MEGA — The Maximizing Expected Generalization Algorithm for Learning Complex Query Concepts

Edward Chang and Beitao Li
Electrical & Computer Engineering
University of California, Santa Barbara
echang@ece.ucsb.edu

Abstract

Specifying exact query concepts has become increasingly challenging to end-users. This is because many query concepts (e.g., those for looking up a multimedia object) can be hard to articulate, and articulation can be subjective. In this study, we propose a query-concept learner that learns query criteria through an intelligent sampling process. Our concept learner aims to fulfill two primary design objectives: 1) it has to be expressive in order to model most practical query concepts, and 2) it must learn a concept quickly and with a small number of labeled instances since online users tend to be too impatient to provide much feedback. To fulfill the first goal, we model query concepts in k-CNF, which can express almost all practical query concepts. To fulfill the second design goal, we propose our *maximizing expected generalization algorithm* (MEGA), which converges to target concepts quickly by its two complementary steps: sample selection and concept refinement. To shorten MEGA's learning time, we further propose a divide-and-conquer method. The method judiciously divides a learning task into G subtasks to achieve a speedup of $O(G^{k-1})$. Through examples, analysis, experiments, and prototype demonstration, we show that MEGA converges to query concepts significantly faster than traditional methods.

Keywords: query concept, relevance feedback, active learning.

1 Introduction

One key design task, when constructing image database, is the creation of a relevance feedback component, which can learn user's query concepts effectively and efficiently. Indeed, without knowing the criteria by which the query is to be made (i.e., the query concept), a database system cannot conduct an accurate search. For many search tasks, however, a query concept is hard to articulate, and articulation can be subjective. For instance, in a multimedia search, it is difficult to describe a desired image using low-level features such as color, shape, and texture. Different users may perceive the same image differently; and even if an image is perceived similarly, users may use different vocabulary (i.e., different combinations of low-level features) to depict it. Furthermore, most users are not trained to specify simple query criteria using for example, SQL. It is thus both necessary (for capturing subjective concepts) and desirable (for alleviating users' need to specify complex queries) to build intelligent search engines that can quickly learn users' query concepts via relevance feedback.

Traditional relevance feedback and learning techniques, however, may not be suitable for online query-concept learning for at least two reasons. First, an online user is typically impatient and cannot be expected to provide a great deal of feedback, but traditional learning methods such as decision trees and neural networks require a large number of training instances to learn a concept [1, 22, 25, 28]. Second, almost all traditional relevance feedback methods [12, 16, 21, 23, 26, 34, 38] require users to have good examples at hand to seed a query. (For example, a good sunset image must be provided to search for more sunset pictures.) However, finding good seeds is the job of the search engine itself, and this circular requirement leaves the core problem— learning and understanding users' query concepts—unsolved.

In this study, we propose the *Maximizing Expected Generalization Algorithm* (MEGA) that 1) learns a concept quickly, with a minimum number of labeled instances, through an intelligent sampling process, and that 2) can sustain learning even when no relevant object is provided at the start of a query. MEGA models query concepts in k-CNF [17], which can model virtually all practical query concepts[1]. To ensure that target concepts can be learned with a small number of samples[2], MEGA employs an intelligent sampling scheme that can gather maximum information quickly. MEGA maximizes the usefulness of each sample it generates to query the user and therefore learns a query concept with a much smaller number of samples than traditional sampling schemes require. MEGA judiciously selects samples at each iteration and uses positive-labeled instances to learn the target concept in k-CNF. At the same time, it uses negative-labeled instances to shrink the candidate sampling space bounded by k-DNF. Even if all of the samples generated in the previous round of its relevance feedback are irrelevant, MEGA can use the negative-labeled instances to reduce substantially the set of potentially relevant samples and increase the probability that a relevant sample will be generated in the next round.

Learning a k-CNF concept can be time consuming. For instance, given M as the dimensionality of feature vectors, the time complexity for learning a k-CNF is polynomial to M [17]. When M is large, learning a k-CNF concept can be computationally intensive. To solve this "combinatorial explosion" problem [29], we further propose a divide-and-conquer method to reduce MEGA's learning time by dividing the learning task into G subtasks (to achieve a speedup of $O(G^{k-1})$ folds). At first sight, dividing features seems to be straightforward. For instance, a typical image system divides features into sets of color, texture, and shape. Our empirical study, however, shows that this "natural" or random division can often yield poor search precision. We show that those divisions that achieve high precision tend to have low intra-group feature correlation

[1] k-CNF is more expressive than k-term DNF, and it has both polynomial sample complexity and time complexity [18, 22].

[2] In this paper, we use the term *sample* when referring to a data item generated by MEGA for user labeling, and the term *instance* when referring to a data item used by MEGA to learn a query concept.

(or high inter-group feature correlation). Our experiments using both synthesized data and real images exhibit that MEGA converges to a target concept much faster (in terms of both wall-clock time and number of user iterations) than traditional schemes (e.g., MARS [15, 24, 26]). A Perception-based Image Retrieval (PBIR) prototype that we have built [5, 6] on several image sets (each image in our prototype is characterized by a 144-dimensional vector [7]) demonstrates that when the percentage of relevant objects to a query concept exceeds 1% of the dataset, MEGA can usually capture the concept in two to three feedback iterations, and can comprehend the concept very well in three to five iterations.

1.1 Motivating Example

Below we show a simple example problem that illustrates the need for MEGA. This example models an admission concept that consists of a small number of Boolean predicates. In Section 6 we show that MEGA works well to learn complex query concepts of fuzzy predicates for a high-dimensional image dataset.

Suppose Jane plans to apply to a graduate school. Before filling out the forms and paying the application fees, she would like to estimate her chances of being admitted. Since she does not know the admission criteria, she decides to learn the admission concept by induction[3]. She calls up a few friends who applied last year and obtains the information shown in Table 1.

Name	*GPA*	*GRE*	*Has Publications?*	*Was Admitted?*
Joe	*high*	*high*	*false*	*true*
Mary	*high*	*low*	*true*	*true*
Emily	*high*	*low*	*true*	*true*
Lulu	*high*	*high*	*true*	*true*
Anna	*low*	*low*	*true*	*false*
Peter	*low*	*high*	*false*	*false*
Mike	*high*	*low*	*false*	*false*
Pica	*low*	*low*	*false*	*false*

Table 1: Admission Samples.

If we look at the GRE scores in the table, we see that students with either high or low GRE scores were admitted, and both kinds were rejected. Hence, we may conclude that the GRE is irrelevant in the admission process. Likewise, one's publication record does not affect admission acceptance, nor does having a high GPA. It may appear that the admission decision is entirely random. However, the graduate school actually uses a combination of reasonable criteria: it

[3]At first sight, this example may seem very different from, say, an image search scenario, where a user queries similar images by example(s). But if we treat the admission officer as the user who knows what he/she likes and who can, accordingly, label an item as *true* or *false*, and if we treat Jane as the search engine who tries to find out what the admission officer thinks, then it is evident that this example represents a typical search scenario. After we present necessary definitions and notations, we will show an image-searching example in Section 4.4.

requires a high GPA and either a high GRE or publications. In other words, $Admission : GPA = high \wedge (GRE = high \vee Publications = true)$.

An obvious question arises: "Are all the samples in Table 1 equally useful for learning the target concept?" Apparently not, for several reasons. First, it seems that $Pica$'s record may not be useful since she was unlikely to be admitted (i.e., her record is unlikely to be labeled positive). Second, both Emily and Mary have the same record, so one of these two records can be redundant. Third, Lulu's record is perfect and hence does not provide additional insight for learning the admission criteria. This example indicates that choosing samples randomly may not produce useful information for learning a target concept. In this paper, we will show how and why MEGA can quickly learn a target concept like the example of admission criteria, whereas other methods may not.

1.2 Contributions

In summary, the contributions of this work are the following:

1. We propose MEGA, which learns an online query concept with minimum number of labeled instances through active learning.

2. We propose a divide-and-conquer method to divide high-dimensional features into G groups to speed up MEGA learning. We observe that selecting appropriate size for groups, and maximizing inter-group feature correlation (or minimizing intra-group correlation), can speed up query-concept learning by $O(G^{k-1})$ folds, while maintaining high search accuracy.

3. We show that our concept learner can tolerate moderate levels of noise, i.e., it works quite well when a target concept can be approximated by k-CNF and when training data contain some errors.

The rest of this paper is organized into six sections. Section 2 discusses related work. Section 3 shows how we model a query concept and how we measure similarity between an object and the query concept. Section 4 depicts the sample selection and concept refinement steps of MEGA. We also discuss techniques to improve MEGA's robustness and efficiency in this section. Section 5 and Section 6 present our experimental results in synthesized data and real images, respectively. Finally, we offer our conclusions in Section 7.

2 Related Work

In most inductive learning problems studied in the AI (Artificial Intelligence) community, samples are assumed to be taken randomly in such a way that various statistical properties can be derived conveniently [14, 37]. However, for online applications where the number of samples must be small (lest impatient users be turned away), random sampling is not suitable. Although ensemble techniques such as *bagging* [3, 40], arcing [4], and boosting [32, 33] have been recently proposed to improve learning accuracy for decision trees and neural networks, these techniques may not help reduce the number of required samples to learn an online query concept.

Relevance feedback techniques proposed by the IR (Information Retrieval) community do perform non-random sampling [16]. The study of [27] puts these query refinement approaches into three categories: *query reweighting*, *query point movement* and *query expansion*.

- *Query reweighting* and *query point movement* [15, 24, 26, 30]. Both query reweighting and query point movement use nearest-neighbor sampling: They return top ranked objects to be marked by the user and refine the query based on the feedback. If the initial query example is good, this nearest-neighbor sampling approach works fine. However, most users will not have a good example to start a query. MEGA can effectively use negative labeled instances to refining the sampling space to find relevant samples.
- *Query expansion* [27, 38]. The *query expansion* approach can be regarded as a multiple-instances sampling approach. The samples of the next round are selected from the neighborhood (not necessarily the nearest ones) of the positive-labeled instances of the previous round. The study of [27] shows that query expansion achieves only a slim margin of improvement (about 10% in precision/recall) over query point movement.

Of late, Bartolini and co-workers [2] proposed FeedbackBypass , which can expedite relevance feedback learning with the help of past experience from user interactions. FeedbackBypass organizes the past experience into a Simplex tree, and the information can be used to "bypass" some feedback loops or to predict parameters for new query.

Additionally, pool-based learning or active learning has been introduced for choosing good samples from the unlabeled data pool to speed up the learning process. For image retrieval, the PicHunter system [9] uses Bayes' rule to predict the goal image, based upon the users' actions. The system shows that employing active learning can drastically cut down the number of iterations (up to 80% in some experiments). The authors also point out, however, that their scheme is computationally intensive since it recomputes conditional probability for all unlabeled samples after each round of user feedback. Hence it may not scale well with dataset size. SVM_{Active} [35] is another active learning algorithm that we have recently developed for image retrieval. But, SVM_{Active} requires seeding a query with good examples to work well. MEGA scales well with data dimensionality and dataset size since it retrieves only a small number of informative samples via an effective indexer that we have developed [20]. And also, MEGA can sustain the learning process without good initial examples.

Another method, the Query by Committee (QBC) algorithm [13], employs a committee of learners to select unlabeled samples that yield maximum information gain. Two major challenges of applying the QBC algorithm to a learning task are 1) finding useful committee members, and 2) finding informative unlabeled instances efficiently. Our proposed MEGA can be regarded as a variant of the QBC algorithm. MEGA provides an effective method for refining committee members (i.e., a k-CNF and a k-DNF hypothesis), and at the same time delimits the boundary of the sampling space for efficiently finding useful samples.

Finally, much traditional work suffers from model bias. Some systems (e.g., [10, 11]) assume that the overall similarity can be expressed as a weighted linear combination of similarities in features. Similarly, some systems assume that query concepts are disjunctive [38]. When a query concept does not fit the model assumption, these systems perform poorly. MEGA works quite well with moderate model bias or noisy feedback.

