SHOW AND TELL: A Seamlessly Integrated Tool For Searching with Image Content And Text

Zhiyong Zhang Carlos Rojas Olfa Nasraoui
Hichem Frigui
Department of Computer Engineering and Computer Science
University of Louisville
Speed School of Engineering, Louisville, KY 40292
{olfa.nasraoui}@louisville.edu

ABSTRACT

In this paper, an image search tool that combines keyword and image content feature *querying* and *search* is presented. The developed search tool tries to bridge the gap between *commercial search engines*, which are based on *keyword* search, and *CBIR* (Content Based Image Retrieval) systems developed mostly in the academic field, designed to search based on image content. The tool is implemented by building on and extending the open source text-based search engine *Nutch* and its powerful *Lucene* based crawling and indexing capabilities. Several user friendly search options are provided to allow users to query the index using not only words, but also by showing an image example, as well as image feature descriptions. Even though we evaluate the developed tool by running a set of controlled experiments on the *COREL' 5000* image database, the developed search tool is able to crawl images from the World Wide Web at a larger scale.

Table 1: Comparison of Keyword and Content Based Search

Type	Framework	Weakness	Strength
Keyword Based	Search Engines	Information Expr-ession Difficulties	Well-Developed Text Retrieval Methods;Scalability;Easy to Use
Content Based	CBIR	Semantic Gap and Scalability	Ability to express the user's information need using content

1. INTRODUCTION

Current large scale image search engines like Google , Yahoo, Altavista, etc. base their image search capabilities mainly on text features associated to pictures, such as file name, image URL, tags, etc, and, marginally, on criteria such as *image size, color vs. "Black and White" images*, or *image format* (such as JPEG, GIF, etc). Even specialized online stock images *Corbis* (http://corbis .com/), *Getty* (http://gettyimages.com/), and *images.com* are also fundamentally keyword based. The use of keywords may be a very effective way to search for images if every image is described exhaustively and the image-searcher has a clear idea of which descriptors to use, given the set of available descriptors. However, keywords that are associated to images can be incomplete and can often be bad descriptors (e.g. an image called *Image01.jpg* does not tell anything about its content; a picture of a boy playing soccer may be named *tiger.jpg* since he was playing '*like a tiger*'). Moreover, the user may have neither a clear description of what she is looking for (other than a picture in her mind), nor the knowledge of all possible good descriptors to choose from given her idea. For all these reasons, the inclusion of image features (color, texture, shape, ..., etc) seems like a reasonable way to improve the image search results.
On the other hand, current research in CBIR (Content Based Image Retrieval) such as *Blobworld* (http://elib .cs.berkeley.edu/ photos/blobworld/), *FIDS* (http://www.aa-lab.cs.uu.nl/cbirsurvey/cbir-survey/node16.html), *SIMBA* (http://simb a.informatik. uni-freiburg.de/), *WEBSEEk* (http://persia.ee. colu-mbia.edu: 8008/), and *Simplicity*(http://wang.ist.psu.ed u/IM-AGE/), ..., etc, have illustrated the merits of exploiting low level image features for image retrieval. However, on *large scale Web resources*, most CBIR systems' performance may become questionable from a scalability point of view. That is, they have neither been designed to, nor can be expected to index too many images compared to what large scale search engines do. Moreover, they generally do not provide keyword based search functionalities which may be useful in certain search scenarios, depending on the collection at hand, and depending on the user's information need. As can be seen from Table 1, the two methods have the potential to complement each other. If combined together into one search tool, then this tool could tap on the combined strengths of a wealth of well-developed text retrieval techniques and the robustness and scalability of current Search Engine technologies to handle image retrieval. They could also provide a hybrid functionality that enables users to express their information need not only in high level textual format, but also in low-level image feature format, and possibly using a query image as example. A combined tool also promises to alleviate the weaknesses of either type of search working in isolation.
In this paper, we present a seamless combination of the two approaches in an integrated image search system, which can be accessed through: *http://webmining.spd.louisville.edu:8080.*

2. RELATED WORK

One of the earliest content based image retrieval systems was *QBIC* [12], which presented a user interface that enables users to organize their own queries by choosing a query image or manipulating image features by drawing a sketch or choosing a color from a color wheel. Later, *Blobworld* [29], [7], [5] combined color, texture, and position features of an image to form the feature vectors and used the Expectation-Maximization (EM) algorithm to segment the image into different regions (*blobs*). The system also allowed users to select relevant blobs and corresponding weights for matching. A more recent system, *SIMPLIcity* [33], [18] is based on an Integrated Region Matching(*IRM*) method. Other content-based image retrieval systems include *WALRUS* [22], which extracts the image features using wavelet transforms and sliding windows, then computes the image similarity by analyzing the region matches; *CIRES* [15], which combines structure with color and texture features for analyzing the images, and perceptual grouping for extracting structure information; *C-BIRD* [20], which allows users to search by illumination invariance and

.

Table 2: Comparison of Different Image Search Systems

System Name	Indexing		Querying		
	textual	content	text	img example	img feature
Google	✓		✓		
QBIC		✓		✓	✓
BlobWorld		✓		✓	
SIMPLicity		✓		✓	
WebSeer	✓	✓	✓		
FIRE	✓	✓		✓	
Webface	✓	✓		✓	
"Show and Tell"	✓	✓	✓	✓	✓

object modeling capabilities; as well as several other systems [30] [19] [13].
All these CBIR systems are based on image content features such as color, texture, shape, structure, ..., etc. Some systems choose to segment the images, and decompose the whole image into smaller regions to perform matches. However, image segmentation itself is still an unsolved problem. Also, these systems seldom take any textual information or image annotations into consideration. One well known reason for missing this important feature is the notorious *semantic gap* between the image content and the image annotations and the fact that different people may have different annotations for the same image due to their different perspectives and domain knowledge. However, this should not lead us to neglect the wealth of information contained in people's description of images using *words*. Another reason why text-based information is often overlooked, may be that most CBIR systems' application domains and image sources tend to be narrow, and they generally do not consider the entire breadth of the World Wide Web as a source of their searchable images. When dealing with images from the World Wide Web, where the image source is not narrow, human tags or annotations can play an important role in differentiating between different images.
Some image retrieval systems have recently started considering their image source from Web collections, and also considering textual information. For example, *WebSeer* [31] combines textual and image content information, although the image content part deals largely with image *metadata* that are contained in the header of an image file (such as image size, creation date etc.). Recently, *FIRE* [11], which was developed for medical image retrieval, also combined textual and image content information. Finally, *Webfaces* [1] was developed as a face detection engine by using both the image textual and content features. However the input query can only be in the form of an image and not text. These and other similar systems [8] [6] combine textual and content information for image retrieval, but the way they use textual information is mostly for building internal relationships or models between a set of training images and corresponding textual information (typically the names of the subjects whose faces are in the images) by using machine learning, classification, clustering, or Latent Semantic Indexing. The textual information in these systems remains hidden to users; that is, users are not allowed to query the system using these textual keywords. Hence, they share limitations that are similar to the above mentioned CBIR systems when it comes to the user interface. Users in these systems are only allowed to query by submitting an example query image [16], or by progressively browsing a set of images [35]. For a comparison, see Table 2.

