To Appear in the ACM/Springer Multimedia Systems Journal, Special Issue on Content-Based Retrieval, 2002

Yoda, An Adaptive Soft Classification Model: Content-based Similarity Queries and Beyond

Yi-Shin Chen and Cyrus Shahabi
Integrated Media Systems Center
Computer Science Department
University of Southern California
Email: {yishinc,shahabi}@usc.edu

Providing a customized result set based upon a user preference is the ultimate objective of many content-based image retrieval systems. There are two main challenges in meeting this objective: First, there is a gap between the physical characteristics of digital images and the semantic meaning of the images. Second, different people may have different perceptions on the same set of images. To address both these challenges, we propose a model, named Yoda, that conceptualizes content-based querying as the task of soft classifying images into classes. These classes can overlap, and their members are different for different users. The "soft" classification is hence performed for each and every image feature including both physical and semantic features.

Subsequently, each image will be ranked based on the weighted aggregation of its classification memberships. The weights are user dependent and hence different users would obtain different result sets for the same query. Yoda employs a fuzzy-logic based aggregation function for ranking images. We show that in addition to some performance benefits, fuzzy aggregation is less sensitive to noise and can support disjunctive queries as compared to weighted-average aggregation used by other content-based image retrieval systems.

Finally, since Yoda heavily relies on user dependent weights (i.e., user profiles) for the aggregation task, we utilize the users' relevance feedback to improve the profiles using genetic algorithms (GA). Our learning mechanism requires less user interaction and results in faster convergence to the user's preferences as compared to other learning techniques.

1. INTRODUCTION

As the demand for digital information and the amount of available data increases dramatically, the task of retrieving accurate information becomes increasingly essential in various domains. Whether we are looking for the latest news on websites, seeking interesting papers for research from digital libraries, searching for images in virtual museums, downloading music clips, or purchasing products in e-commerce sites, we need high-quality mechanisms to locate the information we desire for alleviating the problem of information overload.

Content-based retrieval systems are a type of information locating mechanisms that can find the queried information according to its content. In these systems [Niblack et al. 1993; Blackburn and DeRoure 1998; Li 2000; Lew 2000; Lin

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and IIS-0082826, NIH-NLM R01-LM07061, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash gifts from NCR, Microsoft, and Okawa Foundation.
Address: Integrated Media Systems Center, Los Angeles, CA 90089-2564, U.S.A.

et al. 2001], items are represented as a set of different vectors, each extracted from a physical feature such as color, shape, and texture. Each vector space has its corresponding distance function for comparing the similarity/dissimilarity between a pair of items. By aggregating the outputs of the functions, these systems can retrieve the items similar to the query example from the database.

However, because there is a gap between the physical features of the multimedia data and the semantic meaning as perceived by the user, retrieving items based on pure low-level features is unsatisfactory. In addition to physical features, user-perceived semantic features (such as style and quality) should be considered in the query processes as well. In practice, semantic features are usually used in classification tasks. For example, a user might classify images into a specific style, or he/she may sort images into different classes based on their quality. Hence, the major difficulty of adopting both semantic and physical features in the system is to simultaneously consider two types of data - the similarity values and the classification data.

Building on this premise, we propose a model, named *Yoda*, that conceptualizes content-based querying as the task of classification. Although this model can be adapted for different contents (e.g., see [Shahabi et al. 2001] for application of Yoda in Web recommendation engines), we concentrate on image retrieval as our application domain in this paper. For example, the query "finding images similar to x" can be treated as "finding images in class C_x, where C_x is the class of all images that are similar to x." The images in the database are "softly" classified into overlapping classes, and the classification could differ for different users. 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 hence be uniformly supported by Yoda.

Moreover, by adapting the hypothesis of collaborative filtering, we assume that "if user x believes in expert(s) y, the images that satisfy the query criteria based on y's judgments can be retrieved for x." Based on this assumption, users can specify their confidence degrees to different experts (either classification methods or real *human experts*) and obtain the customized result set based on their preferences. Each image will be ranked according to the weighted aggregation of its classification membership values during the retrieval process. The weights are confidence degrees, which are user dependent, and stored in the corresponding user profile.

Yoda employs a fuzzy-logic based aggregation method for ranking images. Compared to the weighted-average aggregation used in the majority of content-based retrieval systems, our fuzzy aggregation is more efficient. To be specific, the time complexity of the weighted-average aggregation is $C \times I$, while the time complexity of our optimized fuzzy aggregation is[1] $f \times I$, where C is the number of classes, I is the number of items in the database, and f is the number of fuzzy sets (which is a small constant). In addition to the performance benefits, fuzzy aggregation is also less sensitive to noise and can support disjunctive queries.

On the other hand, because Yoda heavily relies on weights for providing accurate

[1] Trivially, both Yoda and the other content-based retrieval systems can benefit from multidimensional index structures such as the hash index structure proposed in [Gionis et al. 1999] to reduce the complexity.

query results, the system performance will decline if weighting values are inaccurate. For example, some users may be uninterested in providing the data or may unintentionally input incorrect information. To solve this problem, Yoda utilizes the users' relevance feedback to improve the profiles automatically using genetic algorithms (GA) [Holland 1975].

Instead of acquiring users' feedback during the learning processes, which easily frustrates users and is commonly adopted in most learning methods [Aggarwal et al. 2000; Doulamis et al. 2000; YRui et al. 1998], user feedback is only needed prior to learning processes with Yoda. Yoda's learning mechanism not only requires less interaction with the users, but also results in faster convergence to the user's preferences as compared to other techniques. Our experimental results indicate a significant increase in the system performance after integrating this learning mechanism.

In summary, three major contributions of this paper are:

—We propose a model that conceptualizes content-based querying as the task of classifying images into classes. By this design, both physical and semantic features can be uniformly incorporated into the query process.

—We design an *optimized* fuzzy aggregation method, which has lower time complexity than the weighted-average aggregation used in most content-based retrieval systems. Moreover, it is less sensitive to noise and can support disjunctive queries.

—We develop a GA-based learning mechanism to improve the accuracy of query results automatically. Our learning mechanism requires less interaction from the user and results in faster convergence to the user's preferences as compared to other learning techniques.

The remainder of this paper is organized as follows. Section 2 reviews the background and the related work. In Section 3, we explain the model, Yoda, and its fuzzy aggregation method. Section 4 discusses the GA-based learning mechanism. Section 5 reports on our experimental results. Section 6 concludes the paper.

2. BACKGROUND AND RELATED WORK

With the aim of building multimedia systems, researchers extend the traditional database management systems (DBMSs) and search engines by incorporating the content-based retrieval methods proposed in various studies [Niblack et al. 1993; Blackburn and DeRoure 1998; Li 2000; Lew 2000]. This method extracts the low-level features, such as color, shape, texture, and pitch contour, from multimedia data as their representatives and uses distance functions to retrieves the similar data by matching the representative of a query example to those in the database. Through these distance functions, the multimedia databases not only can support traditional queries but also "similarity queries", i.e., users can retrieve items that are similar to a given example.

Image retrieval systems are the most studied applications among all multimedia systems because of the popularity of images. The common approach in the majority of image content-based retrieval systems [Lin et al. 2001; Wu et al. 2000] is designing or utilizing one ideal distance function. Based on this function, users can retrieve customized results by performing range queries relative to a selected example or multiple examples. However, since different people may have different perceptions

on the same set of items, using one distance function as a universal solution cannot satisfy all users.

To address this issue, some systems, such as MARS [Ortega et al. 1998] and Garlic [Fagin and Maarek 2000; Fagin and Wimmers 2000], have adapted query expansion models, which can simulate a customized distance function for each user by weighting the relative importance of different features. Items in these systems are represented as different vectors extracting from the physical features, and each vector space has its corresponding distance function. By performing a weighted aggregation of the distance measures in various vector spaces, these systems can also retrieve customized result sets for each user. Some of the systems [Aggarwal et al. 2000; Doulamis et al. 2000; YRui et al. 1998; Bartolini et al. 2001] can even adjust the weights according to users' relevance feedback.

