TR-IT-0200

A Study on Continuous Speech Recognition Based on Polynomial Segment Models

Sophie Aveline
ソフィー アベリン

Toshiaki Fukada
深田 俊明

1996.12.20

In this paper, we present a speech recognition system based on polynomial segment models (PSMs). To date, several PSM-based studies have been shown that the performance of the PSMs was better than that of regular HMMs. However, most of the comparisons have been done for classification tasks or for rescoring the HMM-based recognition results because the computational requirement for PSM is quite high. In our approach, to reduce the computational requirement dramatically, a recurrent neural network (RNN) based landmark detector, which can estimate boundary candidates of phonemes accurately, is first developed. Then, PSM-based recognition is performed by evaluating landmark candidates obtained from the landmark detector. Our preliminary experimental results on the TIMIT database showed that the proposed system gave equivalent recognition performance to that of a conventional HMM system.

Contents

1 Introduction

This system is used to improve present systems of speech recognition. These latter ones are based on the following principle: the recognition occurs at the same time as the separation of the phonemes (segmentation).

In this system of speech recognition based on polynomial segment models [2], there are two separated stages in the process. The first one is the separation of the phonemes through a bi-directional Recurrent Neural Network [8]. The other one is the recognition itself (identification). This is illustrated by Figure 1.

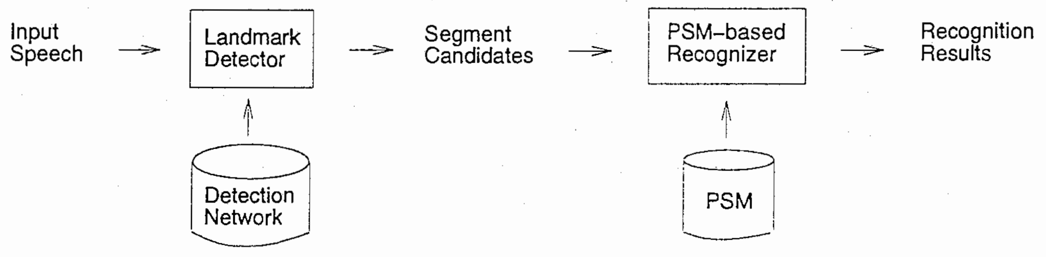


Figure 1: Principle of the system

Short explanation of each block of Figure 1:

- Detection Network: It is a Recurrent Neural Network trained with 3696 sentences in order to identify the boundaries of each phoneme.
- Landmark Detector: From the Input speech, with the RNN, it is able to determine the boundaries of the phonemes of the unknown sentences. Thus the system has demarcated each phoneme.
- PSM: Polynomial Segment Model, it tries to identify each phoneme in order to be able to reconstruct the sentence.
- PSM-based Recognizer: From the possible boundaries of phonemes obtained with the Landmark Detector, it identifies the most probable succession of phonemes.

First, we will explain how the Detection Network gives the boundaries of the phonemes. After that, we will show the working of the PSM-based Recognizer. Finally, we will present the results compared to the results of the HMM-based recognizer.

Part I

Landmark Detector

2 Detection Network

2.1 Inputs and Outputs

The inputs are 26 feature parameters which have been combined in order to take less memory.
Indeed, from the waveforms, feature parameters can be extracted through the HCode tool. The results are files whose extension is ".mfc". In these files, there are 12 parameters plus the energy for each frame (a frame covers 10ms of speech and they overlap each others). Then new files ".phnnew" are created. They contain all the phonemes of a sentence. Each phoneme is demarcated with the number of the frame in which it begins and the one in which it ends. After that, a codebook can be generated in order to reduce storage for spectral analysis information and computation for determining similarity of spectral analysis vectors. It contains vectors of 26 parameters. To finish this stage, the files are combined and then splited to make smaller files. These files will contain the inputs of the Recurrent Neural Network.

There is only one output for each frame. It is a value from 0 to 1, it represents the possibility for each frame to be a boundary of a phoneme. The value "1" is given when the frame is surely a boundary. The value "0" is given when the frame is not a boundary.

The Figure 2 illustrates a curve representing the true values of the boundaries and another one representing the estimated values of the boundaries of the frames. The first one is constructed from a file containing the number of the frames which demarcate each phoneme (1 if it is a boundary, and the values decrease to 0 through the values 0.75, 0.5, 0.25). The latter is the results of the RNN network for testing data with 20 forward and backward states, running on 100 cycles, it is the output of the Landmark Detector.

In the remaindering of these explanations, we will show the improvement of the system with the example I (Figure 2), which was tested with a RNN with 20 forward and backward states, 100 iterations.

2.2 Training and testing of the network

To train the Recurrent Neural Network (use of the tolls of M. Schuster [8]), we have 3696 files which contain different sentences. The goal is to find the best structure for this Recurrent Neural Network. This is realized by finding the network (right number of forward states and of backward states) that minimizes the Minimum Square Error (MSE). The program used is the Drnntrain one and the outputs are the files drnntrain_fXbX (X represents the number of forward and backward states).
The training of the network with different number of forward and backward states gives the results of the Table 1 (for 100 cycles).

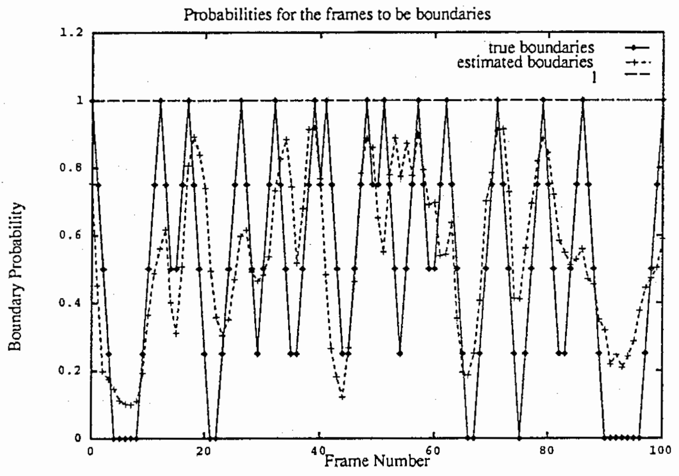


Figure 2: Curves of the true probability for the frames to be a boundary and the estimated one through the RNN network.

Nb of backward and forward states	MSE - training
2 - 2	8.419055e-02
5 - 5	7.043952e-02
10 - 10	6.323325e-02
20 - 20	6.087387e-02
30 - 30	5.928701e-02

Table 1: MSE-training values according the number of states in the network for 100 iterations.

The tests:

In order to test the recurrent neural network, we have a set of 1344 sentences. Thanks to the Drnntest program (the outputs are the files test.drnntest_fXbX), it is possible to test it and, in Table 2, there are the values of the MSE obtained according to the structure of the network for 100 cycles.

Nb of backward and forward states	MSE - testing
2 - 2	8.492939e-02
5 - 5	7.153326e-02
10 - 10	6.438741e-02
20 - 20	6.206396e-02
30 - 30	6.053929e-02

Table 2: MSE-testing values according the number of states in the network for 100 iterations.

Nb iterations	MSE - training	MSE - testing
20	1.034510e-01	1.016970e-01
50	7.248991e-02	7.450458e-02
100	6.087387e-02	6.206396e-02
200	5.606307e-02	5.771254e-02

Table 3: MSE - training values and testing values according to the number of iterations for a 20 forward and backward state network.

The values of the MSE are quite good but they are not sufficient to determine which number of forward and backward will give the best results to find the boundaries of the phonemes. As a network with 20 forward states and 20 backward states seems to be a good compromise between time and results, it is important to know if the choice of 100 cycles is suitable to study the performance.
The Table 3 displays the values of the MSE for the training and the testing of a network with 20 forward and backward states with 20, 50, 100, 200 iterations.

3 How to identify the possible boundaries

We would like to confirm that the network with 20 states and run on 100 iterations is a good one to estimate the results. Now, it is necessary to find among the frames which ones can be the boundaries. As the network is not perfect, the value 1 does not appeared in the outputs but some frames are boundaries.
That is the reason why some parameters need to be introduced to try to identify the boundaries of the phonemes.

The way to use all the following programs is explained in the Annex A.
The results shown in the remainder of this chapter are obtained for the testing data with a network whose structure is made of 20 forward and backward states. The number of cycles is 100 when it is not specified.

3.1 First Idea: a threshold and a margin

3.1.1 Conception

In order to separate the frames which can be boundaries of phonemes, the estimated boundaries are first considered as the points which are local maxima above a threshold. Then, it is important to see whether each estimated boundary corresponds to a true one. As the estimated boundaries and the true ones are not exactly in the same frame (it may be one frame of difference or more), the parameter "margin" allows an error around each true boundary and it is now possible to define the Insertion, Deletion numbers and the Accuracy.

This example underlines the importance of such a parameter. For instance, if margin equals 1 and there is a true boundary in the frame 95, then:

1. If there is no estimated boundary in the frames 94, 95, 96 then the Deletion number is increased of one unit.
2. If there is one estimated boundary in the frame 94, 95, 96, then nothing is changed.
3. If there are more than one estimated boundary (N) in the frames 94, 95, 96, then the Insertion number is increased of N-1 units.

The Accuracy is defined as:

$$Accuracy\ (\%) = 100 * \frac{N - D - I}{N}$$

with: N total number of real boundaries
I number of insertions
D number of deletions

Explanation of the results that are obtained for the example I (upper threshold = 0.7):

- The true boundaries are at the frames: 0, 12, 17, 26, 32, 39, 41, 48, 51, 57, 62, 71, 79, 86, 100.
- The estimated boundaries are at the frames: 0, 18, 34, 39, 48, 53, 55, 57, 72, 79.

- 15 True boundaries
- for margin = 0 Insertion = 5 (frames 18, 34, 53, 55, 72)

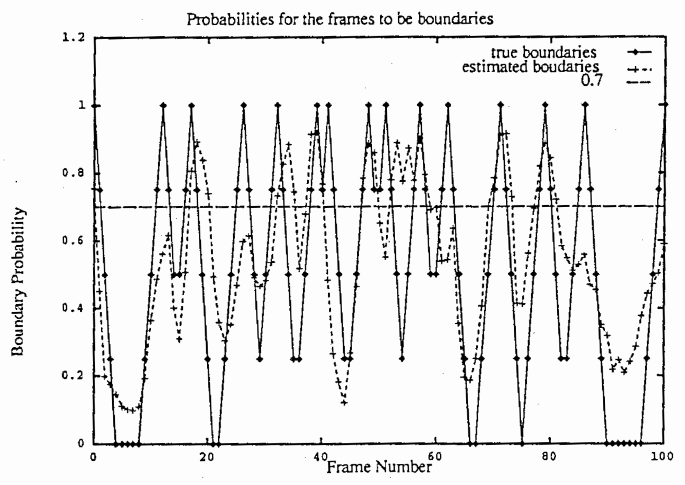


Figure 3: Curves of the true and estimated probabilities to have a boundary in each frame with the one threshold

Deletion = 10 (frames 12, 17, 26, 32, 41, 51, 62, 71, 86, 100)
Accuracy = 0

- for margin = 2 Insertion = 1 (frame 55, because the true one 51 goes with the estimated 53 and the true one 57 goes with the estimated 57)
Deletion = 6 (frames 12, 26, 41, 62, 86, 100)
Accuracy = 53.33

3.1.2 Results

To find those estimated boundaries, it is necessary to determine the accurate values for the threshold and for the margin. The Table 4 gives the value of the Accuracy depending on the threshold value (vertical) and the margin value (horizontal).

