In the Proceedings of the International Conference on Enterprise Information Systems, Setbal, Portugal, July 2001

A UNIFIED FRAMEWORK TO INCORPORATE SOFT QUERY INTO IMAGE RETRIEVAL SYSTEMS [*]

Cyrus Shahabi and **Yi-Shin Chen**
Integrated Media Systems Center and Computer Science Department
University of Southern California, Los Angeles, CA 90089-2564
E-mail:$[shahabi, yishinc]@usc.edu$

Key words: Image retrieval, information retrieval, fuzzy logic, soft query, clustering, incomplete data

Abstract: We explore the use of soft computing and user defined classifications in multimedia database systems for content-based queries. With multimedia databases, due to subjectivity of human perception, an object may belong to different classes with different probabilities ("soft" membership), as opposed to "hard" membership supported by conventional database systems. Therefore, we propose a unified model that captures both hard and soft memberships. In practice, however, our model significantly increases the computation complexity (both online and off-line) and the storage complexity of content-based queries. Previously, we introduced a novel fuzzy-logic based aggregation technique to address the online computation complexity. In this paper, we propose novel techniques to cluster sparse user profiles (i.e., items with missing data) to reduce both the off-line computation complexity and the storage complexity.

1 INTRODUCTION

Several applications in digital library, entertainment industry, consumer products, and e-commerce domains require accesses and queries of repositories of image data. Examples are virtual museums, online art galleries, family photo search tools, movie special-effect software packages, and e-store catalogue search tools. The challenge is to bridge the gap between the physical characteristics of digital images (e.g., color, texture, shape, etc.) that are used for low level comparison, and the semantic meaning of the images conceptualized by humans to query the database.

Several studies propose supporting similarity queries based on image perceptual features. Some studies (Niblack *et al.* 1993; Ennesser & Medioni 1995; Stricker & Orengo 1995; Shahabi & Safar 1999; Mirmehdi & Petrou 2000) proposed algorithms for automatic extraction of features such as color, texture and shape. Others (Faloutsos & et al. 1994; Fagin 1996; 1998; Simth & Chang 1996; Wu *et al.* 1996; Ma & Manjunath 1997; Hirata *et al.* 1999) focused on developing efficient indexing techniques for the extracted features. Both groups of studies are orthogonal to our study and can be used by our proposed system as well. However, their main shortcoming is their independence from the user perceptions. Therefore, other studies (Fagin & E.L.Wimmers 1997; J.Huang, S.R.Kumar, & Mitra 1997; Squire, Muller, & Muller 1997; Rui & Huang 1999; Wood, Campbell, & B.Thomas 1999; Doulamis *et al.* 1999; Porkaew, Mehrotra, & Ortega 1999) try to address this shortcoming by proposing various learning mechanisms to modify system parameters iteratively after obtaining user relevance feedback. This fine-tuning is either performed for the entire user body collectively, or for a single user during a single iterative session.

However, different people may have different perceptions on the same set of images and users do not want to train the system for every query, iteratively. Hence, the system should store some sort of a profile for each user to support full personalization. Towards this end, we uniformly conceptualize content-based querying as the task of classifying images into classes. These classes can overlap and are different for different users. In our previous work (Shahabi & Chen 2000b), we show how all the conventional content-based queries (e.g., finding images with similar color/texture/shape) as well as customized semantic queries (e.g., finding paintings of a specific style) can be uniformly supported within our soft query model. There, we explain that the user profile is simply the previous classification history of each user (i.e., the user's previous queries and feedbacks). Hence, for a new query the soft query model utilizes the user profile as well as the conventional image retrieval techniques. However, we realize that the system cannot assume the existence of profile (or past behavior) for every user. Hence, we extended our

[*]This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and ITR-0082826, NASA/JPL contract nr. 961518, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash/equipment gifts from NCR, IBM, Intel and SUN.

model to take the profiles of other users trusted by the current user into consideration[1]. Here, we borrowed concepts from fuzzy logic (Zadeh 1978) to aggregate users profiles and normalize their different (and maybe conflicting) definitions of fuzzy words (e.g., $high$ or low). In the worst case, our model reduces itself to the conventional image retrieval systems by only considering the image features and properties for querying. Our experiments reported in (Shahabi & Chen 2000b) demonstrated that the soft query model are consistently superior to the conventional image retrieval system in matching the users' expectations.

One major shortcoming of our soft query model is that it increases the time and space complexity of query processing due to maintaining profiles for every user and polling the experts profiles for every query. To be specific, both time and space complexity is $O(U \times D)$, where D is the number of images (i.e., size of the database), and U is the number of users in the system. To improve the efficiency of the soft query model, in (Shahabi & Chen 2000a) we proposed a novel method to compute fuzzy aggregations that reduces the time complexity to $O(f \times D)$ where f is the number of fuzzy sets (which is a small constant). Note that this time complexity is now comparable with the complexity of conventional image retrieval techniques[2].