In general, these weight-based multimedia systems (WMS) go through three phases for retrieving query results. In the first phase, WMS obtains the subjective weights from users. The weight $w_{u,o}$ in WMS represents the importance of distance function (feature) o for user u. In other words, $w_{u,o_i} > w_{u,o_j}$ means function o_i has been more emphasized than function o_j. Furthermore, these weighting values are constrained by the following equation.

$$\sum_{j=1}^{n} w_{u,o_j} = 1 \tag{1}$$

In the second phase, an aggregation method is performed. Each item is ranked according to the weighted aggregation of its distances. The majority of WMS, such as MARS and Garlic, employ a simple weighted average function as follows:

DEFINITION 2.1.: Let $o(i,c)$ be the membership values of item i in class c given by function o. The membership of item i in class c for user u is computed by Equation (2).

$$\lambda_{i,c}^{WMS} = \sum_{j=1}^{n} w_{u,o_j} \times o_j(i,c)) \tag{2}$$

∎

In the final phase, learning methods such as Bayesian Inference [Meilhac and Nastar 1999] and statistical analysis learning methods (see [YRui et al. 1998; Aggarwal et al. 2000; Doulamis et al. 2000] for details) are applied for identifying users' emphasis on functions. WMS first obtains users' relevance feedback. Subsequently, with learning methods, WMS evaluates the similarity between user feedback and its weighted aggregation. Finally, each feature is assigned a weight according to its overlap degrees. The more similar the user feedback to the distance values of function o, the larger the weight of $w_{u,o}$.

Unlike WMS that weighted aggregate different distance values of features, some multimedia systems turn feedback acquisition into classification tasks. These systems classify images into relevant and irrelevant classes. Generally, these systems

$M^{image\_similarity}_{QBIC\_color\_function}$

	I1	I2	I3	I4	I5
I1	1	0.37	0.28	0.55	0.12
I2	0.37	1	0.55	0.72	0.45
I3	0.28	0.55	1	0.69	0.84
I4	0.55	0.72	0.69	1	0.54
I5	0.12	0.45	0.84	0.54	1

$M^{image\_similarity}_{Garlic\_shape\_function}$

	I1	I2	I3	I4	I5
I1	1	0.76	0.53	0.27	0.14
I2	0.76	1	0.3	0.13	0.5
I3	0.53	0.3	1	0.41	0.17
I4	0.26	0.13	0.41	1	0.76
I5	0.14	0.5	0.17	0.76	1

$M^{image\_similarity}_{MARS\_texture\_function}$

	I1	I2	I3	I4	I5
I1	1	0.25	0.07	0.73	0.20
I2	0.12	1	0.8	0.72	0.1
I3	0.07	0.8	1	0.53	0.57
I4	0.73	0.72	0.53	1	0.18
I5	0.20	0.1	0.57	0.18	1

Fig. 1. Example metric space of traditional image retrieval systems

first employ an ideal distance function to extract image vector representatives. Afterward, they separate relevant images from irrelevant ones with a hyperplane. Finally, in order to refine the hyperplane, these systems utilize learning techniques, such as neural network [Wood et al. 1998] and Support Vector Machine (SVM) [Tong and Chang 2001; Zhou and Huang 2001], to improve the classification results.

Although these systems can enhance the accuracy of query results during the interaction with users, they cannot memorize the learning results across multiple query sessions. Thus these systems require restarting the learning processes repeatedly for every new query. This drawback can be solved by FeedbackBypass [Bartolini et al. 2001]. Basically, FeedbackBypass stores the learning results, which are sets of weights, using a wavelet-based data structure. At the retrieval phase, the system initiates the weights with the favorable set of weights in the system based upon the input query example. On the other hand, because the user information is not considered in the searching processes, each input query is assigned its matching weights regardless of the user submitting the query. As a result, the accuracy of initial weights in FeedbackBypass declines as the number of users increases. In other words, the result is not customized for each user.

3. ADAPTIVE SOFT CLASSIFICATION DESIGN OF YODA

In order to generate customized query results for users, Yoda needs to go through two main phases: customized ranking of the objects and adjusting user settings based on relevance feedback. This section focuses on the first phase, and Section 4 concentrates on the second phase. We first introduce the basic model of Yoda. Subsequently, the fuzzy-logic based aggregation technique is described in Section 3.2. Finally, a comparison between the general weighting systems and Yoda is given in Section 3.3.

3.1 A Soft Classification Model

The main objective of content-based retrieval systems is retrieving items similar to certain specified examples. The similarity/dissimilarity between items is measured by an ideal distance function. Some content-based retrieval systems, such as FALCON [Wu et al. 2000], perform these retrieval operations using one single distance function.

EXAMPLE 3.1.: Assume matrix $M_x^{image\_similarity}$ in Figure 1 represents the similarity values, which are calculated by distance function x, between images in the

system. Consider the numbers in the matrix cells as corresponding distances between images in corresponding rows and columns. For example, distance between I_1 and I_2 is 0.37, if it is measured using QBIC color function while it is 0.76 when using Garlic shape function. FALCON only retrieves the similarity values from matrix $M^{image\_similarity}_{QBIC\_color\_function}$ during the query processes. ∎

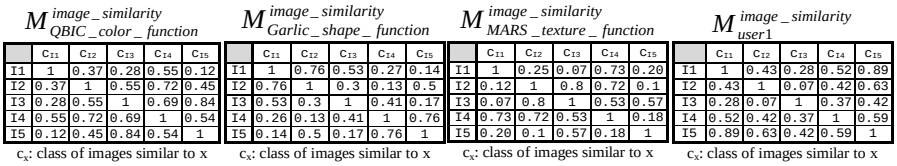
Because of the subjectivity of human perceptions, however, using one single distance function as a universal solution cannot satisfy all users. To solve this problem, other content-based retrieval systems, such as MARS [Ortega et al. 1998; YRui et al. 1998] and Garlic [Fagin and Maarek 2000; Fagin and Wimmers 2000], employ different distance functions, where each function corresponds to a physical feature. These systems simulate the ideal distance function through a weighted combination of the distance functions based on subjective weightings for each user. Hence, the query results generated by the simulated function are closer to the true expectations of each user.

EXAMPLE 3.2.: Consider the data in Example 3.1. MARS and Garlic can aggregate data from multiple matrices in Figure 1, i.e., $M^{image\_similarity}_{QBIC\_color\_function}$, $M^{image\_similarity}_{Garlic\_shape\_function}$, and $M^{image\_similarity}_{MARS\_texture\_function}$. If a user requests the images which are similar to $I5$ in MARS, the final similarity value of each image is derived from the weighted average of its corresponding values in the last column of each respective matrix.∎

Due to the fact that there is a gap between the physical features of digital images and the semantic meaning as perceived by a user, simulating the ideal distance function from pure physical-feature comparison functions remains unsatisfactory. To solve this problem, semantic features, such as style and quality, should be considered in the query processes as well. In the real world, semantic features are usually employed in classification tasks. For example, a user might ask for a class of images belonging to a specific style (such as classical style or modern style), or he/she may look for a class of images with high visual quality. Hence, the major difficulty of adopting both semantic features and physical features in the system is to uniformly consider two types of data – similarity values and classification data. Building on this premise, we proposed a model [Shahabi and Chen 2000b] that conceptualizes content-based querying as the task of soft classification, which is performed for both physical and semantic features uniformly.

With Yoda, we model the classification data as **"soft-relation matrices"**, which could be acquired from different kinds of **experts** such as feature-comparison functions, algorithms, human experts, or clusters of user behaviors.