	Margin				
Threshold	0	1	2	3	4
0.4	-23.44	48.64	62.00	66.21	68.98
0.45	-19.57	50.77	63.60	67.55	70.12
0.5	-15.96	52.43	64.56	68.21	70.54
0.55	-12.53	53.31	64.70	68.04	70.17
0.6	-9.65	51.46	64.07	67.12	68.95
0.65	-7.25	52.31	61.78	64.35	65.87
0.7	-5.19	49.76	58.19	60.30	61.56
0.75	-3.48	45.34	52.57	54.15	55.03
0.8	-1.89	38.32	43.84	44.92	45.52

Table 4: Accuracy values according to the threshold and margin values.

The Figure 4 illustrates the variation of the Accuracy according to the threshold value for different margin values.

In Figure 4, the accuracy values for margin equal to 0 can be explained by the fact that the numbers of insertions and deletions are very high (very few frames of estimated boundaries exactly correspond to the true ones). When the threshold value increases, there are

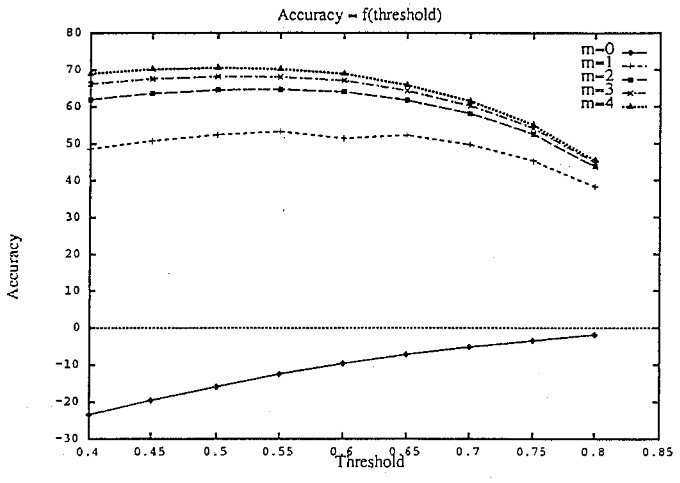


Figure 4: Accuracy curves according to the threshold for the margin equal to 0, 1, 2, 3 , 4.

less insertions because there are less local maxima above a high threshold and the number of deletions does not really change. That is why the value of the accuracy increases with the threshold.
The other curves illustrate the fact that, when the threshold value increases, the number of insertions decreases significantly but the Deletion increases more rapidly. So, for a given value of the margin, the accuracy decreases while the threshold increases.

For instance, if threshold = 0.7 and margin = 2, for the 1344 testing files (example II):
Number of true boundaries = 53006
Insertion = 2770
Deletion = 19390
Accuracy = 58.19
Percentage of deletion = 36.58

The percentage of deletion indicates the ratio of true boundaries which cannot be found among the estimated ones.

$$Percentage\ of\ Deletion\ (\%) = 100 * \frac{D}{N}$$

with: N total number of real boundaries
D number of deletions

Tables 5 and 6 show the variations of the Accuracy according to the margin value for different threshold values.
With the Figures 5 and 6, it seems that the best values are about 3 for the margin and about 0.5 for the threshold. Indeed, the curve for threshold equal to 0.6 is a mark to be able to compare these two figures. Consequently, the curve for the threshold 0.5 is the highest one.

However, the problem is to reduce the number of deletions. Indeed if the insertion is high, it is not so important because it is always possible to say that a frame is not a boundary of a phoneme, whereas if one frame is not in the list of the possible boundaries it will never be considered as a possible one, here is the real problem.

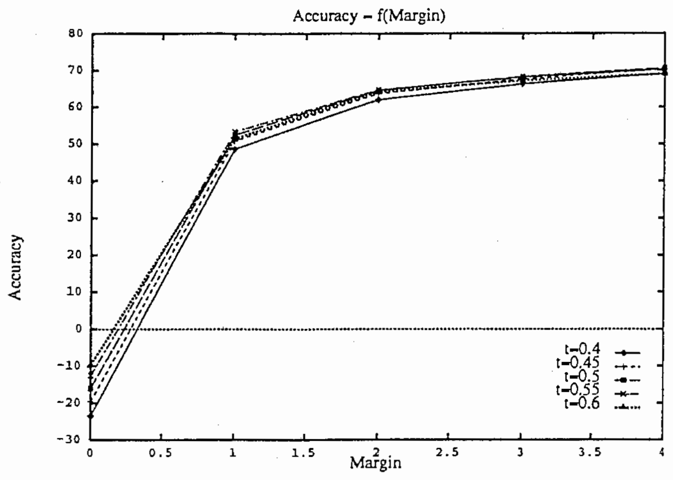


Figure 5: Accuracy values according to the margin for the threshold values from 0.4 to 0.6.

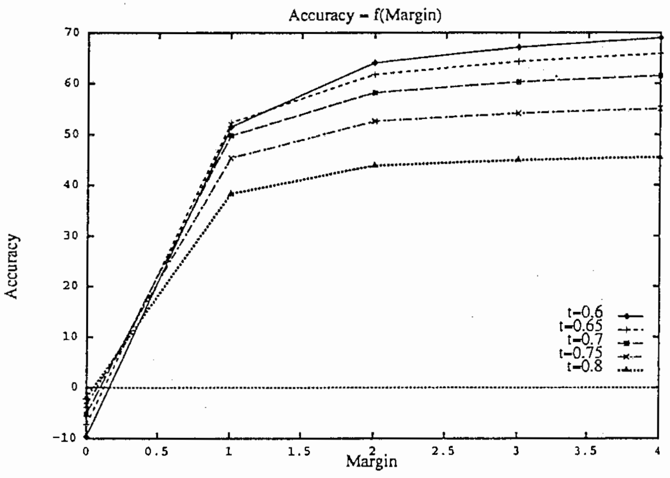


Figure 6: Accuracy values according to the margin for the threshold values from 0.6 to 0.8.

3.2 Second Idea: two thresholds and a margin

3.2.1 Conception

To decrease the number of deletions, the number of points which are considered as estimated boundaries has to be risen. To find new estimated points, a second threshold needs to be used. Now, what is called estimated boundary is each point above the upper threshold and each local maximum between the two thresholds.

In this case, the number of deletions is considerably decreased but the number of insertions is increased.

Explanation of the results obtained for the example I illustrated with Table 7 (upper threshold = 0.7, lower threshold = 0.1):

- The true boundaries are at the frames: 0, 12, 17, 26, 32, 39, 41, 48, 51, 57, 62, 71, 79, 86, 100.
- The estimated boundaries are at the frames: 0, 13, 17, 18, 19, 20, 27, 32, 33, 34, 35, 38, 39, 40, 47, 48, 49, 52, 53, 54, 55, 56, 57, 58, 60, 63, 69, 70, 71, 72, 73, 78, 79, 80,

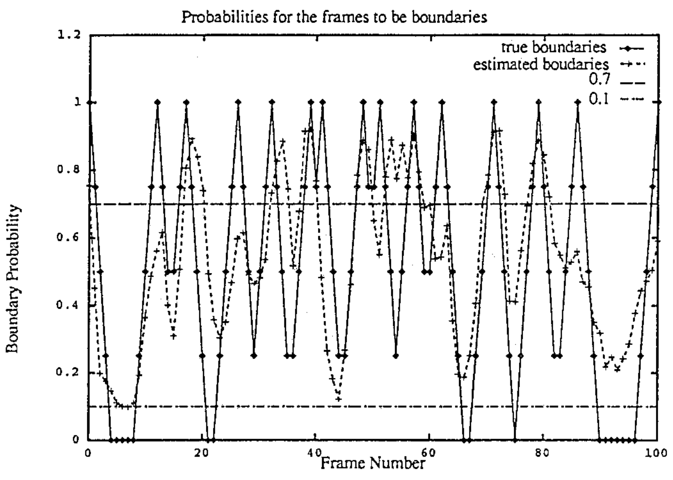


Figure 7: Curves of the true and estimated probabilities to have a boundary in each frame with the two thresholds

81, 86, 92, 100.

- 15 True boundaries	
- for margin = 0	Insertion = 28
	Deletion = 5 (frames 12, 26, 31, 41, 51)
	Accuracy = -120.00
- for margin = 2	Insertion = 23
	Deletion = 0
	Accuracy = -53.33

For margin = 2, the Deletion is lower because more frames can correspond to a true boundary. There are more possibilities to find an estimated boundaries among 5 frames than among 3.

3.2.2 Results

For the example II: Upper threshold = 0.7
Lower threshold = 0.1
Margin = 2

For these 1344 testing data files, the results are:
Number of true boundaries = 53006
Insertion = 81044
Deletion = 2118
Accuracy = -56.89
Percentage of deletion = 4.00

The accuracy is negative because the Insertion is very high. The positive point is the low value of the deletion which is underlined by the percentage of deletions.
With these results, it is possible to compare the impact of the number of cycles.

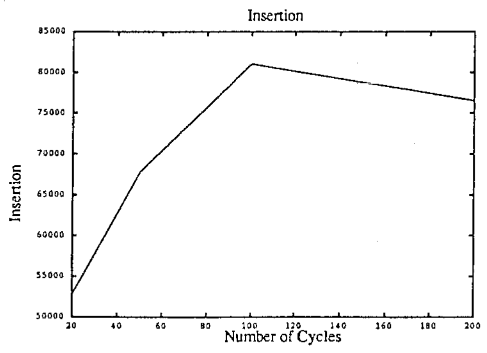
For the following values: Upper threshold = 0.7
Lower threshold = 0.1

Margin = 2

We obtain the results of the Figure 5 according the number of cycles to run the RNN network.

Nb cycles	Insertion	Deletion
20	52752	5724
50	67795	2608
100	81044	2118
200	76521	2278

Table 5: Number of Insertions and Deletions according the number of cycles of the RNN network.