3 Definitions and Notations

Our learner models query concepts in k-CNF and uses k-DNF to bound the uncertain region for sampling.

Definition 1: k-**CNF**: For constant k, the representation class k-CNF consists of Boolean formulae of the form $c_1 \wedge \cdots \wedge c_L$, where each c_i is a disjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$. No prior bound is placed on L.

Definition 2: k-**DNF**: For constant k, the representation class k-DNF consists of Boolean formulae of the form $d_1 \vee \cdots \vee d_L$, where each d_i is a conjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$. No prior bound is placed on L.

In our retrieval system, query concepts are Boolean expressions consisting of predicates connected by the Boolean operators $\vee$ (or) and $\wedge$ (and). A predicate on feature x_k in our system is denoted as P_{x_k}. Note that the membership function [8] of P_{x_k} can be either crispy or fuzzy. Our goal is to find the proper operators to combine individual predicates to form the query concept. We choose k-CNF to model query concepts, since it can express most practical queries and can be learned via positive-labeled instances in polynomial time [17, 22]. In addition, we use non-positive-labeled instances to reduce our sampling space (bounded by k-DNF), which we will discuss in detail in Section 4.

A k-CNF has three characteristics:

1: The terms (or literals) are combined by the $\wedge$ operator.

2: The predicates in a term are combined by the $\vee$ operator.

3: A term can have at most k predicates.

Suppose we have three predicates P_{x_1}, P_{x_2} and P_{x_3}. The most specific 2-CNF concept of these predicates is

$$P_{x_1} \wedge P_{x_2} \wedge P_{x_3} \wedge (P_{x_1} \vee P_{x_2}) \wedge (P_{x_1} \vee P_{x_3}) \wedge (P_{x_2} \vee P_{x_3}).$$

Suppose a dataset contains N objects, denoted as O_i, where $i = 1 \ldots N$. Suppose each object can be depicted by M features, each of which is denoted by x_k, where $k = 1 \ldots M$ (in the reminder of this paper, when we refer to "a feature", we mean a single number in a feature vector). To find objects that best satisfy a k-CNF concept, we need to compute the membership value of an object for the concept. We first compute the membership values at predicate level and then aggregate them into the membership value at concept level.

At the predicate level, suppose $F_{x_k}(O_i)$ is the value of object i on feature x_k. We transform $F_{x_k}(O_i)$ to the predicate level membership value using a membership function μ: $F_{x_k}(O_i) \rightarrow [0,1]$. Let $P_{x_k}(O_i)$ denote the predicate level membership value of object i for predicate P_{x_k}. $P_{x_k}(O_i) = 1$ means that object i fully satisfies predicate P_{x_k}, $P_{x_k}(O_i) = 0$ means that object i does not satisfy predicate P_{x_k} at all, and $0 < P_{x_k}(O_i) < 1$ means that object i satisfies predicate P_{x_k} to a degree. For example, people's heights can vary from 3 feet to 9 feet. If we want to grasp the fuzzy predicate of "tall", we use a membership function to transform each height value between 3 feet and 9 feet to a membership value between 0 and 1. If we consider people over 7 feet to be definitely tall, then we map heights over 7 feet to 1.0. If we consider people under 5 feet definitely not tall, then we map heights under 5 feet to 0.0. Heights between 5 and 7 feet are considered tall to a degree and we map them to membership values between 0.0 and 1.0.

At the concept level, we use the standard fuzzy rules, as defined by Zadeh [10, 39], to aggregate membership values on individual predicates. To combine M predicate level membership values into one concept level membership value, we need an *M-ary* aggregation function that maps $[0,1]^M$ to $[0,1]$. The rules are as follows:

Conjunctive rule: $P_{x_1 \wedge x_2 \wedge \ldots \wedge x_M}(O_i) = \min\{P_{x_1}(O_i), P_{x_2}(O_i), \ldots P_{x_M}(O_i)\}$.

Disjunctive rule: $P_{x_1 \vee x_2 \vee \ldots \vee x_M}(O_i) = \max\{P_{x_1}(O_i), P_{x_2}(O_i), \ldots P_{x_M}(O_i)\}$.

Negation rule: $P_{\neg x_k}(O_i) = 1 - P_{x_k}(O_i)$.

The objects that best satisfy a concept have the largest aggregated membership values. To assist the reader, Table 2 summarizes the parameters that we have introduced and that we will discuss in the rest of the paper.

4 The MEGA Algorithm

Our learning algorithm consists of two major parts:

- Refinement: Refine the query concept based on positive-labeled instances. The refinement step is carefully designed to tolerate noisy data.

Parameter	*Description*
U	Unlabeled dataset
M	The number of features for depicting a data object
N	The number of data objects in U
u	A set of samples selected from the unlabeled set U
x_i	The i^{th} feature
O_j	The j^{th} object
y_j	The label of the j^{th} object
y	The labeled set u
y^+	The positive-labeled set
y^-	The negative-labeled set
QCS	The set representation of the query concept space in k-CNF
CCS	The set representation of the candidate concept space in k-DNF
d_i	The i^{th} disjunctive term in QCS
c_i	The i^{th} conjunctive term in CCS
t_i	d_i or c_i
$F_{x_k}(O_i)$	The value of O_i on feature x_k
$P_{x_k}(O_i)$	The membership value of O_i on predicate P_{x_k}
$P_{t_i\|y_j}$	The probability of removing term t_i given y_j
$P_{t_i\|y}$	The probability of removing term t_i given y
K_α	Sample size
K_c	The threshold of eliminating a conjunctive term, c_i
K_d	The threshold of eliminating a disjunctive term, d_i
γ	Voting parameter
$f()$	Func. computing the prob. of removing term t_i given y_j
$Vote()$	Func. computing the aggregated probability of removing t_i
$Sample()$	Sampling func., which selects u from U
$Feedback()$	Labeling function
$Collapsed?()$	The version space has collapsed? true or false
$Converged?()$	The version space has converged? true or false

Table 2: Parameters.

- Sampling: Refine the sampling space based on negative-labeled instances and select samples judiciously for expediting the learning process.

4.1 Refinement

We use Valiant's learning algorithm [36] as the starting point to refine a k-CNF concept. We extend the algorithm to

1: Handle the fuzzy membership functions (described in Section 3),

2: Select samples judiciously to expedite the learning process (Section 4.2), and

3: Tolerate user errors (Section 4.5).

MEGA initializes the query concept space (QCS) as the most specific k-CNF and the candidate concept space (CCS) as the most general k-DNF. The target concept that MEGA learns is more general than the initial QCS and more specific than the initial CCS. It will become evident shortly that MEGA learns the QCS, while at the same time refining the CCS to delimit the boundary of the sampling space. (The shaded area in Figure 1 shows the sampling space between

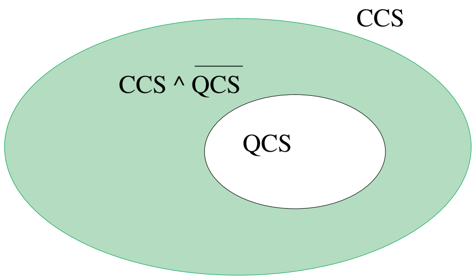


Figure 1: MEGA's Sampling Space: $CCS \wedge \overline{QCS}$.

the QCS and the CCS).

Figure 2 depicts the learning algorithm. The first step initializes QCS and CCS. The second step describes how a query concept is refined through relevance feedback. This refinement step consists of the four substeps shown.

Step 2.a: MEGA Generates sample set u from the unlabeled pool U. This step passes QCS, CCS and U to procedure *Sample* to generate K_α samples. We will discuss procedure *Sample* in Section 4.2.

Step 2.b: MEGA solicits user feedback. A user marks an object positive if the object fits his/her query concept. An unmarked object is considered negative.

Step 2.c: This is MEGA's term reduction step. It refines both QCS and CCS.

Step 2.c.1 refines QCS. For each disjunctive term in the QCS, MEGA examines each positive-labeled instance and uses function f to compute the probability that the disjunctive term should be eliminated. MEGA then calls procedure *Vote* to tally the votes among the positive-labeled instances and compares the vote with threshold K_d to decide whether that disjunctive term is to be removed. If sufficient numbers of positive-labeled instances contradict the QCS with respect to a disjunctive term (i.e., if the threshold is exceeded), the term is removed from the QCS. We discuss procedure *Vote*, which decides how aggressive MEGA is in eliminating terms, in Section 4.5.

Step 2.c.2 refines CCS. Similar to the steps in 2.c.1, for each conjunctive term in the CCS, MEGA examines each negative-labeled instance, and uses function f to compute the probability that the conjunctive term should be eliminated. MEGA then calls procedure *Vote* to tally the votes among the negative-labeled instances. Then it compares the vote with threshold K_c to decide whether that conjunctive term is to be removed. If sufficient numbers of negative-labeled instances satisfy the CCS with respect to a conjunctive term, the term is removed from the CCS.

Step 2.d: MEGA performs bookkeeping by reducing the unlabeled pool.

The refinement step terminates when the learning process converges to the target concept ($Converged? = true$) or the concept is collapsed ($Collapsed? = true$). ($Converged?$ and $Collapsed?$ are defined below.) In practice, the refinement stops when no unlabeled instance

Algorithm MEGA

Input: U, K_c, K_d, K_α;

Output: QCS;

Procedure calls: $f()$, $Vote()$, $Sample()$, $Feedback()$, $Collapsed?()$, $Converged?()$;

Variables: c, d, u, y, $P_{t_i|y_j}$, $P_{t_i|y}$;

Begin

1 Initialize the version space

$QCS \leftarrow \{d_1, d_2, \ldots\}$; $CCS \leftarrow \{c_1, c_2, \ldots\}$;

2 Refine query concept via relevance feedback

While (*not* $Collapsed?(QCS, CCS)$ *and not* $Converged?(QCS, CCS)$)

2.a Select informative samples

$u \leftarrow Sample(QCS, CCS, U, K_\alpha)$;

2.b Solicit user feedback

For each $u_i \in u$

$y_i \leftarrow Feedback(u_i)$;

2.c Reduce irrelevant terms

2.c.1 Refine k-CNF using positive samples

For each $d_i \in QCS$

For each $y_j \in y^+$

$P_{d_i|y_j} \leftarrow f(d_i, O_j, QCS)$;

$P_{d_i|y^+} \leftarrow Vote(y^+, P_{d_i|y_j \in y^+}, \gamma)$;

If $(P_{d_i|y^+} > K_d)$

$QCS \leftarrow QCS - \{d_j\}$;

2.c.2 Refine k-DNF using negative samples

For each $c_i \in CCS$

For each $y_j \in y^-$

$P_{c_i|y_j} \leftarrow f(c_i, O_j, CCS)$;

$P_{c_i|y^-} \leftarrow Vote(y^-, P_{c_i|y_j \in y^-}, \gamma)$;

If $(P_{c_i|y^-} > K_c)$

$CCS \leftarrow CCS - \{c_j\}$;

2.d Bookkeeping

$U \leftarrow U - u$;

3 Return query concept

Output QCS;

End

Figure 2: Algorithm MEGA.

u can be found between the QCS and the CCS.

Definition 3: $Converged?(QCS, CCS)$

$$Converged?(QCS, CCS) \leftarrow \ true\ if\ CCS == QCS; \quad false\ otherwise.$$

Definition 4: $Collapsed?(QCS, CCS)$

$$Collapsed?(QCS, CCS) \leftarrow \ true\ if\ CCS \in QCS; \quad false\ otherwise.$$

Remarks

Notice that MEGA does not require seeding a query with relevant images. MEGA intelligently generates images to seek for user feedback. If all of the images generated in the first round of its relevance feedback are irrelevant, it will use these irrelevant images to reduce the sampling space so that a relevant image has a higher probability of being generated in the next round.

