Rapid Prototyping for Indexing and Retrieval for Text and Images

Monika Mogra[1], Ratnesh Prasad Srivastava[2], Vivek Srivastava[1]

[1]Department of Information Tech, Institute of Technology & Management, Bhilwara, Rajasthan-249225, India

[2]Department of Information Tech, College of Technology, GBPUA&T, Pantnagar-263145, India

Abstract.

In today's complex and interconnected world, information increasingly exists in forms that can be stored and transmitted electronically. The challenging part is to provide information required by the user. This task of providing information requested by the user is done by the search engines. Indexing plays an important role in optimizing the speed and performance of finding relevant documents for a search query. Without an index, the search engine would scan every document in the corpus, which would take a lot of time and computing power. For example, an index of 10,000 documents can be queried within milliseconds, where a sequential scan of every word in 10,000 large documents could take hours. Indexing means transformation of a raw document collection into an easily accessible representation of documents [6]. The system uses three-layered neural network to train the system using image clusters as training dataset by a supervised approach. Also after taking the feedback from the user, the image clusters are re-clustered by rearranging the images after a successful retrieval. Given a user query as an image, the neural system retrieves related images by computing similarities with images in the given image clusters. To provide preference, from all the retrieved images user selects an image as relevant one and all other are hence treated as irrelevant ones. So, the rank of the selected image is increased while the ranks of other images are decreased. With this feedback, the system refinement method estimates global approximations and adjusts the similarity probabilities.

Keywords: XML, Genetic Algorithm, Neural Network, CBIR System, Relevance Feedback, GLCM Structure, Heuristic Algorithm

1 Introduction

Indexing the documents is one of most widely used methods which help in the fast and efficient way of retrieving information from the set of the documents. A typical genetic algorithm is used to prepare this index tables from the group of documents that are given Due to increasing online storage of documents indexing has become a field of interest for effective information retrieval. Many software are present for ready indexing of the documents. Many research oriented organizations and universities are taking up this work of indexing to make the search for relevant documents easy. **Conceptual indexing for precision information retrieval [4]** , the central focus is the paraphrase problem where the words in the query does not match the words in the documents(the documents user wanted). Conceptual indexing builds taxonomy of words and phrases from the indexing tables such that the meaning of the word or the phrase matches the document meaning. **Text Indexing with Apache Jakarta Lucene [7],** this is a java library that helps in adding text indexing and searching for an application. It is not a complete application but a simple and powerful application. In Lucene the text that has to be indexed is first passed through the analyzers which analyze the data and then the words are indexed. **Multimedia indexing and retrieval [5] ,** this method is based on multi source information indexing and merging. The novelty of this method is the use heterogeneous source of information and combination or cross fusion of the information obtained from different sources. Here indexing is done and effective retrieval is done for the multimedia data. **Indexing oral presentations [6],** this is for the effective multimedia information retrieval. Slides contain the information delivered in speech and fundamental facts and figures. With help of automatic synchronization of information from different streams, indexing can be done for one stream to retrieve other stream. This project is undertaken by the IDIAP research institute Rue Du Simplon 4 - ch1920 Martigny (Switzerland).**A Contrastive Indexing Method, t**he meaning of the word is based on the linguistic or situational context. Words co-occurring around a hub-word share some recurring seem organized in structure. It is up to the user to select which of the words are semantically around the hub-word. This method is integrated in the Aquarelle folders. The Aquarelle project is managed by ERCIM which stands for

European Research Consortium for Informatics and Mathematics.
This paper is organized as followed: section II introduce the details of implementation of indexing based on Genetic Algorithm ,section III introduce the details of implementation of retrieval based on XML information retrieval with preprocessing and storing the XML documents in an appropriate format ,term frequency calculation parameter, query expansion, searching the index structure, ranking and presentation of result, section IV introduces Content Based Image Retrieval System(CBIR) which is a multidisciplinary domain closely related to various computer science and research fields such as Databases, Artificial Intelligence, Image Processing, Statistics, Pattern Recognition, Computer Vision, High Performance Computing and Human Computer Interaction, section V introduces implementation part of CBIR and section VI introduces conclusion and future work.

Review of Related Work

Implementation of indexing using genetic algorithm in done in four phases:
(i) Getting words from documents.
(ii) Making the chromosome set and calculating fitness.
(iii) Applying crossover and mutation.
(iv) Making the Index table.

Input for the Program: Group of text documents whose index table is to be formed.

Output of the Program: Chromosome sets in the a text document and a text document having the indexed words.

