RELEVANCE RESPONSE APPROACH FOR IMAGE RETRIEVAL USING DYNAMIC FUSION

PROJECT REPORT

Submitted by

S. DINESH KRISHNAN

Reg. No: 0920108002

In partial fulfillment for the award of the degree of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 049

Department of Computer Science and Engineering

PROJECT WORK

APRIL 2011

This is to certify that the project entitled

RELEVANCE RESPONSE APPROACH FOR IMAGE RETRIEVAL USING DYNAMIC FUSION

is the bonafide record of project work done by

S. DINESH KRISHNAN

Register No: 0920108002

of M.E. (Computer Science and Engineering) during the year 2010-2011.

Project Guide

Head of the Department

Submitted for the Project Viva-Voce examination held on 29.4.2011

Internal Examiner

External Examiner

DECLARATION

I affirm that the project work titled "**RELEVANCE RESPONSE APPROACH FOR IMAGE RETRIEVAL USING DYNAMIC FUSION**" being submitted in partial fulfillment for the award of M.E. degree is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

S. Dinesh Krishnan

DINESH KRISHNAN. S

Register No: 0920108002

I certify that the declaration made above by the candidate is true

R. Dinesh

Mr. R. DINESH, M.S. (Wisconsin),

Assistant Professor

Department of Computer Science and Engineering,
Kumaraguru College of Technology,
(An Autonomous Institution)
Coimbatore-641 049.

EDUCATIONAL AND CHARITABLE TRUST
HECT

TÜV
85 100 001 06700

HINDUSTHAN COLLEGE OF ENGINEERING & TECHNOLOGY

COIMBATORE - 32

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Mr/Ms.....S. DINESHKRISHNAN.....
....KUMARGURU COLLEGE OF TECHNOLOGY.....
has participated and presented a paper entitled..Relevance response approach for ...
..image retrival using dynamic fusion.....
in the National Conference on Recent Trends in Communication, Networking and Computing, held at Hindusthan college of Engineering and Technology, Coimbatore, on March 10th 2011.

Organising Secretary

Convener

Principal

ABSTRACT

Even if relevance response (RR) has been widely revised in the content-based image retrieval, no marketable Web image search engines support RR because of scalability, efficiency, and effectiveness issues. In this paper, we propose a relevance response approach for image retrieval using dynamic fusion. Our approach shows advantage over established RR mechanisms in the following three ways.

First, during the RR process, both textual and visual feature are used in a chronological way. Effortlessly combine textual and visual feature-based RR. Second, the textual feature mechanism employs an effective Fuzzy Cluster algorithm to obtain salient phrases, based on which we could construct an exact and low-dimensional textual space for the resulting images. Last, a new user interface (UI) is proposed to support implicit RR. On the one hand, unlike traditional RR UI which enforces users to make explicit judgment on the results, the new UI regards the users' click-through data as implicit relevance response in order to release burden from the users. And the users better understand the response process.

ஆய்வுச் சுருக்கம்

இணையதளத்தில் பொருத்தமான பின்னூட்டை பயன்படுத்தி இணையபடங்களை மீட்கப்படுகிறது. பரவலாக திருத்தி அமைக்கப்பட்ட பொருத்தமான பின்னூட்டு எந்த வர்த்தக இணையதளமும் பயன்படுத்துவது இல்லை. ஏனென்றால் மாறும் தன்மை, ஆற்றல், திறன் போன்ற இடுவுகள் தான் காரணம். இந்த பாதகத்தை நாம் சாதகமாக மாற்ற மூன்று இயக்க முறையை பயன்படுத்த உள்ளோம்.

முதலில் பொருத்தமான பின்னூட்டில் உரைசார், விழிசார் அம்சங்களை ஒரே நேரத்தில் பயன்படுத்துதல். இரண்டாவது உரைசார், விழிசார் அம்சங்களை ஒன்றாக பயன்படுத்துதல், இதற்கு பஸ்ஸி லாஜிக் வழிமுறையை பயன்படுத்த வேண்டும். இதன்மூலம் சரியான இணையபடங்களை கண்டறிய முடியும். இது, பயன்படுத்துவோரின் வேலையை குறைப்பதோடு மட்டுமல்லாமல் பயனள்ள மிகச் சரியான தகவளை தரும்.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT[English]	iv
	ABSTRACT[Tamil]	v
	ACKNOWLEDGEMENT	vi
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
	LIST OF SYMBOLS	x
1	INTRODUCTION	1
	1.1 General	1
	1.2 Image Processing	2
	1.3 Digital Image	4
	1.3.1 Black And White Image	4
	1.3 .2 Color Image	4
	1.3.3 Grey Scale Image	5
	1.4 Need for Transform	5
	1.5 Image Representation	6
	1.6 Textual Feature	7
	1.7 Visual Feature	8
	1.8 Dynamic Fusion	8
2	LITERATURE REVIEW	9
	2.1 A Content Based Image Retrieval Using Color, Texture and Shape Features	9
	2.2 A Unified Framework for Semantics and Feature Based Image Retrieval	10

	2.3 Interactive Content-Based Image Retrieval Using Relevance Feedback	11
	2.4 Accurately Interpreting Click through Data as Implicit Feedback	12
	2.5 Gaussian Mixture Model For Relevance Feedback In Image Retrieval	13
	2.6 Relevance Feedback A Power Tool for Interactive Content Based Image	14
	2.7 Learning to Cluster Web Search Results	15
3	SYSTEM STUDY	16
	3.1 EXISTING SYSTEM	16
	3.1.1 Problem of Existing System	16
	3.1.1 Drawback in Existing System	16
	3.2 PROPOSED SYSTEM	16
4	SYSTEM ARCHITECTURE	19
	4.1 SYSTEM ARCHITECTURE	19
5	IMPLEMENTATION AND RESULTS	21
6	CONCLUSION AND FURTHER ENHANCEMENTS	25
	APPENDIX -1 CODE SAMPLES	26
	APPENDIX -1 SCREEN SHOTS	44
	REFERENCES	55

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
Fig.1.1	Fundamental steps in image processing	3
Fig.4.1	System Architecture of image retrieval	19
Fig.4.2	Dataflow Diagram	20
Fig.5.2	Textual Feature	22

LIST OF ABBREVIATIONS

TF	TEXTUAL FEATURE
VF	VISUAL FEATURE-BASED RR
FC	FUZZY CLUSTERING
SVM	SUPPORT VECTOR MACHINE
RR	RELEVANCE RESPONSE
CBIR	CONTENT BASED IMAGE RETRIEVAL
DSP	DIGITAL SIGNAL PROCESSING
ADC	ANALOG TO DIGITAL CONVERTER
GMM	GAUSSIAN MIXTURE MODEL

LIST OF SYMBOLS

$\overrightarrow{F^T}$	TEXTUAL FEATURE OF AN IMAGE *I*
ω_i	THE WEIGHT OF THE ith TERM IN *I*'S TEXTUAL SPACE
L	THE NUMBER OF ALL DISTINCT TERMS OF ALL IMAGES TEXTUAL SPACE
N	TOTAL NO OF IMAGES
n_i	THE NO OF IMAGES WHOSE METADATA CONTAINS THE ith TERM
tf_i	THE FREQUENCY OF THE ith TERM IN *I*'S TEXTUAL SPACE

1. INTRODUCTION

1.1 GENERAL

The large numbers of image collections have posed increasing technical challenges to computer systems to store/transmit and index/manage image data effectively. The storage and transmission challenge is tackled by image compression. The challenge to image retrieval is studied in the context of image database and has also been attempted by researchers from a wide range of disciplines from computer vision, image processing to traditional database areas for over a decade. Researchers are discovering that the process of locating a desired image in a large and varied collection can be a source of considerable frustration.

Problems with traditional methods of image indexing have led to the rise of interest in techniques for retrieving images on the basis of automatically-derived features such as color, texture and shape – a technology now generally referred to as Content - Based Image Retrieval (CBIR). After a decade of intensive however, the technology still lacks maturity, and is not yet being used on a significant scale. In the absence of hard evidence on the effectiveness of CBIR in practice, opinion is still sharply divided about their usefulness in handling real-life queries in large and diverse image collections.

A wide range of possible applications for CBIR technology has been identified. Some of these are art galleries, museums, archaeology, architecture/engineering design, geographic information systems, weather forecast, medical imaging, trademark databases, criminal investigations, image search on the Internet. The problems of image retrieval are becoming widely recognized, and the search for solutions is an increasingly active area for research and development. There are mainly two streams of research for image retrieval.

The database community is mainly focusing on image indexing whereas image processing groups are concentrating on representing the image content in the form of

some feature descriptors. Most of the current image indexing practices mainly rely on color, texture or shape features. The performance of the image retrieval technique improves if these features are combined and considered together.

In case of color and texture features combination, the red, green and blue planes are considered separately and then some texture features are extracted from these color planes. The limitation of extracting color texture features is that they represent the individual color planes of image separately. Considering the red, green and blue color planes together to obtain color texture features is the better way of image color feature representation.

1.2 Image Processing

In Image Processing, an image can be defined as a two-dimensional function f(x, y), where x and y are spatial (plane) coordinates, and f represents the intensity or gray level at (x, y) point in the image. When x, y and the amplitude values of f are all finite, discrete quantities, is a digital image.

The field of digital image processing refers to processing digital images by means of a digital computer. The digital images are composed of finite number of elements. Each of the elements has a particular location and value. These elements are referred to as picture elements, image elements and pixels. Digital image processing is now being used in a vast area and is developing very fast. For example, it is used in SAR Images, X-Ray Imaging, and microwave band and so on.

The objective of image enhancement is to process images in order to make the images more suitable for a specific application than the original images. For example, in this project work has more interested in the horizontal lines. Image enhancement methods have two categories: spatial domain methods and frequency domain methods. The wavelet algorithm used belongs to the set of frequency domain methods; and the other two algorithms belong to the set of spatial domain methods.

There are several steps of image processing, as shown in Fig 1.1. The first process is image acquisition. It involves preprocessing. Image enhancement is the simplest and most appealing areas of digital image processing. Image restoration is an area that also deals with improving the appearance of an image. Color image processing is an area that has been gaining in importance because of the significant increase in the use of digital images over the Internet. Image compression deals with techniques for reducing the storage space required saving an image or the bandwidth required transmitting it. Segmentation procedures partition an image into its constituent parts or objects. Recognition is the process that assigns a label to an object based on its descriptors

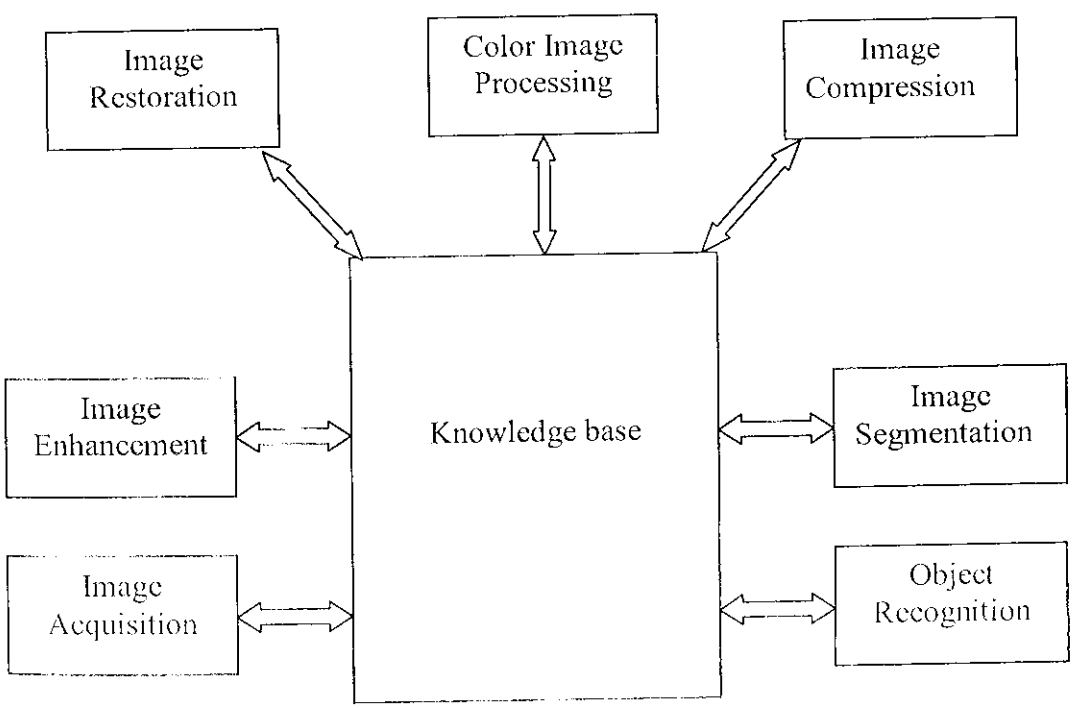

Fig. 1.1 Fundamental steps in image processing

1.3 DIGITAL IMAGE

Digital images are composed of pixels (short for picture elements. Each pixel represents the color (or gray level for black and white photos) at a single point in the image. So a pixel is like a tiny dot of a particular color. By measuring the color of an image at a large number of points, we can create a digital approximation of the image from which the copy of the original can be reconstructed.

1.3.1 BLACK AND WHITE IMAGE

A black and white image is made up of pixels each of which holds a single number corresponding to the gray level of the image at a particular location. These gray levels span the full range from black to white in a series of very fine steps, normally 256 different grays.