4.2 Sampling

MEGA invokes procedure *Sample* to select the next K_α unlabeled instances to ask for user feedback. From the college-admission example presented in Section 1, we learn that if we would like to reduce the required number of samples (i.e., call a minimum number of friends), we should choose our samples judiciously. But, what constitutes a good sample? We know that we learn nothing from a sample if

- It agrees with the concept in all terms.
- It has the same features as another sample.
- It is unlikely to be labeled positive.

To make sure that a sample is useful, MEGA employs two strategies:

1. Bounding the sample space: MEGA avoids choosing useless unlabeled instances by using the CCS and QCS to delimit the sampling boundary.

2. Maximizing the usefulness of a sample: MEGA chooses a sample that shall remove the maximum expected number of disjunctive terms in QCS. In other words, MEGA chooses a sample that can maximize the expected generalization of the concept.

It may appear that if we pick a sample that has more dissimilar disjunctions (compared to the QCS), we may have a better chance of eliminating more disjunctive terms. This is, however, not true. A sample must be labeled by the user as positive to be useful for refining QCS. Unfortunately, a sample with more disjunctions that are dissimilar to the target concept is less likely to be labeled positive. Therefore, in choosing a sample, there is a tradeoff between those with more contradictory terms and those more likely to be labeled positive.

Let Ψ denote the number of disjunctions remaining in the concept. The number of disjunctions that can be eliminated in the current round of sampling (denoted as ψ) is between zero and Ψ. For a sample contradicting exactly ψ disjunctive terms in QCS, we can write the probability of its being labeled positive as $P_e(\psi)$. $P_e(\psi)$ is a monotonically decreasing function of ψ.

Our MEGA scheme attempts to find the ψ that can eliminate the maximum expected number of disjunctive terms, given a sample. MEGA's objective function can be written as

$$\psi^* = argmax_{\psi} E(\psi) = argmax_{\psi}(\psi \times P_e(\psi)). \tag{1}$$

To solve ψ^*, we must know $P_e(\psi)$, which can be estimated by the two methods below: *probabilistic estimation* and *empirical estimation*.

4.3 Probabilistic Estimation

We first consider how to estimate ψ^* using a probability model. As we have seen in the college-admission analogy, if a sample contradicts more disjunctive terms, it is more likely to be labeled negative (i.e., less likely to be labeled positive). For example, a sample that contradicts term T_1 is labeled negative only if T_1 is in the user's query concept. A sample that contradicts both terms T_1 and T_2 is labeled negative if either T_1 or T_2 is in the user's query concept.

Formally, let random variable Φ_i be 1 if T_i is in the concept and 0 otherwise. For simplicity, let us assume that the Φ_i's are iid (independent and identically distributed), and the probability of Φ_i being 1 is p ($0 < p < 1$). For a sample contradicting ψ disjunctive terms, it is marked positive only when none of these ψ terms appears in the user's query concept. This probability is $(1-p)^{\psi}$. If we substitute $P_e(\psi)$ by $(1-p)^{\psi}$ on the right-hand side of Equation 1, we get

$$E(\psi) = \psi(1-p)^{\psi}.$$

If we take the derivative of $E(\psi)$, we can find the optimal ψ value, denoted by ψ^*:

$$\psi^* = \Psi,\ if\ \frac{1}{ln\frac{1}{1-p}} > \Psi, \quad \psi^* = \frac{1}{ln\frac{1}{1-p}},\ otherwise.$$

Of course, it may be too strong an assumption that the probability p of all disjunctions is iid. However, we do not need a precise estimation here for the following two reasons:

1. Precise estimation may not be feasible and can be computationally intensive.

2. An approximate estimation is sufficient for bootstrapping. Once the system is up and running for a while and collects enough data, it can empirically estimate $P_e(\psi)$ using its past experience. We discuss this process next.

4.4 Empirical Estimation

MEGA can estimate $P_e(\psi)$ based on its past experience. For each sample it presents, it records how many disjunctions the sample contradicts with respect to the query concept and whether the sample is labeled positive. Once the system collects a sufficient amount of data, we can estimate $P_e(\psi)$ empirically. We then pick the ψ^* that can eliminate the maximum expected number of disjunctive terms.

Again, a reasonable approach to estimate $P_e(\psi)$ is to use probabilistic estimation when the system first starts and then switches to empirical estimation when the system has collected sufficient data. In such a case there can be an abrupt transition when MEGA switches from one estimation to the other, since the two estimates of $P_e(\psi)$ may differ substantially. This may lead to a sudden change in behavior of the sampling component of the active learner. To remedy this problem, we employ a Bayesian smoothing approach. Essentially the probabilistic estimation is the prior guess at the distribution over ψ and the empirical approach is the guess based purely on the data that MEGA has seen so far. The Bayesian approach combines both of these guesses in a principled manner. Before we start, we imagine that we have seen a number of samples of ψ. After refinement iteration we gather new samples for ψ; then we add them to our current samples and adjust $P_e(\psi)$.

For example, before we start, we assume that we have already seen samples with $\psi = 1$ being labeled positive three out of five times and samples with $\psi = 2$ being labeled positive seven out of 20 times. In other words, we have successfully eliminated $\psi = 1$ term three times out of five, and we have successfully eliminated $\psi = 2$ terms 7 times out of 20. Thus initially $P_e(\psi = 1) = 3/5 = 0.6$ and $P_e(\psi = 2) = 7/20 = 0.35$. Now suppose we do a query in which we observe a sample with $\psi = 2$ being labeled positive. Then our new distribution is $P_e(\psi = 1) = 3/5$ and $P_e(\psi = 2) = 8/21$. We continue in this manner. At first the prior assumption has quite an effect on our guess about the distribution. The more imaginary samples we have in our prior assumption, the larger its effect. For instance, if we assume that $\psi = 1$ being labeled positive 30 out of 50 times and that $\psi = 2$ being labeled positive 70 out of 200 times, it takes more real samples to change $P_e(\psi)$. With time, the more real samples we get, the less the effect of the prior assumption becomes, until eventually it has virtually no effect, and the observed data dominate the expression. This procedure gives us a smooth transition between the "probabilistic" and the "empirical" methods.

Remarks

Once the criteria for selecting samples is determined, we use Clindex, a clustering/indexing combined indexer to retrieve the most informative samples. We discuss this high-dimensional access method in detail in [20].

Sampling Example

We use an example to illustrate the essential ideas of our sampling algorithm. In this simple example, we use 1-CNF to model query concepts and use 1-DNF to bound the sampling space. We consider four predicates:

$$P_b = Blue\ High \qquad P_g = Green\ High \qquad P_w = White\ High \qquad P_y = Yellow\ High$$

Suppose during the query process, the QCS has been refined to be:

$$QCS = P_b \wedge P_g \wedge P_w \wedge P_y$$

and CCS has been refined to be:

$$CCS = P_b \vee P_g \vee P_w \vee P_y$$

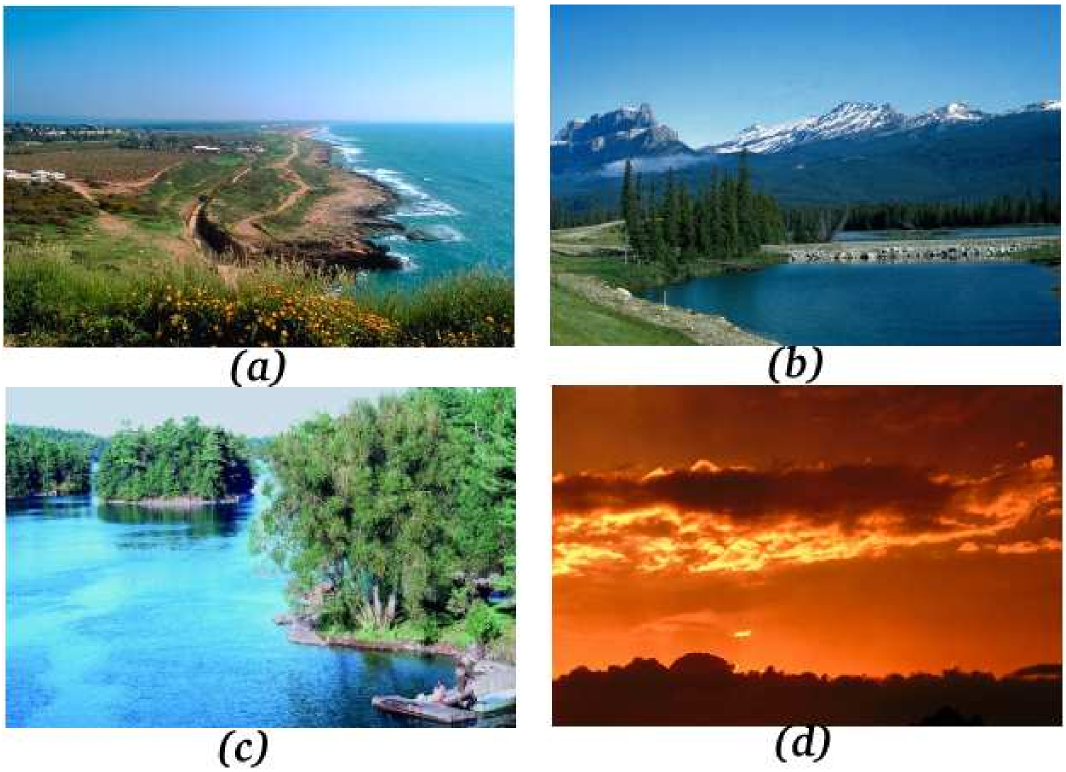


Figure 3: Four candidate images for sampling

Suppose there are four candidate images (as shown in Figure 3) that the system can pick to present to the user. In Figure 3(a), all four colors exist substantially in this image. Since this image agrees with the current QCS concept (does not contradict it by any term). It will not be useful in refining the query concept. In contrast, Figure 3(d) does not have any of the four colors. Using the CCS as the bound for the sampling space, we find that this image falls out of the sampling bound (i.e. it does not satisfy any term of CCS). For this reason, it will be labeled negative and will not be helpful since it cannot shrink the sampling bound. As to the remaining candidates. Figure 3(b) has blue, green and white colors; thus it contradicts only one term (P_y) in the QCS. Figure 3(c) has blue and green colors; thus it contradicts two terms (P_y and P_w) in the QCS. To decide which one is more useful for refining the query concept, we estimate the probability of their being labeled positive. For instance, if we estimate that (b) has 70% probability of being labeled positive while (c) has 45% probability. Then the expected number of terms (b) can reduce is $70\% \times 1 = 0.7$, while (c) can reduce $45\% \times 2 = 0.9$. From this analysis we decide that (c) is more useful in refining the current query concept, which propels the system to sample image (c).

This admittedly simplified example ($M = 4$, $k = 1$) explains how MEGA selects the most informative samples. As we will show in Section 6, an image concept is usually depicted by a large number of features (e.g., $M > 100$) and the value of k is typically higher as well.□

4.5 Procedure Vote

In the previous example, we assume that all samples are noise-free. This assumption may not be realistic. There can be two sources of noise:

- Model bias: The target concept may not be in k-CNF.
- User errors: A user may label some positive instances negative and vice versa.

Procedure Vote
Input: y, $P_{t_i|y_j \in y}$, γ;
Output: $P_{t_i|y}$;
Begin
Sort $P_{t_i|y_j}$ in the descending order;
Return the γ^{th} highest $P_{t_i|y_j}$;
End

Figure 4: Procedure Vote.

Procedure *Vote* controls the strictness of voting using γ. The larger the value of γ is, the more strict the voting is and therefore the harder it is to eliminate a term. When the noise level is high, we have less confidence in the correctness of user feedback. Thus, we want to be more cautious about eliminating a term. Being more cautious means increasing γ. Increasing γ, however, makes the learning process converge more slowly. To learn a concept when noise is present, one has to buy accuracy with time. In Section 5.3 we show how the voting threshold γ affects search accuracy and convergence speed through our experimental results, and we suggest heuristics for selecting a proper γ.