3. CONTRIBUTIONS

Even though the *indexing* component cannot be de-emphasized when assessing the capabilities of an image retrieval system, the importance of the *querying* component has often been neglected from a quality perspective, and has instead been only emphasized from a user-friendliness or flexibility point of view, along the line of interest in Human Computer Interaction. However, this attitude overlooks the important role that a user's Information Need, and the way that it can be expressed, plays in most real-life search scenarios. Regardless of the effort spent in modeling and indexing information, if the user cannot accurately express their information need, then all these spent efforts risk being in vain. Humans often harness the full potential of their senses (including *visual* senses) to process information, and typically *communicate* this information through *language* or *words*. Hence, *both linguistic* (*text*) and *visual* (*image content*) are inseparable and crucial components for expressing a user's *information need*, and thus for formulating *queries*.

From Table 2, we can see that the earliest systems focused either on the textual part or the image content part, but always exclusively one or the other. For other system that combine text and content together, the final *interface* for a user's query turns out to be either contented based or keyword based, where one feature is exclusive of the other. On the other hand, the proposed system, that we named *"Show and Tell"*, can handle the user's query both by using *text/keywords*, as in current commercial image search engines such as *Google* and *Yahoo*; as well as by using an *example image* or using a description of *image content features*, as in most CBIR systems. Most importantly, the proposed system allows users to input *Boolean queries* that *combine* all these different features together to express a myriad of possible conditions. For example, users can search for images using the combined *word* and image *feature* query "*roses imgcolor:red*" to find pictures of *red roses*, or they can issue the query "*roses -imgcolor:white*" to search for pictures of *roses that do not contain the white color*.
Overall, the following list summarizes our contributions and the functionalities of the proposed tool.

1. We build an image search engine based on modifying the *Lucene* based open source search engine "nutch". In this way, we extend nutch's powerful capability of indexing and searching to image retrieval.

2. We encode the *low level image features* (limited to using a color histogram in this paper) into *text* so that nutch can index and search them. We then combine the color features with textual annotations (or tags) of the image to yield a seamlessly integrated and indexable representations of the images in a text-like domain. Hence, the tool can be used to deal with World Wide Web image collections. This approach can improve current commercial image search engines so that they can handle the content part of images. Also it compensates for the inefficiency of some content-based image retrieval systems by exploiting a full-fledged crawling, indexing and searching scheme.

3. We not only provide the query *by keyword* capability that is used in current commercial image search engines such as *Google*, but also provide the *query by example image* compatibility that is widely used in current CBIR systems. And most importantly, we provide the *query by image feature* (currently illustrated by color features) capability, which is seldom addressed by both of the above two types of image retrieval systems.

4. Our real-time clustering capability can help the user locate the desired images more quickly. By real-time clustering, we mean that the clustering process is performed live (if the user chooses to), after obtaining the search results, in a similar manner to the text-based search engine *Vivissimo*.

5. We have conducted our experiments for different distance measures over 5,000 images from the Corel database (to provide a controlled experimental environment), in order to study the precision and recall of the results.

A demo of the developed tool can be visited through the URL: *http://webmining.spd.louisville.edu:8080*. We organize the paper as follows. In Sections 4 and 5, we introduce our main method of indexing and querying images. Section 6 describes our implementation, while Section 7 gives the experiments and evaluations. Finally Section 8 concludes the paper.

4. INDEXING IMAGES

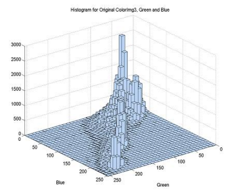
Similar to many other image retrieval systems, we decided to start with color features for image representation because Human eyes are most sensitive to this feature. Also, the color histogram is easy to compute, and tends to be robust against image transformations such as shift and rotation. Even though we currently index only color features, it should be clear that this framework could be extended to other features, such as texture and shape, which are left for future work.

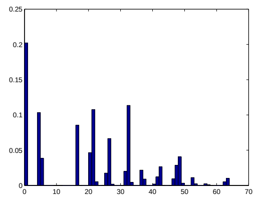
4.1 Colorwords

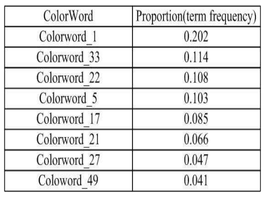
Current indexing and retrieval techniques for text documents, such as Web pages, pdf documents, etc, are nearly full-fledged. Most of them use an inverted list for indexing. That is, they decompose the documents into keywords, and build the index upon such keywords. The general ranking of the returned results is generally based on the *TF-IDF* method, in addition to Web citation or *link analysis*, and web page analysis. When a user submits a query containing several keywords, the search system would return the documents that contain these keywords.
Inspired by this framework, we decided to encode the image features into *text-like keywords*, and therefore tap on the strengths of the *storing*, *indexing* and *retrieval* techniques used for processing text documents. We then build the index based on the keywords encoded from the image features. Although we currently use only the color histogram for encoding features, the principle can be extended to include other image content features. We name the encoded color feature keywords as "*colorwords*". Just like a web page or pdf document may contain many keywords, an image may contain many colorwords.
As mentioned above, the ranking of resulting documents is usually based on the TF-IDF method. That is, when a user searches for a keyword, if one document contains many such keywords (their term frequency is high), this document would be ranked higher in the returned list for that keyword. In contrast, if the keyword is contained in too many documents, (such as "this", "that", ..., etc), then its IDF value will be low, and it will not play the role of a good descriptor for a document. Hence, its effect on the ranking of the documents will be lower. Similarly, we can transplant such document-based concepts into the image and colorword domain. If an image contains a high frequency of a specific colorword, and a user searches for that specific colorword, then the ranking for that image would be higher. On the other hand, if a colorword, such as one corresponding to background color, is contained in too many images, then it will not be a good descriptor for searching images. Hence, its effect on ranking the resulting images would be lower than other less frequent colorwords
Intuitively, when the user searches for color "*red*", the retrieval system will rank the images that contain more "red" components higher since their "*TF*" value is higher. And if the user issues a boolean query "*violet + black*", the images that contain a given level of "*violet*" would generally be ranked higher than the images that contain a comparable level of "*black*" since generally, more pictures tend to contain "*black*" than "*violet*". hence, the *IDF* value of the colorword "*black*" is expected to be lower than that of "*violet*".

4.2 Color Histogram And their Colorword Representation

The *color histogram* has widely been used in most current CBIR systems since it has proved to be a dominant feature for identifying an image. Although it cannot tell the specific location of an object in an image, it does summarize the general color components of an image and it has the merits of being robust to noise and image transformations such as rotation and shift. Therefore, in our implementation, we use the global color histogram to represent the image content features.
The 3-D (in *RGB* or *Red-Blue-Green* space) color histogram is hard to visualize, but the 2-D color histogram can be visualized as in Figure 1.







