An Active Learning Framework for Content Based Information Retrieval

Cha Zhang and Tsuhan Chen
Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213
Email: {czhang, tsuhan}@andrew.cmu.edu

Abstract

In this paper, we propose a general active learning framework for content-based information retrieval. We use this framework to guide hidden annotations in order to improve the retrieval performance. For each object in the database, we maintain a list of probabilities, each indicating the probability of this object having one of the attributes. During training, the learning algorithm samples objects in the database and presents them to the annotator to assign attributes to. For each sampled object, each probability is set to be one or zero depending on whether or not the corresponding attribute is assigned by the annotator. For objects that have not been annotated, the learning algorithm estimates their probabilities with biased kernel regression. *Knowledge gain* is then defined to determine, among the objects that have not been annotated, which one the system is the most uncertain of. The system then presents it as the next sample to the annotator to assign attributes to. During retrieval, the list of probabilities works as a feature vector for us to calculate the semantic distance between two objects, or between the user query and an object in the database. The overall distance between two objects is determined by a weighted sum of the semantic distance and the low-level feature distance. The algorithm is tested on both synthetic databases and real databases of 3D models. In both cases the retrieval performance of the system

. Contact Author: Prof. Tsuhan Chen, Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213. Tel: (412) 268-7536, Fax: (412) 268-3890, Email: tsuhan@cmu.edu
. Work supported in part by NSF CAREER Award 9984858.
. Accepted by *IEEE Trans. on Multimedia, Special Issue on Multimedia Database*.

improves rapidly with the number of annotated samples. Furthermore, we show that active learning outperforms learning based on random sampling.

Key words: active learning, attribute tree, content-based information retrieval, biased kernel regression, semantics, 3D model retrieval.

I. Introduction

Content-based information retrieval (CBIR) has attracted a lot of research interest in recent years. A typical CBIR system, e.g., an image retrieval system, includes three major aspects: feature extraction, high dimensional indexing, and system design [1]. Among the three aspects, feature extraction is the basis of content-based information retrieval. However, features we can extract from the data are often low-level features. As a result, two semantically similar objects may lie far from each other in the feature space, while two completely different objects may stay close to each other. Although many features have been designed for general or specific CBIR systems, and some of them showed good retrieval performance, the gap between low-level features and high-level semantic meanings of the objects has been the major obstacle to more successful retrieval systems.

Relevance feedback and hidden annotation have been shown to be two of the most powerful tools for bridging the gap between low-level features and high-level semantics. Widely used in text retrieval [2], relevance feedback was first proposed by Rui *et al.* as an interactive tool in content-based image retrieval [3]. Since then it has been proven to be a powerful tool and has become a major focus of research in this area [4][5][6][7]. Relevance feedback moves the query point towards the relevant objects or selectively weighs the features in the low-level feature space based on user feedback. However, if the low-level features of a set of semantically similar objects lie in the space as several clusters, querying with an object in one cluster would not be able to retrieve semantically similar objects in other clusters by reweighing the space. In [9] and [10], similar approaches were proposed to use relevance feedback to build semantic relationships inside the database. Their systems grouped the objects in the database into small semantic clusters and related the clusters with semantic weights. The updating of the clusters and semantic weights are based on the user's feedback. Another solution to the above problem is hidden annotation. By attaching Boolean attributes to images in the database, Cox *et al.* did some experiments on hidden annotation in their Bayesian image retrieval system, *PicHunter*, and showed positive results [8]. In this paper, we study the hidden annotation as a preprocessing stage of a retrieval system, referred to as the learning stage, before any user can use the system.

Most existing systems using hidden annotation either annotate all the objects in the database (full annotation), or annotate a subset of the database manually selected (partial annotation). As the database becomes larger, full annotation is increasingly difficult because of the manual effort involved. Partial annotation is relatively affordable and trims down the heavy manual labor. Once the database is partially annotated, traditional pattern classification methods are often used to derive semantics of the objects not yet annotated. However, it is not clear how much annotation is sufficient for a specific database, and what the best subset of objects to annotate is. In this paper, we use active learning to determine which objects should be annotated. During the learning stage, the system provides sample objects automatically to the annotator. The sample objects are selected based on how much information annotation of each sample object can provide to decrease the *uncertainty* of the system. The object, once annotated, giving the maximum information or knowledge gain to the system is selected. In machine learning literature, the idea of maximizing the expected information from a query has been studied under the name "active learning" or "learning with queries" [12]. It was revisited by Cox *et al.* when they updated the display of the query result in [8]. We will present a more detailed survey of the active learning literature in Section II-B.

The key assumption we make throughout this paper is that, although the low-level feature space cannot describe the semantic meaning, it is *locallay inferable*. This means that in the low-level feature space, if two objects are very close to each other, they should be semantically similar, or be able to infer some knowledge to each other. Notice that this assumption allows objects with the same semantic meaning to lie in different places in the feature space, which cannot be handled by normal relevance feedback. If the above assumption does not hold, neither relevance feedback nor hidden annotation will be able to help improving the retrieval performance, even if the database is fully annotated. The only solution to this circumstance is to find better low-level features for the objects.

We assume that the semantic meanings of the objects in the database can be characterized by a multi-level attribute tree. To make the attribute tree general, the attributes at the same level of the tree are not necessarily exclusive of each other. For each object in the database, we maintain a list of probabilities, each of them indicating the probability of this object having the corresponding attribute. If an object is annotated, the probabilities are set to be one or zero depending on whether the corresponding attributes are annotated to characterize the object or not. For each of the objects that have not been annotated, we estimate its attribute probabilities based on its annotated neighbors. Kernel regression is employed to fulfill this task. With this list of probabilities, we are able to tell which object the system is most uncertain of, and propose it as a sample to the annotator. The list of probabilities also works as a feature vector to calculate the semantic distance between two objects or between a query and an object. The final similarity measurement between any two objects is determined by a weighted sum of the semantic distance and the low-level feature distance. Using both synthetic database and a 3D model database as examples, we show that with our algorithm, the performance of the retrieval system improves rapidly with the number of annotated models, and in all cases outperforms the approach of randomly choosing the objects to annotate.

