Searching 100M Images by Content Similarity

Paolo Bolettieri[1], Fabrizio Falchi[1], Claudio Lucchese[1], Yosi Mass[2], Raffaele Perego[1], Fausto Rabitti[1], Michal Shmueli-Scheuer[2]

[1] ISTI-CNR, Pisa, Italy
[2] IBM Haifa Research Lab, Israel

Abstract. In this paper we present the web user interface of a scalable and distributed system for image retrieval based on visual features and annotated text, developed in the context of the SAPIR project. Its architecture makes use of Peer-to-Peer networks to achieve scalability and efficiency allowing the management of huge amount of data and simultaneous access by a large number of users. Describing the SAPIR web user interface we want to encourage final users to use SAPIR to search by content similarity, together with the usual text search, on a large image collection (100 million images crawled from Flickr) with realistic response time. On the ground of the statistics collected, it will be possible, for the first time, to study the user behavior (e.g., the way they combine text and image content search) in this new realistic environment.

1 Introduction: the SAPIR Project

Non-text data, such as images, music, animations, and videos is nowadays a large component of the Web. However, web tools for performing image searching, as the ones provided by Google, Yahoo! and MSN Live Search, simply index the text associated with the image.

Image indexing methods based on content analysis or pattern matching (i.e. features, such as colors and shapes) are usually not exploited at all. In fact, for this kind of data the appropriate search methods are based on similarity paradigms (e.g. range queries and nearest neighbor queries) that are computationally more intensive than text search. The reason is that conventional inverted indexes used for text are not applicable for such data.

The European project SAPIR (Search on Audio-visual content using Peer-to-peer Information Retrieval)[1] aims at breaking this technological barrier by developing a large-scale, distributed Peer-to-Peer infrastructure that will make it possible to search for audio-visual content by querying the specific characteristics (i.e., features) of the content. SAPIR's goal is to establish a giant Peer-to-Peer network, where users are peers that produce audiovisual content using multiple devices (e.g., cell phones) and service providers will use more powerful peers that maintain indexes and provide search capabilities

[1] `http://www.sapir.eu/`

"A picture is worth a thousand words" so using an image taken by a cell phone to find information about e.g. a monument we bump into or singing a melody as a search hint for a full song, combined with optional metadata annotations and user and social networking context will provide the next level of search capabilities and precision of retrieved results.

2 SAPIR Architecture

Although many similarity search approaches have been proposed, the most generic one considers the mathematical metric space as a suitable abstraction of similarity [14]. The metric space approach has been proved to be very important for building efficient indexes for content based similarity searching. A survey of existing approaches for centralized structures (e.g. M-tree), can be found in [14]. However, searching on the level of features eploiting similarity paradigms, typically exploiting range queries and nearest neighbor queries, exhibits linear scalability with respect to the data search size.

Recently scalable and distributed index structures based on Peer-to-Peer networks have also been proposed for similarity searching in metric spaces and are used in the context of the SAPIR project - i.e. GHT*, VPT*, MCAN, M-Chord These index structures have been proved to provide scalability for similarity search adding resources as the dataset grows (see [2] for a comparison of their performances). Peer-to-Peer architectures are convenient approach and a common characteristic is the autonomy of the peers with no need of central coordination or flooding strategies. Since there are no bottlenecks, the structures are scalable and high performance is achieved through parallel query execution on individual peers.

In SAPIR also text is indexed using a Peer-to-Peer architecture called MINERVA [3]. In MINERVA each peer is considered autonomous and has its own local search engine with a crawler and a local index. Posting meta-information into the Peer-to-Peer network the peers share their local indexes. This meta-information contains compact statistics and quality-of-service information, and effectively forms a global directory. The Peer-to-Peer engine uses the global directory to identify candidate peers that are most likely to provide good query results. More information about MINERVA can be found in [3].

An IR-style query language for multimedia content based retrieval has been developed for SAPIR. It exploits the XML representation of MPEG-7 and it is an extension of the "ML Fragments" query language that was originally designed as a Query-By-Example for text-only XML collections. Detailed information can be found in [10].

In SAPIR it is also possible to perform complex similarity search combining result lists obtained using distinct features, GPS information and text. To this aim, state of the art algorithms for combining results are used (e.g., [6]). In Section 4 combined search algorithms and functions are described.