DEFINITION 3.1.: Each feature $\mathcal{F}$ partitions the set of all images into k related classes, $c^{\mathcal{F}}_{x1}, c^{\mathcal{F}}_{x2}, \ldots, c^{\mathcal{F}}_{xk}$, where the classes might overlap and/or be partial. Let the *class-set* be the set of all classes in the system, i.e., $\mathcal{C} = \{c | \text{“}c'' \textit{ is a class in the system}\}$. The membership of an image to a class is determined by an expert in a soft-relation. Moreover, different experts can classify the same image into different classes. ∎



	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
I1	1	0.37	0.28	0.55	0.12
I2	0.37	1	0.55	0.72	0.45
I3	0.28	0.55	1	0.69	0.84
I4	0.55	0.72	0.69	1	0.54
I5	0.12	0.45	0.84	0.54	1

	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
I1	1	0.76	0.53	0.27	0.14
I2	0.76	1	0.3	0.13	0.5
I3	0.53	0.3	1	0.41	0.17
I4	0.26	0.13	0.41	1	0.76
I5	0.14	0.5	0.17	0.76	1

	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
I1	1	0.25	0.07	0.73	0.20
I2	0.12	1	0.8	0.72	0.1
I3	0.07	0.8	1	0.53	0.57
I4	0.73	0.72	0.53	1	0.18
I5	0.20	0.1	0.57	0.18	1

	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
I1	1	0.43	0.28	0.52	0.89
I2	0.43	1	0.07	0.42	0.63
I3	0.28	0.07	1	0.37	0.42
I4	0.52	0.42	0.37	1	0.59
I5	0.89	0.63	0.42	0.59	1

Fig. 2. Example classification space of Yoda

DEFINITION 3.2.: Let $S_u^{\mathcal{F}}$ be a soft-relation from the item set I to the class-set $\mathcal{C}$ by expert u, where $S_u^{\mathcal{F}} : I \times \mathcal{C} \rightarrow \Re$ and $S_u^{\mathcal{F}}(i,c) \in [0,1]$ (0 represents that i does not belong to class c, and 1 represents that i completely belongs to class c). Choose orderings of I and C. All soft-relation matrices are with respect to these orderings. A 2-dimensional matrix $\mathcal{M}_u^{\mathcal{C}}$ of soft-relation $S_u^{\mathcal{F}}$ can be used to represent this mapping. ∎

REMARK 3.1.: The system contains various soft-relation matrices which have the same underlying sets. For example, $\mathcal{M}_1^{\mathcal{F}}, \mathcal{M}_2^{\mathcal{F}}, \ldots, \mathcal{M}_k^{\mathcal{F}}$ are all soft-relation classification matrices created by different experts with the same underlying sets I and $\mathcal{C}$.∎

By using our model, we not only can represent the classification data, but also can capture conventional content-based retrieval operations such as finding all images similar to a query image in color, shape or texture. Moreover, this model can also utilize the presence of multiple algorithms for the same comparison function as well as opinions from real human experts. To illustrate, consider the following examples.

EXAMPLE 3.3.: By these definitions, the similarity data in Figure 1 can be represented as classification data in Figure 2. Note that the rows are no longer images but classes. Hence, C_x is defined as class of images similar to x. Subsequently, the membership value of image I to C_x is the same as similarity of i to x given the corresponding distance function. Compared to FALCON, MARS, and Garlic, Yoda has a more general form. It can incorporate similarity values and classification data during the query processes. In this example, Yoda can also employ a user's soft classification data, i.e., $M^{image\_similarity}_{user1}$. When a user asks for the images similar to $I5$, the final similarity value of each image in the database is derived from the weighted aggregation of the values in the last column of each respective matrix for the image.∎

After having these classification memberships, users can obtain the customized query results, in which each item is ranked based on the weighted aggregation of the classification memberships. Since these weights are user dependent, Yoda captures them into user profiles. A user profile is composed of two parts: *user confidence data* and *user fuzzy cut value*. A user fuzzy cut value represents a perceptual threshold, which indicates the minimum membership value in the result set. User confidence data are employed as weights in aggregation processes, and we provide the formal definition of user confidence data as follows:

DEFINITION 3.3.: Let $\mathcal{E}$ denote a set of experts in the database, let F be the set of fuzzy terms, and let U represent the set of users who assign reference confidence values to the experts. π is a confidence value for an user u to a expert e; $\pi : u \in U, e \in \mathcal{E} \rightarrow F$. Note that the value of $\pi(u, e)$ is a form of human judgment and is represented as a fuzzy term.

In practice, asking people to describe their perceptions with precise values is almost impossible. Moreover, different people have different interpretations of words. That is, the information describing personalities, physical features, preferences and personal evaluation is imprecise. In order to handle this uncertainty during the query processes, fuzzy logic (FL) [Zadeh 1978] is adopted by our system. The concept of FL was first introduced by Zadeh [Zadeh 1978] to problems for which precise formulation is not possible. The original FL has the weakness that uncertainty cannot be considered during the computation. Therefore, Karnik and Mendel advocated type 2 FL [Karnik and Mendel 1998; Karnik and Mendel 2000] for overcoming this disadvantage. However, for the sake of simplicity, we only consider the original FL in this paper.

With the help of FL, Yoda 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's perceptions according to the user's fuzzy cut value.

3.2 Fuzzy Aggregation Function

According to the definitions in Section 3.1, our problem objective can be defined as follows:

Problem:	**Retrieve result set of class c for user u**
Given	Item-set I
	Soft-relation matrices $\mathcal{M}$
	Confidence values of user u to experts
	The fuzzy cut value of user u
Rank	Items based on their corresponding memberships λ

The membership $\lambda_{i,c}$ of item i belonging to class c is aggregated from the relationships in soft-relation matrices based on the confidence values of user u. This membership $\lambda_{d,c}$ can be computed using various aggregation functions. However, the aggregation functions with a triangular norm [Fagin 1996] are preferred with our system. A triangular norm aggregation function g satisfies the following properties:

$$\begin{aligned} \text{Monotonicity:} \quad & g(x,y) \leq g(x',y') \quad \text{if } x \leq x' \text{ and } y \leq y' \\ \text{Commuatativity:} \quad & g(x,y) = g(y,x) \\ \text{Associativity:} \quad & g(g(x,y),z) \quad = g(x,g(y,z)) \end{aligned}$$

With these properties, the query optimizer can replace the original query with a logically equivalent one and still obtain exactly the same result. The aggregation function we proposed [Shahabi and Chen 2000a] is:

DEFINITION 3.4.: First, experts are grouped based on their reference confidence values assigned by user u.

$$\mathcal{E}_f = \{e \mid e \in \mathcal{E}, \pi(u,e) = f\} \tag{3}$$

Then, the membership $\lambda_{i,c}$ of item i belonging to class c is computed as:

$$\begin{aligned} \theta_{i,c,f} &= f \times \max_{e \in \mathcal{E}_f}\{S_u^{\mathcal{F}}(i,c)\} \\ \lambda_{i,c} &= \max_{f \in F}\{\theta_{i,c,f}\} \end{aligned} \tag{4}$$

Basically, this aggregation function partitions the preference values into $\|F\|$ different subgroups according to the confidence values. Subsequently, the system maintains a list of maximum relationship values for all subgroups. Next, the system computes the memberships of all items by iterating through all subgroups. Finally, by using the fuzzy cut as user's perceptual standard, the system removes items whose final memberships are below the fuzzy cuts from result sets. ∎

The proposed aggregation function not only has the advantage of being incorporable with query optimizers, but also reduces the complexity of query evaluation. A naive weighted aggregation function may use the user confidence data as the weight factors when retrieving the items. Hence, its complexity becomes a function of the product of the number of experts to the number of images: $O(\|\mathcal{E}\| \times \|I\|)$ where $\|I\|$ is the number of items and $\|\mathcal{E}\|$ is the number of experts in the system. In comparison, the total computation complexity of our aggregation function is only $O(\|F\| \times \|I\|) = O(\|I\|)$, where $\|F\|$ is the number of fuzzy terms. This is because we compute the final membership by only iterating through possible values of the fuzzy term f.