The paper is organized as follows. In Section II, we introduce the general criterion for active learning in our approach. Section III presents the details of the proposed algorithm. We discuss the joint semantic and low-level feature similarity measurement in Section IV. We show the experimental results in Section V and conclude the paper in Section VI.

II. The General Criterion for Active Learning

A. The Learning Interface and the Attribute Tree Structure

Figure 1 shows the learning/annotation interface of our system. On the left hand side is a list of attributes to be annotated. On the right hand side is a sample object (e.g., an image or a 3D model) the system proposes. The basic operation for the annotator is to check some of the attributes for this sample model and press the "Annotate" button for the system to get the annotation information of the sample model.

In our system, the attributes form a tree structure with multiple levels. In the attribute tree, each node is an attribute. The attributes at higher-level nodes are more general than those at the lower-level nodes. By default we assume that once an attribute at a lower-level nodes is checked, the attributes at the higher-level nodes or its parent nodes are also checked. As a simple example, "Aircraft" lies at the first level that is the highest level in the tree structure, thus it is more general than "Jets", which lies at the second level. An object that is a "Jets" is also an "Aircraft". Unlike the decision tree in classification applications, the nodes with the same parent node in our attribute tree are not necessarily exclusive of each other. For example, an aircraft can be both a "Classic" and a "Jets". This makes our tree structure more general and more natural to use for annotation.

The annotation starts with no attributes in the tree structure. When necessary, the annotator may add, rename or remove any attributes at any level. The annotator is asked to check all the attributes that the sample object has. For an attribute that the annotator does not check, we assume the annotator implies that this object does not have that attribute.

B. Active Learning and the General Criterion to Choose the Samples

For many types of machine learning algorithms, one can find the statistically "optimal" way to select the training data. The pursuing of the "optimal" way by the machine itself was referred to as *active learning*. While in traditional machine learning research, the learner typically works as a passive recipient of the data, active learning enables the learner to use its own ability to collect data. Some representative work on active learning can be found in [22][23][24].

To be more specific, what we are interested is a specific form of active learning, i.e., *selective sampling*. The goal of selective sampling is to reduce the number of training samples that need to be annotated, by examining objects that are not yet annotated and selecting the most informative ones for the annotator. Many approaches have been proposed for selective sampling. In [16], Seung *et al.* proposed an algorithm called *query by committee* (QBC). Query by committee generates a committee of classifiers and the next query is chosen by the principle of maximal disagreement among these classifiers. Freund *et al.* [17] extended the QBC result to a wide range of classifier forms. They gave some theoretical proofs that, under some assumptions, the effect of training on annotated data can be achieved for the cost of obtaining data that are note yet annotated, and labeling only a logarithmic fraction of them. In [18], Nigam and McCallum modified the QBC algorithm by a combination of active learning and the traditional Expectation-Maximization (EM) algorithm. In [21], Muslea *et al.* introduced an algorithm called *co-testing*. It is similar to the QBC algorithm and is designed to apply to problems with redundant views or problems with multiple disjoint sets of attributes (features) that can be used to learn the target attribute. Lewis and Gale [19] described in their paper another approach called *uncertainty sampling*. The idea is to use only one classifier not only tells which class a sample is, but also gives an uncertainty score for each data sample not yet annotated. The next sample is chosen based on which one the classifier has the least confident with. With uncertainty sampling, it was reported [19][20] that the size of the training data could be reduced as much as 500-fold for text classification.

We need to find a general criterion to measure how much information the annotation can provide to the system. Let $O_i, i = 1, 2, ..., N$ be the objects in the database, and $A_k, k = 1, 2, ..., K$ be the K attributes the annotator wants to use for annotation. These attributes form the attribute tree. For each object O_i, we define probability P_{ik} to be the probability that this object has attribute A_k, where $P_{ik} = 1$ if the object O_i has been annotated as having attribute A_k, and $P_{ik} = 0$ if it has been annotated as not having attribute A_k. If the object has not been annotated, P_{ik} is estimated by its neighboring annotated objects, as will be described in Section III-B. In order to derive the expected information gain when we annotate a certain object, we define an uncertainty measurement as follows:

$$U_i = \Psi(P_{i1}, P_{i2}, ..., P_{iK}), \quad i = 1, 2, ..., N \tag{1}$$

where U_i is the uncertainty measurement, $\Psi(\cdot)$ is a function on all the attribute probabilities of object O_i. We want the uncertainty measurement U_i to have the following properties:

1. If object O_i has been annotated, $U_i = 0$;
2. If $P_{ik} = 0.5$, for $k = 1, 2, ..., K$, i.e., we know nothing about the object, $U_i = U_{max}$;
3. Given $P_{ik}, k = 1, 2, ..., K$, if it is uncertain that object O_i has or does not have some attributes, U_i should be large.

Since the third property of U_i is not presented in a strict sense, various functions can be defined to satisfy these properties. For instance, let us assume that $K = 1$. In this case, only one attribute is concerned. The well-known *entropy* is a good uncertainty measurement:

$$U_i = \Psi(P_{i1}) = H(P_{i1}) = -P_{i1} \log P_{i1} - (1 - P_{i1}) \log(1 - P_{i1}). \tag{2}$$

where H represents the entropy function. We will define uncertainty measurement for multiple attributes in Section III-C.

There is another important factor that affects the benefit the annotator can give to the system. It is the distribution of the objects in the low-level feature space. Annotating objects at a high probability region and a low probability region may give the system different amounts of information, which in turn leads to different retrieval performance. Therefore, we define the *knowledge gain* the annotator can give to the system by annotating object O_i as:

$$G_i = q_i \cdot U_i = q_i \cdot \Psi(P_{i1}, P_{i2}, ..., P_{iK}), \qquad i = 1, 2, ..., N. \tag{3}$$

where G_i is the defined *knowledge gain*; q_i is the *probability density function* around object O_i, which will be estimated in section III-A; U_i is the uncertainty measurement defined in (1). The criterion of choosing the next sample object is to find the unlabeled object O_i that has the maximum knowledge gain G_i.

III. The Proposed Approach

The proposed approach is as follows. We first initialize the probability lists with prior probabilities that we have about the whole database. The probability density function is also estimated. A small number of objects are randomly chosen and annotated as the initialization step of the algorithm. The probability list is re-calculated based on the randomly annotated objects. The system then start to select the object that has the maximum knowledge gain, and ask the annotator to annotate it. After that some of the objects in the database update their probability lists because one of their neighbors is newly annotated. The system then searches for the object that has the maximum knowledge gain again, and asks the annotator to annotate it. This loop keeps going until the annotator stops or the database is fully annotated.