Implementation Details:

Getting words from documents: Documents whose index table is to be created are taken as input. Each line is read at a time. Unwanted characters like fullstop, semicolon, colon etc are removed. Remaining words are checked with the stop words and if any stop word is found they are removed and the remaining words are taken into consideration for forming the chromosome set.

Making the chromosome set and calculating the fitness: Once the words from the documents are obtained, all the words of all the documents are taken as a single set. Count of the words may be high. After taking all the words, chromosome set of each document is calculated i.e. if the word is present in the document then the position is kept as one else it is taken as zero. Once the chromosome set is calculated Jaccard's coefficient is calculated for each document with the other document. After calculating the Jaccard's Coefficient for a document average of all the coefficients is taken for that document which gives the fitness of that document. In the same manner the Jaccard's Coefficient is calculated for all documents and the fitness of all documents is calculated. Average fitness of the chromosome set is calculated. Calculating Jaccard's Coefficient is pretty simple. If ch [] and ch1 [] are the chromosome sets of two documents then the Jaccard's score is

∑ (ch[i] and ch1 [i])/∑ (ch[i] or ch[i])

```
Jaccard's Score of DOC0 and DOC0 = 1.000000
Jaccard's Score of DOC0 and DOC1 = 0.120000
Jaccard's Score of DOC0 and DOC2 = 0.120000
Jaccard's Score of DOC0 and DOC3 = 0.115384
Jaccard's Score of DOC0 and DOC4 = 0.083333

Average Fitness (Jaccard's Score) of Document0: 0.28774

Chromosomes  Fitness

111111111111111111110000000000000                    [0.287744]
000010000000100010000111110000000                    [0.411692]
000010000000000101000010011110000                    [0.367556]
000000000001100100000010101001110                    [0.427473]
000000000001000100000010101000001                    [0.451212]

Average Fitness = 0.3891
```

Performing Crossover and Mutation: After getting the initial chromosome set we apply crossover on the chromosome set with a probability 'du'. After applying crossover we get a new set of chromosome sets and the fitness is calculated. We check whether there is any increase in the average fitness. If yes we take the chromosome set have high fitness and omit the others having less fitness. This is repeated until there is no increase in the fitness or for a fixed number of iterations. Mutation is changing the chromosome structure that is changing a bit in the chromosome. This is done very.
Rarely ones in thousand iterations. We have implemented both one point crossover and two point crossover over a probability of du for fixed iterations.

One point crossover example: at position

3 parents

1	0	1	1	1	0	1	0
1	0	0	1	0	0	1	1

Offspring

1	0	0	1	1	0	1	0
1	0	1	1	0	0	1	1

Two point crossover example: at position

2 and 7 parent

1	1	0	1	1	0	1	0
1	0	0	1	0	0	1	1

Offspring

1	0	0	1	1	0	1	1
1	1	0	1	0	0	1	0

Making the Index Table: Once we get a final chromosome where we get a maximum fitness or after a considerable iterations we take that chromosome set and then checks for zeros and ones in and take the words at whose position one is present and checks for the documents where we can find the word and then make a inverted index table. This index table is stored in a text file.

2 RETRIEVAL

For the purpose of retrieval we use different approaches as following

2.1 Database-oriented Approach [1]:

Databases that manage XML documents have to support queries on their logical structures and contents. Since access is based on logical structure it is decomposed and stored according to their tree structures, which are then stored in databases. In order to retrieve XML documents from such databases, XML queries are translated into database queries (typically in SQL). There are two approaches to designing database schemas for XML documents, as follows.

Structure-mapping approach. Database schemas represent the logical structure (or DTDs if they are available) of target XML documents. In a basic design method, a relation or class is created for each element type in the XML documents. In the structure-mapping approach, a database schema is defined for each XML document structure or DTD.

Model-mapping approach. Database schemas represent constructs of the XML document model. In this approach, a fixed database schema is used to store the structure of all XML documents. In both of the approaches to database schema design above, XML documents are decomposed into fragments composed of logical units. These decomposition approaches have drawbacks—it takes time to restore the entire or a large sub portion of the original XML documents, and processing certain text operations such as a proximity search beyond the boundaries of elements becomes very complex.

2.2 Information Retrieval-oriented Approach [2][3]:

Information retrieval techniques are applied to the processing and retrieval of XML documents, each of which can be considered to be a text document, with additional markup overheads. There are several methods to deal with the markup tags. One method simply discards all the tags. The advantage is its simplicity but the disadvantage is the loss of information, leading to lower retrieval performance. Another method is to extract important structural and contextual information from the XML document to index. A more comprehensive method is to index the tags as if they were the index terms.

