Frame-Level Speech/Music Discrimination using AdaBoost

Norman Casagrande
University of Montreal
Department of Computer Science
CP 6128, Succ. Centre-Ville
Montreal, Quebec H3C 3J7 Canada
casagran@iro.umontreal.ca

Douglas Eck
University of Montreal
Department of Computer Science
CP 6128, Succ. Centre-Ville
Montreal, Quebec H3C 3J7 Canada
eckdoug@iro.umontreal.ca

Balázs Kégl
University of Montreal
Department of Computer Science
CP 6128, Succ. Centre-Ville
Montreal, Quebec H3C 3J7 Canada
kegl@iro.umontreal.ca

ABSTRACT

In this paper we adapt an AdaBoost-based image processing algorithm to the task of predicting whether an audio signal contains speech or music. We derive a frame-level discriminator that is both fast and accurate. Using a simple FFT and no built-in prior knowledge of signal structure we obtain an accuracy of 88% on frames sampled at 20ms intervals. When we smooth the output of the classifier with the output of the previous 40 frames our forecast rate rises to 93% on the Scheirer-Slaney (Scheirer and Slaney, 1997) database. To demonstrate the efficiency and effectiveness of the model, we have implemented it as a graphical real-time plugin to the popular Winamp audio player.

1 Introduction

The ability to automatically discriminate speech from music in an audio signal is useful in domains where a particular type of information is of interest, such as in automatic audio news transcription of a radio broadcast, where non-speech would presumably be discarded. Previous models have employed a mixture of simple features that capture certain temporal and spectral features of the signal (Scheirer and Slaney, 1997; Saunders, 1996). including for example pitch, amplitude, zero crossing rate, cepstral values and line spectral frequencies (LSF). More recently, other approaches have used the posterior probability of a frame being in a particular phoneme class (Williams and Ellis, 1999), HMMs that integrate posterior probability features based on entropy and "dynamism" (Ajmera et al., 2002), and a mixture of Gaussians on small frames (Ezzaidi and Rouat, 2002).

We have adapted a successful and robust approach for object detection (Viola and Jones, 2001) to this task. Our model works by exploiting regular geometric patterns in speech and non-speech audio spectrograms. These regularities are detectable visually, as demonstrated by the ability of trained observers to identify speech structure (e.g. vowel formant structure, consonant onsets) and musical structure (e.g. note onsets and harmonic pitch structure) through visual inspection of a spectrogram. We demonstrate in this paper that by exploiting geometric regularities in a two-dimensional representation of sound, we are able to obtain good accuracy results (88%) for 20ms frame categorization with no built-in prior knowledge and at very low computational cost. When smoothing is employed over 40 previous frames (800ms), our accuracy rises to 93%. This compares favorably with other models on the same dataset.

Despite being motivated by work in vision, this model is well suited for audio signal processing. Though it treats individual 20ms slices of music as having fixed geometry, it places no limitations on the geometry of entire songs. For example, it places no constraints on song length nor does it require random access to the audio signal. In other words, this approach is causal and is able to process audio streams online and in real time.

2 The algorithm

In order to build a good binary discriminator, one must first find a set of salient *features* that separate the two classes with the largest margin possible. To detect objects in an image, Viola and Jones employed a set of simple *Haar-like* (first proposed by Papageorgiou et al. (1998)) rectangles depicted in Figure 1. These features compute and subtract the sum of pixels in the white area from the sum of pixels in the black area. The areas can have different shapes and sizes, and can be placed at different x and y coordinates of the image. A discriminator using a single of these features is called a *weak learner* because, used alone, it cannot achieve very good discrimination. However, when these features are combined in an additive model, the resulting classifier can perform very well. In their work on two-dimensional images, Viola and Jones showed that with enough features, it is possible to detect complex objects like faces.

