ACTIVE LEARNING FOR VIDEO ANNOTATION

[1]Barbara A. Hohlt and [2]Belle L. Tseng

[1]Electrical Engineering and Computer Sciences
University of California, Berkeley
387 Soda Hall #1776
Berkeley, CA 94720-1776

[2]IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532

ABSTRACT

In this paper, we present an approach to *active learning* for video annotation. We use active learning to aide in the semantic labeling of video databases. A software library and simulator have been developed as a tool to investigate and measure the results of active learning. A list of confidence values and nearest neighbor values is maintained for each video segment in the database. A variety of similarity measures are used interchangeably to create hierarchies (or cluster trees) of features for each video. The learning approach proposes sample video segments to the user for annotation and updates the database with the new annotations. It then uses its accumulative knowledge to label the rest of the database, after which it proposes new samples for the user to annotate. The proposed samples are selected by their *knowledge gain* to the active learner.

1. INTRODUCTION

Since the 1990's, Content-Based Image Retrieval (CBIR) has become the most popular framework for image retrieval [13]. IBM's Query By Image Content (QBIC) [15] was the first commercial content-based image retrieval system based on low-level features.

Today's state-of-the-art CBIR systems use the combination of low-level features, relevance feedback [10][12], and text-annotation [7][11] to bridge the gap between low-level features and their high-level semantic meaning. Studies have shown that semantic information and relevance feedback greatly facilitate image retrieval [9]. However, the old problems of labor-intensive manual annotation and subjectivity of human perception still persist.

Recently, a machine learning technique called *active learning* has recently been used to improve query performance in image retrieval systems [4] [7]. The concept is to maximize the expected information from a query as a result of user feedback in order to minimize the total number of iterations required for the search.

This research proposes an approach to *active learning* for video annotation. The goal of active learning when applied to annotation is to significantly reduce the number of images annotated by the user. We use active learning to aide in the semantic labeling of video databases. The learning approach *proposes* sample video segments to the user for annotation and updates the database with the new annotations. It then uses its accumulative knowledge to propagate the labels to the rest of the database, after which it proposes new samples for the user to annotate. The sample images are selected based on their ability to increase the *knowledge gained* by the system.

We have developed a video annotation library, Val, and simulator as a tool to investigate and measure the results of active learning. The Val software library consists of modules for maintaining matrices, data clustering, sample selection, and simulation. A list of confidence values and corresponding nearest neighbor values are maintained for each video segment in the database. A variety of similarity measures are used interchangeably to create hierarchies (or cluster trees) of features for each video segment.

The Val is specifically designed and developed as an extension to the video annotation tool, VideoAnn, developed at IBM T. J. Watson. The VideoAnn annotation tool assists authors in the task of annotating video sequences. Each scene, or *shot*, in the video sequence can be annotated with static scene descriptions, key object descriptions, event descriptions, and other keywords. The inputs to VideoAnn are a video sequence and a corresponding shot file denoting scenes in the video.

2. RELATED WORK

Content Based Image Retrieval, commonly referred to as CBIR, is an active area of research due to the growing need for effective searching and retrieval of multimedia content. CBIR tries to match low-level features such as the color, shape and texture of an image, to a given query. However low-level features alone are not complete enough to represent the high-level semantic meanings of the objects in a database. Relevance feedback and hidden annotation are used to bridge the gap between low-level features and high-level semantic meaning.

2.1 Relevance Feedback

Rui and Huang, UIUC MARS project, pioneered the relevance feedback based approach to CBIR[10][12][13]. In [12] relevance feedback is defined as "the process of automatically adjusting an existing query using the information fed-back by the user about the relevance of previously retrieved objects".

Most CBIR systems now apply relevance feed back on the image's low-level features. The iFind [9] system goes a step further and performs relevance feedback on the image's low-level features as well as its semantic content represented by keywords.

2.2 Hidden Annotation

The PicHunter project at MIT uses hidden annotation for image retrieval [7]. In this system images are manually annotated with a set of pre-defined semantic attributes. These attributes are represented as a boolean vector for each image and regarded as an additional feature during image retrieval.

2.3 Active learning

Active learning is a machine learning method used to optimally select what data are added to a learner's training set. It was originally coined and described by Cohn, Ghahramani, and Jordan in [14] where they applied the principle to statistically based machine learning. Recently active learning techniques have been applied to image retrieval systems [3][4][5][6][7][8].