4.6 Improving Learning Efficiency

The MEGA scheme described so far is not yet concerned with scalability in regard to M (the dimensionality of a feature vector for depicting an object). In this section, we describe a divide-and-conquer method for tackling this *combinatorial explosion* problem.

The number of disjunctions in a k-CNF (and, likewise, the conjunctives in a k-DNF) can be written as

$$\sum_{i=1}^{k} \binom{M}{i}. \tag{2}$$

When M is large, a moderate k can result in a large number of disjunctive terms in a k-CNF, which causes high space and time complexity for learning. For instance, an image database that we have built [5] characterizes each image with 144 features ($M = 144$). The initial number of disjunctions in a 3-CNF is half a million, and in a 4-CNF is eighteen million.

To reduce the number of terms in a k-CNF, we divide a learning task into G sub-tasks, each of which is to learn a subset of the features. Dividing a feature space into G subspaces reduces both space and time complexity by a factor of $O(G^{k-1})$. For instance, setting $G = 12$ in our image database reduces both space and time complexity for learning a 3-CNF by about 144 times (the number of terms is reduced to $3,576$), and for learning a 4-CNF by $1,850$ times (the number of terms is reduced to $9,516$). The savings is enormous in both space and learning time. (The wall-clock time is less than half a second for one learning iteration for a 4-CNF concept on a Pentium-III processor.)

This divide-and-conquer approach may trade precision for speed, since some terms that involve features from more than one feature subsets can no longer be included in a concept. For instance, if we divide features a, b, c, and d into two groups $\{a, b\}$ and $\{c, d\}$, a 2-CNF can no longer have the following four terms: $a \vee c$, $a \vee d$, $b \vee c$, and $b \vee d$. Missing terms may or may not hurt search accuracy. If a query concept does not contain the missing terms, its search accuracy is not affected; conversely, its accuracy decreases. The loss of precision can be reduced by organizing a feature space in a multi-resolution fashion. The term *feature resolution* and a weak form of feature resolution that we call *feature correlation* are defined as follows:

Definition 5: **Feature resolution**: Feature P_i is said to have higher resolution than feature P_j if the presence of P_i implies the presence of P_j (or the absence of P_j implies the absence of P_i). Let $P_i \in P_j$ denote that P_i has higher resolution than P_j. We say that $P_i \in P_j$ if and only if the conditional probability $P(P_j|P_i) = 1$.

Definition 6: **Feature correlation**: A feature P_i is said to have high correlation with feature

P_j if the presence of P_i implies the presence of P_j and vice versa with high probability. We say that $P_i \sim P_j$ if and only if the conditional probability $P(P_j|P_i)/P(P_j) = P(P_i|P_j)/P(P_i) \geq \delta$.

MEGA takes advantage of feature resolution and correlation in two ways— *inter-group* multi-resolution and *intra-group* multi-resolution —for achieving fast and accurate learning.

- Inter-group multi-resolution features. If features can be divided into groups of different resolutions, we do not need to be concerned with terms that involve inter-group features. This is because any inter-group terms can be subsumed by intra-group terms. Formally, if P_i and P_j belong to two feature groups and $P(P_i|P_j) = 1$, then $P_1 \vee P_2 = P_2$ and $P_1 \wedge P_2 = P_1$. After exploring feature resolution, we can explore the weak form of resolution: feature correlation. Minimizing intra-group feature correlation also tends to maintain high search accuracy.
- Intra-group multi-resolution features. Within a feature group, the more predicates involved in a disjunctive term, the lower the resolution of the term. Conversely, the more number of predicates involves in a conjunctive term, the higher resolution the term is. For instance, in a 2-CNF that has two predicates P_1 and P_2, term P_1 and term P_2 have a higher resolution than the disjunctive term $P_1 \vee P_2$ and a lower resolution than the conjunctive term $P_1 \wedge P_2$. The presence of P_1 or P_2 makes the presence of $P_1 \vee P_2$ useless. Base on this heuristic, MEGA examines a term only when all its higher resolution terms have been eliminated.

5 Empirical Study with Synthetical Data

In this section we report our experimental results on synthesized data. (We show another set of experiments on a high-dimensional image dataset in Section 6.) The goals of our experiments were to evaluate

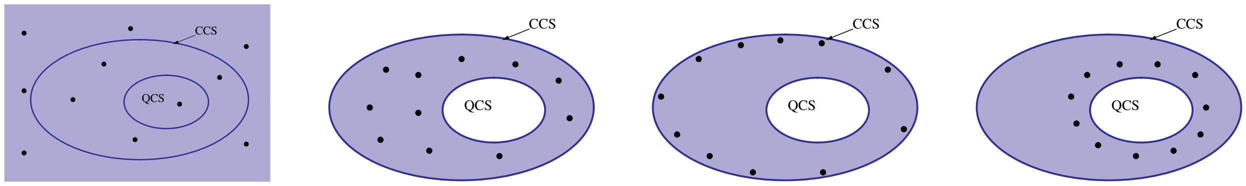
1: Whether MEGA can learn a concept accurately in the presence of a large number of irrelevant features.

2: Whether MEGA can converge to a target concept faster than traditional sampling schemes.

3: Whether MEGA is robust for noisy data or under situations in which the unknown target concept is not expressible in the provided hypothesis space.

We assumed all target concepts are in 3-CNF. We generated two datasets using different distributions: uniform and Gaussian. Each instance had 10 features between 0.0 and 1.0. The values of each feature in a dataset were independently generated. For the Gaussian distribution, we set its mean to 0.5 and its standard deviation to 1/6. Each dataset had $10,000$ vectors.

To learn a query concept, MEGA generated K_α samples in each relevance-feedback round to ask for user feedback. A sample was marked either positive or negative. In out tests, we defined an object that has a membership value larger than 0.9 a positive example. Since, given a target query concept, we knew which objects were positive and which were negative, we could automate



(a) Random (b) B-Random (c) Aggressive (d) MEGA

Figure 5: Sampling Schemes.

the labeling process in our experiments. After each round of relevance feedback, MEGA refined QCS and CCS, and returned top-k results that satisfied the QCS. We then matched these top-k results with the objects in the dataset that satisfied the target query concept. We used *precision* and *recall* to measure performance. Precision is the percentage of the top-k results that satisfy the target concept. Recall is the number of retrieved relevant objects over the total number of relevant objects in the dataset. Note that precision and recall can both be 100% if k objects meet the query concept, and the top-k retrieval returns all these k objects. We tallied precision/recall for up to ten iterations.

We compared MEGA with the four sampling schemes employed by some traditional systems: *random* (used by traditional AI schemes such as decision trees and neural networks), *bounded random* (used by QBC), *nearest neighbor* (used by MARS [15, 24, 26]), *query expansion* (used by [27, 38]). We also examined an *aggressive* sampling strategy that attempts to eliminate the maximum number of terms without considering whether a sample might be labeled positive or not.

Figure 5 shows how some of these sampling algorithms work. The main features of the sampling schemes are given below.

- *Random*: Samples are randomly selected from the bulk of the domain (Figure 5(a)).
- *Bounded Random*: Samples are randomly selected from between QCS and CCS (Figure 5(b)).
- *Nearest Neighbor*: Samples are selected from the nearest neighborhood of the center of the positive-labeled instances.
- *Query Expansion*: Samples are selected from the neighborhood of multiple positive-labeled instances.
- *Aggressive*: Samples are selected from unlabeled ones that satisfy the most general concepts in CCS (Figure 5(c)).
- *MEGA*: Samples are selected from between QCS and CCS to eliminate the maximum expected number of terms (Figure 5(d)).

We ran experiments on datasets of different distributions and repeated each experiment ten times. We tested many target concepts on the two synthesized datasets. We present only three representative test cases, those that represent a disjunctive concept, a conjunctive of disjunctions, and a complex concept with more terms. The three tests are

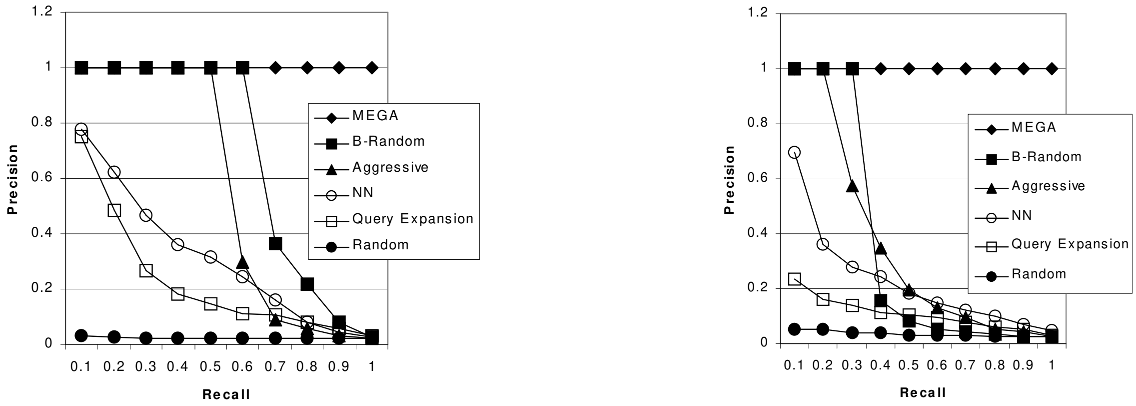
1: $P_1 \vee P_2$,

2: $(P_1 \vee P_2) \wedge P_3$,

3: $P_1 \wedge (P_2 \vee P_3) \wedge (P_4 \vee P_5 \vee P_6) \wedge (P_2 \vee P_4 \vee P_7)$,

We first assume that the dataset is free of user errors, and we set the sample size K_α to 20. In the remainder of this section, we first report our initial results, and then we report the effects of model bias (Sections 5.2), of user errors (Section 5.3), and of sample size K_α (Section 5.4) on MEGA.

5.1 Experimental Results



(a) $(P_1 \vee P_2) \wedge P_3$ (b) Complex

Figure 6: Precision vs. Recall After Three Iterations.

Figure 6 presents the precision/recall after three user iterations of the six sampling schemes learning the two concepts, $(P_1 \vee P_2) \wedge P_3$ and $P_1 \wedge (P_2 \vee P_3) \wedge (P_4 \vee P_5 \vee P_6) \wedge (P_2 \vee P_4 \vee P_7)$. The performance trend of the six schemes is similar at different numbers of iterations. We deem three iterations a critical juncture where a user would be likely to lose his/her patience, and thus we first present the results at the end of the third iteration. The performance curve of MEGA far exceeds that of the other five schemes at all recall levels. Note that for learning both concepts, MEGA achieves 100% precision at all recall levels.

Next, we were interested in how search accuracy improved with respect to the number of user iterations. This improvement trend tells us how fast a scheme can learn a target concept. We present below a set of tables and charts where we fix recall at 0.5 and examine the improvement in precision with respect to the number of iterations.

Rnd #	*Random*	*B-Random*	*N-Neighbor*	*Q-Expansion*	*Aggressive*	*Algorithm MEGA*
1	0.23715	0.23715	0.20319	0.20319	0.23715	0.23715
2	0.44421	0.44421	0.48207	0.44422	0.44421	0.30098
3	0.49507	0.50389	0.41036	0.45219	0.50389	1.00000
4	0.50389	1.00000	0.36753	0.51394	1.00000	1.00000
5	1.00000	1.00000	0.35857	0.78088	1.00000	1.00000
6	1.00000	1.00000	0.33865	0.88247	1.00000	1.00000
7	1.00000	1.00000	0.32669	0.93028	1.00000	1.00000
8	1.00000	1.00000	0.32271	0.93028	1.00000	1.00000
9	1.00000	1.00000	0.29880	0.93028	1.00000	1.00000
10	1.00000	1.00000	0.32570	0.93028	1.00000	1.00000

Table 3: Learning $P_1 \vee P_2$ Applied to a Uniform Dataset.