Since the eye can barely distinguish about 200 different gray levels, this is enough to give the illusion of a step less tonal scale as illustrated below. Assuming 256 levels, each black and white pixel can be stored in a single byte (8 bits) of memory.

1.3 .2 COLOR IMAGE

A color image is made up of pixels each of which holds three numbers corresponding to the red, green and blue levels of the image at a particular location. Red, green and blue(sometimes referred to as RGB) are the primary colors for mixing light-these so called additive primary colors are different from the subtractive primary colors used for mixing paints(cyan, magenta and yellow).

Any color can be created by mixing the correct amounts of red, green, and blue light. Assuming 256 levels for each primary, each color pixel can be stored in three bytes (24 bits) of memory. This corresponds to roughly 16.7 million different possible colors. Note that for images of the same size, a black and white version will use three times less memory than a color version.

There are several problems with using indexed color to represent photographic images. First, if the image contains more different colors than are in the palette, techniques such as dithering must be applied to represent the missing colors and this degrades the image.

1.3.3 GREY SCALE IMAGE

Grayscale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi-level or binary images). Grayscale images have many shades of gray in between. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

1.4 NEED FOR TRANSFORM

In DSP, engineers study digital signals in one of the following domains, time domain (one dimensional signal), spatial domain (multidimensional signals), frequency domain, autocorrelation domain, and wavelet domains. They choose the domain in which to process a signal by making an educated guess as to which domain best represents the essential characteristics of the signal.

A sequence of samples from a measuring device produces a time or spatial domain representation that is the frequency spectrum. Autocorrelation is defined as the cross-correlation of the signal with itself over varying intervals of time or space. With the increasing use of computers usage and need of digital signal processing has

increased. In order to use an analog to digital converter (ADC). Sampling is usually carried out in two stages, discretization and quantization.

In the discretization stage, the space of signals is partitioned into equivalence classes and discretization is carried out by replacing the signal with representative signal of the corresponding equivalence class. In the quantization stage the representative signal values are approximated by values from a finite set.

FREQUENCY DOMAIN

Signals are converted from time or space domain to the frequency domain usually through the Fourier transform. The Fourier transform of each frequency. Often the Fourier transform is converted to the power spectrum.

The most common purpose for analysis of signals in the frequency domain is analysis of signal properties. The engineer can study the spectrum to get information of which frequencies are present in the input signal and which are missing.

There are some commonly used frequency domain transformations. For example, the spectrum converts signal to the frequency domain through Fourier Transform, takes the logarithm, and then applies another Fourier transform. This emphasizes the frequency components with smaller magnitude while retaining the order of magnitudes of frequency components.

1.5 IMAGE REPRESENTATION

The images collected from several photo forum sites, e.g., photosig have rich metadata such as title, category, photographer's comment and other people's critiques. These images constitute the evaluation dataset for the proposed relevance feedback framework. For example, a photo of photosig1 has the following metadata. In order to facilitate later citation of this photo, we denote it by.

- Title : early morning.
- Category: landscape, nature, rural.
- Comment: I found this special light one early morning in Pyrences along the Vicdessos River near our house.
- One of the critiques: wow I like this picture very much I guess the light has to do with everything the light is great on the snow and on the sky (Strange looking sky by the way) greatly composed nice crafted border a beauty.

$\mathbf{F}^T = (\omega_i, \ldots, \omega_L)$

$\omega_i = tf_i.\ln(N/n_i)$

Where

N is the total number of images;

Ni is the number of images whose metadata contains the ith term

tf_i is the frequency of ith term in I's textual space

L is the number of all distinct terms of all images' textual space

ω_i is the weight of ith term in I's textual space

1.6 TEXTUAL FEATURE

To perform relevance response in textual feature, Rocchio's algorithm is used. The algorithm was developed in the mid-1960s and has been proven to be one of the most effective RF algorithms in information retrieval.

The key idea of Rocchio's algorithm is to construct a so-called optimal query so that the difference between the average score of a relevant document and the average score of a nonrelevant document is maximized. Cosine similarity is used to calculate the similarity between an image and the optimal query.

1.7 VISUAL FEATURE

To perform relevance response in visual feature, Rui's algorithm is used. Assume clicked images to be relevant, both an optimal query and feature weights are learned from the clicked images. More specifically, the feature vector of the optimal query is the mean of all features of clicked images. The weight of a feature dimension is proportional to the inverse of the standard deviation of the feature values of all clicked images. Weighted Euclidean distance is used to calculate the distance between an image and the optimal query.

1.8 DYNAMIC MULTIMODAL FUSION

There has been some work on fusion of relevance feedback in different feature spaces. Nonlinear combination using support vector machine (SVM).Since textual features are more semantic-oriented and efficient than visual features while visual features have finer descriptive granularity than textual features, we combine the relevance response in both feature spaces in a sequential way. The flowchart of the relevance response of our framework. First, relevance response in textual space is performed to rank the initial resulting images. Then, relevance response in visual space is performed to re-rank the top images. The re-ranking process is based on a dynamic linear combination of the relevance response in both visual and textual spaces. Note that restricting the re-ranking only on the top images has two advantages. First, the relevance of the top images could be guaranteed by the former relevance response in textual space. Second, the efficiency of relevance response process could be ensured, for relevance response in visual feature could possibly be inefficient on a very large image set.

2. LITERATURE SURVEY

2.1. TITLE : A Content Based Image Retrieval Using Color, Texture and Shape Features

AUTHOR : Peter N.Yianilos

Novel framework for combining all the three i.e. color, texture and shape information, and achieve higher retrieval efficiency. The image is partitioned into non-overlapping tiles of equal size. The color moments and moments on Gabor filter responses of these tiles serve as local descriptors of color and texture respectively. This local information is captured for two resolutions and two grid layouts that provide different details of the same image.

An integrated matching scheme, based on most similar highest priority (MSHP) principle and the adjacency matrix of a bipartite graph formed using the tiles of query and target image, is provided for matching the images. Shape information is captured in terms of edge images computed using gradient vector flow fields. Invariant moments are then used to record the shape features. The combination of the color, texture and shape features provide a robust feature set for image retrieval. The experimental results demonstrate the efficacy of the method.

2.2. TITLE : A Unified Framework for Semantics and Feature Based Image Retrieval

AUTHOR : F. Jing, M. J. Li, H. J. Zhang

The relevance feedback approach to image retrieval is a powerful technique and has been an active research direction for the past few years. Various ad hoc parameter estimation techniques have been proposed for relevance feedback. In addition, methods that perform optimization on multi-level image content model have been formulated. However, these methods only perform relevance feedback on the low-level image features and fail to address the images' semantic content. We propose a relevance feedback technique, **iFind, to** take advantage of the semantic contents of the images in addition to the low-level features.

By forming a semantic network on top of the keyword association on the images, we are able to accurately deduce and utilize the images' semantic contents for retrieval purposes. The accuracy and effectiveness of our method is demonstrated with experimental results on real-world image collections. To address the limitations of the current relevance feedback systems, we propose a framework that performs relevance feedback on both the images' semantic contents represented by keywords and the low-level feature vectors such as color, texture, and shape.

The contribution of our work is twofold. First, it introduces a method to construct a semantic network on top of an image database and uses a simple machine learning technique to learn from user queries and feedbacks to further improve this semantic network. In addition, we propose a framework in which semantic and low-level feature based relevance feedback can be seamlessly integrated.

2.3. TITLE : Interactive Content-Based Image Retrieval Using Relevance Feedback

AUTHOR : Ingemar J.Cox

Database search engines are generally used in a one-shot fashion in which a user provides query information to the system and, in return, the system provides a number of database instances to the user. A relevance feedback system allows the user to indicate to the system which of these instances are desirable, or relevant, and which are not. Based on this feedback, the system modifies its retrieval mechanism in an attempt to return a more desirable instance set to the user.

We present a relevance feedback technique that uses decision trees to learn a common thread among instances marked relevant. We apply our technique in a preexisting content-based image retrieval (CBIR) system that is used to access high resolution computed tomographic images of the human lung. We compare our approach to a commonly used relevance feedback technique for CBIR, which modifies the weights of a K nearest neighbor retriever.

2.4. TITLE : Accurately Interpreting Click through Data as Implicit Feedback

AUTHOR : T. Joachims, L. Granka, B. Pan, H. Hembrooke

This paper examines the reliability of implicit feedback generated from clickthrough data in WWW search. Analyzing the users' decision process using eyetracking and comparing implicit feedback against manual relevance judgments, we conclude that clicks are informative but biased. While this makes the interpretation of clicks as absolute relevance judgments difficult, we show that relative preferences derived from clicks are reasonably accurate on average.

We explore and evaluate strategies for how to automatically generate training examples for learning retrieval functions from observed user behavior. In contrast to explicit feedback, such implicit feedback has the advantage that it can be collected at much lower cost, in much larger quantities, and without burden on the user of the retrieval system. However, implicit feedback is more difficult to interpret and potentially noisy. In this paper we analyze which types of implicit feedback can be reliably extracted from observed user behavior, in particular click through data in WWW search.

2.5. TITLE : Gaussian Mixture Model For Relevance Feedback In Image Retrieval

AUTHOR: Bo Zhang, Fang Qian, Hong-Jiang Zhang.

Relevance Feedback (RF) has become a powerful technique in content-based image retrieval. Most RF methods assume that positive images follow the single Gaussian distribution, which is not sufficient to model the actual distribution of images due to the gap between the semantic concept and low-level features. Gaussian mixture model (GMM) is applied to represent the distribution of positive images in relevance feedback, and a novel method is proposed to estimate the parameters of GMM.

Both positive and negative examples are used to estimate the number of Gaussian components. Furthermore, due to the lack of training samples, unlabeled data are also incorporated to estimate the covariance matrices. Experimental results show that our GMM-based RF method outperforms that based on single Gaussian model.

2.6. TITLE: Relevance Feedback A Power Tool for Interactive Content Based Image

AUTHOR: T. S. Huang and S. Mehrotra, and M. Ortega, Y. Rui

CBIR has attracted great research attention ranging from government industry to universities. Even ISOIEC has launched a new work item MPEG to done a standard Multimedia Content Description Interface Many special issues from leading journals have been dedicated to CBIR and many CBIR systems both coming digital images video audio graphics and text data.

In order to make use of this vast amount of data techniques to retrieve multimedia information based on its content need to be developed Among the various media types images are of prime importance Not only mercial have been developed recently Despite the extensive research sort the retrieval techniques used in CBIR systems lag behind the corresponding techniques in today's best text search engines such as it is the most widely used media type besides text but it is also one of the most widely used bases for representing and retrieving videos and other multimedia information .

2.7. TITLE: Learning to Cluster Web Search Results

AUTHOR: Matt L.Miller, Thomas P.Minka

Organizing Web search results into clusters facilitates users' quick browsing through search results. Traditional clustering techniques are inadequate since they don't generate clusters with highly readable names. We reformalize the clustering problem as a salient phrase ranking problem. Given a query and the ranked list of documents (typically a list of titles and snippets) returned by a certain Web search engine, our method first extracts and ranks salient phrases as candidate cluster names, based on a regression model learned from human labeled training data. The documents are assigned to relevant salient phrases to form candidate clusters, and the final clusters are generated by merging these candidate clusters. Experimental results verify our method's feasibility and effectiveness.

3. SYSTEM ANALYSIS

3.1 Existing System

In existing most research efforts in this field have focused on designing effective algorithms for traditional relevance response. Given that a CBIR system can collect and store users' relevance response information in a history log. Image retrieval either depends on textual feature-based or visual feature-based.

In the existing commercial Image retrieval systems solely depend on textual information. It's increase user burden. On the one hand, unlike traditional relevance response UI this enforces the users to make explicit judgment on the results.

3.2 Drawback in Existing System

- In traditional method user post their queries and get the result.
- Retrieval accuracy is severely limited.
- Explicit UI supported by user.
- In existing method either support textual based image retrieval or visual based image retrieval. But not support both the features.
- Algorithm efficiency is not good for earlier method.

  The search process is passive i.e., disregarding the information between users and retrieval systems

3.3 Proposed System

We propose a relevance response for Image retrieval. There are three main contributions of the paper. A dynamic multimodal fusion scheme is proposed to seamlessly combine textual feature and visual feature. More specifically, a textual feature algorithm is first used to quickly select a possibly relevant image set. Then, a visual feature algorithm is combined with the textual feature algorithm to further re-rank the resulting Images.

The dynamic fusion technique is used to combine both the feature seamlessly. The textual feature mechanism employs an effective fuzzy clustering algorithm to obtain salient phrases, based on which we could construct an accurate and low-dimensional textual space for the resulting Images. As a result, we could integrate into Image retrieval in a practical way. A new UI is proposed to support implicit relevance response. This recommendation scheme is used to help the user better understand the relevance response given by user.

A common limitation of most of the existing Web image retrieval systems is that their search process is passive, i.e., disregarding the informative interactions between users and retrieval systems. An active system should get the user into the loop so that personalized results could be provided for the specific user. To be active, the system could take advantage of relevance feedback techniques. Relevance feedback, originally developed for information re retrieval in a practical way, an efficient and effective mechanism is required for constructing an accurate and low-dimensional textual space with respect to the resulting Web images.

Although all existing commercial Web image retrieval systems solely depend on textual information, Web images are characterized by both textual and visual features. Image retrieval has the following two characteristics when comparing with text retrieval. In this paper, we propose a unified relevance feedback framework for Web image retrieval. There are three main contributions of the paper.

In this paper, we propose relevance response Web image retrieval. There are three main contributions of the paper.