2.1 AdaBoost

To additively combine the weak learners, we use the ADABOOST algorithm (Freund and Schapire, 1996),

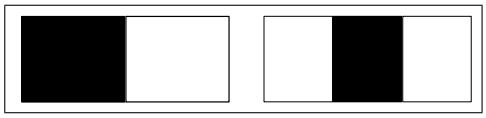

Figure 1: The two Haar-like features used in our additive model.

which is one of the best general purpose learning methods developed in the last decade. It has inspired several learning theoretical results and, due to its simplicity, flexibility, and excellent performance on real-world data, it has gained popularity among practitioners.

ADABOOST is an *ensemble* (or *meta-learning*) method that constructs a classifier in an iterative fashion. In each iteration, it calls a simple learning algorithm (the *weak learner*) that returns a classification. The final classification will be decided by a weighted "vote" of the weak classifiers, where each weight is proportional to the correctness of the corresponding weak classifier. This incremental process of combining weak classifiers weighed by their performance is called *boosting*. The weak classifiers need only be slightly better than a random guess, which lends great flexibility to the design of the weak classifier (or feature) set. If there is no particular a-priori knowledge available on the domain of the learning problem, small decision trees or, in the extreme case, *decision stumps* (decision trees with two leaves) are often used. A decision stump can be defined by three parameters, the index j of the attribute[1] that it cuts, the threshold θ of the cut, and the sign of the decision. Formally,

$$h_{j,\theta+}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^j \geq \theta, \\ -1 & \text{otherwise,} \end{cases} \quad (1)$$

and

$$h_{j,\theta-}(\mathbf{x}) = -h_{j,\theta+}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^j < \theta, \\ -1 & \text{otherwise.} \end{cases} \quad (2)$$

Although decision stumps may seem very simple, when boosted, they yield excellent classifiers in practice. Also, finding the best decision stump using exhaustive search can be done efficiently in $O(nd)$ time, where n is the number of training points, and d is the dimension of the input space (the number of Haar-like features in our case). Alternatively, by using a random sampling and a gradient following approach (as explained below), even the exhaustive search can be avoided.

For the formal description of ADABOOST, let the training set be $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is the observation vector, and y_i is its binary ($+1$ or -1, representing speech or music, respectively) label. The algorithm maintains a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points. The weights are initialized uniformly at the beginning, and are updated in each iteration. The weight distribution remains normalized in each iteration, that is, $\sum_{i=1}^n w_i^t = 1$ for all t. In general, the weight of a point will be proportional to how hard it is to correctly classify.

[1] In our case, the jth attribute is the output of the jth filter in the filter bank (see Section 2.2).

Given: $D_n = \{(x_1, y_1), ..., (x_n, y_n)\}$
where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize weights $w^1(i) = 1/n$.
For $t = 1, ..., T$:

- Find the feature h^t that minimizes the error ϵ^t.
- Compute confidence $\alpha^t = \frac{1}{2} \ln \frac{1-\epsilon^t}{\epsilon^t}$.
- Update the weight vector $\mathbf{w}$:

$$w^{t+1}(i) \quad = \quad w^t(i) \times \begin{cases} \frac{1}{2(1-\epsilon^t)} & \text{if } h^t(x_i) = y_i \\ \frac{1}{2\epsilon^t} & \text{if } h^t(x_i) \neq y_i \end{cases}$$

Output final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha^t h^t(x)\right).$$

Figure 2: Pseudocode of the ADABOOST algorithm.

