Sequence-based Kernels for Online Concept Detection in Video

Werner Bailer
JOANNEUM RESEARCH Forschungsgesellschaft mbH
DIGITAL – Institute for Information and Communication Technologies
Steyrergasse 17, 8010 Graz, Austria
werner.bailer@joanneum.at

ABSTRACT

Kernel methods, e.g. Support Vector Machines, have been successfully applied to classification problems such as concept detection in video. In order to capture concepts and events with longer temporal extent, kernels for sequences of feature vectors have been proposed, e.g. based on temporal pyramid matching or sequence alignment. However, all these approaches need a temporal segmentation of the video, as the kernel is applied to the feature vectors of a segment. In (semi-)supervised training, this is not a problem, as the ground truth is annotated on a temporal segment. When performing online concept detection on a live video stream, (i) no segmentation exists and (ii) the latency must be kept as low as possible. Re-evaluating the kernel for each temporal position of a sliding window is prohibitive due to the computational effort. We thus propose variants of the temporal pyramid matching, all subsequences and longest common subsequence kernels, which can be efficiently calculated for a temporal sliding window. An arbitrary kernel function can be plugged in to determine the similarity of feature vectors of individual samples. We evaluate the proposed kernels on the TRECVID 2007 High-level Feature Extraction data set and show that the sliding window variants for online detection perform equally well or better than the segment-based ones, while the runtime is reduced by at least 30%.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

General Terms

Algorithms

1. INTRODUCTION

Concept and event detection based on (audio)visual features has become an active research area, also thanks to benchmarking initiatives such as the TRECVID [12] High-level Feature Extraction (HLFE)/Semantic Indexing (SIN) tasks. While early approaches only use visual features from key frames, the temporal dimension has been increasingly addressed, especially for concepts representing dynamic actions (see proceedings of the TRECVID workshop series for an overview).

In this paper, we are interested in the problem of online detection of segments representing a concept in a video segment S. The segment is represented by m samples s_i (e.g. frames, key frames). Each of these samples is described by a feature vector x_i, consisting of arbitrary features of this sample (or a temporal segment around this sample). In order to represent the video segment, we concatenate the individual feature vectors to form a feature vector $X = (x_1, \ldots, x_n)$ of the segment. Clearly, not every segment has the same length and/or consists of the same number of samples, thus the lengths of the feature vectors of different segments will differ. We thus need to be able to determine the similarity between such feature vectors with different lengths.

Kernel methods, most notably Support Vector Machines (SVMs), have been widely applied to concept detection problems, also due to the availability of toolkits such as LibSVM [4]. Sequence-based kernels, i.e., kernel functions that are able to determine the similarity of sequences of feature vectors, have been proposed and we review some of them in detail below. Experiments have shown that they outperform kernels matching the individual feature vectors of the samples of a segment independently (see works discussed in Section 2 for the results).

In the online detection case, we encounter a stream of input samples s_i, and we need to process samples in a time window, i.e., a sequence of feature vectors $X = (x_1, \ldots, x_\tau)$. In the next step, we need to process the feature vectors from the next sliding window position $X' = (x_{1+\delta}, \ldots, x_{\tau+\delta})$. The application of sequence-based kernels to online detection is not straight forward for the following reasons. All kernels for sequences of feature vectors need some segmentation of the content (e.g., shots, subclips), which does not exist for a live stream. One cannot simply choose arbitrary segments in the input stream, as many of the kernels will not provide satisfactory results if the model and the input samples are not temporally well aligned. Many of the proposed sequence-based kernels determine some optimal alignment between the elements of the sequences to be matched. Thus, the value of the kernel function for a position of the sliding window cannot be easily derived from that of the previous position, but requires re-evaluation of the kernel,

AIEMPro'11, December 1, 2011, Scottsdale, Arizona, USA.

which is an extremely inefficient and impractical solution. Some sequence-based kernels require a specific base distance between the feature vectors of the individual samples. In order to be generally applicable, we are interested in kernels that allow an arbitrary kernel function to be used to determine the similarity of feature vectors of two samples.

The contributions of this paper are the following: We analyze kernels for sequences of feature vectors w.r.t. their applicability in the online detection case. We then propose an approach for efficiently calculating the kernel function for an updated value for the current sliding window position based on information about matching elements from previous positions. The approach is described for three types of kernels and the runtime and memory complexity of these kernels are analyzed.

