A Filter-Based Approach to Detect End-of-Utterances from Prosody in Dialog Systems

Olac Fuentes, David Vera and Thamar Solorio

Computer Science Department

University of Texas at El Paso

El Paso, TX 79968

Abstract

We propose an efficient method to detect end-of-utterances from prosodic information in conversational speech. Our method is based on the application of a large set of binary and ramp filters to the energy and fundamental frequency signals obtained from the speech signal. These filter responses, which can be computed very efficiently, are used as input to a learning algorithm that generates the final detector. Preliminary experiments using data obtained from conversations show that an accurate classifier can be trained efficiently and that good results can be obtained without requiring a speech recognition system.

1 Introduction

While there have been improvements and a significant number of methods introduced into the realm of dialog-based systems, there are aspects of these methods which can be further improved upon. One such aspect is end-of-utterance (EOU) detection, which consists of automatically determining when a user has finished his/her turn and is waiting to receive an answer from the system. Current dialog-based systems use a simple pause threshold, which commonly results in either unnecessary long waiting times or interruptions from the system when the user makes a pause in the middle of an utterance. These problems can annoy and discourage users using even simple dialog systems.

Most previous methods aimed at improving upon pause thresholds for detecting end-of-utterances use spectral energy measures (Hariharan et al., 2001; Jia and Xu, 2002). Other methods use prosodic features with (Ferrer et al., 2002) and without speech recognition systems (Ferrer et al., 2003) in conjunction with decision trees to determine end-of-utterances as quickly as possible. For this and related problems, the choice of features is critical. Most common is to use a fixed inventory of features, chosen based on the linguistics literature and past experience (Shriberg and Stolcke, 2004). Recently we have experimented with alternative approaches, including features hand-tailored to specific discrimination problems (Ward and Al Bayyari, 2006) and random exploration of the feature space (Solorio et al., 2006). In this paper we explore yet another approach, using a large battery of very simple and easy to evaluate features.

In this paper we present a method to improve the accuracy that can be obtained in end-of-utterance detection that uses prosodic information only, without a speech recognizer. We adapt and extend a filter-based approach originally proposed in computer graphics (Crow, 1984) and later exploited successfully in computer vision (Viola and Jones, 2001) and music retrieval (Ke et al., 2005).

Our approach consists of applying simple filters, which can be computed in constant time, in order to generate attributes to be used by a learning algorithm. After the attributes have been generated, we test different learning algorithms to detect end-of-utterances. Our results show that the features yield good results in combination with several of the classifiers, with the best result being obtained with bagging ensembles of decision trees.

2 Method

The first stage in our system is to extract prosodic information from the raw audio signal. Using the audio analysis tool Didi, the log energy and fundamental frequency signals are extracted from the source sound wave. After computing log energy and pitch, we apply a large set of filters in the time domain to the energy and pitch signals in order to generate attributes suitable for classification. We compute the filter responses for both signals at every time step using three types of filters, each applied at many different times scales.

The first filter type, shown in Figure 1a), is a two-step binary filter, split approximately in half. The first half of the filter consists of a sequence of 1's. The second half consists of -1's. The second filter type is a three-step binary filter (as shown in Figure 1b)), split in approximate thirds alternating between 1 and -1. Finally, the third filter is an upward slope ranging from -1 to 1.

Although simple, these filters, in particular when they are applied at multiple scales, can characterize most of the prosodic features that are known to be relevant in identifying dialog phenomena including raises and falls in pitch and pauses of different lengths.

The response of any of these filters over the signal at any time is given by the dot product of the filter and signal