The *Active Learning* project at CMU [3] uses active learning to produce hidden annotation on a database of 1750

3D models. For un-annotated data the system estimates their probabilities based on a potential function. Their study showed active learning outperforms learning based on random sampling. Like the CMU project we also use active learning to produce hidden annotation as a stage separate from retrieval. We differ in our use of hierarchical data clustering for classification, sample selection, and data set (ours is a video data base of 5882 shots).

Active learning *proposes* which images should be annotated. The sample images are selected on their ability to increase the *knowledge gained* by the system. The goal of active learning when applied to annotation is to significantly reduce the number of images annotated by the user. Our system is similar to those cited in the literature in that we use low-level features, text-annotation, relevance feedback, and active learning. However, we differ from most in that we are using active learning to produce hidden annotation as a separate stage from retrieval. Most fundamentally, our system is not a retrieval system. It is software extension to the VideoAnn annotation tool aimed at speeding the process of video annotation.

3. VIDEO ANNOTATION WITH EXTENSION

In this section, we outline the functionalities and feature attributes generated by our annotation tool, called *IBM VideoAnn Annotation Tool*. Four major components describe the annotation process and are depicted in Figure 1. First, video segmentation is performed to cut up the video sequence into smaller video units. Second, a semantic lexicon is defined in order to regulate the video content descriptions. Third, the annotator labels the video segments with the semantic descriptions. Fourth, the MPEG-7 descriptions of the annotation process are directly outputted from the *IBM VideoAnn Annotation Tool*. The goal of the video annotation is to categorize the semantic content of each video unit, and output the MPEG-7 XML description file. The following subsections describe these components in further detail.

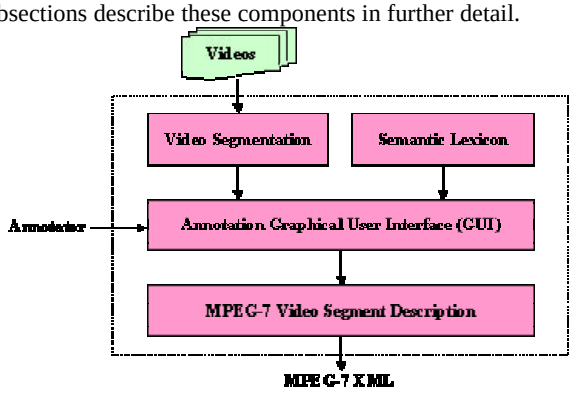

Figure 1. VideoAnn with VAL Extension

3.1 Video Segmentation

For a video sequence, shot boundary detection is performed to divide the video into multiple shots. The *IBM CueVideo Toolkit* performs the shot detection algorithm, which is based on the multiple timescale differencing of the color histogram [6]. *CueVideo* segments our video content into shorter shots, where scene cuts, dissolves, and fades are effectively detected. Because each video shot can be described and retrieved independently of each other, the next step is to define our lexicon for shot descriptions.

3.2 Semantic Lexicon

Given the segmentation of video content into video shots, the second step is to define the semantic lexicon in which to label the shots. A video shot can fundamentally be described by three attributes. The first is the background surrounding of where the shot was captured by the camera, which is referred to as the *static scene*. The second attribute is the collection of significant subjects involved in the shot sequence, which is referred to as the *key object*. Lastly, the third attribute is the corresponding action taken by some of the key objects, which is referred to as the *event*. These three types of lexicon define the vocabulary for our video content.

Using the defined vocabulary for static scenes, key objects, and events, the lexicon is imported into our *IBM VideoAnn Annotation Tool* for describing and labeling each video shot. The shots are labeled for its content with respect to the selected lexicon. Note that the lexicon definitions are database and application specific, and can be easily modified and imported into the annotation tool.

3.3 Annotation Graphical User Interface

The *IBM VideoAnn Annotation Tool* assists authors in the task of annotating video sequences. Each shot in the video sequence can be annotated with static scenes, key objects, events, and other keywords. These descriptions are labeled for each shot and are stored as MPEG-7 descriptions in the output XML file. *VideoAnn* can also save, open, and retrieve MPEG-7 files in order to display the annotations for corresponding video sequences.

VideoAnn is divided into four graphical sections as illustrated in Figure 2. On the upper right-hand corner of the tool is the *Video Playback* window with shot information. On the upper left-hand corner of the tool is the *Shot Annotation* with a key frame image display. On the bottom portion of the tool is two different *Views Panel* of the annotation preview. A fourth component, not shown in Figure 2, is the *Region Annotation* pop-up window for specifying annotated regions. These four sections provide interactivity to assist authors of the annotation tool. A more detailed description of the annotation tool is shown in [1][2].