- A dynamic multimodal fusion scheme is proposed to seamlessly combine textual feature-based RF (TBRF) and visual feature-based RF (VBRF).
- More specifically, a TBRF algorithm is first used to quickly select a possibly relevant image set. Then, a VBRF algorithm is combined with the TBRF algorithm to further re-rank the resulting Web images.

- The fusion of VBRF and TBRF is query concept dependent and automatically learned. The textual feature-based RF mechanism employs an effective search result

clustering (SRC) algorithm.

- A new UI is proposed to support implicit RF.

4. SYSTEM ARCHITECTURE

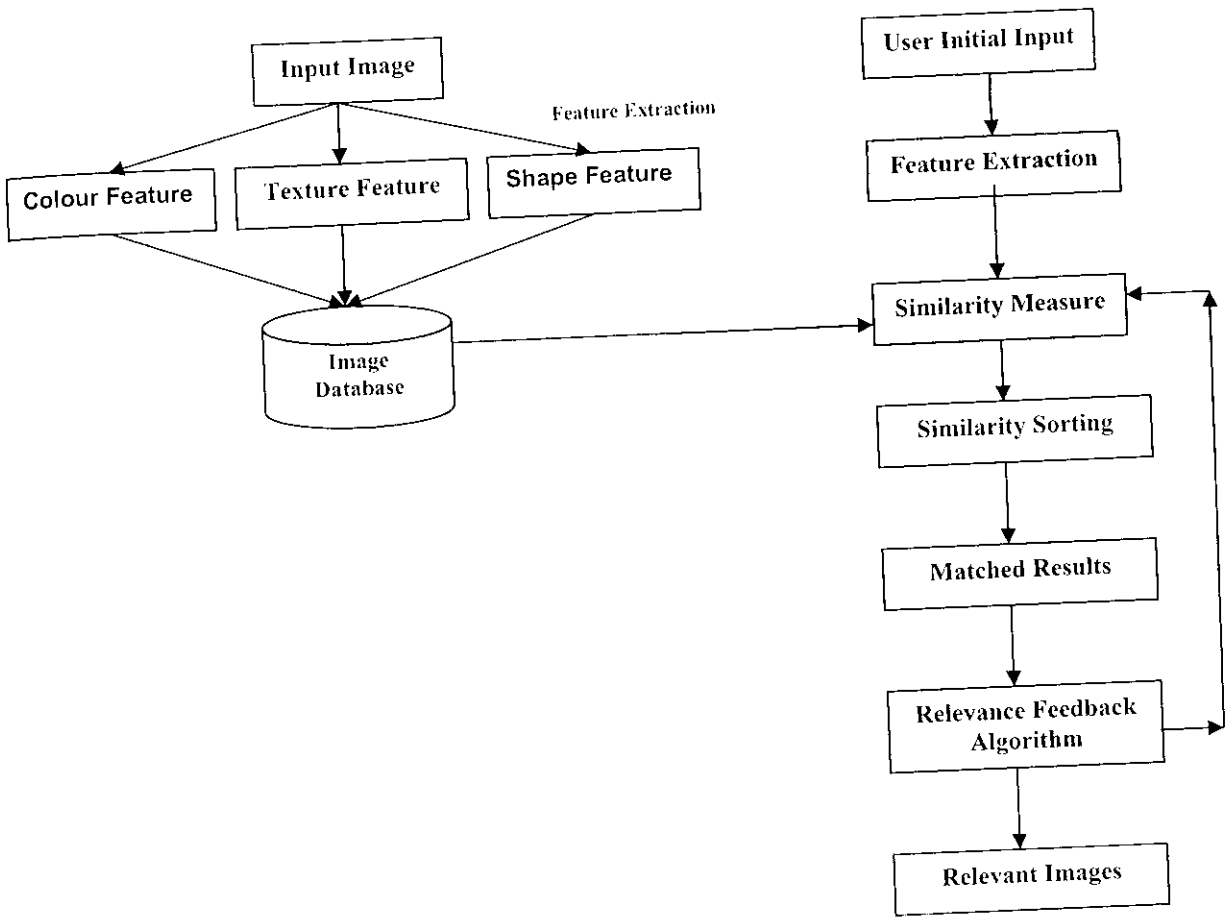

Fig 4.1: System Architecture of image retrieval

4.2 DATAFLOW DIAGRAM:

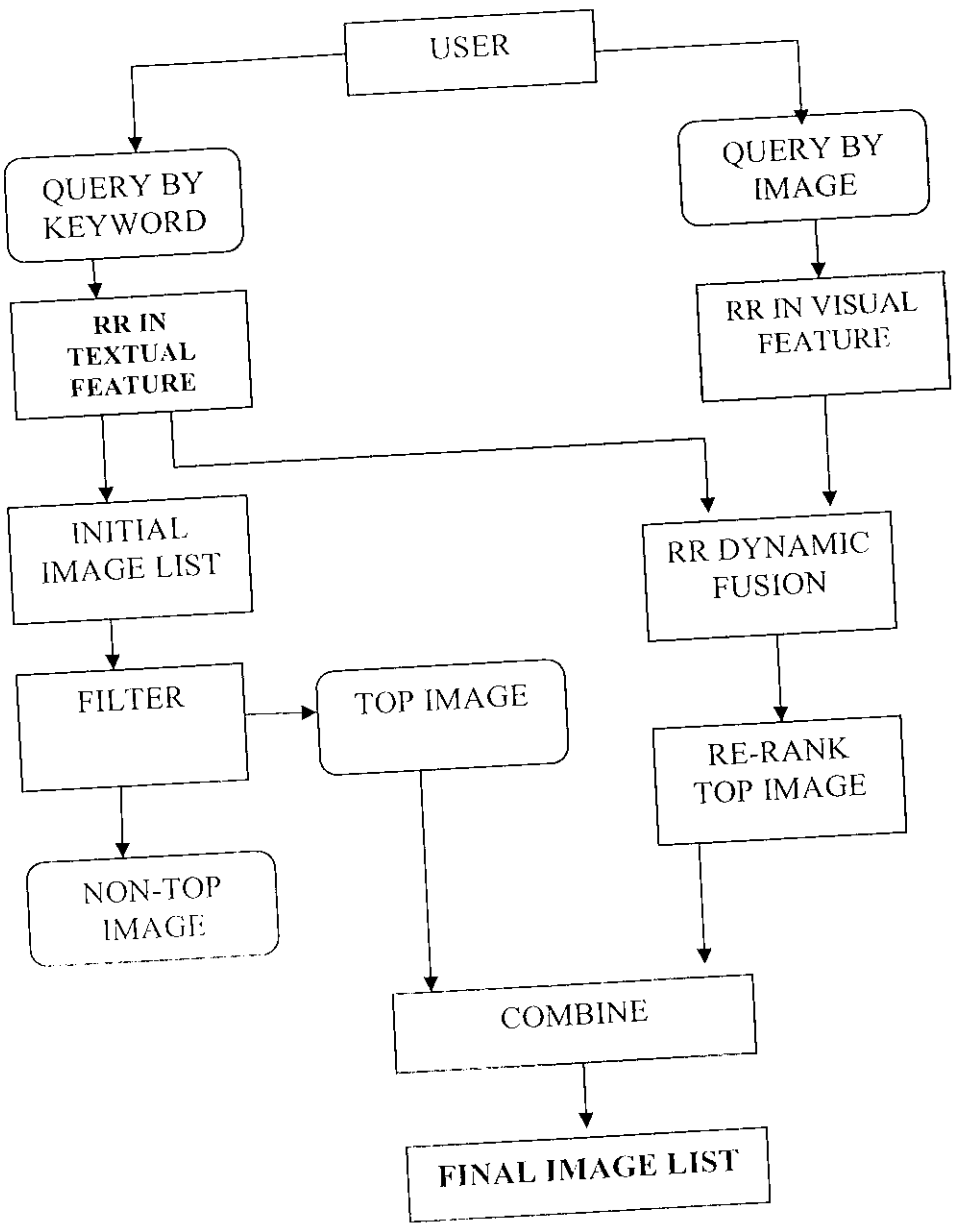

Fig 4.2: Dataflow Diagram

5. IMPLEMENTATION AND RESULTS

In implementation phrase it contains four modules as follows. Since for this kind of retrieval there are no readymade databases. We want to create dataset, which contain collection of images. The modules are

- Image Representation
- Textual Feature
- Visual Feature
    - Texture
    - Color
    - Shape
- Dynamic Fusion

Image Representation:

The images collected from several photo forum sites, e.g., photosig have rich metadata such as title, category, photographer's comment and other people's critiques. These images constitute the evaluation dataset for the proposed relevance feedback framework. For example, a photo of photosig1 has the following metadata. In order to facilitate later citation of this photo, we denote it by.

- Title : early morning.
- Category: landscape, nature, rural.
- Comment: I found this special light one early morning in Pyrences along the Vicdessos River near our house.
- One of the critiques: wow I like this picture very much I guess the light has to do with everything the light is great on the snow and on the sky (Strange looking sky by the way) greatly composed nice crafted border a beauty.

$F^T = (\omega_i , \ldots , \omega_L)$

$\omega_i = tf_i . ln(N/n_i)$

Where

N is the total number of images;

Ni is the number of images whose metadata contains the ith term

tf_i is the frequency of ith term in I's textual space

L is the number of all distinct terms of all images' textual space

ω_i is the weight of ith term in I's textual space

Textual Feature:

To perform relevance response in textual feature, Rocchio's algorithm is used. The algorithm was developed in the mid-1960s and has been proven to be one of the most effective RF algorithms in information retrieval.

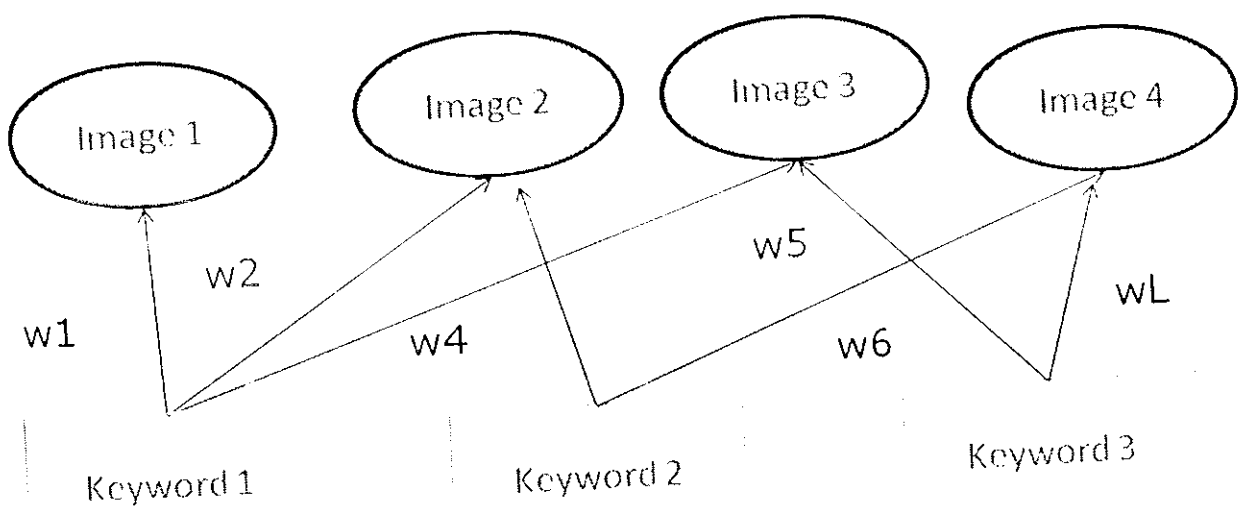

Figure.5.1. Textual Feature

Rocchio Classification is based on a method of relevance feedback found in information retrieval systems which stemmed from the SMART Information Retrieval System around the year 1970. Like many other retrieval systems, the Rocchio feedback approach was developed using the Vector Space Model. The algorithm is based on the assumption that most users have a general conception of which documents should be denoted as relevant or non-relevant. Therefore, the user's search query is revised to include an arbitrary percentage of relevant and non-relevant documents as a means of increasing the search engine's recall, and possibly the precision as well. The number of relevant and non-relevant documents allowed to enter a query is dictated by the weights.

Visual Feature:

Visual feature extraction is the basis of any content-based image retrieval technique. Widely used features include color, texture, shape and spatial relationships. Because of the subjectivity of perception and the complex composition of visual data, there does not exist a single best representation for any given visual feature. Multiple approaches have been introduced for each of these visual features and each of them characterizes the feature from a different perspective.

Color is one of the most widely used visual features in content-based image retrieval. It is relatively robust and simple to represent. Various studies of color perception and color spaces have been proposed, in order to find color-based techniques that are more closely aligned with the ways that humans perceive color. The color histogram has been the most commonly used representation technique, statistically describing combined probabilistic properties of the various color channels (such as (R)ed, (G)reen, and (B)lue channels), by capturing the number of pixels having particular properties. For example, a color histogram might describe the number of pixels of each red channel value in the range [0, 255]. It shows an image and three of its derived color histograms, where the particular channel values are shown along the x-axis, the numbers of pixels are shown along the y-axis, and the particular color channel used is indicated in each histogram. It is well known that histograms lose information related to the spatial distribution of colors and that two very different images can have very similar histograms. There has been much work done in extending histograms to

capture such spatial information. Two of the well-known approaches for this are correlograms and anglograms. Correlograms capture the distribution of colors of pixels in particular areas around pixels of particular colors, while anglograms capture a particular signature of the spatial arrangement of areas (single pixels or blocks of pixels) having common properties, such as similar colors. We note that anglograms also can be used for texture and shape features.