In SAPIR the possibility of retrieving the results of content-based queries from a *cache* located in front of the system has also been investigated [8]. The aim is to reduce the average cost of query resolution, thus boosting the overall performance. The used cache is very different from a traditional cache for WSEs. In fact, our cache is able to return an answer without querying the underlying content-based index in two very different cases: (a) an *exact* answer when exactly the same query was submitted in the past, and its results were not evicted from the cache; (b) an *approximate* answer composed of the closest objects currently cached when the quality of such approximated answer is acceptable according to a given measure. For further information see [8].

For the scope of improving throughput and response time, during the SAPIR project a metric cache was developed [7]. Unlike traditional caching systems, the proposed a caching system might return a result set also when the submitted query object was never seen in the past. In fact, the metric distance between the current and the cached objects is used to drive cache lookup, and to return a set of approximate results when some guarantee on their quality can be given.

3 Dataset: CoPhIR

The collection of images we used consists of a set of 100 million objects randomly selected from the CoPhIR collection[2]. CoPhIR is the largest publicly available collection of high-quality images metadata. Each contains five MPEG-7 visual descriptors (*Scalable Color, Color Structure, Color Layout, Edge Histogram, Homogeneous Texture*), and other textual information (title, tags, comments, etc.) of about 60 million photos (still increasing) that have been crawled from the Flickr photo-sharing site[3].

Since no collection of this scale was available for research purpose, we had to tackle the non-trivial process of image crawling and descriptive feature extraction using the European EGEE computer GRID. In particular, we had the possibility to access the EGEE (Enabling Grids for EsciencE) European GRID infrastructure[4] provided to us by the DILIGENT IST project[5].

4 Combined Search: Algorithms and Functions

Queries in SAPIR can combine both image and text. Top-k queries are used to find the best results that match both a given image and a given text. Given a query it is possible to get from the image index and from the text image a list of objects sorted by descending order of relevance to the appropriate query. Top-k queries are usually done by merging those lists into a single ranked result list using some aggregate function over the objects' scores from the different lists.

[2] `http://cophir.isti.cnr.it` - CoPhIR stands for COntent-based Photo Image Retrieval

[3] `http://www.flickr.com`

[4] `http://www.eu-egee.org/`

[5] `http://www.diligentproject.org/`

4.1 Merge algorithms

The state-of-the-art solution for merging several lists (also known as the top-k problem) is the family of Fagin's TA (Threshold Algorithm) and NRA (No Random Access) algorithms [5]. Although these algorithms have been proved to be instance optimal, their running time can degrade into complete scans of the input lists. Moreover, we show that their basic form is not appropriate for a P2P setting since they may consume high network bandwidth. In this section we describe briefly those algorithms and then describe various optimizations and extensions we developed in SAPIR, in particular:

- P2P Optimizations to TA
- P2P Optimizations to NRA
- Filtered algorithm

P2P Optimizations to TA Inspired by the state-of-the-art algorithms, we implemented Fagin's TA [5] algorithm with several extensions and optimizations. The TA algorithm defines the notion of sorted and random accesses. In sorted access, the next object in the descending order of scores is retrieved from the list associated, whereas, random access retrieves the score of a random given object from the list. A TA algorithm performs a mixture of sorted and random accesses to the lists. At any time during the execution of such an algorithm, there is complete knowledge of the already seen objects. Given m lists, the algorithm starts with sorted access to list i, "sees" object o, and then performs random accesses to the remaining lists to fetch o's score, thus having the complete score for o. In addition, the TA maintains the score of the object at the current cursor position for every list i (denoted as $high_i$). An object whose aggregated score is within the best k already seen objects becomes part of the top-k set. The TA terminates when the object with the lowest score in the top-k set is higher than the threshold value defined as the aggregated score of the $high_i$'s.

We now discuss different optimizations and improvements that we applied on top of the TA algorithm.