Information Retrieval-oriented Approach mainly focuses on two different views of XML:

The document-centric view focuses on structured documents in the traditional sense (based on concepts from electronic publishing, especially SGML) where XML is used for logical markup of texts.

The data-centric view uses XML for exchanging formatted data in a generic, serialized form between different applications (e.g., spreadsheets, database records). This is especially important for e-business applications (e.g., for exchanging orders, bills).
IR oriented approach incorporates the IR approach of vagueness and imprecision for XML retrieval by means of the following features:

Index term weighting for both search terms and document terms, thus producing ranked results,

Specificity-oriented search for retrieving the most relevant parts of documents,

Data types with vague predicates for dealing with specific content in micro level markup, and

Structural vagueness, in order also to find close matches for structural query conditions.

2.3 Hybrid Approach [8]:

The basic idea of hybrid approach is that an inverted list of all the terms appearing in the content of the elements and values of the attributes are indexed. The index maintains the association of terms and the corresponding Xpaths and document identifier reaching that term. The Xpaths are generated and matched dynamically during searching. The rank value of the XML document is the aggregate of all the weights of the matched terms in the document. In another hybrid method, a multi-level solution is adopted. Upon a search request, category search based on information retrieval techniques is first performed. Once the category is found, a standard database query (e.g. In SQL) can be posted. These standard database queries are typically presented as a form that basically abstracts out the concept of a view in database, using XSL transformations. The benefit of this approach is that it is able to filter out fast the best set of databases to be searched for, and the database queries then allow very efficient retrieval of required information.

2.4 Probabilistic Information Retrieval Approach [9]:

Probabilistic Information Retrieval Approach makes use of ranking functions that are based on probabilistic information retrieval (PIR) ranking models. Probabilistic Information Retrieval (PIR) makes use of the following basic formulae from probability theory:

Bayes' rule: $$p(a\,|\,b)=\frac{p(b\,|\,a)p(a)}{P(b)}$$

Product rule: $$p(a,b\,|\,c)=p(a\,|\,c)p(b\,|\,a,c).$$

In probabilistic information retrieval, documents are ranked by decreasing order of their odds of relevance, defined as the following score

$$Score(t)=\frac{p(R\,|\,t)}{p(\bar{R}\,|\,t)}=\frac{\dfrac{p(t|R)p(R)}{p(t)}}{\dfrac{p(t|\bar{R})p(\bar{R})}{p(t)}}\propto\frac{p(t|R)}{p(t|\bar{R})}.$$

The final simplification in the above equation follows from the fact that p(R) and 1- p(R) are the same for every document and thus mere constants that do not influence the ranking of documents. In our adaptation of PIR models for structured databases, each tuple in a single database table D is effectively treated as a “document.” For a (fixed) query Q, our objective is to derive Score (t) for any tuple t, and use this score to rank the tuples. X is the set of attributes specified in the query, and Y is the remaining set of unspecified attributes. Denoting any tuple t as partitioned into two parts, t(X) and t(Y), where t(X) is the subset of values corresponding to the attributes in X , and t(Y) is the remaining subset of values corresponding to the attributes in Y . Replacing t with X and Y we get

$$Score(t) \propto \frac{p(t|R)}{p(t|d)} = \frac{p(X,Y|R)}{p(X,Y|D)} = \frac{p(Y|R)}{p(Y|D)} \cdot \frac{p(X|Y,R)}{p(X|Y,D)},$$

3 Preprocessing and Indexing Module [10][11]:

Before indexing of XML document preprocessing of document is to be done which includes stop word removal and stemming.

3.1 Stop Word Removal

Stop words is the name given to words which are filtered out prior to, or after, Processing of natural language data (text). In written or spoken natural language communication, stop words can be viewed as a type of signal noise which disrupts the ability to quickly ascertain the relevance of search results or the meaning and importance of words in a document. By filtering out such words, the message becomes clearer or more useful. Filtering of stop words to reduce index size (which is partly measured by the number of distinct words in the index) or to assist users in providing search queries that will net better results, by avoiding searches for words which appear in almost every document searched, which does not provide a way for the system to distinguish among documents and rank them appropriately.

A stop list (or stop list), the name commonly given to a set or list of stop words, is typically language specific, although it may contain words (and other character sequences like numbers and punctuation). A search engine or other natural language processing system may contain a variety of stop lists, one per language, or it may contain a single stop list that is multilingual.