Texture refers to the patterns in an image that present the properties of homogeneity that do not result from the presence of a single color or intensity value. It is a powerful discriminating feature, present almost everywhere in nature. However, it is almost impossible to describe texture in words, because it is virtually a statistical and structural property. There are three major categories of texture-based techniques, namely, probabilistic/statistical, spectral , and structural approaches. Probabilistic methods treat texture patterns as samples of certain random fields and extract texture features from these properties.

The well known features include coarseness, contrast, directionality, line-likeness, regularity, and roughness. Different researchers have selected different subsets of these heuristic descriptors. It is believed that the combination of contrast, coarseness, and directionality best represents the textural patterns of color images.

Shape representation is normally required to be invariant to translation, rotation, and scaling. In general, shape representations can be categorized as either boundary-based or region-based. A boundary-based representation uses only the outer boundary characteristics of the entities, while a region-based representation uses the entire region. Shape features may also be local or global . A shape feature is local if it is derived from some proper subpart of an object, while it is global if it is derived from the entire object.

A combination of the above features are extracted from each image and transformed into a point of a high-dimensional vector space. Using this representation, the many techniques developed by the image retrieval community can be used to advantage. As the dimensionality of the underlying space is still quite high, however, the many disadvantages caused by the curse of dimensionality also prevail.

6. CONCLUSION AND FURTHER ENHANCEMENTS

Textual feature is used to provide possible set of initial image list. Visual feature is used to provide finer set of resultant image list. A dynamic multimodal fusion scheme is proposed to seamlessly combine textual feature-based RF (TBRF) and visual feature-based RF (VBRF). A visual feature algorithm is combined with the textual feature algorithm to further re-rank the resulting Images. To integrate Relevance Response into Image retrieval. Besides explicit relevance feedback, implicit relevance feedback, e.g., click-through data, can also be integrated into the proposed mechanism.

From a practical viewpoint, our approach is also suitable for many applications of the same kind. For example, we could easily design a series of image retrieval systems using this approach. To achieve a better retrieval result, we also require more statistical analyses of past query performance to adjust the system parameters.

It extends to implement this image retrieval concept to neural network. Artificial Intelligence concept is used for machine learning techniques to identify the image for image retrieval process. In future, to reduce multilevel features to single level features. Besides explicit relevance feedback, implicit relevance feedback, e.g., click-through data, can also be integrated into the proposed mechanism. Speech Reorganization based implicit relevance response can be applied in future. It extends to medical and scientific / research field for getting accurate image.

APPENDIX-1: CODE SAMPLES

Indexing thread

```
package liredemo;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ByteArrayInputStream;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.awt.image.BufferedImage;

import net.semanticmetadata.lire.DocumentBuilder;
import net.semanticmetadata.lire.DocumentBuilderFactory;
import org.apache.lucene.analysis.SimpleAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.IndexWriter;
import com.drew.metadata.Metadata;
import com.drew.metadata.MetadataException;
import com.drew.metadata.exif.ExifReader;
import com.drew.metadata.exif.ExifDirectory;
import com.drew.imaging.jpeg.JpegProcessingException;

import javax.imageio.ImageIO;

import javax.swing.JOptionPane;
import liredemo.indexing.ParallelIndexer;


public class IndexingThread extends Thread {
```

```
LireDemoFrame parent;
/** Creates a new instance of FlickrIndexingThread
 * @param parent
 */
public IndexingThread(LireDemoFrame parent) {
    this.parent = parent;
}

// TODO: make parallel
public void run() {
    DecimalFormat df = (DecimalFormat) DecimalFormat.getInstance();
    df.setMaximumFractionDigits(0);
    df.setMinimumFractionDigits(0);
    try {
        parent.progressBarIndexing.setValue(0);
        java.util.ArrayList<java.lang.String> images =
                getAllImages(
                new java.io.File(parent.textfieldIndexDir.getText()), true);
        if (images==null) {
            JOptionPane.showMessageDialog(parent, "Could not find any files in " +
parent.textfieldIndexDir.getText(), "No files found",
JOptionPane.WARNING_MESSAGE);
            return;
        }
        IndexWriter iw = new IndexWriter(parent.textfieldIndexName.getText(), new
SimpleAnalyzer(), !parent.checkBoxAddToExisintgIndex.isSelected());
        int builderIdx = parent.selectboxDocumentBuilder.getSelectedIndex();
        DocumentBuilder builder =
DocumentBuilderFactory.getFullDocumentBuilder();
        int count = 0;
        long time = System.currentTimeMillis();
        Document doc;
```

```
ParallelIndexer indexer = new ParallelIndexer(images, builder);
new Thread(indexer).start();
while ((doc = indexer.getNext())!=null) {
  try {
    iw.addDocument(doc);
  } catch (Exception e) {
    System.err.println("Could not add document.");
    e.printStackTrace();
  }
  count++;
  float percentage = (float) count / (float) images.size();
  parent.progressBarIndexing.setValue((int) Math.floor(100f*percentage));
  float msleft = (float) (System.currentTimeMillis() - time) / percentage;
  float secLeft = msleft * (1 - percentage) / 1000f;
  String toPaint = "~ " + df.format(secLeft) + " sec. left";
  if (secLeft>90) toPaint = "~ " + Math.ceil(secLeft/60) + " min. left";
  parent.progressBarIndexing.setString(toPaint);
 }
 long timeTaken = (System.currentTimeMillis() - time);
 float sec = ((float) timeTaken) / 1000f;
 System.out.println("Finished indexing ...");
 parent.progressBarIndexing.setString(Math.round(sec) + " sec. for " + count + "
files");
 parent.buttonStartIndexing.setEnabled(true);
 parent.progressBarIndexing.setValue(100);
 iw.optimize();
 iw.close();

 } catch (IOException ex) {
  Logger.getLogger("global").log(Level.SEVERE, null, ex);
 }
}
```

```
public static ArrayList<String> getAllImages(File directory, boolean
descendIntoSubDirectories) throws IOException {
    ArrayList<String> resultList = new ArrayList<String>(256);
    File[] f = directory.listFiles();
    for (File file : f) {
      if (file != null && (file.getName().toLowerCase().endsWith(".jpg") ||
file.getName().toLowerCase().endsWith(".png"))&& !file.getName().startsWith("tn_"))
{
        resultList.add(file.getCanonicalPath());
      }
      if (descendIntoSubDirectories && file.isDirectory()) {
        ArrayList<String> tmp = getAllImages(file, true);
        if (tmp != null) {
          resultList.addAll(tmp);
        }
      }
    }
    if (resultList.size() > 0)
      return resultList;
    else
      return null;
  }

  private BufferedImage readFile(String path) throws IOException {
    BufferedImage image = null;
    FileInputStream jpegFile = new FileInputStream(path);
    Metadata metadata = new Metadata();
    try {
      new ExifReader(jpegFile).extract(metadata);
      byte[] thumb = ((ExifDirectory)
metadata.getDirectory(ExifDirectory.class)).getThumbnailData();
```

```
            if (thumb!=null) image = ImageIO.read(new ByteArrayInputStream(thumb));
//              System.out.print("Read from thumbnail data ... ");
//              System.out.println(image.getWidth() + " x " + image.getHeight());
          } catch (JpegProcessingException e) {
            System.err.println("Could not extract thumbnail");
            e.printStackTrace();
          } catch (MetadataException e) {
            System.err.println("Could not extract thumbnail");
            e.printStackTrace();
          } catch (Exception e) {
            System.err.println("Could not extract thumbnail");
            e.printStackTrace();
          }
          // Fallback:
          if (image == null) image = ImageIO.read(new FileInputStream(path));
          return image;
        }

      }
```

```
package liredemo;

import javax.swing.*;
public class ProgressMonitor {
    JProgressBar pbar;

    public ProgressMonitor(JProgressBar progressBar) {
        pbar = progressBar;
```

```
    pbar.setMinimum(0);
    pbar.setMaximum(100);
  }

  public int getProgress() {
    return pbar.getValue();
  }

  public void setProgress(int progress) {
    progress = Math.max(0, progress);
    progress = Math.min(100, progress);
    pbar.setString(Math.round(progress) + "% finished");
    pbar.setValue(progress);
  }
}
```

Search result control

```
package liredemo;

import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.File;
import java.io.ByteArrayInputStream;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import java.net.URL;
import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import net.semanticmetadata.lire.DocumentBuilder;
import net.semanticmetadata.lire.ImageSearchHits;
import net.semanticmetadata.lire.utils.ImageUtils;
import com.drew.metadata.Metadata;
import com.drew.metadata.exif.ExifReader;
import com.drew.metadata.exif.ExifDirectory;

public class SearchResultsTableModel extends DefaultTableModel {
    DecimalFormat df = (DecimalFormat) DecimalFormat.getInstance();
    ImageSearchHits hits = null;
    private ArrayList<ImageIcon> icons;

    /**
     * Creates a new instance of SearchResultsTableModel
     */
    public SearchResultsTableModel() {
        df.setMaximumFractionDigits(2);
        df.setMinimumFractionDigits(2);
    }

    public int getColumnCount() {
        return 2;
    }

    public String getColumnName(int col) {
        if (col == 0) {
            return "File";
```

```
    } else if (col == 1) {
      return "Preview";
    } else if (col == 2) {
      return "Preview";
    }
    return "";
  }

  public Class getColumnClass(int column) {
    if (column == 0) {
      return String.class;
    } else {
      return ImageIcon.class;
    }
  }

  public int getRowCount() {
    if (hits == null) return 0;
    return hits.length();
  }

  public Object getValueAt(int row, int col) {
    if (col == 0) {
      String text =
hits.doc(row).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue();
      if (hits.doc(row).getField("Flickr") != null) {
        text = hits.doc(row).getField("FlickrTitle").stringValue() + " - " +
hits.doc(row).getField("FlickrURL").stringValue();
      }
      return df.format(hits.score(row)) + ": " + text;
//    } else if (col == 1) {
```

```
//          return
hits.doc(row).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue();
      } else if (col == 1) {
        return icons.get(row);
      }
      return null;
    }

    /**
     * @param hits
     * @param progress
     */
    public void setHits(ImageSearchHits hits, JProgressBar progress) {
      this.hits = hits;
      icons = new ArrayList<ImageIcon>(hits.length());
      progress.setString("Searching finished. Loading images for result list.");
      for (int i = 0; i < hits.length(); i++) {
        ImageIcon icon = null;
        try {
          BufferedImage img = null;
          String fileIdentifier =
  hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue();
          if (!fileIdentifier.startsWith("http:")) {
            // check isf it is a jpg file ...
            if (fileIdentifier.toLowerCase().endsWith(".jpg")) {
              Metadata metadata = new ExifReader(new
  FileInputStream(fileIdentifier)).extract();
              if (metadata.containsDirectory(ExifDirectory.class)) {
                ExifDirectory exifDirectory = (ExifDirectory)
  metadata.getDirectory(ExifDirectory.class);
                if (exifDirectory.containsThumbnail()) {
```

```
                    img = ImageIO.read(new
ByteArrayInputStream(exifDirectory.getThumbnailData()));
                }
              }
            }
            if (img == null) {
              img = ImageIO.read(new FileInputStream(fileIdentifier));
            }
          } else {
            img = ImageIO.read(new URL(fileIdentifier));
          }
          icon = new ImageIcon(ImageUtils.scaleImage(img, 128));
          progress.setValue((i * 100) / hits.length());
        } catch (Exception ex) {
          Logger.getLogger("global").log(Level.SEVERE, null, ex);
        }
        icons.add(icon);
      }
      progress.setValue(100);
      fireTableDataChanged();
    }

    /**
     * @return
     */
    public ImageSearchHits getHits() {
      return hits;
    }
    public boolean isCellEditable(int row, int column) {
      return false;
    }
  }
```

```
package liredemo.flickr;

import net.semanticmetadata.lire.DocumentBuilder;
import net.semanticmetadata.lire.utils.ImageUtils;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.LinkedList;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;


public class FlickrDownloadThread implements Runnable {
    public final int NUMBER_OF_SYNC_DOWNLOADS = 5;
    private List<FlickrPhoto> images;
    private LinkedList<Document> finished;
    private int currentIndex = 0;
    DocumentBuilder builder;

    public FlickrDownloadThread(List<FlickrPhoto> images, DocumentBuilder builder)
{
        this.images = images;
        finished = new LinkedList<Document>();
        this.builder = builder;
    }
```

```
  public void run() {
     ExecutorService pool =
Executors.newFixedThreadPool(NUMBER_OF_SYNC_DOWNLOADS);
     for (FlickrPhoto photo : images) {
       // System.out.println("photo.title = " + photo.title);
       pool.execute(new SinglePhotoThread(photo, this));
     }
     pool.shutdown();
     while (!pool.isTerminated()) {
       try {
         Thread.sleep(500);
       } catch (InterruptedException e) {
         e.printStackTrace();  //To change body of catch statement use File | Settings |
File Templates.
       }
     }
     if (currentIndex < images.size()) {
       try {
         Thread.sleep(1500);
       } catch (InterruptedException e) {
         e.printStackTrace();
       }
       System.out.println("WARNING: " + (images.size() - currentIndex) + " images
left");
       currentIndex = images.size();
     }
   }


   public synchronized Document getCurrentDoc() {
     if (currentIndex < (images.size() - 1)) {
```

```
        currentIndex++;
        while (finished.size() < 1) {
          try {
            Thread.sleep(250);
          } catch (InterruptedException e) {
            e.printStackTrace();
          }
        }
        return finished.removeFirst();
      } else return null;
    }

    public DocumentBuilder getDocumentBuilder() {
      return builder;
    }

    public void addDocumentToFinished(Document doc) {
      if (doc == null) currentIndex++;
      else finished.add(doc);
    }
  }

  class SinglePhotoThread implements Runnable {
    FlickrPhoto photo;
    FlickrDownloadThread fdt;

    SinglePhotoThread(FlickrPhoto photo, FlickrDownloadThread fdt) {
      this.photo = photo;
      this.fdt = fdt;
    }

    public void run() {
```

```
    try {
        BufferedImage image = ImageIO.read(new URL(photo.photourl));
        image = ImageUtils.scaleImage(image, 320);
        File cachedImage = new File(FlickrIndexingThread.cacheDirectory +
photo.photourl.substring(photo.photourl.lastIndexOf("/") + 1, photo.photourl.length()));
        ImageIO.write(image, "jpg", cachedImage);
        Document doc = fdt.getDocumentBuilder().createDocument(image,
cachedImage.getAbsolutePath());
        doc.add(new Field("FlickrURL", photo.url, Field.Store.YES,
Field.Index.UN_TOKENIZED));
        doc.add(new Field("FlickrTitle", photo.title, Field.Store.YES,
Field.Index.UN_TOKENIZED));
        for (String tag : photo.tags) {
          doc.add(new Field("FlickrTag", tag, Field.Store.YES,
Field.Index.UN_TOKENIZED));
        }
        fdt.addDocumentToFinished(doc);
      } catch (IOException e) {
        System.out.println("Warning: Exception reading & indexing image " +
photo.photourl + ": " + e.getMessage());
        fdt.addDocumentToFinished(null);
      }

    }
  }
```