We suppose that we are given a set $\mathcal{H}$ of weak classifiers and a weak learner algorithm that, in each iteration t, returns the weak classifier $h^t \in \mathcal{H}$ that minimizes the weighted error

$$\epsilon^t = \sum_{i=1}^{n} I_{\{h^t(\mathbf{x}_i) \neq y_i\}} w_i^t, \quad (3)$$

where the indicator function $I_{\{A\}}$ is 1 if its argument A is true and 0 otherwise. The coefficient α^t of h^t is the *confidence* we have in our weak learner. It is set to $\alpha^t = \frac{1}{2} \ln \frac{1-\epsilon^t}{\epsilon^t}$ in each iteration. Since $\epsilon^t < 1/2$ (otherwise we would flip the labels and return $-h^t$), the weight update formulas (see in the pseudocode in Figure 2), indicate that we increase the weights of misclassified points and decrease the weights of correctly classified points. As the algorithm progresses, the weights of frequently misclassified points will increase, so weak classifiers will concentrate more and more on these "hard" data points. After T iterations[2], the algorithm returns the weighted average $f^T(\cdot) = \sum_{t=1}^{T} \alpha^t h^t(\cdot)$ of the weak classifiers. The sign of $f^T(\mathbf{x})$ is then used as the final classification of $\mathbf{x}$.

2.2 The features

Our goal is to classify 20ms frames of audio as being either speech or music. We represent each training sample by its spectrogram $S_i = \{S(t, \phi)\}_i$, where $S(t, \phi)$ is the signal intensity at time t and frequency ϕ. We then convolve the image of the spectrogram with Haar-like filters (depicted in Figure1), find the best filter that discriminates the training data, and compute a stump (1-2) over the output of the filter.

Each filter contains two or three rectangular black or white blocks with different sizes and locations. For a black block with its upper left corner placed at (t, ϕ), and

[2] T is an appropriately chosen constant that can be set by, for example, cross-validation.

with size $w_t \times w_\phi$, we compute the convolution

$$B_{t,\phi,w_t,w_\phi}(S) = \sum_{i=t}^{t+w_t} \sum_{j=\phi}^{\phi+w_\phi} S(i,j).$$

For a white block, we compute the negative convolution

$$W_{t,\phi,w_t,w_\phi}(S) = -B_{t,\phi,w_t,w_\phi}(S) = -\sum_{i=t}^{t+w_t} \sum_{j=\phi}^{\phi+w_\phi} S(i,j).$$

So, for example, a three block white-black-white feature placed at (t,ϕ), and with block size $w_t \times w_\phi$ would output the value

$$W_{t,\phi,w_t,w_\phi}(S)+B_{t,\phi+w_\phi,w_t,w_\phi}(S)+W_{t,\phi+2w_\phi,w_t,w_\phi}(S).$$

The major advantage of these features over more complicated filters usually used in sound-processing that they can be computed at an extremely low cost. The main trick is to pre-compute the so called *integral image*

$$\Sigma(t,\phi) = \sum_{i=1}^{t} \sum_{j=1}^{\phi} S(i,j) \tag{4}$$

for each spectrogram in the training set. Then any convolution B or W can be computed in constant time by using the equation

$$\begin{aligned} B_{t,\phi,w_t,w_\phi}(S) \quad = \quad & \Sigma(t,\phi) + \Sigma(t+w_t,\phi+w_\phi) \\ & -\Sigma(t,\phi+w_\phi) - \Sigma(t+w_t,\phi). \end{aligned}$$

This allows us to evaluate a very large number of candidate features in every boosting iteration. Formally, each Haar-like filter g_j returns a real number $g_j(S_i)$ for each spectrogram S_i, which is the jth attribute x_i^j in the observation vector $\mathbf{x}_i$. Then for each filter, the best decision stump is found. Finally, we select the weak learner h^t which minimizes the weighted training error (3) among all the candidates.

Despite the simplicity of the filters, they can discriminate between speech and music by capturing local dependencies in the spectrogram. For example, the three-block feature depicted in Figure 3 is well-correlated with the speech signal and quasi-independent of the music signal. Figure 4 displays the real-valued output of the filter for all training points, and the threshold of the optimal decision stump. This feature, selected in the first iteration of AdaBoost, has a 30% error rate on the test set.