As a side effect of this optimization, we need to perform an off-line process to partition the user space into f sets (corresponding to the f fuzzy terms) in $O(U^2)$ and also keep these partitions updated as new users register with the system. Moreover, the storage complexity remained at $O(U \times D)$. Hence, in this paper, we incorporate a clustering technique into the optimized soft query model to reduce the storage and off-line computation complexity to $O(k \times D) = O(D)$ and $O(k^2) = O(1)$, respectively, where k is a small constant representing the number of clusters. The challenge of clustering here is that the user profiles (i.e., users classification behaviors) are very sparse in practice. Therefore, we investigate different methods to perform the clustering on incomplete data sets. Finally, we report on two sets of experimental studies. One is comparing different methods to cluster items with missing data and the other is comparing the performance of our soft query methods with that of the conventional methods. Our results demonstrate that the *optimized Soft Query Model* outperforms the conventional methods significantly. The experimental results reported in this paper are more comprehensive than those reported in (Shahabi & Chen 2000b; 2000a). This is because reducing the computational complexity of our algorithms allowed us to run the experiments for larger populations. Furthermore, the new experiments on comparing soft query with and without clustering as well as incomplete data clustering are new.

The remainder of this paper is organized as follows. Section 2 explains our $optimized$ model for the soft query system. In Section 3, we discuss the behavior of the system and how the proposed model can be used to process various queries. The results of our evaluations as well as the details of the system implementation and our benchmarking method are described in Section 4. Section 5 concludes the paper.

[1]This is similar in concept to a popular TV game-show in the US where the players can poll the audience when they do not know the answer to a posed question. In our case, we only poll those members of the audience who we think they "know" the answer and we also provide appropriate weight to each member of audience based on our level of trust.

[2]Trivially, both our soft query system and the conventional system can benefit identically from some sort of a multidimensional index structure on the images such as the hash index proposed in (Gionis *et al.* 1999).

2 Soft Query Model

The primary objective of soft queries is to incorporate the user perceptions into the system. Two different types of data will be utilized during the query processing. The first is image information such as size, style and color similarity. This type of information is modeled as *Image Classes*, which is described in Section 2.1. The other is user confidence information, which we formally define in Section 2.2. This model has the disadvantage that the storage requirement and query processing time increase as the number of users grows. In order to alleviate this problem, we incorporate clustering techniques into our model and describe the method in Section 2.3.

2.1 Image Classes

Each image is associated with two categories of attributes. The first category as defined in Definition 2.1 is termed *features* ($\Im$), whose attributes are user-independent. The other category as defined in Definition 2.2 is called *properties* (T), whose attributes are user-dependent. Every image retains several features and/or properties.

Definition 2.1: The domain D is an image set. The domain V is a feature value set. $\Im$ is a function from D onto V.

$D = \{n | \, n \text{ is an image}\}$
$V = \{v | \, v \text{ is a value}\} \quad , \Im : d \in D \rightarrow v \in V$ ▮

Definition 2.2: Each property T_i partitions the set of all images into k classes, $C_1^{T_i}, C_2^{T_i} \cdots C_k^{T_i}$, where the classes might overlap and/or be partial. The membership of an image, n, to a class $C_j^{T_i}$ is determined by an evaluator (denoted as e) via the following probability function: $P(n \in C_j^{T_i} \mid e)$, whose value could be either a real number within the range of 0 to 1 or words describing user perceptions (e.g., *high* or *low*). Moreover, different evaluators can classify the same image into different classes. ▮

In practice, asking people to describe their perceptions with very precise values is almost impossible. Moreover, different people have different interpretations of words. That is, the information describing personalities, physical features, preferences and personnel evaluation is imprecise. In order to handle this uncertainty during the query processing, fuzzy logic (FL) is adapted by our system. The concept of FL was first introduced by Zadeh (Zadeh 1978) to problems for which precise formulation is not possible. A variety of studies (Fagin 1998; 1996; Nepal, Ramakrishna, & Thom 1999; Doulamis *et al.* 1999) in content-based image retrieval adapted this concept for shortening the gap between the physical characteristics of digital images and the semantic meaning of the images conceptualized by humans.

With the help of FL, the soft query system can store and employ human's fuzzy perceptions. First, users pick up the words already defined in the system (hereafter denoted as fuzzy sets) to express their opinions. Then, all fuzzy words will be converted to real values customized for the user perceptions(see (Shahabi & Chen 2000b) for details).

Note that using the same model, we can capture conventional image retrieval operations such as finding all images similar to a query image in color, shape or texture(see (Shahabi & Chen 2000b) for examples). Moreover, the model can utilize the presence of multiple algorithms for the same comparison function. However, the actual processing for capturing the evaluation by users and functions are different. The membership values, which are determined by real users, exist in the system a priori. Conversely, the membership values determined by functions or algorithms are computed on demand when needed.