ColorWord	Proportion(term frequency)
Colorword_1	0.202
Colorword_33	0.114
Colorword_22	0.108
Colorword_5	0.103
Colorword_17	0.085
Colorword_21	0.066
Colorword_27	0.047
Coloword_49	0.041

Figure 1: From image to *colorwords*: clockwise from top left: original image, color histogram 2-D projection, *colorwords* and their corresponding frequencies (*TF*), and finally color histogram on 64 bins

One important question concerns the number of bins to use. If too many bins are used, the computation will become a burden, while if too few bins are used, some information will be lost. In our implementation, we used 64 (4x4x4) bins with 4 values for each of Red, Blue, and Green color components.
The transformation process is shown in Figure 1. For 64 bins, each bin corresponds to a color index number, which is the centroid of the color indices contained in that bin. After counting the occurrence of the image pixels that fall into each bin, we obtain the 3-D color histogram. We then normalize the histogram so that large pictures and small pictures can be comparable from a color perspective, and we encode these 64 indices into *text* (i.e. *words*). When indexing an image using nutch, we do not use all the 64 bin values of the image since according to our analysis, for an image, only a few bin values of the total 64 values tend to stand out, while most other values are very small in comparison. For our implementation, we used the 8 most important values (colors) of the 64 bin color histogram to represent an image.
When building the index using nutch, to exploit the *TF-IDF* matching mechanism of the search engine, we map the 8 textual index values to integers, so that the index with the smallest value is mapped to 1. For the larger values (that are n times as frequent as the lowest value), we just repeat their colorterms n times when building the index. This trick was done because search engines index only keywords that are *countable*.

4.3 Combining Image Colorwords and Textual Information for Indexing

The image *textual* information or annotations are extracted from World Wide Web image collections by parsing the following components:

1. *Image URL*: Generally, the image URL can contain important information about the annotations of an image. Thus, it may contain the image category and image name information, which provide good image annotations. We exploit *Nutch*'s powerful functionality to parse and tokenize the image URL and extract these image annotations.

2. *Image Anchor Text*: The Anchor text is another important source for getting the annotations, since it provides a description of the link to the image from inside a Web page.

Certainly these sources may not be perfect. For example, an image named by a number cannot yield a useful annotation, though its anchor might do so. Also, the above sources may not be sufficient to extract all the image annotations. We left for future work, the extraction of annotations from the HTML file that contains the image. This may include "*ALT*" tags, more text

Table 3: Image Color Palette

Red	Green	Blue	Cyan	Magenta	Yellow
Brown	Orange	Violet	Gray	Black	White

located in the vicinity of the anchor text, as well as other tags that may be available when images are embedded in XML documents.

Figure 4.3 illustrates how the information obtained after pars-

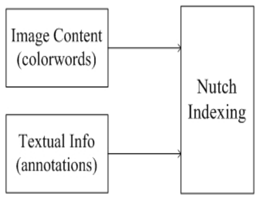


Figure 2: Indexing Images

ing an image contents, i.e. its *colorwords*, and parsing the image *textual* annotations from the parent HTML file, is fed to *Nutch* for building an inverted searchable index.

5. QUERYING THE SEARCHABLE INDEX

Some users may not be inclined to search for one specific colorword. Hence, some user friendly search capabilities need to be provided, as explained below.

5.1 Query by Image Color-Image Color palette

After the colorwords have been extracted from the image and indexed, we need to provide a search function for them. For this end, we need to correlate the colorwords with human beings' linguistic color descriptions such as *red, blue, brown*, ..., etc. The image color description of human beings are individual dependant. For example, some users might describe gray as light black, while others might describe certain dark shades of pink as almost red. For these reasons, we use an *image color palette*[1]. In our color palette, we use 12 colors as shown in Table 3. Notice that the color definitions are not categorical or clear-cut. Each of the above colors forms a group of several similar colorwords and each colorword may belong to multiple groups. For example, for the color red, we not only assign the color with RGB value FF0000 (in our 64bins, it is contained in the bin with RGB value E00000), but also assign the color with RGB value equal to A02020 and 602020, which are colors between brown and dark red. To formulate a query, we use a Boolean logic "OR" so that all the pictures that contain a color that is close to red are retrieved. Here the traditional information retrieval technique *TF-IDF* plays a reincarnated role of matching and ranking according to *textual* and *image content* (except that *image content* has been mapped to *feature words*). Recall that during the indexing process, we repeated a colorword n times of if this colorword occured n times in the image's 3-D color histogram. So if one colorword occurs more frequently, its *TF* value will be large, and therefore, the image will be ranked higher than other images. For example, if a user submits the query "*roses imgcolor:red*", the results that contain the keyword "*roses*" and contain more of the *color red* will be ranked higher than the ones containing less of the red color component.

5.2 Query by Image Example

From Table 2, we can see the Query by image example option is provided by most CBIR systems. In this case, the system should

[1]http://webmining.spd.louisville.edu:8090/images/color palette.html

return results that are ranked by similarity with the query images. Besides the widely used Euclidean distance and Quadratic distance, some other distance measure like Cosine angle distance [23], Earth Mover's Distance, [28] and other distance measures have been explored in several systems [27][24] [32] [34] [21].
For real-time image retrieval, there is a tradeoff between retrieval speed and similarity measure accuracy. For example, when using color histograms, if too few bins are chosen, the image representation will be coarse and may lose significant color information; while if too many bins are chosen, the high dimensionality will impose a big computational burden. For this reason, some methods use adaptive binning [17] and dominant colors or representative color [25] [10] to represent images efficiently while reducing the dimensionality.
Putting our concern in the effectiveness and efficiency of the image search process, we tested three similarity measures: Cosine Angle Distance, Euclidean Distance, and Quadratic Distance.
Since colorwords are used to encode the image color content, they will be exploited to calculate the image similarity. The cosine similarity, which is commonly used to compare term vectors, is given by

$$SIM(Q, D_i) = \frac{\sum_i W_{Q,j} W_{i,j}}{\sqrt{\sum_j W_{Q,j}^2}\sqrt{\sum_i W_{i,j}^2}} \tag{1}$$

where Q is a query image, D is a document (image) relevant to Q and W are weights. The normalized term frequency is used for the weights as follows

$$W_{ij} = TF_{ij}/TF_{i,max}$$

where TF_{ij} is the frequency of term j in Document/image i, and $TF_{i,max}$ is the maximal TF frequency among all terms in Document/image i. During the indexing phase, we encode the 8 most important color components into colorwords. However, when we used the boolean logic "AND" between the colorwords, only one image (the query image) was returned. In this case, it will not be comparable with other results. Hence, we reduced the resolution to 5, which can still give a meaningful number of results for evaluation.
The Cosine similarity tends to benefit from a low computational cost without sacrificing much accuracy. This is especially true because most images contain only several important color components. We have also explored the Euclidean and Quadratic distance measures from the point of view of image ranking and computational efficiency. The Euclidean distance is given by

$$d_{Q,D} = \sqrt{\sum_{k=1}^{n}(Q_k - D_k)^2} \tag{2}$$

Note that if all vectors are normalized, then the ranking result of the Cosine Angle distance and Euclidean distance can easily be shown to be identical. See [23] for further information.
For Quadratic Distance is given by

$$d_A(D, Q) = \sqrt{(D-Q)^T A (D-Q)} \tag{3}$$

Where A is the cross-bin similarity matrix $A = [a_{i,j}]$, and $a_{i,j}$ denotes the rank similarity $|i - j|$ between bin i and bin j.