The time complexity of the aggregation function can be further reduced, if the objective is a k nearest neighbors search. In [Fagin 1996], Fagin proposed an optimized algorithm, the $\mathcal{A}_0$ algorithm, to retrieve k best items from a collection of subsets of items with a time complexity proportional to k rather than the total number of items. Here, since our aggregation function is in triangular norm form[2], the $\mathcal{A}_0$ algorithm can be incorporated into Yoda by taking the subgroups of items (as described above) as the subsets. Applying the $\mathcal{A}_0$ algorithm to generate the result set, we reduce the time complexity to $O(\|F\| \times k) = O(k)$, where $k \ll \|I\|$.

3.3 Analysis

Although both weight-based multimedia systems (WMS) described in Section 2 and Yoda aim to incorporate users' perceptions into query procedures, their approaches are different. With WMS, the system focuses on obtaining customized results from a few feature-comparison functions such as the texture similarity function and the shape similarity function. In comparison, besides various feature-comparison functions, Yoda also consults the real opinions of a large number of human users. Thus, WMS and Yoda employ different methods for integrating users' perceptions into

[2]Triangular norm form is the requirement for employing the $\mathcal{A}_0$ algorithm.

their systems. Users' perceptions are represented by precise weights (emphasis degrees) in WMS, but the user perceptions are captured by fuzzy confidence values in Yoda. Compared to the design of WMS, Yoda has several advantages:

(1) **Insensitivity to Inaccurate Inputs** Yoda is based on the spirit of fuzzy logic, which models the vagueness of the real world well. Because human perceptions are imprecise and inaccurate, it is easier for users to describe their feelings using fuzzy terms rather than picking precise numbers. Moreover, because of using MAX operator in Equation (4), Yoda is only affected by the noise in one of the confidence values. Conversely, WMS accumulate the noise from each emphasis degree. Hence, the effect of noise is enlarged in WMS. As a result, Yoda is less sensitive to inaccurate inputs than WMS. We illustrate this observation with the following example.

EXAMPLE 3.4.**:** Assume the emphasis degrees $w_{u,o}$ in WMS and the user confidence values $\pi(u,e)$ in Yoda have the same deviation d_u. Furthermore, suppose all membership values of item i to class c are equivalent to $o(i,c)$ in both WMS and Yoda. Let w^*_{u,o_j} denote the accurate emphasis degree of user u on feature-comparison function o_j, and let $\pi^*(u,e)$ represent the accurate confidence value of user u to expert e . The inaccurate w_{u,o_j} and $\pi(u,e)$ can be represented as:

$$\begin{aligned} w_{u,o_j} &= w^*_{u,o_j} + d_u \\ \pi(u,e) &= \pi^*(u,e) + d_u \end{aligned} \tag{5}$$

Let $\bar{\lambda}^{WMS}_{i,c}$ be the accurate final membership in WMS, and let $\bar{\lambda}_{i,c}$ be the accurate final membership in Yoda. According to Equation (2) and Equation (4), the predicted memberships $\lambda^{WMS}_{i,c}$ in WMS and $\lambda_{i,c}$ in Yoda are:

$$\begin{aligned} \lambda^{WMS}_{i,c} &= \sum_{j=1}^{n}[(w^*_{u,o_j} + d_u) \times o_j(i,c)] = \sum_{j=1}^{n}(w^*_{u,o_j} \times o_j(i,c)) + \sum_{j=1}^{n}(d_u \times o_j(i,c)) \\ &= \bar{\lambda}^{WMS}_{i,c} + \sum_{j=1}^{n} d_u \times o_j(i,c) \end{aligned} \tag{6}$$

$$\begin{aligned} \lambda_{i,c} &= \max_{f \in F}\{(\pi^*(u,M_s) + d_u) \times \max_{\mathcal{M}_s \in \mathcal{M}_f}\{S^{\mathcal{F}}_u(i,c)\}\} \\ &= \max_{f \in F}\{(\pi^*(u,M_s) + d_u) \times \max_{\mathcal{M}_s \in \mathcal{M}_f}\{S^{\mathcal{F}}_u(i,c)\}\} \\ &= \bar{\lambda}_{i,c} + (d_u \times \max_{\mathcal{M}_s \in \mathcal{M}_f}\{S^{\mathcal{F}}_u(i,c)\}) \end{aligned} \tag{7}$$

Equation (6) and (7) demonstrate that the impact of noise on Yoda is independent of the number of experts. Conversely, the impact of noise on WMS increases as the number of experts grows. In other words, this example illustrates that Yoda is less sensitive to inaccurate inputs. ∎

(2) **Supporting Disjunctive Queries** The aggregation function of Yoda (similar to FALCON [Wu et al. 2000]) can manage disjunctive queries while the aggregation function of WMS cannot. For instance, assuming that the system

knows all the information about users and user u is interested in the items recommended by feature-comparison function A *or* B. The aggregation function of WMS can only generate the list of items recommended by both A *and* B while our aggregation method can retrieve the expected list. We illustrate this idea by the following example.

EXAMPLE 3.5.**:** Suppose the confidence values for all experts are identical. Assume the expert $M1$ gives the images in red color ($RGB = \#FF0000$) the higher similarity scores, the expert $M2$ assigns the images in green color ($RGB = \#00FF00$) the higher similarity values, and the expert $M3$ award the images in gray-yellow color ($RGB = \#808000$) the higher similarity degrees. The input query is "searching all images similar to $I2$ by expert $M1$ and $M2$". The WMS will returns the images in gray-yellow color (i.e., the average of red and green) that are more similar to expert $M3$'s results. However, this result set is neither close to expert $M1$'s expected result nor close to expert $M2$'s result. In contrast, Yoda can retrieve both expert $M1$ and $M2$'s result sets (i.e., green color and red color images) and thus the result set is closer to the user's expectation. ∎

(3) **Better Scalability** The computational complexity of the aggregation method in Yoda is independent of the number of experts and feature-comparison functions. The cost of computing the final membership for each item in WMS is $O(Y)$, where Y is the number of feature-comparison functions in the system. This means the aggregation function of WMS cannot scale up. In comparison, the complexity of computing the overall membership for each image in Yoda is $O(f) = O(1)$.

(4) **Fewer Modifications When Updating** The sum of confidence values in Yoda is not constrained by Equation 1. Hence, the user profiles require fewer modifications during the update processes. For example, when user u wants to modify the confidence value $\pi_{u,y}$ for feature-comparison function y, Yoda only needs to modify $\pi_{u,y}$. In contrast, WMS needs to recalculate the weights for all experts and modify all weights related to user u in the database.

(5) **Incorporating Richer Users Information** In addition to the relative emphasis placed on the experts, Yoda could further trace a user's absolute degree of confidence to the experts. For example, when a user puts equal emphasis on all the feature-comparison functions in the system, he may be equally unsatisfied with all of their classification data or fully satisfied with all of them. While Yoda could differentiate these two cases and provide different result sets, WMS would only generate identical result sets for both cases.

(6) **More Precision in Results** Yoda maintains records of user standards (user fuzzy cut values) as their perceptual thresholds and filters out the items whose final memberships are below the thresholds. The resulting information would be closer to user's expected result set. For example, user o_1 and user o_2 both query for "red" images. However, the definition of "red" in the minds of user o_1 and o_2 are "pure red" and "reddish", respectively. Because this difference is

reflected by a higher fuzzy cut value for o_1 than for o_2, Yoda can thus provide more accurate result sets for each of these two users while WMS cannot.

