IBM Research TRECVID-2008 Video Retrieval System

Apostol Natsev[*], Wei Jiang[†], Michele Merler[†], John R. Smith[*],
Jelena Tešić[*], Lexing Xie[*], Rong Yan[*]

Abstract

In this paper, we describe the IBM Research system for indexing, analysis, and retrieval of video as applied to the TREC-2008 video retrieval benchmark. This year, focus of the system improvement was on large-scale learning, cross-domain detection, and interactive search.

A. High-level concept detection:

1. *A_ibm.Baseline_5: Baseline runs with random-subspace bagging;*
2. *A_ibm.BaseSSL_4: Fusion of baseline runs and principal component semi-supervised support vector machines(PCS^3VM);*
3. *A_ibm.BaseSSLText_3: Fusion of A_ibm.BaseSSL_4 and text search results;*
4. *C_ibm.CrossDomain_6: Learning on data from web domain;*
5. *C_ibm.BNet_2: Multi-concept learning with baseline, PCS^3VM and web concepts;*
6. *C_ibm.BOR_1: Best overall runs by compiling the best models based on heldout performance for each concept.*

   Overall, almost all the individual components can improve the mean average precision after fused with the baseline results. To summarize, we have the following observations from our evaluation results: 1) The baseline run using random-subspace bagging offers a reasonable starting performance with a more efficient learning process than standard SVMs; 2) By learning on both feature space and unlabeled data, PCS^3VM is able to improve the MAP by 12% after combined with baseline runs; 3) The additional development data collected from the web domain are shown to be informative on a number of the concepts, although its average performance is not comparable with baseline yet;

B. Interactive search:

1. *I_A_2_IBM.SearchTypeA_2: Type-A interactive run with 20 semantic concepts targeted in the 2008 HLF task.*
2. *I_C_2_IBM.SearchTypeC_1: Type-C interactive run with 96 semantic concepts, trained on additional web data.*

   Different system analytics such as clustering and visual near-duplicates have notably helped, especially in increasing recall. There were no significant different between the two interactive runs, which used the same system setup except for number of semantic concepts available.

Keywords—*Multimedia indexing, content-based retrieval, Support Vector Machines, Model Vectors, Model-based reranking.*

1 Introduction

This year the IBM team has participated in the TREC Video Retrieval Track, and submitted results for the High-Level Feature Detection and Interactive Search tasks. This papers describe the IBM Research system and examine the approaches and results for both tasks.

The IBM team continues its investigation on high-level feature detection along two main directions: large-scale

[*]IBM T. J. Watson Research Center, Hawthorne, NY, USA
[†]Dept. of Computer Science, Columbia University

learning and cross-domain detection. First, we significantly increased the number of low-level feature types from less than 10 to be around 100. To efficiently learn from such a large pool of features, we generated the baseline results using random subspace bagging, which fuses an ensemble of SVMs learned on randomly selected feature subspace and bootstrapped data samples. Multiple sampling and combination strategies have been tested. To leverage the potential benefits of unlabeled data from different domains, we also evaluated a new semi-supervised learning algorithm that merges feature space learning and transfer concept detection into a unified framework. Promising results have been observed. In addition, we provided a Type-C baseline run to verify if training data downloaded from WWW can be contributive to detecting concepts in the news domain. Finally, the relationship between the multimedia ontology had been utilized to augment the detection performance of individual concepts. The official evaluation results show that almost all the individual components can bring in performance improvement, and as a result, our best run achieved 30% improvement over the baseline run in terms of mean average precision.

We also conducted two interactive search runs with identical system setup and different number of semantic concepts. The system has four different types of visual analytics and supports numerous types of user interaction for aggregating, tagging, re-arranging and trimming the result lists. The different analystics and system features helps improve precision and recall, while there is no marked different between the two sets of semantic concepts in the 24 target queries.