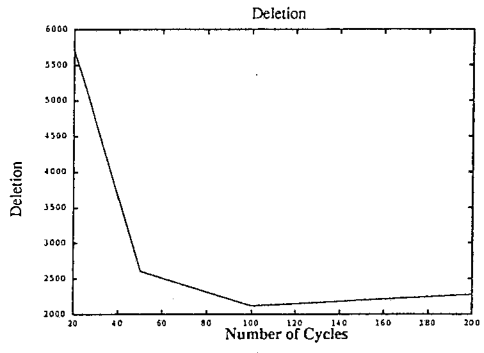


Figure 8: The number of Insertions and Deletions depending on the number of cycles of the RNN network.

These two graphs of Figure 8 lead to the conclusion that when the number of cycles increases to 200 cycles, the number of insertion decreases which is necessary for this work because it is too high. We can see that, for a RNN run on 200 cycles, the number of deletions is higher than for a RNN run on 100 cycles. As it is important to obtain a number of deletions as low as possible, we will continue with the network run on 100 cycles.

As we have seen, the number of insertions is too high. More the insertion is high, more it will be long and difficult to carry out the identification of the phonemes because the system will have to try all the combinations possible.

3.3 Third Idea: two thresholds, a margin and a step

3.3.1 Conception

To reduce the number of insertions without reducing the number of deletions, the system carries on using the three parameters (the two thresholds and the margin) as they are used till now. It is added another one with is called "step" and this one will allow the reduction of the insertion.

With the value step, we consider a ratio "1/step" for all the following points above the upper threshold without counting the local maxima above the upper threshold. Indeed, when for instance the value of the step is 2 and there are 5 following points above the upper threshold, we only consider the first one, the third one and the fifth one as possible (or estimated) landmarks, if one of those points is a local maximum, we just do as if there was no point. If in this example the fourth point is a local maximum, then we only consider the first one, the third one.

Explanation of the results that are obtained for the example I (upper threshold = 0.7, lower threshold = 0.1, margin = 0):

- The true boundaries are at the frames: 0, 12, 17, 26, 32, 39, 41, 48, 51, 57, 62, 71, 79, 86, 100.
- The estimated boundaries are at the frames: 0, 13, 17, 18, 19, 20, 27, 32, 33, 34, 35, 38, 39, 40, 47, 48, 49, 52, 53, 54, 55, 56, 57, 58, 60, 63, 69, 70, 71, 72, 73, 78, 79, 80, 81, 86, 92, 100.

- 15 True boundaries
- for step = 1 Insertion = 28
Deletion = 5
Accuracy = -120.00
- for step = 2 now the estimated boundaries are : 0, 13, 17, 19, 27, 32, 34,38, 39, 47, 48, 52, 54, 56, 57, 60, 63, 69, 71, 73, 78, 79, 81, 86, 92, 100.
Insertion = 16
Deletion = 5
Accuracy = -40.00

For step = 2, the Insertion is lower because less frames are considered as estimated boundaries. The deletion is not reduced, because there are still enough frames in the list of the estimated boundaries and, with margin = 2, each true boundary can find an estimated one among 5 frames. If those 5 frames are above the upper threshold without being local maxima, then those 5 frames are changed in a list of 3 frames. Using the parameters margin and step, we have to be careful not to miss some estimated boundaries. For instance, if margin equals 1, the step must not be higher than 2, otherwise it is possible that some estimated boundaries would miss for the segmentation into phonemes.

3.3.2 Results - Insertion

The Table 6 shows the difference in the values of insertion for step 1 and for step 2 according to the values of the two thresholds, with margin equal to 2. With the step value equal to 2, the insertion is very reduced without spoiling the previous results.

The Figure 9 illustrates the fact that the number of insertions is decreased with a step equal to 2. It is nearly divided by a factor 2 for every values. For instance, with the upper threshold equal to 0.7 and the lower threshold equal to 0.1 with the margin 2, the insertion value was 81044 with step 1 and becomes 49953 with step 2. It is a huge gain in the number of estimated boundaries.

3.3.3 Results - Accuracy

Consequently, the accuracy value is also changed in a positive way (Table 7).

The Table 10 underlines the fact that the values of the accuracy are better when the step value equals 2. Indeed the number of insertions is lower in every cases, the number of deletions is not changed nor the number of true boundaries.

	Lower threshold				
Upper threshold	0.0	0.05	0.1	0.15	0.2
0.5	147841 84637	147841 84637	147564 84360	142635 79431	137360 74161
0.55	130360 74499	130360 74499	130083 74222	125155 69295	119885 64028
0.6	113414 65627	113414 65627	113137 65350	108210 60423	102944 55159
0.65	97073 57596	97073 57596	96796 57319	91869 52392	86605 4712
0.7	81321 50230	81321 50230	81044 49953	76120 45029	70858 39768
0.75	66166 43315	66166 43315	65889 43038	60966 38116	55707 32858
0.8	52069 37704	52069 37704	51792 37427	46869 32505	41613 27249

Table 6: Insertion values according to the two thresholds with step = 1 (first line) and step = 2 (second line).

3.3.4 Results - Reducing rate

To find the best set of parameters, several criteria become important. The first one was not to have a too big number of deletions because those points could not be used as possible boundaries anymore. The second one was to reduce the number of inserted points among the possible boundaries in order not to have too many points to check during the identification of the phonemes. These reasons lead to the introduction of a new rate : Reducing rate.

It is defined:

$$Reducing\ Rate\ (\%) = 100 * \frac{N + I - D}{Nb\_frame}$$

with:
N total number of real boundaries
I number of insertions
D number of deletions
Nb_frame number of frames in the data files

It corresponds to the percentage of frames that will be checked for the identification of the phonemes.

The Table 8 indicates the values of the reducing rate for the step equal to 2 and the margin equal to 2, and it shows the gain of points to be examined to find the best boundaries for the phonemes.

The reducing rate (Figure 11) is much better for a step equal to 2 because they are less frames that will have to be checked to find the boundaries of the phonemes. The best rate is 18.67, it is obtained with 0.2 as the lower threshold and 0.8 as the upper threshold. Indeed the insertion is the less high for these values, but the deletion is a bit too high. May be the best set of parameters is around 0.7 for the upper threshold, 0.1 for the lower threshold, 2 for the margin, 2 for the step.

With a higher value for the parameter "step", the deletion increases because some true boundaries do not have their equivalent among the estimated ones anymore.

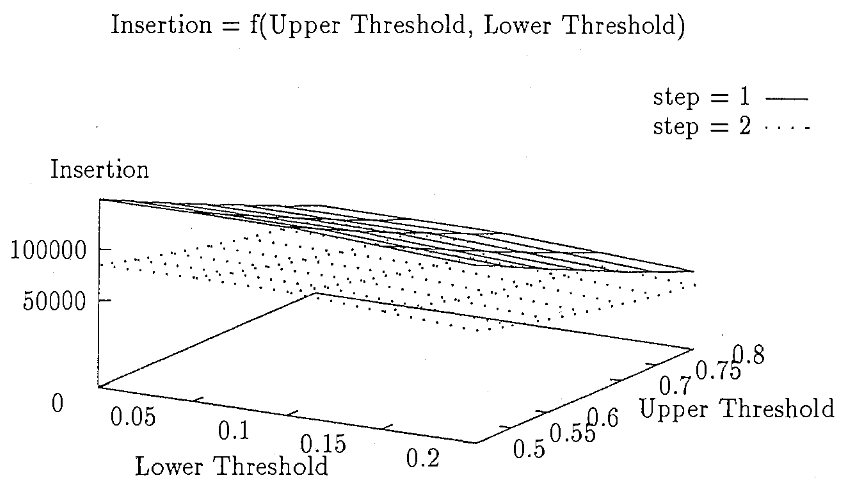


Figure 9: Insertion values according to the two thresholds with step = 1 and step = 2.

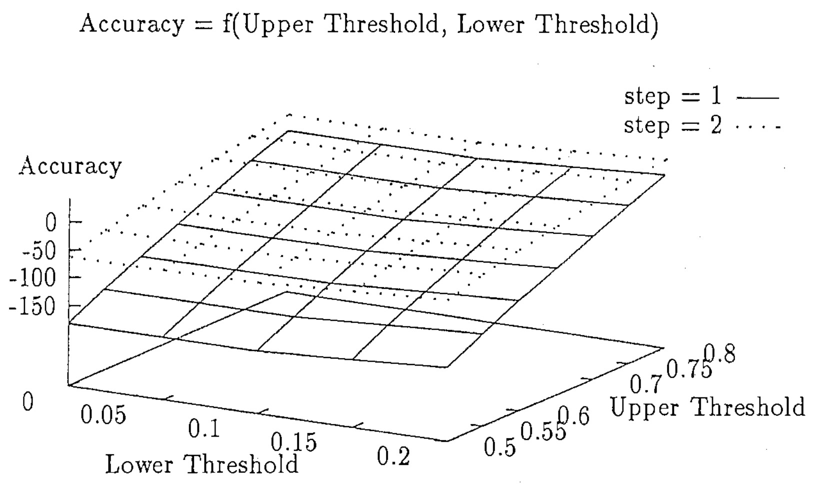


Figure 10: Accuracy values according to the two thresholds with step = 1 and step = 2.

3.3.5 Results - Example

Here are the results for the example II with step=2 (to be compared with the results of the previous sections):

For the 1344 testing files, the results are:
Number of true boundaries = 53006
Insertion = 49953
Deletion = 2118 (it has not changed since the latter results)
Accuracy = 1.76
Percentage of deletion = 4.00

	Lower threshold				
Upper threshold	0.0	0.05	0.1	0.15	0.2
0.5	-180.75	-180.75	-180.23	-171.05	-161.50
	-61.51	-61.51	-60.99	-51.81	-42.27
0.55	-148.21	-148.21	-147.69	-138.52	-128.98
	-42.82	-42.82	-42.30	-33.13	-23.60
0.6	-116.73	-116.73	-116.21	-107.04	-97.52
	-26.57	-26.57	-26.05	-16.88	-7.37
0.65	-86.48	-86.48	-85.96	-76.79	-67.28
	-12.00	-12.00	-11.48	-2.31	7.20
0.7	-57.41	-57.41	-56.89	-47.73	-38.23
	1.25	1.25	1.76	10.92	20.42
0.75	-29.78	-29.78	-29.26	-20.11	-10.62
	13.33	13.33	13.85	23.00	32.49
0.8	-4.43	-4.42	3.91	5.25	14.73
	22.67	22.67	23.19	32.35	41.83

Table 7: Accuracy values according to the two thresholds with step = 1 (first line) and step = 2 (second line).

Now, we have elaborated a good way to find the frames which may be boundaries of phonemes in our set of data (sentences). Nevertheless, these estimated boundaries are only a stage in the process. Indeed, it is important to know how many good phonemes could be found with the set of estimated phonemes created.