A. Estimate the Probability Density Function

The probability density function is one of the important factors in the defined *knowledge gain* in Equation (3). In machine learning literature, there have been many efficient ways for density estimation, such as the Naïve density estimator, the Bayesian networks, the mixture models, the density trees [28] and the kernel density estimator [27] (also known as Parzen windows). We use the kernel density estimator. The kernel method is also used in updating the probability lists in the next subsection. Since the probability density estimation is independent of the latter probability list updating and needs only to be calculated once offline before the annotation, any of the above algorithms can be employed.

We choose the kernel as an isotropic Gaussian function (assume the features has been normalized). The window of the estimation is a hyper-sphere centered at the concerned object O_i. Let the radius of the super-sphere be r_i, which was named the *bandwidth* of the kernel density estimator in the literature. Normally, $r_i = r, \text{for } i = 1,2,\cdots,N$, where r is a constant bandwidth. Let $\mathbf{x}_i$ be the feature vector of object O_i. The density estimation at the position where O_i locates is given by:

$$q_i = \sum_{j=1}^{N} \text{kernel}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sum_{j=1}^{N} \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2}{2r_j^2}\right), \quad \text{for } i = 1, 2, ..., N. \tag{4}$$

where $\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2$ is the Euclidian distance between the neighboring object O_j and the center object O_i.

The choice of the bandwidth r has an important effect on the estimated probabilities. If the size of the neighborhood is too large, the estimation will suffer from low resolution. On the other hand, a too small size may cause local overfitting, which hurts the generalization ability of the estimation. The optimal Parzen window size has been studied extensively in the literature. The optimal bandwidth can be determined by minimizing the *Integrated Squared Error* (ISE) or the *Mean Integrated Squared Error* (MISE) [29]. Adaptive bandwidth was also proposed in the literature [30]. For simplicity, we choose a constant bandwidth r based on the maximum distance from any object to its closest neighbor, D, i.e.,

$$r = lD = l \max_{\mathbf{x}_k}\left[\min_{\mathbf{x}_l}\left(\left\|\mathbf{x}_k - \mathbf{x}_l\right\|_2\right)\right] \tag{5}$$

where l is a scalar. Through experiments we find that with a well-normalized feature space, selecting l between 1 and 10 often gives good results. Detailed experiments will be shown in Section V.

B. Update the Probability Lists

We assume that we have some prior knowledge about the probability lists before the annotation. That is,

$$P_{ik} = P^{(k)}, \qquad \text{for } i = 1, 2, ..., N, k = 1, 2, ..., K. \tag{6}$$

where $P^{(k)}$ is the prior probability for an object to have attribute A_k. Experimental results show that the guess of the prior probability will not influence the annotation efficiency too much. During the annotation, the annotator is supposed to check all the attributes the query model has, and all the other attributes the annotator does not check are assumed to be not belonging to the object unless they have some of their children nodes checked. Let Θ_i be the set of attributes the annotator annotated for object O_i, including those having children nodes checked. The new list of probabilities for object O_i after the annotation is:

$$P_{ik} = \begin{cases} 1, & \text{if } A_k \in \Theta_i, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Recall the basic assumption we made in Section I, annotated models tend to infer knowledge to their nearby neighbors. If a model has some of its neighbors annotated, its probability list needs to be updated. Meanwhile, if the objects are far from any of the annotated objects, we do not want to link the semantic meanings between them. Such semantic meaning extension fits the framework of kernel regression very well. The annotated objects are anchor points that have known probability values. For those objects that have not been annotated, their attribute probabilities can be regressively interpolated. We assume in this paper that for each attribute the probabilities can be independently interpolated with kernel regression.

As we mentioned before, if an object O_i is annotated as having attribute A_k, the probability P_{ik} will increase to 1. Otherwise, it will drop to 0. These annotated objects are considered as anchor points in the low-level feature space. Let us consider for example one of the attributes A_k. Let $\tilde{\mathbf{x}}_m, m = 1, \cdots, M$ be the feature vectors of all the currently annotated objects, and the corresponding probabilities P_{mk} are defined as in Equation (7), i.e.,

$$P_{mk} = \begin{cases} 1, & \text{if the object corresponding to } \tilde{\mathbf{x}}_{\mathrm{m}} \text{ has } A_k, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

We proposed a simple *biased kernel regression* algorithm to estimate, given an un-annotated object whose feature vector is $\mathbf{x}$, the probability of this object having attribute A_k:

$$P(\mathbf{x} \in A_k) = \frac{\sum_{m=1}^{M} w_m P_{mk} + w_0 P^{(k)}}{\sum_{m=1}^{M} w_m + w_0} \tag{9}$$

where $P^{(k)}$ is the prior probability of any object that belongs to attribute A_k. w_0 is the tendency of the object toward the prior probability. If $w_0 = 0$, Equation (9) degenerates to the normal kernel regression. When the weight w_0 is less than 1, there exists an equivalent distance r_0, which satisfies:

$$w_0 = \exp\left(-\frac{r_0^2}{2r'^2}\right) \tag{10}$$

where r' is the kernel bandwidth at the to be predicted object. In this case, biased kernel regression can be viewed by putting a virtual anchor point at a distance r_0 to the point to be predicted, and set the probability of the virtual anchor point having attribute A_k as the prior probability.

The weights w_m are defined as:

$$w_m = \exp\left(-\frac{\left\|\mathbf{x}-\tilde{\mathbf{x}}_m\right\|_2^2}{2r_m'^2}\right), m = 1,\cdots M. \tag{11}$$

where r_m' is the bandwidth for object $\tilde{\mathbf{x}}_m$. This bandwidth will have very similar effects on the final result as that when we estimate the probability density function using kernel density estimator. Actually, we will use the same kernel bandwidth in the probability density function estimation in the last subsection and kernel regression here. Obviously, from Equation (11), an annotated object that is closer to the query point $\mathbf{x}$ will be assigned a higher weight, which gives more influence on the predicted value $P(x \in A_k)$. This is coherent with our basic assumption.