2.2 User Recommendation and Confidence

The model described in Section 2.1 assumes that there always exists a membership value for any image-user pair to a given class. However, this assumption may not always hold true. For example, a user may have not assigned membership values for some newly introduced images, or his/her classification data may not be recorded in the database. In these cases, the system requires some reference data (e.g. the values assigned by other evaluators) to estimate the membership values of images to the class. Therefore, storing each user's level of confidence to other users is critical in this model. The formal definition is given as follows.

Definition 2.3: E denotes a set of evaluator representatives in the database. O represents the set of observers who have assigned reference confidence values to evaluators. π is a confidence value for an observer o to an evaluator e; $\pi \ : \ o \in O, \ e \in E \ \ \rightarrow b \in F$. Note that because the confidence value is a form of human judgment, we restrict it to fuzzy sets (denoted as F). This approach not only fits human nature but also reduces the computation complexity (described in Section 3.1.1).▮

Since the human judgments are represented by fuzzy sets, they will be converted to real values by using FL as described in Section 2.1.

2.3 Clustering Users Based on the Image Property Memberships

Without loss of generality and to simplify the discussion, we assume all evaluators are real users so that the storage complexity of the system is $O(U \times D \times Z + U^2)$, where U is the total number of users in the system, D is the number of images, and Z is the number of properties. In most cases, $D > U$ and Z is a constant, therefore, the storage complexity can be reduced to $O(U \times D)$. During clustering, a user-behavior item can be defined as a list of image property memberships determined by the corresponding user. Every evaluator has its fixed-size user-behavior item. The dimension of this item is $D \times Z$ and the total number of items is originally U. In order to reduce the complexity, we incorporate clustering techniques into our soft query model to cluster similar users. After clustering, the storage complexity can be reduced to $O(k \times D) = O(D)$, where k is a small constant representing the number of clusters. Furthermore, clustering also reduces the off-line processing complexity of aggregation functions as discussed later in Section 3.1.1.

Numerous clustering algorithms (Duran & Odell 1974; Janssen, Marcotorchino, & Proth 1983; Kaufman & Rousseeuw 1990; Linde, Buzo, & Gray 1980; Ng & Han 1994; Zhang, Ramakrishnan, & Livny 1996) have been proposed in the literature with various niches. Some of them stress on time complexity and others concentrate on obtaining optimal clustering representatives. To the best of our knowledge, almost all clustering techniques rely on the assumption that data sets are complete without missing data in any dimensions. However, this assumption is often not true and especially invalid with our application because a large part of the data is missing. In practice, the system seldom obtains complete information. For example, a user may not provide a membership value for an image to a property class, he/she has not yet assigned membership values for some images to a newly introduced class, or the user may not feel confident enough to perform the classification. In order to cluster data under these circumstances, we describe some approaches to address this problem in Section 2.3.1. The method of generating cluster representatives is presented in Section 2.3.2.

2.3.1 Clustering Incomplete Data

Generally speaking, existing methods for handling an incomplete data set in clustering fall into three categories (Kaufman & Rousseeuw 1990). Below, we list these approaches and discuss their advantages and weaknesses.

Interpolation (InP) With this approach, the missing data will be replaced based on the values of its neighbors (within the same dimension or the same item) prior to clustering. It is the most common approach in statistics (Little & Rubin 1987). The replacing values can be calculated by techniques such as mean imputation and the Expectation-Maximization (EM) algorithms (Ghahramani & Jordan 1994; Little & Rubin 1987). Mean imputation is a common heuristic where the missing values are replaced with the means of their neighbors and EM is an iterative method of determining maximum likelihood for fitting a model to data.

However, due to large amount of missing data and diverse user opinions with our application, mean imputation cannot provide accurate estimation. Furthermore, since there might be no relationship between nearby dimensions of the same item or corresponding dimensions in neighboring items, obtaining likelihood parameters could be inaccurate. Moreover, EM algorithm cannot guarantee finding the global optimal.

Elimination (EL) This approach is simply removing dimensions consisting of insufficient data. The main benefit is the ease of implementation. However, unlike interpolation, this approach is seldom mentioned in the literature. The primary reason is that it is only good for applications with a small portion of data missing.

Elimination during distance computation (EDC) During clustering, while the system is calculating the distance between a pair of items, it only uses those dimensions whose values are present for both items. If a pair of items does not have any common measured dimensions, two remedies can be applied. One is removing one or both of the items from the data set, and the other is filling in some arbitrary values for the dimensions. The advantages of this approach are no preprocessing task is required prior to clustering and more correct data is available to work with during clustering. There are two weaknesses to this approach. One is that the result of distance measurements must be normalized and the other is that this approach could only be applied in the condition that all dimensions have the same weight.

With the aim of solving the missing data problem in all circumstances, we propose the following two new methods that are the combinations of interpolation and elimination approaches.

Elimination and interpolation (EI) The procedure can be described in two steps. First, we perform the elimination process followed by clustering. Second, in order to replace the missing data, we perform interpolation processes within each cluster. This method can alleviate the drawback of elimination because it works with more data during the final clustering. Moreover, because interpolation is done within each cluster, the values of corresponding dimensions in neighboring items are closer.