Tables 3 and 4 present the precision of six sampling schemes in learning $P_1 \vee P_2$ in ten rounds of relevance feedback. These tables show that MEGA consistently converges to the target concept in the smallest number of iterations. Applied to the Gaussian dataset, MEGA converges after four iterations. The random sampling scheme requires on average two more iterations to converge. The performance of the bounded random scheme and the performance of the aggressive scheme fall between that of the random scheme and that of MEGA. On the aggressive scheme, which attempts to remove as many terms as possible, the chosen samples are less likely to be labeled positive and hence make less of a contribution to the progress of learning the QCS. It is notable that B-Random and aggressive schemes achieve close performance as MEGA. This is because the query concept is still a very simple one, such that most schemes (even random sampling scheme) will be able to learn the concept well. We will show shortly (in Figure 7) that the gaps in performance between MEGA and the other schemes increase as the target concept becomes more complex.

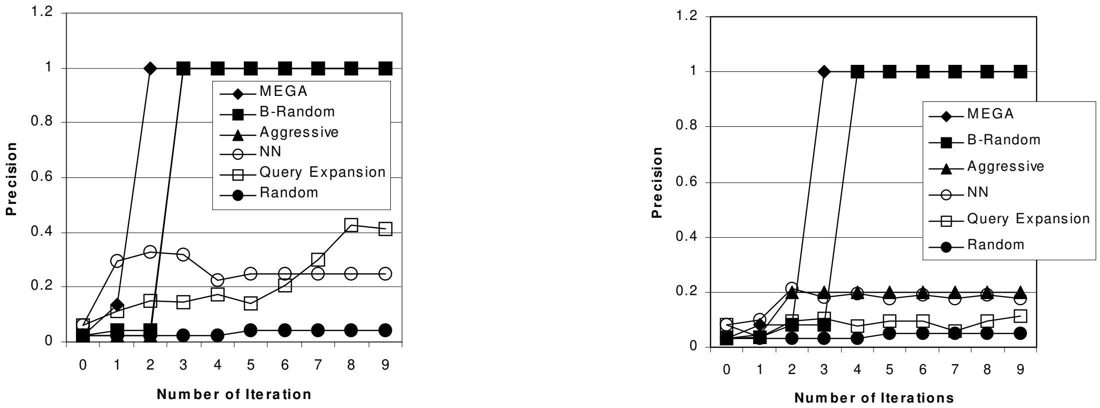
Rnd #	*Random*	*B-Random*	*N-Neighbor*	*Q-Expansion*	*Aggressive*	*Algorithm MEGA*
1	0.08236	0.08236	0.29970	0.29970	0.08236	0.08236
2	0.22178	0.22178	0.65722	0.46684	0.36241	0.32438
3	0.37332	0.37332	0.64907	0.47027	0.80584	0.65982
4	0.38200	0.51249	0.64134	0.46598	0.80584	1.00000
5	0.51249	1.00000	0.63941	0.66237	0.80584	1.00000
6	1.00000	1.00000	0.62782	0.46491	0.80584	1.00000
7	1.00000	1.00000	0.61000	0.47135	0.80584	1.00000
8	1.00000	1.00000	0.61000	0.61258	0.80584	1.00000
9	1.00000	1.00000	0.61000	0.48830	0.80584	1.00000
10	1.00000	1.00000	0.61000	0.64198	0.80584	1.00000

Table 4: Learning $P_1 \vee P_2$ Applied to a Gaussian Dataset.

The results of all datasets and all subsequent tests show that both the nearest neighbor and the query expansion schemes converge very slowly. The result is consistent with that reported in [27, 31], which shows that the query expansion approach does better than the nearest neighbor approach but both suffer from slow convergence. Sampling in the nearest neighborhood tends to result in low precision/recall if the initial query samples are not perfect.

The precision at a given recall achieved by the experiments applied to the Gaussian dataset is lower than that of the experiments applied to the uniform dataset. This occurs because when an initial query point falls outside of, say, two times the standard deviation, we may not find enough positive examples in the unlabeled pool to eliminate all superfluous disjunctions. Since this situation is rare, the negative effect on the average precision/recall is insignificant. The performance gaps between the six sampling schemes were similar when we applied them to the two datasets; therefore, we report only the results of the experiments on the uniform dataset in the remainder of this section.

Figure 7 depicts the results of the second and third tests on the uniform dataset. The figure shows that MEGA outperforms the other scheme (in precision at a fixed recall) by much wider margins. It takes MEGA only three iterations to learn these concepts, whereas the other schemes progress more slowly. Schemes like nearest-neighbor and query expansion fail miserably because they suffer from severe model bias. Furthermore, they cannot eliminate irrelevant features quickly.



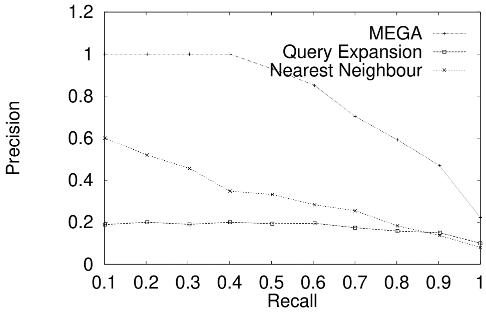
(a) $(P_1 \vee P_2) \wedge P_3$ (b) *Complex Query*

Figure 7: Precision of Six Schemes at Recall = 50%.

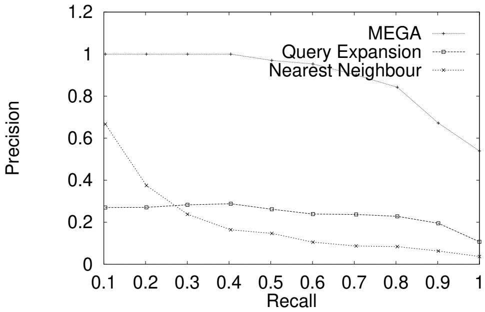
5.2 Model Bias Test

We have shown that MEGA outperforms the other five sampling schemes significantly when the target query concept is in k-CNF. We now present test cases that favor a convex concept, which can be expressed as a linear weighted sum of features to examine how MEGA performs. The target concept we tested is in the form of $\alpha P_1 + (1-\alpha)P_2$, where the value of α is between zero and one.

In this set of tests, we compared MEGA with the nearest neighbor scheme and the query expansion scheme, which are the representative schemes designed for refining convex concepts.



(a) $\alpha = 0.3$



(b) $\alpha = 0.5$

Figure 8: Recall vs. Precision (Model Bias Test).

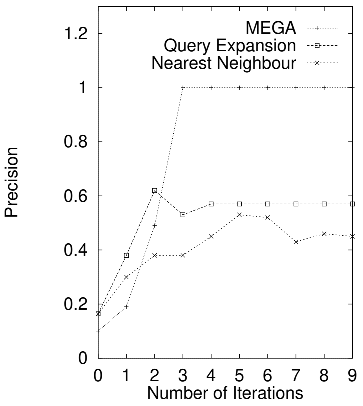
We started by picking 20 random images to see how fast each scheme would converge to the target concepts. Again, we repeated each experiment ten times and recorded each scheme's average precision and recall.

We tested six convex concepts by setting $\alpha = 0, 0.1, \ldots, 0.5$. Below, we report the precision/recall of the three learning methods on two concepts: $0.3P_1 + 0.7P_2$ ($\alpha = 0.3$) and $0.5P_1 + 0.5P_2$ ($\alpha = 0.5$). Setting α in this range makes MEGA suffer from model bias. (We will discuss the reasons shortly.) Figure 8 presents the precision/recall of the three schemes for learning these two concepts after three user iterations. Even though MEGA is not modeled after a convex concept, the performance curve of MEGA exceeds that of the other two schemes in learning both concepts, which indicates MEGA is a faster learner than other two schemes.

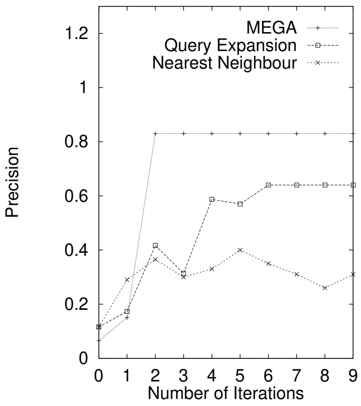
To understand the reasons why MEGA works better than the nearest neighbor and query expansion schemes and how each scheme improves from one iteration to another, we present a set of charts where we fix recall at 0.5 and examine the trend of precision with respect to the number of iterations. (The trend at other recall levels is similar.) Figure 9(a) shows the result of learning concept P_2 (setting $\alpha = 0$). MEGA does very well in this experiment, since it suffers no model bias. Neither the nearest neighbor nor the query expansion scheme does as well because they are slow in refining query concepts and reducing irrelevant features.

What if a user does have a weighted linear query-concept in mind? Even so, MEGA can approximate this model fairly well. Figures 9(b), (c), (d), (e), and (f) all show that MEGA achieves higher precision faster than either the nearest neighbor or the query expansion scheme under all α settings. We summarize our observations as follows:

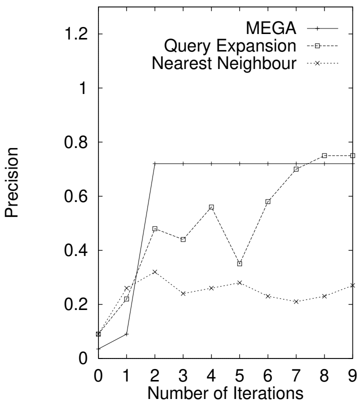
1. When $\alpha = 0$ (or 1), the concept has only one predicate and MEGA has better precision by a wide margin than the two traditional schemes, since it can converge much faster. Even when α is near 0 or 1, the precision of MEGA decreases slightly but still outperforms the traditional schemes, as shown in Figure 9(b). This is because although MEGA suffers slightly from model bias, its fast convergence makes it a better choice when the number of



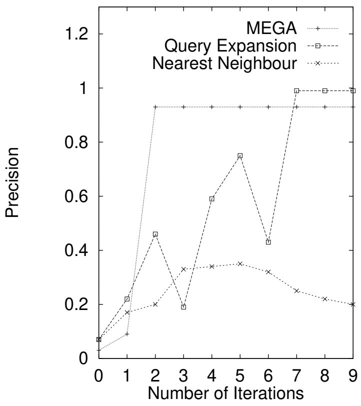
(a) $\alpha = 0(1.0)$



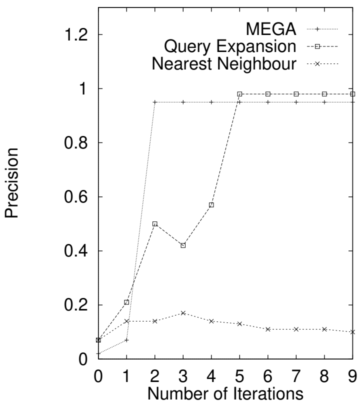
(b) $\alpha = 0.1(0.9)$



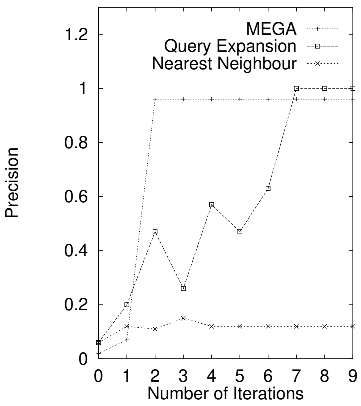
(c) $\alpha = 0.2(0.8)$



(d) $\alpha = 0.3(0.7)$



(e) $\alpha = 0.4(0.6)$



(f) $\alpha = 0.5$

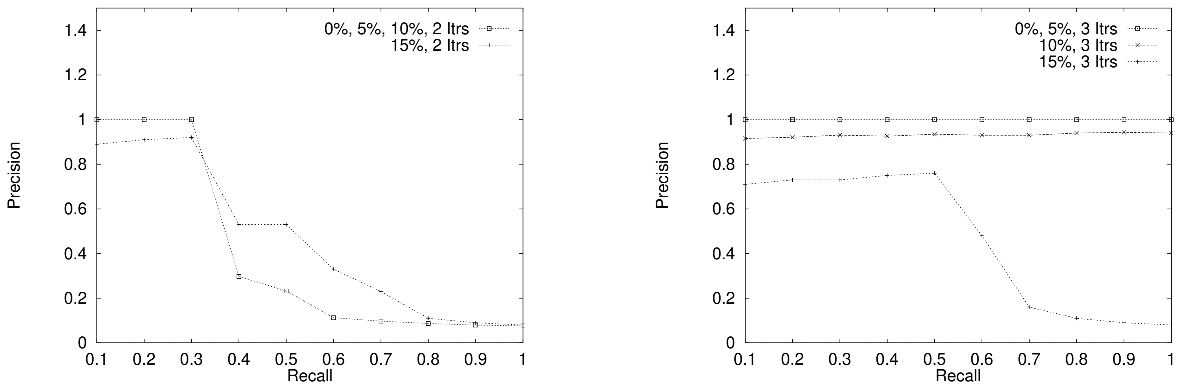
Figure 9: The Effect of Different α's.

iterations is relatively small.

