Discriminative Word-Spotting Using Ordered Spectro-Temporal Patch Features

Tony Ezzat, Tomaso Poggio

Center for Biological and Computational Learning,
McGovern Institute for Brain Research
Massachusetts Institute of Technology, Cambridge, MA
tonebone@mit.edu, tp@ai.mit.edu

Abstract

We present a novel architecture for word-spotting which is trained from a small number of examples to classify an utterance as containing a target keyword or not. The word-spotting architecture relies on a novel feature set consisting of a set of *ordered spectro-temporal patches* which are extracted from the exemplar mel-spectra of target keywords. A local pooling operation across frequency and time is introduced which endows the extracted patch features with the flexibility to match novel unseen keywords. Finally, we describe how to train a support vector machine classifier to separate between keyword and non-keyword patch feature responses. We present preliminary results indicating that our word-spotting architecture achieves a detection rate of 70-95% with false positive rates of about 0.25-2 false positives per minute.

Index Terms: word-spotting, spectro-temporal patches, support vector machines

1. Introduction

In this work, we present a novel architecture for performing "word-spotting". Given an utterance U, the goal of word-spotting is to detect whether a certain target word w is present in the utterance or not. Applications for this task are numerous, ranging from audio information retrieval to speech surveillance to general speech recognition itself.

Our system requires only a few (about 20) positive exemplars of the target word, and about 5 minutes of speech in which the target word is not present. From this data our system trains a discriminative support vector machine classifier [9], which is able to classify whether a window W of sound contains the target word or not. By sliding this window W across the utterance and classifying each portion, the entire utterance can be classified as containing the target word or not.

2. Motivation

Our motivation in this work is to explore one of the central issues at the heart of auditory processing, namely how to build a model of a keyword that is simultaneously *selective* and *invariant*:

On the one hand, a word-spotting system requires a set of auditory *features* that are *selective* for the target word. These features allow the system to respond selectively if the target word occurs, and not otherwise.

On the other hand, the system must also be *invariant* to the types of variations which typically occur to the target word, such as variations in duration, pitch, speaker, noise, and channel.

Most modern word-spotting algorithms [1, 2, 3, 4] address this trade-off through the currently dominant "HMM-MFCC" paradigm: frame-based Mel-frequency cepstral coefficients are extracted as speech features, which are then used to train hidden markov models for the keywords of interest.

In this paradigm, MFCCs provide a mechanism for achieving *spectral selectivity*: each spectral frame is projected onto a low-order DCT basis representing different types of spectral modulations. The incorporation ofΔ and $\Delta\Delta$ cepstral coefficients also adds temporal sensitivity to the MFCC feature set.

The HMM component, on the other hand, serves as a mechanism for achieving invariance: specifically, the use of several Gaussian mixture components per state models the spectral variations due to coarticulation, pitch changes, or formant changes which routinely occur during different instantiations of the target word. In addition, the forward-backward dynamic-time-warping (DTW) component of the HMM is a mechanism for modelling the temporal duration changes of the keyword.

Unfortunately, there are many problems associated with the HMM-MFCC framework: firstly, as we will argue in the next section, while MFCC features are selective for spectral features, they are not explicitly selective for a host of other important temporal and spectro-temporal phenomena in speech which occur over longer time-scales; secondly, HMMs usually need a large number of positive exemplars for training, requiring on average upwards of 100 positive exemplars of a keyword in order to achieve reasonable detection rates; finally, most word-spotting implementations using HMMs require embedding the trained keyword HMMs in a graph in order to perform Viterbi decoding. The graph must include other "filler" models consisting of monophone or triphone HMMs. Training these filler models seems like an overly tedious thing to do when our main concern is to mainly detect a keyword.

In this work, we present a novel architecture for word-spotting that aims to avoid some of the afore-mentioned shortcomings. We motivate our novel architecture in the following section.