	Lower threshold				
Upper threshold	0.0	0.05	0.1	0.15	0.2
0.5	48.65	48.65	48.58	47.36	46.03
	33.48	33.48	33.42	32.20	30.87
0.55	44.33	44.33	44.27	43.05	41.72
	30.93	30.93	30.86	29.65	28.31
0.6	40.15	40.15	40.08	38.86	37.53
	28.67	28.67	28.61	27.39	26.06
0.65	36.10	36.10	36.03	34.81	33.48
	26.61	26.61	26.54	25.32	23.99
0.7	32.18	32.18	32.11	30.89	29.56
	24.68	24.68	24.61	23.40	22.06
0.75	28.37	28.37	28.30	27.08	25.75
	22.84	22.84	22.77	21.55	20.22
0.8	24.77	24.77	24.71	23.49	22.16
	21.29	21.29	21.22	20.01	18.67

Table 8: Reducing rate values according to the two thresholds with step = 1 (first line) and step = 2 (second line).

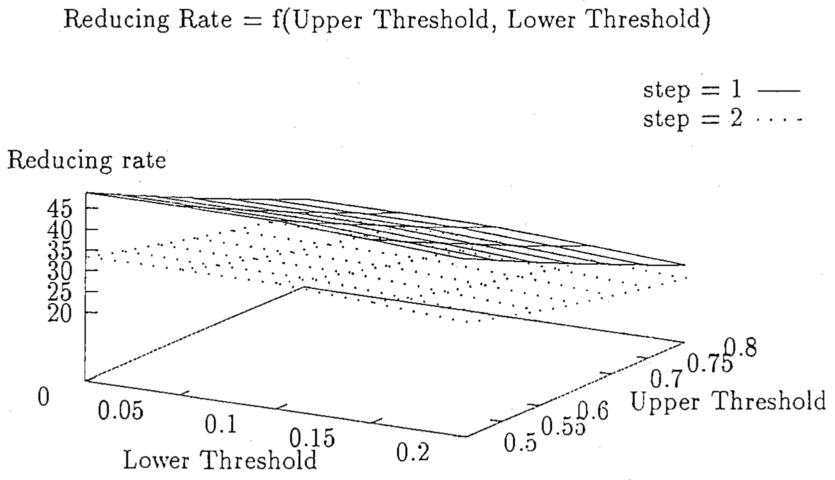


Figure 11: Reducing rate values according to the two thresholds with step = 1 and step = 2.

4 How to find the good phonemes with the set of possible boundaries

As it will be too long to consider all the possible combinations of the boundaries to find the phonemes, it is necessary to choose some of them as "compulsory boundaries". Indeed, if there was no difference among the possible boundaries, the first phoneme would begin at the first phoneme and it would end at the second, third, fourth... or any frame else; then the second phoneme would begin at this frame end would end at any other one,... The system would have to find all the possibilities of combinations.

4.1 First Idea: two kinds of estimated boundaries

4.1.1 Conception

To reduce the number of possible combinations to find the phonemes, we have created two series of estimated boundaries:

- Main boundaries: these boundaries are the local maxima above the upper threshold.
- Second boundaries: they are the estimated boundaries which are not "main boundaries".

As they are local maxima above the upper threshold, the main boundaries seem to be more surely boundaries of landmarks. That is the reason why we have chosen these frames to be, in a way, some barriers in the search of the phonemes. Indeed, one phoneme would never be demarcated by two boundaries between which there is a main boundary.

Here is an example to illustrate this rule:
Considering that there are main boundaries at the frames 46, 65, 90, and second boundaries at the frames 49, 56, 70, 84.

- From the main boundary 46: The phonemes whose boundaries are 46-49, 46-56, 46-65 may exist. However it is impossible to consider the following phonemes: 46-70, 46-84 and 46-90 because there is a main boundary, frame 46, between the frames demarcating the phonemes.
- From the second boundary 49: The phonemes whose boundaries are 49-56 and 49-65 may exist. However it is impossible to consider the following phonemes: 49-70, 49-84 and 49-90 because there is a main boundary, frame 46, between the frames demarcating the phonemes.

Explanation of the results that are obtained for the example I (upper threshold = 0.7, lower threshold = 0.1, step = 2, margin = 2):

- The true boundaries are at the frames: 0, 12, 17, 26, 32, 39, 41, 48, 51, 57, 62, 71, 79, 86, 100.
- The estimated boundaries are at the frames: 0, 13, 17, 18, 20, 27, 32, 34, 38, 39, 47, 48, 52, 53, 55, 57, 60, 63, 69, 71, 72, 78, 79, 81, 86, 92, 100.

- 15 True boundaries
- 10 main boundaries (frames: 0, 18, 34, 39, 48, 53, 55, 57, 72, 79)
- 17 second boundaries (frames : 13, 17, 20, 27, 32, 38, 47, 52, 60, 63, 69, 71, 78, 81, 86, 92, 100)

After each main boundary, the meter concerning the "step" is put to 0, that is why a second boundary will never follow a main boundary immediately. This system splits the problem to find the boundaries of the phonemes from a very large scale to several smaller sets of frames. It is easier to find the phonemes from the frame 0 to the frame 18, from 18 to 34... than between the two ends of each file.

4.1.2 Results

Now we can look at the results concerning phonemes for the example II.

Results: 51662 True phonemes (i.e. in the original data)
42852 Good phonemes (i.e. estimated phonemes which correspond to original ones)
7338 Created phonemes (i.e. which do not exist in the original data)
50190 Estimated phonemes (good phonemes + created phonemes)
Percentage of good phonemes (among the true ones): 82.95

The percentage of good phonemes corresponds to:

$$Percentage\ of\ good\ phonemes\ (\%) = 100 * \frac{Good\ phonemes}{True\ phonemes}$$

The percentage of the good phonemes in the example is not very high compared with the value that could have been expected. Indeed, the percentage of deletion for the boundaries was equal to 4.00, so there was 100 - 4.00 = 96.00 per cent of the true boundaries which could be found among the estimated ones.

These results of Example II can be explained as follows:

1. If two main boundaries correspond to true ones and if between them another main boundary does not correspond to any true one, then it is inserted between two true boundaries. By counting the number of inserted boundaries there is only one frame; but two phonemes are created whereas, in the data file, there was only one phoneme. One main boundary "inserted" may lead to the creation of two phonemes.

2. On the contrary, if there is no estimated boundary corresponding to a true one, the number of false boundaries will be one but the number of false phonemes will be 2. Indeed, there would be only one phoneme which does not correspond to any of the two true phonemes.

We think these are the main reasons to explain the low percentage of good phonemes.

The next stage is to reduce the number of created phonemes and at the same time to increase the number of good phonemes.

4.2 Second Idea: too close main boundaries should be considered as second ones

4.2.1 Conception

The idea consists in:
When there are two main boundaries which are not more far than a given distance (new parameter) and if all points between them are values above the upper threshold then, they

are considered as second landmark.

The parameter "distance" corresponds to the number of frames that are needed to space out two main boundaries. The distance parameter equal to 1 corresponds to the previous tries because the main boundaries are local maxima, so two main boundaries can not follow each other except if they have the same values.

It is true that the number of estimated boundaries is increased just a little by the fact that we have considered as following points above the upper threshold the points which are local maxima above the upper threshold (the point just after a local maximum is not removed automatically anymore). The lists of main and second boundaries have changed a little. Some frames which were before considered as main boundaries have turned to second boundaries because two frames in the main list were too close.

The performances are improved because sometimes two local maxima, only separated by one frame, correspond to one true boundary and such a configuration creates too many and false phonemes in the segmentation into phonemes.

Explanation of the results that are obtained for the example I (upper threshold = 0.7, lower threshold = 0.1, step = 2, margin = 2):

- The true boundaries are at the frames: 0, 12, 17, 26, 32, 39, 41, 48, 51, 57, 62, 71, 79, 86, 100.
- The estimated boundaries above the upper threshold are at the frames: 0, 17, 18, 20, 32, 34, 38, 39, 47, 48, 52, 53, 55, 57, 69, 71, 72, 78, 79, 81.
- The estimated boundaries between the two thresholds are at the frames: 13, 27, 60, 63, 86, 92, 100.

if distance = 1: - 15 True boundaries, 14 True phonemes
- 10 main boundaries (frames: 0, 18, 34, 39, 48, **53**, **55**, **57**, 72, 79)
- 17 second boundaries (frames : 13, 17, 19, 27, 32, 38, 40, 47, 49, 52, 54, 56, 58, 60, 63, 69, 71, 73, 78, 80, 86, 92, 100)
- 14 good phonemes, 1created phoneme.

if distance = 2: - 15 True boundaries
- 10 main boundaries (frames: 0, 18, 34, 39, 48, 72, 79)
- 17 second boundaries (frames : 13, 17, 19, 27, 32, 38, 40, 47, 49, 52, **53**, **55**, **57**, 58, 60, 63, 69, 71, 73, 78, 80, 86, 92, 100)
- 14 good phonemes, 0 created phoneme.

4.2.2 Results - Reducing rate

With Figure 12 we can see that, for a constant value of the lower threshold, the reducing rate decreases from the lowest values of the upper threshold to the highest one. There is a break in the shapes of the curves between the values 0.7 and 0.75. The difference between the reducing rates for a high value of the upper threshold and for a lower one is quite big, nearly 14 points from 34% to 20%. On the contrary, for a given value of the upper threshold, the difference between the reducing rates for a low value of the lower threshold and for a higher one is quite slight, about 2 points from 34% to 32% for instance. Consequently, it seems that it is not very important to decide which value between 0 and 2 is the good one for the lower threshold whereas it is very important to choose carefully the value of the upper threshold between 0.5 and 0.8.

	Lower threshold				
Upper threshold	0.0	0.05	0.1	0.15	0.2
0.5	88.83 34.99	88.83 34.99	88.82 34.92	88.58 33.71	87.81 32.37
0.55	88.38 33.16	88.38 33.16	88.37 33.09	88.12 31.87	87.34 30.54
0.6	87.89 31.32	87.89 31.32	87.89 31.26	87.65 30.04	86.88 28.70
0.65	87.21 29.44	87.21 29.44	87.21 29.38	86.98 28.16	86.22 26.82
0.7	86.22 27.35	86.22 27.35	86.21 27.28	85.99 26.06	85.24 24.73
0.75	84.57 25.04	84.57 25.04	84.57 24.97	84.35 23.76	83.60 22.42
0.8	82.06 22.71	82.06 22.71	82.06 22.64	81.87 21.42	81.17 20.09

Table 9: Percentage of good phonemes (first line) and Reducing rate (second line) according to the two thresholds with distance = 2.

4.2.3 Percentage of good phonemes

The same observation (Figure 13) can be done for these curves. The choice of the lower threshold does not need to be very accurate between 0 and 0.2. In Table 9 , the highest value of success in the segmentation is 88.83% and the lowest one is 81.17%. The curves have also a strong break around the upper threshold 0.7.
The Figure 13 underlines the fact that for a high value of the upper threshold (above 0.7) the percentage of the good phonemes is not so different between distance equal to 1 and distance equal to 2. This is due to the low number of frames above the upper threshold.

