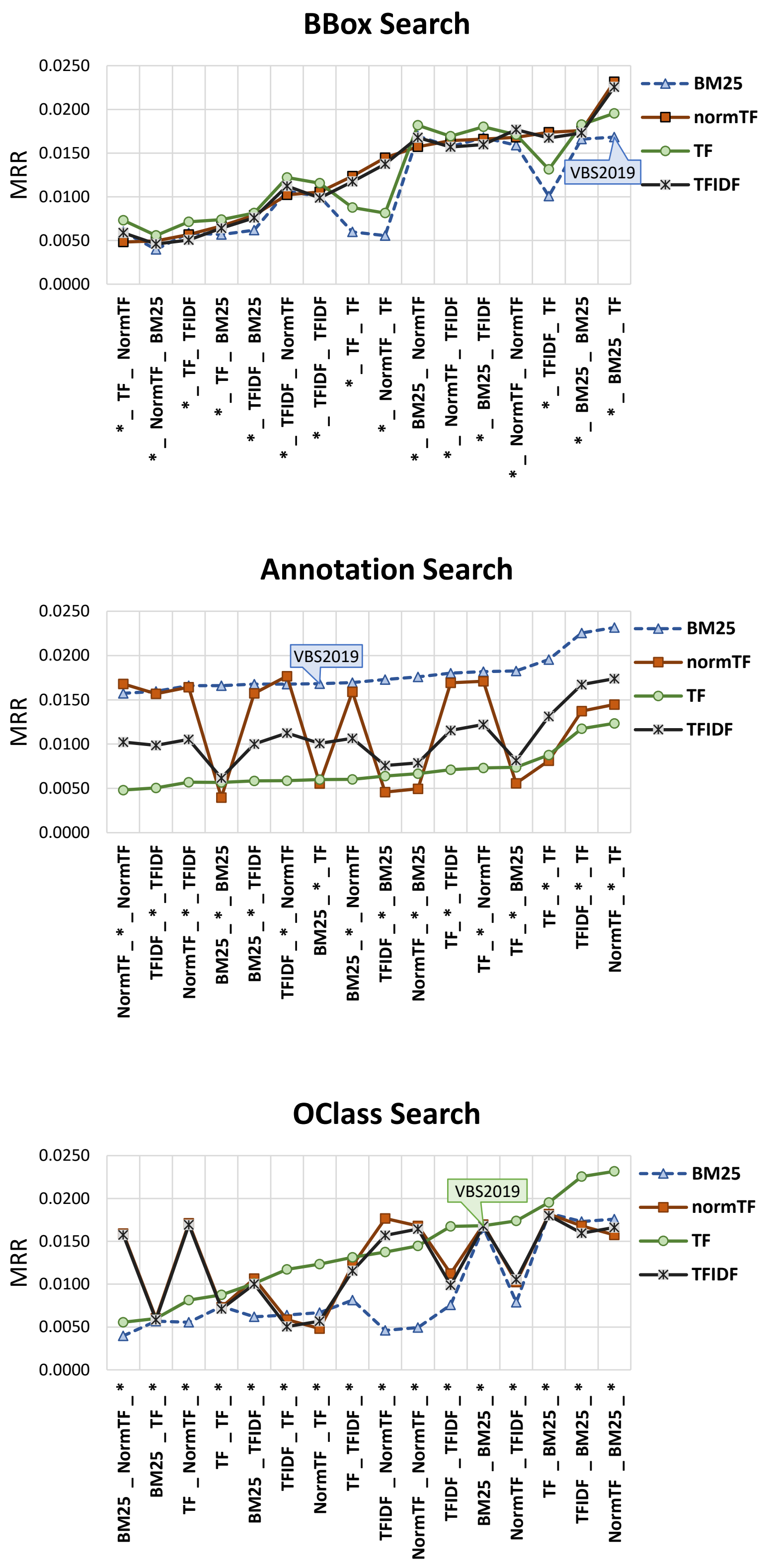

BBox Search
MRR
0.0250
0.0200
0.0150
0.0100
0.0050
0.0000
BM25
normTF
TF
TFIDF
VBS2019
* _ TF _ NormTF
* _ NormTF _ BM25
* _ TF _ TFIDF
* _ TF _ BM25
* _ TFIDF _ BM25
* _ TFIDF _ NormTF
* _ TFIDF _ TFIDF
* _ TF _ TF
* _ NormTF _ TF
* _ BM25 _ NormTF
* _ NormTF _ TFIDF
* _ BM25 _ TFIDF
* _ NormTF _ NormTF
* _ TFIDF_ TF
* _ BM25 _ BM25
* _ BM25 _ TF
Annotation Search
MRR
0.0250
0.0200
0.0150
0.0100
0.0050
0.0000
BM25
normTF
TF
TFIDF
VBS2019
NormTF _ * _NormTF
TFIDF _ * _ TFIDF
NormTF _ * _ TFIDF
BM25 _ * _BM25
BM25 _ * _ TFIDF
TFIDF _ * _NormTF
BM25 _ * _ TF
BM25 _ * _ NormTF
TFIDF _ * _BM25
NormTF _ * _BM25
TF _ * _ TFIDF
TF _ * _NormTF
TF _ * _ BM25
TF _ * _ TF
TFIDF _ * _ TF
NormTF _ * _ TF
OClass Search
MRR
0.0250
0.0200
0.0150
0.0100
0.0050
0.0000
BM25
normTF
TF
TFIDF
VBS2019
BM25 _NormTF _ *
BM25 _TF _ *
TF _NormTF _ *
TF _TF _ *
BM25 _TFIDF _ *
TFIDF _TF _ *
NormTF _TF _ *
TF _TFIDF _ *
TFIDF _NormTF _ *
NormTF _NormTF _ *
TFIDF _TFIDF _ *
BM25 _BM25 _ *
NormTF _TFIDF _ *
TF _BM25 _ *
TFIDF _BM25 _ *
NormTF _BM25 _ *

Figure 12.

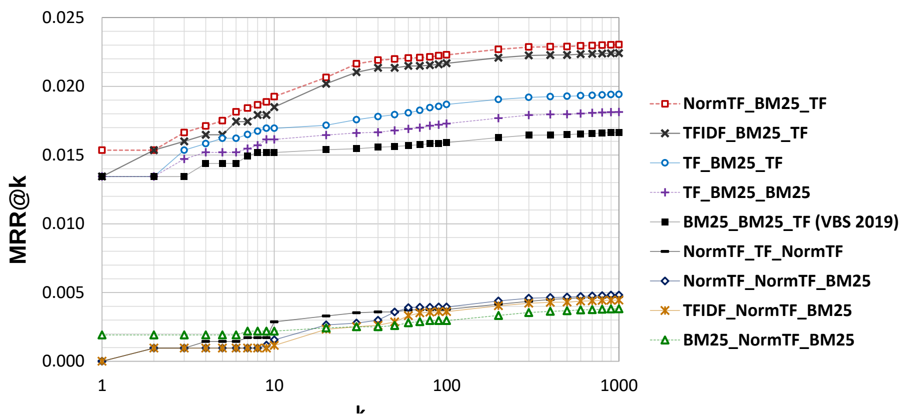


Figure 13.

to the scalability of visual similarity as we rely on the technique used to index R-MAC descriptors based on scalar quantization, the reader is referred to the work [49], in which the scalability of this approach is proven. On the other hand, as far as objects and colors are concerned, we have analyzed the sparsity of the inverted index corresponding to synthetic documents and we have seen that it is around 99.78%. Moreover, since the queries are similar in length to those of natural language search scenarios (i.e. they have few terms), the scalability of the system is guaranteed at least as much as that of full-text search engine scenarios.

6. Conclusions

In this paper, we described a frame-based interactive video retrieval system, named VISIONE, that participated to the Video Browser Showdown contest in 2019. VISIONE includes several retrieval modules and supports complex multi-modal queries, including query by keywords (tags), query by object/color location, and query by visual example. A demo of VISIONE running on the VBS V3C1 dataset is publicly available at http://visione.isti.cnr.it/.