2.3 Reducing the training time

One of the problems with the Haar-like filters approach is that at each iteration only one optimal feature is selected, among hundreds of thousands of possible candidates. This results in a computational cost, for the training process, in the order of days on a single Pentium 4. To reduce the computational demand, we have implemented a solution that comes from the observation that small changes in the feature's property do not change the overall error significantly. Figure 5 shows the error map of the horizontal feature at the first iteration. Notice that the two-dimensional

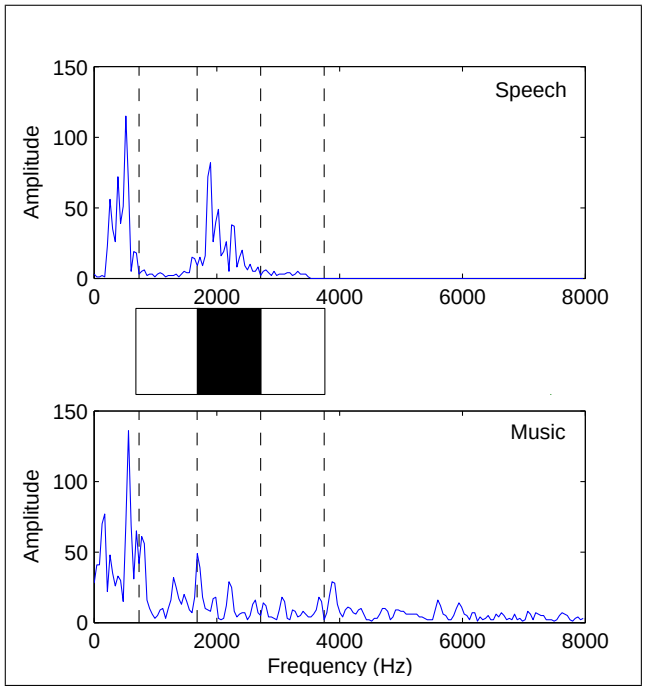

Figure 3: This three-block feature can distinguish speech from music. It is well correlated with the speech signal (its output is 347), and independent of the music signal (its output is -577).

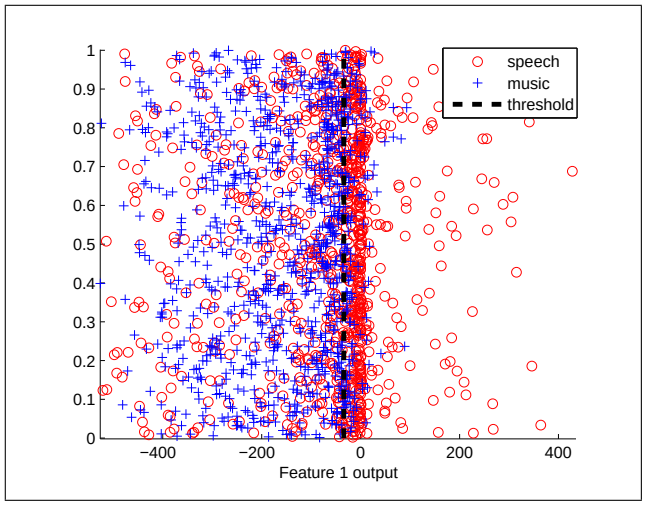

Figure 4: The output of the feature showed in Figure 3 for all training points, and its decision stump's threshold. The data has been randomly distributed on the vertical axis for clarity.