Sorted access in Batches The TA as described above considers only costs for sorted and random accesses. However, in a peer-to-peer (P2P) environment, one should not ignore the network and communication overhead. Specifically, the overhead comprises the network latency incurred by message rounds and the network bandwidth consumption incurred by the data exchange among the peers. The abovementioned TA execution in a P2P environment will generate communication message to get the next object as well as performing the random accesses, which can result in high overheads. Thus, the first optimization we applied is to reduce the network overhead by a "fetch in batches" execution. As suggested in [12], to reduce network communication, successive Result Objects can be batched into one message; instead of getting only one object every time that the peer contacts a list, it will receive B sorted objects. To support the batched execution in the SAPIR implementation, one of the parameters for the query execution is the batchSize, the size of the result list that a peer wants to fetch.

Random Access in batches In the original TA, the random accesses are done immediately when a new object is seen, means that a communication message is send to the list after each new object. As discussed above, these communication overheads are expensive. Thus, in our implementation, for each list, the random accesses requests are batched into one array and only one communication message is sent.

P2P Optimizations to NRA We now discuss the NRA algorithm [5]. The main assumption in this algorithm is that no random accesses to the lists are allowed; thus, with sorted only access it needs to determine the k best results. The NRA starts with sorted access to the different lists, in each step it sees the next object. Thus, at any time during the execution some objects may have been only partially seen in a subset of lists, so there is some uncertainty about the final score of the object. The algorithm therefore keeps, for each seen object d, two values to bound its final score: $worstScore(d)$ and $bestScore(d)$. $worstScore(d)$ is computed as the sum of the seen scores of d, assuming a score of 0 for the remaining dimensions, and serves as a lower bound for d's final score. $bestScore(d)$ is computed as the sum of $worstScore(d)$ and the $high_i$ values of lists where d has not yet been seen, where $high_i$ is the value at the current scan position of list i, $bestScore(d)$ is therefore an upper bound for d's final score. Objects are then kept in two sets: The k objects with the currently highest worstScores form the current top-k answers, and the remaining objects reside in the candidates set. The algorithm can safely stop when the object with the highest bestScore of the candidates set has a bestScore that is smaller than the worstScore of the object with the min worstScore from the top-k set. Similar to the TA case, we applied the "Sorted access in Batch" optimization to the NRA. In addition we applied two more optimizations: Bounded Candidate List and Update Upper Bound Once which are described in the following subsections.

Bounded Candidates List As described above, every object o that does not qualify for the top-k set ($worstScore(o) < worstScore(d)$ where d is the object with the min score from the top-k set) and could not be eliminated ($bestScore(o) > worstScore(d)$), is inserted into the candidates set. However, many of these objects have a very low probability to be qualified for the top-k. Keeping all the objects in the candidates set means maintaining a very large set. The cost of maintaining such a set is O(n) which is not suitable for an online algorithm [13]. Thus, as suggested in [13] we can limit the size of the set and keep only the r (typical r could be 200) best candidates.

Update Upper Bound Once The $bestScore(d)$ value is based on the scores for the unseen lists at the current position ($high_i$). When the NRA algorithm scans the next row, the bestScore of all the relevant objects need to be updated. Again, such updates could impose very high overheads on an online algorithm. It is worth noting, that when the query processor gets the results in batches, it can exploit this situation as follows - whenever an object is seen in one of the lists, it is then immediately probed in the other lists with negligible cost. To update the bestScore efficiently, if the object appears in the remaining lists, the worstScore and bestScore are updated. However, if not, then the worstScore is set to 0, and the bestScore is set to the lowest score of the list.

Filtered Algorithm The main purpose of this merge algorithm is to improve the efficiency by considering only the results that were returned by one of the indices and then re-rank or filter out the results by the other index. For example the query can be first sent to the image index and then the returned results are sent to the text index to check if the queried text appears in each of the results. This algorithm does not allow the text to introduce results that did not already appear in the image list.

4.2 Aggregate functions

The majority of top-k techniques assume monotonic aggregation functions. Using monotone aggregation functions is common in many practical applications, especially in web settings [11]. Thus, many top-k processing scenarios involve linear combinations of multiple scoring predicates. Specifically, in SAPIR we have implemented the following functions: Sum, Weighted Sum, Fuzzy AND and Fuzzy OR.