VISIONE exploits a combination of artificial intelligence techniques to automatically analyze the visual content of the video keyframes and extract annotations (tags), information on objects and colors appearing in the keyframes (including the spatial relationship among them), and deep visual descriptors. A distinct aspect of our system is that all these extracted features are converted into specifically designed text encodings that are then indexed using a full-text search engine. The main advantage of this approach is that VISIONE can exploit the latest search engine technologies, which today guarantee high efficiency and scalability.

The evaluation reported in this work shows that the effectiveness of the retrieval is highly influenced by the text scoring function (ranker) used to compare the textual encodings of the video features. In fact, by performing an extensive evaluation of the system under several combinations, we observed that an optimal choice of the ranker used to sort the search results can improve the performance in terms of Mean Reciprocal Rank up to an order of magnitude. Specifically, for our system we found out that *TF*, *NormTF*, and *BM25*, are particularly effective for comparing textual representations of object/color classes, object/color bounding boxes, and tags, respectively.

Author Contributions: Conceptualization, C.G., G.A., and L.V.; methodology, C.G., G.A, and L.V; software, G.A, P.B., F.C., and F.D; validation, C.G.,and L.V.; formal analysis, F.D., and L.V; investigation, C.G., P.B., and L.V.; data curation, C.V., P.B.; writing—original draft preparation, P.B.,F.D.,F.F,C.G.,L.V. and C.V.; writing—review and editing, G.A, F.D., C.G., L.V. and C.V.; visualization, F.D., L.V.;supervision, G.A.; funding acquisition, G.A., F.F., All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by H2020 project AI4EU under GA 825619, by H2020 project AI4Media under GA 951911, by "Smart News: Social sensing for breaking news", CUP CIPE D58C15000270008, by VISECH ARCO-CNR, CUP B56J17001330004, and by "Automatic Data and documents Analysis to enhance human-based processes" (ADA), CUP CIPE D55F17000290009

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: The V3C1 dataset, which consists of 7475 video files, amounting for 1000h of video content (1082659 predefined segments), is publicly available. In order to download the dataset (which is provided by NIST), please follow the instruction repeorted at https://videobrowsershowdown.org/call-for-papers/existing-data-and-tools/

Acknowledgments: We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rossetto, L.; Gasser, R.; Lokoc, J.; Bailer, W.; Schoeffmann, K.; Muenzer, B.; Soucek, T.; Nguyen, P.A.; Bolettieri, P.; Leibetseder, A.; Vrochidis, S. Interactive Video Retrieval in the Age of Deep Learning - Detailed Evaluation of VBS 2019. *IEEE Transactions on Multimedia* **2020**, pp. 1–1. doi:10.1109/TMM.2020.2980944.
2. Cobârzan, C.; Schoeffmann, K.; Bailer, W.; Hürst, W.; Blažek, A.; Lokoč, J.; Vrochidis, S.; Barthel, K.U.; Rossetto, L. Interactive video search tools: a detailed analysis of the video browser showdown 2015. *Multimedia Tools and Applications* **2017**, *76*, 5539–5571. doi:10.1007/s11042-016-3661-2.
3. Lokoč, J.; Bailer, W.; Schoeffmann, K.; Muenzer, B.; Awad, G. On influential trends in interactive video retrieval: Video Browser Showdown 2015-2017. *IEEE Transactions on Multimedia* **2018**, *20*, 3361–3376. doi:10.1109/TMM.2018.2830110.
4. Berns, F.; Rossetto, L.; Schoeffmann, K.; Beecks, C.; Awad, G. V3C1 Dataset: An Evaluation of Content Characteristics. Proceedings of the 2019 on International Conference on Multimedia Retrieval; Association for Computing Machinery: New York, NY, USA, 2019; ICMR '19, pp. 334—-338. doi:10.1145/3323873.3325051.
5. Amato, G.; Bolettieri, P.; Carrara, F.; Debole, F.; Falchi, F.; Gennaro, C.; Vadicamo, L.; Vairo, C. VISIONE at VBS2019. MultiMedia Modeling; Springer International Publishing: Cham, 2019; pp. 591–596. doi:10.1007/978-3-030-05716-9_51.
6. Frome, A.; Corrado, G.S.; Shlens, J.; Bengio, S.; Dean, J.; Ranzato, M.; Mikolov, T. Devise: A deep visual-semantic embedding model. Advances in neural information processing systems, 2013, pp. 2121–2129.
7. Kiros, R.; Salakhutdinov, R.; Zemel, R.S. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539* **2014**.
8. Karpathy, A.; Joulin, A.; Fei-Fei, L.F. Deep fragment embeddings for bidirectional image sentence mapping. Advances in neural information processing systems, 2014, pp. 1889–1897.
9. Dong, J.; Li, X.; Snoek, C.G. Word2visualvec: Image and video to sentence matching by visual feature prediction. *arXiv preprint arXiv:1604.06838* **2016**.
10. Miech, A.; Zhukov, D.; Alayrac, J.B.; Tapaswi, M.; Laptev, I.; Sivic, J. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. Proceedings of the IEEE international conference on computer vision, 2019, pp. 2630–2640.
11. Mithun, N.C.; Li, J.; Metze, F.; Roy-Chowdhury, A.K. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, 2018, pp. 19–27.
12. Otani, M.; Nakashima, Y.; Rahtu, E.; Heikkilä, J.; Yokoya, N. Learning joint representations of videos and sentences with web image search. European Conference on Computer Vision. Springer, 2016, pp. 651–667.
13. Pan, Y.; Mei, T.; Yao, T.; Li, H.; Rui, Y. Jointly modeling embedding and translation to bridge video and language. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4594–4602.
14. Xu, R.; Xiong, C.; Chen, W.; Corso, J.J. Jointly Modeling Deep Video and Compositional Text to Bridge Vision and Language in a Unified Framework. AAAI. Citeseer, 2015, Vol. 5, p. 6.
15. Schoeffmann, K. Video Browser Showdown 2012-2019: A Review. 2019 International Conference on Content-Based Multimedia Indexing (CBMI), 2019, pp. 1–4. doi:10.1109/CBMI.2019.8877397.
16. Lokoč, J.; Kovalčík, G.; Münzer, B.; Schöffmann, K.; Bailer, W.; Gasser, R.; Vrochidis, S.; Nguyen, P.A.; Rujikietgumjorn, S.; Barthel, K.U. Interactive Search or Sequential Browsing? A Detailed Analysis of the Video Browser Showdown 2018. *ACM Trans. Multimedia Comput. Commun. Appl.* **2019**, *15*. doi:10.1145/3295663.
17. Lokoč, J.; Kovalčík, G.; Souček, T. Revisiting SIRET Video Retrieval Tool. Multimedia Modeling. MMM 2018. Springer, 2018, Lecture Notes in Computer Science, pp. 419–424. doi:10.1007/978-3-319-73600-6_44.
18. Rossetto, L.; Amiri Parian, M.; Gasser, R.; Giangreco, I.; Heller, S.; Schuldt, H. Deep Learning-Based Concept Detection in vitrivr. MultiMedia Modeling; Springer International Publishing: Cham, 2019; pp. 616–621. doi:10.1007/978-3-030-05716-9_55.