The rest of this paper is organized as follows. Section 2 reviews prior work on sequence-based detection of concepts and events. In Section 3 we analyze some candidate sequence-based kernels in more detail, and describe in Section 4 how three of these kernels can be adapted for efficient online prediction. Section 5 presents evaluation results and Section 6 concludes the paper.

2. RELATED WORK

In [14] a sequence alignment kernel for matching trajectories is proposed. Segments of trajectories are labeled to obtain both a symbol and a feature sequence. The sequence alignment kernel is defined as the tensor product of the kernels evaluated on the symbol and feature sequences. The authors of [10] propose a temporal kernel for concept classification. They train a HMM for each concept and define a distance between HMMs based on the Kullback-Leibler divergence, which is plugged into a Gaussian kernel.

Several approaches based on the idea of the pyramid match kernel have been proposed. The original pyramid match kernel [7] partitions the feature space in each of the dimensions of the input feature vector. Its efficiency advantage is based on avoiding explicit distance calculation, but only counting elements that end up in the same bin of the pyramid. This assumes that L_1 distances can be applied to the feature vectors. The vocabulary guided pyramid matching approach proposed in [8] addresses this problem, as it uses a clustering step to construct the pyramid, allowing the use of an arbitrary distance measure.

The approach has been extended to spatio-temporal matching in [5], using sets of clustered SIFT and optical flow features as local descriptors. Their approach is similar to spatial pyramid matching proposed in [9], which applies pyramid matching only to the image space, but uses clustering in the feature space.

Another temporal matching method based on the pyramid match kernel is described in [15]. Temporally constrained hierarchical agglomerative clustering is performed to build a structure of temporal segments. The similarity between segments is determined using the earth mover's distance and the pyramid match kernel is applied to the similarities on the different hierarchy levels. This approach explicitly assumes that the temporal order of the individual subclips is irrelevant (as is e.g. the case for news stories). The temporal order within the clips is aligned using linear programming.

The authors of [16] use the Levenshtein distance between sequences of clustered local descriptors for classification of still images. Recently, a kernel for event classification based on sequences of histograms of visual words has been proposed [3]. The authors consider different similarity measures between the histograms and use them instead of symbol equality in the Needleman-Wunsch distance and plug it into a Gaussian kernel. In [2] a kernel based on longest common subsequence (LCSS) matching of sequences has been proposed. An arbitrary kernel can be plugged in to determine the similarity between two elements of the sequences, and the kernel value is determined as the normalized sum of the similarities along the backtracked longest common sequences.

All of these methods are designed to work on pairs of segments of input data, and thus cannot be directly applied to online concept detection.

3. KERNELS FOR SEQUENCES OF FEATURE VECTORS

In the following, we review some kernels for sequences of feature vectors and adapt the formulation of the kernel functions to our requirements. We denote as $\mathcal{U} = \{U_1, \ldots, U_K\}$ the set of support vectors of the model of a concept. Each U_k consists of feature vectors $U_k = (u_1, \ldots, u_n)$ of the samples in the segment, containing a sequence describing an instance of the concept. The term sequence denotes a possibly non-contiguous subsequence. As we intend to support arbitrary ground distances between the feature vectors of the input samples, we use a kernel for matching the feature vectors of elements of the feature sequences, denoted as $\kappa_f(x_i, u_j)$.

3.1 Temporal Pyramid Match

In order not to constrain the choice of distances in the feature space, pyramid matching is only applied to the temporal domain, in a similar way as proposed for spatial [9] or spatio-temporal [5] pyramid matching. As we do not perform clustering of the feature vectors in advance, we define a threshold θ to determine whether two feature vectors match or not. The temporal pyramid match kernel is then defined as $\kappa_{\text{TPM}}(X, U) = \sum_{l=1}^{L} \frac{1}{2^{L-l+1}} \Gamma^l + \frac{1}{2^L} \Gamma^0$, where L is the number of pyramid levels ($L = \lceil \log_2 \max(|X|, |U|) \rceil$) and Γ^l is the number of elements matching on level l, i.e., the number of elements falling into the same temporal bin on level l for which $\kappa_f(x_i, u_j) \geq \theta$.