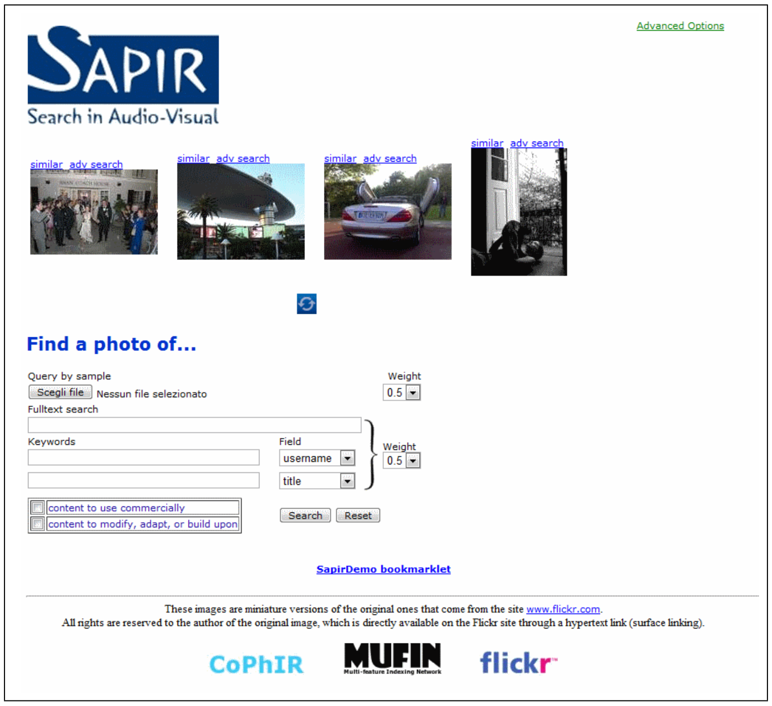
The following aggregation functions were implemented in SAPIR.

- Sum: $\displaystyle\sum_{i=0}^{n-1} x_i$
- Weighted Sum: $\displaystyle\sum_{i=0}^{n-1} w_i \cdot x_i$
- Fuzzy AND: $\min_{i=0}^{n-1}(w_i \cdot x_i)$
- Fuzzy OR: $\max_{i=0}^{n-1}(w_i \cdot x_i)$
- Weighted AND: $\left\{\begin{array}{ll} \sum_{i=0}^{n-1} w_i \cdot x_i, & \text{if } \forall x_i, x_i \neq 0 \\ 0, & \text{else} \end{array}\right\}$

where n is the number of lists, x_i and w_i are the score and the weight of object x in list i correspondingly. It is worth noting that for the Fuzzy AND we only considered the image score.

The main purpose of supporting different aggregation functions is to give the user high flexibility and sometimes improve the effectiveness as follows.

The AND operations namely, fuzzy and weighted AND, are stricter in the sense that they require that the object will appear in all lists. Objects that appear in both lists basically have more "evidence" so that the probability that it is a good object increases. This is very important in the presence of merging content-based and metadata. Previous works [9, 4] suggested that only content-based image search is not effective enough because of the gap between visual feature representations and metadata such as user tagging and extracted semantic concepts. Thus a combination of content based search with associated



These images are miniature versions of the original ones that come from the site www.flickr.com.
All rights are reserved to the author of the original image, which is directly available on the Flickr site through a hypertext link (surface linking).

Fig. 1. SAPIR demo homepage

metadata is expected to yield the best results. Nevertheless, when the user has only a broad idea about the results that she wants and if she can tolerate more fuzziness, then aggregation function such as Weighted Sum and Sum might be more adequate.

5 Guided tour of the tool

For both testing and demonstration, we developed a web user interface to search between indexed images. In the following we briefly describe the web user interface which is public available at `http://sapir.isti.cnr.it/`.

In Figure 1 we report a snapshot of the dynamic web page that is used as starting point for searching. From that page it is possible to perform a fulltext search, a similarity search starting from one of the random selected images, a similarity search starting from an image uploaded by the user or a combined search.