EI and iterative interpolations (EICI) This method is very similar to EI except for the additional clustering and interpolation processes. After acquiring the complete clustered data with EI, EICI performs one more clustering pass on the complete data. The original missing values are replaced by using interpolation within each cluster. This process can be repeated until the system acquires satisfying result. However, in order to simplify the experiments, we fix the number of interpolations at two. With additional interpolation steps, the final result would become more accurate than the result of EI.

The corresponding experimental setup and results are presented in Section 4. Our results indicate the superiority of EICI over all the other approaches.

2.3.2 Generation of the Cluster Representative

Currently, we conduct our experiments utilizing the K-means (Linde, Buzo, & Gray 1980) [3] clustering algorithm for its simplicity. However, K-means has some major drawbacks. First of all, the results heavily depend on the initial cluster representatives. Secondly, the results are not robust to the existence of outliers. In other words, it cannot deal adequately with “noise” produced by users who cannot get clustered naturally with other users. Finally, it is possible that some cluster representatives never get the chance to be updated because no user will be classified into their clusters. In order to alleviate these problems, we perform the clustering procedure for several times and then pick the best clustering result.

Thus, using the K-means method with any distance measure, we can cluster U items into k sets and obtain k cluster representatives, which will be the only referenced and stored evaluators within our soft query model. The cluster representative is identified as the cluster centroid provided by K-means.

3 Soft Query Behavior

The soft query results are obtained by incorporating user perceptions into the query processing. Therefore, the system needs a special mechanism for integrating image information and user perceptions into the query

[3]Other clustering algorithms can also be incorporated into our system, but, the focus of this study is not on investigating different clustering algorithms.

processing. The flow of image query processing is described in Section 3.1. The system evaluation method is then presented in Section 3.2.

3.1 Query Processing Model

Each query statement can be decomposed into several atomic clauses. Some clauses access and process feature-values, and the others utilize property classes. Therefore, our soft query model consists of two different query processing methods: feature query and property query. These two query models compute the image membership separately and combine their results at the final step. The final query results are ordered according to their membership values. The feature query model is straightfoward and simply involves the Boolean evaluation of the predicates (e.g., image.cost $\geq$ \$5000). The property query model is described in Section 3.1.1, and the combined features-property process is discussed in Section 3.1.2.

3.1.1 Property Queries

A sample property query is: "Find all images with the *classic* style." The results of this type of queries depend on the user submitting the queries. The probability of an image belonging to the result set is presented by a real number between 0 and 1. The challenge is how to take all the information about the trusted evaluators and their classifications into account when processing a query.

The membership $\lambda_{d,c}$ of image d to a property class c given by user u can be computed using various aggregation functions. However, the aggregation functions with a triangular norm (Fagin 1998) are preferred with our system. These aggregation functions g satisfy the properties - *Conservation, Monotonicity, Commuatativity* and *Associativity*.

With these properties, the query optimizer can replace the original query with a logically equivalent one and still obtain the exactly same result. The optimized aggregation function we propose:

$$\begin{aligned} E_f &= \{e \mid e \in E, \pi_{u,e} = f\} \\ \theta^*_{d,c,f} &= f \times \max_{e \in E_f}\{P(d \in c \mid e)\} \\ \lambda_{d,c} &= \max_{f \in F}\{\theta^*_{d,c,f}\} \end{aligned} \tag{1}$$

Note that there are cases where the system does not have enough information to compute $P(d \in c \mid e)$ or $\pi_{u,e}$. In these cases, the system estimates these values from the "*system profile*", which can be obtained by simply averaging the data in the database or by a specific algorithm defined in advance.

These aggregation functions not only have the advantage of being incorporable into query optimizers, but also reduce the complexity of query evaluation. A naive weighted aggregation function may use the user confidence data as the weight factors when retrieving the images. Hence, its complexity becomes a function of the product of the number of evaluators to the number of images: $O(U \times D)$ where D is the image set and U is the number of users in the system. In comparison, the total computation complexity of our aggregation function as described in Equation(1) is only $O(f \times D) = O(D)$. This is because we only compute the final soft query membership by only iterating through possible values of the fuzzy term f(see (Shahabi & Chen 2000a) for more details).

As a side effect of this optimization, we need to perform an off-line process to partition the user space into f sets (corresponding to the f fuzzy terms) in $O(U^2)$ and also keep these partitions updated as new users register with the system. Hence, after incorporating our clustering technique into the optimized soft query model, the off-line computation complexity is $O(k^2) = O(1)$, where k is a small constant representing the number of clusters.