Figure 2 explains the reason why biased kernel regression is bettern than the normal kernel regression. The horizontal axis is the feature value, and the vertical axis is the probability that the corresponding object has the certain attribute. Notice that for normal kernel regression in Figure 2 (a), when the feature value is far away from the anchor points (e.g., at the two ends of the horizontal axis), and the weight is very small, the predicted probability is still close to 1 or 0. This is mainly due to the normalization of the weights at the denominator. This effect is not what we expected. Again, our assumption is that close annotated objects can infer knowledge to the current object, but far objects cannot. In other words, if an object has only very far neighbors being annotated, we expect its probability to remain similar to the prior probability. Figure 2 (b) shows the result of biased kernel regression, where r_0 is set to be equal to r', and the prior probability is set to be 0.5. It is obvious that the biased kernel regression is very suitable for our approach, as we will estimate $P(x \in A_k)$ as the prior probability if all the annotated objects are far away.

Notice that the predicted curve does not pass the anchor point, for both figures in Figure 2. This is actually a nice property of kernel regression. From the probabilistic point of view, if the object with a certain feature vector is annotated as having attribute A_k, it is still probable that another object having the same feature vector does not have this attribute. It can be proved that when the number of anchor points goes infinite and the kernel bandwidth becomes very small, the result of kernel regression asymptotically converges to the actual probability distribution [25].

Although the database might be huge, the computational cost on the probability list updating is actually low. That's because in kernel regression, a newly annotated object would not change an object's probability list if it were far away. The bandwidth of the kernel function determines the hyper-sphere inside which the object's probability lists have to be updated. These objects can be easily found if the database is organized by R-tree [26] or similar structures.

The probabilities of the objects can also be estimated parametrically. For example, we can assume the models belonging to each attribute follow a Gaussian distribution, or a Gaussian mixture distribution if necessary. Having a new model annotated is equivalent to adding a new training example to the Gaussian or the Gaussian mixture. The model to be chosen to annotate could be the one with which the system is the least confident. Such an approach does not have a smooth transition from no annotation to full annotation, because after the database is fully annotated, the parametric model does not record any annotation for each object in the database. Moreover, such kind of approach imposes a very strong global structure over the low-level feature space. When the model of the distribution is not right, the performance of the system may suffer. As a comparison, our approach also imposes some structures on the feature space but they are local. Local structures offer better opportunity to fit the database well when we do not have enough knowledge about the database.

C. The Uncertainty Measure

After all the probabilities have been updated, the learning algorithm searches among models that have not been annotated for another model whose annotation, once given by the annotator, will provide the most extra information. According to the discussion in section II-B, this model is the one that produces the maximum knowledge gain. In order to calculate the gain, we need to find the uncertainty measurement and the probability density for each model. We have described the estimation of the probability density function in Section III-A. Hereinafter we discuss the way to determine the uncertainty measurement.

In section II-B, we gave some general properties for the uncertainty measurement we want. We mentioned that if we only have one attribute to annotate for all the objects, the *entropy* is a good measure of uncertainty

$$U_i = \Psi(P_{i1}) = H(P_{i1}) = -P_{i1}\log P_{i1} - (1-P_{i1})\log(1-P_{i1}) \tag{12}$$

where U_i is the uncertainty measurement for object O_i, H is the entropy function, P_{i1} is the probability for object O_i to be characterized by the attribute. If there are multiple attributes, the uncertainty should be defined based on the joint probability of all the attributes. That is:

$$\begin{aligned} U_i &= \Psi(P_i(A_1,\cdots,A_K)) \\ &= H(P_i(A_1,\cdots,A_K)) \\ &= -\sum P_i(A_1,\cdots,A_K)\log P_i(A_1,\cdots,A_K) \end{aligned} \tag{13}$$

where $P_i(A_1,\cdots,A_K)$ represents the joint probability of object O_i having or not having the attributes. The sum is taken over all the possible combinations of attributes that an object can have.

As it is often impossible to estimate the joint probabilities, we use a simplified method to approximate Equation (13). For a certain object O_i and a certain attribute A_k, we define the individual entropy as:

$$H_{ik} = -P_{ik}\log P_{ik} - (1-P_{ik})\log(1-P_{ik}). \tag{14}$$

The overall uncertainty for an object O_i is defined by a weighted sum of the entropies for all the attributes, i.e.,

$$U_i = \sum_{k=1}^{K} w_k^{(S)} H_{ik} \tag{15}$$

where K is the total number of attributes, and $w_k^{(S)}$ is the semantic weight for each attribute. The semantic weights are related with which level in the tree the attributes are at. Let ℓ_k be the level attribute A_k is at. The weights are defined as:

$$w_k^{(S)} = a^{\ell_k - 1} \tag{16}$$

where a is a constant between 0 and 1. In our current implementation, we set a to be 0.6 based on experiments. The justification of the above weighted entropy approximation is detailed in [32].

With the uncertainty measure in Equation (15) and the probability density estimate in Section III-A, we are able to calculate the knowledge gain by simply multiplying them together as in Equation (3). The system then proposes the object with the maximum gain and asks the annotator to annotate it.

IV. Joint Similarity Measure for Semantic and Low-Level Features

The hidden annotation needs to be integrated into the retrieval system in order to provide better retrieval performance. In previous work, annotation was often regarded as a Boolean vector. In [8] and [10], normalized Hamming distance was used to combine the influence of the annotation and acted as a new feature for the retrieval. When the database is partially annotated and the annotations are used for learning, as in [11], neural networks are often used to train the similarity measurement.