4. THE ADAPTIVE MECHANISM

In order to provide accurate query results, Yoda heavily relies on user profiles. In Section 3.2, we assumed that users would supply accurate data for creating the user profiles. However, in practice, obtaining user profiles has been challenging. For example, users may be too busy to provide the data or they might un-intentionally input the incorrect information. A more appropriate approach should offer a learning mechanism to correct user errors. Building on this premise, we utilize the users' relevance feedback thus improving the profiles automatically using a genetic algorithm (GA).

This learning mechanism [Chen and Shahabi 2001] is an automatic and off-line process. It employs GA for improving the user profile by decoding the best chromosome to replace existing user profiles in the database after its evolution. User involvement is only needed for providing the relevance feedback as the goal of GA prior to the beginning of evolution. Note: the learning mechanism is only triggered by the user feedback and not needed during query processing.

In Section 4.1, we briefly review the background of GA. Subsequently, the proposed design of the learning mechanism is described in Section 4.2. Finally, a comparison analysis is provided in Section 4.3.

4.1 Background on Genetic Algorithms

GA [Holland 1975] is an iterative search technique based on the spirit of natural evolution. By emulating biological selection and reproduction, GA can efficiently search through the solution space of complex problems. Basically, GA operates on a population of candidate solutions called *chromosomes*. A chromosome, which is composed of numerous genes, represents an encoding of the problem and associates it with a fitness value evaluated by the fitness function. This fitness value determines the goodness and the survival ability of the chromosome.

Generally, GA starts by initializing the population and evaluating its corresponding fitness values. Before it terminates, GA produces newer generations iteratively. At each generation, a portion of the chromosomes is selected according to the survival ability for reproducing offspring. The offspring are generated through crossover and mutation processes and are used for replacing some chromosomes in the population with a probability consistent with their fitness values. In other words, with the help of the fitness function to point out the correct direction, GA could construct better and better chromosomes from the best partial genes of past samplings. Please reference [Goldberg 1989] for mathematical foundations.

In summary, GA is composed of a fitness function, a population of chromosomes and three operators - selection, crossover and mutation. The parameter settings of the operators can be chosen depending on the applications or remain unchanged even when the applications are varied. However, the fitness function and the coding method are required to be specially designed for each problem.

4.2 Proposed Design

First, we explicate the coding design for GA in our learning mechanism. The chromosomes represent possible user profiles of a specific user, and each gene corresponds to a record in the user profiles. Two types of records are involved in the genes. One is user confidence information with k records, where k is the number of experts in the system. The value of the ith gene is an integer in $[0, L-1]$, where L is the number of fuzzy terms used in the system, and denotes the user's confidence level to expert i. The other is a user fuzzy cut value which is associated with the $(k+1)$th gene. The value of fuzzy cut is $(t+1)/L$, where $t \in [0, L-1]$ is the value of this gene.

EXAMPLE 4.1.: Suppose that there are 50 experts and 8 different fuzzy terms in the system, there will be 51 genes per chromosome where the first 50 genes represent the corresponding confidence values to the experts, and the last gene represents the value of the user fuzzy cut. Additionally, after decoding, the value of 0 in gene i indicates that the confidence level to expert i is "none" and the value of 6 in gene 51 indicates that the value of fuzzy cut is $(6+1)/8 = 0.875$. Likewise, after encoding, "full" confidence level to expert i is represented by a number 7 in gene i and the 0.75 fuzzy cut is denoted by a number 5 in gene 51. ∎

This coding method can guarantee a one-to-one mapping of profiles to chromosomes. That is, a chromosome will be decoded to one and only one legal user profile, and a user profile will be encoded to one and only one chromosome. Consequently, the solution space will be equal to the searching space in GA. This implies that our coding method is effective.

Next, we describe our GA fitness function, which heavily utilizes the users' relevance feedback. Users can modify the ranks of the images by adjusting the corresponding membership values and marking the irrelevant items with values in $(-1, 0)$, where -1 represents "highly irrelevant" and 0 represents "slightly irrelevant". The fitness function first decodes the chromosome into a user profile. Then, it obtains the refined result set according to this profile using Equation (4). Unlike the expected result set, which only consists of relevant items, the refined result set also contains irrelevant items, whose membership values λ^* are:

$$\lambda^* = \lambda - (t+1)/L \tag{8}$$

In other words, this process needs to interact with the database for obtaining the classification data. Finally, it generates the fitness value by measuring the similarity between the query result and the users' relevance feedback. The similarity values are computed by Equation (12) that is based on three measurements. Equation (9) evaluates the similarity on ranking, Equation (10) measures the precision values of query results, Equation (11) calculates the recall degrees of refined query results based on users' relevance feedback. The fitness equation is defined as follows.

DEFINITION 4.1.: Let Q represent the refined query set and B denotes the relevance feedback. Let $|BQ_r|$ be the number of items in the intersection of relevant

items in B and Q, $|B_r|$ be the number of relevant items in B, and $|Q_r|$ be the number of relevant items in Q. Further, let α_c represent the emphasis on similarity value Cos $\theta(Q,B)$, α_p represent the emphasis on precision value $P^*(B,Q)$, and α_r represent the emphasis on precision value $R^*(B,Q)$.

$$\text{Similarity (Cos } \theta(Q,B)) = \frac{\sum_{i=1}^{n} Q_i \times B_i}{\sqrt{\sum_{i=n}^{n} Q_i^2 \times \sum_{i=n}^{n} B_i^2}} \tag{9}$$

$$\text{Precision } (P^*(B,Q)) = \frac{|BQ_r|}{|Q_r|} \tag{10}$$

$$R^*(B,Q) = \frac{|BQ_r|}{|B_r|} \tag{11}$$

$$\text{Fitness}(Q,B) = \alpha_c \times \text{Cos } \theta(Q,B) + \alpha_p \times P^*(B,Q) + \alpha_r \times R^*(B,Q) \tag{12}$$

Note that since selecting the values of α are straightforward, we empirically compare results for various values of α. Our experiments show that the setting of $\alpha_c = 0.27,\ \alpha_p = 0.7,\ \alpha_r = 0.03$ has the better results. Therefore, we utilize the settings throughout all experiments. However, various applications might require different settings.

Once a user offers his/her user feedback to trigger the learning process, the learning mechanism first encodes the corresponding user profile to a chromosome and randomly generates other chromosomes as the initial population. Subsequently, GA iteratively discover better user profiles until it achieves the terminal condition such as the fitness value of one chromosome being 1 or the generation number being 50. In the end, the learning mechanism decodes the best chromosome to a user profile for replacing the current user profile in the database.

4.3 Analysis

Comparing to other learning mechanisms, such as the statistic learning method in MARS [YRui et al. 1998], the wavelet-based searching technique in FeedbackBypass [Bartolini et al. 2001], support vector machine method in SVM$_{Active}$ [Tong and Chang 2001], Bayesian Inference in Surfimage [Meilhac and Nastar 1999], and the GA-based learning mechanism in Amalthea [Sheth 1993; Moukas 1996], our learning mechanism has several advantages:

First, because of employing user profiles, Yoda can converge faster. For example, many learning mechanisms in content-based retrieval systems, such as MARS, SVM$_{Active}$, and Surfimage, cannot memorize the learning results, which are customized query parameters, across multiple query sessions. Thus these systems require restarting the learning process from the default parameter values for every new query. To address this drawback, FeedbackBypass maintains all query parameters over time and searches the best parameter settings for the corresponding query in the entire user population. As a result, FeedbackBypass can converge faster than MARS, SVM$_{Active}$, and Surfimage. However, since FeedbackBypass ignores the fact that different users have different perceptions, its convergence speed decreases as the number of users increases. In contrast, our learning mechanism resolves these

two drawbacks by keeping the parameter settings for each user in his/her profile. Because each query process is based on a particular up-to-date user profile, users can obtain customized query results across multiple query sessions. Moreover, each evolution explores the best user profile for each user. Consequently, because the learning objective is user specific, the convergence speed of our learning mechanism is independent of the number of users.