The cases where the query includes a Boolean combination of atomic clauses also need to be considered. We use the standard rules of fuzzy logic instead of traditional Boolean operations. They are:

$$\begin{array}{rll} \text{Conjunction rule:} & Q_A \wedge Q_B & = \min\{Q_A, Q_B\} \\ \text{Disjunction rule:} & Q_A \vee Q_B & = \max\{Q_A, Q_B\} \\ \text{Negation rule:} & \sim Q_A & = 1 - Q_A \end{array}$$

3.1.2 Combination of Feature and Property Queries

Finally, in this section, we explain the processing of queries on both features and properties of images. For example, "Find all *classic* images that cost less than \$5000." For this type of queries, we continue to use the standard rules of fuzzy logic as in Section 3.1.1. The only difference is that the possibility of an image belonging to a feature class is either 0 or 1.

Therefore, our proposed query processing model for *soft query* can capture user queries on both features and properties in a unified manner. To compare our model with the conventional image retrieval systems, consider a typical "query by example" on color, shape, and texture of images. Conventional systems compute a weighted average over these perceptual features to measure the similarity distance between two images. The weights are assigned and fine-tuned either directly by the user, or by the system after several iterations of monitoring the users' feedbacks. For example, the system will assign higher weights to the color feature for a color-oriented user. These perceptual features can also be modeled within our system as different properties. Subsequently, their weights are assigned not only by the user and his/her previous feedbacks but also by the other users/evaluators trusted by this user. In addition, our model can employ various feature extraction algorithms, semantic classes (e.g., image style) and soft memberships.

3.2 Query Evaluation

To evaluate the accuracy of our proposed approach, we need to compare the results returned by our system for a query submitted by a user u with the exact results that u expects to observe. Two most commonly used measures in information retrieval are precision and recall. These two evaluation methods are not quite applicable to our soft query system. Unlike current image retrieval systems, the soft query system deals with the ambiguities associated with classifying images. The classification problem has no standard answer.

Therefore, we use an alternative relevant performance measure: *prediction quality*. The system measures the degree of similarity between these two vectors by cosine function as:

$$\text{Similarity}(Q, B) = \frac{\sum_{i=1}^{n} Q_i \times B_i}{\sqrt{\sum_{i=n}^{n} Q_i^2 \times \sum_{i=n}^{n} B_i^2}} \quad (2)$$

This similarity value can be considered as an acceptance rate. For example, if the user does not modify any rating in the query result, the similarity value will be 1. This suggests that the user is in total agreement with the query result. On the other hand, if the measure is less than 1, it represents that the user does not totally agree with the query result.

4 Performance Evaluation

In this section, two sets of experiments and their corresponding results are described. In the first set, we compare the performances of incomplete data clustering techniques described in Section 2.3.1. With the second set, to compare whether soft query model provides better retrieval performance than conventional image retrieval methods, we simulate various setups and discuss their performances. Each section contains the experimental setups, benchmarking methods and the details of our results.

4.1 Incomplete Data Clustering Techniques

4.1.1 Benchmarking Method

The incomplete data clustering techniques are implemented in Java and on top of Microsoft Access 2000, which is running on a Pentium II 233MHZ processor with Microsoft NT4.0.

In order to populate data for evaluation purposes, we propose a parametric algorithm. By changing the parameters of the algorithm, we can simulate various database contents. The benchmarking method incorporates a 3-dimensional matrix (hereafter denoted as cube $\mathring{A}_0$) that contains the perfect knowledge about the classification behaviors of all users. $\mathring{A}_0$ has the following three dimensions: users, classes, and images. Each cell $\mathring{a}(i, j, q)$ determines the membership value of image q to class j assigned by user i. The cells are each assigned a random value distributed between 0 and 1.

Before populating cube $\mathring{A}_0$, the algorithm first randomly generates k cluster representatives (or centroids). Each cluster representative is a matrix, Ç, with D rows and Z columns, where D represents the number of images and Z is the number of classes. The element $ç_{ij} \in$ Ç represents the membership for image i to class j defined by cluster representative Ç.

Subsequently, the system constructs clusters with each having U/k user matrices generated around the cluster's centroid, where U is the number of users. Each element of user matrices is assigned a random value according to $ç_{ij}$. Specifically, if $Ł^y$ is the yth user matrix, then the element l_{ij}^y will have the value $l_{ij} \pm \tilde{a}$, where $\tilde{a}$ is a random number in $[0, 0.2]$. The parameter $\tilde{a}$ introduces noise to the perfect knowledge of $\mathring{A}_0$.

Finally, by hiding some of the cells of cube $\mathring{A}_0$ from the clustering techniques, we can simulate various levels of missing information. The number of hidden cells is determined by a missing factor. Thus, the lower the missing factor, the more complete the data in the cube.

In each clustering process, the result is obtained by averaging the distances (Dis) of all elements to their cluster centroids. In order to measure the performances of these techniques, the system will perform an extra clustering on the complete data, i.e., the original $\mathring{A}_0$, and obtain the averaging distance (Dis_0). Denoting the result of incomplete data clustering process as Dis_i, the system can calculate the percentage of degradation defined as follows.