2. When $\alpha = 0.5$, MEGA can approximate the convex concept by $P_1 \wedge P_2$. Figures 9(e) and (f) show that when α is near 0.5, MEGA trails the query expansion by only a slim margin after five/six user iterations. Although the query expansion scheme eventually converges to the target concept, MEGA's fast improvement in precision in just a couple of iterations makes it more attractive, even though slower learning schemes might eventually achieve a slightly higher precision.

3. Figures 9(c) through (e) show that when α is between 0.2 and 0.4, MEGA suffers from model bias and its achievable precision can be low. However, our primary concern is with the range between three and five iterations that will probably conform to the patience of online users. For this purpose, MEGA is more attractive even with its model bias. When $\alpha = 0.2$, MEGA reaches 70% precision after two iterations whereas the query expansion scheme requires seven iterations to reach the same precision.

5.3 User Error Test



(a) *Two Iterations* (b) *Three Iterations*

Figure 10: Precision/Recall Under 0%, 5%, 10% and 15% Noise.

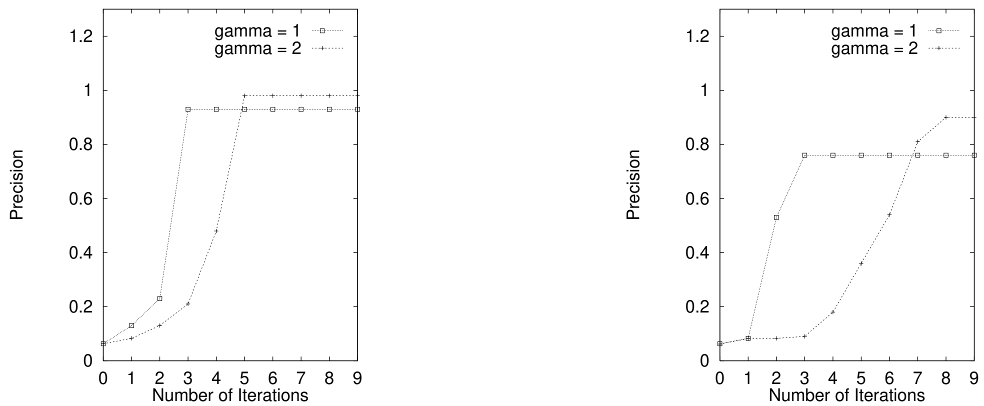
In this experiment MEGA learned the $(P_1 \vee P_2) \wedge (P_3 \vee P_4)$ concept under three different error rates, 5%, 10% and 15%. (A five percent error rate means that one out of twenty samples is mislabeled.) We also used two different γ settings (one and two) to examine the tradeoff between learning speed and accuracy. Figure 10 presents the precision/recall after two or three user iterations under different error rates. MEGA enjoys little to no performance degradation when the noise rates are less than or equal to 10%. When the error rate is 15%, MEGA's search accuracy starts to deteriorate. This experiment shows that MEGA is able to tolerate mild user errors.

Next, we fixed recall at 50% and examined how different error rates and γ settings affect learning precision. Figure 11(a) shows that under both $\gamma = 1$ and $\gamma = 2$ settings, MEGA reaches high precision. However, MEGA's precision improves much faster when $\gamma = 1$ than when $\gamma = 2$. This result does not surprise us, since a lower γ value eliminates terms more aggressively and hence leads to faster convergence. When the noise level is high (15%), Figure 11(b) shows that a low γ setting hinders accurate learning of the target concept. This is because MEGA eliminates terms too aggressively, and the high noise level causes it to eliminate some terms wrongfully. But if we set $\gamma = 2$, we can learn the concept with higher accuracy by slowing down the learning pace. This experiment shows a clear tradeoff between learning accuracy and convergence speed. When the noise level is low, it is preferable to use a less strict voting scheme (i.e., setting a smaller γ) for achieving faster convergence. When the noise level is high, a stricter voting scheme (i.e., using a larger γ) will better maintain high accuracy.

5.4 The Effects of Sample Size K_α

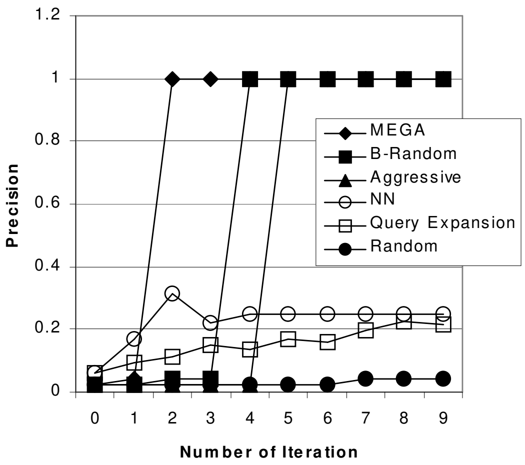
We have so far reported the results of using $K_\alpha = 20$ as the sample size. Intuition tells us that the larger the sample, the faster the learning speed. However, using a large sample may not be realistic. For image retrieval, providing 20 or 30 samples at a time to solicit user feedback is reasonable. For text and music searches, however, the desired sample size tends to be smaller.

We fixed the recall at 50% and listed the precision achieved for learning $(P_1 \vee P_2) \wedge P_3$ by the six schemes under two different sample sizes, 20 and 10, in Figures 12(a) and 12(b). When K_α is cut in half, from 20 to 10, MEGA still converges to the target concept in just two iterations, while other schemes either take more iterations to converge to the target concept, or achieve a lower precision. Figures 12(c) and 12(d) show the effect of the sample size on the convergence rate of the query expansion scheme and MEGA, respectively, for different sample sizes and different numbers of iterations. We can see that the performance of MEGA is relatively less sensitive to the

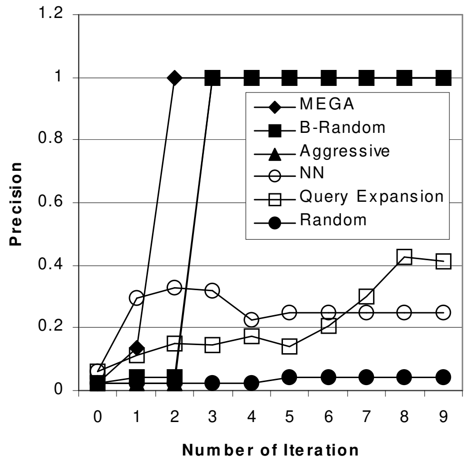


(a) 10% (b) 15%

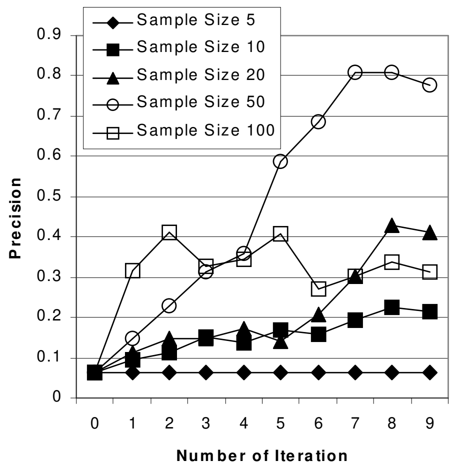
Figure 11: Effects of Noise.



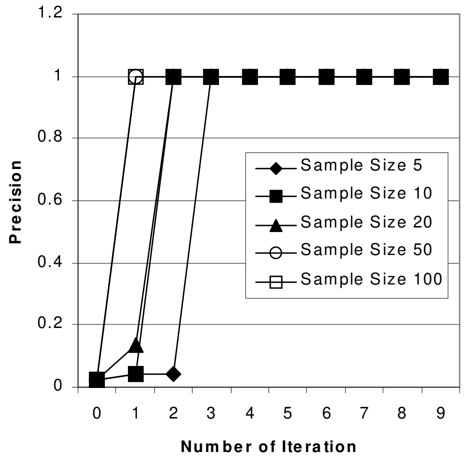
(a) $K_\alpha = 10$



(b) $K_\alpha = 20$



(c) *Query Expansion*



(d) *MEGA*

Figure 12: The Effect of Sample Size.

sample size. Even when the sample size is 5, MEGA needs only one more iteration to converge to the target concept. These results reveal that the quality of the samples may be more critical than the quantity of samples.

5.5 Observations

We can summarize the above experimental results as follows:

1. Convergence speed: MEGA converges much faster than the other schemes in all cases.

2. Model accuracy: MEGA outperforms the other schemes by a wide margin when the target query concept is in k-CNF. When a user's query concept is a weighted linear function,

MEGA will suffer from model bias. However, MEGA can approximate certain kinds of non-CNF concepts fairly well. The fact that MEGA can achieve a high convergence ratio in a small number of iterations makes it an attractive online learning scheme.

3. Noise tolerance: MEGA does well under noise conditions, including model bias and user errors.

6 Empirical Study on An Image Database

In this section we report our experimental results on image data. The main goals of our experiments were to

1: Confirm that MEGA works both effectively and efficiently on a realistic high-dimensional dataset.

2: Study the tradeoff between learning time and learning accuracy (Sections 6.1 and 6.2).

3: Find ways to group features so that high retrieval precision can be achieved within a few iterations (Section 6.3).

We conducted experiments on a $51,000$-image dataset collected from Corel image CDs and the Internet. Each image was characterized by a 144-dimension feature vector, which contained information about color histograms, color moments, textures, etc. (These features are highlighted in Table 5, and discussed in [7, 35].)

	Color Features (features 1 *to* 108*)*	*Texture Features (features* 109 *to* 144*)*
Channels	Twelve color channels: black, gray, white, red, yellow, green, blue, brown, purple , pink, orange and outlier color.	Nine texture channels: vertical coarse texture, horizontal coarse texture, diagonal coarse texture, vertical medium texture, horizontal medium texture, diagonal medium texture, vertical fine texture, horizontal fine texture and diagonal fine texture.
Detail Information	Each color channel has nine features: histograms, spreadness, elongation, & color average and standard deviation in hue, saturation and intensity.	Each texture channel has four features: total energy, energy variance, texture spreadness and texture elongation.

Table 5: Feature Description.