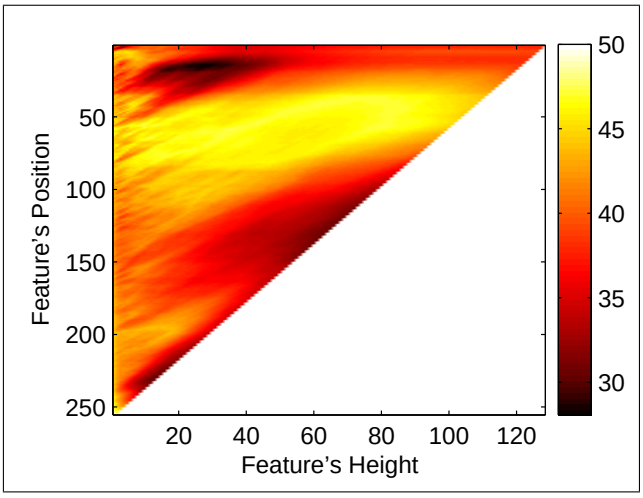

Figure 5: The error as a function of the two feature's parameters. Dark areas represents local minima that the algorithm is looking for. The height of the feature decreases linearly with the position, therefore the map is triangular. Observe that the resulting two-dimensional terrain is relatively smooth, suggesting that exhaustive sampling of these parameters may be unnecessary.

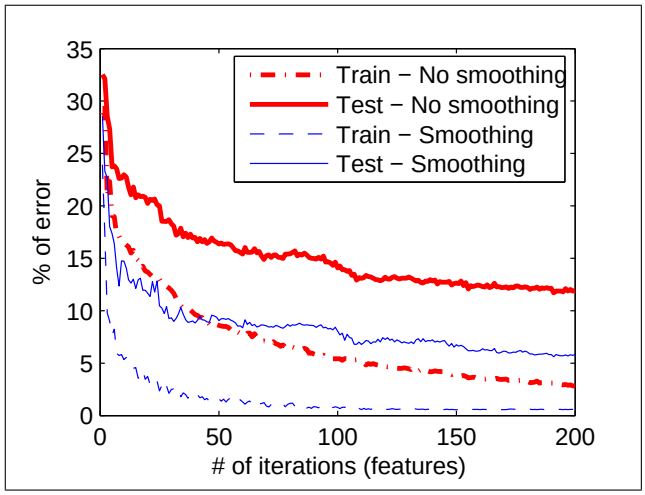

Figure 6: The results on a 20ms FFT frame, without and with smoothing. The benefits of smoothing are clearly seen both in test error and train error.

space is smoothly changing. This suggests that it is unnecessary to exhaustively sample all parameter combinations. After first selecting randomly a set of initial sampling parameters, we iteratively adjust them until we arrive at a local minimum in a *gradient following* fashion.

The feature thus discovered is suboptimal with respect to the exhaustive search. However, ADABOOST requires only that features support discrimination better than chance. For instance, Escudero (2000) showed with his *LazyBoosting* that even with randomly (homogeneous) features it was possible to achieve results comparable to those obtained from the optimal features. To compensate for the poorer features, a larger number was required.

2.4 Experimental results

In the experiments, we used 240 digital audio files of 15 second radio extracts published by Scheirer and Slaney (1997)[3]. We extracted 11200 20ms frames from this data, normalized them, and then processed them with FFT, RASTA (Hermansky et al., 1992), and log-scale FFT. We chose the first because it is the simplest representation of the frequency spectrum, and the other two because of their popularity in speech processing. The FFT represents the biggest frequency space with 256 points, followed by the other two (respectively 26 and 86). The size of this space has an important impact on the training time, but thanks to the gradient approach it is limited. During detection, moreover, the size of the space does not play any role, thanks to the integral image representation (4).

Figure 6 shows the training and test errors using the FFT. The choice for the optimal number of features does not have the same importance as in other machine learning algorithms (e.g., neural networks) because of the intrinsic resistance of ADABOOST to over-fitting: even if the training error tends to zero, the test set error does not increase. It is therefore less important to find a specific stopping point, except for efficiency reasons. We can observe that at a frame level, on a simple FFT we already obtain an error rate of about 13% after 150 iterations, which is far better than the 37% of the best frame-level feature in (Scheirer and Slaney, 1997). We then decided to tie the classification of a frame to classification at previous frames with a simple smoothing function. Let $f(\mathbf{x}_\tau)$ be the output of the strong learner, where $\mathbf{x}_\tau$ is a frame at time τ. Then, the new output used for classification is