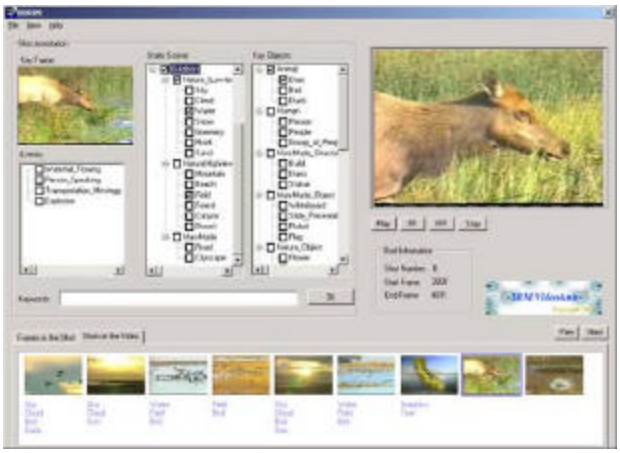

Figure 2. IBM Video Ann Annotation Tool.

4. VIDEO ANNOTATION LIBRARY

4.1 The Val Software Library and Simulation

We designed and developed a software library and simulator to aide in the semantic labeling of video and image databases. Written in C++, the Val software library can be interfaced to the VideoAnn or other annotation software. We use the simulator to develop our learning algorithms and to analyze the results. Val consists of six software modules;

1. The Val Learner Interface

2. Shot Label and Feature Data
3. Difference Matrices and NN Algorithms
4. Cluster Trees and Data Clustering Algorithms
5. Annotations and Knowledge
6. Truth and Simulation

The first module serves as the interface between the software library and the application. To initialize the system, modules two thru four process feature vector data in a pre-processing stage and output data files that will be used during annotation, or in our case simulation. Module six, Truth and Simulation, simulates a human annotator and is only used for simulation and analysis.

Presently, the Learner operates on one label name and one feature model at a time. Each shot has a value for label that can either be positive one, negative one, or zero. For example if our label name is "water', then the value of positive one for label would indicate the presence of water. When simulation begins the Learner expects a label name, a difference matrix, a shot id-cluster group table, and a truth table as input. The truth table, or oracle, is the table of positive and negative annotations for every shot.

A simulation loop goes as follows:

1. Ask Oracle for the annotations of q samples
2. Update the master annotation and group lists
3. Update Shot Label Table with new annotations
4. Update sample selection criteria
5. Propagate labels to the rest of the database
6. Select q samples to be annotated
7. Output analysis

4.2 Data Clustering

The Val currently uses a simple classifier based on seven data clustering algorithms described by Anil Jains and Richard Dubes [16]. Each data clustering algorithm takes as input a difference matrix. The Val currently supports four difference methods; city block (manhattan), euclidean difference, 1-pearson correlation coefficient, and square difference. The clustering algorithms were adapted from an existing implementation by the University of Groningen [17].

In a preprocessing stage the system reads a file of low-level feature data vectors and produces a difference matrix. The features we use are 166-bin HSV color-space and spatial-frequency energy texture-space. A clustering of n groups is made from the difference matrix. Each shot then belongs to a cluster group. Additionally, for every shot, the k-nearest neighbors from the difference matrix are saved. For our experiments we used a euclidean difference matrix as input to the minimum variance data clustering algorithm.

4.3 Label Propagation

Label propagation is a simple classification scheme consisting of fine grain and course grain majority votes. A running count per cluster group of all positive and negative annotations so far is kept. The coarse grain vote for a cluster group is then positive if the positive count is greater or negative if the negative count is greater. For the fine grain vote, at each iteration we count the number of positive and negative annotated k-nearest neighbors. If the majority is positive then the vote is positive and vice versa. If they are equal then we take the sums of the difference values for both the positive and the negative annotated k-nearest neighbors, and the smaller sum wins.

Our simple label propagation scheme goes as follows:

For every shot do
If there is a fine grain vote
    Shot label = knn vote
Else if there is a coarse grain vote
    Shot label = cluster group vote
Else
    No label

If we have no information on a particular cluster group it will not have a vote for that particular iteration.