In our system, each model has a list of attribute probabilities, including the query model the user provides. If the query model is chosen from the database, we already have this probability list. This is the normal case as hidden annotation is largely for improving the performance of inside-database queries. If the query model is selected from outside the database, we can estimate its probabilities as in Section III-B as well. Alternatively, the user can annotate the query model before providing it to the retrieval system. The probability list is a complete description of all the annotations we have ever made and is associated with high-level semantics. We can treat this list of probabilities as a feature vector, similar to low-level features such as color, texture, and shape. The semantic distance $d_{12}^{(S)}$ between any two objects O_1 and O_2 is defined as:

$$d_{12}^{(S)} = \sum_{k=1}^{K} w_k^{(S)} \left[P_{1k}(1 - P_{2k}) + P_{2k}(1 - P_{1k}) \right] \qquad (17)$$

where K is the total number of attributes, $w_k^{(S)}$ is the semantic weight for each attribute as defined in (16), and P_{1k} and P_{2k} are the attribute probabilities for the two models. The item $P_{1k}(1 - P_{2k}) + P_{2k}(1 - P_{1k})$ is actually the probability of objects O_1 and O_2 disagree with each other on attribute A_k (i.e., one of them has A_k but the other one does not have). We choose the form of weighted sum to measure the overall disagreement because it is simple and effective in practice. For attributes at a lower level, the weight is smaller, that is, we give a less penalty on disagreement on attributes at a lower level. Intuitively, the disagreement between a "car" and an "aircraft" is larger than that between "Classic Aircraft" and a "Jets Aircraft". Another good property of the defined semantic distance is that, if the objects to be compared have been annotated and the probabilities are either 0 or 1, the defined semantic distance will automatically degenerate into a Hamming distance (assume one-level attribute tree), which is widely used in the literature. Many other forms of semantic distances can be defined as well. However, we assumes that all the attributes in our system are not exclusive, thus distance measures that assume one normalized probability vector for each object are not good, such as the KL divergence [31].

We need another distance measure that is the distance in the low-level feature space. For two objects O_1 and O_2, we simply use the weighted Euclidean distance

$$d_{12}^{(L)} = \sqrt{\sum_{j=1}^{J} w_j^{(L)} (f_{1j} - f_{2j})^2} \qquad (18)$$

where J is the total number of features, f_{1j} and f_{2j} are the j^{th} normalized low-level features of the two objects O_1 and O_2, and $w_j^{(L)}$ is the weight set based on the importance of each feature. In the current implementation, the features are equally weighted after normalization.

The overall distance between the two models is a weighted sum of the semantic distance and the low-level feature distance:

$$d_{Overall12} = w^{(S)} d_{12}^{(S)} + w^{(L)} d_{12}^{(L)} \tag{19}$$

where $w^{(S)}$ and $w^{(L)}$ are the semantic weight and the low-level feature weight respectively and $w^{(S)} + w^{(L)} = 1$. There are several methods to specify these two weights. For example, they can be fixed as a constant, or they can be proportional to the number of objects that have been annotated in the whole database. In the current system, the weights are inverse proportional to the entropy of the query object. Therefore, if the retrieval system knows what the query is, it prefers to search the database mainly by semantic features instead of the low-level features.

As hidden annotations are made by the annotator, models that are far away from each other may be annotated as having the same attributes, which means they have small semantic distance. The integration of semantic annotations into the similarity measurement effectively works as warping of the low-level feature space to make semantically similar objects closer to each other. As illustrated in Figure 3, aircrafts are distributed beside cars in the low-level feature space. Two aircrafts and one car are annotated. When we compute the final similarity, the aircrafts will have similar attribute probabilities, and their final similarity score will be higher than when only low-level features are considered due to the introducing of item $w^{(S)} d_{12}^{(S)}$ in (19).

V. Experiments

We performed experiments on both synthetic database and real database. Due to the page limit, we refer the detailed experiments to [32]. Several main conclusions drawn there are:

1. Active learning always performs better than random sampling on the database tested, and the saving of the number of annotations can be as large as 50%;

2. For a fairly large range of the kernel bandwidth around D (defined in Equation (5)), the performance is similar, though a larger bandwidth typically gives more stable performance;
3. Adaptive kernel bandwidth does not help improve the performance;
4. The bias weight w_0 has almost no impact on the system performance.

Here we show some experimental results on a real retrieval system. We use a database of 3D objects downloaded from the Internet. The database consists of 1750 objects. More than one third of the objects are aircrafts. Some features for comparing 3D models have been proposed in the literature, e.g., those in [13][14][15]. In our system, we have 10 features extracted for each object. They are region-based features proposed in [15], including the volume-surface ratio, the aspect ratio, moment invariants and Fourier transform coefficients. The features are normalized to be within range $(-1, 1)$.

In the first experiment, we use our active learning algorithm to distinguish between aircrafts and non-aircrafts.

We measure the annotation efficiency by testing the final retrieval performance of our retrieval system. Since our system has a multi-level attribute tree structure, we define the performance measurement as follows. For any specific query q and its top R retrieved results, the average matching error for these results is measured by:

$$e_q^{(S)} = \frac{1}{R} \sum_{\substack{j \in top\, R\, results \\ of\ query q}} d_{qj}^{(S)} \tag{20}$$

where $d_{qj}^{(S)}$ is the semantic distance between the query and the j^{th} retrieved object, which is calculated by (17) with the ground truth data. The $e_q^{(S)}$ indicates the average matching error for the top R retrieved objects with respect to the query. The smaller the $e_q^{(S)}$, the better the performance of the system for the query q. The overall system performance is evaluated by:

$$Err = \frac{1}{N}\sum_{i=1}^{N} e_i^{(S)} \tag{21}$$

where N is the number of objects in the database, as we take every object in the database as a query and calculate the average matching error. The final performance of the system is measured by taking average of the average matching error for all the objects.

The performance comparison between our active learning algorithm and the random sampling algorithm on the 3D model database is given in Figure 4. To start both algorithms, 50 models are randomly chosen and annotated. We use fixed bandwidth for kernel density estimation and kernel regression. The bandwidth is set to be $2D$, where D is defined in Equation (5). In the biased kernel regression, we choose the weight of the prior probability as e^{-2}. Only the performance on the top 20 retrieved results are reported.

The horizontal axis of Figure 4 is the number of samples that have been annotated, including the initial 50 randomly drawn samples. The vertical axis is the average matching error for the whole database measured by (21). A curve closer to the bottom-left corner is considered to have a better performance. From Figure 4, it is obvious that our active learning algorithm works much better than the random sampling approach.

Finally, we test the algorithm on the 3D model database with 13 attributes. The airplane category is further extended into 8 subcategories such as biplane, jets, helicopters, etc., while the non-airplanes are divided into 4 categories such as characters, shapes, buildings, etc. Figure 5 shows the performance of our algorithm versus random sampling. Our algorithm still works better than the random sampling algorithm, though the improvement is not as significant as in Figure 4 and the synthetic database. This may be due to the feature space of the 3D model database. When more and more classes are specified, some of the classes may violate the assumption we gave in Section I. That is for objects in some classes the local inferable property may not hold any more.

VI. Conclusions and Discussions

In this paper, we proposed a general approach to add hidden annotation using active learning for information retrieval. We considered a natural attribute tree structure for the annotation. The object to be annotated at any instant was determined by the knowledge gain of the system by annotating it. We defined the knowledge gain as the product of the probability density function and the uncertainty measurement. In order to evaluate the uncertainty of an object, we gave each object a list of attribute probabilities, computed based on its neighboring annotated objects through kernel regression. We obtained the uncertainty of an object by giving an explicit function on these probabilities. The proposed algorithm outperforms the random sampling algorithm in all the experiments we performed, which shows that hidden annotation with active learning is a very powerful tool to improve the performance of content-based information retrieval.

Hidden annotation and relevance feedback have different strategies and serve for different purposes. Normal relevance feedback does not accumulate semantic knowledge, and is able to tune to each query very quickly. Hidden annotation, on the other hand, tries to accumulate all the knowledge given by the annotator. It is application-dependent to choose between them or to choose both. The knowledge accumulation can also be smartly turned on or off depending on whether the user is trustable or not.

In active learning, one may have the concern whether the annotator's preference is always the same as a user. Although the annotator may be an expert, the user may have his/her own criteria for similarity. We can modify our algorithm to allow user feedback. Because we define the overall distance as a weighted sum of the semantic distance and the low-level feature distance as in (19), where the weights between semantic distance and low-level feature distance can also be adjusted by the relevance feedback. This is part of our future work.

To compare our approach with Lewis and Gale's approach in [19], their approach is designed for text classification, while ours is for information retrieval. Our approach is more general and can be easily modified to be used in other classification problems.

Acknowledgement

This work is supported in part by NSF CAREER Award 9984858. The authors would like to thank Dr. David Cohn for the helpful discussions and suggestions. We would also thank the reviewers who helped us improve the quality of the paper.

Reference

[1] Yong Rui, Thomas S. Huang, and Shih-Fu Chang, "Image Retrieval: Past, Present, and Future", Proceeding of International Symposium on Multimedia Information Processing, Dec. 1997.

[2] Donna Harman, "Relevance Feedback Revisited", Proceedings of the Fifteenth Annual International ACM SIGIR conference on Research and development in information retrieval, pp. 1-10, 1992.

[3] Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-based Image Retrieval", IEEE Trans. On Circuits and Systems for Video Technology, pp. 644-655, Vol. 8, No. 5, Sep. 1998.

[4] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Query Database through Multiple Examples", Proceeding of the 24th VLDB Conference, New York, 1998.

[5] Yong Rui and Thomas S. Huang, "Optimizing Learning in Image Retrieval", Proceeding of IEEE int. Conf. On Computer Vision and Pattern Recognition, Jun. 2000.

[6] Qi Tian, Pengyu Hong, Thomas S. Huang, "Update Relevant Image Weights for Content-based Image Retrieval Using Support Vector Machines", Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on, pp. 1199-1202, Vol. 2 , 2000.

[7] T. P. Minka and R. W. Picard, "Interactive Learning Using a 'Society of Models'", M. I. T Media Laboratory Perceptual Computing Section Technical Report No. 349.

[8] Ingemar J. Cox, Matt L. Miller, Thomas P. Minka, Thomas V. Papathomas, and Peter N. Yianilos, "The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments", IEEE Trans. On Image Processing, pp. 20-37, Vol. 9, No. 1, Jan. 2000.

[9] Catherine S. Lee, Wei-Ying Ma and Hong Jiang Zhang, "Information Embedding Based on User's Relevance Feedback for Image Retrieval", Invited paper, SPIE Int. Conf. Multimedia Storage and Archiving Systems IV, Boston, 19-22, Sep. 1999.

[10] Ye Lu, Chunhui Hu, Xingquan Zhu, Hong Jiang Zhang, Qiang Yang, "A Unified Framework for Semantics and Feature Based Relevance Feedback in Image Retrieval Systems", ACM Multimedia, 2000.

[11] W. Y. Ma and B. S. Manjunath, "Texture Features and Learning Similarity", Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on, pp. 425-430, 1996.

[12] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective Sampling Using the Query by Committee Algorithm", Advances in Neural Information Processing Systems, Cambridge, MA: MIT Press, 1993.

[13] Eric Paquet and Marc Rioux, "Nefertiti: A Query by Content Software for Three-dimensional Models Databases Management", 3-D Digital Imaging and Modeling, 1997. Proceedings. International Conference on Recent Advances in, pp. 345-352, 1997.

[14] Titus, Zaharia, François Prêteux, Marius Preda, "3D Shape Spectrum Descriptor", MPEG-7 ISO/IEC JTC1/SC29/WG11 MPEG99/M5242, Melbourne, Australia, Oct. 1999.

[15] Cha Zhang and Tsuhan Chen, "Efficient Feature Extraction for 2D/3D Objects in mesh representation", ICIP 2001, Thessaloniki, Greece, 2001.

[16] H. S. Seung, M. Opper, H. Sompolinsky, "Query by Committee", Proceedings of the fifth annual ACM workshop on Computational learning theory, July 27 - 29, Pittsburgh, PA USA, 1992.

[17] Y. Freud, H. S. Seung, E. Shamir, and N. Tishby, "Information, Prediction, and Query by Committee", Advances in Neural Informations Processing Systems 5, San Mateo, CA, 1992. Morgan Kaufmann.

[18] Kamal Nigam and Andrew McCallum, “Employing EM in Pool-Based Active Learning for Text Classification”, Proceedings of ICML-98, 15th International Conference on Machine Learning, 1998.

[19] David D. Lewis and William A. Gale, “A Sequential Algorithm for Training Text Classifiers”, ACM-SIGIR 94, pp. 3-12, Springer-verlag, London, 1994.

[20] David D. Lewis, “A Sequential Algorithm for Training Tex Classifiers: Corrigendum and Additional Data”, SIGIR Forum, pp. 13-19, Vol. 29, No. 2, Fall 1995.