5.3 Realtime Clustering

Clustering was used in [36],[2], [3] and [9] for different purposes. We use it to obtain a more user friendly interface. The realtime clustering is helpful to quickly locate the images that a user is seeking. For example, if a user inputs a keyword to the search engine and enables the clustering option, he or she will see a few representative images in each cluster (currently, we display all the images contained in each cluster). The user can then click on the image in any of the clusters to refine their query based on that image as example. This will play the role of *zooming* in search. In real applications, it may not be feasible to show all the images retrieved in the first page, but the user may want to get a whole picture of all the results to avoid resorting to the next page blindly.

We use K-means to cluster the image color features. Choosing the number of clusters is beyond the scope of this paper. Hence we chose $k = \sqrt{N}$, for N result images. We use the extracted image tags as an easy way to compute the purity of clusters.

6. IMPLEMENTATION

6.1 A Brief Overview of Nutch

Nutch[2] is an open source search engine application. It uses *Lucene* for the search and indexing component, and a powerful fetcher (crawler robot). Nutch has a highly modular architecture allowing developers to create plugins for the following activities: media-type parsing, data retrieval, querying, and clustering. In June 2003 there was a successful 100 million page demo system. See [4], [14], and [26] for further information.

6.2 Image Sources and Crawling

Nutch has its own robot for crawling the World Wide Web. Currently it can parse various file formats such as html, pdf, rtf, ..., etc, but it does not have a plugin to parse images. Hence, we wrote our own image parsing plugin, *Jpeg-parser*, that can parse .jpg images. During the crawling process, Nutch's crawler fetches and parses web pages from the World Wide Web, and when the specific file types are found, it invokes the corresponding parsers to handle this file. To get the textual information of the images or the images tags, we need to find a method that matches the image annotations to the image. However, the annotations can occur anywhere within an HTML file. For example, the image annotations can be in the image file name, the image URL, the "ALT" text, the anchor text, etc.
First, we started by using the image file name and the URL as image annotations. But for some web sites, the image file name are just numbers and the image URL has no relation with the image content. Therefore, we used only several well formed image repositories: *Washington* database[3], *Bigfoto* http://www.bigfoto.com/, and the SIMPLICity[4] annotated database that has been stored on our server to be crawled on a private port[5] for experiment.
During the crawling process, the HTML parser is invoked to extract the image links, and the jpeg parser will then be used to handle the image links. After crawling, the indexing part is invoked to build an index so that all crawled data becomes searchable. During the indexing period, we only build the index for the jpeg images and skip other file formats.

6.3 Modifying Nutch's Settings and Code for Image Crawling and Indexing

By default, nutch cannot parse images, in order to handle the image indexing and querying, we had to customize it for dealing with images. This includes modifying nutch's configuration file, Indexer, Searcher, and Clustering parts for image searching purposes. One of the most desirable features of nutch is the ability to expand its functionality by writing plugins that are not invasive to its inner indexer and searcher. Hence, we developed the following plugins.

1. *Image Parsing Plugin -JpegParser.* This is to handle jpeg images in the crawling and indexing phase.

2. *Online Clustering Plugin -clustering-imgfeatures.* We added this module to use low level image features for grouping images.

3. *Image Query Plugin -query-imgURL.* In the same spirit as the query by site (*site: keyword*) and query by URL (*URL: keyword*) functionalities, we developed the query by image URL (*imgURL: keyword*) plugin. And it is for this functionality that the customized sorting plays a role.

4. *Image Feature Query Plugin -query-imgcolor.* Based on this plugin, users are allowed to query by image colors.

[2]http://lucene.apache.org/nutch/
[3]http://www.cs.washington.edu/research/imagedatabase/groundtruth/
[4]http://wang.ist.psu.edu
[5]http://webmining.spd.louisville.edu:8085/

Table 4: Query Syntax and Examples

Query Type	Examples
Keyword	*roses*
Image Color	*imgcolor:red*
Image Example	*imgURL:http://xxx.jpg*
+(-)Keyword +(-)Image Color	*roses -imgcolor:red*
+(-)Keyword +Image Example	*roses imgURL:http://xxx.jpg*
+(-)Image Color +Image Example	*imgcolor:red imgURL:http://xxx.jpg*

Note: by default, Boolean "AND" is used

Table 5: Categories and Subcategories of Tested Images

Category	Subcategories
Food	barbecue;vegetable;dining;fruit;seeds
Flower	roses;orchids;cactus;flowerbeds;msc
Vehicle	British car;buses;trains;Rare car;transport
Animals	tigers;elephants;horses;antelope;lion
People	youth;women;men;tribal;model;
City	Bali;London;Hongkong;Hawii;Toronto
Country	Italy;Africa;Canada;Kenya;Egypt
Art	Bonsai;Dinosaur art;Bird art;craft;painting;
Sports	polo;surfing;golf;skiing;msc
Scene	mountain;desert;beach;dawn.dusk;msc

6.4 User Interface And Query Functionalities

Table 4 lists he syntax and examples of six different querying modes implemented so far, including querying by text keywords, image features (currently color), and image example, as well as Boolean combinations thereof. For some snapshots of query examples, see the attached figures.

7. EVALUATION AND EXPERIMENTAL RESULTS

In our evaluation, we used 10 categories, (see Table 5), from the COREL image database, where each category contains 5 semantically coherent sub-categories, Altogether, there are 5000 images in total for testing. Our testing is conducted by combining the keyword (category name) and an image from a subcategory as query. We count the first 10, 20, ...100 result set to see how many result images fall into the same subcategory as the query image and transform this as precision. We chose one image from each subcategory and altogether got 50 queries. We averaged the 50 result sets to avoid random results. Figure 3 shows an example test result distribution for animals. For the animals case the precision is for the first 100 result set and is calculated as follows:

$$\begin{aligned} p'_{average} &= (p'_1 + p'_2 + p'_3 + p'_4 + p'_5)/5 = 1 \\ &= (0.46 + 0.66 + 0.33 + 0.32 + 0.56)/5 = 0.466 \end{aligned} \tag{4}$$

For 50 queries we calculated the average precision by summing all precisions divided by 50. Figure 4 shows the experiment results for three different distance measures. We ran this test on our a linux system, with Intel(R) Xeon(TM) CPU 2.80GHz, 2cpu, with 2G memory. The precision for the Euclidean Distance measure is slightly better than Cosine Angle Distance. The computationally intensive Quadratic Distance measure doesn't exhibit any benefit as expected, although it considers the cross-bin similarity. From a computational speed perspective, Cosine angle measure wins without doubt with an average speed of 0.236s, compared to Euclidean distance (1.837s) and Quadratic Distance(7.104s).