Subsequently, our GA-based learning mechanism has less user involvement during the learning process. In MARS, SVM$_{Active}$, and Surfimage, users need to provide relevance feedback whenever the system provides a refined query result. Similarly, the learning mechanism of Amalthea needs to acquire users' feedback in every generation. Alternatively, although FeedbackBypass can reduce the unnecessary interactions with users at the beginning of learning processes, the user relevance feedback are still required during the later phases. Because of the need for ongoing user interactions, this approach frustrates users. On the contrary, in our design, user involvement is only needed for providing the feedback prior to the beginning of the learning process.

Finally, because our GA-based learning method takes negative feedback into consideration during evolution, the learning mechanism of Yoda can improve the accuracy of query results even when the majority of feedback (or all feedback) is negative while the statistic learning method of MARS cannot. For example, if the initially predicted result set is totally different from the perfect result set of user o, he/she would give negative ratings to all elements in the initial list. However, the statistic learning method in MARS cannot improve the weights because the negative sum is treated as zero in MARS (reference [YRui et al. 1998] for more details). In contrast, because the feedback of each item is compared individually in Equation 9 (no matter whether it is positive or negative), Yoda still can improve user profiles under this situation. Although the SVM$_{Active}$ also can improve the query results while the majority of feedback is negative, the SVM algorithm has one additional limitation that it requires at least one relevant and one irrelevant images to operate. Therefore, the query results of first few rounds (before the system locates the nearly correct hyperplane) will be unstable and less accurate. This weakness might lead users to stop interacting with the system during the initial rounds.

5. PERFORMANCE EVALUATION

We conducted two different experiments for comparing the performance of Yoda with MARS [YRui et al. 1998]. MARS is one of the very few content-based retrieval systems that can improve the query parameters for each user by acquiring the user's relevance feedback. Moreover, unlike other classifier systems such as SVM$_{Active}$ [Tong and Chang 2001], which employs only one distance function, Yoda and MARS both can handle numerous feature-comparison functions. Therefore, the feature-comparison algorithms proposed in traditional image retrieval works can be uniformly incorporated into Yoda and MARS. Moreover, since the learning approaches in Yoda and MARS are very different and the complexity of Yoda is already demonstrated in Section 3.2, we emphasize the accuracy comparison between Yoda and MARS in our experiments. The first experiment focuses on the comparison of aggregation methods, and the other experiment compares the accuracy of

Parameter	Definition
I	Number of items
F	Number of fuzzy terms
E	Number of experts (functions)
O	Number of known experts ($E \geq O$)
P	Percentage of interesting items within the item set
U	Number of users
N	Level of Noise

Table 1. Benchmarking Parameters

learning mechanisms by employing end-to-end simulations.

Both Yoda and MARS are implemented in C and run on a Pentium II 233MHZ processor with Microsoft Window 2000. Moreover, we developed a GA for Yoda using SUGAL [Hunter 1995] for its wide range of operators and data types. This section is structured as follows. In Section 5.1, we describe our benchmarking method. Section 5.2 discusses the details of our experimental results.

5.1 Experimental Methodology

Theoretically, classification data collected from human experts are the best source of benchmarks. However, since access to human experts is limited, it is very difficult to collect a large amount of classification data from these experts. Moreover, we have observed that the classification behaviors of non-professional users are usually inconsistent, so it is inadvisable to use volunteers to substitute for the experts. Instead, we conduct all the experiments by utilizing synthetic data generated from the benchmarking method, which can ensure the consistency of data, and therefore the experimental results can be fairly compared.

In order to populate data for evaluation purposes, we propose a parametric algorithm to simulate various benchmarks (see Table 5.1). The benchmarking method maintains two 3-dimensional matrices, $\mathring{A}_1$ and $\mathring{A}_2$, for holding the perfect user classification behaviors as standard answers, and two 3-dimensional matrices, $\dot{\mathcal{U}}_1$ and $\dot{\mathcal{U}}_2$, for providing user feedback. This is performed as follows. First, the algorithm randomly generates E experts and populates the classification data to each soft-relation matrix $\bar{M}_e$ (where $e \in E$). The cell $\bar{m}(i,j) \in \bar{M}_e$ represents the membership value of item i to class j by expert e.

Next, the system randomly generates a list of confidence values $\ddot{\pi}$ and a fuzzy cut value $\ddot{c}$ for each active user. Each confidence value is represented by a fuzzy term, which is an integer in the range of $[0, 7]$, where 0 represents "no confidence at all" and 7 represents "full confidence". Finally, the system populates the classification memberships $\ddot{a}_1$ to $\mathring{A}_1$ by aggregating $\ddot{\pi}$ and $\bar{M}$ based on the disjunctive query approach (i.e., using Equation (4)) and populates the classification memberships $\ddot{a}_2$ to $\mathring{A}_2$ based on weighted average approach(i.e., using Equation (2)). In other words, the user classification behaviors $\mathring{A}_1$ and $\mathring{A}_2$ are more favorable to Yoda and MARS, respectively.

Subsequently, using fuzzy cut $\ddot{c}$ as each user's standard, the user feedback $\ddot{u} \in \dot{\mathcal{U}}_1 \vee \dot{\mathcal{U}}_2$ are generated according to Equation (13), and the item memberships that are below $\ddot{c}$ in $\mathring{A}_1$ and $\mathring{A}_2$ are reset to 0.

$$\ddot{u} = \ddot{a} - \ddot{c} \tag{13}$$

To simulate imperfect user profiles and user feedback, the systems tune the perfect knowledge, such as the confidence values $\ddot{\pi}$ and user classification behaviors retrieved either from $\mathring{A}_1$ or $\mathring{A}_2$, by a noisy process according to the noise level. Noise level 0 represents perfect feedback and noise level 10 represents complete chaos. Finally, in order to simulate the learning processes, the systems replace all user profiles by a set of default values. We use Yoda and MARS to generate predicted query result I_u for each user u, and evaluate the accuracy of these systems before and after learning processes. Note that the learning mechanisms of two systems can only obtain the classification data of "known" experts. Therefore, the systems will only learn the confidence values (or weights) of these known experts.

5.2 Experimental Results

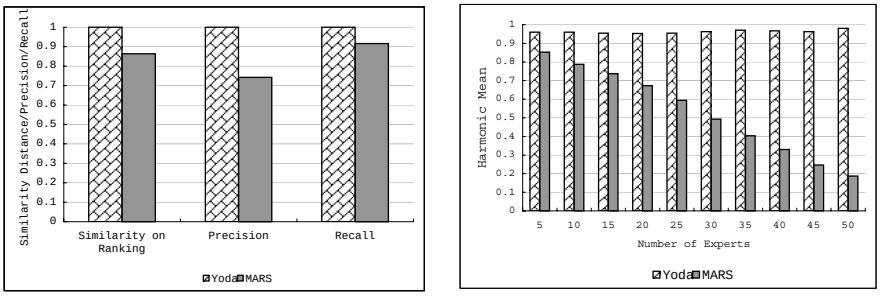
5.2.1 *Aggregation Comparison.* We conducted several sets of experiments to compare the aggregation method of Yoda with that of MARS. In these experiments, we observed a significant margin of improvement over MARS in matching the users' expectations for various settings. It also shows that the performance of Yoda is independent of the number of users. However, we only stress the accuracy comparison on the impacts of noise and the number of experts. For an accurate comparison, each aggregation method employs the corresponding user classification behaviors as standard answers, i.e. the user classification behaviors for Yoda and MARS are $\mathring{A}_1$ and $\mathring{A}_2$, respectively. The results shown for each set of experiments are averaged over many runs, where each run is executed with different seeds for the benchmarking procedure. The common benchmark settings of the following figures, i.e., I, F, U, P, are fixed at 5000, 8, 50, 5%, respectively.