3.2 (Weighted) All Subsequences

The all subsequences kernel [11] is defined as $\kappa_{\text{ASS}}(X, U) = \sum_{\sigma \in \Sigma^*} \phi_\sigma(X) \phi_\sigma(U)$, where σ denotes a sequence from the possible set of sequences Σ^*, and $\phi_\sigma(X)$ counts the number of times σ occurs as a subsequence of X. Clearly, $\phi_\sigma(X)\phi_\sigma(U)$ is only non-zero, if σ is a subsequence of both X and U. Thus a dynamic programming approach can be applied to determine the set of *common* subsequences [11]. The approach is based on the observation that the kernel can be defined recursively. This kernel assumes sequences of discrete values, which does not generally hold for feature vectors. Thus we introduce a threshold θ and consider elements in the sequence as matching, iff $\kappa_f(x_i, u_j) \geq \theta$. The kernel is then defined as

$$\begin{aligned} &\kappa_{\text{ASS}}(X, \emptyset) = 1, \\ &\kappa_{\text{ASS}}((x_1, \ldots, x_{n-1}), U) = \kappa_{\text{ASS}}((x_1, \ldots, x_{n-2}), U) + \\ &\sum_{k:\kappa_f(x_{n-1}, u_k) \geq \theta} \kappa_{\text{ASS}}((x_1, \ldots, x_{n-2}), (u_1, \ldots, u_{k-1})). \end{aligned} \quad (1)$$

where $\emptyset$ denotes the empty sequence. The kernel value is normalized by the possible maximum number of common sequences of X and U. In addition, we want to weight the result by the similarities of the matching elements. This can be done by summing $\kappa_f(x_i, u_j)$ for all elements for which $\kappa_f(x_i, u_j) \geq \theta$ and normalizing. We denote the kernel including the weighting as $\kappa_{\text{WASS}}(\cdot)$.

3.3 (All) Longest Common Subsequence(s)

A kernel based on the longest common subsequence (LCSS) algorithm has been proposed in [2]. In contrast to the ASS kernel it only considers a single or a few longest common subsequences. This kernel already allows plugging in any kernel for measuring the distance between the feature vectors of the samples of the two sequences, and includes the similarities in the result of the kernel. The kernel uses a recursive definition of LCSS and a threshold θ to decide if two feature vectors are considered as matching. The kernel function to determine the length of the single longest common subsequence is given as $\kappa_{\text{LCSS}} = \text{LCSS}(X, U)$. Similarity weighting can be achieved by performing backtracking of the longest sequence, summing the values of $\kappa_f(\cdot)$ of the matches and normalizing.

In [2] the authors propose to consider all subsequences ending in the last element of either of the two sequences:

$$\kappa_{\text{ALCSS}} = \begin{array}{l} \sum_{i=m}^{1} \text{LCSS}((x_1, \ldots, x_i), U) + \\ \sum_{j=n-1}^{1} \text{LCSS}(X, (u_1, \ldots, u_j)). \end{array} \quad (2)$$

This requires backtracking of all sequences ending in the last element of either X or U, i.e., $O(n^2)$ backtracking steps.

The result of the kernel function is normalized to account for sequences of different lengths.

4. ADAPTATION TO SLIDING WINDOWS

In order to efficiently evaluate the kernel function in the online detection case, we need to adapt the way the kernel is evaluated in the prediction step. The training step is left unmodified for all the kernels. In the case of supervised learning the ground truth defines the segments to be used in training. For the sequence alignment kernels, such segments can even be obtained in a semi-supervised learning setting, as the boundaries of matching segments are a result of these algorithms.

Given the result for the samples of a sliding window $X = (x_1, \ldots, x_\tau)$, we want to evaluate $X' = (x_{1+\delta}, \ldots, x_{\tau+\delta})$. The size of the sliding window needs to be at most the number of elements of the longest support vector $|U_{max}| = \max_{k=1}^{K} |U_k|$. Kernels based on sequence alignment can handle sliding window sizes $< |U_{max}|$, as they support partial matches. A lower bound for the size of the sliding window is the shortest support vector $|U_{min}| = \min_{k=1}^{K} |U_k|$ of the model.

The different kernels impose different constraints on the sampling structure of training and test data. Different sampling rates are supported by all the kernels discussed in the previous section. Changes in the frame rate may require a change of the size of the sliding window, i.e., when the sampling rate of the test data is higher than that of the training data, the sliding window duration for matching has to be adapted to ensure that the duration of the concept is adequately covered. The kernels based on sequence alignment methods also support irregular sampling on one or both of the data sets, and are more robust when evaluating sequences which only partially represent the concept of interest.