Flicker indexing thread

```
package liredemo.flickr;

import liredemo.LireDemoFrame;
import net.semanticmetadata.lire.DocumentBuilder;
import net.semanticmetadata.lire.DocumentBuilderFactory;
import org.apache.lucene.analysis.SimpleAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.IndexWriter;
import org.xml.sax.SAXException;

import javax.imageio.ImageIO;
import javax.xml.parsers.ParserConfigurationException;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.text.DecimalFormat;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;


public class FlickrIndexingThread extends Thread {
    LireDemoFrame parent;
    public static final String cacheDirectory = "./flickrphotos/";
    private int numberOfPhotosToIndex = 100;

    /** Creates a new instance of FlickrIndexingThread
```

```
* @param parent
*/
public FlickrIndexingThread(LireDemoFrame parent) {
  this.parent = parent;
}

public FlickrIndexingThread(LireDemoFrame parent, int maxNumberOfPhotos) {
  this.parent = parent;
  this.numberOfPhotosToIndex = maxNumberOfPhotos;
}

public void run() {
  DecimalFormat df = (DecimalFormat) DecimalFormat.getInstance();
  df.setMaximumFractionDigits(0);
  df.setMinimumFractionDigits(0);
  try {
    File cacheDir = new File(cacheDirectory);
    if (!cacheDir.exists()) cacheDir.mkdir();
    parent.progressBarIndexing.setValue(0);
    parent.progressBarIndexing.setString("Getting photos from Flickr");
    List<FlickrPhoto> images = new LinkedList<FlickrPhoto>();
    HashSet<String> titles = new HashSet<String>(numberOfPhotosToIndex);
    try {
      while (images.size() < numberOfPhotosToIndex) {
        List<FlickrPhoto> photos = FlickrPhotoGrabber.getRecentPhotos();
        for (FlickrPhoto photo : photos) {
          // check if it is already there:
          if (!titles.contains(photo.url)) {
            titles.add(photo.url);
            if (images.size()<numberOfPhotosToIndex) images.add(photo);
          } else {
            try {
```

```
                    Thread.sleep(150);
                  } catch (InterruptedException e) {
                    e.printStackTrace();
                  }
                }
              }
              parent.progressBarIndexing.setString("Getting photos from Flickr: " +
images.size() + " found.");
            }
          } catch (SAXException e) {
            e.printStackTrace();
          } catch (ParserConfigurationException e) {
            e.printStackTrace();
          }
          IndexWriter iw = new IndexWriter(parent.textfieldIndexName.getText(), new
SimpleAnalyzer(), !parent.checkBoxAddToExisintgIndex.isSelected());
          int builderIdx = parent.selectboxDocumentBuilder.getSelectedIndex();
          DocumentBuilder builder =
DocumentBuilderFactory.getFullDocumentBuilder();
          int count = 0;
          long time = System.currentTimeMillis();
          FlickrDownloadThread downloader = new FlickrDownloadThread(images,
builder);
          new Thread(downloader).start();
          Document doc = null;
          while ((doc = downloader.getCurrentDoc())!=null) {
            try {
              iw.addDocument(doc);
            } catch (Exception e) {
              System.err.println("Could not add document");
              // e.printStackTrace();
            }
```

```
            count++;
            float percentage = (float) count/ (float) images.size();
            parent.progressBarIndexing.setValue((int) Math.floor(100f*percentage));
            float msleft = (float) (System.currentTimeMillis() - time) / percentage;
            float secLeft = msleft * (1 - percentage) / 1000f;
            String toPaint;
            if (secLeft>60) toPaint = "~ " + Math.ceil(secLeft/60) + " min. left";
            else if (secLeft>30) toPaint = "< 1 min. left";
            else toPaint = "< 30 sec. left";
            parent.progressBarIndexing.setString(toPaint);
          }
          long timeTaken = (System.currentTimeMillis() - time);
          float sec = ((float) timeTaken) / 1000f;
          parent.progressBarIndexing.setValue(100);
          parent.progressBarIndexing.setString(Math.round(sec) + " sec. for " + count + "
files");
          parent.buttonStartIndexing.setEnabled(true);
          iw.optimize();
          iw.close();
        } catch (IOException ex) {
          Logger.getLogger("global").log(Level.SEVERE, null, ex);
        }
      }
      private BufferedImage readFile(String path) throws IOException {
        BufferedImage image = ImageIO.read(new URL(path));
        return image;
      }
}
```

APPENDIX-2: SCREEN SHOTS

The Home Page:

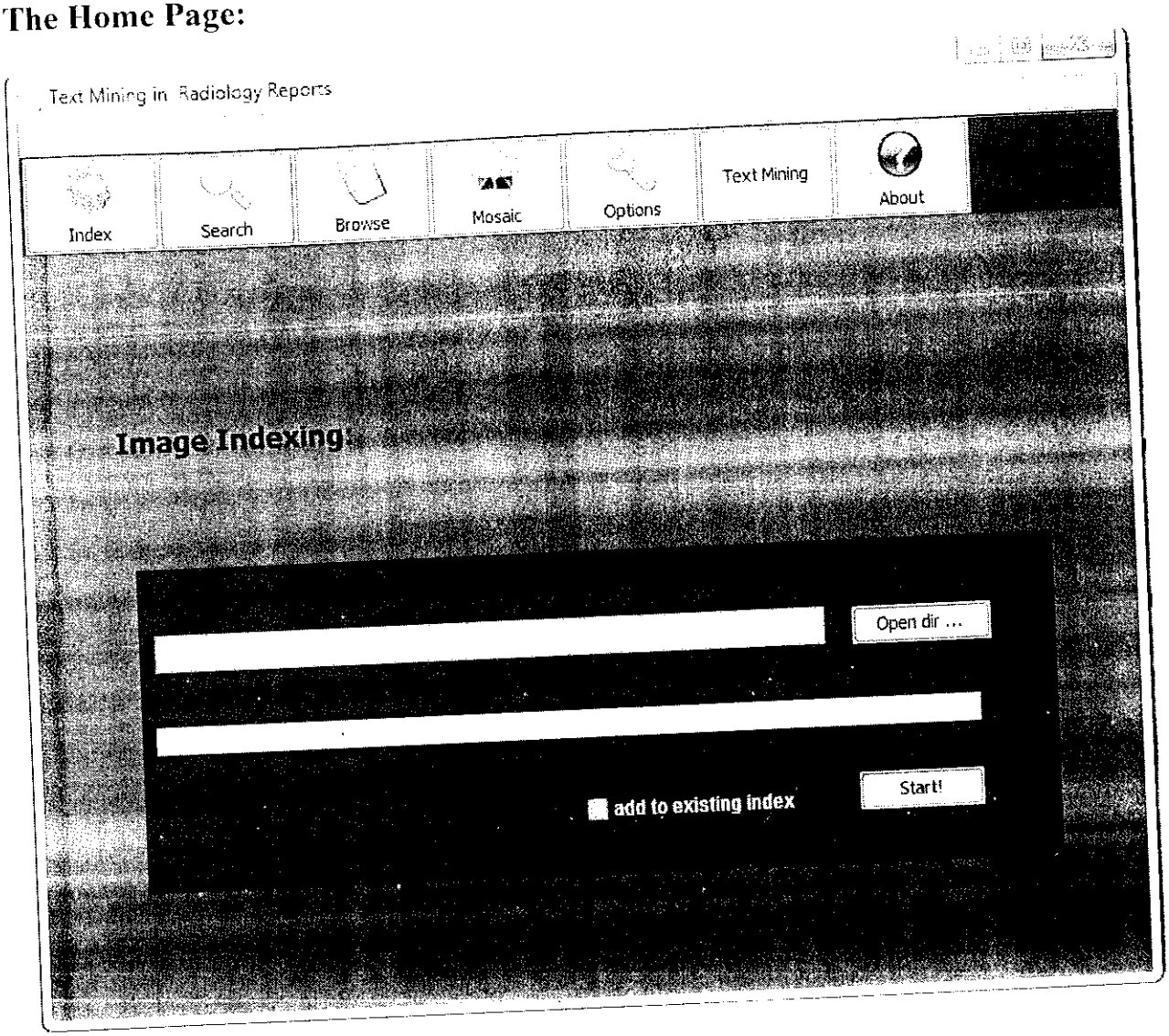

Figure. a. The Home Pages

Load Dataset Images:

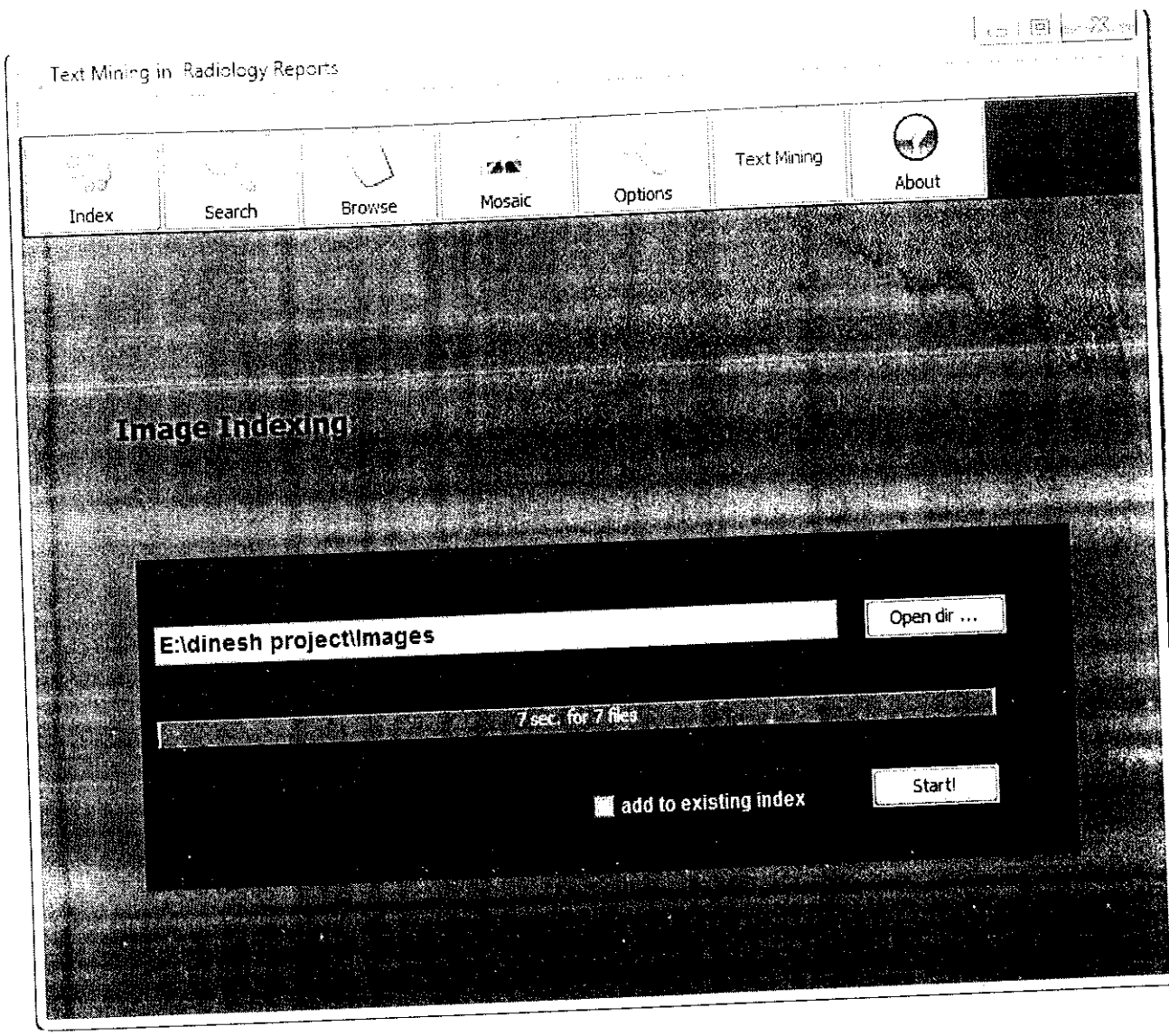

Figure. b. Load Image Datasets

Select input image Dataset by using open dir button. Choose the Dataset and press open button to open particular Dataset. And click start button to upload all images on that particular Dataset.