So we think that 0.7 may be the right value for the upper threshold because the reducing rate is not too high and the results are not too bad (about 85% of the original phonemes are demarcated correctly).

4.2.4 Results - Distance

	Distance			
Upper threshold	1	2	3	4
0.5	87.63	88.83	90.05	90.66
0.55	87.54	88.38	89.26	89.57
0.6	87.33	87.89	88.47	88.68
0.65	86.85	87.21	87.53	87.63
0.7	86.07	86.22	86.37	86.41
0.75	84.54	84.57	84.63	84.59
0.8	82.06	82.06	82.07	82.05

Table 10: Percentage of good phonemes according to the values of the distance and the upper threshold (lower threshold = 0.1).

In Figure 14, when the value of the upper threshold is lower than 0.7, the performance of the segmentation is improved with a high value of distance. Indeed, more estimated boundaries are considered as second ones, so there are less mistakes (less insertions because less frames

Reducing rate = f(Upper Threshold, Lower Threshold)

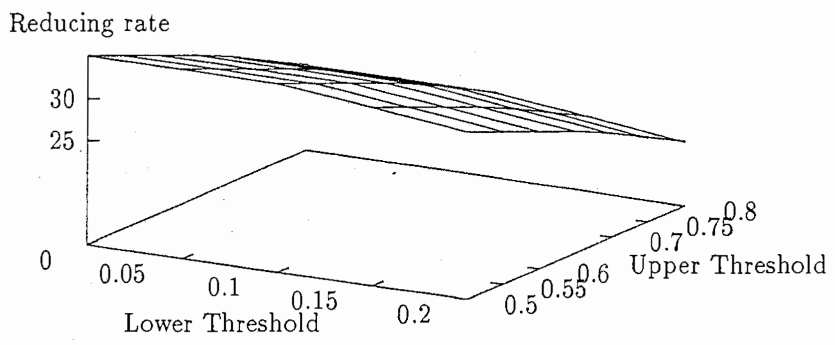


Figure 12: The Reducing rate with distance equal to 2 according to the two thresholds.

are forced to be boundaries).

The break around the value 0.7 can be explained by the fact that if the upper threshold is too high a lot of frames are not in any lists of estimated boundaries whereas they correspond to a true one (we have seen before that for such values the number of deleted boundaries was too high).

	Reducing rate			
Upper threshold	1	2	3	4
0.5	35.51	34.92	33.93	33.06
0.55	33.57	33.09	32.30	31.58
0.6	31.63	31.26	30.66	30.15
0.65	29.67	29.38	28.96	28.61
0.7	27.48	27.28	27.02	26.84
0.75	25.08	24.97	24.82	24.75
0.8	22.68	22.64	22.58	22.56

Table 11: Reducing rate according to the values of the distance and the upper threshold (lower threshold = 0.1).

In Figure 15, the break around 0.7 is explained by the same reasons as just before. As there are much few estimated boundaries, the reducing rate is much better (lower).

4.2.5 Results - Example

The example II has new results which are improved with the new parameter distance equal to 2.

Results: 51662 True phonemes (i.e. in the original data)
44540 Good phonemes (i.e. estimated phonemes which correspond to original ones)
5501 Created phonemes (i.e. which do not exist in the original data)
50041 Estimated phonemes (good phonemes + created phonemes)
Percentage of good phonemes (among the true ones): 86.21

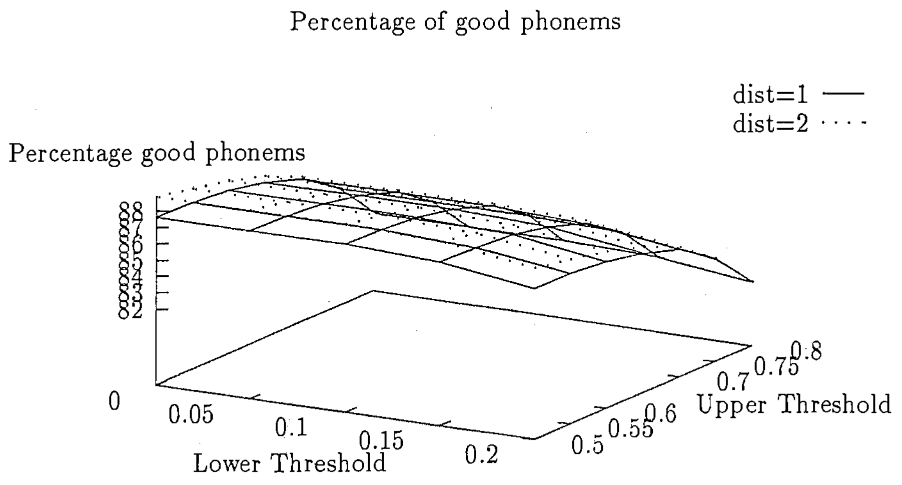


Figure 13: The percentage of good phonemes with distance equal to 2 according to the two thresholds.

Reducing rate 27.28

The percentage of good phonemes is a bit better (it was 86.07%) and the reducing rate is not increased (it was 27.48%). Indeed, there is approximately the same number of estimated boundaries. We think that what is also important is to see that the number of the phonemes which are created and which do not correspond to true ones has been significantly reduced. Without this notion of distance, it was equal to 6687 so it has been reduced of about 1000 phonemes.

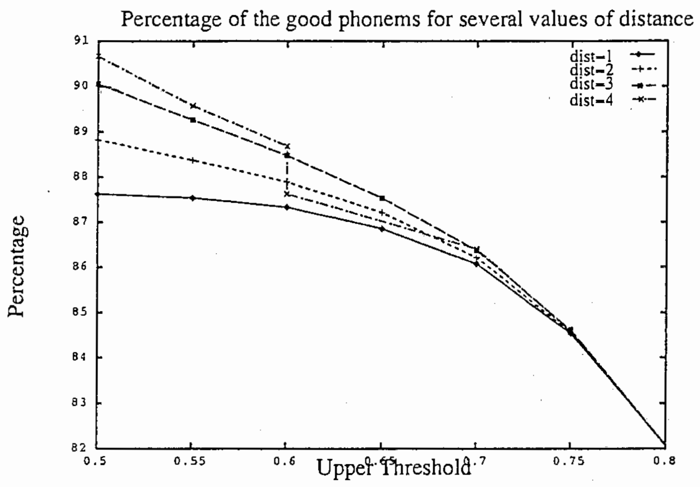


Figure 14: Percentage of the good phonemes for different values of the parameter distance (lower threshold = 0.1).

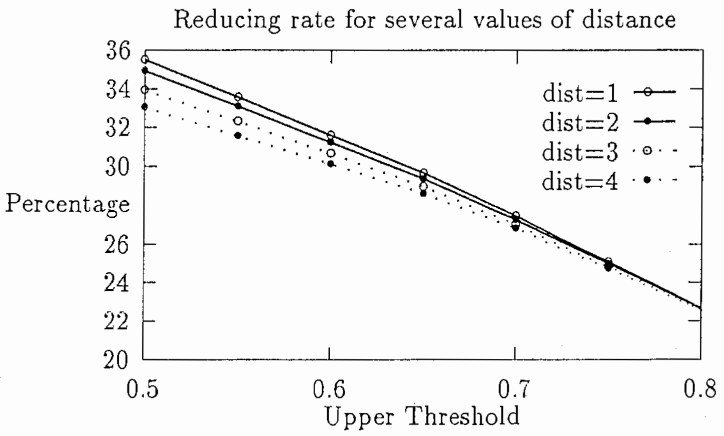


Figure 15: Reducing rate for different values of the parameter distance (lower threshold = 0.1).

Part II

Recognizer

5 Speech Recognition Based on Polynomial Segment Models

We have obtained a list of the possible boundaries of the phonemes. Now, we have to find the "best" path to go from the first frame to the last one for each file. In order to evaluate this path, that is to say to find the most likely succession of phonemes, we use the Maximum Likelihood.

5.1 Structure of the network

The network built for the detection of the phonemes has to respect strict rules:

- In the list of the possible boundaries, there are main boundaries and second boundaries. The main boundaries are crossed through by all paths.
- If two second boundaries are too close from each other (less than 1 frame between them), there is no link between them (cf. Figure 16 between node 3 and node 4). This is due to the latdec program that we will use (see part 7.5.2).
- The dictionary contains 61 phonemes. Between two nodes (i.e. two boundaries), we have to calculate the Likelihood for each phoneme in order to choose which one is the best.
- The best path will be the one with maximizes the likelihood.
- From the last node and the values of likelihood obtained, it is possible to find the best path by coming back through the nodes.

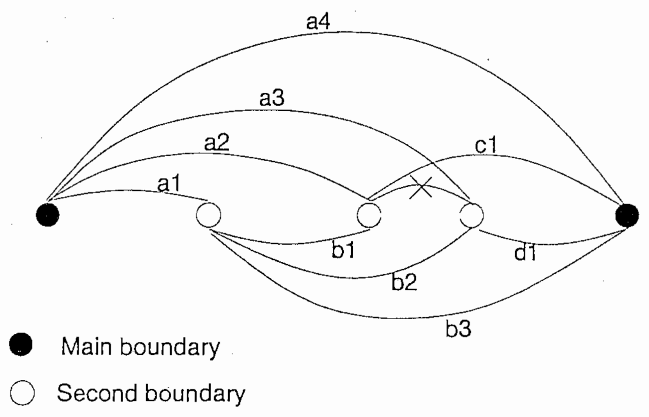


Figure 16: Representation of the network of the nodes with main and second boundaries

To decrease the computation time, it was not necessary to keep the second boundaries which are too close to a main one in the list of the nodes. Indeed, there would have been no link between each of them and the preceding or following main boundary. As we have seen before all paths cross over all the main boundaries, so the paths crossing such second nodes cannot be possible.

5.2 Polynomial Segment Model

To find the best path, that is to say the path closest to the original one, we have to calculate the maximum likelihood for each link [2].

First, we can model each feature dimension of a speech segment as:

$$c(n) = \mu(n) + e(n) \quad for\ n = 1, ..., N \tag{1}$$

where c(n) are the observed cepstral features in segment of length N, μ(n) are the mean features as functions of frame number and represent the dynamics of the features in the segment, and e(n) are the residual error terms.

If we use a matrix notation (example with a quadratic function of time), we will have:

$$\begin{aligned}\mu(n) &= b_1 + b_2 n + b_3 n^2 \quad for\ n = 1, ..., N \\ &= \underline{z}'.\underline{b}\end{aligned} \tag{2}$$

where $\underline{z}$' = [1 n n^2] and $\underline{b}$' = [b$_1$ b$_2$ b$_3$].