The temporal offset δ between two positions of the sliding window can be in the range of $1 \leq \delta \leq |U_{min}| - 1$ samples. The choice of this number has an impact on the latency of the detection as well as on the effort for re-evaluation in each step. If we choose $\delta = 1$ (remember that one sample does not necessarily mean one frame, but represents typically a longer time span), we only need to match the last $n = |U_k|$ samples for the k^{th} support vector, as all other possible matches are already covered by other window positions. This saves time compared to the segment-based variant for long segments, especially for rather short events, where the support vectors are shorter than the typical segment (e.g., shot).

To evaluate the kernel function for one position of the sliding window, we need to evaluate $\kappa_f(\cdot)$ for each pair of elements (x_i, u_j), which has a runtime complexity of $O(n^2 T_{\kappa_f})$ for each support vector, where T_{κ_f} is the time needed to evaluate $\kappa_f(x_i, u_j)$. Once this kernel matrix of κ_f is available, the respective sequence-based kernel can be applied.

We can reduce the runtime by storing information from the kernel evaluation at the previous sliding window position. Note that we need to keep data for each support vector of the current model. Obviously, when moving the sliding window by one element, only the distances between the new element x_τ and the current support vector $\tilde{U}$ need to be calculated. This reduces the runtime to $O(nT_{\kappa_f})$ per support vector, but requires storing the kernel matrix (memory complexity $O(n^2)$).

The memory complexity can be further reduced by storing only the matches from the past $n - 1$ samples for each support vector. For each match, we store the index j of the matching $\tilde{u}_j$ and the corresponding result of $\kappa_f(\cdot)$. In the worst case, i.e., when matching two similar and nearly constant sequences, the memory complexity is still $O(n^2)$. However, in most cases we expect only few matches per element of the feature vector, so that the memory complexity is $O(cn), 0 \leq c \ll n$ in the average case.

The runtime and memory complexities of the different kernels are listed in Table 1. In the following, we discuss the specific approaches for the different types of kernels.

4.1 Temporal Pyramid Match

The pyramid match approach requires that each sample falls into its own bin on the pyramid level with the finest resolution [7]. Thus, moving the sliding window by one temporal sample can completely change the pyramid structure. This means that we need to keep all matches for the time window in order to construct the pyramid for a new sliding window position.

As we have the full set of matches, it is possible to determine the matching elements in each of the bins. Based on the temporal distance between the position of the matching element in the feature vector and the index of the stored matches, the pyramid level on which the elements match can be efficiently determined. The runtime of the algorithm is linear in the number of matches, i.e., $O(cn)$.

4.2 (Weighted) All Subsequences

The definition of the all subsequences kernel is recursive, so including a new sample into the temporal sliding window is trivial. However, removing the outdated sample on the

other end of the temporal range is not. As the information about matches is accumulated over both dimensions (i.e., the temporal extent of both sequences), there is no efficient way to remove a sample without propagating the change through the whole time window. It is thus efficient to adopt a solution like for pyramid matching, where the data structure is reconstructed from the list of stored matches.

The algorithm for non-contiguous subsequences ([11], Alg. 11.20) can be adapted to processing a sliding window as follows. The typical implementation using a dynamic programming table can be modified to a two-column array representing the accumulated values for two adjacent samples. This array is updated over the sliding window using the stored matches. The runtime complexity is $O(n^2)$ as it is necessary to iterate over all positions in the (virtual) dynamic programming table. Weighting and normalization can be performed in a similar way as in the segment-based case.

4.3 (All) Longest Common Subsequence(s)

The LCSS based kernel is similar in structure to the ASS kernel, thus the same issues apply. It is also usually implemented using dynamic programming, thus the algorithm is the same as ASS, only with a different summation. In addition, backtracking needs to be performed as a post-processing step, which increases the total runtime to $O(n^2 + 2n^2)$. The backtracking step is the same as in the segment-based algorithm. Like for the all subsequences kernel, weighting and normalization can be performed similar as in the segment-based case.