In Figure 2 we report a typical results page from which it is possible to: go back to the home page, access the advanced options described before open a

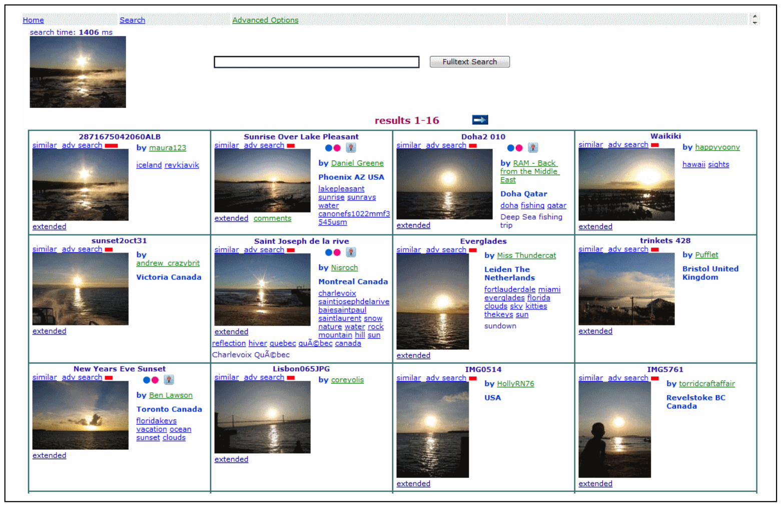
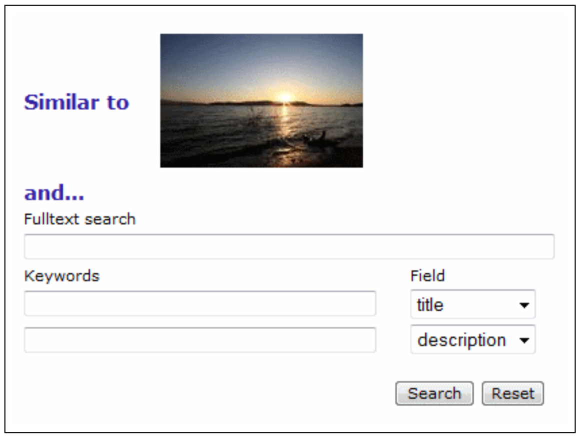


Fig. 2. Results page

window from which it is possible to start with a new query, lunch a new text query. For each result two text links are reported just over the image:

– *similar*: can be used to perform a similarity search with the given result as query. The similarity is evaluated comparing the five MEPG-7 visual descriptor used in CoPhIR. The weight of each descriptor has been fixed following the work reported in [1].
– *adv search*: can be used to access a pop-up window from which it is possible to perform a combined search using both the result as query for similarity and any given text combination as shown below:



For each result displayed the following information is reported:

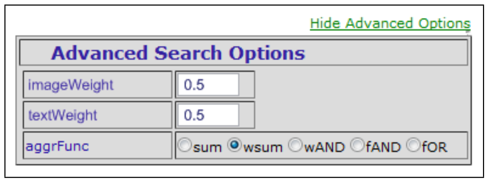
– the image title

– *score*: a red bar visually reports the score assigned to each result
– ●● and 📍 buttons are used to link respectively to Flickr maps and Googlemaps whenever the geographic position during the take is present
– clicking on the result image itself it is possible to access the related Flickr page
– below these buttons we report the author's name
– the location name is reported
– the image tag
– comments can be found following the comments link
– the image description

Finally, at the bottom of the page there is button that can be used to see the next results in order of relevance to the query

The setting of the combined image and text search can be configured in the Advanced option screen. In particular it is possible to set:

1. imageWeight: the weight to give to the image
2. textWeight: the weight to give to the text
3. aggFunc: the aggregate function to be used (for details see Section 4.2)


Hide Advanced Options

Advanced Search Options

imageWeight	0.5
textWeight	0.5
aggrFunc	sum [✓ wsum] wAND fAND fOR

In the SAPIR demo homepage, a link is reported that can be used as a bookmarklet. Adding the bookmarklet to the browser bookmarks, it is possible to use any given image found on any web page as query. In Figure 3a we show the results botained for a text search using Google Images. Clicking on the bookmarklet the images that are on the displayed webpage are reported in a separate page (see Figure 3b). Clicking on one of them, the selected one is used as query and then the results are displayed in Figure 3c.

6 Main Research Results and Future Work

Making this tool available to a large community of user will be important for two main reasons

– From the point of view of search engine technology, it will be the first time that a prototype system based on similarity search for multimedia data is actually used by many users concurrently on such a large image and text collection. We will collect information on the weakness and strength of the system under realistic load.