Some of the more frequently used stop words for English include "a", "of", "the", "I", "it", "you", and "and". These are generally regarded as 'functional words' which do not carry meaning (are not as important for communication).

3.2 Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The process of stemming, often called conflation, is useful in search engines for query expansion or indexing and other natural language processing problems.

A stemmer for English, for example, should identify the string "cats" (and possibly "catlike", “catty” etc.) as based on the root "cat", and "stemmer", "stemming", "stemmed" as based on "stem". A stemming algorithm reduces the words "fishing", "fished", “fish", and “fisher” to the root word, “fish". The first ever published stemmer was written by Julie Beth Lovins in 1968. This paper was remarkable for its early date and had great influence on later work in this area. A later stemmer was written by Martin Porter and was published in the July 1980 issue of the journal Program. This stemmer was very widely used and became the de-facto standard algorithm used for English stemming.

There are two widely used stemming algorithms: Lovins algorithm and Porter’s algorithm.
Lovins algorithm specifies 260 suffix patterns and uses an iterative heuristic approach. The design of the algorithm was much influenced by the technical vocabulary with which Lovins found her working (subject term keywords attached to documents in the materials science and engineering field). The subject term list may also have been slightly limiting in that certain common endings are not represented (ements and ents for example, corresponding to the singular forms ement and ents), and also in that the algorithm’s treatment of short words, or words with short stems, can be rather destructive. The Lovins algorithm is noticeably bigger than the Porter algorithm, because of its very extensive endings list. But in one way that is used to advantage: it is faster. It has effectively traded space for time, and with its large suffix set it needs just two major steps to remove a suffix, compared with the eight of the Porter algorithm.

The Porter’s algorithm is a simpler version than Lovins algorithm. It uses 60 rules that are organized into sets. Conflicts within a set of rules are resolved before applying the next set. The rules are also separated into five distinct phases numbered 1 to 5. They are applied to the words in a document from phase 1 moving on to phase 5.Each phase will remove a type of suffix of the word. After the five stages, the stem of the word will be left. Since Porter’s algorithm is simpler and faster, we employed Porter’s algorithm in our implementation. During the process, all stop words and words having less than three letters will also be dropped, and any upper case characters will be changed to lower case.

3.3 Generation of Structural terms

XML retrieval requires taking into consideration the structural context of terms. We make use of the vector space model to represent this structural context. For example In Fig.1 all the structural terms for a XML document are shown.

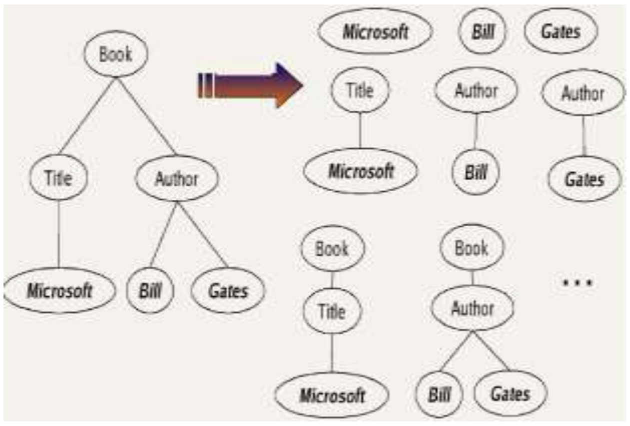


Fig. 1. A mapping of an XML document (left) to a set of structural terms (right)

3.4 Indexing

The last processing phase of this step involves the estimation of the distribution ("within document frequency") of all terms (literals and tags).Any path and term duplication is removed so that the resulting XML summary tree contains each path at most once. Thus the summary tree is smaller than the original XML document and it contains only those literal terms and tags that are important as far as indexing is concerned.

The term distribution may be in addition used to generate weights for each term in the summary tree in order to facilitate ranking.

The second processing step deals with the loading of the summary trees into the index structure. This involves the separation of content data from path data. Content data is raw text aimed to be stored in the inverted file and path data is structured text aimed to be

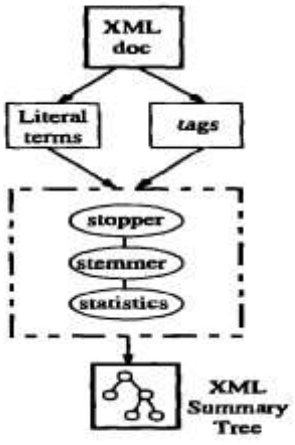


Fig. 2. XML Document Normalization Process [10]

Stored in the path index. The path index is a hierarchy of tags, which records every single path in the collection. A list of documents (posting list) is also stored alongside each entry of the inverted file.