5. EVALUATION

We perform evaluation on the TRECVID [12] 2007 High-level Feature Extraction (HLFE) data set (50 hours training data and 50 hours test data), using the ground truth from the collaborative annotation effort [1] for the training set and the truth judgments provided by NIST for the test set. These annotations are available for 20 concepts. For the sequence-based kernels we need more key frames than those provided in the TRECVID reference key frame set. We use a denser key frame set provided by the K-Space project [13], which has at least a few key frames for each shot and more than 100 key frames for the longest shots. In total, the training set has 11.8K frames and the test set has 82.7K frames (the set of key frames used for training is smaller as it uses a balanced set of positive/negative samples for the 20 concepts). All experiments are performed using this set of key frames.

We extract the following features from the key frames: ColorLayout, using the 3 DC coefficients, 5 AC coefficients for the Y channel and 2 AC coefficients each for the Cb/Cr channels (12 element vector), DominantColor with up to 3 dominant colors (12 element vector), ColorStructure (32 bin histogram) and EdgeHistogram (80 bin histogram). As kernel function $\kappa_f(\cdot)$ to determine the similarity between the feature vectors of two key frames we use the MPEG-7 kernel proposed in [6]. We also use the method proposed in [2] to determine the relative weights of the different MPEG-7 features. For the kernels that need a similarity threshold, we set $\theta = 0.03$. In our experiments we find that the mean number of matches per support vector is 9.67 (median 3.00), this corresponds to a mean of $0.46\times$ the length of the respective support vector (median $0.33\times$). Thus storing only the matches is an improvement over storing the complete matrix.

Following the definition of the TRECVID HLFE task, we use the mean average precision (MAP) of up to 2000 results (on shot granularity) per concept for evaluation. As a baseline, we also compare with the RBF kernel on individual key frames, with the RBF's γ parameter optimized by grid search. The results confirm those reported in the literature, i.e., all the kernels taking the sequence of feature vectors into account outperform the RBF kernel applied to individual feature vectors. The results for the different types of sequence kernels are quite similar. At least for the concepts of this data set, there is no significant difference in whether a kernel performs alignment of the sequences ((W)ASS, (A)LCSS) or not (TPM).

The models used for the sliding window-based variants of the kernels have been trained using the segment-based variants. As the ground truth on this data set is given per segment, we determine the value of the kernel function for one segment as the maximum of the values for the sliding window positions overlapping this segment:

$$\kappa((x_{\tau_1}, \ldots, x_{\tau_2}), U_k) = \arg\max_i \kappa((x_i, \ldots, x_{i+|U_{max}|}), U_k), \quad i = \tau_1 - |U_{max}| + 1, \ldots, \tau_2 - |U_{max}|. \tag{3}$$

On a live video stream, one would expect that the sliding window variants perform at least as good as the segment-based ones, as they will detect concepts at positions for which a segment-based approach might not find a sufficiently good alignment. However, as this data set is not a live stream, but edited video, another effect will also occur. At the beginning of a segment, samples from previous segments are included, which can cause false positives in some cases.

Figure 1 compares the results of segment-based and sliding window variants. The results show that the sliding window variants perform slightly better than the segment-based ones, although the difference is not significant. For some concepts, the sliding window variant clearly outperforms the segment-based one, this is especially the case for the ALCSS kernel, which is more successful in finding an optimal alignment in the sliding window case. Overall, the sliding window variants of TPM and ALCSS perform slightly better than the segment-based versions, and slightly worse for ASS (both in terms of mean and median). As ASS considers all subsequences, using information from the previous shot will lead to a higher number of false positives. Thus we also evaluate ASS in the case where the stored data is discarded at shot boundaries, and we see that this eliminates the performance decrease of ASS at the cost of a slightly higher latency at the beginning of each shot.

We have measured the improvement in runtime performance when using the sliding window versions of the kernels (using one core of an Intel Core 2 Duo 3GHz processor). The question whether we achieve real time performance depends on the number of classifiers that need to be applied (i.e., the number of concepts to be detected) and the complexity of each of these models. For all 20 concepts in the test set, prediction takes between $2.2 - 2.7\times$ real time for the segment-based versions of the different kernels and between $1.5 - 1.9\times$ real time for the sliding window based versions. The mean prediction time factor for the 20 models is between $0.11 - 0.13\times$ real time in the segment-based case, and around $0.07\times$ real time in the sliding window case. For the