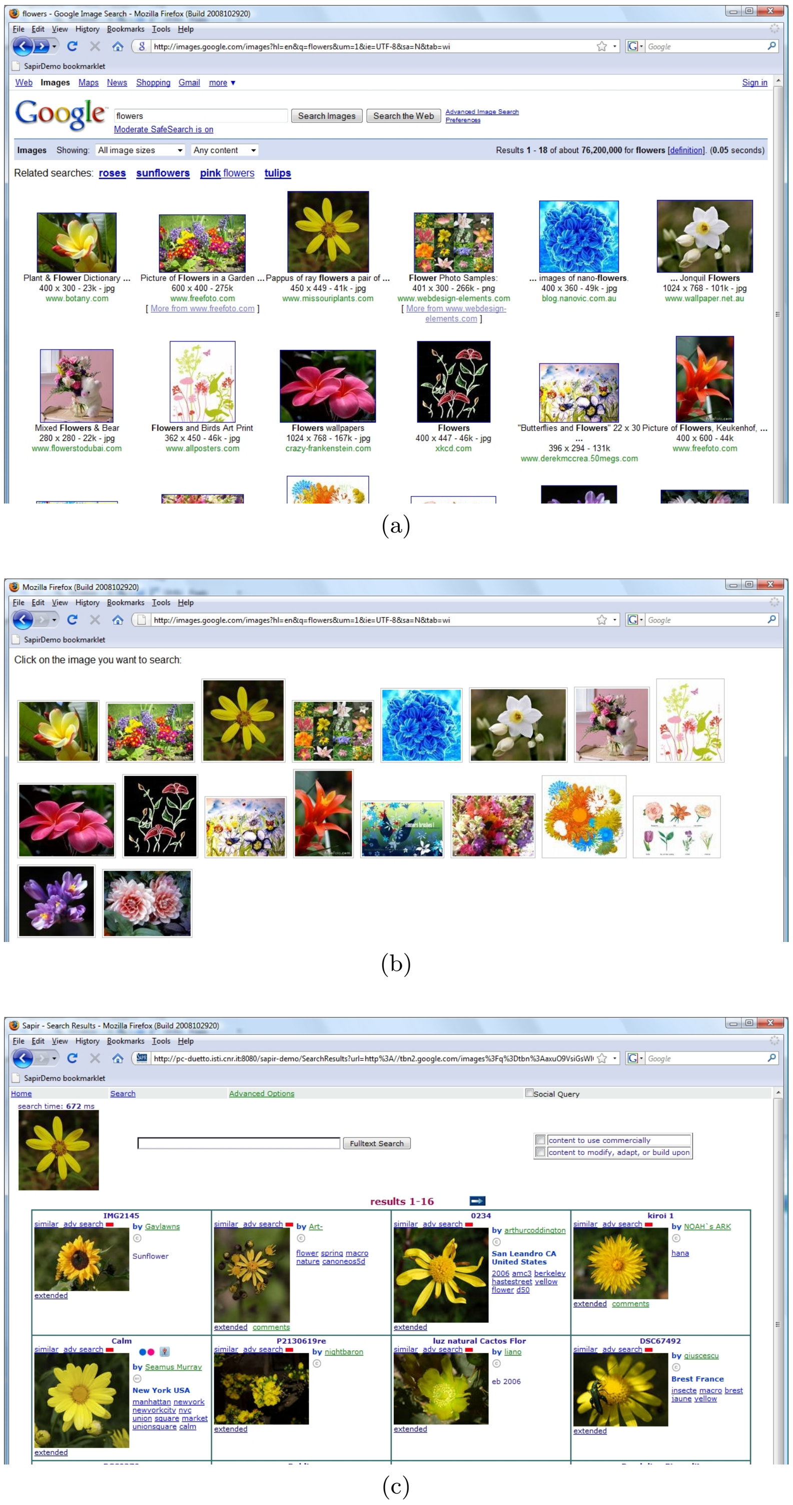

flowers - Google Image Search - Mozilla Firefox (Build 2008102920)
http://images.google.com/images?hl=en&q=flowers&um=1&ie=UTF-8&sa=N&tab=wi
SapirDemo bookmarklet
Search Images
Search the Web
Results 1 - 18 of about 76,200,000 for flowers [definition]. (0.05 seconds)
Related searches: roses sunflowers pink flowers tulips
(a)
Mozilla Firefox (Build 2008102920)
Click on the image you want to search:
(b)
Sapir - Search Results - Mozilla Firefox (Build 2008102920)
http://pc-duetto.isti.cnr.it:8080/sapir-demo/SearchResults?url=http%3A//tbn2.google.com/images%3Fq%3Dtbn%3AaxuO9VsiGsWl
Home
Search
Advanced Options
Social Query
search time: 672 ms
Fulltext Search
content to use commercially
content to modify, adapt, or build upon
results 1-16
similar adv search
extended
comments
(c)