Load Input Image for Search:

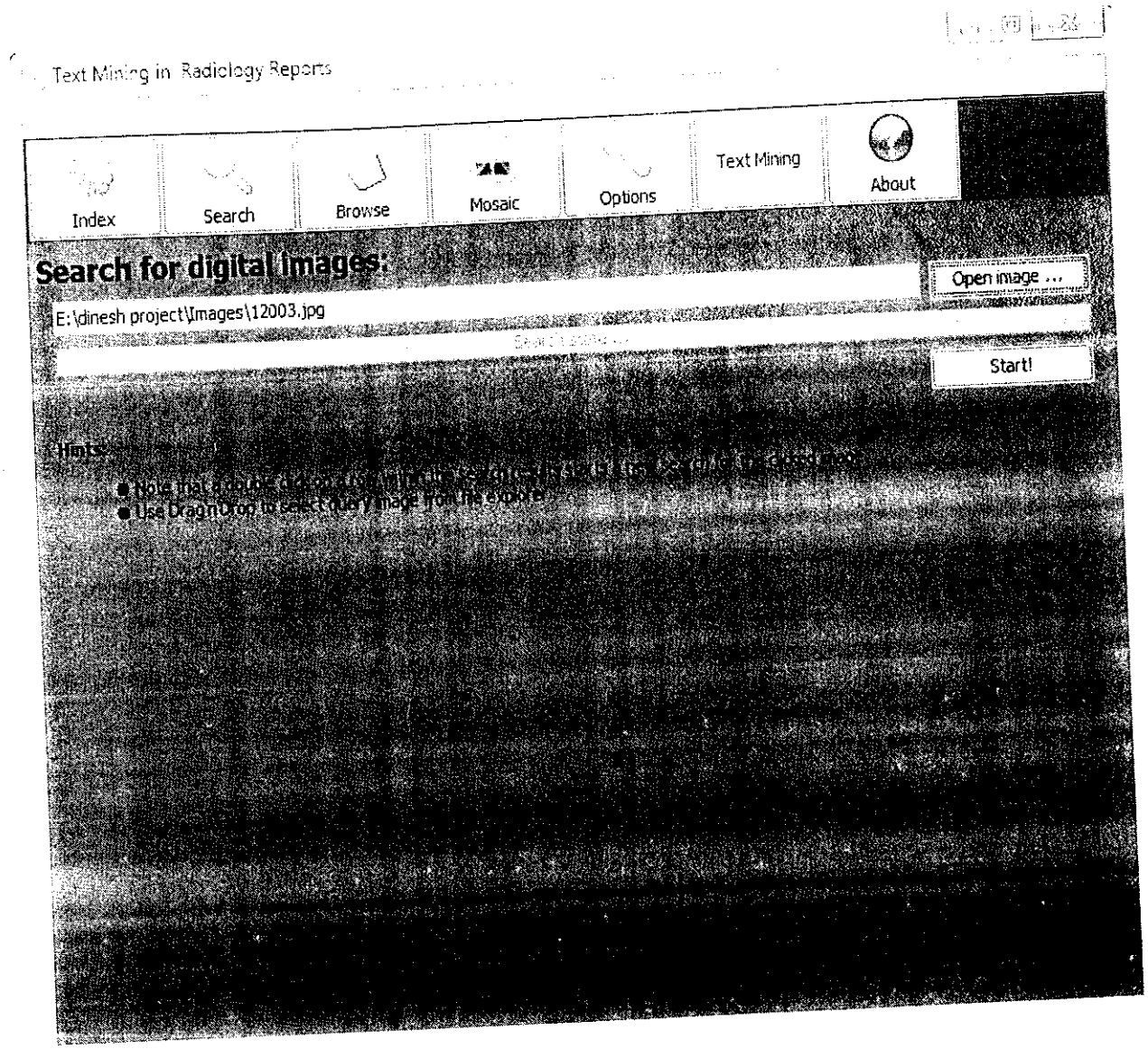

Figure. c. Load Input Image for search

Select input image by using open image button. Choose the input image from particular folder which already uploads. And click start button to search all images that related to input image.

Search Results Based on Input Image:

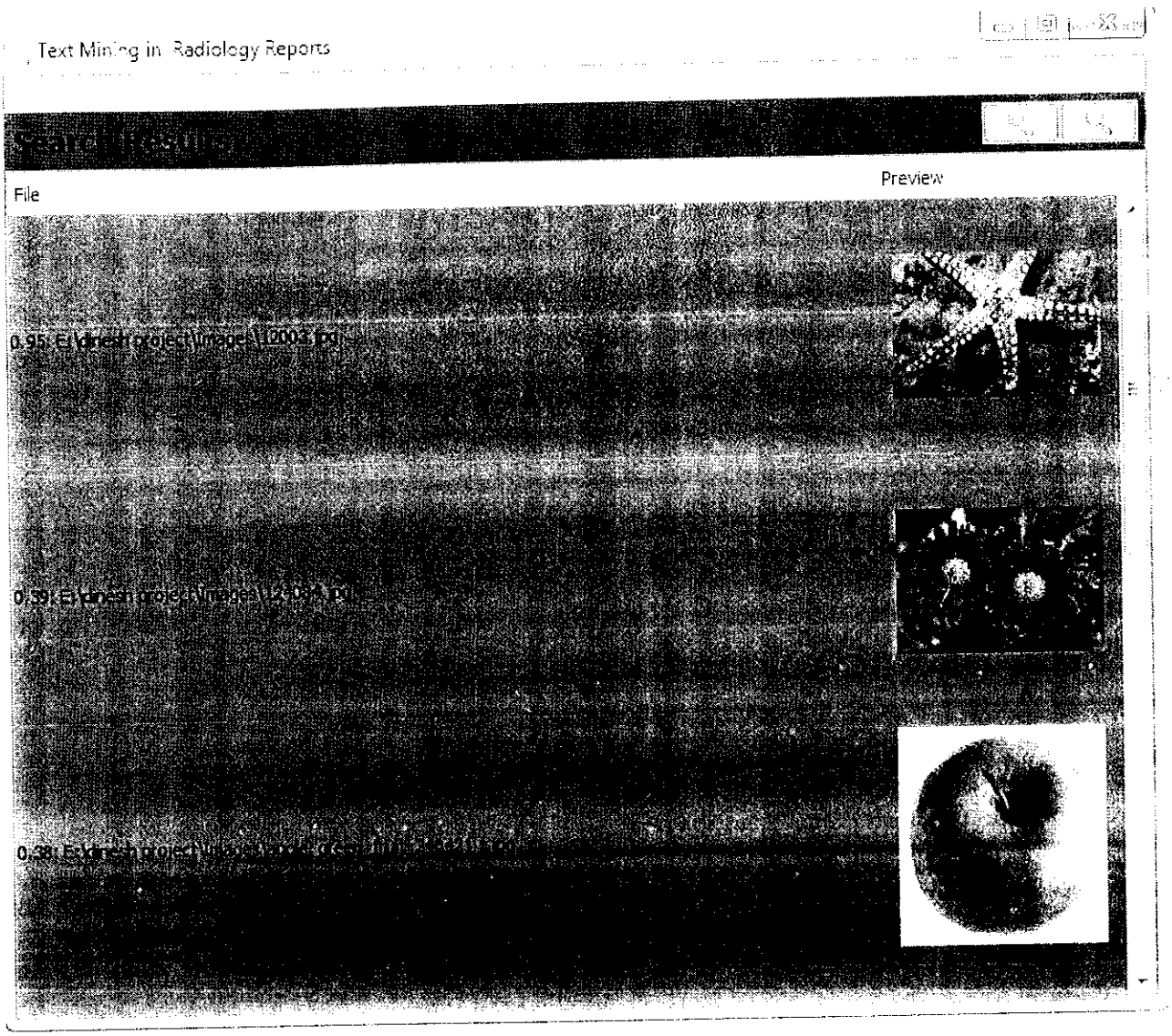

Figure. d. Search result for Input Image

Displayed search results based on the input image. Based on ranking images are displayed. If we want to choose the input image from result images, double click the particular image then it provide search result for that image.

Search Results Based on Input Image:

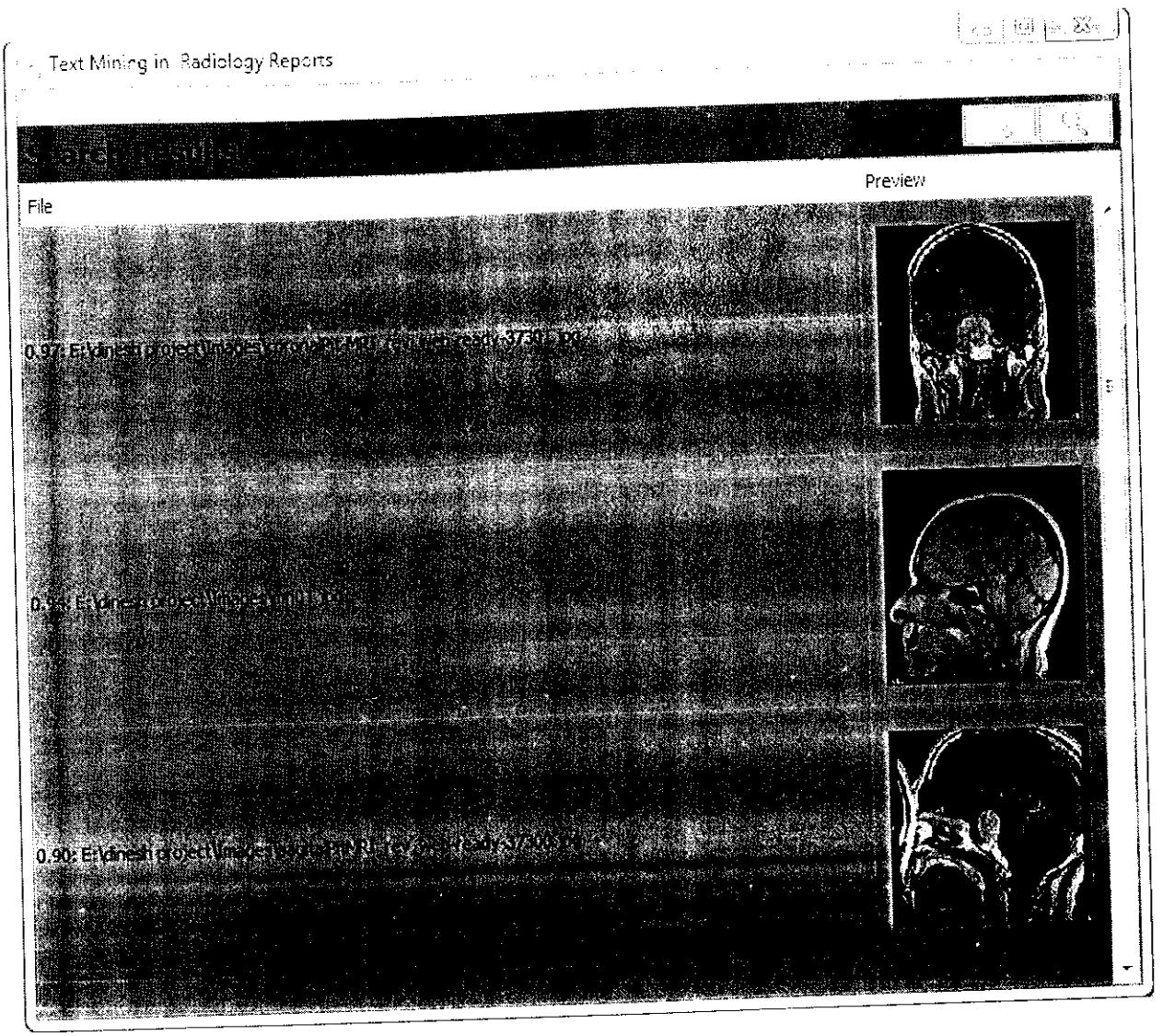

Figure. e. Search result for Input Image

Displayed search results based on the input image. Based on ranking images are displayed.