Kernel	Information needed	Memory	Required processing	Runtime ($\delta = 1$)
TPM	matches for $\lvert U_{max}\rvert$ samples	$O(cn), 0 \leq c \ll n$	match new x_i, rebuild pyramid	$O(nT_{\kappa_f}) + O(cn)$
ASS	matches for $\lvert U_{max}\rvert$ samples	$O(cn), 0 \leq c \ll n$	match new x_i, sum sequences	$O(nT_{\kappa_f}) + O(n^2)$
WASS	matches (and κ_f) for $\lvert U_{max}\rvert$ samples	$O(cn), 0 \leq c \ll n$	match new x_i, sum sequences, calculate weights	$O(nT_{\kappa_f}) + O(n^2)$
LCSS	matches (and κ_f) for $\lvert U_{max}\rvert$ samples	$O(cn), 0 \leq c \ll n$	match new x_i, sum sequences, backtrack longest subsequence	$O(nT_{\kappa_f}) + O(n^2) + O(n)$
ALCSS	matches (and κ_f) for $\lvert U_{max}\rvert$ samples	$O(cn), 0 \leq c \ll n$	match new x_i, sum sequences, backtrack all subsequences	$O(nT_{\kappa_f}) + O(n^2) + O(2n^2)$

Table 1: Information to be stored for efficient evaluation of the kernel in the next sliding window, and resulting memory and runtime complexity. All values are given per support vector.

most complex model (i.e., the one with the largest number of support vectors) it takes $0.55 - 0.65\times$ real time in the segment-based case and around $0.46\times$ real time in the sliding window case. This means that with the current implementation real time prediction on a single core is possible for 14 concepts (assuming models of average complexity).

6. CONCLUSION

Our results confirm those of other authors, i.e., that sequence-based kernels outperform approaches processing feature vectors of the samples of a segment individually. It is thus important to enable the application of these kernels also in the online detection case.

We proposed sliding window variants of three approaches for sequence-based kernels (temporal pyramid matching, all subsequences, longest common subsequence) and show how they can be efficiently applied to online concept detection. The proposed kernels allow plugging in an arbitrary kernel for matching feature vectors of individual samples of the input sequences, and can thus be applied to sequences of any type of feature.

We have evaluated the proposed kernels on the TRECVID 2007 data set and the results show that the sliding window based versions perform as well or better than the corresponding segment-based variants. Even if the performance is only slightly improved, the proposed kernels have two important benefits: First, they can be applied in online cases, where no input segmentation is present and detection results need to be reported with low latency. Second, the proposed kernels can be evaluated more than 30% faster in the prediction step, allowing to run more than a dozen concept detectors in real time on a single core.

Acknowledgments

The author would like to thank Christian Schober, Georg Thallinger and Werner Haas for their support. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. FP7-248138, "FascinatE – Format-Agnostic SCript-based INterAcTive Experience" (`http://www.fascinate-project.eu/`).

7. REFERENCES

[1] S. Ayache and G. Quénot. TRECVID 2007: Collaborative annotation using active learning. In *Proc. TRECVID Workshop*, 2007.

[2] W. Bailer. A feature sequence kernel for video concept classification. In *Proceedings of 17th Multimedia Modeling Conference*, Taipei, TW, Jan. 2011.

[3] L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra. Video event classification using string kernels. *Multimedia Tools Appl.*, 48(1):69–87, 2010.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] J. Choi, W. J. Jeon, and S.-C. Lee. Spatio-temporal pyramid matching for sports videos. In *Proc. 1st ACM MIR*, 2008.

[6] D. Djordjevic and E. Izquierdo. Relevance feedback for image retrieval in structured multi-feature spaces. In *Proc. 2nd international conference on Mobile multimedia communications*, 2006.

[7] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. IEEE CVPR*, 2005.

[8] K. Grauman and T. Darrell. Approximate correspondences in high dimensions. In *NIPS*, 2006.

[9] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[10] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, M. Wang, and H.-J. Zhang. Correlative multilabel video annotation with temporal kernels. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(1):1–27, 2008.

[11] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, 2004.

[12] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In *Proc. 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, 2006.

[13] P. Wilkins et al. K-Space at TRECVid 2007. In *Proceedings of the TRECVid Workshop*, 2007.

[14] G. Wu, Y. Wu, L. Jiao, Y.-F. Wang, and E. Y. Chang. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In *Proc. ACM Multimedia*, 2003.