Each of the query concepts we tested belongs to one of the eighteen image categories: *architecture*, *bears*, *big cities*, *clouds*, *elephants*, *fabrics*, *fireworks*, *flowers*, *food*, *landscapes*, *people*, *objectionable images*, *stars*, *textures*, *tigers*, *tools*, and *waves*. Many categories were added to make the learning task challenging. For instance, the *big cities* and *architecture* categories both have buildings. Three animal categories —*bears*, *elephants*, and *tigers*— all have landscape background which makes the discernment of these categories difficult. The *fireworks* and *stars* categories both share the same dark sky background. *Clouds* and *waves* can appear very similar to each other,

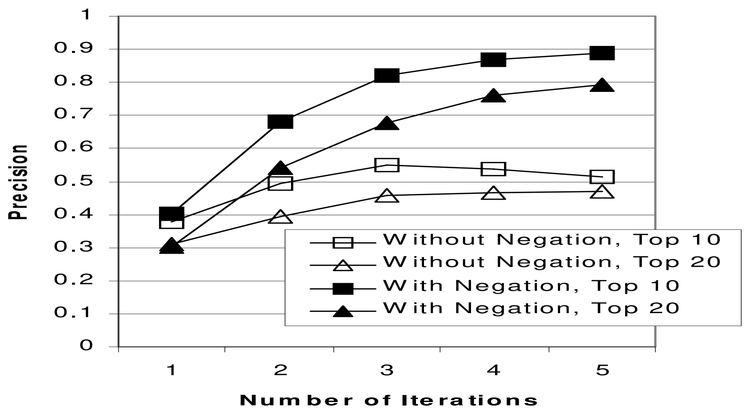


Figure 13: The Effect of Negations.

as can *flowers* and *fabrics*. Finally, various texture images and randomly crawled Internet images were added to raise the learning difficuly for all categories.

We used *precisions* of the top-10 and top-20 retrievals to measure performance. Since in the online image query scenario, 10 and 20 are practical numbers of images to present to a user, and the ratio of these images fitting the user's target concept is of major concern. We tallied precision for up to only five iterations, since we deemed it unrealistic to expect an interactive user to conduct more than five rounds of relevance feedback. To be objective, we ran each experiment ten times.

6.1 The Effect of Negations in Boolean Formulae

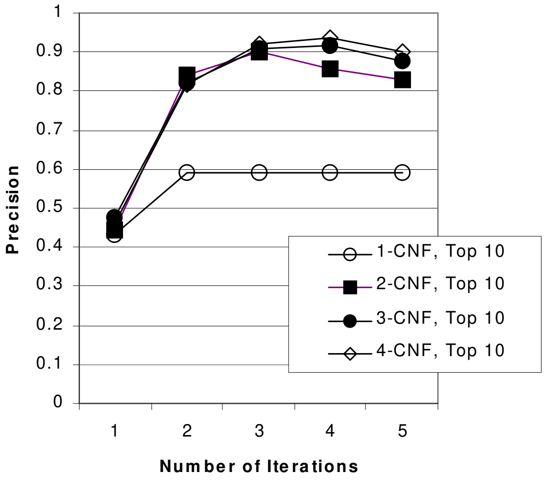
As previously stated, each of our images was characterized by a 144-dimension vector. Each feature has a corresponding predicate. We further expanded the method to include negations of these predicates. In other words, the Boolean variables in a Boolean formula not only can be of form P_{x_i}'s, but also can be their negations, ($P_{\neg x_i}$'s). We first ran experiments without including negations of predicates and then included them with all other conditions fixed. The average top-10 and top-20 retrieval precisions are compared in Figure 13.

The figure shows that the precisions of both top-10 and top-20 retrievals are significantly increased when negations are included. This is because many query concepts can be expressed only with negations. For instance, an *elephant* image usually does not involve red color. The negation of the predicate "red high" can be useful for distinguishing an elephant image from, say, a flower image that may satisfy the "red high" predicate. Since including negations can substantially improve retrieval precision, we employ negations in our system.

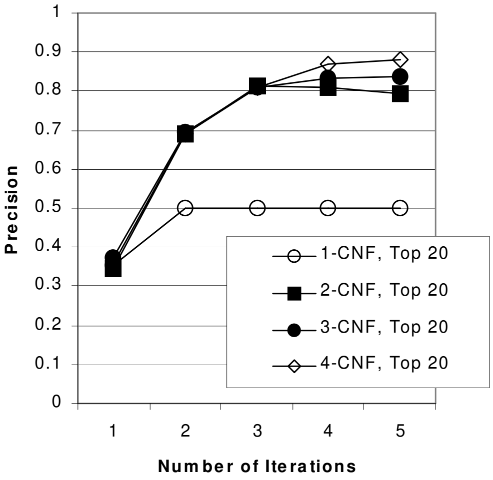
6.2 The Effect of k and G

We model query concepts as k-CNF. The larger the value of k, the more complex concepts we can model. However, learning a more complex concept requires higher space and computational complexity. We should find a k that can adequately capture the query concepts for our applications, and can also be learned in a reasonable amount of space and time.

We experimented with different values of k, from $k = 1$ to $k = 4$. Figure 14 shows the average precisions of top-10 and top-20 retrievals. Figure 15 shows the running time for employing different k's. (We set $G = 20$ in this experiment. We explain shortly how the value of G is selected.)



(a) Top-10 Retrieval



(b) Top-20 Retrieval

Figure 14: The Effect of k on Precision

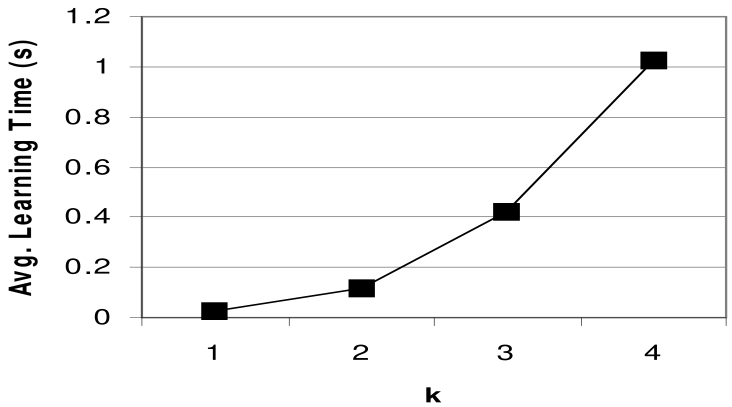
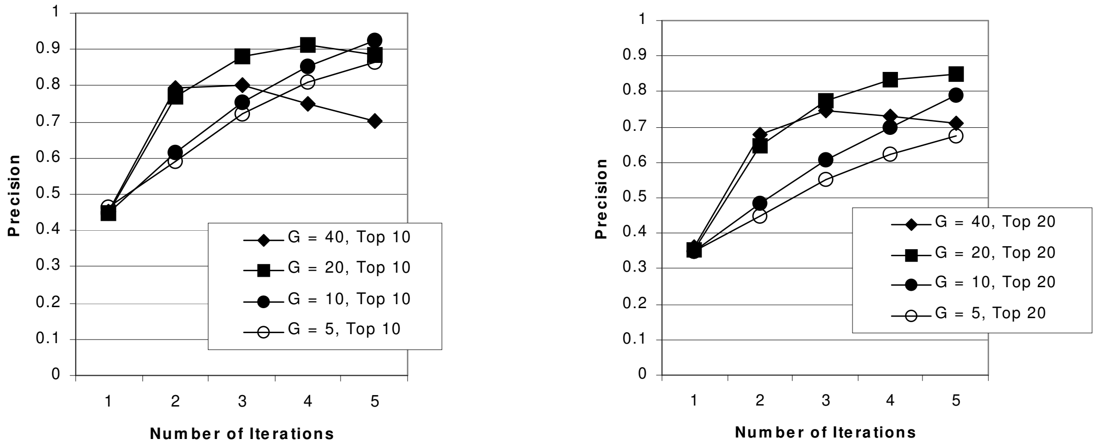


Figure 15: The Effect of k on Learning Time.

Since the trends of the top-10 and top-20 retrievals are similar, we discuss only the top-10 results. For 1-CNF, although the learning time is very short, the achieved search precision is under 60%. After we increased k to two, precision reached 90% after just three iterations. The learning time increased to about 0.1 second, which is satisfactory. When we further increased k to three, the precision improved, and the learning time (0.4 second) remained low. Further increasing k to four would not be productive, since the precision gain is marginal but the learning time increases substantially. For this image dataset, it is thus sufficient to model query concepts in 3-CNF.

Next, we examine the effect of G. In section 4.6 we discussed dividing features into G groups to speed up learning time. We experimented on different G values: 40, 20, 10 and 5.

Figures 16 and 17 show that tradeoffs exist between running time and search accuracy. When G is large ($G = 40$), the average running time is short (0.2 second), but the achieved precision can be dismal (around 80% for top-10 retrieval). At the other extreme, when we set $G = 5$, the precision is good but the running time is unacceptably high. When features are divided into too many groups, we are left with fewer terms to represent some query concepts. When features are divided into too few groups, the number of terms can be very large and hence would require lengthy learning time. On the other hand, when features are divided into too many groups, some query concepts may not be expressed well since they require disjunction terms of features in different groups to express. Setting G to between 10 and 20 seems to give us a reasonable tradeoff between precision and running time. In the remainder of our experiments, we thus divided features into $G = 20$ groups.



(a) Top-10 Retrieval

(b) Top-20 Retrieval

Figure 16: The Effect of G on Precision.

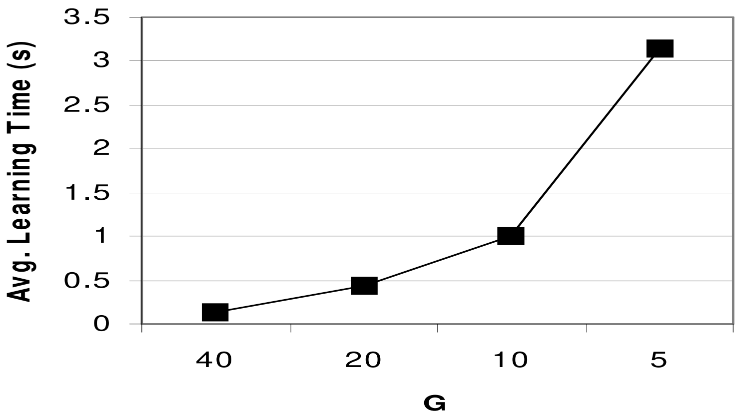


Figure 17: The Effect of G on Learning Time.

6.3 Optimal Grouping Study

Based on the above experimental results, we set G to 20 and model query concepts in 3-CNF. In this subsection, we are interested in finding how features should be divided into $G = 20$ groups.

We used a genetic algorithm [19] to study the pattern of groupings. We started with 40 randomly generated groupings. We ran experiments on each of them. Based on their results (fitness), we generated the next generation of groupings. We ran the experiment up to 120 generations. Figure 18 plots the precision that can be achieved in those 120 generations.

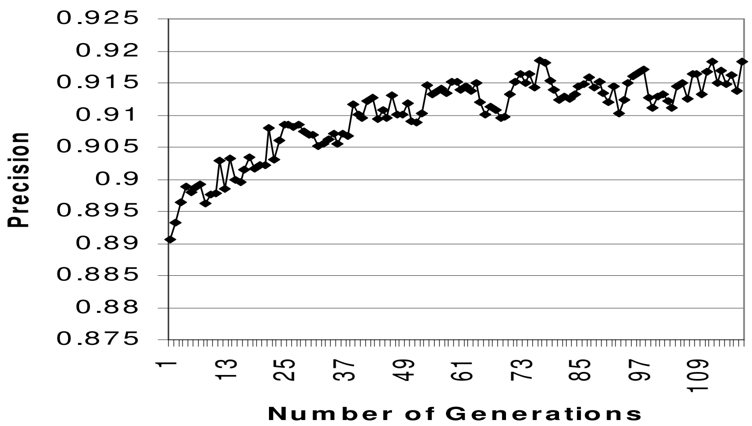


Figure 18: Average of the Highest Precisions for Each Generation.