Upload Input as Image:

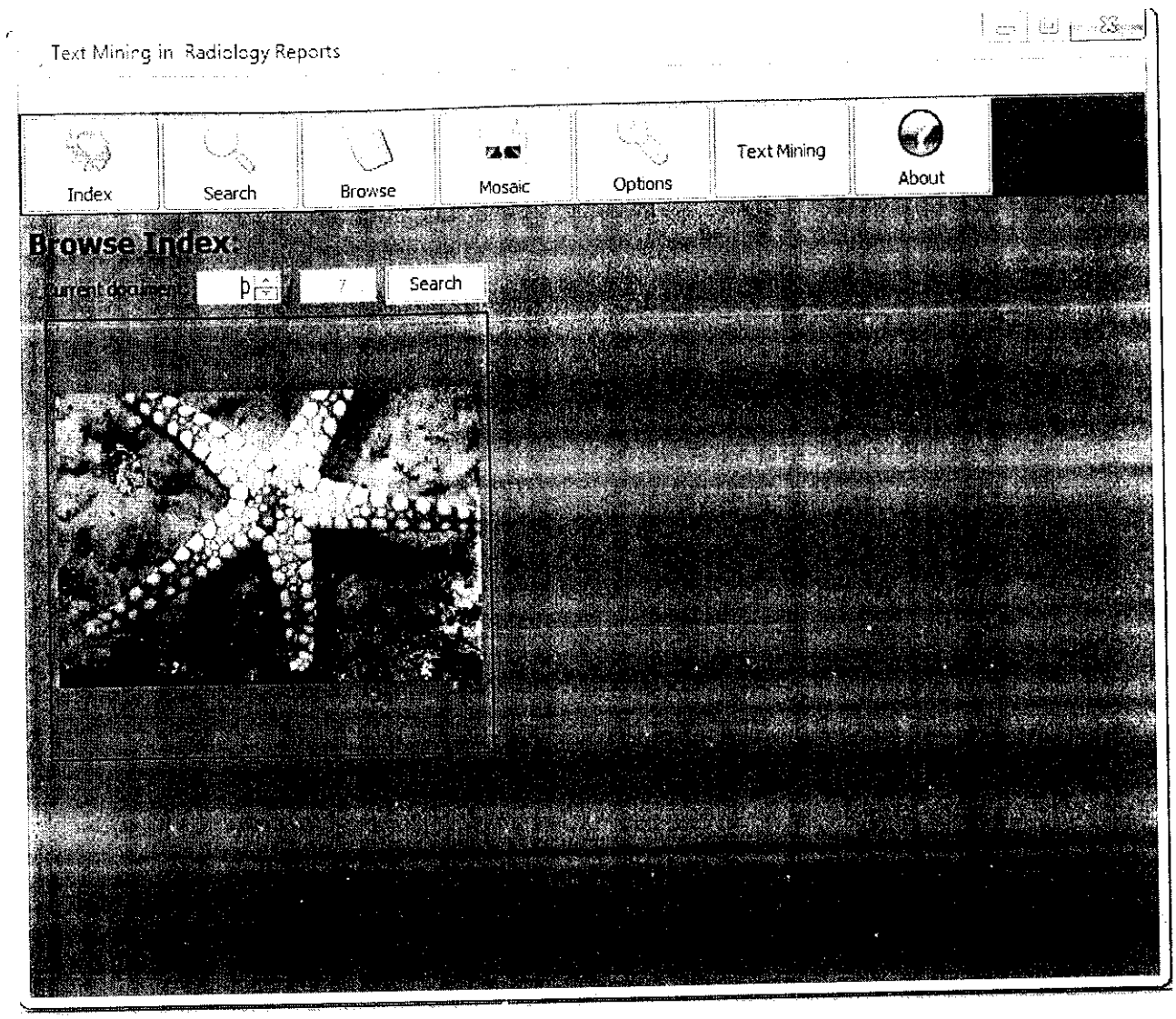

Figure. f. upload Input as Image

Select input as image by using arrow button. Choose the image from particular folder which already uploads. And click search button to search all images that related to input image.

Result Based on Input Image:

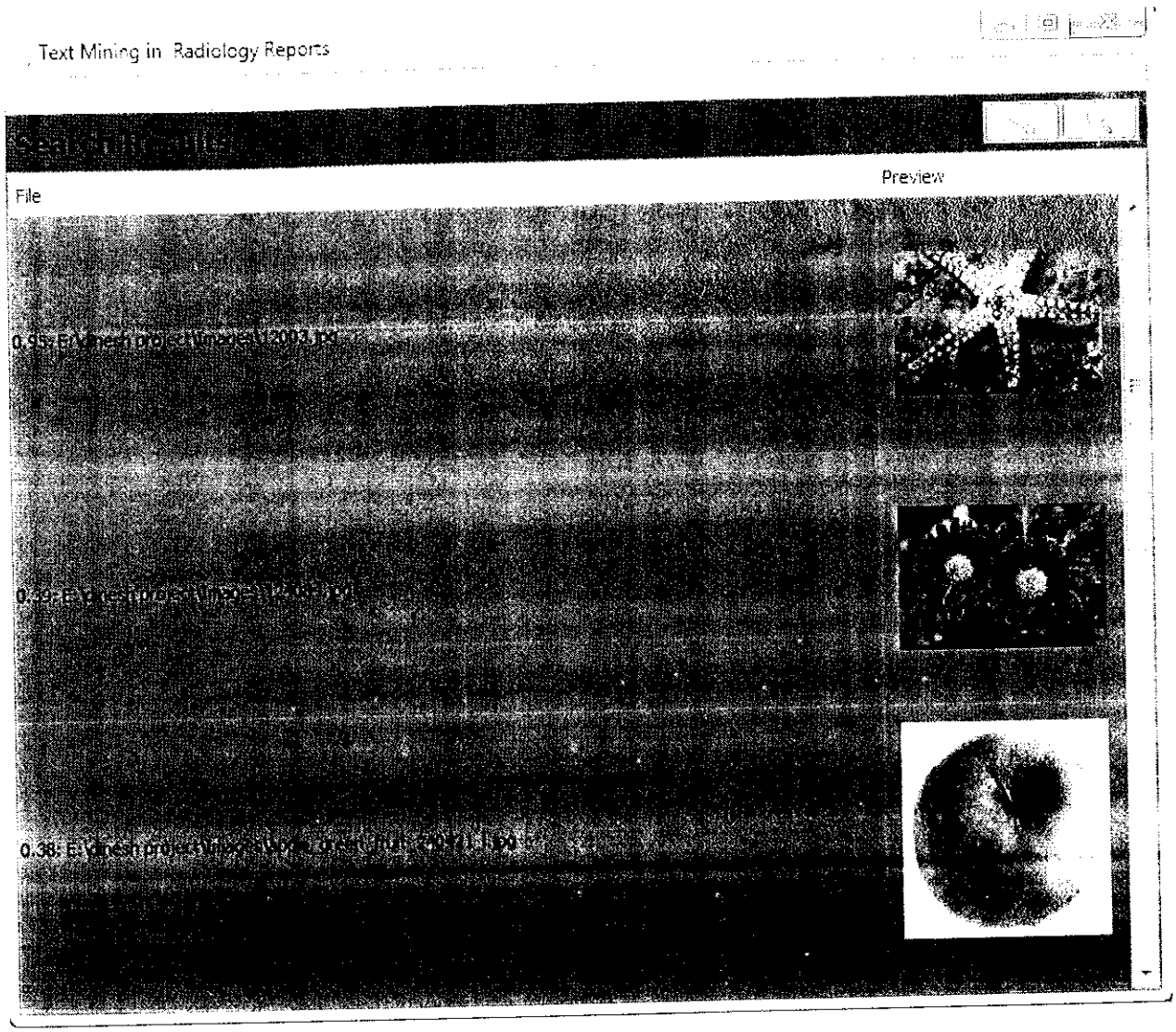

Figure. g. Search result for Input Image

Displayed search results based on the input image. Based on ranking images are displayed. If we want to choose the input image from result images, double click the particular image then it provide search result for that image.

Select Image to Create Mosaic:

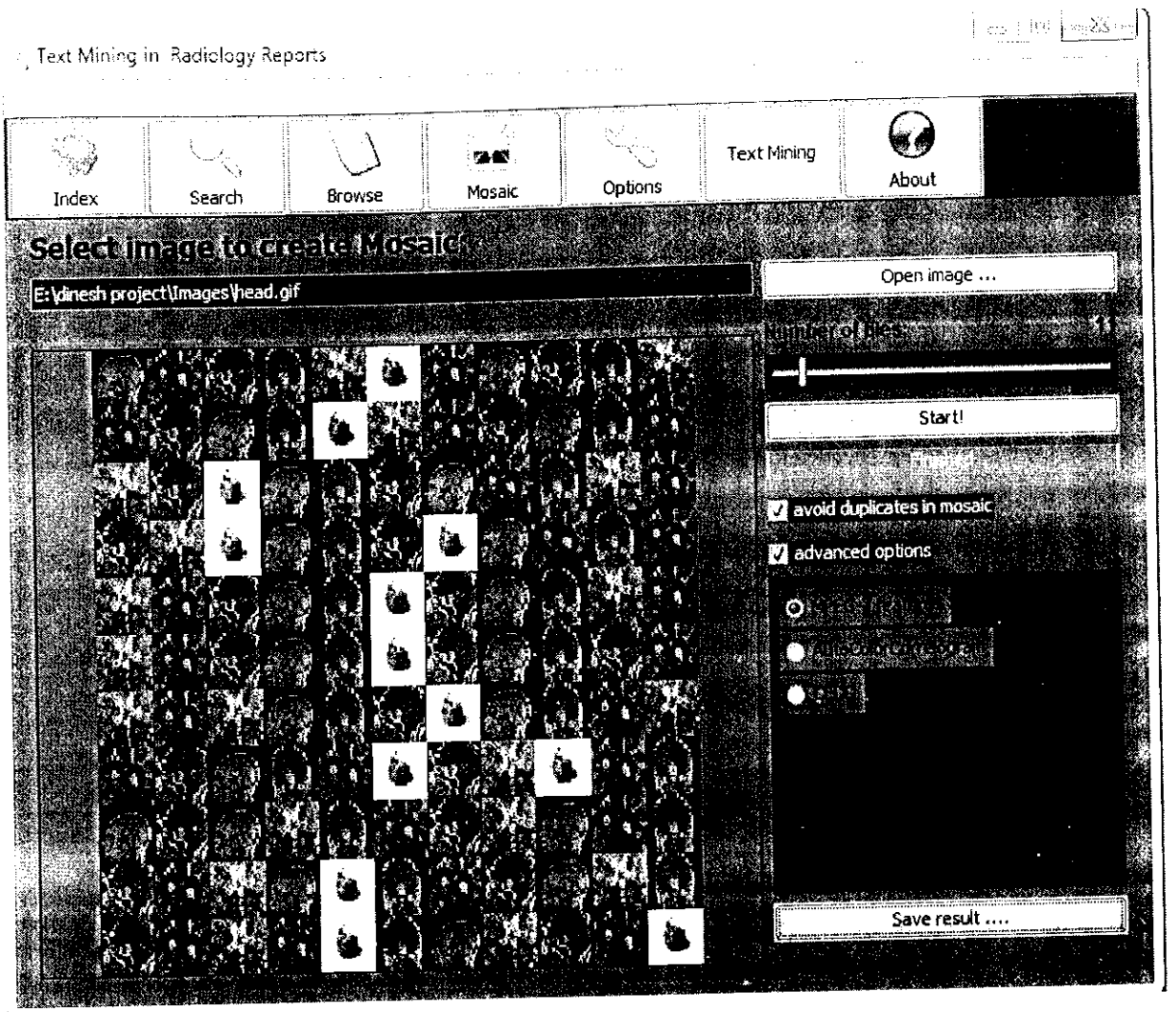

Figure. h. Select image to Create mosaic

Select input image by using open image button. Choose the input image from particular folder which already uploads. And click start button to search all images that related to input image and it create mosaic format. In this mosaic it displayed images based on ranking.

Select options for Search Result:

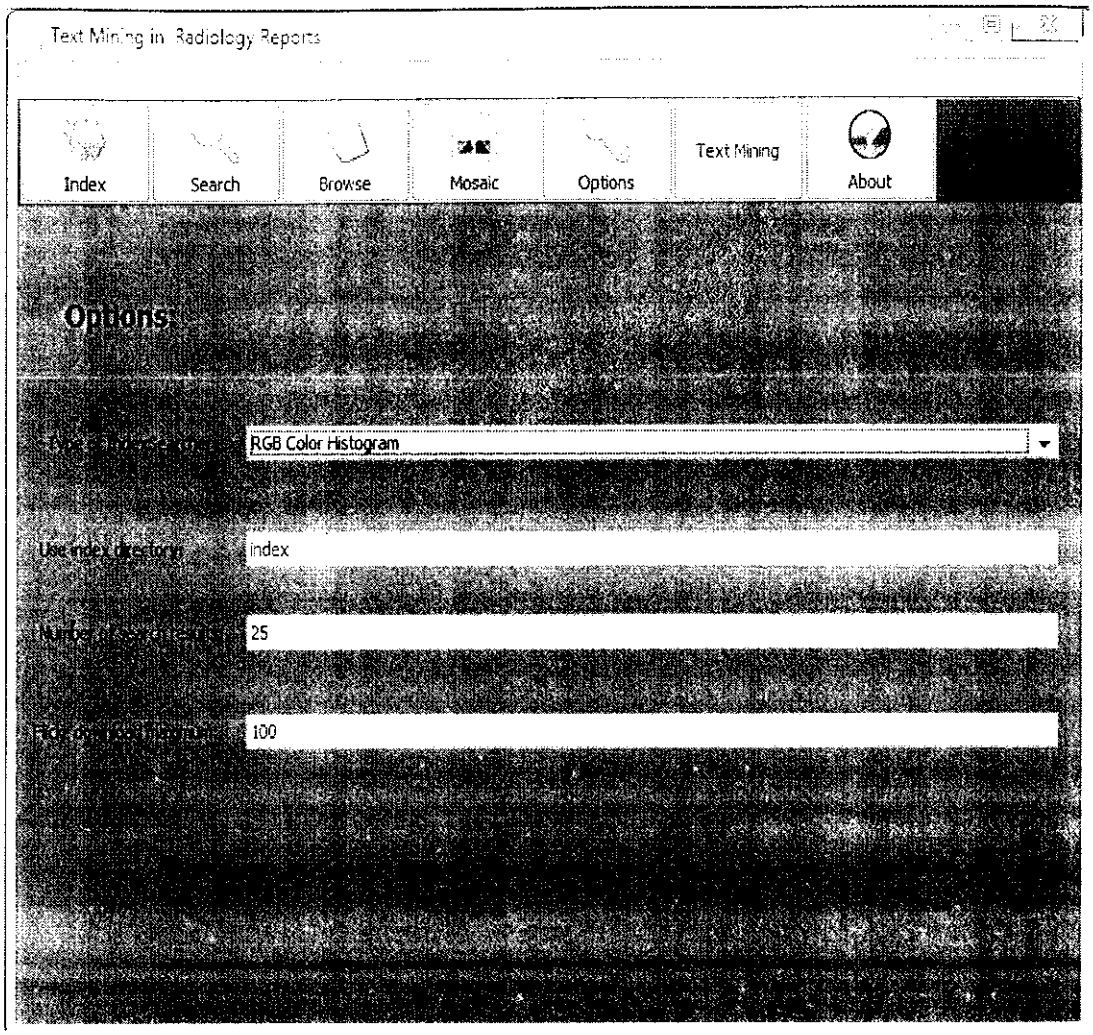

Figure. i. select Option for Search result

Choose the options like type of index searcher, number of search results, download maximum and so on. Result image set based on the option already selected.