a) Type I filter
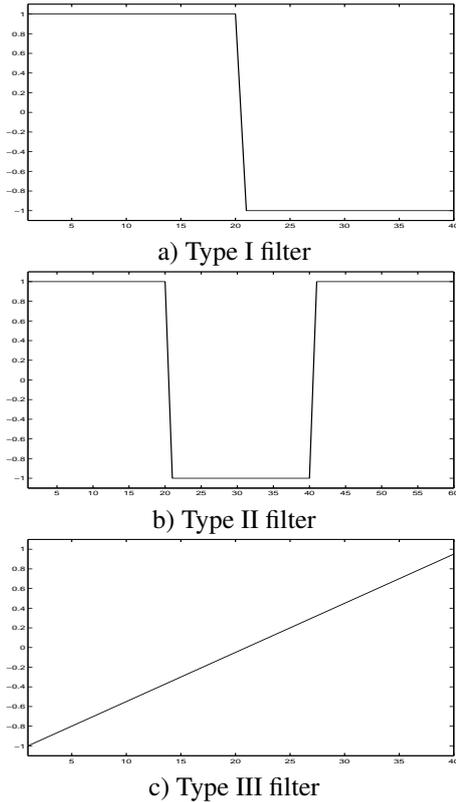
b) Type II filter

c) Type III filter

Figure 1: The three types of filters, the first two being binary only having values -1 or 1, and the last having an upward slope from -1 to 1.

window of the same length. Computing this dot product is slow, especially over larger time window sizes. This cost is even greater when many filter responses are taken over the course of the entire signal length.

Given the large number of filters and the size of a normal audio signal, the straightforward dot-product-based computation of the filter responses is prohibitively expensive. Fortunately, it is possible to device methods to compute these responses efficiently, as explained in the next subsection.

2.1 Efficient Filter Computation

This constant time computation of binary filters for two-dimensional signals was first presented by Crow (Crow, 1984) in the field of computer graphics and later applied successfully in computer vision (Viola and Jones, 2001). Here we show how that can be adapted to one-dimensional signals and extended to the case of non-binary filters, such as ramps.

Let s be the signal corresponding to either the log energy or the fundamental frequency. Let f be a filter of size n (in arbitrary time units) and let k be the time instant for which we want to compute the filter response.
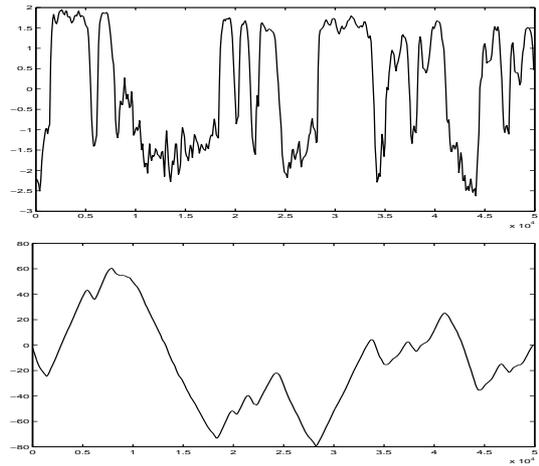

Figure 2: Energy, with mean subtracted, and its corresponding integral signal.

The filter response F is given by

$$F(s,f,k) = \sum_{i=0}^{n-1} s_{k+i} * f_i$$

The standard computation of F takes $O(n)$ operations; however, for the special case of binary filters like the ones shown in figures 1a) and 1b), we can compute this response in constant time with some preprocessing as follows. Let I be the integral signal, where each element of I is given by

$$I_j = \sum_{i=0}^{j} s_i$$

It can be seen that

$$\sum_{i=j}^{k} s_j = I_k - I_{j-1}$$

Thus this summation can be computed with two accesses to memory, after pre-computing and storing the values of I in an array. Figure 2 shows an example of a signal (with its mean subtracted) and the corresponding integral signal.

Consider a binary filter f such as the one shown in 1a), $f = \{1^{n/2}, -1^{n/2}\}$, that is, f consists of $n/2$ ones followed by $n/2$ negative ones. Then the filter response of a signal can then be computed in constant time using three references to the integral signal I:

$$F(s,f,k) = 2I_{k+n/2-1} - I_{k-1} - I_{k+n-1}$$

Similarly, the response to a filter like the one shown in Figure 1 b), given by $f = \{1^{n/3}, -1^{n/3}, 1^{n/3}\}$ can be computed with four memory references.

$$F(s,f,k) = I_{k+n-1} - 2I_{k+2n/3-1} + 2I_{k+n/3-1} - I_{k-1}$$

The third filter is an upward ramp ranging from -1 to 1. Whereas the binary filters are simple to calculate using look-up values, and their application to 1-dimensional signals is a simple adaptation to the 2-D algorithm, a ramp is more difficult and requires separate preprocessing for filters of different lengths. Regardless, it is still possible to compute its response in constant time after preprocessing.