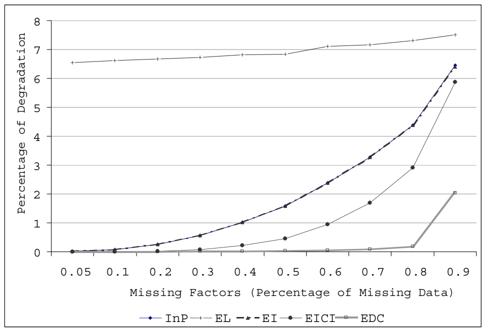
$$\text{\% of Degradation}\,(Dis_i, Dis_0) = \frac{Dis_i - Dis_0}{Dis_0} \quad (3)$$

4.1.2 Experimental Results

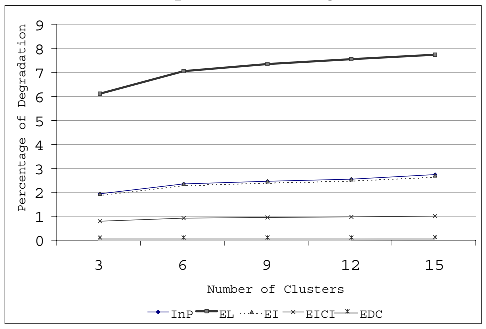
We conducted several sets of experiments to compare the different incomplete data clustering techniques. In order to perform a fair comparison, we selected cosine distance (see Equation(2)) as the common distance measure for all clustering processes. The results shown for each set of experiments are averaged over several runs, where each run is executed with different seeds for the random generator functions. The coefficient of variances among these runs is always smaller than 5% [4], which shows the independence of our results from a specific run.

The results are consistent as different settings are applied; therefore, we only select a few key figures to discuss the performances of these methods. We fix the number of images at $D = 60$, and the number of classes at $Z = 3$. The Y-axis in Figures 4.1.2 and 4.1.2 depicts the percentage of degradation as computed by Equation (3).

[4]The variances among these runs are always smaller than 1.5%.

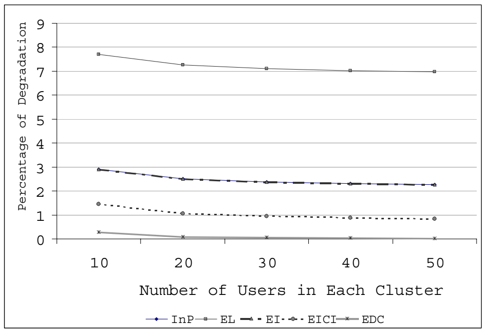


a. The impact of missing factors

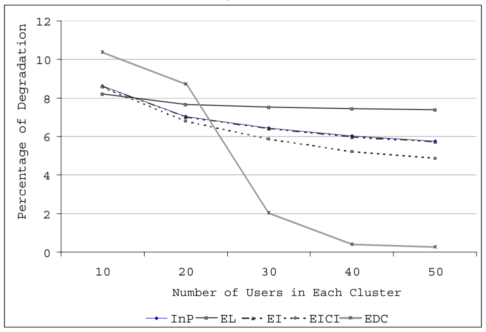


b. The impact of the number of clusters

Figure 1: Comparison of incomplete data clustering techniques



a. Missing factor = 0.6



b. Missing factor = 0.9

Figure 2: The impacts of the number of users

Figure 4.1.2.a depicts the impact of missing factor on performance degradation. The X-axis in this figure is the value of missing factor. We set the number of clusters at 6, and the number of users in each cluster at 30 in this experiment. This figure indicates that the EDC method surpasses other methods. The reason is that EDC can incorporate more correct data to work with during clustering. It is not surprising that the elimination method has the worst performance among all because the least amount of data is used. However, if the output of distance measurements cannot be normalized, EICI will be a better choice.

Figure 4.1.2.b illustrates the impact of the number of clusters on performance degradation. The X-axis in this figure is the number of clusters varying from 3 to 15. We set the number of users in each cluster at 50, and the missing factor at 0.6 in this experiment. This figure points out that EICI is only slightly impacted by the number of clusters, and EDC is not affected at all.

Figures 4.1.2.a and 4.1.2.b illustrate the impacts of the number of users on performance degradation. The X-axis in these two graphs is the number of users. We fix the number of clusters at 6 for both experiments. The missing factor for Figure 4.1.2.a is 0.6 and for Figure 4.1.2.b is 0.9. Both graphs indicate that the more information in each cluster, the better the performance. However, Figure 4.1.2.b shows that for small clusters when most of the data is missing and there is not enough information to be incorporated, EDC has the worst performance.

In summary, EICI will be the best choice under either of the two circumstances: 1) the data-missing rate is extremely high and there is not enough information to be incorporated, or 2) the output of distance measures cannot be normalized (see 2.3.2 for details).

4.2 Soft Query Model

4.2.1 Benchmarking Method

Our soft query system is implemented in Java and on top of Informix Universal Database Server V9.14 with Excalibur Image DataBlade Module which is running on a 300MHZ dual processor Sun Enterprise 450 Server with SunOS 5.6.