19. Kratochvíl, M.; Veselý, P.; Mejzlík, F.; Lokoč, J. SOM-Hunter: Video Browsing with Relevance-to-SOM Feedback Loop. MultiMedia Modeling; Springer International Publishing: Cham, 2020; pp. 790–795. doi:10.1007/978-3-030-37734-2_71.
20. Lokoč, J.; Kovalčík, G.; Souček, T. VIRET at Video Browser Showdown 2020. MultiMedia Modeling; Springer International Publishing: Cham, 2020; pp. 784–789. doi:10.1007/978-3-030-37734-2_70.
21. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.
22. Li, X.; Xu, C.; Yang, G.; Chen, Z.; Dong, J. W2VV++ Fully Deep Learning for Ad-hoc Video Search. Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1786–1794.
23. Sauter, L.; Amiri Parian, M.; Gasser, R.; Heller, S.; Rossetto, L.; Schuldt, H. Combining Boolean and Multimedia Retrieval in vitrivr for Large-Scale Video Search. MultiMedia Modeling; Springer International Publishing: Cham, 2020; pp. 760–765. doi:10.1007/978-3-030-37734-2_66.
24. Rossetto, L.; Gasser, R.; Schuldt, H. Query by Semantic Sketch, 2019, [arXiv:cs.MM/1909.12526].
25. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015, pp. 91–99.
26. Chang, E.Y.; Goh, K.; Sychay, G.; Wu, G. CBSA: content-based soft annotation for multimodal image retrieval using Bayes point machines. *IEEE Trans. Circuits Syst. Video Techn.* **2003**, *13*, 26–38. doi:10.1109/TCSVT.2002.808079.
27. Carneiro, G.; Chan, A.; Moreno, P.; Vasconcelos, N. Supervised Learning of Semantic Classes for Image Annotation and Retrieval. *IEEE transactions on pattern analysis and machine intelligence* **2007**, *29*, 394–410. doi:10.1109/TPAMI.2007.61.
28. Barnard, K.; Forsyth, D. Learning the semantics of words and pictures. Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, 2001, Vol. II, pp. 408–415. doi:10.1109/ICCV.2001.937654.
29. Li, X.; Uricchio, T.; Ballan, L.; Bertini, M.; Snoek, C.G.M.; Bimbo, A.D. Socializing the Semantic Gap: A Comparative Survey on Image Tag Assignment, Refinement, and Retrieval. *ACM Comput. Surv.* **2016**, *49*. doi:10.1145/2906152.
30. Pellegrin, L.; Escalante, H.J.; Montes, M.; González, F. Local and global approaches for unsupervised image annotation. *Multimedia Tools and Applications* **2016**. doi:10.1007/s11042-016-3918-9.
31. Amato, G.; Falchi, F.; Gennaro, C.; Rabitti, F. Searching and annotating 100M Images with YFCC100M-HNfc6 and MI-File. Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing. ACM, 2017, CBMI '17, pp. 26:1–26:4. doi:10.1145/3095713.3095740.
32. Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *CoRR* **2013**, *abs/1310.1531*, [1310.1531].
33. Babenko, A.; Slesarev, A.; Chigorin, A.; Lempitsky, V. Neural codes for image retrieval. Proceedings of 13th European Conference on Computer Vision. Springer, 2014, ECCV 2014, pp. 584–599. doi:10.1007/978-3-319-10590-1_38.
34. Razavian, A.S.; Sullivan, J.; Carlsson, S.; Maki, A. Visual instance retrieval with deep convolutional networks. *arXiv preprint arXiv:1412.6574* **2014**.
35. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2014, CVPR 2014, pp. 580–587. doi:10.1109/CVPR.2014.81.
36. Razavian, A.S.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: an astounding baseline for recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. IEEE Computer Society, 2014, CVPRW 2014, pp. 512–519. doi:10.1109/CVPRW.2014.131.
37. Tolias, G.; Sicre, R.; Jégou, H. Particular object retrieval with integral max-pooling of CNN activations. *CoRR* **2015**, *abs/1511.05879*.
38. Gordo, A.; Almazán, J.; Revaud, J.; Larlus, D. End-to-End Learning of Deep Visual Representations for Image Retrieval. *International Journal of Computer Vision* **2017**, *124*, 237–254. doi:10.1007/s11263-017-1016-8.
39. Gordo, A.; Almazan, J.; Revaud, J.; Larlus, D. End-to-end learning of deep visual representations for image retrieval. *arXiv preprint arXiv:1610.07940* **2016**.
40. Najva, N.; Bijoy, K.E. SIFT and tensor based object detection and classification in videos using deep neural networks. *Procedia Computer Science* **2016**, *93*, 351–358. doi:10.1016/j.procs.2016.07.220.
41. Anjum, A.; Abdullah, T.; Tariq, M.; Baltaci, Y.; Antonopoulos, N. Video stream analysis in clouds: An object detection and classification framework for high performance video analytics. *IEEE Transactions on Cloud Computing* **2016**. doi:10.1109/TCC.2016.2517653.
42. Yaseen, M.U.; Anjum, A.; Rana, O.; Hill, R. Cloud-based scalable object detection and classification in video streams. *Future Generation Computer Systems* **2018**, *80*, 286–298. doi:10.1016/j.future.2017.02.003.
43. Rashid, M.; Khan, M.A.; Sharif, M.; Raza, M.; Sarfraz, M.M.; Afza, F. Object detection and classification: a joint selection and fusion strategy of deep convolutional neural network and SIFT point features. *Multimedia Tools and Applications* **2019**, *78*, 15751–15777. doi:10.1007/s11042-018-7031-0.
44. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR* **2018**, *abs/1804.02767*, [1804.02767].
45. Redmon, J.; Farhadi, A. YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271. doi:10.1109/CVPR.2017.690.
46. Redmon, J.; Farhadi, A. YOLOv3 on the Open Images dataset. https://pjreddie.com/darknet/yolo/, 2018. [Online; accessed 28-February-2019].