[21] Ion Muslea, Steven Minton, and Craig A. Knoblock, “Selective Sampling with Co-testing”, The CRM Workshop on "Combining and Selecting Multiple Models With Machine Learning", Montreal, Canada - April 2000.

[22] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan, “Active Learning with Statistical Models”, pp. 129-145, Journal of Artificial Intelligence Research 4, 1996.

[23] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems, volume 7, Cambridge, MA, 1995. MIT Press.

[24] M. Hasenjäger, H. Ritter, and K. Obermayer. Active learning in self-organizing maps. In E. Oja and S. Kaski, editors, Kohonen Maps, pages 57-70. Elsevier, Amsterdam, 1999.

[25] Richard O. Duda, Peter E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973.

[26] A. Guttman, “R-trees: a dynamic index structure for spatial searching”, Proceedings of the SIGMOD Conference, Boston, MA, June 1984, pages 47-57.

[27] Silverman B.W., Density Estimation for Statistics and Data Analysis, New York: Chapman and Hall, 1986.

[28] D. Koller and R. Fratkina, “Using learning for approximation in stochastic processes”, ICML-98.

[29] S.-T. Chiu, “A Comparative Review of Bandwidth Selection for Kernel Density Estimation”, Statistica Sinica, pp.129-145, No.6, 1996.

[30] Terrell, G. R. and Scott, D. W. “Variable Kernel Density Estimation”, The Annals of Statistics, pp. 1236-1265, No. 20, 1992.

[31] Cover, T. M., & Thomas, J. A. (1991). Elements of Information Theory. John Wiley and Sons, New York.

[32] Cha Zhang and Tsuhan Chen, “A New Active Learning Approach for Content-Based Information Retrieval”, Technical Report, AMP 01-04 2002.

List of figures

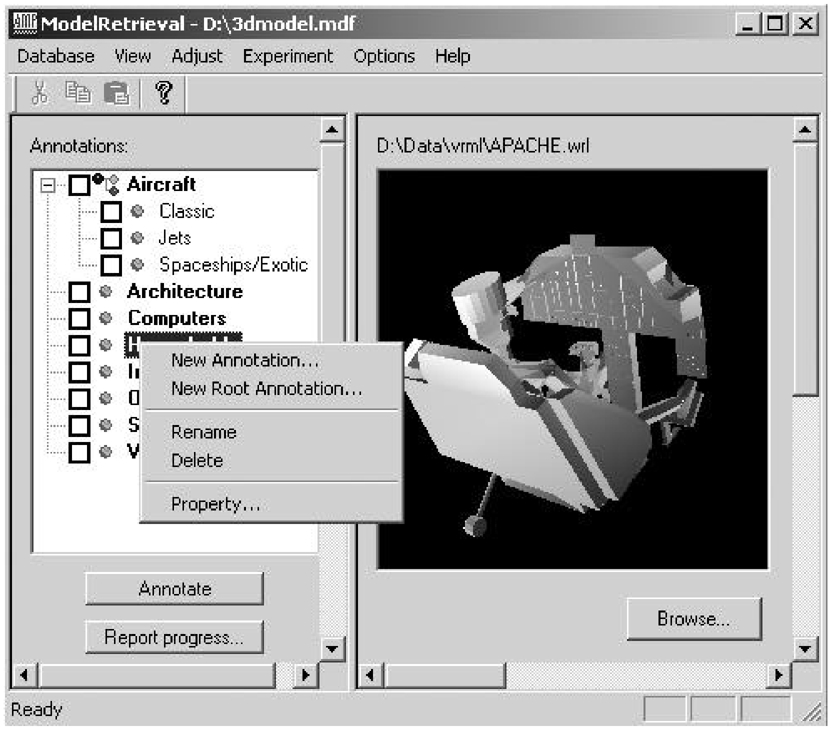

Figure 1 The learning interface and the tree annotation structure.

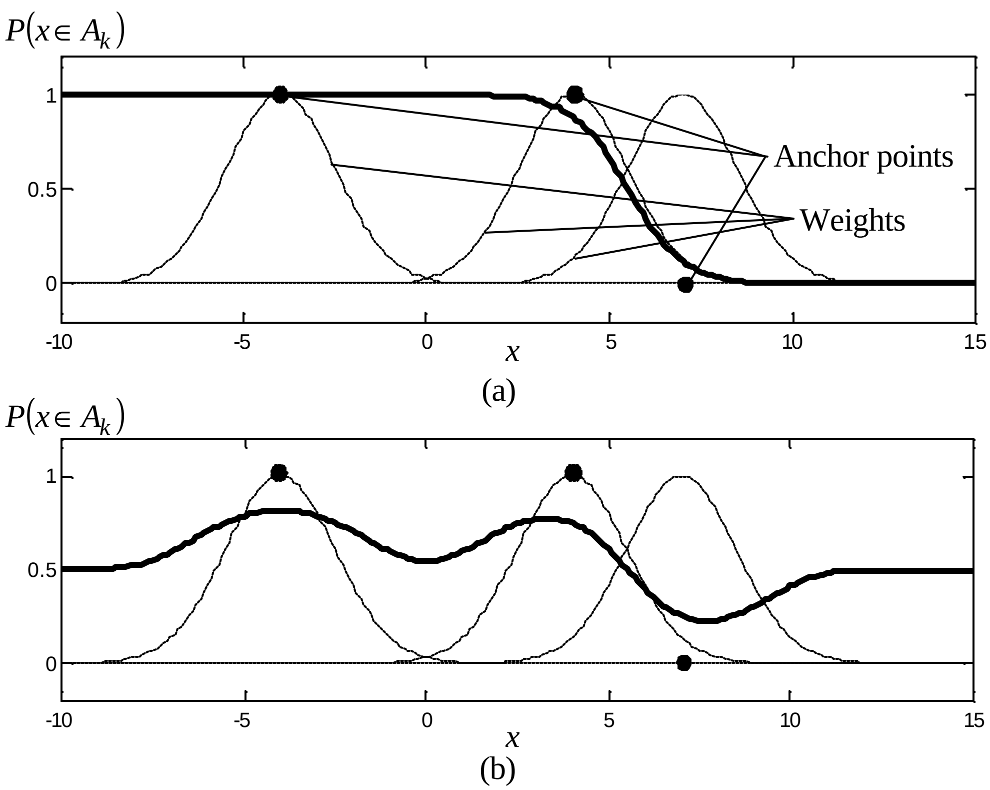

Figure 2 Normal kernel regression (a) and biased kernel regression (b).