4.4 Sample Selection

Sample selection selects the next q unlabeled shots to be used in the next iteration. The Val currently supports two simple algorithms. For testing and comparison we have the naïve algorithm that selects q unlabeled shots at random. The second algorithm, *k-ratio*, is based on the *maximum knowledge gain* function described by Zhang and Chen [3].

$$G[i] = P[i] * U[i]$$

P[i] is a local density function related to the local density of the k-nearest neighbors around shot i. U[i] is the uncertainty measure of the label of shot i. We use the local density function of shot i defined in [3] as

$$P[i] = D[i,k]^{-y}$$

where D[i,k] is the maximum distance between shot i and its k-nearest neighbors. U[i] is our *k-ratio* function, and is a simple function based on k-nearest neighbors. For each iteration a k-nearest neighbor ratio is calculated for every shot. This ratio is the sum of the distances of annotated positive nearest neighbors to the sum of the distances of annotated negative nearest neighbors.

Our next planned algorithm will be *stratified sampling*. Here we will randomly select samples from each data cluster group at each iteration.

5. ANALYSIS

All of our measurements were performed on the Millenium cluster at UC Berkeley on Pentium III machines running Linux kernel 2.4.1. Our video database consists of 5882 scenes from the TREC Video corpus of MPEG video. The features we use are 166-bin HSV color-space and spatial-frequency energy texture-space. The data were partitioned into 20 groups using euclidean distance matrices as input and the minimum variance clustering algorithm.

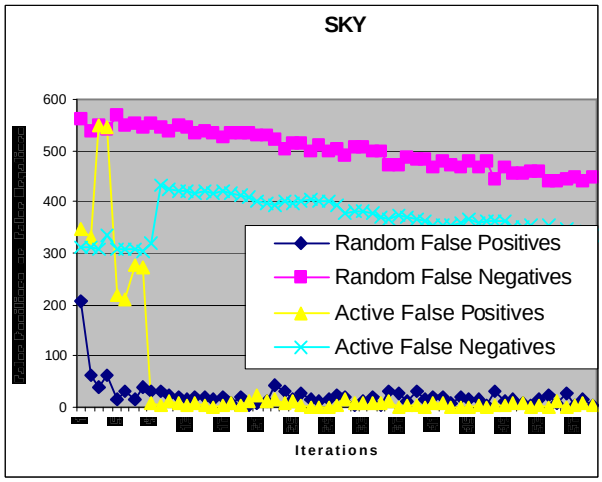

Figure 3. False Positives and False Negatives on Color Features for Sky.

To provide our ground truth, video scenes were pre-annotated with the following labels: airplane, boat, rocket, vehicle, outdoors, sky, water, outer-space, greenery, face, fire-smoke, land, sand, canyon, and rock.

We ran two test sets on the color features and the texture features; one set with random sample selection and the other set with k-ratio sample selection. In the k-ratio sample selection we did not set gamma, so it is effectively –1. For each test we attempted to label 5882 shots. We sampled 20 shots at each iteration and iterated 60 times totaling 1200 sampled shots for 5882 labels for each test.

The running time for each test averages around 5 minutes to label 5882 shots. Of that, 4 minutes (about 95 cpu seconds) are spent loading the matrix, cluster, and truth files and 1 minute is spent on the 60 iterations total. Each iteration takes about 121 cpu seconds. This tells us that it is feasible for a human to annotate video assisted by a learner in real time.

Of the random experiments, in the color feature-SKY label experiment 5431 shots are correct, and 451 are wrongly labeled. In the color feature-WATER label experiment 5381 are correct, and 501 are wrongly labeled.

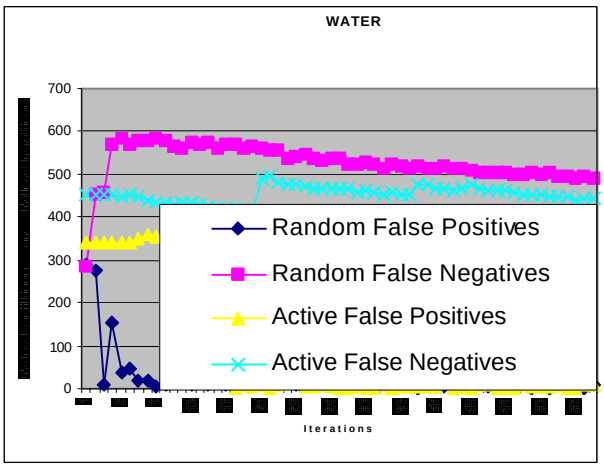

Figure 4. False Positives and False Negatives on Color Features for Water.