[15] D. Xu and S.-F. Chang. Visual event recognition in news video using kernel methods with multi-level temporal alignment. In *IEEE CVPR*, 2007.

[16] M.-C. Yeh and K.-T. Cheng. A string matching approach for visual retrieval and classification. In *Proc. 1st ACM international conference on Multimedia information retrieval*, pages 52–58, 2008.

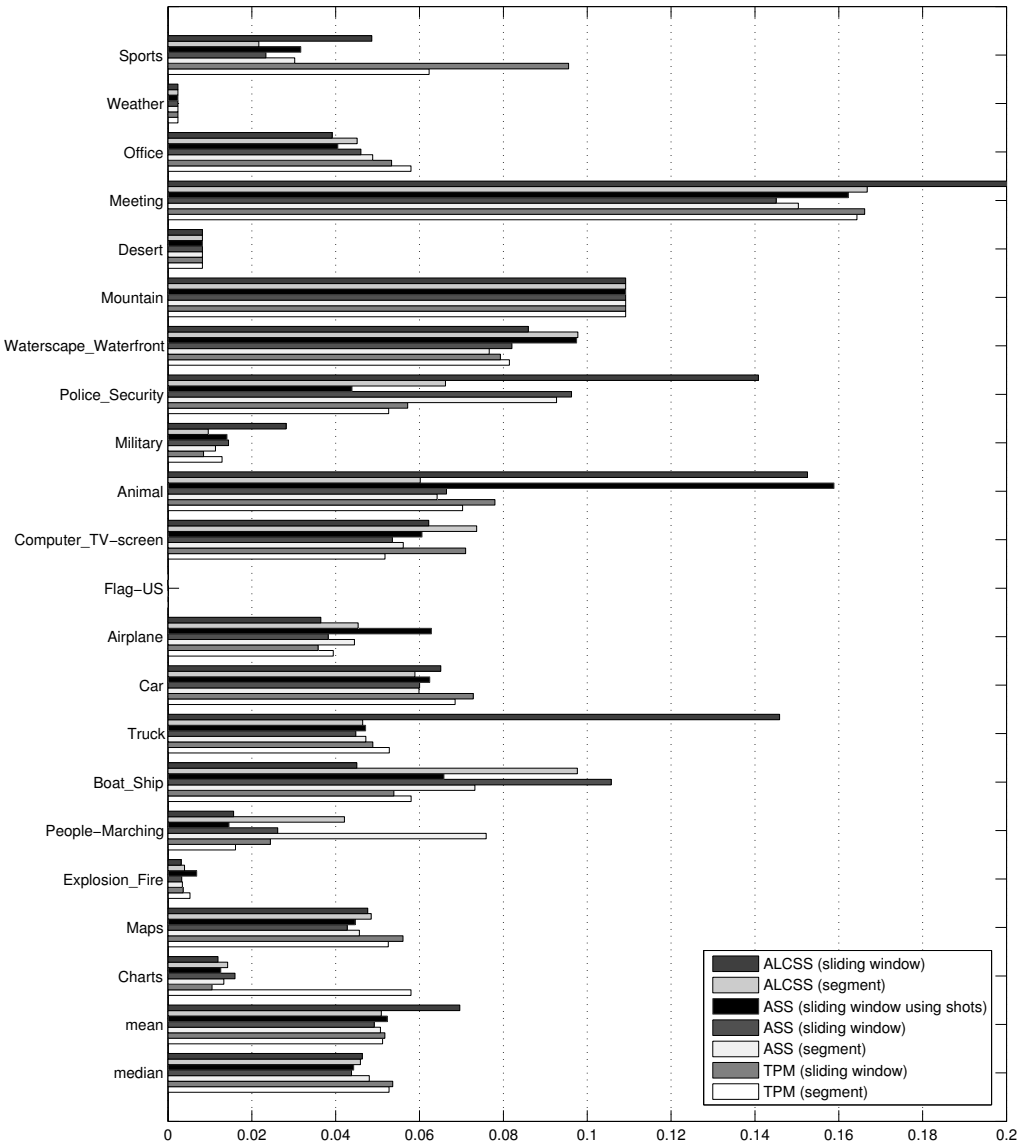

Figure 1: MAP of segment-based vs. sliding window variants of three kernels.