Figure 3.a illustrates the performances of two aggregation methods when both systems are initiated with perfect user profiles. The benchmark settings of this experiment, N, E, O, are set at 0, 10, and 10, respectively. As can be seen, Yoda can reproduce perfect classification data for users while MARS cannot. This is because Yoda keeps records of user standards and employs them to filter out unexpected items. Hence, Yoda can achieve more precision than MARS while both of them originate with the perfect knowledge of user perceptions.

In Figure 3.b and Figure 4, we compare the impact on noise as the number of expert increases. The benchmark settings of Figure 4, N, E, O, are fixed at 1, 20, 20, respectively. The X-axis of Figure 3.b depicts the number of experts, and the X-axis of Figure 4 illustrates the level of noise. The Y-axes of Figure 3.b and Figure 4.a represent the harmonic mean computed by Equation (14). Let H_o be the harmonic mean of results that generated from the perfect user profiles and H_i represent the harmonic mean of results that are generated from defective user profiles. The Y-axis of Figure 4.b indicates the decline rate computed by Equation (15) .

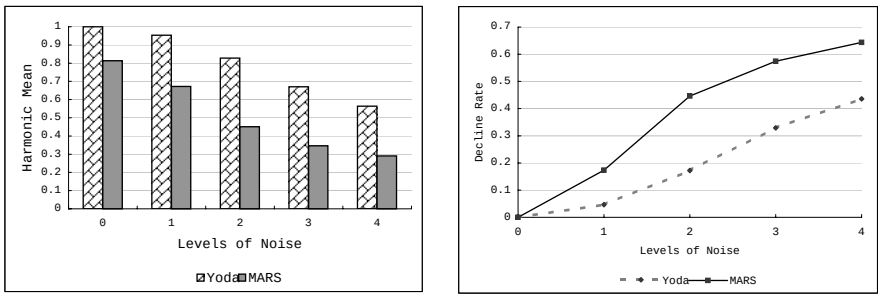
$$\text{Harmonic Mean}(_{Similarity,Precision,Recall}) = \frac{3}{\frac{1}{Similarity} + \frac{1}{Precision} + \frac{1}{Recall}} \tag{14}$$

$$\text{Decline Rate}(H_o, H_i) = \frac{(H_o - H_i)}{H_0} \tag{15}$$



a. With perfect profiles ($N = 0$) b. With imperfect profiles ($N = 1$)

Fig. 3. Accuracy comparison



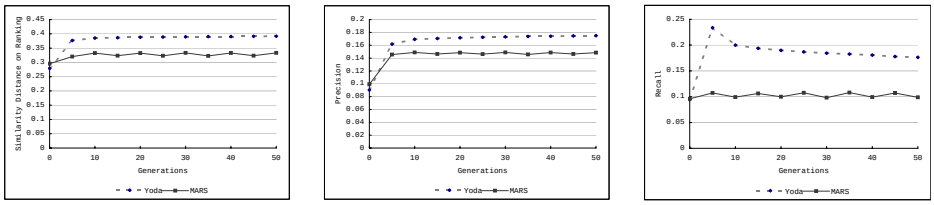
a. Harmonic mean b. Decline rate

Fig. 4. Impact on noise

As revealed by Figure 3.b, the accuracy of MARS declines as the number of experts grows. Figure 4 further shows not only the accuracy of Yoda is superior to that of MARS across the different levels of noise, but the decline rate of Yoda is less than that of MARS. These figures support the analysis in Section 3.3 and show the aggregation methods used in Yoda is less insensitive to noise.

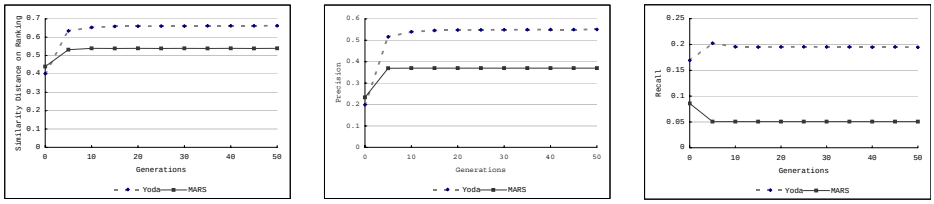
5.2.2 *Comparison on Learning Abilities.* We conducted several sets of experiments to the compare the learning mechanisms. Across these experiments, we observed a significant margin of improvement over MARS in matching the user expectations in various settings. It has also shown that the performance improvements of learning mechanisms are independent of the number of users. Moreover, the computation time is linearly increased with the number of experts. However, we only stress the improvement achieved by applying learning mechanisms. The results shown for each set of experiments are averaged over many runs, where each run is executed with different seeds for the benchmarking procedure. The common benchmark settings of the following figures, i.e., I, F, U, P, are fixed at 5000, 8, 50, 5%, respectively.

Figure 5 and Figure 6 depict the improvement achieved by learning mechanisms. The user classification behaviors for Figure 5 is $\mathring{A}_2$, which is more favorable to Yoda,



a. Similarity values b. Precision c. Recall

Fig. 5. The system improvement by using user classification behaviors $\mathring{A}_1$



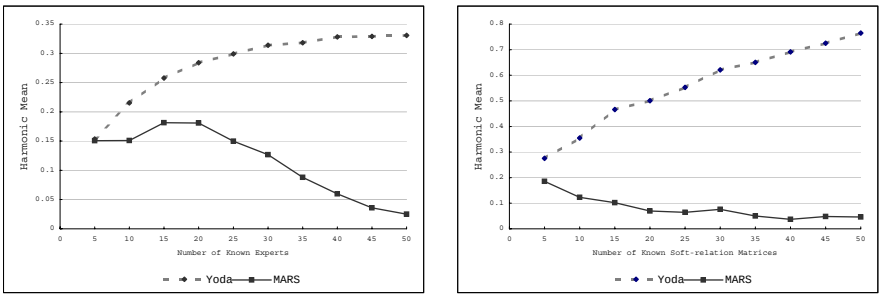
a. Similarity values b. Precision c. Recall

Fig. 6. The system improvement by using user classification behaviors $\mathring{A}_2$

and the user classification behaviors for Figure 6 is $\mathring{A}_1$, which is more preferable to MARS. The benchmark settings of these two figures, N, E, O, are set at 0, 50, and 10, respectively. In other words, both systems only obtain partial information. The X-axes of Figure 5 and Figure 6 represents the number of generations. The Y-axes illustrate the accuracy in different measurements.

As shown in Figure 5 and Figure 6, the Yoda learning mechanism constantly outperforms the MARS learning method, regardless of which classification data are used. For example, in Figure 5.b, although the standard answers (user classification behaviors) are expected to be more favorable to MARS, Yoda increases its precision by nearly 90% while MARS only achieves a 40% improvement. Moreover, as can be seen, the GA-based learning mechanism of Yoda can improve the accuracy of the result set regardless of the measurements, i.e., the similarity values on ranking, precision, and recall. On the other hand, the statistical learning method of MARS can only enhance the accuracy in similarity and the precision measurements. In some circumstances, such as the one shown in Figure 6.c, MARS learning method even decreases its recall rates. Generally, the accuracy of result sets are improved as the number of generations increases, especially at the beginning phases of learning processes. However, it should be noted that the Yoda learning mechanism only acquires feedback from users once, while MARS does this in each generation.

Figure 7 demonstrates the impact of O (number of known experts) in the improvement achieved by learning methods. The benchmark settings of this figure, N and E, are set at 0 and 50. The X-axis depicts the number of experts, and the Y-axis represents the harmonic mean computed by Equation (14). As shown in Figure 7, Yoda can constantly benefit from the gaining knowledge (i.e., additional



a. With $\mathring{A}_1$ data b. With $\mathring{A}_2$ data

Fig. 7. The system improvement after 50 generations

classification data of known experts), while MARS usually cannot. However, as can be seen in Figure 7.a, the accuracy of MARS improves as O increases while O is less than 20, and afterward the accuracy of MARS dramatically drops. This is probably because the impact of noise on MARS is larger than the advantage of gaining knowledge as the number of known experts grows.