Fig. 3. Bookmarklet usage example

– From the point of view on user experience in searching, it will be the first time that a population of user will have the possibility to make their search using the search by content similarity paradigm (together the usual text search) on a large image collection with realistic response time. We will collect statistics on user behavior (access logs), such as the way they combine text and image content search. This is the first time such experience can be studied in a realistic environment.

Acknowledgments

This work has been partially supported by the SAPIR (Search In Audio Visual Content Using Peer-to-Peer IR) project, funded by the European Commission under IST FP6 (Sixth Framework Programme, Contract no. 45128).

The development and preparation of the demo has involved a large number of people from several SAPIR project partners. In particular we would like to mention: Benjamin Sznajder from IBM Haifa Research Lab; Paolo Bolettieri, Andrea Esuli, Claudio Gennaro, Matteo Mordacchini and Tommaso Piccioli from ISTI-CNR; Michal Batko, Vlastislav Dohnal, David Novak and Jan Sedmidubsky from Masaryk University; Mouna Kacimi and Tom Crecelius from Max-Planck-Institut für Informatik.

References

1. G. Amato, F. Falchi, C. Gennaro, F. Rabitti, P. Savino, and P. Stanchev. Improving image similarity search effectiveness in a multimedia content management system. In *MIS 2004 - 10th International Workshop on Multimedia Information System, College Park, MD, USA, August 25-27*, pages 139–146, 2004.
2. M. Batko, D. Novak, F. Falchi, and P. Zezula. On scalability of the similarity search in the world of peers. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 20, New York, NY, USA, 2006. ACM Press.
3. M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P Search. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1263–1266. VLDB Endowment, 2005.
4. T. Deselaers, T. Weyand, D. Keysers, W. Macherey, and H. Ney. Fire in imageclef 2005: Combining content-based image retrieval with textual information retrieval. In C. Peters, F. C. Gey, J. Gonzalo, H. Müller, G. J. F. Jones, M. Kluck, B. Magnini, and M. de Rijke, editors, *CLEF*, volume 4022 of *Lecture Notes in Computer Science*, pages 652–661. Springer, 2005.
5. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, pages 102–113, New York, NY, USA, 2001. ACM Press.
6. R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. *CoRR*, cs.DB/0204046, 2002.
7. F. Falchi, C. Lucchese, S. Orlando, R. Perego, and F. Rabitti. Caching content-based queries for robust and efficient image retrieval. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 780–790, New York, NY, USA, 2009. ACM.

8. F. Falchi, C. Lucchese, S. Orlando, R. Perego, and F. Rabitti. Caching content-based queries for robust and efficient image retrieval. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint-Petersburg, March 23-26 2009, Proceedings*, ACM International Conference Proceeding Series. ACM, 2009, forthcoming.
9. F. Jing, M. Li, H. Zhang, and B. Zhang. A unified framework for image retrieval using keyword and visual features. *IEEE Transactions on Image Processing*, 14(7):979–989, 2005.
10. J. Mamou, Y. Mass, M. Shmueli-Sheuer, and B. Sznajder. Query language for multimedia content. In *Procedding of the Multimedia Information Retrieval workshop held in conjunction with the 30 th Annual International ACM SIGIR Conference 27 July 2007, Amsterdam*, 2007.
11. A. Marian, N. Bruno, and L. Gravano. Evaluating top- queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.
12. S. Michel, P. Triantafillou, and G. Weikum. Klee: A framework for distributed top-k query algorithms. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, editors, *VLDB*, pages 637–648. ACM, 2005.
13. M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB*, pages 648–659, 2004.
14. P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search. The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer Science + Business Media, Inc., 233 Spring Street, New York, NY 10013, USA, 2006.