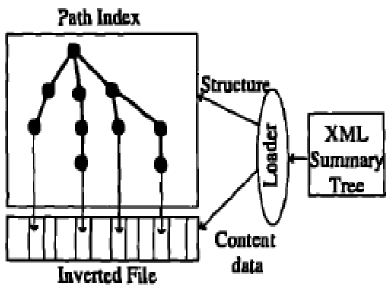


Fig. 3. Loading an XML document summary tree into the index structure

4 Query Evaluation [10][8]:

Structural terms are also generated for the user query after preprocessing (stop word removal and stemming).These structural terms are then to be matched with the structural terms generated from XML documents according to formula to evaluate context –resemblance

$$\mathrm{Cr}\,(q, d) = 1 + |q|/1 + |d|$$

Where q and d are the number of nodes in the query path and document path, respectively. The context resemblance function returns 0 if the query path cannot be extended to match the document path. Its value is 1.0 if q and d are identical. The following figure illustrates this.

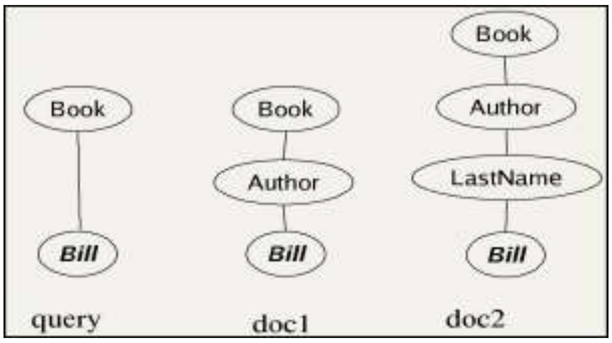


Fig. 4. Query-document matching for extended queries. Extended queries match a document if the query path can be transformed into the document path by insertion of additional nodes. The context resemblance function cr is a measure of how similar two paths are. Here we have cr (q, $d1$) = 3/4 = 0.75 and cr (q, $d2$) = 3/5 =0.6.

Upon receiving a query, the query evaluation process decomposes the query into its constituent conjunctive and disjunctive terms. Each query term consists of a path and raw text (literal part). The path is checked against the path index of the index structure. If there is a match, all the inverted lists are identified by generating a candidate list of documents and a candidate list of vocabulary terms. A ranking algorithm is also applied at this point to rank the documents based on frequency information held in the entries of the inverted lists.

4.1 Ranking [10]:

The proposed approach is simple and involves no complexity in implementation. Ranking does not only rely on the term weight based on the term distribution both within the XML document and the entire XML collection, but also on the structural position of the term. The final weight of a term is composed by multiplying the two estimates. The first component evaluates the weight as if the term was from an unstructured document. We will adopt the IDF which for a given term i is as follows:

$$IDF_i = log_2 \frac{N - n_i}{n_i}$$

Where N is the number of XML documents in the collection and n_i is the number of occurrences of the term i in the collection. Based on the above measure, the weight of the term i in the XML document j is given by

$$w_{ij} = freq_{ij} \times IDF_i$$

Where freq_{ij} is the within document frequency of term i in the document j. We refer to the measure w_{ij} as statistical weight

The final score for a document is computed by calculating the context resemblance similarity between the query and the document using the following formula

$$\text{sim-cr}(q,d) = \frac{\sum_{t \in V} \sum_{c_k \in C} \sum_{c_l \in C} \text{weight}(q,t,c_k)\,\text{weight}(d,t,c_l)\,\text{cr}(c_k,c_l)}{|\vec{q}|\,|\vec{d}|}$$

Here *V* is the vocabulary of (non-structural) terms; *C* is the set of all contexts (paths) occurring in the collection and the queries; and weight (*d*, *t*, c_k) is the weight of term *t* in context c_k in document *d*.

4.2 Artificial Neural Network for Image Retrieval: -

Artificial Neural Network (ANN) [13] is a network constructed using a basic processing unit called "neuron". Each neuron has a weight assigned to it and can do some simple operations such as thresh-holding using a function such as tan-sigmoid function, linear function etc. It takes an input performs a simple operation and gives an output. It has various applications. It can be used as a learning tool, classifier etc. The learning process of the network is a series of steps wherein the weights are changed to achieve a given performance criterion. The learning methods of ANN are supervised learning and unsupervised learning. In the training phase of supervised learning, both the input data and the desired output are given to the network. If the output of the network differs from the desired output, the internal weights of the network are adjusted. In unsupervised learning, only the input vectors are given the network and the network creates abstractions in the input space.