8. CONCLUSIONS AND FUTURE WORK

For image search, keywords can play a filtering role, while image content plays a semantic role for getting better visual results.

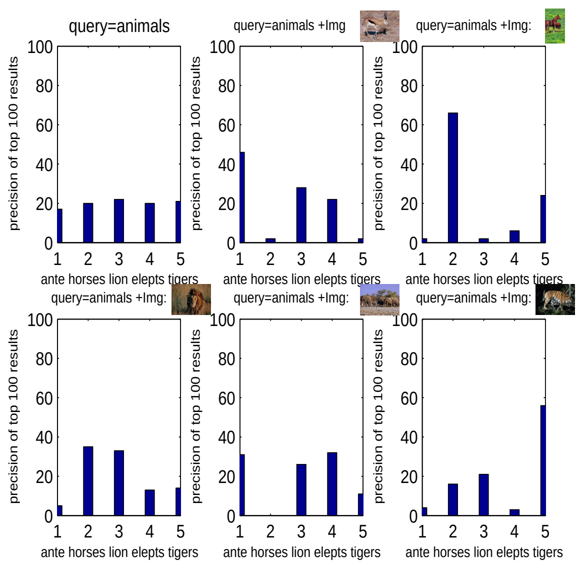


Figure 3: Example Query Result Distribution

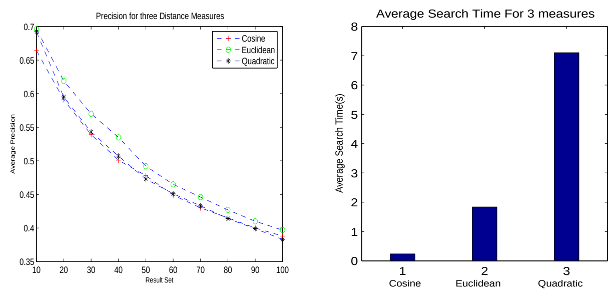


Figure 4: Experiment Result

Using only one of them in a query could lead to false positives. For example, using image *content* query *alone*, a "*rose*" image may be deemed very similar to an image of "*a woman wearing a red hat*", although the two images are distinct. Another example is when a user searches for an image of a tiger by using the *keyword* "*tiger*" as query, and getting as result the image of a man named "*tiger*" such as "*Tiger Woods*". However, the *combination* of the image *keyword* and image *content together* in the same query, would reduce the occurrence of the above false positives.

We have described our development of a search tool benefiting from an integration of text and image content in querying and indexing. This framework promises to inherit the scalability and performance of powerful text search engines. For future work, we consider crawling the world wide web images to get annotations and extending our method to include image texture, shape and other information to improve the performance.

9. REFERENCES

[1] R. Baeza-Yates, J. R. del Solar, R. Verschae, C. Castillo, and C. Hurtado. Content-based image retrieval and characterization on specific web collections. *Conference on Image and Video Retrieval (CIVR), Springer LNCS*, pages 189–198, 2004.

[2] K. Barnard, P. Duygulu, and D. Forsyth. Clustering art. *Computer Vision and Pattern Recognition*, pages 435–439, 2001.

[3] K. Barnard and D. Forsyth. Learning the semantics of words and pictures. *International Conference on Computer Vision*, 2:408–415, 2001.

[4] M. Cafarella and D. Cutting. Building nutch: Open source search. *ACM Queue*, 2(5), April 2004.

[5] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8), August 2002.

[6] C. Carson and V. E. Ogle. Storage and retrieval of feature data for a very large online image collection. *IEEE Computer Society Bulletin of the Technical Committee on Data Engineering*, 19(4), December 1996.

[7] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. *Proc. Third International Conf. Visual Information Systems*, pages 509–516, 1999.

[8] M. L. Cascia, S. Sethi, and S. Sclaroff. Combining textual and visual cues for content-based image retrieval on the world wide web. *Proceedings of the IEEE Workshop on Content - Based Access of Image and Video Libraries(CBAIVL)*, 1998.

[9] Y. Chen, J. Z. Wang, and R. Krovetz. An unsupervised learning approach to content-based image retrieval. *Proc. IEEE International Symposium on Signal Processing and its Applications*, pages 197–200, July 2003.

[10] Y. Deng, B. S. Manjunath, a. M. C. Kenney, and H.Shin. An efficient color representation for image retrieval. *IEEE Transactions on Image Processing*, 10(1):140–147, Jan 2001.

[11] T. Deselaers, T. Weyand, D. Keysers, W. Macherey, and H. Ney. Fire in imageclef 2005: Combining content-based image retrieval with textual information retrieval. *Working Notes of the CLEF Workshop, Vienna, Austria*, September 2005.

[12] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, and et al. Query by image and video content: the qbic system. *IEEE computer*, 28(9):23–32, Sept 1995.

[13] Y. Gong. Advancing content-based image retrieval by exploiting image color and region features. *Multimedia Systems*, 7(6), 1999.

[14] E. H. O. GOSPODNETIC. *Lucene in Action.* Manning Publications Co., Greenwich, CT, 2005.

[15] Q. Iqbal and J. Aggarwal. Cires: A system for content-based retrieval in digital image libraries. pages 205–210. 7 International Conference on Control Automation, Robotics and Vision (ICARCV), December 2002.

[16] V. Kovalev and S. Volmer. Color co-occurrence descriptors for querying-by-example. *Proc. of the 5th Int'l Conf. on Multimedia Modeling*, pages 32–38, October 1998.

[17] W. K. Leow and R. Li. Adaptive binning and dissimilarity measure for image retrieval and classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:234–239, 2001.

[18] J. Li and J. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. on PAMI*, 25(9), 2003.

[19] X. Li, S.-C. Chen, M.-L. Shyu, and B. Furht. An effective content-based visual image retrieval system. *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, 2002.

[20] Z.-N. Li, O. R. Zaiane, and B. Yan. C-BIRD: Content-based image retrieval from digital libraries using illumination invariance and recognition kernel. In *DEXA Workshop*, pages 361–366, 1998.

[21] Q. Lv, M. Charikar, and K. Li. Image similarity search with compact data structures. *In Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM'04), Washington,D.C.*, November 2004.

[22] A. Natsev, R. Rastogi, and K. Shim. WALRUS: a similarity retrieval algorithm for image databases. SIGMOD, Philadelphia, PA, 1999.

[23] G. Qian, S. Sural, Y. Gu, and S. Pramanik. Similarity

between euclidean and cosine angle distance for nearest neighbor queries. *SAC'04*, March 2004.

[24] S. Ravela and R. Manmatha. On computing global similarity in images. *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, 1998.