2 Video Descriptors

2.1 Visual Features

All of the visual features are extracted from the representative keyframes of each video shot. These keyframes are provided by LIG[AQ07] and AT&T [LGZ+07]. Because learning on a rich set of low-level features has been shown to be effective in improving the concept detection performance, we have significantly increased the number of feature types to be 98, by means of generating 13 different visual descriptors on 8 granularities (i.e., global, center, cross, grid, horizontal parts, horizontal center, vertical parts and vertical center)[1]. The relative performance within a given feature modality (e.g., color histogram vs color correlogram) is shown to be consistent across all concepts/topics, but the relative importance of one feature modality vs. another may change from one concept to the other.

We apply cross validation on the development data to evaluate the generalizability of each individual feature. In the following, we have listed a sample set of descriptors which had achieved the top overall performance for the concept modeling task:

- Color Histogram (CH)—global color represented as a 166-dimensional histogram in HSV color space.
- Color Correlogram (CC) — global color and structure represented as a 166-dimensional single-banded auto-correlogram in HSV space using 8 radii depths.
- Color Moments (CM) — localized color extracted from a 5x5 grid and represented by the first 3 moments for each grid region in Lab color space as a normalized 225-dimensional vector.
- Wavelet Texture (WT)—localized texture extracted from a 3x3 grid and represented by the normalized 108-dimensional vector of the normalized variances in 12 Haar wavelet sub-bands for each grid region.
- Edge Histogram (EH)—global edge histograms with 8 edge direction bins and 8 edge magnitude bins, based on a Sobel filter (64-dimensional).

2.2 Semantic Features

In order to expand the semantic coverage for the task of interactive retrieval, we generate the model vector features consisting of the detection confidences for each of the 20 concept models in the official list, together with the following:

- Web – a model vector consisting of concept scores of 200+ models trained on consumer and web images.

[1]The final number of features is slightly smaller than expected because some of the visual descriptors are only generated on a selected set of granularities

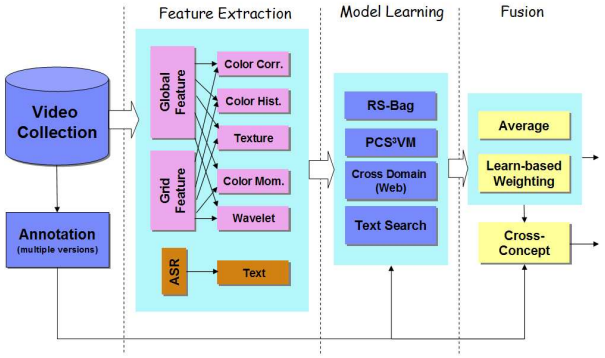

Figure 1: Overview of the IBM 2008 TRECVID High-level Feature Detection System.

3 High-level Feature Detection

Figure 1 illustrates the IBM high level feature detection system. Our current system includes multiple base and meta-level learning algorithms such as random subspace bagging with SVMs, principal-component semi-supervised support vector machines, cross-domain learning with web data, text search, and so on. It also consists of different fusion strategies and cross-concept learning components for leveraging multi-modal and multi-concept relationship. We continue improving the general learning algorithms that have been proven to be successful[NNT05] to accommodate a much larger set of visual features, and re-implement the learning algorithms on a distributed learning system. The details of these components are explained in details in the rest of this section.

3.1 Baseline: Random-Subspace Bagging with SVMs

This year two shared versions of annotations are officially recommended by NIST, where one is from the collaborative annotation forum organized by LIG [AQ07] and the other is provided by the MCQ-ICT-CAS team [ea08]. In view of the noticeable difference between these two annotations, we use their union to serve as the baseline relevance judgment. In the learning process, the development data are randomly partitioned into three collections: 60% as the training set, 10% as the validation set, and 30% as the held-out set. Most of our following algorithms are learned on the training and validation data, while the fusion strategies are determined based on the held-out data.