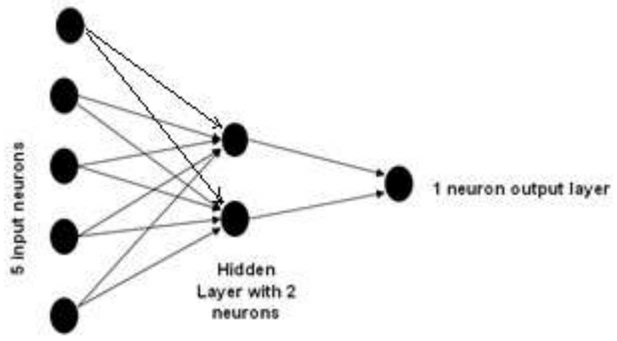


Fig. 5. Architecture of Neural Network

Some common architectures of neural networks are feed-back, feed-forward, and self-organizing neural networks. Feed-forward networks, such as the multi-layer perception (MLP) network, are built by successive layers of neurons through which the input propagates to give the output by the layer of the network. In feed-back networks [7], a fraction of the output is given back to the input layers of the network. In self-organizing map or competitive-learning networks, Neurons in all the layers receive identical input and they compete with their responses to the input. Self-Organizing Map (SOM) introduced by Kohonen is widely used unsupervised neural network learning algorithm while back-propagation algorithm is widely used supervised neural network learning algorithm.

5 IMPLEMENTATION

Architecture has been divided into two categories

1. CBIR Engine (built using Mat Lab R2008a)
2. Web Server (built using ASP .NET with IIS Server)

5.1 CBIR Engine:-

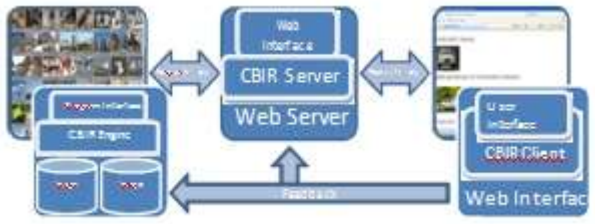


Fig. 6. Architecture of CBIR System

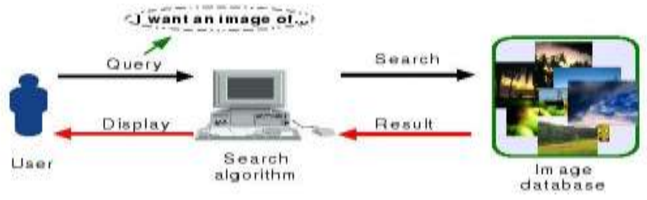


Fig. 7. Execution of CBIR System

Detailed Description of the CBIR System.

The system takes an image as an input by the user, extracts features from the image, measures similarity of the image with existing clusters of images using neural networks and ranks the retrieved images using relevance feedback. The system works in two stages: training and testing. The training consists of features extraction of the all images in the training image dataset and then separate neural network is trained for each cluster of the image database. In this process features of all the images of each cluster are extracted and then input to train a separate neural network for each cluster. The image database consists of several clusters of images where each cluster contains images of a particular type. In our training dataset we have taken 10 different clusters of images e.g. 'Cars', 'Buildings', 'Flowers', 'Weapons' etc. Each cluster contains around 50 training images whose features are used to train a separate neural network for each cluster. Content of an image can be expressed in terms of different features such as color, texture, shape etc. Retrieval based on these features varies depends on how the features are extracted and used. Since features in color, texture, and shape are extracted using different computation methods, different features may give different similarity measurements. In our system we have used following texture features of an image: Entropy, Homogeneity, Energy, Correlation, and Contrast. These features are extracted from the image and then input to the neural network for training. In testing Process user is asked to input an image and then features are extracted from the image and input to all the trained neural network of each cluster and if output of one or more than one neural network is greater than a certain threshold then all the most relevant images of those image clusters are shown to the user as the most similar images to the input image

Stage One: Texture Features Extraction: - The first stage of the system is image processing and feature extraction. With initial clusters of images the feature extraction process is applied to each image, but before feature extraction all the images are resized to image of dimension 256 X 256 that is 256 rows and 256 columns. This is necessary as the dimensions of images in a cluster may vary. So, all the images of a cluster are resized to 256 X 256 and converted from RGB to Gray level image. Now the 'Correlation', 'Energy', 'Contrast', 'Homogeneity', 'Entropy' features are extracted from the image. First a gray-level co-occurrence matrix from image matrix is created by calculating how often a pixel with gray-level (grayscale intensity) value i occurs horizontally adjacent to a pixel with the value j. Now all the above mentioned features can be extracted from the co-occurrence matrix.