In the active learner experiments, not all the shots end up labeled by the active learner. For example, in the color feature-SKY label experiment 4558 shots are correct, 983 have no label, and 341 are wrongly labeled. In the color feature-WATER label experiment 5281 are correct, 149 have no label, and 452 are wrongly labeled.

When we look at our ground truth we see that out of 5882 shots, 583 shots have sky and 603 have water. So counting the number of false positives and false negatives becomes more interesting to us as a goodness metric especially since our goal is to generate good annotation.

Random does better in terms getting more labels correct, but it also gets more labels wrong. It would do better if we had a stronger classifier. Our k-ratio sample selection does better than random in terms of false positives and false negatives. But our problem here is that we are still not getting enough representation for each cluster group, so the coarse grain votes are hurting us.

We believe that stratified sampling will do better than the two above. This is directly related to the type of classifier we are using that is based on hierarchical trees.

6. CONCLUSION

We have demonstrated a system that aides in the semantic labeling of video scenes. Currently, the system uses a coarse hierarchical data clustering and a fine grain nearest neighbor approach for classifying. We demonstrated our framework for using *active learning* for annotating a video database using a naïve sample selection algorithm.

Initial tests lead us to two observations:

1. It is feasible for a human to annotate video assisted by an active learner in real time.
2. Goodness metrics are tightly coupled with the data set.
3. Sample selection is dependent on the classification method used.

This approach to video annotation is of use in situations where manual annotation of video and image databases is too demanding of human resources.

7. REFERENCES

[1] B. L. Tseng, C-Y Lin, J. R. Smith, "Video Summarization and Personalization for Pervasive Mobile Devices." SPIE Electronic Imaging 2002.

[2] M. Naphade, C-Y. Lin, J.R. Smith, B. L. Tseng, S. Basu, "Learning to Annotate Video Databases." SPIE Storage and Retrieval for Media Databases 2002.

[3] C. Zhang, T. Chen, "Active Learning for Information Retrieval: Using 3D Models As An Example." CMU TR AMP 01-04 April 2001.

[4] S. Tong, E. Chang, "Support Vector Machine Active Learning for Image Retrieval." ACM ICMM 2001.

[5] E. Chang, T. Cheng, W. Lai, C. Wu, C. Chang and Y. Wu, "PBIR: A System that Learns Subjective Image Query Concepts." ACM ICMM 2001.

[6] A. Amir, D. Ponceleon, B. Blanchard, D. Petkovic, S. Srinivasan, G. Cohen, "Using Audio Time Scale Modification for Video Browsing." Hawaii Int. Conf. On System Sciences 2000.

[7] I. Cox, M. Miller, T. Minka, T. Papathomas, P.Yianilos, "The Bayesian Image Retrieval System, *PicHunter*: Theory, Implementations, and Psychophysical Experiments." IEEE TOIP 2000.

[8] V. Iyengar, C. Apte, T. Zhang, "Active Learning using Adaptive Resampling." ACM KDD 2000.

[9] Y. Lu, C. Hu, X. Zhu, H. J. Zhang, Q. Yang, "A Unified Framework for Semantics and Feature Based Relevance Feedback in Image Retrieval Systems." ACM MM 2000.

[10] Y. Rui, T. Huang, "A Novel Relevance Feedback Technique in Image Retrieval." ACM ICM 1999.

[11] S. Paek, C. Sable, V. Hatzivassiloglou, A. Jaimes, B. Schiffman, S.-F. Chang, K. McKeown, "Integration of Visual and Text-Based Approaches for the Content Labeling and Classification of Photographs." Workshop on Multimedia Indexing & Retrieval, ACM SIGIR 1999.

[12] Y. Rui, T. Huang, M. Ortega, S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval." IEEE TCSVT SPIE 1998.

[13] Y. Rui, T. Huang, S.-F. Chang, "Image Retrieval: Past, Present, and Future." ISMMIP1997.

[14] D. Cohn, Z. Ghahramani, M. Jordan, "Active Learning with Statistical Models." Journal of Artificial Intelligence Research 1996.

[15] W. Niblack, R. Barber, et al., "The QBIC Project: Querying Images By Content Using Color Texture and Shape." SPIE SRIVD 1994.

[16] A. Jain, R. Dubes, "Algorithms for Clustering Data." Michigan State University, Prentice Hall 1988.

[17] P. Kleiweg, University of Gronigan (Netherlands), http://odur.let.rug.nl/~kleiweg/clustering/clustering.html