We define a ramp filter of length n as $f = \{-1, \frac{2}{n-1} - 1, \frac{4}{n-1} - 1, ..., 1 - \frac{2}{n-1}, 1\}$. The response to this filter is

$$F(s, f, k) = \sum_{i=0}^{n-1} s_{k+i} * f_i$$

$$= \sum_{i=0}^{n-1} (\frac{2i}{n-1} - 1)s_{k+i}$$

$$= \frac{2}{n-1} \sum_{i=0}^{n-1} i s_{k+i} - \sum_{i=0}^{n-1} s_{k+i}$$

$$= \frac{2}{n-1} \sum_{i=0}^{n-1} i\, s_{k+i} - (I_{n-1} - I_{k-1})$$

Let $\sum_{i=0}^{n-1} i\, s_{k+i}$ be denoted by An_k. Clearly, if An_k can be computed in constant time, then $F(s, f, k)$ can also be computed in constant time. This can be done, with a little preprocessing, as follows. Let An_0 be computed in the standard ($O(n)$ time) way,

$$An_0 = \sum_{i=0}^{n-1} i s_i$$

Then to compute values of An_k for $k > 0$ we first observe that

$$An_k = s_{k+1} + 2s_{k+2} + \ldots + (n-1)s_{k+n-1}$$

and

$$An_{k+1} = s_{k+2} + 2s_{k+3} + \ldots + (n-1)s_{k+n}$$

From this, we can see that

$$An_{k+1} = An_k - s_{k+1} - s_{k+2} - \ldots$$

$$-s_{k+n-1} + (n-1)s_{k+n}$$

$$= An_k - \sum_{i=k+1}^{k+n-1} s_i + (n-1)s_{k+n}$$

$$= An_k - (I_{k+n-1} - I_k) + (n-1)s_{k+n}$$

Thus after pre-computing vectors I and An, which takes linear time in the size of the signal, we can compute any filter response in constant time. However, while we can derive all binary-filter responses from vector I, computing ramp-filter responses requires the pre-computation of a separate An for every filter length n. Nevertheless, this cost is small compared to the cost of computing a dot product for every time instant in the input signal.

The integral signal representations are computed from the two prosodic feature signals, and filter features are calculated along the timeframe of the signal. Once the filter responses are obtained, they are used as attributes for machine learning algorithms, which are used to generate the final classifiers. The data is then used to train several learning algorithms, as implemented in Weka (Witten and Frank, 2005).

3 Experimental Results

Experiments were conducted to test the end-of-utterance system on pre-recorded conversations, measuring precision and recall of positive classifications. The conversations used contained speech by both male and female users to compare the robustness among different vocal range frequencies.

3.1 Data

The training set of data is derived from about 22 minutes of conversations, with the audio split such that each speaker is on a separate voice channel (see (Hollingsed, 2006)). In 17-minutes worth of these, volunteers were asked to list eight exits on the east side of El Paso on Interstate 10. This provided a clear way to measure end-of-utterances as if a system were prompting users for input. This set of conversations contained a large number of turn-switches, which also simulated voice-portal systems well. For most of the time in this set, the same person (a female) is conducting the quiz. However, the speakers taking the quiz have distinctly different voices and are mixed in gender.

Five minutes of the training set were taken from a casual conversation also containing a male and female speaker combination. The speakers in this conversation are different from the speakers in the other dataset. Adding these data balances the training set, reducing the probability of the system learning only the specific quiz format used in much of the training data.

Didi was used to extract the prosodic features, and the filter responses were computed for each of the three filter types, in sizes ranging from 200ms to 3 seconds in increments of 50ms, totaling 342 features per time instance. The class was set to 0 or 1, signaling non-end-of-utterances and a confirmed end-of-utterances, respectively. 992 instances were created for the experiments, split equally in two between positive examples of end-of-utterances, and randomly selected negative examples for both channels in the source audio.