One of the major changes we made this year is to switch the baseline methods from traditional SVMs to an ensemble approach called "random subspace bagging" (RS-Bag), which enjoys several advantages over SVMs such as being highly efficient in learning/prediction, robustly performed with theoretical guarantee, and easy to parallelize on a distributed learning system [CHL$^+$07]. The details of the random subspace bagging approach is shown in Algorithm 1, which contains multiple improvements over the version we used last year. This algorithm first works on the training collection. It learns $F \cdot T$ base models using SVMs with the RBF kernel. Each model is constructed based on N_d balanced bootstrap samples from the positive and the negative data, together with Mr_f random samples from the feature dimensions for each type of visual features. Both sampling parameters are determined by the input parameters. The default parameters for N_d is 500 and r_f is 1. Each model is associated with its 3-fold cross validation performance, where average precision is chosen in this case.

To minimize the sensitivity of the parameters for each base model, we choose the SVM parameters based on a grid search strategy. In our experiments, we build the SVM models with different values on the RBF kernel parameters, the relative cost factors of positive vs. negative examples, the feature normalization schemes, and the weights between training error and margin. The optimal learning parameters are selected based on the performance measure on the same 3-fold cross validation on training data. For each low-level feature, we select one optimal configuration to generate the concept model.

After all base models are generated, the algorithm iterates over the base models with either *Greedy* or *Combined* strategy, and incrementally combines them into a series of composite classifiers. The greedy strategy simply ranks the base models with a decreasing cross-validation performance on training data. In contrast, the combined strategy attempts to find the next base model that can achieve the most improvement on validation data after combination. Finally, it outputs the classifier with the highest performance on the validation data. To generate the baseline run, we compute multiple configurations of the RS-Bag algorithm such as selecting greedy/combined strategies and tuning data/feature sampling ratios. For each indi-

Algorithm 1 The random subspace bagging (RSBag) algorithm for high-level feature detection.

Input: concept l, training data $\mathcal{T} : \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1...N}$, validation data $\mathcal{V}$, maximum number of base models T for each feature type, number of feature types F, number of sample data N_d, feature sampling ratio $r_f (\leq 1)$.

1. For f = 1 to F,
   For t = 1 to T,
   (a) Take a bootstrap sample X_+^t from positive data $\{\mathbf{x}_i\}$, $|X_+^t| = N_d$;
   (b) Take a bootstrap sample X_-^t from negative data $\{\mathbf{x}_i\}$, $|X_-^t| = N_d$;
   (c) Take a random sample F^t from the feature f with indices $\{1, ..., M\}$, $|F^t| = Mr_f$;
   (d) Learn a base model $h^{ft}(\mathbf{x})$ using X_+^t, X_-^t and F^t. SVMs with RBF kernel are used and the model parameters are chosen based on 3-fold cross validation;
2. For n = 1 to $F \cdot T$,
   (a) Select the n^{th} base model $h_n(\mathbf{x})$ with either *Greedy* or *Combined* strategy;
   (b) $F^t(\mathbf{x}) \leftarrow F^t(\mathbf{x}) + h_n(\mathbf{x})$;
   (c) Evaluate the composite classifier performance on validation data.
3. Output the best classifier on validation data.

vidual concept, we select the models with the best configuration on the held-out data to produce baseline detection results.

3.2 Text search

For the text-based concept detection, we leveraged our speech-based retrieval system [CHL+07] to generate search-based results for each concept. Specifically, we manually created text-based queries for each concept based on interactive search results on the development set. We used the automatic query expansion and word suggestion capabilities of our interactive system to identify relevant keywords for each concept and to optimize the performance of the retrieval system. The queries were then applied automatically on the test set and the search results were used as our text-based concept detection run.

3.3 Principle Component Semi-Supervised SVM

Small sample learning is one of the central issues in semantic concept classification of images/videos. The amount of available unlabeled testing data is large and growing, but the amount of labeled training data remains relatively small. Furthermore, the dimensionality of the low-level feature space is generally very high, the desired classifiers are complex and, thus, small sample learning problems emerge.

There are two primary techniques for tackling the above issues. *Semi-supervised learning* is a potent method to incorporate knowledge about unlabeled test data into the training process so that the learned classifier can be better leveraged for classifying test data [Zhu05]. *Feature subspace learning* is a popular approach to learn a good feature subspace for capturing the underlying data manifold and achieving better classification performance [SSK98].

We address both issues of feature subspace learning and semi-supervised classification. We propose an algorithm, namely *Principle Component Semi-Supervised SVM* (PCS^{3}VM), to jointly learn an optimal feature subspace as well as a large margin SVM classifier in a semi-supervised manner. A joint cost function is optimized to find a discriminative feature subspace as well as an SVM classifier in the learned feature subspace. The following highlight some aspects of the proposed algorithm:

1. The target of PCS^{3}VM is both feature subspace learning and semi-supervised classification. A feature subspace is jointly optimized with an SVM classifier so that in the learned feature subspace the labeled data can be better classified to have optimal margin, and the variance of both labeled and unlabeled data can be preserved.
2. PCS^{3}VM can naturally extend to classifying novel unseen test examples, and can be learned in the original feature space or in a Reproducing Kernel Hilbert Space (RKHS). In other words, we formulate

a kernel-based PCS^{3}VM which permits the method to handle real applications where nonlinear classification may be necessary.

In terms of speed, our algorithm is fast in the testing stage, with complexity similar to that of the standard SVM classification. In the training stage, two-step iterative optimization process is a little time consuming: the generalized eigenvalue problem in step 1 has a time complexity of O(n^3) (where n is the total number of labeled and unlabeled data). It can be further reduced by exploiting the sparse implementation. For the SVM learning in step 2, the standard quadratic programming optimization for an SVM is O(n_L^3) (where n_L^3 is the number of labeled training data).

3.4 Learning from Web Domain

This year, our main emphasis was on leveraging targeted downloads from various web resources. Online channels provide rich sources for multimedia training data. User-generated and user-tagged multimedia content can help us understand the visual semantics through crowdsourcing, and improve the process of high-level feature detection in terms of cross-domain applicability and robustness.

How well can a semantic classifier perform, if only textual definition of a visual concept is available, no knowledge of the training or the testing data? People tag their image and photo collections based on the perceived semantics. Given the specific text query (i.e. "Hand"), web search engines give us the best text match. Using TRECVID high-level feature description, we formulated 2-5 queries per each topic, and retrieved web images from various user sites. Concept "Hand" is defined as "015 Hand: a close-up view of one or more human hands, where the hand is the primary focus of the shot." We formulated 5 text queries: (1) hand, (2) hand gesture, (3) hand in hand, (4) wave goodbye, and (5) handshake, and downloaded top 100-200 images from 2 sites for each of the queries. We repeated the process for all topics and ended up with 30000 images to annotate for 20 topics. Team members annotated sets for training for each concept from this pool of web images.

How well the semantic concepts defined by a web user match TRECVID semantic concepts? We have used the training set to select the best normalization and fusion (MAX, AND, OR) of the visual semantic concepts trained solely on the web data. This helped us solve the ambiguities not captured by the textual definition of a concept. The best performance for an Emergency Vehicle concept was a definition as a land vehicle, more specifically Ambulance Vehicle or Fire Truck or Police Vehicle. Individual concepts were trained the same as the Type A concepts.

How much are we learning the semantics of the training data, and how much we are leaning the crowdsourced semantics? TREVID 2008 training and testing data are different domain than the web images. Concept of a "dog" trained on the training set, learned the dog from the commercial that happen to appear in the testing set. For the concepts where the system was learning the semantics in terms of the domain and application, web-train classifiers did not perform well. On a more universal set of concepts, models trained exclusively on the external data performed better than the baseline, as show in Table 1.