The benchmarking method for soft query model also utilizes a cube for populating data in a similar fashion. However, in order to deal with the missing data problem, the cube used for populating *Confidence-Data* and *Soft-Membership-Data* relations is first processed by EICI due its superiority over the other techniques. Then, the clustering process is performed to obtain k user representatives. All cells of these representatives will be populated into the *Soft-Membership-Data* relation.

The *system profile* is considered as a user representative. Its membership values for each image x to class c are obtained by averaging over all original users' membership values for x to c. Since the system profile is the only information utilized by conventional image retrieval systems, its membership value is computed assuming the perfect knowledge (however, simply averaged) on user classification behaviors. To incorporate imperfect or wrong knowledge into our system, all information except system profile will be tuned in a noisy process, which utilizes the Gaussian distribution prior to clustering. The level of noise varies from 0 (no noise introduced) to 10.

Subsequently, to assign confidence values for evaluators in the relation, *Confidence-Data*, we compare their classification behaviors (randomly selected from the cube and restricted to user representatives) to the classification behavior of the observer. The closer their behaviors, the higher the confidence values. The number of referenced evaluator representatives that we use to populate *Confidence-Data* is determined by a *selectivity factor*($\mathcal{P}$). Thus, the lower the selectivity factor, the less knowledge our system has about its users. Finally, all the real values between 0 and 1 must be translated into fuzzy terms. To achieve this, we use the randomly populated *User-Perception-Standard* of the *User* relation.

4.2.2 Experimental Results

We conducted several sets of experiments to compare our soft query system with the conventional image retrieval approach (denoted hereafter as CIR). In these experiments, we observed a significant margin of improvement over CIR in matching the user expectations in various settings. It is also shown that the performance of our soft query system is independent of the number of images and classes. However, we only stress the improvements achieved by applying the clustering process in this paper. Please consult our previous publication (Shahabi & Chen 2000b) for more results about other settings.

The results shown for each set of experiments are averaged over many runs, where each run is executed with different seeds for the random generator functions. The coefficient of variance among these runs is always smaller than 5% [5], which shows the independence of our results from a specific run.

$$\text{\% of Improvement}(SQ, CIR) = \frac{SQ - CIR}{CIR} \quad (4)$$

In Figure 4.2.2, we demonstrate the improvements achieved by applying the clustering process and compare the system performance under three different levels of noise introduced into the soft query system. The X-axis is the number of clusters varying from 5 to 300, where having 300 clusters (given 300 users) represents no clustering. The Y-axis of Figure 4.2.2 depicts the similarity distance computed by Equation (2) and the percentage of improvement of soft query (level of noise is 0) over CIR computed by Equation (4). The benchmark settings of this figure are set at the number of images $X = 500$, the number of users $Y = 300$, the number of classes $Z = 3$, and the number of user representatives trusted by each user is 3.

Figure 4.2.2 indicates that our soft query system achieves better performance with an increasing number of clusters. This is because the more clusters the system has, the closer the cluster representative to the user. As can be seen, in the majority of cases, the performance of the soft query degrades with higher noise levels, but it always outperforms CIR. The reason is that soft query uses Equation(1) to cancel out the impact of noisy data by considering the entire population as opposed to averaging without any consideration.

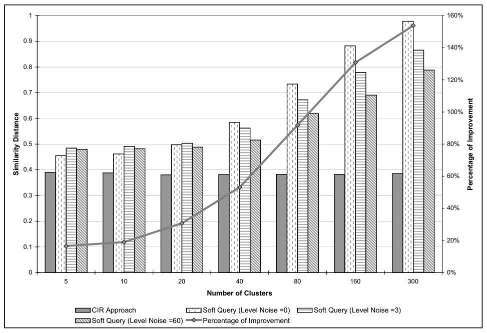


Figure 3: Soft query vs. CIR

5 Conclusion

We proposed a unified and efficient model to support soft querying and classification on image databases. The model borrows from the field of fuzzy logic and maintains two kinds of profiles. The image membership profile contains the membership values for each evaluator representative. The reference profile for each user contains the level of confidence that this user has to evaluator representatives. The experimental results show that this model is more accurate than the conventional methods. However, the time and space complexity of query increases dramatically when the number of users grows. We addressed this problem by introducing clustering techniques into the system. Moreover, comparing to other aggregation functions such as weighted averaging, our aggregation functions do not increase the complexity as the number of users grows.

We also proposed and compared different techniques that can perform the clustering on the users sparse profiles. Here, we investigate some traditional

[5] The variance of these runs is smaller than 1.5%.

techniques to cluster items with missing data as well as proposing some novel techniques. Our experimental results illustrate that one of our proposed technique, EICI, outperformed all the other techniques for our application scenario.

References

Doulamis, A.; Avrithis, Y.; Doulamis, N.; and Kollias, S. 1999. Interactive content-based retrieval in video databases using fuzzy classification and relevance feedback. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 2.