3. Spectro-Temporal Phenomena

A typical mel-spectrogram of speech, such as the one shown in Figure 1, displays several important and well-known phenomena: harmonicity, which is exemplified by the presence of

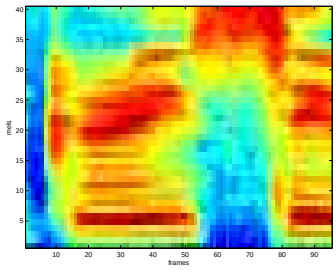

Figure 1: The mel-spectrum of an example of the word ''Greasy''

thin horizontal lines related to the pitch of the speaker; formant-related amplitude modulations, exemplified by broader horizontal lines; vertical onset/offset edges in time, related to plosive sounds in speech; and noise, related to fricatives, aspirants, and other phonemes which generate noise.

All of these speech-related phenomena may be viewed as different types of spectro-temporal modulations, or "edges", that come at various orientations, durations, and scales. Clearly, one of the challenges of designing any feature set is that it must be equally selective for all of these various features.

Frame-based MFCC features may be viewed as a feature that spans a vertical line on a spectrogram. As a result, MFCC features are predominantly sensitive to *horizonal* spectral modulations such as harmonic lines and formants. On the other hand, they are not explicitly senstive for temporal modulations such as offsets/onsets found in plosives, or more complex spectro-temporal patterns such as those found in fricatives (speckle noise patterns) or diphthongs (rising/falling formants). While it is true that the addition of Δ and $\Delta\Delta$ parameters to MFCCs adds temporal selectivity, this selectivity extends only for a short time scale of about 30-50 msec, and cannot capture longer temporal modulations lasting 100-200msec.

Additionally, MFCC features may be viewed as a *global* feature: the spectral frame from which the MFCC parameters are computed spans the entire frequency range. On the other hand, many phenomena which occur in a spectrogram are *local*: one can see for example that harmonic, formant, or noise patterns do not affect the entire spectral frame but only local spectro-temporal portions.

Recent neurophysiological evidence from a number of animals such as ferrets [5] and birds [6] indicates that cells in the auditory cortex are, in fact, tuned to localized spectro-temporal modulations. The spectro-temporal receptive fields (STRFs) of these cortical cells are selective for complex but localized spectro-temporal patterns. Overall, the general picture painted by these neurophysiological findings is that the features that auditory cortical cells are selective for are more *patch-like* than *frame-like*.

4. Ordered Bags of Spectro-Temporal Patch Features

Motivated by these observations, we propose a new speech feature set in this work which consists simply of *2D spectro-temporal patches*. These patches are extracted from the exemplars of the target word at *random locations in frequency and time*. The height and width of each extracted patch is drawn uniformly from a spectral range F_{range} and temporal range T_{range} (defined in the Section 10 section).

Each extracted patch feature $P_k(f,t)$ may be viewed as a matched filter that is selective for other similar patterns. Since the patches are extracted non-parametrically from the target word exemplars, they can in principle capture any type of spectro-temporal phenomenon occurring in the target word, such as harmonic lines, formant ridges/sweeps, vertical onset/offset plosives, or noise patterns.

Since speech is a temporal phenomenon, it is necessary to keep track of the *order* in which certain patches are activated for a target word. This is achieved in our case by recording the location in frequency f_k and *relative time* rt_k at which patch P_k occurred in the target word. Relative-time is measured with respect to the overall duration of the target exemplar, so a patch extracted at time $t = 400msec$ from a target word of duration 800msec has the same relative-time index of $rt = 0.5$ as a patch extracted at $t = 200msec$ from a word of duration 400msec.

Our extracted patch dictionary thus consists of a set of K extracted patches $\{P\}_{k=1}^{K}$, their center locations in frequency $\{f\}_{k=1}^{K}$, and their center locations in relative time $\{rt\}_{k=1}^{K}$. We call this dictionary an *ordered* bag of spectro-temporal patches to emphasize that a rough measure temporal order is preserved. Shown in the left part of Figure 2 are some example patches extracted from a spectrogram of the word greasy.