47. Gennaro, C.; Amato, G.; Bolettieri, P.; Savino, P. An approach to content-based image retrieval based on the Lucene search engine library. Proceedings of the International Conference on Theory and Practice of Digital Libraries. Springer Berlin Heidelberg, 2010, TPDL 2010, pp. 55–66. doi:10.1007/978-3-642-15464-5_8.
48. Amato, G.; Bolettieri, P.; Carrara, F.; Falchi, F.; Gennaro, C. Large-Scale Image Retrieval with Elasticsearch. Proceeding of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2018, SIGIR 2018, pp. 925–928. doi:10.1145/3209978.3210089.
49. Amato, G.; Carrara, F.; Falchi, F.; Gennaro, C.; Vadicamo, L. Large-scale instance-level image retrieval. *Information Processing & Management* **2019**, p. 102100. doi:https://doi.org/10.1016/j.ipm.2019.102100.
50. Amato, G.; Carrara, F.; Falchi, F.; Gennaro, C. Efficient Indexing of Regional Maximum Activations of Convolutions using Full-Text Search Engines. Proceedings of the ACM International Conference on Multimedia Retrieval. ACM, 2017, ICMR 2017, pp. 420–423. doi:10.1145/3078971.3079035.
51. Thomee, B.; Shamma, D.A.; Friedland, G.; Elizalde, B.; Ni, K.; Poland, D.; Borth, D.; Li, L.J. YFCC100M: The New Data in Multimedia Research. *CACM* **2016**, *59*, 64–73. doi:10.1145/2812802.
52. Miller, G. *WordNet: an electronic lexical database*; Language, speech, and communication, MIT Press, 1998.
53. Amato, G.; Gennaro, C.; Savino, P. MI-File: Using inverted files for scalable approximate similarity search. *Multimedia Tools and Applications* **2012**, pp. 1–30. doi:10.1007/s11042-012-1271-1.
54. Truong, T.D.; Nguyen, V.T.; Tran, M.T.; Trieu, T.V.; Do, T.; Ngo, T.D.; Le, D.D. Video Search Based on Semantic Extraction and Locally Regional Object Proposal. MultiMedia Modeling. MMM 2018. Springer, 2018, Lecture Notes in Computer Science, pp. 451–456. doi:10.1007/978-3-319-73600-6_49.
55. Rubner, Y.; Guibas, L.; Tomasi, C. The Earth Mover"s Distance, MultiDimensional Scaling, and Color-Based Image Retrieval. Proceedings of the ARPA image understanding workshop., 1997, Vol. 661, p. 668.
56. Shang, W.; Sohn, K.; Almeida, D.; Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units. Proceedings of the 33rd International Conference on Machine Learning. JMLR.org, 2016, Vol. 48, *ICML 2016*, pp. 2217–2225.
57. Robertson, S.E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M.; Gatford, M. Okapi at TREC-3. Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994. National Institute of Standards and Technology (NIST), 1994, Vol. 500-225, *NIST Special Publication*, pp. 109–126.
58. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* **1972**, *28*, 11–21.