[25] J. Ren, Y. Shen, and L. Guo. A novel image retrieval based on representative colors. *Image and Vision Computing New Zealand*, 2003.

[26] K. S. A. R. Rohit Khare, Doug Cutting. Nutch: A flexible and scalable open-source web search engine. In *CommerceNet Labs Technical Report 04-04*, November 2004.

[27] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding*, 84(1):25–43, October 2001.

[28] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[29] S.Belongie, C.Carson, H.Greenspan, and J.Malik. Color and texture-based image segmentation using em and its application to content-based image retrieval. *IEEE ICCV*, pages 675–682, Jan 1998.

[30] T. K. SHIH, J.-Y. HUANG, C.-S. WANG, J. C. HUNG, and C.-H. KAO. An intelligent content-based image retrieval system based on color , shape and spatial relations. *Proc. Natl. Sci. Counc. ROC(A)*, 25(4), 2001.

[31] M. J. Swain, C. Frankel, and V. Athitsos. Webseer: An image search engine for the world wide web. *IEEE Computer Vision and Pattern Recognition Conference*, 1997.

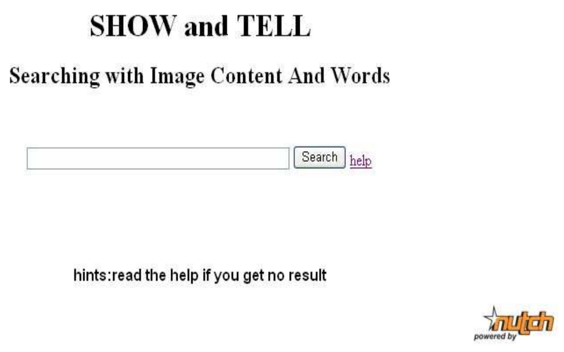
[32] K. Vu, K. A. Hua, and J. Oh. A noise-free similarity model for image retrieval systems. *Proc. of IS&T/SPIE conference on Storage and Retrieval for Media Databases*, pages 1–11, Jan 2001.

[33] J. Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(9), Sept 2001.

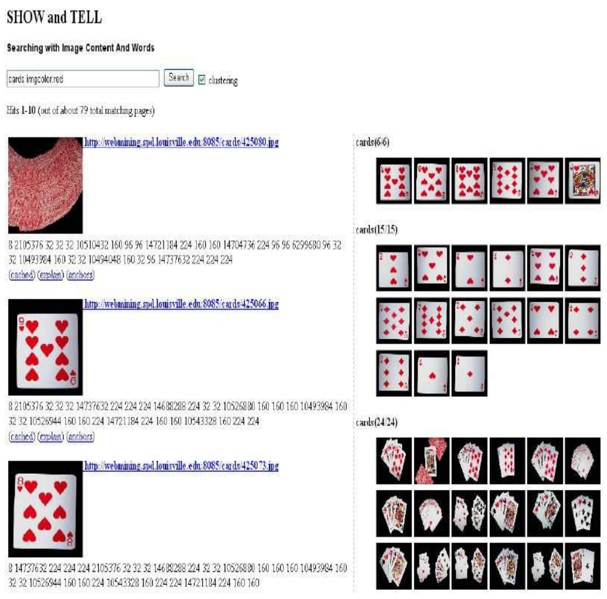
[34] W. Xiaoling and X. Kanglin. Application of the fuzzy logic in content-based image retrieval. *JCS&T*, 5, April 2005.

[35] C. C. Yang. Content based image retrieval: A comparison between query by example and image browsing map approaches. *Journal of Information Science*, 30(3):257–270, 2004.

[36] R. Zhang and Z. M. Zhang. A clustering based approach to efficient image retrieval. *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)*, 2002.


SHOW and TELL
Searching with Image Content And Words
Search
help
hints:read the help if you get no result
powered by nutch

Search Interface


SHOW and TELL

Searching with Image Content And Words

cards imgcolor:red [Search] ☑ clustering

Hits 1-10 (out of about 79 total matching pages)

http://webmining.spd.louisville.edu:8085/cards/425080.jpg

8 2105376 32 32 32 10510432 160 96 96 14721184 224 160 160 14704736 224 96 96 6299680 96 32 32 10493984 160 32 32 10494048 160 32 96 14737632 224 224 224

(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/cards/425066.jpg

8 2105376 32 32 32 14737632 224 224 224 14688288 224 32 32 10526880 160 160 160 10493984 160 32 32 10526944 160 160 224 14721184 224 160 160 10543328 160 224 224

(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/cards/425073.jpg

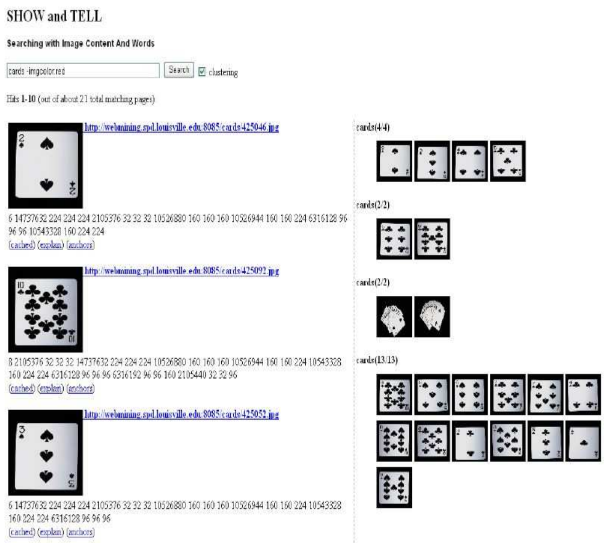
8 14737632 224 224 224 2105376 32 32 32 14688288 224 32 32 10526880 160 160 160 10493984 160 32 32 10526944 160 160 224 10543328 160 224 224 14721184 224 160 160

cards(6/6)

cards(15/15)

cards(24/24)

Query = *cards imgcolor:red*


SHOW and TELL

Searching with Image Content And Words

cards -imgcolor:red [Search] ☑ clustering

Hits 1-10 (out of about 21 total matching pages)

http://webmining.spd.louisville.edu:8085/cards/425046.jpg

6 14737632 224 224 224 2105376 32 32 32 10526880 160 160 160 10526944 160 160 224 6316128 96 96 96 10543328 160 224 224

(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/cards/425092.jpg

8 2105376 32 32 32 14737632 224 224 224 10526880 160 160 160 10526944 160 160 224 10543328 160 224 224 6316128 96 96 96 6316192 96 96 160 2105440 32 32 96

(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/cards/425052.jpg

6 14737632 224 224 224 2105376 32 32 32 10526880 160 160 160 10526944 160 160 224 10543328 160 224 224 6316128 96 96 96

(cached) (explain) (anchors)

cards(4/4)