If the speech segment has a duration of N frames which are represented by D dimensional feature vectors, we can use a matrix notation to express the segment:

$$C = \begin{bmatrix} c_{1,1} & \cdots & c_{1,D} \\ c_{2,1} & \cdots & c_{2,D} \\ \vdots & & \vdots \\ c_{N,1} & \cdots & c_{N,D} \end{bmatrix} = [\ \underline{C}_1 \ \cdots \ \underline{C}_D\ ] \tag{3}$$

So Equation 1 can be modeled as follows:

$$C = ZB + E \tag{4}$$

where Z is a N $\times$ R design matrix, B is a R $\times$ D trajectory parameter matrix, and E is a residual error matrix. R is the number of parameters in the trajectory model (R = 1 linear, R = 2 linear, R = 3 quadratic).

The matrix equation 4 can be written for each feature dimension i:

$$\underline{C}_i = Z\underline{B}_i + \underline{E}_i \quad for\ i = 1, ..., D \tag{5}$$

where $\underline{B}_i$ is a R $\times$ 1 trajectory vector for feature i, $\underline{E}_i$ is a residual error vector for feature i, and $Z\underline{b}_i$ is the trajectory component for feature i analogous to μ(n), n = 1,...,N in Equation 2.

As an example, for a quadratic trajectory model and a segment with N frames:

$$\begin{bmatrix} c_{1,i} \\ c_{2,i} \\ \vdots \\ c_{N,i} \end{bmatrix} = \begin{bmatrix} z_{1,1} & z_{1,2} & z_{1,3} \\ z_{2,1} & z_{2,2} & z_{2,3} \\ \vdots & \vdots & \vdots \\ z_{N,1} & z_{N,2} & z_{N,3} \end{bmatrix} \begin{bmatrix} b_{1,i} \\ b_{2,i} \\ b_{3,i} \end{bmatrix} + \begin{bmatrix} e_{1,i} \\ e_{2,i} \\ \vdots \\ e_{N,i} \end{bmatrix} \quad for\ i = 1, ..., D \tag{6}$$

or

$$c_{n,i} = z_{n,1}b_{1,i} + z_{n,2}b_{2,i} + z_{n,3}b_{3,i} + e_{n,i} \tag{7}$$

$$for\ n = 1, ..., N\ and\ i = 1, ..., D \tag{8}$$

Given the segment model in the Equation 4, assuming that the errors are independent and identically distributed (normal with covariance Σ), the Maximum Likelihood estimate of the trajectory parameter matrix, $\widehat{B}_k$, is given by the linear least squares estimate:

$$\widehat{B}_k = [Z_k' Z_k]^{-1} Z_k' C_k \tag{9}$$

for a segment k with data matrix, C_k, and design matrix, Z_k.

With $\widehat{B}_k$ estimated, the residual covariance matrix for the segment, $\widehat{\Sigma}_k$, is given by:

$$\widehat{\Sigma}_k = \frac{\widehat{E}_k' \widehat{E}_k}{N_k} = \frac{(C_k - Z_k \widehat{B}_k)'(C_k - Z_k \widehat{B}_k)}{N_k} \tag{10}$$

where N_k is the number of frames in segment k.

After parameter estimation, each segment is replaced by its set of statistics: ($\widehat{B}_k$, $\widehat{\Sigma}_k$, N_k). This collection of parameters is sufficient to calculate the likelihood of the frames compromising the original segment (to know more abut the Estimation Model, see [2]).

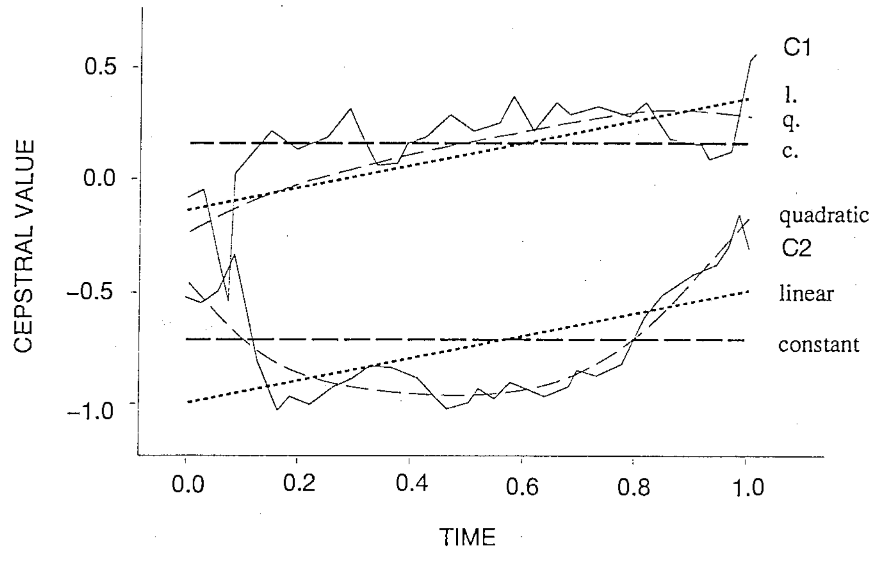


Figure 17: Temporal variation of Cepstral features C1 and C2 of a phoneme segment (curved line), and the fit of constant, linear, and quadratic trajectory models (dotted lines) to the features.

6 Latdec program

6.1 Conception

The Latdec program, made by M.Bacchiani and M. Ostendorf [5] [1] , gives the best (maximum likelihood) path in the input file in order to demarcate the phonemes and to identify them.

The inputs are a mfcc_file and a lattice file. The mfcc file contains the waveform of the sentence and the lattice file lists all the nodes of the mfcc_file and the links that exist between them (cf "structure of the network" page 25).The lattice file has the following structure:

> NODES=*number of nodes in this sentence*
> LINKS=*number of links between second, main and second, and main*
>
> I=1 *number of this node*
> time=*number of this frame expressed in seconds*
> I=
> time=
> ⋮
>
> J=1 *number of this link*
> START=*number of the node in which this link starts*
> END=*number of the node in which this link ends*
> WORDID=*identification number of a phoneme of the phoneme alphabet*
>
> J=
> START=
> ⋮

It creates a file with the numbers of the frames in which each phoneme starts and ends and the phoneme identified between those boundaries. With such files, it is possible to obtain the results in terms of accuracy and percentages of good sentence and good phones.

6.2 First results

First, we have tested these programs with the testing data trained on 100 cycles with the RNN whose structure was made with 20 forward and backward states.

The results for 100 cycles are not as good as we hope because the latdec program does not only compare the numbers of the possible boundaries, but it evaluates the maximum likelihood by comparing the waveform of the testing data between two boundaries with the 61 phonemes of a dictionary. It has to find which phoneme fits the best the part of the waveform demarcated by the boundaries. So it is not only a problem of boundaries, but also a problem of the distortion of the original waveform.

In order to obtain best results, we have tried with the testing data trained on the RNN with 200 cycles and the same structure as the previous one. Table 12 shows these results for the two networks with only one threshold and main boundaries.

Table 12 can let us hope that the results with more iterations in the RNN may give slightly better results.

	Threshold								
Nb cycles	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100 cycles	8.10	21.63	28.72	34.17	36.22	35.75	32.90	23.76	4.75
200 cycles	12.96	25.63	31.59	35.98	36.84	36.22	32.54	24.71	5.90

Table 12: Accuracy of the testing data of the directory dr1 for different kinds of RNN and only one threshold to demarcate the main boundaries of the phonemes.

There are quite a lot of parameters that may improve the performance of the system and each of them will be examined in the next paragraphs. This is a list of these parameters that should be taken into account.

- Value for the parameter -D in the latdec program.
- Number of iterations for the RNN.
- Change in the probabilities for frames to be boundaries as outputs of the RNN.
- Structure of the RNN with hidden layers.

All results need to be compared with the results obtained with the HMM system with 3 states and 1 mixture for each of them. This should be calculated for the same testing data. The accuracy obtained with such a system for the directory dr1 of the testing data is equal to 40.52%, which is much better than the present results.

7 Improvements of the present system

In this section, the results are obtained with various kinds of RNN. The structure of these RNNs is made of 10 backward, 10 forward states and 10 hidden states when nothing is specified.

- When the results are obtained for 100 or 200 iterations, the RNNs were trained with only a sample of 200 sentences (whereas the full training set contains 3696 sentences).
- When the results are obtained for 500 or 1000 iterations, the RNNs were trained with the full training data.

7.1 Duration probability weighting

The parameter -D corresponds to the duration probability scaling. Indeed, the parameter is a coefficient that multiplies the time part of the likelihood of a segment.

For each segment (and for the 61 phones) the likelihood corresponds to:

$$log\ Likelihood = \sum_{t=1}^{N_1} \log(P_A^{aa}(t)) + W * log(P_D^{aa}(N_1)) \tag{11}$$

with: aa One of the 61 phonemes
N_1 Length of the segment
P_A Output Probability
P_D Duration Probability
W Weight

The second part (duration factor) of Equation 11 is added in order to adjust dynamically the length of the phoneme to the likelihood. Indeed the length of the phoneme may be an element to identify it.

If D = 0 then Weight = 0, the duration of the phoneme does not take part of the estimation of the likelihood.
If D = 1 then Weight = 1, the likelihood only consider the logarithm of the duration of the phoneme; the longer a phoneme is, the less important the duration factor is.
If D = -1 then Weight = N_1, this factor adapts better the duration factor because it considers the length of the phoneme of the dictionary.

The different values of this parameter were tried (Table 13) on data which were obtained with a RNN, trained with a new kind of output data (values 1-0.5-0, see paragraph 7.3). We can see that the best value for this parameter seems to be -1 and, in this case, the maximum accuracy is reached for the threshold value 0.3 and is equal to 37.61%.

	Threshold							
Parameter D	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
1	21.86	34.71	36.75	35.98	33.55	28.77	22.01	7.18
0	21.24	34.38	36.43	35.33	32.93	28.21	21.51	7.12
-1	22.60	35.98	37.61	37.17	34.53	29.49	22.49	7.21

Table 13: Accuracy of the testing data of the directory dr1 for three values given to parameter D (RNN trained on 100 iterations).

As the results are much better with parameter D set to -1, in the remaining part of this work, we will keep this value.

To improve the results of the Polynomial based recognition, we will try to improve the structure of the RNN by increasing the number of forward and backward states and adding hidden states. We will also try to change the output of the RNN in order to bring out the frames where there is a boundary.

7.2 Number of iterations

The problem is to know whether the increase of the number of iterations to train the Recurrent Neural Network influences the improvement of our work or not.
The test is done with data whose values should be 1-0.5-0 (outputs of the RNN network for the training, see paragraph 7.3) and with two numbers of iterations: 100 and 500.

	Threshold							
Nb. of Iterations	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
100	22.60	35.98	37.61	37.17	34.53	29.49	22.49	7.21
500	23.41	37.17	39.57	38.89	36.22	31.80	23.88	10.68

Table 14: Accuracy of the testing data of the directory dr1 for two numbers of iterations to train the RNN.

Table 14 underlines the fact that the increase of the number of iterations to train the Recurrent Neural Network improves the results regarding the accuracy. That is the reason why we will train a Recurrent Neural Network on 1000 iterations when we will have decided which parameters are needed. Indeed, such RNN is very long to train and it is not necessary to do it with different parameters if we know that they are not so good.

7.3 Output Data

At the beginning of this work, the structure chosen for the outputs of the RNN was the values: 1, 0.75, 0.5, 0.25 and 0. The value "1" characterizes a boundary and the following frames have the other values in decreasing. It is possible to use other kinds of outputs as these sets: 1, 0.5 and 0 or 1 and 0. Figure 7.3 illustrates those different ways to consider the outputs of the RNN.

With these three sets of outputs of the RNN, there are three different RNNs and their performance is not exactly the same.
Table 15 shows that the set 1, 0.5 and 0 gives the best performance concerning the accuracy. This can be explain by the fact that with the values 1 and 0, the "jump" from the value 0 (i.e. no boundary) to the value 1 (i.e. boundary) is too fast, it does not last enough time so the local maxima for the testing data are not high enough. Concerning the original set we used, the way from 0 to 1 is too long and the boundary probability does not always has the value 0 between two boundaries, consequently the difference between boundary and no boundary is not large enough. This explains also that the threshold value giving the best results for each data set is changed.

As the best performance (accuracy: 39.57%) is reached with the data set 1, 0.5 and 0, the next experiment will be done with it.

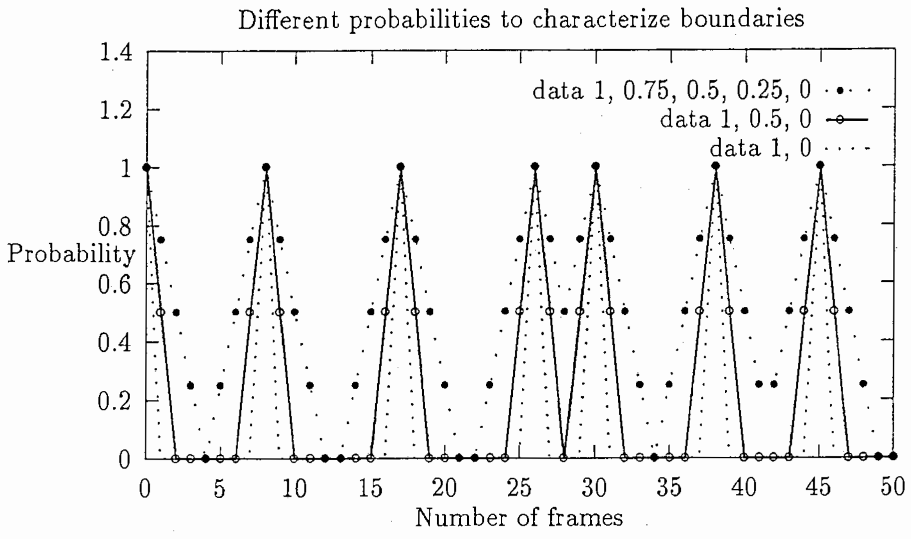


Figure 18: Different ways to give values to the output of the RNN regarding boundaries. There are boundaries at frames 0, 8, 17, 26, 30, 38, 45

	Threshold							
Probabilities	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
1 0.75 0.5 0.25 0	23.05	31.33	35.21	38.03	38.12	36.93	33.70	26.28
1 0.5 0	23.41	37.17	39.57	38.89	36.22	31.80	23.88	10.68
1 0	34.92	37.97	35.00	29.61	22.84	14.83	6.56	2.94

Table 15: Accuracy of the testing data of the directory dr1 for different kinds of outputs of the RNN (trained with 500 iterations with 10 hidden layers).

7.4 RNN structure

7.4.1 Forward and backward states

It is difficult to compare all the results obtain by a lot of different RNNs. We have decided to find the best number of forward and backward states first, and after we will try to find the best number of hidden states. These networks were trained with the small set of data to gain time (200 sentences).

Table 16 shows the values of Minimum Square Error (MSE) for different kinds of RNNs by changing the number of forward, backward and hidden states. To train a network with a lot of a states is longer than to train one with a few states. More, in this case, for a given number of hidden states, the networks with 30 or 40 forward and backward states do not converge very fast, they need more iterations to have a MSE-value lower than the MSE-value of the network which has only 10 forward and backward states. This can be pointed out for the training data as well as the testing data.
This is the reason why we will continue this work with a RNN made of 10 forward and 10 backward states.

If the number of forward and backward states does not change a lot the results, the number of hidden states, by augmenting, makes the MSE-value become lower. The difference between MSE-values for 10 hidden states and MSE-values for 40 hidden states can be quite high, about 0.01 for the testing data of the RNN with 10 forward and backward states

	Number of Hidden States			
Number of Forward and Backward States	0	10	20	30
10-10	5.86e-02		5.10e-02	4.80e-02
	6.81e-02		6.50e-02	6.26e-02
20-20	6.37e-02		4.91e-02	4.68e-02
	7.77e-02		7.11e-02	6.67e-02
30-30	6.80e-02		4.70e-02	4.44e-02
	7.88e-02		7.10e-02	6.82e-02
40-40	7.56e-02		4.98e-02	5.67e-02
	8.62e-02		6.81e-02	7.03e-02

Table 16: MSE-values for the training (first line) and testing (second line) data obtained with 500 iterations.

(Table 16).

7.4.2 Hidden states

We have chosen to use a Recurrent Neural Network made of 10 forward and 10 backward states and we know that the number of hidden states may have an important influence on the results. That is why we have tested the system for different number of hidden states (0, 10, 20, 30). Contrary to the latest tests, we use the all training data and the RNNs were trained on 500 or 1000 iterations.

Tables 17 and 18 show that the performance of the phoneme recognizer becomes better with a high number of hidden states. How ever the differences between the performance depending on those RNNs are less significant as the number of hidden states becomes high. Indeed, the number of hidden states has to be increased significantly to obtain the same difference of accuracy as between 0 and 10 hidden states (Figure 19).
With these tables, we can also conclude that the number of iterations (500 or 1000) to train the RNN does only have a little impact on the performance.

	Threshold							
Nb. Hidden states	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0	19.37	34.65	37.67	37.14	34.17	29.43	21.89	9.70
10	23.41	37.17	39.57	38.89	36.22	31.80	23.88	10.68
20	26.85	38.03	40.08	39.42	36.75	32.13	23.64	10.65
30	27.11	37.59	40.14	39.19	37.02	32.81	25.66	11.60

Table 17: Accuracy of the testing data of directory dr1 for various numbers of hidden states with a RNN network of 10 forward and 10 backward states trained on *500 iterations* with the full set of training data.

The only point that can be improved now is the latdec program itself.

	Threshold							
Nb. Hidden states	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0	19.13	33.79	36.52	36.52	34.08	28.89	22.04	10.68
10	24.47	37.35	39.87	38.86	36.49	31.95	23.49	10.68
20	27.65	38.59	40.31	39.60	36.96	32.75	24.95	10.98
30	26.85	38.09	40.70	39.66	37.53	33.17	26.13	11.48

Table 18: Accuracy of the testing data of directory dr1 for various numbers of hidden states with a RNN network of 10 forward and 10 backward states trained on *1000 iterations* with the full set of training data.

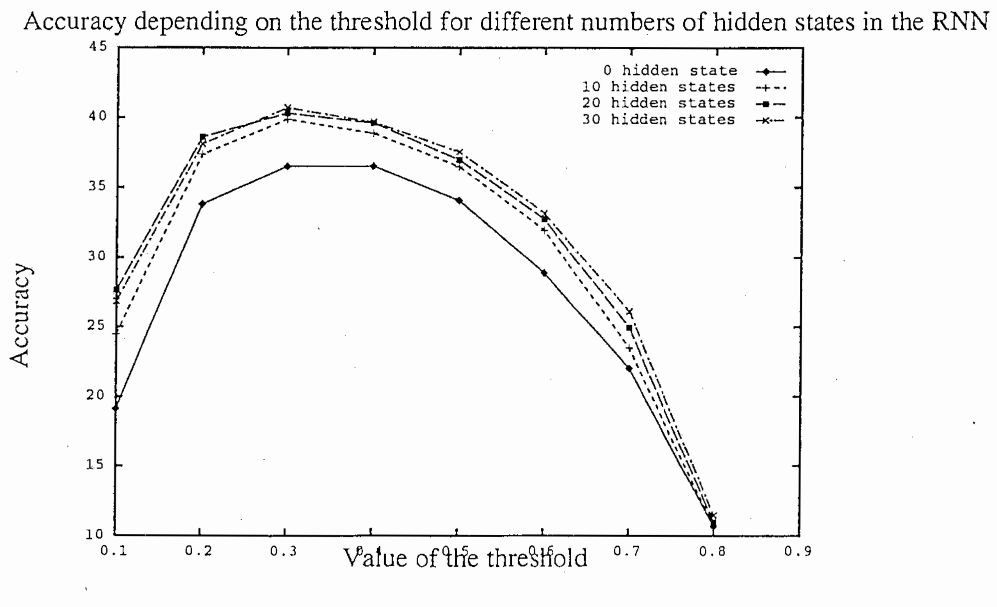


Figure 19: Accuracy of the testing data (directory dr1) for various numbers of hidden states(0, 10, 20, 30) with a RNN network of 10 forward and 10 backward states trained on 1000 iterations with the full set of training data.

7.5 Latdec program

7.5.1 Positive and negative time window

By changing the values of the parameters of the positive and negative time window, we tried to obtain best results.

Those parameters were set to 0 till now; by setting both of them to 1, we increase the number of possible links between the nodes because the system considers the links between the frames just before and just after each node (Figure 20).

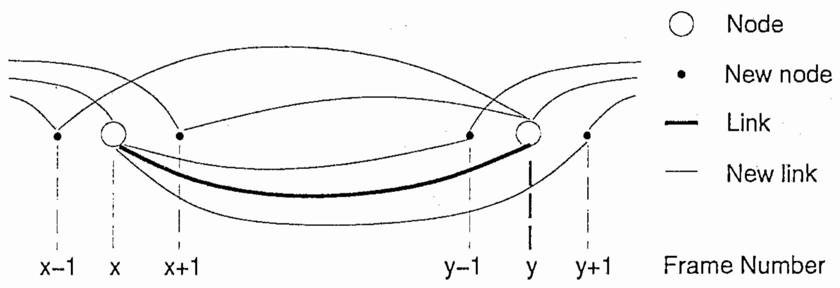


Figure 20: Representation of the network of the links between two nodes with parameters -b and -c set to 1.