From Figure 18, we can see that the genetic algorithm increases the precision from 89% to 92%. To understand the factors for achieving that improvement, we also studied the average intra-group feature correlation of each generation, which is presented in Figure 19. From both Figures 18 and 19 we can identify a pattern: that having a low intra-group correlation tends toward producing a higher search precision. To verify that our observation was valid, we conducted

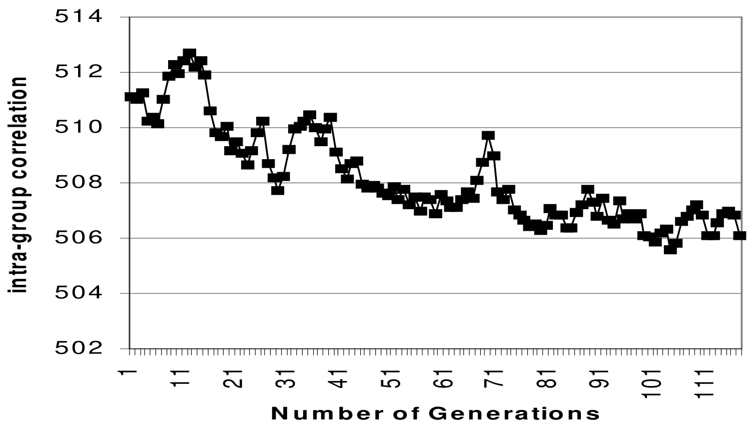


Figure 19: Average of Intra-group correlations for Each Generation.

extensive experiments on precision achieved by different intra-group correlations. Figure 20 shows that the low intra-group correlation grouping can achieve 92% precision, whereas the high intra-group correlation grouping can achieve only 85% precision. These data confirms that the loss of precision can be reduced by organizing a feature space intelligently.

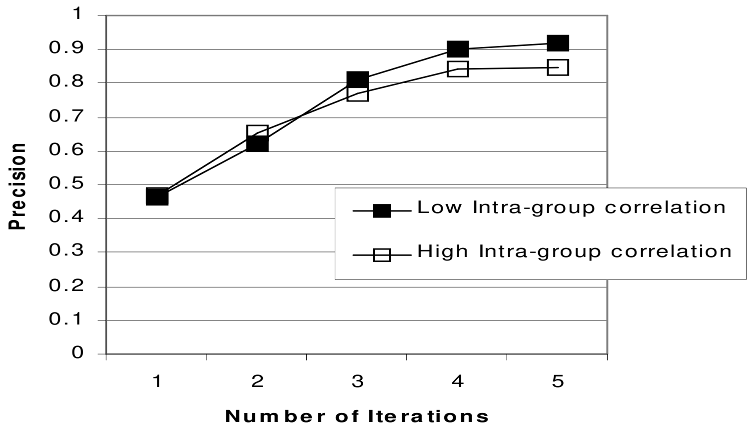


Figure 20: Precision of Top-10 Retrieval for Different Grouping Schemes.

6.4 Observations

We make the following observations from our empirical study on the image dataset:

1. *Including negations.* It is critical to include both predicates and their negations in a Boolean formula. Including negations helps to express complex concepts more accurately and hence enhance search accuracy.

2. *Setting k to achieve good search accuracy and time tradeoff.* For our dataset, it is sufficient to model query concepts in 3-CNF to achieve high precision at low search cost. In general, we need to choose a proper k through empirical study in order to adequately model query concepts while keeping the computational cost low.

3. *Choosing G to achieve good search accuracy and space/time complexity tradeoff.* For our dataset, it is desirable to divide features into 20 groups. We need to choose a proper G to speed up learning time while maintaining high search accuracy.

4. *Organizing features to minimize intra-group correlation for maintaining high search accuracy.* The groupings with smaller intra-group correlation tend to yield higher search accuracy.

7 Conclusion

We have proposed MEGA, a query-concept learner that learns query concepts quickly and with a small number of labeled instances. Instead of returning the "best" results after each round of relevance feedback as traditional methods do, MEGA generates intelligent samples to maximize the collection of relevant information for trimming the candidate concept-space. We have shown that the sample selection and concept refinement steps of MEGA work in a complementary manner to achieve fast convergence.

Our experimental results showed that MEGA outperforms traditional sampling schemes in all query scenarios when the number of labeled instances is limited. In addition, we have shown that MEGA can cope well with moderate model bias and user errors. We have implemented a Perception-Based Image Retrieval (PBIR) system, which employs MEGA as its query-concept learner [5, 6]. Through MEGA, the PBIR system can fuzzily capture a query concept, usually in two to three feedback iterations, and can comprehend a target concept very well in three to five iterations.

To speed up learning and to reduce the learning space requirement, we have proposed a divide-and-conquer method that divides the concept-learning task into G subtasks. We noted that a task must be divided carefully, or search accuracy might suffer. Through analysis and empirical studies, we observed that organizing image features in a multi-resolution manner, and maximizing inter-group feature correlation (or minimizing intra-group correlation), can speed up query-concept learning by $O(G^{k-1})$ folds, while maintaining high search accuracy.

Of course, MEGA has its limitations, and some of them are.

1. The number of required training instances to learn a query concept still seems to be large for some application scenarios. Although MEGA has substantially reduced the number of

labeled instances to learn a target concept, it can still require a moderate number of labeled instances to infer a concept accurately. For example, in our 144-dimensional image dataset, MEGA needs about 60 labeled instances to capture an image query concept. For image retrieval where relevance feedback is convenient to conduct, MEGA is attractive. For other applications such as document and music searches, the number of required training instances needs to be further reduced to make relevance feedback more practical.

2. Although MEGA does not require relevant objects to start a query, the number of user iterations still depends on the percentage of the relevant objects in the database. When the percentage is very low ($< 0.1\%$), it can take MEGA more than 10 user iterations to find one relevant example.

3. MEGA will not be able to capture a query concept accurately when extracted features cannot adequately represent the concept.

Although MEGA is not the silver bullet that would solve all the above problems entirely, our prototype [5, 6] shows that MEGA can be practically implemented and can learn user's query concepts more effectively and efficiently than traditional schemes. Meanwhile, we are exploring the possibility of combining the strengths of MEGA and other techniques (such as SVM_{Active} [35] and FeedbackBypass [2]) to learn the query concepts even faster and more accurately. More specifically, we are exploring the following three ways to further improve MEGA's effectiveness and efficiency:

1. Combining the strength of MEGA with that of SVM_{Active}. We plan to use MEGA to narrow down the version space quickly and then switch to SVM_{Active} to learn query concept precisely. Also, MEGA and SVM_{Active} can share positive- and negative-labeled instances with one another to speed up the learning progress.

2. Using past information to make more accurate predications in selecting samples. We will investigate if feedback history can be used to help trim the uncertain version space more aggressively, and hence reduce the number of feedback iterations.

3. Supporting multimodal retrieval. We believe when the relevant objects are rare, using keywords to find some relevant instances quickly can provide the learning task a good bootstrapping. We plan to investigate models that can integrate text with image features effectively.

References

[1] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: An interactive approach to decision tree construction. In *Knowledge Discovery and Data Mining*, pages 392–396, 1999.

[2] I. Bartolini, P. Ciaccia, and F. Waas. Feedbackbypass: A new approach to interactive similarity query processing. *The 27^{th} VLDB Conference*, September 2001.
[3] L. Breiman. Bagging predicators. *Machine Learning*, 24:123–140, 1996.
[4] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–849, 1998.
[5] E. Chang, K.-T. Cheng, and L. Chang. PBIR — perception-based image retrieval. *ACM Sigmod (Demo)*, May 2001.
[6] E. Chang, K.-T. Cheng, W.-C. Lai, C.-T. Wu, C.-W. Chang, and Y.-L. Wu. PBIR — a system that learns subjective image query concepts. *Proceedings of ACM Multimedia, http://www.mmdb.ece.ucsb.edu/~demo/corelacm/*, pages 611–614, October 2001.
[7] E. Chang, B. Li, and C. Li. Towards perception-based image retrieval. *IEEE Content-Based Access of Image and Video Libraries*, pages 101–105, June 2000.
[8] C. Chen. *Fuzzy Logic and Neural Network Handbook*. McGraw-Hill, 1996.
[9] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The bayesian image retrieval system, pichunter: Theory, implementation and psychological experiments. *IEEE Transaction on Image Processing (to appear)*, 2000.
[10] R. Fagin. Fuzzy queries in multimedia database systems. *ACM Sigacr-Sigmod-Sigart Symposium on Principles of Database Systems*, 1998.
[11] R. Fagin and E. L. Wimmers. A formula for incorporating weights into scoring rules. *International Conference on Database Theory*, pages 247–261, 1997.
[12] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information System: Integerating Artificial Intelligence and Database Technologies*, 3(3-4), 1994.
[13] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.
[14] K. Fukanaga. *Introduction to Statistical Pattern Recognition*. Boston: Academic Press, 1990.
[15] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. *VLDB*, 1998.
[16] K. S. Jones and P. W. (Editors). *Readings in Information Retrieval*. Morgan Kaufman, July 1997.
[17] M. Kearns, M. Li, and L. Valiant. Learning boolean formulae. *Journal of ACM*, 41(6):1298–1328, 1994.
[18] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
[19] B. Li, W.-C. Lai, E. Chang, and K.-T. Cheng. Mining image features for efficient query processing. *Proceedings of the First IEEE Data Mining Conference, San Jose*, 2001.
[20] C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold. Clustering for approximate similarity queries in high-dimensional spaces. *IEEE Transaction on Knowledge and Data Engineering (to appear)*, 2001.
[21] J. Li, J. Z. Wang, and G. Wiederhold. Irm: Integrated region matching for image retrieval. *ACM Multimedia*, 2001.
[22] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
[23] A. Natsev, R. Rastogi, and K. Shim. Walrus: A similarity retrieval algorithm for image databases. *SIGMOD*, 1999.
[24] M. Ortega, Y. Rui, K. Chakrabarti, A. Warshavsky, S. Mehrotra, and T. S. Huang. Supporting ranked boolean similarity queries in mars. *IEEE Transaction on Knowledge and Data Engineering*, 10(6):905–925, December 1999.
[25] P. Perner, U. Zscherpel, and C. Jacobsen. A comparision between neural networks and decision trees based on data from industrial radiographic testing. *Pattern Recognition Letters*, 22:47–54, 2001.
[26] K. Porkaew, K. Chakrabarti, and S. Mehrotra. Query refinement for multimedia similarity retrieval in mars. *Proceedings of ACM Multimedia*, November 1999.
[27] K. Porkaew, S. Mehrota, and M. Ortega. Query reformulation for content based multimedia retrieval in mars. *ICMCS*, pages 747–751, 1999.
[28] P. Raghu, R. Poongodi, and B. Yegnanarayana. Unsupervised texture classification using vector quantization and deterministic relaxation neural network. *IEEE Transactions on Image Processing*, 6(10):1376–1387, 1997.
[29] J. A. Robinson and A. Voronkov. *Handbook of Automated Reasoning*. Elsevier Science Publishers, 2000.

[30] Y. Rui and T. Huang. Optimizing learning in image retrieval. *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 236–243, 2000.

[31] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool in interactive content-based image retrieval. *IEEE Tran on Circuits and Systems for Video Technology*, 8(5), Sept 1998.

[32] R. Schapire. Theoretical views of boosting and applications. *In Tenth International Conference on Algorithmic Learning Theory*, 1999.

[33] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceeding of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann, 1997.

[34] J. Smith and S.-F. Chang. An image and video search engine for the world-wide web. *Storage and Retrieval for Image and Video Databases V, Proc SPIE 3022*, 1997.

[35] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *Technical Report http://www-db.stanford.edu/∼echang/svmactive.ps*, March 2001.

[36] L. Valiant. A theory of learnable. *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 436–445, 1984.

[37] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[38] L. Wu, C. Faloutsos, K. Sycara, and T. R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. *The* 26^{th} *VLDB Conference*, September 2000.

[39] L. A. Zadeh. Fuzzy sets. *Information and Control*, pages 338–353, 1965.

[40] S. Zemke. Bagging imperfect predictors. *Proceedings of Artificial Neural Networks in Engineering*, pages 1067–1072, 1999.