Text Mining Search:

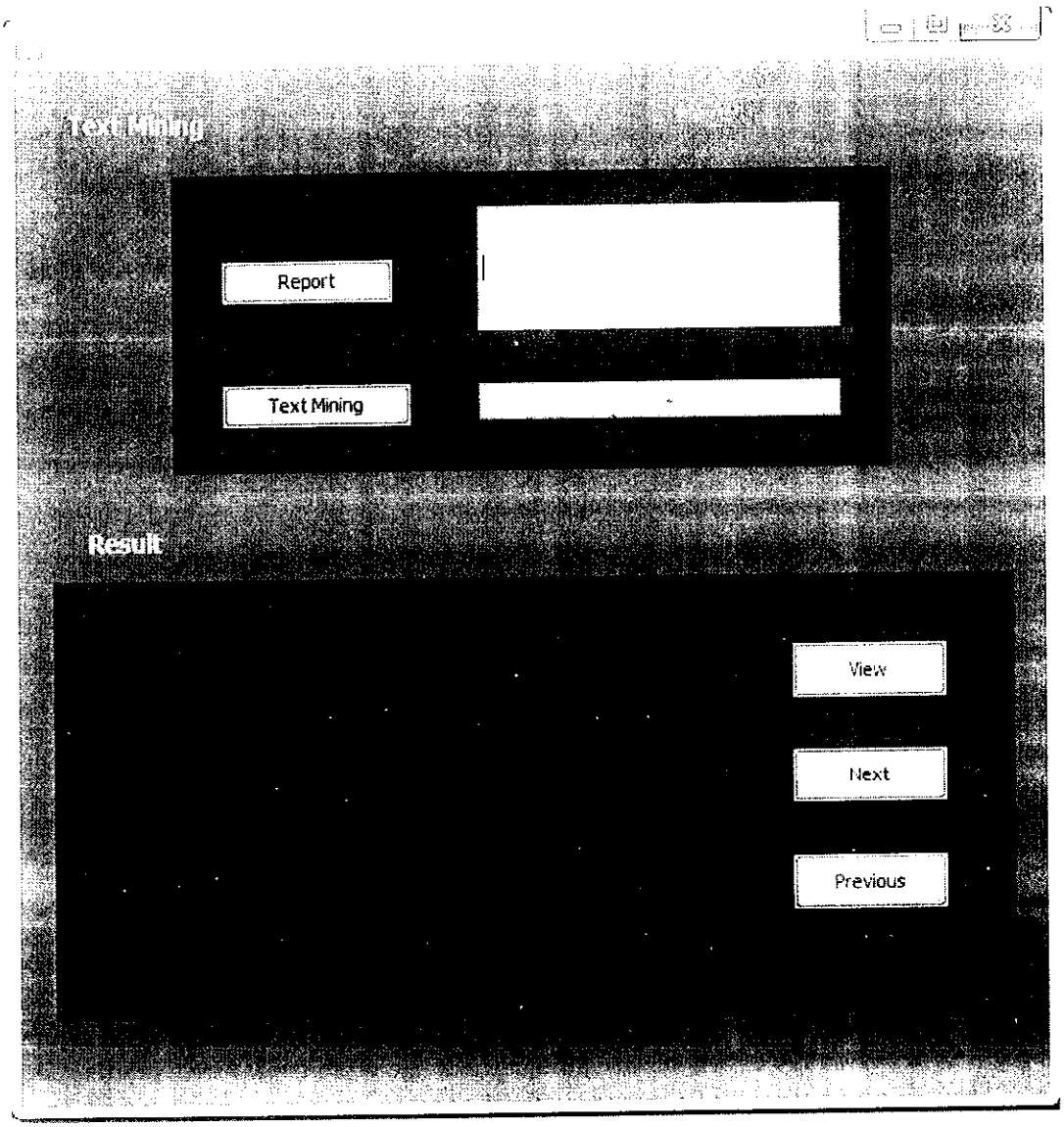

Figure. j. Text Mining Search Window

Text Mining Search Result:

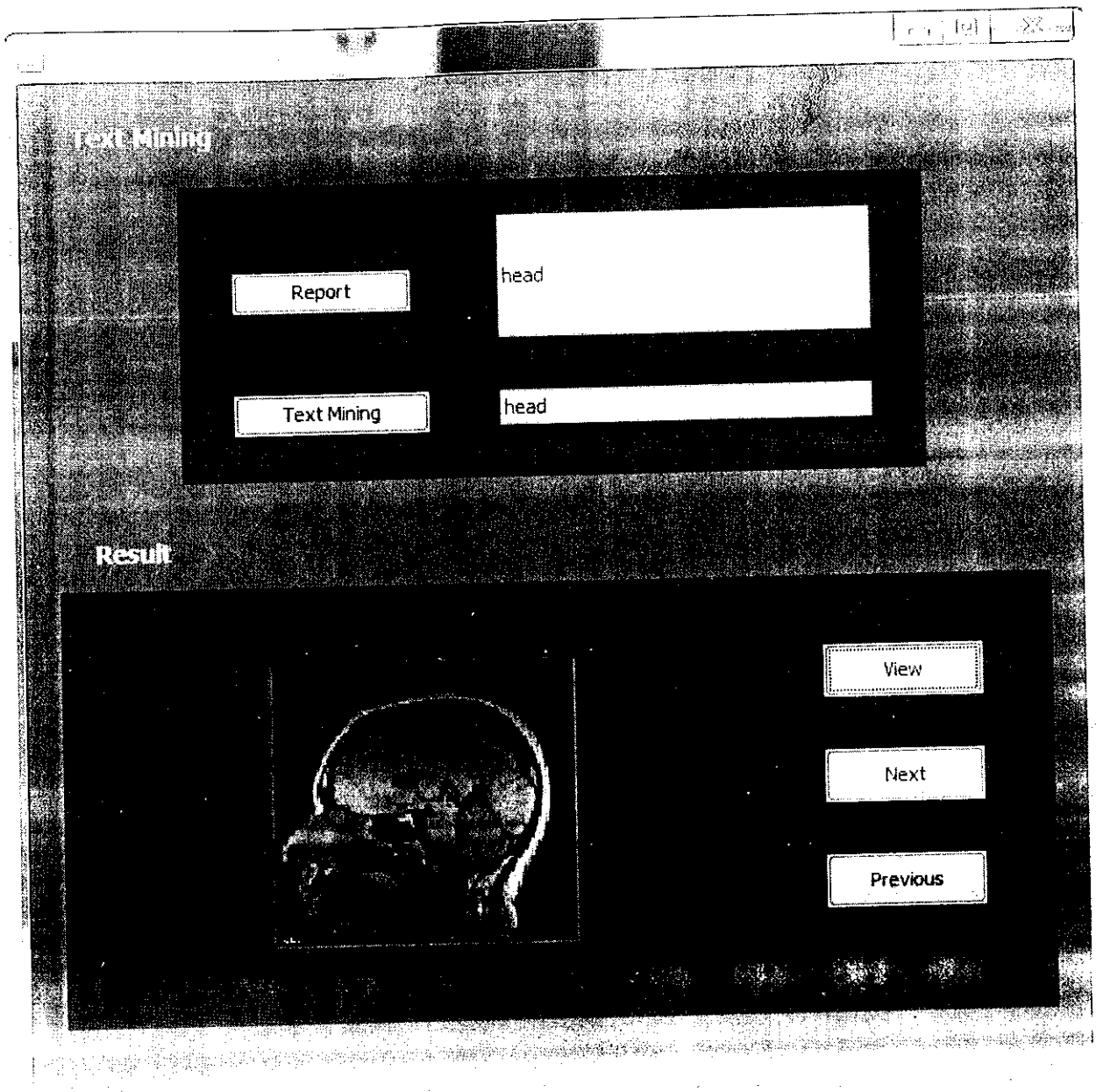

Figure. k. Search result by Text Mining

REFERENCES

1. En Cheng,Feng Jing and Lei Zhang.(2009) **"A Unified Relevance Feedback for Web Image Retrieval,"** IEEE Trans.Image Process.,vol.18,no.6.
2. Hu.C,Lu.Y,Yang.Q,X.Zhu and Zhang.H.J.(2000) **"A Unified framework for semantics and feature based relevance feedback in image retrieval systems,"** in Proc. 8th Annu. ACM Int. Conf. Multimedia, pp.31–38.
3. Jing.F, Li.M.J, Zhang.J, and Zhang.B(2005) **"A weighted framework for image retrieval using keyword and visual features,"** IEEE Trans. Image Process., vol. 14, no. 7, pp. 979–989.
4. Gay.G,Granka.L,Hembrooke.H,Joachims.T,Pan.B.(2005)**"Accurately interpreting clickthrough data as implicit feedback,"** in Proc.28th Annu. Int. ACM SIGIR Conf. Research and Development in Information Retrieval,pp. 154–161.
5. Ruba Al-Haj and Zaher Al Aghbari,(2006) **"Building SSeg-Tree for Image Representation and Retrieval", ICGST Int. Journal on Graphics, Vision and Image Processing (GVIP), Special Issue on Image Retrieval and Representation",** Vol. 6, pp. 101-109.
6. Tanuja K. Sarode and Kekri.H.B.(2005) **"Clustering Algorithms for Codebook Generation Using Vector Quantization,"** National Conference on Image Processing , TSEC, Mumbai.
7. Kekre.H.B., Sudeep D. Thepade and Tanuja Sarode and (2008) **"DCT Applied to Row Mean and Column Vectors in Fingerprint Identification",** In Proceedings of Int. Conf. on Computer Networks and Security (ICCNS), 27-28 , VIT, Pune.
8. Kotani.P,Ohmi,T.**(2005)"Enhanced fast encoding method for vector quantization by finding an optimally-ordered Walsh transform kernel",** ICIP05, IEEE Int. Conf., Vol.1, pp I-573.
9. Chin-Chen Chang, Wen-Chuan Wu.(2007) **" Fast Planar-Oriented Ripple Search Algorithm for Hyperspace VQ Codebook",** IEEE Transaction on image processing, vol 16, No. 6.

10. S. Wang.J and Yang.C.H.(2005) **"Hierarchy-oriented searching algorithms using alternative duplicate codewords for vector quantization mechanism,"** Appl. Math. Comput., vol. 162, No. 234, pp. 559–576.

11. Jung Kim and Robert Li .(2000) **"Image Compression Using Fast Transformed Vector Quantization"**, IEEE Applied Imagery Pattern Recognition Workshop, 2000 Proceedings, Volume 29, pp.141 – 145.

12. Sarode.K,K.Tanuja,H.B.Kekre,(2008) **"New Fast Improved Clustering Algorithm for Codebook Generation for Vector Quantization"**, In Proc. of Int. Conf.ICETAETS, Saurashtra University, Gujarat (India), 13–14 .

13. Chang,Chang.K.C.C,Smith.J.R and Wu.Y,**"Optimal multimodal fusion for multimedia data analysis,"** in Proc. 12th Annu.

14. Huang.T.S and Zhou.X.S.(2003) **"Relevance feedback in image retrieval:A comprehensive review,"** ACM Multimedia Syst., vol. 8, no. 6, pp.536–544.

15. T. S. Huang, Mehrotra.S, Ortega.M. (1998) **"Relevance feedback: A power tool for interactive content-based image retrieval,"**IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 5, pp. 644–655.

16. Kekre.H.B., Sudeep D. Thepade, **"Rendering Futuristic Image Retrieval System",**National Conference EC2IT-2009, KJSCOE, Mumbai,20-21 Mar2009.

17. Biswas.K.K,S.K.Gupta,B.G.Prasa.B.G.(2004) **"Region based image retrieval using integrated color, shape, and location index"**, Int. Journal on Computer Vision and Image Understanding Special Issue: Colour for Image Indexing and Retrieval, Volume 94, Issues 1-3, pp.193-233.

18. Kekre.H.B, Sudeep Thepade.D.(2009) **"Ubicomp The Future of Computing Technology"**, TechnoPath :Journal of Science Technology and Management,Volume 1, Issue 2.

19. Bhushan Deshmukh,Kekre.H.B,Sudeep Thepade..(2009)**"WEB 3.0 The Astonishing Avtar of Web"**, TechnoPath: Journal of Science Technology and Management, Volume 1, Issue 2.

20. Minh N. Do, Member, IEEE. and Martin Vetterli, Fellow. IEEE.(2002) **"Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance"**, IEEE Transactions On Image Processing, Vol 11, Num. 2, pp.146-158.