Correlation is a measure of how correlated a pixel is to its neighbor over the whole image and it is defined as:

$$_{i,j} \frac{i \qquad i \quad j \qquad j \; p \; i, j}{\qquad\quad i \quad j \qquad\quad}$$

Energy is the sum of squared elements in the co-occurrence matrix and it is defined as:

$$_{i,j} \; p \; i, j \;\; ^{2}$$

Contrast is a measure of the intensity contrast between a pixel and its neighbor over the whole image and it is defined as:

$$_{i,j} \; |i \qquad j|^{2} \; p \; i, j$$

Homogeneity is a measure the closeness of the distribution of elements in the co- occurrence matrix to the co-occurrence matrix diagonal and it is defined as:

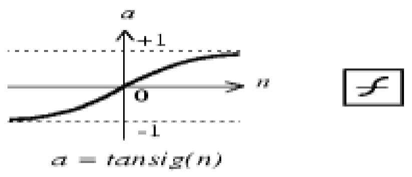
$$_{i,j} \frac{p \;\; i, j}{1 \quad i \quad j}$$

Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image and it is defined as:

$$_{i,j} \quad p(j)\log(p(j))$$

And all these features are normalized to a range of (0-1) for input to the neural network. These features are also stored in a file for later use.

Stage Two: Neural Network Training: - The features extracted are to be input to the neural network. The neural network architecture consists of 5 neurons in the input layer for the 5 features, 5 and 3 neurons in the 1st and 2nd hidden layers respectively and 1 neuron in output layer whose output will be 1 if the image belongs to a cluster or 0 if does not belong to a cluster. The neural network is trained using back-propagation algorithm and Tan-Sigmoid Transfer Function shown as below:



Tan-Sigmoid Transfer Function

Tan-Sigmoid Function:

$$\text{tansig} \quad n \quad \frac{2}{1 \quad e^{-2n}} - 1$$

It takes about 300 epochs to train each neural network belonging to different image clusters. Increase in the number of images in each image cluster in training does not cause much increase in the number of epochs but time required in each epoch is slightly increased. The results improve if number of training images are increased but it is noticed that time required to train the network will also increase slightly. This conclusion support the technique to store the user input image also in the relevant image cluster as it will increase the accuracy while sacrificing slight time complexity. Percentage of false-detections is also reduced by increase in number of training images used.

Stage Three: Testing using image as a query by User:-

The testing starts when user inputs a test image to retrieve similar images or when user selects an image cluster from a given list of image category list. When user inputs an image, the image is input to all the neural networks representing different image cluster. If the probability of input image for one or more neural networks is more than a particular threshold which is 0.6, then the image belongs to those image clusters. It is possible for an image to be in more than one image cluster if the input image contains more than one object. For e.g. If an image contains 'Car', and 'Building' both then it belongs to both categories. So, images from both the clusters should be retrieved with rank according to the output of the heuristic algorithm which adds the neural network probability and the relevance value to determine the rank of the retrieved images. If the user clicks on one of the retrieved image then relevance measure of the retrieved image is increased by a value 0.1 while relevance measure of those images which are not clicked is decreased by 0.01. Also the input image by the user is stored in those relevant clusters whose probabilities are more than the threshold 0.6. The new image is stored in the training dataset. The image is indexed using heuristic algorithm, its features are extracted and the network is re-trained using the features of new image also. These steps are repeated at each query. If the user inputs an image:

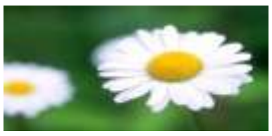


Fig. 8. Test Image

The result after testing for all the neural networks on test image is:

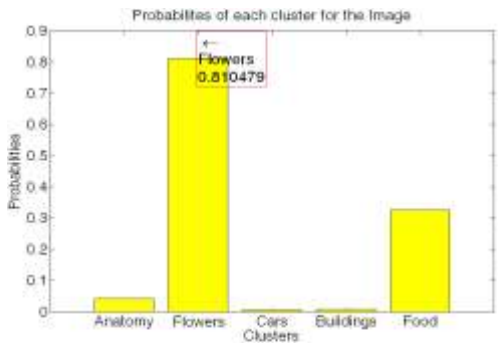


Fig. 9. NN Probability Output by CBIR Engine

6 Conclusion & Future Work

The indexing process generates an inverted file index and a path index containing all possible paths in the collection. The novelty and advantages of the proposed index can be summarized as follows:

1. It combines seamlessly two indexes; an inverted file and a path index.

2. The path index contains normalized tags. This feature may facilitate similarity search by *content and* structure.

3. Both the path index and inverted file are expanded and reduced dynamically by adding and removing XML documents.

The proposed indexing structure is powerful, easy to implement and maintain. The use of an index structure that combines an inverted index and a path index improves search efficiency for large collections of XML documents. The presence of markup tags in XML documents suggests that indexing the tags, or a normalized form of it, might be an effective approach to capturing document structure.

However, there are several problems with this approach. Different XML documents may use different tags to describe the same piece of information. It is really difficult, if not impossible, to form a standard description for every single piece of data. XML has been actually introduced to describe irregular data and as such the problem of having different a tag to describe the same piece of information is quite inherent to the XML language. It is then desirable to provide lists of equivalent tags. Tags with a high degree of semantic proximity must be handled as if they were the same. Synonymous tags (like "journal" and "periodical") must be indexed as if they were the same. Tag transformation might be useful in order to handle variant words using sound-like methods. Eliminating suffixes and prefixes from the tags may further facilitate the content regularity. Devising techniques for handling proximity measures between tags can further increase the accuracy of XML information retrieval. Scope for further research is towards tag similarity searching in order to address the problem of proximity searching and achieve more effective XML document ranking.

References

1. XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases MASATOSHI YOSHIKAWA and TOSHIYUKI AMAGASA ACM Transactions on Internet Technology, Vol. 1, No. 1, August 2001.
2. XIRQL: An XML Query Language Based on Information Retrieval Concepts NORBERT FUHR and KAI GROßJOHANN ACM Transactions on Information Systems, Vol. 22, No. 2, April 2004.
3. An Introduction to Information Retrieval Christopher D. Manning Prabhakar Raghavan Hinrich Schütze Cambridge University Press Cambridge, England.
4. Home>Research>Projects>Conceptual Indexing knowledge Technology Group Sun Microsystems
5. Jan Kuper, Horacio Saggion Hamish Cunningham Thierry Declerck, Franciska de- Jong , Dennis Reidsma, Yorick Wilks ,Peter Wittenburg Intelligent Multimedia Indexing and Retrieval through Multi-Source Information Extraction and Merging. International Joint Conferences on Artificial Intelligence, 10th IJCAI 408-414, 2003. Link http://dli.iiit.ac.in/ijcai/IJCAI-2003/PDF/061.pdf
6. A.Vinciarelli. IDIAP Research Institute. Rue du Simplon 4 - CH1920 Martigny (Switzerland).

7. Introduction to Text Indexing with Apache Jakarta Lucene. O'Really OnJava.Link http://www.onjava.com/pub/a/onjava/2003/01/15/lucene.html
8. A Survey of Search Engines for XML Documents RobertLuk, Alvin Chan, Tharam Dillon (FIEEE) and H.V. Leong Department of Computing, Hong Kong Polytechnic University.
9. Probabilistic Information Retrieval Approach for Ranking of Database Query Results SURAJIT CHAUDHURI GAUTAM DAS VAGELIS HRISTIDIS GERHARD WEIKUM ACM Transactions on Database Systems, Vol. 31, No. 3, September 2006.
10. Structured Information Retrieval in XML documents Evangelos Kotsakis 2O02 ACM SAC2002, Madrid, Spain.
11. An Introduction to Information Retrieval Christopher D. Manning Prabhakar Raghavan Hinrich Schütze Cambridge University Press Cambridge, England.
12. Exploring Interactive Information Retrieval: An Integrated Approach to Interface Design and Interaction Analysis Gheorghe Muresan , Bing Bai Conference RIAO2007, Pittsburgh PA, U.S.A. May 30-June 1, 2007 - Copyright C.I.D. Paris, France.
13. Muneesawang, P.; Guan, L., "A neural network approach for learning image similarity in adaptive CBIR", 2001 IEEE Fourth Workshop on Multimedia Signal Processing, Volume, Issue, 2001