5. Spectro-Temporal Patch Response

Given a novel spectrogram $S(f,t)$ of duration T, the patch dictionary may be applied to that spectrogram to compute the *patch dictionary response* $\{R_k\}_{k=1}^{K}$ as follows: each patch P_k in the dictionary is placed at location $(f_k, rt_k * T)$, and the L_2 norm is computed between the patch and underlying portion of the spectrogram:

$$R_k = \sum_f \sum_t \|P_k(f - f_k, t - rt_k * T) - S(f,t)\|^2 \quad (1)$$

where $\|f - f_k\| \leq \frac{H_k}{2}$ and $\|t - rt_k * T\| \leq \frac{W_k}{2}$. If the patch is placed at a portion near the borders of the spectrogram, then the intervals in the integral of Equation 1 are truncated to their valid limits.

If there are K patches in our dictionary, then the spectro-temporal response $R(k)$ is of dimension K, independent of the length T of the spectrogram $S(f,t)$. In effect, computing the patch dictionary response produces a fixed-length feature vector *independent of the duration of the input*. The computation of the patch response is shown in middle part of Figure 2.

6. Invariance Through Pooling

Computation of our patch dictionary response $R(k)$ as defined in Equation 1 amounts to an operation that is not much different from convolving the entire spectra of the exemplar target keywords with novel input spectra. As such, our approach, while being selective, would not be invariant to the typical variations which occur to patches in the target keywords.

An important class of variations considered in this work consists of patch frequency shifts and patch temporal shifts. A certain patch which occurs at frequency f and relative time rt for a certain target word exemplar, may occur at $f + \Delta f$ and relative time $rt + \Delta rt$ for another target word exemplar.

It is in general not clear how to use an HMM to handle this important class of variations. While the DTW step in an HMM is capable of handling 1-D time shifts (or 1-D frequency shifts), an HMM is not capable for handling joint 2D spectro-temporal shifts of the underlying patch features.

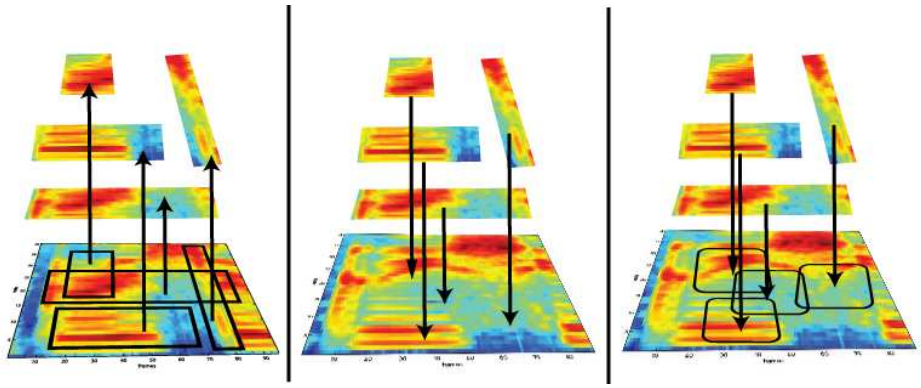

Figure 2: An overview of our word-spotting architecture. Left: Spectro-temporal patches with random positions and sizes are extracted from an exemplar keyword. Middle: The extracted patches are matched with a novel incoming sound at their exact extraction locations in frequency and relative time. Right: The best match to the patches within a local pooling range is computed.

This type of invariance is addressed by drawing upon recent progress in biologically-inspired computer vision models of object recognition [7, 8]. These approaches use a bag-of-features approach in which small (possibly filtered) image patches are extracted from a training set and then subsequently convolved with an incoming novel image at different positions and scales to compute the overall patch response (much as we do in Equation 1). For each dictionary patch, its *best response* across a range of positions and scales is retained. In effect, a pooling operation (via a MAX or MIN operator) is performed to account for the variations of the patch feature across typical variations seen in images.

The local pooling approach is adopted in this work. Specifically, we define a local frequency shift range ΔF_{range} and a local temporal shift range ΔT_{range}. Our *pooled patch response* $\hat{R}_k$ is defined as

$$\hat{R}_k = \min_{\substack{\|f_c - f_k\| \leq \Delta F_{range} \\ \|t_c - rt_k * T\| \leq \Delta T_{range}}} R_k(f_c, t_c) \qquad (2)$$

. The pooled response for a certain dictionary patch is the best match of that patch over a certain frequency shift range and temporal shift range, centered around that patch's center frequency f_k and relative-time location rt_k.

7. Discriminative Classification Via SVMs

Since the pooled patch dictionary responses $\hat{R}(k)$ are of dimension K and independent of time, it is possible to apply standard discriminative learning techniques to learn a decision function that separates positive keyword patch responses from negative non-keyword patch responses.

In this work, support vector machine classifiers [9] are used to learn such a disciminating hyperplane. Given a set of N^{pos} exemplar spectrograms $\{S^{pos}(f,t)\}$ of the target word, the corresponding pooled spectro-temporal response vectors $\{\hat{R}^{pos}\}$ are first computed. These positive patch responses are associated with a positive class label $y^{pos} = 1$.

From the negative training utterances, a randomly selected set of $N^{neg} = N^{pos}$ segments $\{S^{neg}(f,t)\}$ are extracted. The durations of those segments are drawn from a normal distribution with mean μ^{pos} equal to the mean of the target keyword durations, and standard deviation σ^{pos} equal to the standard deviation of the target keyword durations. From these random negative spectrogram segments, the corresponding negative pooled responses $\{\hat{R}^{neg}\}$ are computed. These negative responses are associated with a negative class label $y^{neg} = -1$.

An initial support vector machine is trained to produce a hyperplane that discriminates between the positive feature responses $\{\hat{R}^{pos}\}$ and the randomly chosen negative feature responses $\{\hat{R}^{neg}\}$. The trained support vector machine produces a decision function $y^{test} = f(\hat{R}^{test})$ that takes the form

$$y^{test} = sgn(\underbrace{\sum_{i=1}^{N^{pos}+N^{neg}} y_i \alpha_i K(\hat{R}_i, \hat{R}^{test}) + b}_{rw}) \qquad (3)$$

Typically the α_i are zero except for a few positive and negative "support-vector" exemplars that lie at the boundary between the two classes. In all our experiments, a Gaussian SVM kernel $K(x,x)$ is used. The value rw denotes the distance of the test patch response $\hat{R}^{test}$ from the separating hyperplane: the more positive this value, the more confident that the SVM thinks the test exemplar is a target keyword; the more negative, the more confident that the SVM thinks this test exemplar is not a keyword.

In general, the initial separating SVM hyperplane of Equation 3 estimated using random negative examples may generate too many false positives at test time. To reduce false positives, one round of *bootstrapping* is performed: Using the initial SVM, 1 minute of negative training data is scanned and classified. The false positives from this bootstrapping run are extracted and lumped together along with the initial randomly chosen negative examples, and the SVM is re-trained. This produces a final SVM classifier for each target keyword.

8. "Sliding Window" Classification

In this work, a *sliding-window* approach is adopted for testing, in which a window of width W in frames is slid across the novel test utterance. For each keyword experiment, the width W is set to the average duration in frames of that target keyword (obtained from the target exemplars). Similarly, the window stepsize in frames is set to $0.2 * W$.

For each extracted window, a pooled patch response R^{test} is computed as in Equation 2, and the trained bootstrapped key-

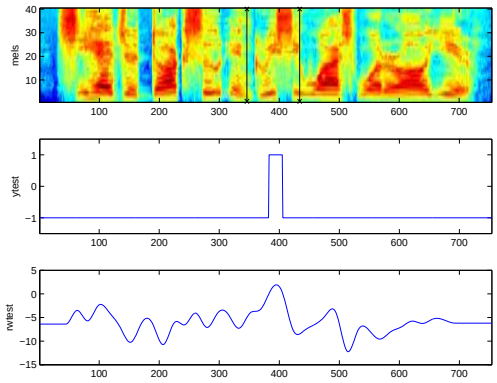

Figure 3: Example output of our system. Top: the mel-spectrum for a novel test utterance, with the word `Greasy` highlighted within black columns. Middle: the $ytest$ output of our SVM classifier. Bottom: the rw distance value of our classifer.

word SVM is applied to that response to produce the class label y^{test} for that window (1 if the target is present, and -1 if not).

It should be noted that the sliding-window approach adopted in this work is fundamentally different from Viterbi-style decoding schemes adopted by most word-spotting systems [1, 2, 3, 4]. The sliding-window approach is *local*, classifying each local portion independent of its context. Viterbi-style decoding schemes, on the other hand, are *global*, finding optimal paths thought a graph network with optimal treatment of context. In principle, however, our approach can be embedded in a Viterbi-style decoding scheme, in which several SVM classifiers for different words are run in parallel. Due to space limitations, we choose to focus on local sliding-window approaches in this work.

9. ROC curves

Since the word spotting task is a *detection* task, receiver operator characteristics, or ROC curves, are employed to summarize our detection experiments.

The vertical dimension in our ROC curves measures the *detection rate*, which is the fraction of of target words present in the test data that were correctly detected. A detection rate of 0.9% thus means that 90 out of 100 target keywords were detected correctly, and 10 were missed.

The horizontal dimension of the ROC curve measures the *false positive rate*, which is the number of false positives per minute of test data. This value is computed by dividing the total number of false positives which occurred in the test data by the total time of the test data.

The rw value from the SVM in Equation 3, which is the distance to the hyperplane extracted for each window classification, constitutes the parameter that is swept over to produce the ROC curve in our SVM experiments.

10. Experimental Results

In the following, we present initial results on word-spotting experiments for four TIMIT keywords: `greasy`, `dark`, `wash`, and `oily`.

10.1. Training and Testing Data

Our dataset is the TIMIT corpus, which was divided into standard train and test sets following the convention in [10]. Additionally, the train and test sets were further split into positive and negative sets containing (and not containing) the exemplar keywords.

The positive training utterances for each keyword consisted only of the keyword, while the negative training sets were longer utterances in which the keyword was not present. The number N^{pos} of positive training exemplars for each keyword was varied across experiments, as describes in the next section, from $N^{pos} = \{1, 5, 10, 20, 200\}$. The number of negative training utterances was always fixed at 100 utterances, or about 5 minutes of negative speech.

The positive and negative test sets consisted of equally long utterances containing (and not containing) the keyword. Across each keyword experiment, the test set consisted of 100 positive utterances containing the keyword, and 100 utterances not containing the keyword, or about 10 minutes of test speech.

10.2. Mel-Spectrogram Analysis

All of the 16KHz TIMIT training and test utterances were converted to mel-spectrograms: First the utterances are pre-emphasized, followed by STFT analysis using a 25msec Hamming window with 4msec hops. An 800-point FFT is applied, followed by truncation to 400 bins due to the symmetry of the Fourier transform. The resulting power spectrum is subsequently passed through a Mel-filterbank consisting of 40 triangular filters equally spaced on the Mel scale ranging from 0 to 8KHz. The mel-filterbank output is passed though a log function to produce the final log-mel-spectra used in all our experiments.

10.3. System Parameters

Patch dictionaries were constructed for each keyword from the positive exemplar melspectra. The parameters F_{range} and T_{range} which determine the extracted patch height and width ranges respectively were set to $1 - 20 mels$ and $1 - 20 frames$ across all keywords. The parameter ΔF_{range} which determines the pooling range in frequency is set to 5 mels for all experiments. The parameter ΔT_{range} which deterimines the temporal pooling range is set to the standard deviation σ^{pos} of the durations of the positive exemplars for that keyword.

10.4. HMM-MFCC Baseline

As a baseline comparison to our approach, we train an HMM for each target word, using 5 states, 3 Gaussian mixtures per state, and diagonal covariances. The target word HMMs are trained using MFCC features, with Δs and $\Delta\Delta$s appended to produce a 39 dimensional feature vector. All comparisons with our approach are made with the HMMs trained on the same number of positive target exemplars.

The target keyword HMMs are incorporated into the same sliding-window framework as that of the keyword SVMs. For each window W, the trained keyword HMM assigns a log-likelihood using the standard forward-backward recursions. This log-likelihood is subsequently thresholded to produce ROC curves for the HMM baseline experiments.

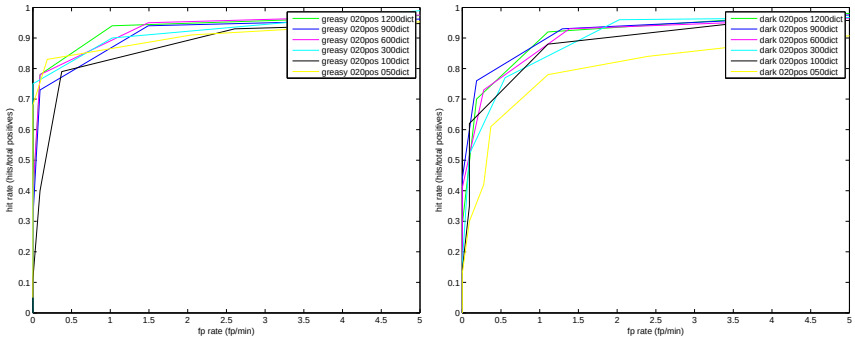

Figure 4: Effect of the patch dictionary size K on detection performance for the keyword ‘‘greasy’’ (left) and ‘‘dark’’ (right)

10.5. Effect of the Dictionary Size

An initial experiment was performed to determine the effect of the patch dictionary size on detection performance. The number of positive exemplars were kept fixed at 20, while the number of patches extracted was varied from $K = \{50, 100, 300, 600, 900, 1200\}$. Shown in Figure 4 are such experiments involving the keywords `greasy` and `dark`.

While increasing the patch dictionary size clearly improves the area under the ROC curve, the increase in performance is small. In fact, a patch dictionary of size $K = 100$ does just about as well one with $K = 200$ or more patches. As a result, all of our subsequent set of experiments were performed with a fixed patch dictionary size of $K = 100$.

10.6. Effect of the number of Positive Examples

In Figure 8, we examine more carefully the difference in performance between our approach and that of the HMM baseline for the four keywords. Additionally, we examine the effect of different numbers of positive exemplars on performance.

Shown in the left column of Figure 8 are the four keyword ROC curves from our approach as we vary the number of positive training exemplars. Shown in the right column are the corresponding four keyword ROC curves for the HMM baseline as we vary the number of positive exemplars.

Firstly, we note that, *across all keywords and all numbers of positive exemplars, our approach clearly outperforms the HMM-MFCC baseline.*

Additionally, we also note that *our approach performs well with as few as 5 examples per keyword.* This is to be expected since support vector machines are known to work well even with very few examples. Only when our system is provided with 1 exemplar does it begin to break down. In contrast, the HMMs need at least 50 exemplars in order for them to start to achieve reasonable detection and false positive rates.

Specifically, for 20 positive exemplars, our approach achieves 95% detection at 0.25 false positives per minute for `greasy`, 90% detection at 0.25 false positives per minute for `wash`, 70% detection at 1 false positive per minute for `dark`, and 60% detection at 2 false positives per minute for `oily`. It is unclear at this time why performance for `oily` is so low for both our approach and that of the HMM-MFCC baseline.

11. Future Work

While this paper presented initial results, more experiments are needed across a larger number of target keywords to assess the approach's average performance. Additionally, more experiments are needed in which the parameters of the system are cross-validated to explore their effect on system performance.

Clearly, it would be interesting to see whether the proposed approach may be able to detect target words *in noise*, and further experiments involving the AURORA speech dataset [11] are planned.

Finally, we plan on exploring the capacity of this approach to detect sub-word units such as syllables, which would allow us to replace HMM acoustic models in a large-vocabulary speech recognition task.

12. References

[1] J.R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden markov modeling for speaker-independent word spotting," *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pp. 627–630 vol.1, 23-26 May 1989.

[2] R.C. Rose and D.B. Paul, "A hidden markov model based keyword recognition system," *ICASSP-90*, pp. 129–132 vol.1, 3-6 Apr 1990.

[3] M. Weintraub, "Keyword-spotting using sri's decipher large-vocabulary speech-recognition system," *ICASSP-93*, vol. 2, pp. 463–466 vol.2, 27-30 Apr 1993.

[4] D.A. James and S.J. Young, "A fast lattice-based approach to vocabulary independent wordspotting," *ICASSP-94*, vol. i, pp. I/377–I/380 vol.1, 19-22 Apr 1994.

[5] T. Chih, P. Ru, and S. Shamma, "Multiresolution spectrotemporal analysis of complex sounds," *Journal of the Acoustical Society of America*, vol. 118, pp. 887–906, 2005.

[6] F.E. Theunissen, K. Sen, and A. Doupe, "Spectral-temporal receptive fields of nonlinear auditory neurons obtained using natural sounds," *Journal of Neuroscience*, vol. 20, pp. 2315–2331, 2000.

[7] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Object recognition with cortex-like mechanisms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2006.

[8] M. Riesenhuber and T. Poggio, "Hierachical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, pp. 1019–1025, 1999.

[9] Nello Cristianini and Bernhard Schölkopf, "Support vector machines and kernel methods: the new generation of learning machines," *AI Mag.*, vol. 23, no. 3, pp. 31–41, 2002.

[10] A. Halberstadt and J. Glass, "Heterogeneous measurements and multiple classifiers for speech recognition," in *Proceedings of ICSLP*, 1998.

[11] H.G. Hirsch and D.Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ASR-2000*, pp. 181–188.

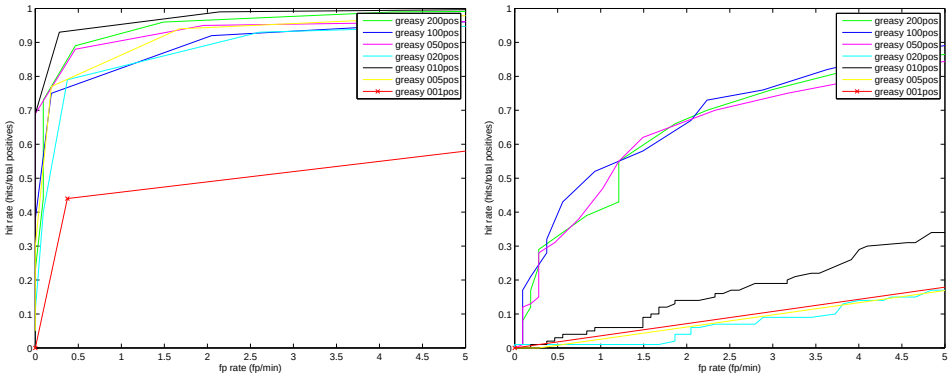

Figure 5: ''Greasy'' ROC curves from our approach (left) versus the HMM-MFCC approach (right), for various numbers of positive training exemplars.

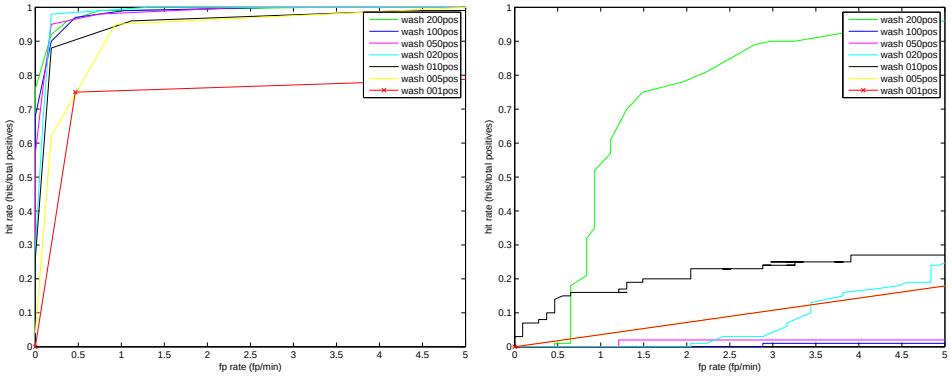

Figure 6: ''Wash'' ROC curves from our approach (left) versus the HMM-MFCC approach (right), for various numbers of positive training exemplars.

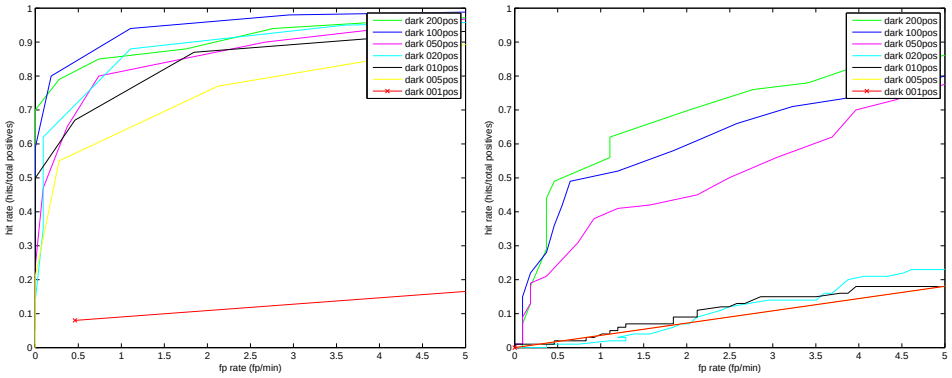

Figure 7: ''Dark'' ROC curves from our approach (left) versus the HMM-MFCC approach (right), for various numbers of positive training exemplars.

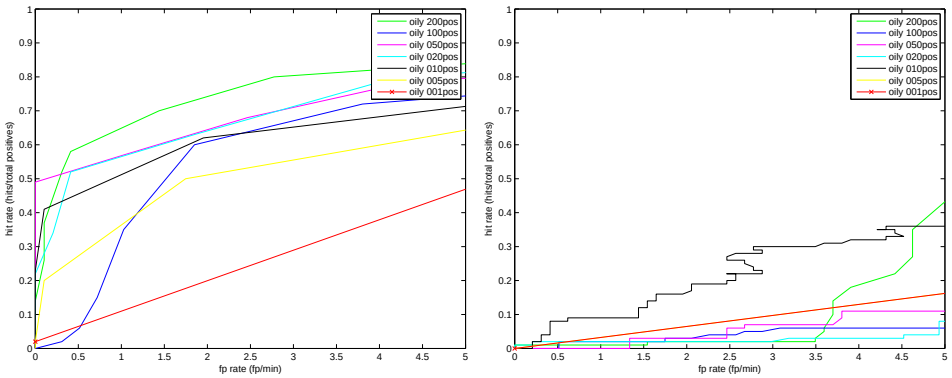

Figure 8: ''Oily'' ROC curves from our approach (left) versus the HMM-MFCC approach (right), for various numbers of positive training exemplars.