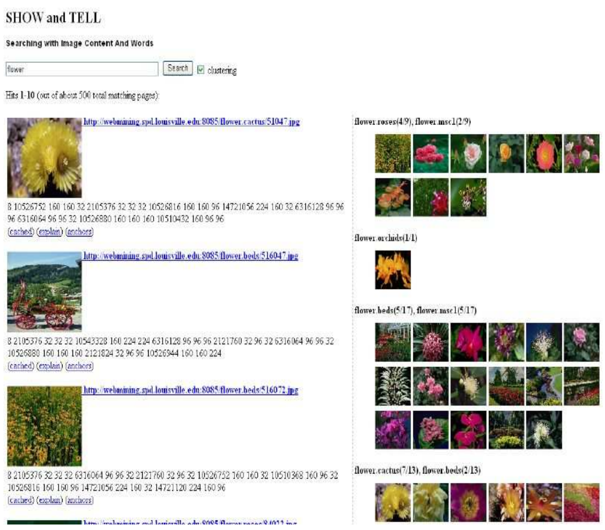
cards(2/2)

cards(2/2)

cards(13/13)

Query = *cards -imgcolor:red*

Figure 5: Interface And Query Example 1



SHOW and TELL

Searching with Image Content And Words

flower Search ☑ clustering

Hits **1-10** (out of about 500 total matching pages):

http://webmining.spd.louisville.edu:8085/flower.cactus/51047.jpg
8 10526752 160 160 32 2105376 32 32 32 10526816 160 160 96 14721056 224 160 32 6316128 96 96 96 6316064 96 96 32 10526880 160 160 160 10510432 160 96 96
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/flower.beds/516047.jpg
8 2105376 32 32 32 10543328 160 224 224 6316128 96 96 96 2121760 32 96 32 6316064 96 96 32 10526880 160 160 160 2121824 32 96 96 10526944 160 160 224
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/flower.beds/516072.jpg
8 2105376 32 32 32 6316064 96 96 32 2121760 32 96 32 10526752 160 160 32 10510368 160 96 32 10526816 160 160 96 14721056 224 160 32 14721120 224 160 96
(cached) (explain) (anchors)

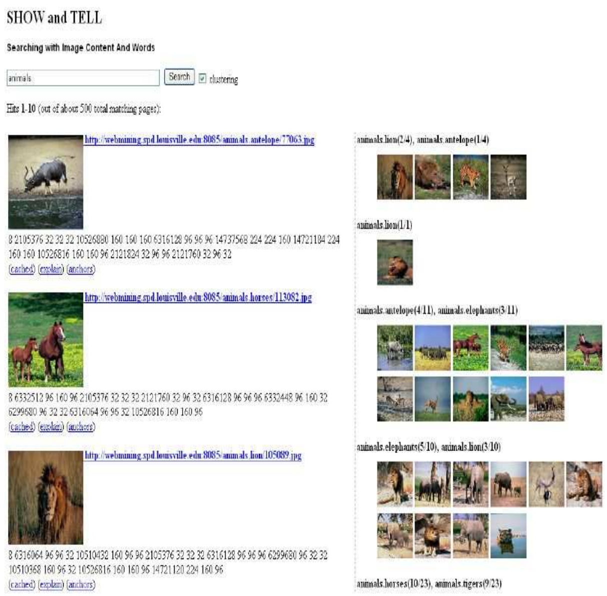
flower.roses(4/9), flower.misc1(2/9)

flower.orchids(1/1)

flower.beds(5/17), flower.misc1(5/17)

flower.cactus(7/13), flower.beds(2/13)

Query = *flower*



SHOW and TELL

Searching with Image Content And Words

animals Search ☑ clustering

Hits **1-10** (out of about 500 total matching pages):

http://webmining.spd.louisville.edu:8085/animals.antelope/77063.jpg
8 2105376 32 32 32 10526880 160 160 160 6316128 96 96 96 14737568 224 224 160 14721184 224 160 160 10526816 160 160 96 2121824 32 96 96 2121760 32 96 32
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/animals.horses/113082.jpg
8 6332512 96 160 96 2105376 32 32 32 2121760 32 96 32 6316128 96 96 96 6332448 96 160 32 6299680 96 32 32 6316064 96 96 32 10526816 160 160 96
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/animals.lion/105089.jpg
8 6316064 96 96 32 10510432 160 96 96 2105376 32 32 32 6316128 96 96 96 6299680 96 32 32 10510368 160 96 32 10526816 160 160 96 14721120 224 160 96
(cached) (explain) (anchors)

animals.lion(2/4), animals.antelope(1/4)

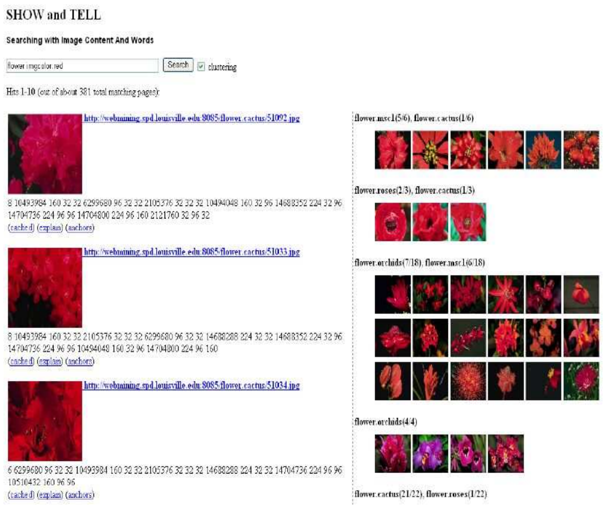
animals.lion(1/1)

animals.antelope(4/11), animals.elephants(3/11)

animals.elephants(5/10), animals.lion(3/10)

animals.horses(10/23), animals.tigers(9/23)

Query = *animals*



SHOW and TELL

Searching with Image Content And Words

flower imgcolor:red Search ☑ clustering

Hits **1-10** (out of about 381 total matching pages):

http://webmining.spd.louisville.edu:8085/flower.cactus/51092.jpg
8 10493984 160 32 32 6299680 96 32 32 2105376 32 32 32 10494048 160 32 96 14688352 224 32 96 14704736 224 96 96 14704800 224 96 160 2121760 32 96 32
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/flower.cactus/51033.jpg
8 10493984 160 32 32 2105376 32 32 32 6299680 96 32 32 14688288 224 32 32 14688352 224 32 96 14704736 224 96 96 10494048 160 32 96 14704800 224 96 160
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/flower.cactus/51034.jpg
6 6299680 96 32 32 10493984 160 32 32 2105376 32 32 32 14688288 224 32 32 14704736 224 96 96 10510432 160 96 96
(cached) (explain) (anchors)

flower.misc1(5/6), flower.cactus(1/6)