Duran, B., and Odell, P. 1974. *Cluster Analysis, A Survey*. Springer-Verlag.

Ennesser, F., and Medioni, G. 1995. Finding waldo, or focus of attention using local color information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17:805–809.

Fagin, R., and E.L.Wimmers. 1997. Incorporating user preferences in multimedia queries. In *Proc. 1997 International Conference on Database Theory*.

Fagin, R. 1996. Combining fuzzy information from multiple systems. In *Proc. Fifteenth ACM Symp. on Principles of Database Systems*, 216–226.

Fagin, R. 1998. Fuzzy queries in multimedia database systems. *ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems*.

Faloutsos, C., and et al. 1994. Efficient and effective querying by image content. *J. of Intelligent Information System* 3:231–262.

Ghahramani, Z., and Jordan, M. 1994. Supervised learning from incomplete data via an em approach. In *Cowan, J.D., Tesauro, G., and Alspector, J. (eds.) Advances in Neural Information Processing Systems 6*, 120–129. San Mateo, CA: Morgan Kaufmann.

Gionis, A.; Indyk, P.; ; and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*.

Hirata, K.; Mukherjea, S.; Li, W.; and Hara, Y. 1999. Integrating image matching and classification for multimedia retrieval on the web. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 1.

Janssen, J.; Marcotorchino, J.; and Proth, J. 1983. *New Trends in Data Analysis and Applications*. Elsevier Science Publisher B.V.(North-Holland).

J.Huang; S.R.Kumar; and Mitra, M. 1997. Combining supervised learning with color correlograms for content-based image retrieval. *ACM Multimedia 97 - Electronic Proceedings*.

Kaufman, K., and Rousseeuw, P. 1990. *Finding Groups in Data: and Introductions to Cluster Analysis*. Jone Wiley and Sons.

Linde, Y.; Buzo, A.; and Gray, R. M. 1980. An algorithm for vector quantization design. *IEEE Transations on Communictions* 28:84–95.

Little, R., and Rubin, D. 1987. *Statistical analysis with missing data*. New York : Wiley.

Ma, W., and Manjunath, B. 1997. Netra: A toolbox for navigating large image database. In *IEEE International Conference on Image PRocessing*.

Mirmehdi, M., and Petrou, M. 2000. Segmentation of color textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22:142–159.

Nepal, S.; Ramakrishna, M.; and Thom, J. 1999. A fuzzy object query language (foql) for image databases. In *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications*, 117–124.

Ng, R., and Han, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference*, 144–155.

Niblack, W.; Barber, R.; Equitz, W.; Flickner, M.; Glasman, E.; Petkovic, D.; and Yanker, P. 1993. The QBIC project: Querying imgaes by content using color, texture and shape. In *Proceeding of SPIE Conference on Storage and Retrieval for Image and Vedio Databases*, volume 1908, 173–187.

Porkaew, K.; Mehrotra, S.; and Ortega, M. 1999. Query reformulation for content based multimedia retrieval in mars. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 2.

Rui, Y., and Huang, T. 1999. A novel relevance feedback technique in image retrieval. In *ACM Multimedia 1999*.

Shahabi, C., and Chen, Y.-S. 2000a. Efficient support of soft query in image retrieval systems. In *Proceeding of SSGRR 2000 Computer and eBusiness Conference*.

Shahabi, C., and Chen, Y.-S. 2000b. Soft query in image retrieval systems. In *Proceeding of SPIE Internet Imaging , Electronic Imaging 2000*, volume 3964, 57–68.

Shahabi, C., and Safar, M. 1999. Efficient retrieval and spatial querying of 2d objects. In *IEEE International Conference on Multimedia Computing and Systems*, volume 1, 611 –617.

Simth, J., and Chang, S. 1996. Visualseek: A fully automated content-based image query system. In *ACM Multimedia 1996*.

Squire, D.; Muller, W.; and Muller, H. 1997. Relevance feedback and term weighting schemes for content-based image retrieval. In *ACM Multimedia 97 - Electronic Proceedings*.

Stricker, M., and Orengo, M. 1995. Similarity of color images. In *Proceeding of SPIE Conference on Storage and Retrieval for Image and Vedio Databases III*, volume 2420, 381–392.

Wood, M.; Campbell, N.; and B.Thomas. 1999. Interactive refinement by relevance feedback in content-based digital image retrieval. In *ACM Multimedia 1998*.

Wu, D.; Agrawal, D.; Abbadi, A.; Singh, A.; and Smith, T. 1996. Efficient retrieval for browsing large image databases. In *SIKM*, 11–18.

Zadeh, L. 1978. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems 1(1)* 3–28.

Zhang, T.; Ramakrishnan, R.; and Livny, M. 1996. Birch: An efficient data clustering method for very large databases. In *Proceeding of ACM SIGMOD Conference on Management of Data*, 103–114.