Table 1: Experimental results of using different classifiers and averaging ten ten-fold-cross-validation evaluations with random seeds per classifier.

	Recall	Precision	F-Measure
Dec.Stump	0.623	0.705	0.660
Dec.Table	0.768	0.799	0.783
C4.5	0.792	0.800	0.796
Boost(DS)	0.792	0.820	0.806
Bag(REP)	0.850	0.833	0.841
Bag(C4.5)	0.786	0.797	0.791

All instances used for training were randomly chosen. The positive examples were chosen from human-determined end-of-utterance intervals, which ranged from the time instant a valid end-of-utterance was recorded to a point either 1.5 seconds after that instant or a start-of-utterance that occurred prior to that time. The negative examples were randomly chosen such that no time instance was chosen prior to the 3-second-mark of the audio file used and none was within a marked end-of-utterance interval.

3.2 Results

Six combinations of classifiers were generated using the Weka data mining tool. Each of these classifier combinations was tested using 10-fold cross-validation. The results reflect the average of ten such cross-validation runs, each using a different random seed. The final classifier combinations used are Weka's implementations of decision stumps, decision tables, C4.5 (Quinlan, 1993) and ensembles of decision stumps using boosting and C4.5 and reduced error pruning (REP) decision trees (Quinlan, 1987) using bagging.

The experiments performed yield interesting results. Table 1 shows that, with the exception of decision stumps, which are perhaps too simple for this task, all classifiers performed well, which shows that our filters produce suitable features for classification. The best results were obtained using bagging and REP trees, but results for other methods yield similar precision and recall.

It is almost certain that better results can be obtained using these methods if bleeding across channels in the audio streams was reduced. The F0 features do a good job of filtering out possible mistakes in the system due to the way the frequencies are calculated. However, bleeding can still mislead the classifiers into perceiving an end-of-utterance from another speaker.

4 Conclusions and Future Work

We have shown a new filter-based method for detecting end-of-utterances in conversation using only basic prosodic information. We adapted and extended previously described methods for fast computation of filter responses, which allows our system to be trained quickly and easily permits real-time performance. Preliminary experiments in the task of classifying windows in dialog recordings as being end-of-utterances or not have yielded very promising results using standard classification algorithms, with an f-measure of 0.84.

Present and future work includes evaluating the method as a component of a real-time dialog system, where its usefulness at decreasing waiting time can be tested. We are also working on methods for feature selection and compression to obtain further speedup, and finally we are experimenting with larger datasets.

References

F. C. Crow. 1984. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*.

L. Ferrer, E. Shriberg, and A. Stolcke. 2002. Is the speaker done yet? Faster and more accurate end-of-utterance detection using prosody. In *Proceedings of ICSLP*.

L. Ferrer, E. Shriberg, and A. Stolcke. 2003. A prosody-based approach to end-of-utterance detection that does not require speech recognition. In *Proceedings of IEEE ICASSP*.

R. Hariharan, J. Hakkinen, and K. Laurila. 2001. Robust end-of-utterance detection for real-time speech recognition applications. In *Proceedings of IEEE ICASSP*.

T. K. Hollingsed. 2006. Responsive behavior in tutorial spoken dialogues. Master's thesis, University of Texas at El Paso.

C. Jia and B. Xu. 2002. An improved entropy-based endpoint detection algorithm. In *Proceedings of ISCSLP*.

Y. Ke, D. Hoiem, and R. Sukthankar. 2005. Computer vision for music identification. In *Proceedings of IEEE CVPR*.

J. R. Quinlan. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufman.

E. Shriberg and A. Stolcke. 2004. Direct modeling of prosody: An overview of applications in automatic speech processing. In *Proceedings of ICSP*.

T. Solorio, O. Fuentes, N. Ward, and Y. Al Bayyari. 2006. Prosodic feature generation for back-channel prediction. In *Proceedings of Interspeech*.

P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE CVPR*.

N. Ward and Y. Al Bayyari. 2006. A case study in the identification of prosodic cues to turn-taking: Back-channeling in Arabic. In *Proceedings of Interspeech*.

I. H. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufman.