A NEW FAST ALGORITHM FOR AUTOMATIC SEGMENTATION OF CONTINUOUS SPEECH

Iman Gholampour,Kambiz Nayebi

Electrical Engineering Department, Sharif University of Technology, Tehran, Iran

ABSTRACT

In this paper a new method for automatic segmentation of continuous speech into phone-like units is addressed. Our method is based on a very fast presegmentation algorithm which uses a new statistical modeling of speech and searching in a multilevel structure, called *Dendrogram*, for decreasing insertion rate. In each step the performance of algorithms have been tested over a large set of *TIMIT* sentences. According to these tests, our final segmentation algorithm is capable of detecting nearly 97% of segments with an average boundary position error of less than 7 msec and average insertion rate of less than 12.6%. The paper will describe the algorithms for determining the acoustic segments. Performance results will also be included.

1. INTRODUCTION

The problem of segmentation of continuous speech is one of the most complicated problems in speech processing. There are a large variety of applications which more or less depends on solving this problem including automatic segmentation of continuous speech corpus, automatic speech recognition and very low bit rate speech coding (segment vocoders). Unfortunately there is no complete solution for this problem, but several algorithms have been proposed for approximate solutions. Ideally, a segmentation algorithm should be able to reliably detect abrupt acoustic events as well as gradual ones. In practice we can only separate homogeneous parts of speech. So, boundary missing and additional boundary insertion are unavoidable. All we can do is to optimize the system parameters to minimize missing and insertion rates. In 1988, Glass and Zue presented a multi-level description of speech segments which contains both coarse and fine information in one uniform structure, called *dendrogram*[1]. Their analysis of dendrogram showed that it can *capture* more than 96% of acoustic phonetic events of interest with an insertion rate of less than 5%. This was a major improvement over the best results previously reported with single-level representation which had a combined deletion and insertion rate of 25% [1]. But they suggest nothing about automatic selection of segments from dendrograms. Our work is in some sense a continuation of Glass and Zue's ideas, but different presegmentation scheme is adopted and search algorithms are added for automatic selection of the best set of segments from dendrograms. The first algorithm is chosen to search for the best path (namely the best collection of segments) through dendrograms without using any additional information. The second one uses some information about phonetic contents of speech utterances and their lengths to constrain the search space. So one can divide our proposed strategy into the following steps :

1) Presegmenting the input speech (16 KHz, 16 bits).

2) Constructing the so called dendrogram using presegmentation results and spectral characteristics of presegments.

3-a) Searching for the best path through the dendrogram using a *Dynamic Programming* approach.

3-b) Searching for the best path using dynamic programming and some information about the phonemic contents of speech.

Its worth mentioning here that (3-b) can be used if the phonemic contents of the input utterances are known. This is useful for segmentation of speech databases in which all the required information is available. Details of the algorithms for each step and results of their performance tests are presented in this paper.

2. PRESEGMENTATION

Keeping the above problems in mind we design a presegmentation algorithm which has very low missing rate and boundary position error, without any control on its insertion rate. So the insertion rates may become too high. We leave the solution of this problem to the next step, which uses several kinds of information about speech segments to decrease insertion rate. Our presegmentation algorithm is based on a new statistical similarity measure between short speech segments. First, we assumed that the short time PDF of speech samples is approximately *Laplacian* [2], [3] :

$$f_\alpha(x) = \frac{1}{2}\alpha e^{-\alpha|x|} \qquad (1)$$

where x is a sample in short time speech frame. In addition to having very good precision, the laplacian PDF has an advantage of incorporating only one parameter which can easily be estimated from speech frame energies. Assuming ergodicity of speech samples in short time frames we have :

$$\prec x_n^2 \succ = \frac{1}{N}\sum_{n=1}^{N} x_n^2 = \int_{-\infty}^{+\infty} x^2 f(x)dx = \frac{2}{\alpha^2} \qquad (2)$$

from which we can easily conclude that :

$$\alpha = \sqrt{\frac{2}{\prec x_n^2 \succ}} \qquad (3)$$

where x_n denotes speech samples in a short time frame that

contains N samples and $\prec \cdot \succ$ denotes time averaging. for testing the Laplacian hypothesis, we compared the normalized amplitude histogram of every 5 msec frame in 840 TIMIT speech files with its estimated Laplacian PDF using Eq. (2). For this comparison we chose *Kullback* distance as a distance measure between two estimated PDFs [3]. This test shows that the Kullback distance is always less than '2' for speech frames and greater than ' 2' for the frames which contains silence, stop clusures and some weak fricatives. According to these results we designed a simple algorithm for distinguishing between speech and silence frames. The position of stop closures and some week fricatives can also be found by this algorithm. we use these information later in our search algorithms. For each pair of speech frames we adopted a similarity measure based on Laplacian models of these frames (namely the Laplacian PDF and its α parameter) and Kullback distance between these models which is defined as :

$$D(f_{\alpha_1} \| f_{\alpha_2}) = \int_{-\infty}^{+\infty} f_{\alpha_1}(x) ln \frac{f_{\alpha_1}(x)}{f_{\alpha_2}(x)} dx \qquad (4)$$

it can be shown easily that for Laplacian PDFs , f_{α_1} and f_{α_2} :

$$D(f_{\alpha_1} \| f_{\alpha_2}) = ln \frac{\alpha_1}{\alpha_2} + \frac{\alpha_1 - \alpha_2}{\alpha_1} \qquad (5)$$

this distance function does not have commutative property, so we use a slightly modified distance measure which is defined as :

$$D(f_{\alpha_1}, f_{\alpha_2}) = D(f_{\alpha_1} \| f_{\alpha_2}) + D(f_{\alpha_2} \| f_{\alpha_1}) \qquad (6)$$

For Laplacian PDFs, f_{α_1} and f_{α_2} Eq. (6) yields :

$$D(f_{\alpha_1}, f_{\alpha_2}) = \frac{(\alpha_1 - \alpha_2)^2}{\alpha_1 \alpha_2} \qquad (7)$$

which is our final choice of similarity measure between speech frames. Our presegmentation strategy then begins with three 5 msec frames F_1, F_2 and F_3 with Laplacian PDF parameters α_1, α_2 and α_2 respectively which are shown in Figure (1-a).

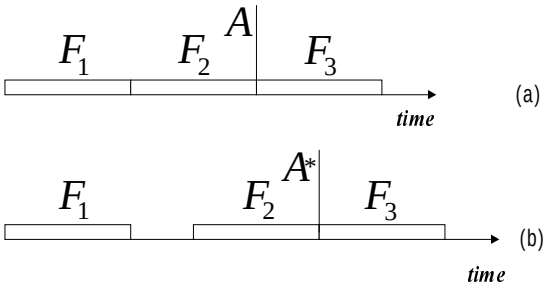

Figure 1 : Frame positions in presegmentation strategy : (a) Beginning state, (b) Final state.

Then we classify F_1 as speech or silence. If it contains silence, we shift F_1, F_2 and F_3 with steps of 2.5 msec to right until it contains speech. In the next step we compute the similarity between F_1, F_2 and F_2, F_3 using Eq. (7). If F_2 is more similar to F_1 than to F_3, we shift F_2 and F_3 2.5 msec to right while F_1 remains unchanged. This step is repeated until F_2 becomes more similar to F_3 than to F_1. In this case we adopt the point A^* in Figure (1-b) as a new boundary. This process is repeated from the new boundary until the input utterance is completely covered. As a result of incorporating only one parameter for each speech frame, this algorithm is very fast and can easily be implemented in real time on an average Pentium PC. Performance test of presegmentation section has been performed over 840 SX sentences in Test directory of TIMIT. Figure (2) shows the histograms of missing rate, average position error and insertion rate for each test sentence. In each histogram, the vertical axis depicts the number of sentences for each region of horizontal axis. For example, there are over 700 sentences in 840 sentences which have missing rate of less than 1%. Unfortunately, in spite of very low averages in missing rates and boundary position errors (0.54% and 3.5 msec respectively), the insertion rate is too high (over 150% in an average). The next part of this paper reports the results of our efforts in the direction of decreasing insertion rates.

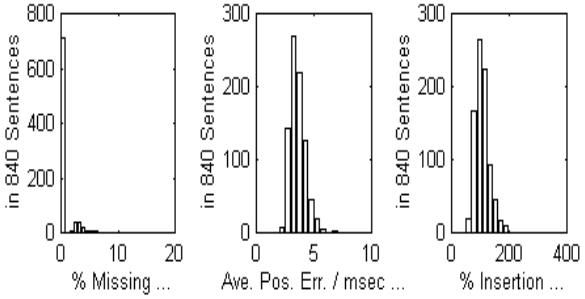

Figure 2 : Test Results of presegmentation algorithm**.**

3. DECREASING INSERTION RATES

As we stated earlier, our strategy is to construct a multi-level description of segments and searching through it for the best collection of segments which covers the input utterance. In the following subsections we will show that significant decrease in insertion rate can be achieved without considerable change in missing rate and boundary position error. We use the algorithm originally proposed by Glass and Zue for constructing dendrograms from the resulting segments of the pre-segmentation process [1], [4], [5]. This multi-level representation is attractive because it is able to capture both coarse and fine information in one uniform structure. Acoustic phonetic segmentation can then be formulated as a path-finding problem in a highly constrained search space[1]. Figure (3) shows this representation for the utterance "*Don't ask...*"'.

3.1. Searching Dendrograms

The basic idea behind this section is searching for a path from the beginning to the end of a dendrogram (corresponding to the beginning and the end of input utterance) such that almost always the highest rectangles are used. For this purpose, we examined several kinds of optimality criterion which led us to the following one. We recall that every segment in the

dendrogram has two parameters, namely segment height and width, as depicted in Figure (3). We define the optimal path as a continuous collection of dendrogram's rectangles, which has minimum sum of width to height ratio. With this criterion, the height of any segment is uniformly distributed over the section of time axis which is covered by that segment. The cost of passing through any segment is also defined as the reciprocal of its corresponding region's weight. Then the search problem can be stated as finding the minimum cost path through the time axis, which is weighted partially by several segments.

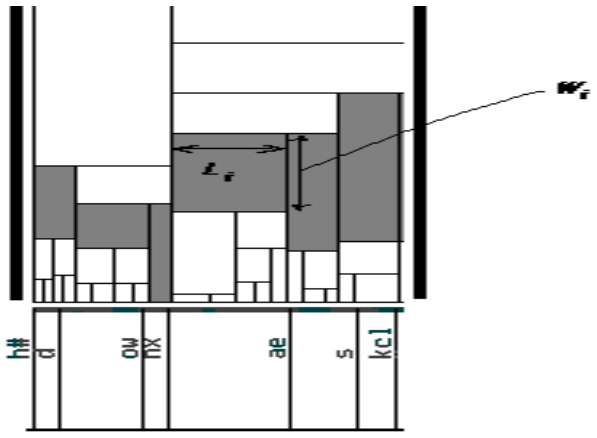

Figure 3 : A sample dendrogram representation for utterance "Don't ask ..."

We present here, a *Dynamic Programming* type solution for this above problem which is based on a *Graph theoretic* point of view [6]. First of all, every boundary in the presegmentation output is defined as a directional graph *node*. So, every dendrogram's rectangle can be thought as an *edge* of this graph, which makes a weighted connection between two nodes. Therefore, for the ***i***th rectangle in the dendrogram, we have an edge of weight L_i/D_i in our graph (L_i and D_i are the rectangle height and width respectively). The graph direction is defined as the direction of the time axis. Our algorithm then searches for the minimum cost path from the starting node to the final one. The path cost is defined to be the sum of reciprocals of its corresponding edge weights. Here is our proposed steps for performing the desired searches :

- Initialization :

$S_0 = \{0\}$; $u_0 = 0$; $i = 0$;

- Recursion :

$$\forall v \in \bar{S}_i : \begin{cases} D_{i+1}(v) = min\{ D_i(v), D_i(u_i) + W(u_i, v)\}; \\ P(v) = u_i, if\ (D_{i+1}(v) \neq D_i(v)); \end{cases}$$

$$u_{i+1} = arg \min_{v \in \bar{S}_i} \{ D(v)\};$$

$$S_{i+1} = S_i \cup \{u_{i+1}\}; \quad i = i + 1;$$

Break if $i \geq N - 1$, else continue;

- Back tracking :

$$BestPath(i) = P(N - 1 - i); \quad i = 0,1,\ldots,N-1;$$

N : Number of graph nodes (0 for starting nodes and N -1 for the final node).

u_i and v : Graph nodes.

$D(v)$: Cost of a path that connects the starting node to the v th node.

$W(u_i, v)$: Weight of the edge connecting the u_i th node to v th node.

$P(v)$: Predecessor of v th node in the best path.

A little inspection shows that above algorithm requires $5N^2/2$ computations and $3N$ memory locations. It can be shown that our algorithm finds not only the best path to the ending node, but also the best path to any other node of the graph from the starting nodes. Finally, the array D contains the costs of these optimal paths. Therefore, the algorithm can be used sequentially for real time applications. The performance of the algorithm was examined over 840 TIMIT sentences, using the presegmentation results of previous section for each sentence as an input. Figure (4) shows the test results. Referring to this figure the average boundary position error and the missing rate increase to averages of 6.8 msec and 2.84% respectively. But the important result is that the average of insertion rate decreases to 22.6% which is approximately $1/7$ of its value in the presegmentation stage. In the next section we use segment length information to decrease the insertion rate even more.

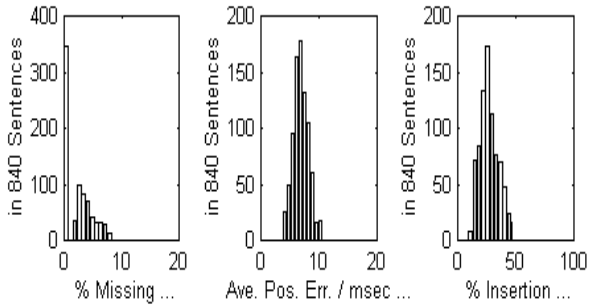

Figure 4 : The results of the first search algorithm.

3.2. Searching Dendrograms For Known Utterances

In this step, we tried to decrease the insertion rate even more when the phonemic contents of input utterances are known. As mentioned earlier, this is useful for automatic segmentation of speech databases where these information are available. First of all, we derived the length histograms for the 54 types of segments, introduced in the TIMIT's transcription. Figure (5) shows one of these histograms which is computed for the segments labeled as *'aa'* from 840 TIMIT transcription files (named as **.phn*). Table (1) outlines some information obtained

from these histograms for each segment type. By incorporating the above minimum and maximum length information in our search algorithm, several paths through the segmentation graph can be rejected in spite of their cost advantages. For this purpose we designed an algorithm which dynamically assigned valid regions on the time axis for each segment's boundaries of known input utterance, during the search time. This algorithm also used the information about closures and weak fricatives positions which our presegmentation algorithm can almost perfectly find them. Valid region assignment restarts after reaching to each segment of these types.

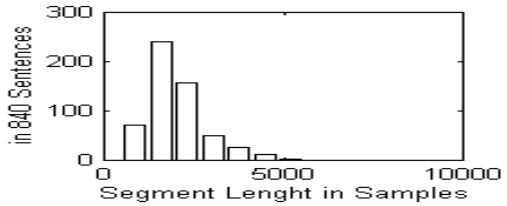

Figure 5 : A sample length histogram which is computed for segments labeled as *'aa'* in TIMIT transcriptions.

Segment	Min	Max	Ave.	Std.
aa	513	7735	2026.371	865.244
ae	943	5259	2153.972	716.612
ah	506	4480	1444.029	567.882
ao	520	5073	1941.280	732.385
aw	1077	6057	2873.734	884.599
b	63	949	283.678	116.993
bcl	209	3114	1084.352	443.766
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
z	460	4240	1433.151	559.891
zh	670	4218	1460.825	624.673

Table 1 : TIMIT segments length Information.

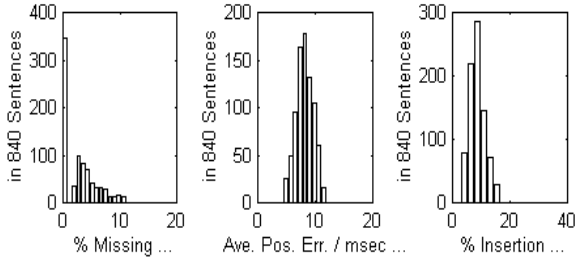

Figure 6 : Test Results of the second search algorithm.

We also designed an algorithm incorporating nearly valid (highly probable) regions instead of completely valid ones, but this algorithm fails for short utterances, which contains small number of phones. This is because of the fact that in utterances which have many phones, there are many segments with highly probable length, but in short utterances this may not be true. A new performance test has been performed, for the search algorithm based on valid regions. The insertion rates decreases to 12.5% without considerable changes in missing rates and boundary position errors (3.08% and 6.9 msec respectively). Figure (6) shows the test results similar to the previous test.

4. CONCLUSION

In summary, we have reported new fast algorithms for acoustic segmentation of continuous speech and results of their performance tests. These algorithms have been divided into three major steps and a performance test has been performed after each step. The 3rd step can be incorporated only if the phonetic contents of speech utterances are known. Test results are outlined in Table (2). According to this table and tests which are performed by Glass and Zue, our segmentation strategy is capable of detecting almost all of the boundaries that are captured in a dendrogram with low position errors and acceptable insertion rate [1]. Table (2) summarizes the performance results after using each algorithm. This table also shows that incorporating additional information can improve the performance of overall segmentation scheme. In addition to acceptable precision, our overall segmentation scheme has very low computation cost and it can be implemented in real time on an average Pentium PC. An important advantage of our algorithm is that no training or threshold computation is needed in realizing it. On the contrary, many known segmentation algorithms usually depend on critical thresholds which must be estimated before using them. Training or threshold estimation phase is generally the most critical and difficult step in realizing segmentation algorithms. This is possibly the most important advantage of our proposed algorithms.

	(a)	(b)	(c)
Presegmentation	0.54%	3.5 msec	>150%
1st Search Alg.	2.84%	6.8 msec	22.6%
2nd Search Alg.	3.08%	6.9 msec	12.5%

Table 2 : Results of performance tests for each step of the proposed algorithms, (a) Average missing rate, (b) Average boundary position error, (c) Average Insertion rate.

5. REFERENCES

[1] J. R. Glass and V. W. Zue, "Multi-level acoustic segmentation of continuous speech", Proc. IEEE ICASSP, 1988.

[2] T. W. Parsons, "Voice and speech processing", McGraw-Hill, 1986.

[3] J. R. Deller, J. G. Proakis and J. H. Hansen, "Discrete-time processing of speech signals", Macmillan, 1993.

[4] P. Cosi, "SLAM : Segmentation and Labeling Automatic Module", Proc. EUROSPEECH, 1993.

[5]M. S. Bagley, R. F. Lyon and M. A. Bush, "Acoustic-Phonetic segment classification and scale space filtering", Proc. IEEE ICASSP, 1987.

[6] J. A. Bondy and U. S. Murty, "Graph theory with applications", American Elsevier, 1976.