The additional development data collected from the web domain are shown to be informative on a number of the concepts, although its average performance is not comparable with baseline. Visual semantic concepts trained on the web data are better than the baseline for 5 out of 20 concepts, and perform close to a baseline for one of them (Boat or Ship). For these 6 concepts, textual definition of the concept matched both the definition of the concept as seen in TRECVID training data and in the crowdsourced image set.

3.5 Multi-concept Learning

Multi-concept learning is to estimate the label $y_c \in \{1, 0\}$ for concept c from a collection of relevance scores for concepts that are related to the concept c, denoted as $\mathbf{x} = [x_1, \ldots, x_M]$. Multi-concept models need to account for two factors of correlations and uncertainty. The first factor is ontological relationship, e.g., a *bus* is likely to be *outdoors* and unlikely to be in an *office*. The second factor is detector performance, e.g., knowing both the presence/absence of *people* and *desert* may help us decide if it is *military* setup, but *people* detectors are generally more reliable and hence shall be given more weight.

We use the Naive Bayes model to estimate the class-conditional probability $P(x_i|y_c)$, in order to find the maximum-posterior class $\hat{c} = \arg\max_c P(y_c|x)$ based

Concept	Run Type A (baseline)	Run Type A (best)	Run Type C
Airplane flying	0.0919	0.2775	0.1339
Boat Ship	0.1568	0.189	0.1411
Bus	0.0172	0.0306	0.029
Demonstration or Protest	0.0099	0.0817	0.0178
Emergency Vehicle	0.0076	0.0251	0.062
Mountain	0.038	0.0671	0.0487

Table 1: Selected Web Concepts

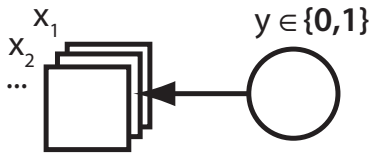

Figure 2: Naive Bayes model for multi-concept learning.

on the Bayes rule (Figure 2). This algorithm models the pair-wise correlations between the target concept c and each input concept $i, i = 1, \ldots, M$. The maximum-likelihood estimate of the class-conditional probability $P(x_i|y_c)$ is obtained by averaging the confidence over all the training data on a given target concept y_c. Because naive Bayes models assume that input x_i are conditionally independent given label y_c, we can factorize the joint class-probabilities and use these estimates to obtain the log-likelihood ratio L_c as shown in Equation (2). It has been shown [XYY08] that this model is robust to highly correlated inputs than logistic regression or SVM.

$$P(y_c|x_{1:M}) \propto P(y_c) \prod_{i=1:M} P(x_i|y_c) \quad (1)$$

$$\begin{aligned} L_c &= \log \frac{P(y_c=1|x_{1:M})}{P(y_c=0|x_{1:M})} \quad (2) \\ &= \log \frac{P(y_c=1)}{P(y_c=0)} + \sum_{i=1:M} \log \frac{P(x_i|y_c=1)}{P(x_i|y_c=0)} \end{aligned}$$

In our implementation, each input observation x_i is sigmoid-normalized and uniformly quantized into 15 bins, and the maximum likelihood estimates of $P(x_i|y_c)$ is smoothed by the prior of x_i as $\hat{P}(x_i|y_c) = (1 - \alpha)P(x_i|y_c) + \alpha P(x_i)$, with $\alpha = 0.06$. The resulting likelihood scores L_c are then used to re-rank the original prediction score with simple averaging. For each TRECVID'08 target concept c, we use the rest of the 20 target concepts, along with a selected set of 200+ concepts trained from web data as the input pool.

3.6 Fusion Methods

We applied ensemble fusion methods to combine all concept detection hypotheses generated by different modeling techniques or different features. In particular, we performed a grid search in the fusion parameter space to select the optimal fusion configuration based on validation data. Fusion parameters include a score normalization method and a score aggregation method. For score normalization methods, we considers using both raw SVM scores, statistical normalization, range normalization and sigmoid normalization. The fusion methods we considered include simple average and weighted average fusion. The fusion strategy are automatically chosen based on the validation performance.

To generate the runs, we detect the concepts first using the following individual approaches and then proceeded to fuse resultant detection results with described fusion techniques.

1. Baseline: Random subspace bagging learned from training set;
2. Text: Text retrieval with manually defined keywords for each concept;
3. PCS^3VM: Principle component semi-supervised SVMs learned from training set;
4. Web: Random subspace bagging learned from Web data;
5. MCL: Multi-concept relationship learning on baseline models and 200 web models;

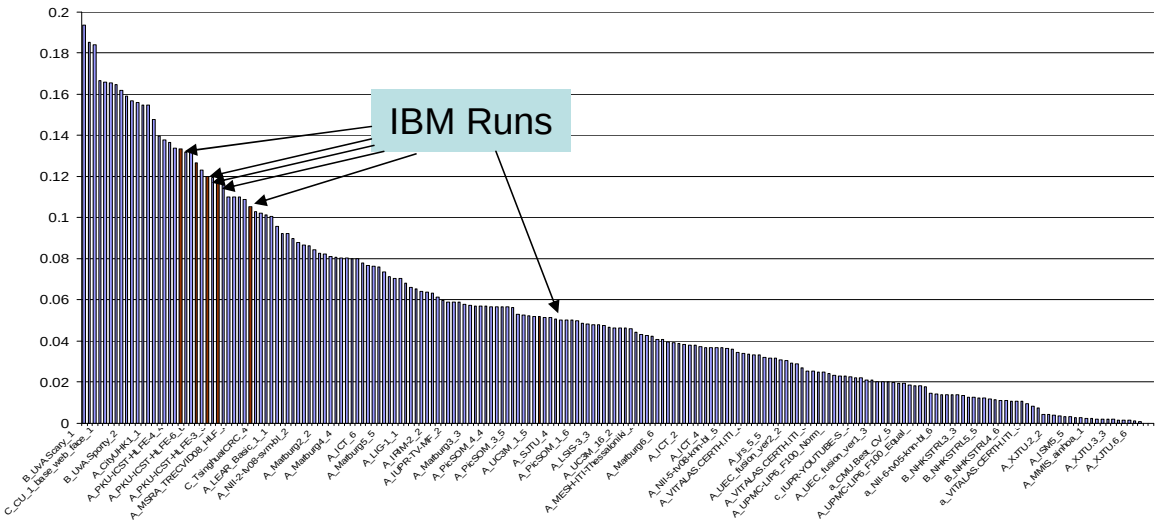

Figure 3: Retrieval performance of IBM high level feature runs in context of all the submissions (including Type A/B/C) using the mean inferred average precision measure.

Description	Run	Type	MAP	Total Time (hours)
Baseline	A_ibm.Baseline_5	A	0.10515	12
PCS^3VM	-	-	0.07500	200
Baseline + PCS^3VM	A_ibm.BaseSSL_4	A	0.11820	-
Text	-	-	0.02300	0.3
Baseline + PCS^3VM + Text	A_ibm.BaseSSLText_3	A	0.12675	-
Web	C_ibm.CrossDomain_6	C	0.05185	48
MCL on Baseline, PCS^3VM, Web	C_ibm.BNet_2	C	0.11995	-
Best Overall Run	C_ibm.BOR_1	C	0.13365	-

Table 2: IBM TRECVID 2008 High level Feature Detection Task – Submitted and Unsubmitted Runs

6. BOR: Best overall run by compiling the best models based on heldout performance for each concept.

3.7 Submitted Systems and Results

We have generated multiple runs of detection results based on the approaches presented before. A number of runs are submitted to NIST for official evaluation with their submission name shown, and all of the remaining runs are evaluated using the ground truth provided by NIST. The mean inferred average precision is used as the measure of the overall performance of the systems. Table 2 lists the performance of the submitted and unsubmitted runs, and Figure 3 summarizes the detection performance of IBM high level feature runs in context of all the submissions. The baseline run offers a reasonable starting performance for the following combination. Although the PCS^3VM approach does not outperform the baseline method on average, it achieves better results on the concepts of "nighttime" and "driver". After combined with the baseline run, it obtains 12% performance gain in terms of MAP on a relative basis. By introducing complementary information beyond visual features, text retrieval results in another 7% improvement over visual-only detection. The additional development data collected from the web domain are also proven to be informative on

a number of the concepts, such as "Emergency Vehicle" (AP:0.062) and "Bus" (AP:0.04). By fusing the web training data with other baseline methods using multi-concept learning approaches, we can improve the performance by another 2%. Good models from web data, such as *airplane flying* or *kitchen* has notably helped in the multi-concept fusion process. Finally, the best overall run brings consistent improvement in MAP over runs of all flavors and raise the MAP to 0.134.

4 Interactive Search

The IBM IMARS Multimedia Analysis and Retrieval System was used for our interactive search runs. This system conceptually consists of two parts: the visual and semantic analytics that were extracted from the test dataset and ingested into the system to assist searching; the interactive interface that takes in user-defined queries, supports multiple modes of interactions, and records the final results. Figure 4 contains a snapshot of the IMARS interface with TRECVID-2008 test dataset.

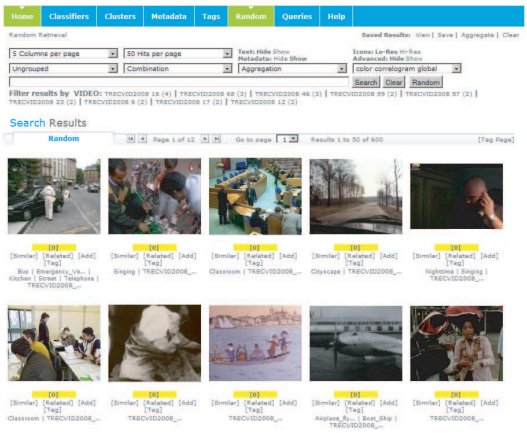

Figure 4: IMARS system for interactive search.

4.1 Visual and Semantic Analytics

The IMARS system have included the following four types of visual and semantic indexes to help searching:

- Low-level image features extracted from each keyframe. Six different features are chosen from the concept detection feature pool (Sec. 2): *color correlogram global*, *color histogram global*, *color moments grid*, *wavelet texture grid*, *edge histogram layout* and *thumbnail vector global*. These features are diverse enough so that nearest-neighbor search would be more effective for different types of visual scenes.

- Visual clusters and significance-based labeling. Kmeans clustering was performed on the test videos with the *color histogram global* feature.

- Visual near-duplicates, generated with the *thumbnail vector global* feature with a tight clustering threshold.

- Visual semantic concept scores.

4.2 Interactive System Features

The IMARS system is capable of two different modes of searching and ranking based on the available analytics, it further supports five other modes of interaction including filtering, tagging, grouping, combining different queries and recording interaction history. It analyzes the video content in an off-line process that involves audio-visual feature extraction, clustering, text analysis and concept detection, as well as speech indexing.

The search algorithms return a ranked list over the pool of available shots based on one or more scoring algorithm. Text search on machine-translated speech transcript is based on the JuruXML semantic search engine [MMA+02], and is the same as last year. We indexed the ASR/MT transcripts corresponding to each subshot from the shot reference provided by the ATT System [LGZ+07]. Temporal expansion to adjacent subshots and query expansion are performed at indexing and retrieval time, respectively. Detailed descriptions can be found in our previous report [CHL+07]. Nearest neighbor search, in which the shots are returned based on their Euclidean distance from the query. This method can be applied to image features and semantic concept score vectors.