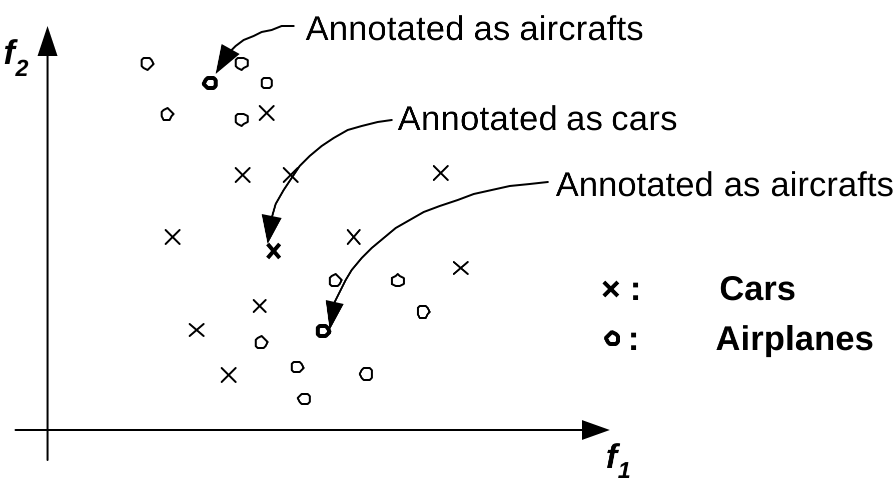

Figure 3 The annotations can "warp" the feature space.

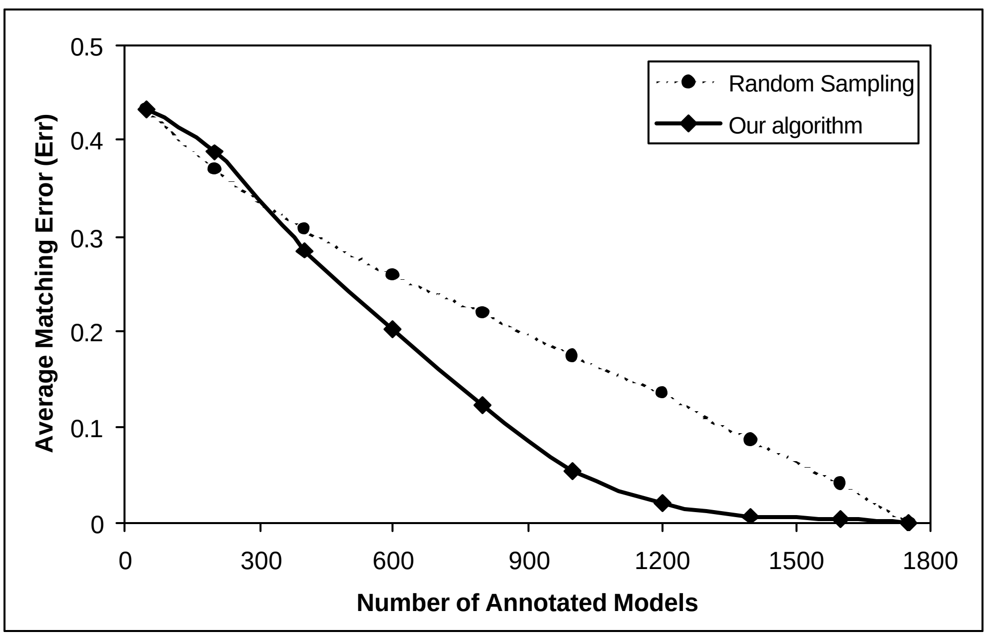

Figure 4 Performance comparison between our algorithm and random sampling for the 3D model database on two attributes.

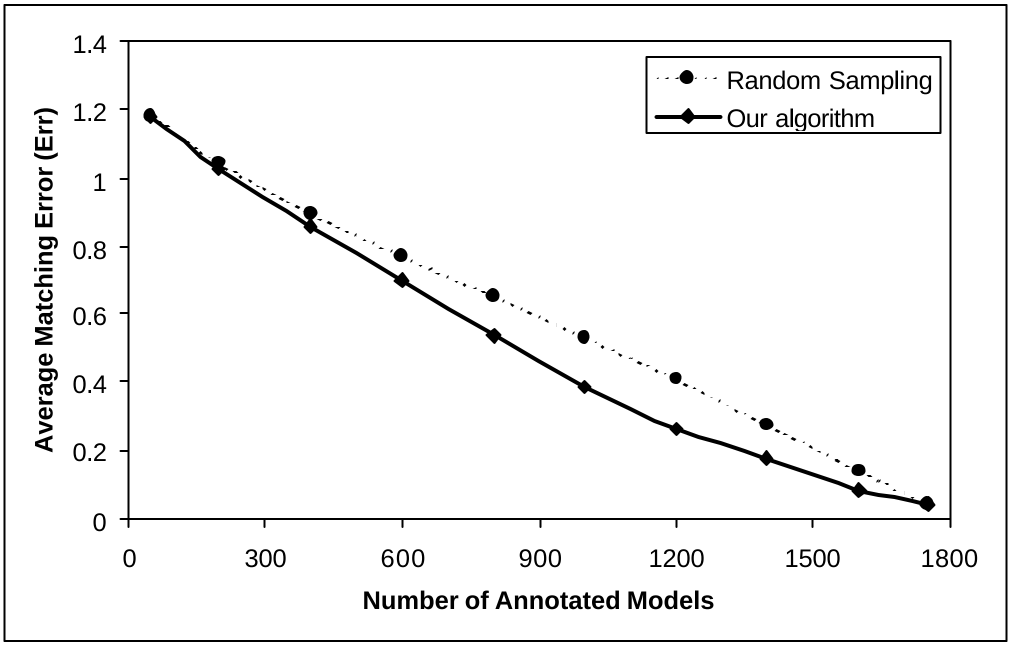

Figure 5 Performance comparison between our algorithm and random sampling for the 3D model database on 13 attributes.