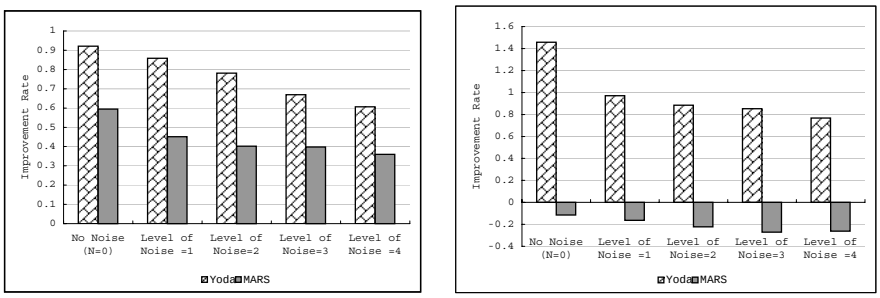
In order to compare the system performance when the user relevance feedback is imperfect, we introduce five different noise levels in the experiment of Figure 8. The benchmark settings of this figure, E, O are fixed at 20 and 15, respectively. Let $\ddot{H}_o$ be the harmonic mean of results that obtain perfect feedback and $\ddot{H}_i$ represent the harmonic mean of results that acquire imperfect feedback after 10 generations. The Y-axis depicts the improvement rate of harmonic means computed by Equation (16) .

$$\text{Improvement Rate}(\ddot{H}_o, \ddot{H}_i) = \frac{(\ddot{H}_i - \ddot{H}_o)}{\ddot{H}_o} \tag{16}$$

As can be seen in Figure 8, although the noise levels affect both systems, our learning mechanism still achieves more improvement than does MARS. To be specific, Yoda can improve the quality of user profiles in the range of 50% to 90% depending on the noise level, while the improvement rates by MARS range from -20% to 45%. Moreover, it also indicated that the noise impacts improvement rates more on Yoda than on MARS. This finding implies that our GA-based learning mechanism can better utilize relevance feedback for exploring the best profile; hence, the imperfect feedback could lead to false directions. Overall, this figure reveals that our learning mechanism has the ability to tolerate noises during the learning process as well as MARS does.

6. CONCLUSION

We proposed a model that conceptualizes content-based querying as the task of classifying images into classes. Our model employs a fuzzy-logic based aggregation function for ranking images. We showed that in addition to some performance benefits, fuzzy aggregation is less sensitive to noise and can support disjunctive queries as compared to weighted-average aggregation used by other content-based



a. With $\check{A}_1$ data b. With $\check{A}_2$ data

Fig. 8. Impact of noise

image retrieval systems.

However, this system heavily relies on user profiles for the aggregation task. The system accuracy may decline if user profiles are inaccurate. Therefore, we introduced a learning mechanism that utilizes the users' relevance feedback to improve the profiles automatically using genetic algorithms. Our learning mechanism requires less interaction from the user and results in faster convergence to the user's preferences as compared to other learning techniques. The experimental results indicated that the accuracy of results significantly increased more than 100% by our GA-based learning mechanism. It also demonstrated that our learning mechanism has the ability of tolerating noises during learning processes and improvement is in the range of 50% to 90% depending on the noise level.

REFERENCES

Aggarwal, G., Ghosal, S., and Dubey, P. 2000. Efficient query modification for image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Volume 2 (June 2000), pp. 255 – 262.

Bartolini, I., Ciaccia, P., and Waas, F. 2001. Feedbackbypass: A new approach to interactie similarity query processing. In *Inernational Conference on Very Large Data Bases* (2001).

Blackburn, S. and DeRoure, D. 1998. A tool for content based navigation of music. In *Proceedings of ACM Multimedia* (Bristol, UK, 1998), pp. 361–368.

Chen, Y.-S. and Shahabi, C. 2001. Automatically improving the accuracy of user profiles with genetic algorithm. In *Proceeding of IASTED International Conference on Artifical Intelligence and Soft Computing* (Cancun, Mexico, May 2001).

Doulamis, N., Doulamis, A., and Kollias, S. 2000. Non-linear relevance feedback: Improving the performance of content-based retrieval systems. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Volume 1 (July 2000), pp. 331 – 334.

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

Fagin, R. and Maarek, Y. 2000. Allowing users to weight search terms. In *Proceedings of RIAO (Computer-Assisted Information Retrieval)'2000* (2000), pp. 682–700.

Fagin, R. and Wimmers, E. 2000. A formula for incorporating weights into scoring rules. *Theoretical Computer Science 239*, 309–338.

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*

(Edinburgh, Scotland, September 1999).

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley, Wokingham, England.

Holland, J. 1975. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan.

Hunter, A. 1995. Sugal programming manual. *http://www.trajan-software.demon.co.uk/sugal.htm*.

Karnik, N. and Mendel, J. 1998. Introduction to type-2 fuzzy logic systems. In *Proceeding of 1998 IEEE FUZZ Conference* (Anchorage, AK, May 1998), pp. 915–920.

Karnik, N. and Mendel, J. 2000. Operations on type-2 fuzzy sets. *Int'l. J. on Fuzzy Sets and Systems*.

Lew, M. S. 2000. Next-generation web searches for visual content. *Computer*, 46–53.

Li, S. 2000. Content-based audio classification and retrieval using the nearest feature line method. *IEEE Transactions On Speech and Audio Processing 8(5)*, 619–625.

Lin, S., Ozsu, M., Oria, V., and Ng, R. 2001. An extendible hash for multi-precision similarity querying of image databases. In *Inernational Conference on Very Large Data Bases* (2001).

Meilhac, C. and Nastar, C. 1999. Relevance feedback and category search in image databases. In *IEEE International Conference on Multimedia Computing and Systems* (Florence, Italy, June 1999).

Moukas, A. 1996. Amalthea: Information discovery and filtering using a multiagent evolving ecosystem. In *1st Int. Conf. on The Practical Applications of Intelligent Agents and MultiAgent Technology (PAAM)* (London, 1996).

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 (1993), pp. 173–187.

Ortega, M., Rui, Y., Chakrabarti, K., and Mehrotra, S. 1998. Supporting ranked boolean similarity queries in mars. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 10*, 905–025.

Shahabi, C., Banaei-Kashani, F., Chen, Y.-S., and McLeod, D. 2001. Yoda: An accurate and scalable web-based recommendation system. In *Proceeding of the sizth International Conference on Cooperative Information Systems* (Trento, Italy, September 2001).

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* (Rome, Italy, August 2000).

Shahabi, C. and Chen, Y.-S. 2000b. Soft query in image retrieval systems. In *Proceedings of the SPIE Internet Imaging (EI14), Electronic Imaging 2000, Science and Technology* (San Jose, California, January 2000).

Sheth, B. 1993. Evolving agents for personalized information filtering. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications* (1993).

Tong, S. and Chang, E. 2001. Support vector machine active learning for image retrieval. In *Proceedings of ACM Multimedia 2001* (Ontario, Canada, September 2001).

Wood, M. E. J., Campbell, N. W., and Thomas, B. T. 1998. Iterative refinement by relevance feedback in content-based digi tal image retrieval. In *ACM Multimedia 1998* (Bristol, UK, September 1998).

Wu, L., Faloutsos, C., Sycara, K., and Payne, T. 2000. Falcon: Feedback adaptive loop for content-based retrieval. In *Inernational Conference on Very Large Data Bases* (2000).

YRui, Huang, T., Ortega, M., and Mehrotra, S. 1998. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology 8*, 644–655.

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

ZHOU, X. S. AND HUANG, T. S. 2001. Small sample learning during multimedia retrieval using biasmap. In *Proceedings of IEEE Conf. Computer Vision and Pattern Recognition* (Hawaii, Decemeber 2001).