$$g(\mathbf{x}_\tau) = \frac{\sum_{i=max(\tau-nframes+1,0)}^{\tau} a^{\tau-i} f(\mathbf{x}_i)}{\sum_{j=max(\tau-nframes+1,0)}^{\tau} a^{\tau-j}}, \quad (5)$$

where a is a decay parameter between 0 and 1 and where $nframes$ is an integer that corresponds to the number of past frames to consider. In order to find the best values of a and $nframes$, we randomly concatenated the audio (wave) files of the validation set and measured the classification error rates for several values for the decay parameter. We chose this procedure in order to approximately simulate audio streaming from a radio station and get the best values at $a = 0.98$ and $nframes = 40$. With these settings the error reaches a value less than 7% with less than a second of information. The error converges after 150 iterations, but even with a much smaller number of features, such as 75, the error level is below 10%.

Surprisingly, RASTA and logarithmic scale FFT representation did not perform as well, even though the results are still below 10%. Table 1 summarizes the errors

Table 1: Testing error rate using single frame filters.

	FFT	**RASTA**	**Log-FFT**
Without smoothing	11.9%	10.8%	12.6%
With smoothing	6.7%	7.2%	7.4%

[3]The data was collected at random from the radio by Eric Scheirer during his internship at Interval Research Corporation in the summer of 1996 under the supervision of Malcolm Slaney.

on the test set with these representations. Also, RASTA and Log-FFT converged much faster than simple FFT, which can be explained in both cases by the higher quality of the information and the limited dimensionality.

3 A Winamp Plugin

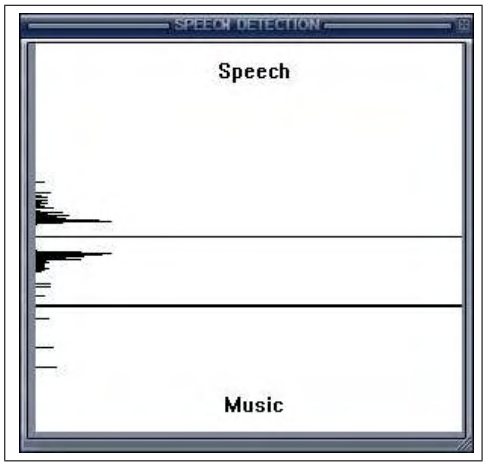

Figure 7: A Winamp plugin version of our model. The long thin horizontal line displays the decision boundary between Speech (top) and Music (bottom). The long thick horizontal line displays the decision of the model, with vertical position proportional to confidence. Here, for example, the model is moderately confident that the stream is Music (in this case, a correct prediction). The shorter horizontal lines on the left provide a histogram of the decisions and confidence ratings of the weak learners. See text for details.

Winamp (`www.winamp.com`) from Nullsoft is a popular audio player for the Windows operating system. Like many multimedia software applications, Winamp uses modular "plug ins" that extend the functionality of the basic player. Because the development kit for writing these plugins is available for public use, we were able to develop a version of our model that runs within Winamp. This plugin is able to predict in real time whether the audio stream being played in the Winamp player contains music or speech.[4] Because the plugin is distributed already trained, it requires few computational resources and should run well on any computer that can run Winamp.

In Figure 7 a screenshot of the plugin is shown. The long thick horizontal line displays the decision of the model. The distance of this decision line from the boundary between Speech (top) and Music (bottom) is proportional to the confidence the model has in its decision. This confidence is computed as a sum of weak learner votes (+1 for Speech, -1 for Music) multiplied by the respective weak learner confidences (alpha, α, in ADABOOST). See Section 2.1 for details.

The shorter horizontal lines on the left in the screenshot display a histogram of the weak learner votes where the vertical position y of each of these shorter lines is given by the vote (+1 for Speech, -1 for Music) multiplied by confidence. In addition, confidence is also used to scale the width of each of these lines. If two weak learners fall on the same vertical position in the plot, their confidence is summed, generating a single longer line. The three horizontal lines near the bottom of the plot are individual weak learners having particularly high confidence that the stream is music.