Aside from ranking, the result list can interact with any categorical attributes with filtering or grouping. Such at-

tribute include visual concepts, clusters, near-duplicates, video or program information. Filtering prunes the list and keeps the subset with one common attribute value, whereas grouping rearranges the list according to the attribute of interest. Different queries can be combined using weight averaging or boolean combination, available through either the simple search box or a list of pull down menus. Tagging provides a convenient way of saving intermediate results, and relevance feedback can also be performed on the already selected positive shots to increase recall. The system also performs query suggestion, by either presenting a context list in the search box on-the-fly, or looking through the result list and presents the user with statistically significant terms or attributes.

A thin web client integrates all the search functionalities above and allows the user to access the system from anywhere with a network connection.

4.3 Runs and Results

We completed two interactive search runs using different visual semantic vocabulary, as shown in Table 4.3. There are eight searchers total, each completing about six randomly chosen queries in either run. The search strategies most use was to identify one or more relevant examples using visual concepts, text, or clusters, and then drill down for more similar examples to increase recall. Overall the performance variations across queries were greater than that across searchers or across different semantic indexes. The infMAP of the two runs are similar (with TypeC slightly better), and we would need more query before making a comparative claim.

5 Conclusion

IBM Research team participated in the TREC Video Retrieval Track Concept Detection and Search tasks. In this paper, we have presented preliminary results and experiments for both tasks. More details and performance analysis on all approaches will be provided at the workshop, and in the final notebook paper.

6 Acknowledgments

This material is based upon work supported by the US Government. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

References

[AQ07] S. Ayache and G. Quenot. Evaluation of active learning strategies for video indexing. In *Proceedings of Fifth International Workshop on Content-Based Multimedia Indexing (CBMI'07)*, 2007.

[CHL+07] M. Campbell, A. Haubold, M. Liu, A. P. Natsev, J. R. Smith, J. Tešić, L. Xie, R. Yan, and J. Yang. IBM research trecvid-2007 video retrieval system. In *Proceedings of NIST TREC Video Retrieval Evaluation*, Gaithersburg, MD, 2007.

[ea08] S. Tang et. al. Trecvid 2008 high-level feature extraction by MCG-ICT-CAS. In *Proceedings of NIST TREC Video Retrieval Evaluation*, Gaithersburg, MD, 2008.

[LGZ+07] Z. Liu, D. Gibbon, E. Zavesky, B. Shahraray, and P. Haffner. A fast, comprehensive shot boundary determination system. In *Proc. of IEEE International Conference on Multimedia and Expo*, 2007.

[MMA+02] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. Maarek, and A. Soffer. JuruXML—an XML retrieval system. In *INEX '02*, Schloss Dagstuhl, Germany, Dec. 2002.

[NNT05] A. Natsev, M. R. Naphade, and J. Tešić. Learning the semantics of multimedia queries and concepts from a small number of examples. In *ACM Multimedia*, Singapore, November 2005.

[SSK98] B. Scholkopf, A. Smola, and K.R.Muller. Nonlinear Component Analysis as a Kernel

Run ID	Run Description	Run Type	Run MAP
I_A_2_IBM.SearchTypeA_2	Type-A interactive run with 20 semantic concepts	A	0.0636
I_C_2_IBM.SearchTypeC_1	Type-C interactive run with 96 semantic concepts	C	0.0643

Table 3: Mean inferred average precision scores for IBM interactive search submissions.

Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[XYY08] Lexing Xie, Rong Yan, and Jun Yang. Multi-concept learning with large-scale multimedia lexicon. In *IEEE Interational Conference on Image Processing (ICIP)*, San Diego, California, October 2008.

[Zhu05] X. Zhu. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Science, University of Wisconsin-Madison, 2005.