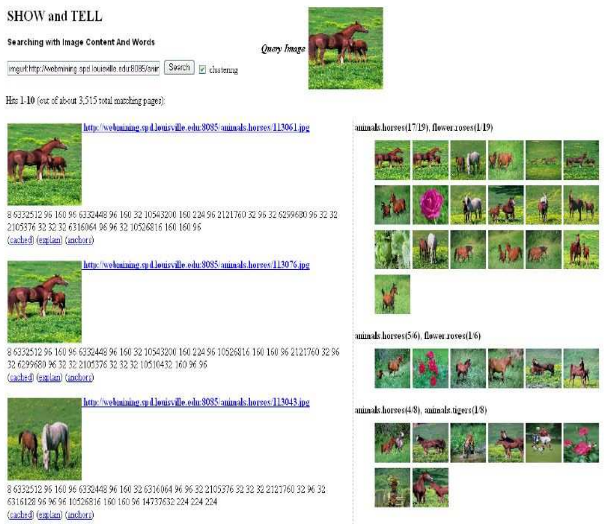
flower.roses(2/3), flower.cactus(1/3)

flower.orchids(7/18), flower.misc1(6/18)

flower.orchids(4/4)

flower.cactus(21/22), flower.roses(1/22)

Query = *flower imgcolor:red*



SHOW and TELL

Searching with Image Content And Words

Query Image

imgurl:http://webmining.spd.louisville.edu:8085/anir Search ☑ clustering

Hits **1-10** (out of about 3,515 total matching pages):

http://webmining.spd.louisville.edu:8085/animals.horses/113061.jpg
8 6332512 96 160 96 6332448 96 160 32 10543200 160 224 96 2121760 32 96 32 6299680 96 32 32 2105376 32 32 32 6316064 96 96 32 10526816 160 160 96
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/animals.horses/113076.jpg
8 6332512 96 160 96 6332448 96 160 32 10543200 160 224 96 10526816 160 160 96 2121760 32 96 32 6299680 96 32 32 2105376 32 32 32 10510432 160 96 96
(cached) (explain) (anchors)

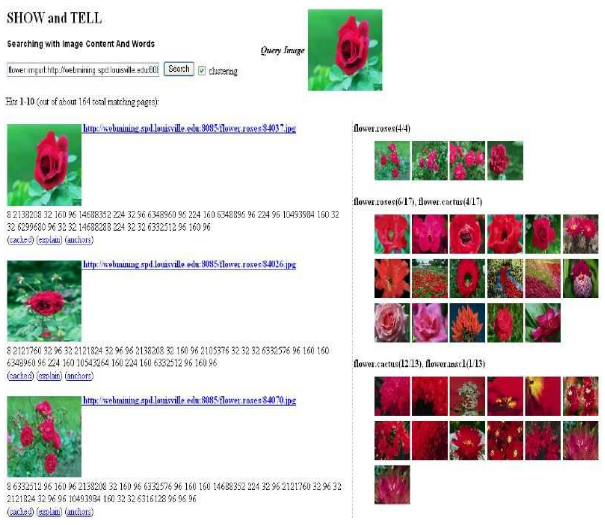
http://webmining.spd.louisville.edu:8085/animals.horses/113043.jpg
8 6332512 96 160 96 6332448 96 160 32 6316064 96 96 32 2105376 32 32 32 2121760 32 96 32 6316128 96 96 96 10526816 160 160 96 14737632 224 224 224
(cached) (explain) (anchors)

animals.horses(17/19), flower.roses(1/19)

animals.horses(5/6), flower.roses(1/6)

animals.horses(4/8), animals.tigers(1/8)

Query = *imgURL:http://webmining.spd.louisville.edu:8085/animals.horses/113061.jpg*



SHOW and TELL

Searching with Image Content And Words

Query Image

flower imgurl:http://webmining.spd.louisville.edu:808 Search ☑ clustering

Hits **1-10** (out of about 164 total matching pages):

http://webmining.spd.louisville.edu:8085/flower.roses/84037.jpg
8 2138208 32 160 96 14688352 224 32 96 6348960 96 224 160 6348896 96 224 96 10493984 160 32 32 6299680 96 32 32 14688288 224 32 32 6332512 96 160 96
(cached) (explain) (anchors)

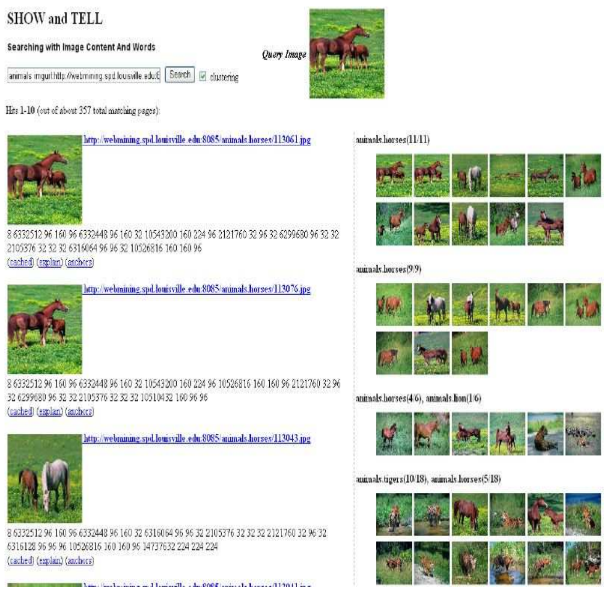
http://webmining.spd.louisville.edu:8085/flower.roses/84026.jpg
8 2121760 32 96 32 2121824 32 96 96 2138208 32 160 96 2105376 32 32 32 6332576 96 160 160 6348960 96 224 160 10543264 160 224 160 6332512 96 160 96
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/flower.roses/84070.jpg
8 6332512 96 160 96 2138208 32 160 96 6332576 96 160 160 14688352 224 32 96 2121760 32 96 32 2121824 32 96 96 10493984 160 32 32 6316128 96 96 96
(cached) (explain) (anchors)

flower.roses(4/4)

flower.roses(6/17), flower.cactus(4/17)

flower.cactus(12/13), flower.misc1(1/13)



SHOW and TELL

Searching with Image Content And Words

Query Image

animals imgurl:http://webmining.spd.louisville.edu:8 Search ☑ clustering

Hits **1-10** (out of about 357 total matching pages):

http://webmining.spd.louisville.edu:8085/animals.horses/113061.jpg
8 6332512 96 160 96 6332448 96 160 32 10543200 160 224 96 2121760 32 96 32 6299680 96 32 32 2105376 32 32 32 6316064 96 96 32 10526816 160 160 96
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/animals.horses/113076.jpg
8 6332512 96 160 96 6332448 96 160 32 10543200 160 224 96 10526816 160 160 96 2121760 32 96 32 6299680 96 32 32 2105376 32 32 32 10510432 160 96 96
(cached) (explain) (anchors)

http://webmining.spd.louisville.edu:8085/animals.horses/113043.jpg
8 6332512 96 160 96 6332448 96 160 32 6316064 96 96 32 2105376 32 32 32 2121760 32 96 32 6316128 96 96 96 10526816 160 160 96 14737632 224 224 224
(cached) (explain) (anchors)

animals.horses(11/11)

animals.horses(9/9)

animals.horses(4/6), animals.lion(1/6)

animals.tigers(10/18), animals.horses(5/18)