The plot is refreshed every 20ms, resulting in a smooth animation of model performance. Though the graphical interface is relatively primitive, it can be instructive to watch model behavior evolve over time in response to an audio stream.

For the Winamp plugin, we trained a version of the classification model using randomly-selected segments from a number of Internet radio streams. Labeling was achieved implicitly by treating talk-radio streams as speech examples and music-radio streams as music examples. Though training streams were not labeled by hand, we did control against contamination of speech with music and vice-versa through careful listening. We did not create a hand-labeled testing set of mixed speech and music radio streams. For this reason we are unable to report a reliable error rate for the Winamp plugin (nor was this our goal). We were able to demonstrate that the model runs efficiently and perform well in a real-world application. We observed that the output of the model is stable and reliable for a range of input streams.

4 Conclusions

We have showed how a simple generic object recognition algorithm can be used also to perform frame-level classification of audio by exploiting geometric regularities in a fixed-sized two-dimensional representation of frame contents. Because of the strong relationship among frames in time, we can increase the performance of the classifier with a simple smoothing on the output of the frame-level classifier. It is also possible to do training directly on a set of subsequent frames to capture local dependencies in the time domain. However, such an approach would have to deal with the problem of the absolute position of the features in time, and therefore would probably be working if the number of frames is limited.

Also it may be helpful to explore the use of different basic features (such as Gaussians or band-passes), and different representations such as wavelets or sine-wave replicas (Liebenthal et al., 2001).

Finally, while the current model is limited to two-class categorization, we are exploring a multi-class version of ADABOOST (Schapire, 1999). This would allow us to extend our work to more challenging classification problems such as speaker identification singer identification, music instrument identification and music genre classification.

ACKNOWLEDGEMENTS

We would like to thank Stanislas Lauly for help with working with data, and Hugo Larochelle for his suggestions. We would also like to thank Dan Ellis for helpful comments and for providing the dataset.

[4]The plugin is available for free download at `www.iro.umontreal.ca/~casagran/winamp/index.html`.

References

Jitendra Ajmera, Iain A. McCowan, and Hervé Bourlard. Robust HMM based speech/music segmentation. *Proc. of ICASSP-02*, 2002.

Gerard Escudero, Lluis Mrquez et German Rigau. Boosting Applied to Word Sense Disambiguation *Proceedings of the 12th European Conference on Machine Learning, ECML. Barcelona*, Spain. 2000.

H. Ezzaidi and J. Rouat. Speech, music and songs discrimination in the context of handsets variability. *Proc. of ICSLP 2002, 16-20* September 2002.

Yoav Freund, Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139, August 1997.

Hynek Hermansky, Nelson Morgan, Arun Bayya, Phil Kohn. RASTA-PLP speech analysis technique. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992. ICASSP-92.

Einat Liebenthal, Jeffrey R. Binder, Rebecca L. Piorkowski and Robert E. Remez. Sinewave speech/nonspeech perception: An fMRI study. *The Journal of the Acoustical Society of America* May 1, 2001 - Volume 109, Issue 5, pp. 2312-2313.

Constantine P. Papageorgiou, Michael Oren, Tomaso Poggio. A general framework for object detection. *International Conference on Computer Vision*, 1998.

John Saunders. Real-time discrimination of broadcast speech/music. *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing (Atlanta, GA)*, pp. 993-996, May 1996.

Robert E. Schapire. A Brief Introduction to Boosting. *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

Eric Scheirer, Malcolm Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. *Proc. of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Volume 2, pp. 1331, 1997.

Paul A. Viola, Michael J. Jones. Robust Real-time Object Detection. *International Conference on Computer Vision*, pp. 747, 2001.

Gethin Williams and Daniel P.W. Ellis. Speech/music discrimination based on posterior probability features. *Proc. European Conference on Speech Communication and Technology*, Sept. 1999, pp. 687-690.