The results are not those expected. For instance for the RNN which has 10 forward, 10 backward and 20 hidden states, with the two thresholds set to 0.3 and considering 500 iterations for the training, the accuracy values obtained for the directory dr1 of the testing data are:

D is set to -1	b and c set to 0: 40.08%
	b and c set to 1: 39.69%
	b and c set to 2: 39.57%

We were expecting better results than in the previous case, but the deletion and the substitution of phonemes increase whereas the number of phonemes well identified does not increase. The other problem is that it takes more time to run such a program with those links added.

7.5.2 Short segment evaluation

With the current latdec program, it is impossible to identify the phonemes of less than 3 frames, they are transformed into one main boundary. In the testing data we use, there are 3797 phonemes of 2 frames, so we have to add this case to the latdec program to be able to find them and to increase the accuracy. To calculate their likelihood, we use the mean values.

As we consider Second order PSM, the function is like this between the t-values 0 and 1:

$$o(t) = at^2 + bt + c \tag{12}$$

We can calculate the areas between 0 and 0.5 (s1), 0.5 and 1 (s2), and their mean values (m1 and m2 respectively):

$$s1 = \frac{a}{24} + \frac{b}{8} + \frac{c}{2}$$

$$\begin{aligned} m1 &= 2 * s1 && (13) \\ s2 &= \frac{7a}{24} + \frac{3b}{8} + \frac{c}{2} \\ m2 &= 2 * s2 && (14) \end{aligned}$$

By using these mean values and the original covariances (σ), log likelihoods for those short segments of 2 frames (x1, x2) can be computed as follows:

$$log\ likelihood = Constant + \sum_{i=1}^{dim} \frac{(x1 - m1)^2}{\sigma} + \sum_{i=1}^{dim} \frac{(x2 - m2)^2}{\sigma} \tag{15}$$

with dim corresponding to the dimension of each model (26 in our case).

Another theoretical approach which gives approximately the same results consists in using the mean values of a linear model. Under the assumption the linear time wrapping is valid, the approximation of a 2 frame segment to a second order polynomial in a normalized time ($at^2 + bt + c$) is to fit the linear model (a't+b') to the curve at time 0 and 1. This leads to the solution:
$b' = c$ and $a' = a + b$

These two methods to estimate the Likelihood of the short segments are very similar as regards as results (about 0.2% of accuracy and percentage of good phonemes of difference). In the next part, we give the best results which are obtained with the second way estimate short segments.

The final results obtained with this new module in the latdec program are presented in the next part in paragraph 9.1.

Part III
Results

8 Results of the Landmark Detector

8.1 Test with the new data and new RNN

As we have found some ways to improve the results of the recognition, we are going to see if such changes in the data and the structure of the RNN will improve the results of the first part: Landmark detector.

We used a RNN with 10 forward and 10 backward states, 30 hidden states, trained on 1000 iterations. The data as outputs of this RNN are the values 1, 0.5 and 0 (for the training set).

From Table 19, we can see the improvements of the results on the testing set (50279 true boundaries without considering the extremities of each file) with the following parameters: upper threshold = 0.4, lower threshold = 0.1, margin = 2, step = 2. The percentage of good boundaries needs to be very high in order to be able to find the good boundaries among the list of the possible ones. In stage 3, the percentage of frames that will need to be checked as possible landmarks is 19.33% (at stage 2 it was 28.31%).
The percentage of deletion and the accuracy are defined as follows:

$$Accuracy = 100 * \frac{N-D-I}{N} \quad and \quad Perc.\ of\ Good\ boundaries = 100 * \frac{H}{N}$$

Performance	*Step 1*	*Step 2*	*Step 3*
Hit (H)	40028	48818	48818
Insertion (I)	2291	67529	30611
Deletion (D)	10251	1461	1461
Perc. of good boundaries	79.62	97.10	97.10
Accuracy	75.05	-37.21	36.21

Table 19: Performance of the different steps according the number of well-identified (H), inserted (I), deleted (D) boundaries, the percentage of deletion, the accuracy and the reducing rate. Parameters: upper threshold 0.4, lower threshold 0.1, margin 2, step 2.

From the first two columns of Table 21 we see that the results concerning the phonemes are not as good as those obtained for the boundaries. For instance the percentage of good phonemes was 97.10% and it drops to 90.40% for the phonemes.

From the last two columns of Table 21, we can see that the parameter "distance" improves a little the results by avoiding to go through some too close nodes which correspond to the same true one. With this parameter the percentage of good phonemes is increased to 90.49%.

In order to compare with the HMM results, we have to use the same criteria. That is the reason why we have calculated the performance of our system one the testing data by using only one threshold and only main boundaries (the local maxima above this threshold).From Table 20 we can see that the maximum of percentage of good phonemes is reached for the threshold 0.5.

From Tables 22 and 24, we can see that with a threshold set to 0.5 (the best value), the percentage of good boundaries is 75.06%, and the percentage of good phonemes is 71.55%

Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Accuracy	-62.97	-17.18	17.10	34.81	40.36	34.00	12.85	-38.00
Perc. good phonemes	19.88	42.78	59.92	68.78	71.55	68.37	57.80	32.37

Table 20: Accuracy and Percentage of good phonemes obtained with an RNN with 10 forward and 10 backward states, 30 hidden states, trained on 1000 iterations with outputs 1, 0.5 and 0. The margin was set to 2 and there are only main boundaries.

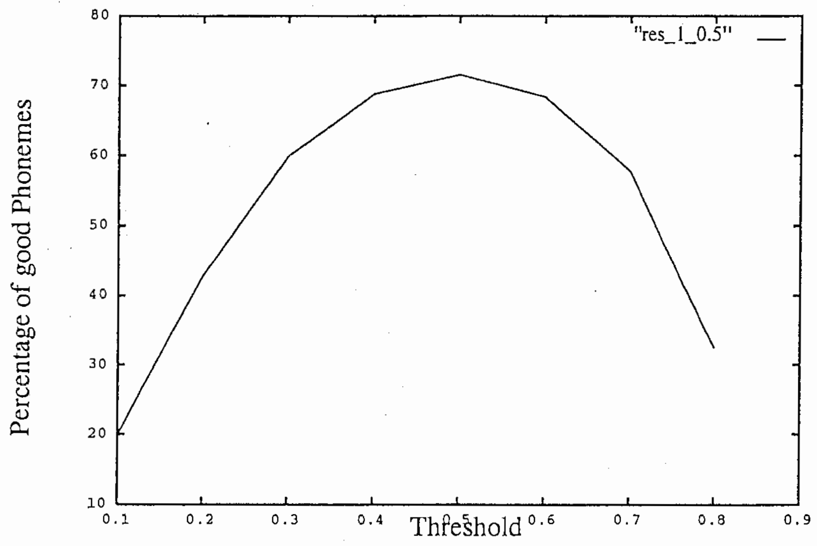


Figure 21: Percentage of good phonemes obtained with an RNN with 10 forward and 10 backward states, 30 hidden states, trained on 1000 iterations with outputs 1, 0.5 and 0. The margin was set to 2 and there are only main boundaries.

with 2 as a margin.

	Boundaries - Step 3	Phonemes - 1	Phonemes - 2
True (N)	50279	48936	48936
Hit (H)	48818	44241	44284
Insertion (I)	30611	5681	5360
Deletion (D)	1461	4695	4652
Perc. good...	97.10	90.40	90.49
Accuracy	36.21	78.79	79.54

Table 21: Comparison of the performance concerning boundaries and phonemes (-1 without the parameter "distance"; -2 the parameter "distance" is set to 2). The other parameters have the same values as for Table 1.

	Boundaries			
	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	50279	50279	50279	50279
Hit (H)	22090	35967	37353	37716
Insertion (I)	16409	2645	1283	922
Deletion (D)	28189	14312	12926	12563
Perc. good...	43.93	71.53	74.59	75.01
Accuracy	11.30	66.27	71.74	73.18

Table 22: Results obtained for the testing data. Comparison between the performance of the boundaries for different values for the parameter margin (there is only one threshold set to 0.5, and only main boundaries)

	Boundaries			
HMM - 1 mixt.	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	50279	50279	50279	50279
Hit (H)	9985	29385	37741	40106
Insertion (I)	33123	13993	5916	3723
Deletion (D)	40294	20894	12538	10173
Perc. good...	19.86	58.44	75.06	79.77
Accuracy	-46.02	30.61	63.29	72.36

	Boundaries			
HMM - 3 mixt.	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	50279	50279	50279	50279
Hit (H)	9045	27791	37851	40760
Insertion (I)	34864	16389	6639	3909
Deletion (D)	41234	22488	12428	9519
Perc. good...	17.99	55.27	75.28	81.06
Accuracy	-51.35	22.68	62.08	73.29

	Boundaries			
HMM - 5 mixt.	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	50279	50279	50279	50279
Hit (H)	8931	27807	37991	41095
Insertion (I)	35174	16575	6690	3755
Deletion (D)	41348	22472	12288	9184
Perc. good...	17.16	55.30	75.56	81.73
Accuracy	-52.19	22.34	62.25	74.26

Table 23: For HMM with 3 states and 1, 3 or 5 mixtures. Comparison of the performance concerning phonemes with different values for the parameter margin.

	Phonemes			
	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	48936	48936	48936	48936
Hit (H)	12623	31953	35016	36057
Insertion (I)	37656	18326	15263	14222
Deletion (D)	36313	16983	13920	12879
Perc. good...	25.79	65.30	71.55	73.68
Accuracy	-51.15	27.84	40.36	44.62

Table 24: Results obtained for the testing data. Comparison between the performance of the phonemes for different values for the parameter margin (there is only one threshold set to 0.5, and only main boundaries).

	Phonemes			
HMM - 1 mixt.	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	48936	48936	48936	48936
Hit (H)	3971	14624	25012	29772
Insertion (I)	46308	35655	25267	20507
Deletion (D)	44965	34312	23924	19164
Perc. good...	8.11	29.88	51.11	60.83
Accuracy	-86.52	-42.98	-0.52	21.68

	Phonemes			
HMM - 3 mixt.	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	48936	48936	48936	48936
Hit (H)	3646	12024	23176	29197
Insertion (I)	46633	38255	27103	21082
Deletion (D)	45290	36912	25760	19739
Perc. good...	7.45	24.57	47.35	59.66
Accuracy	-87.84	-53.60	-8.02	16.58

	Phonemes			
HMM - 5 mixt.	Margin = 0	Margin = 1	Margin = 2	Margin = 3
True (N)	48936	48936	48936	48936
Hit (H)	3514	11959	23091	29875
Insertion (I)	46765	38320	27188	20404
Deletion (D)	45422	36977	25845	19061
Perc. good...	7.18	24.43	47.18	61.04
Accuracy	-88.38	-53.86	-8.37	19.35

Table 25: For HMM with 3 states and 1, 3 or 5 mixtures. Comparison of the performance concerning phonemes with different values for the parameter margin.

8.2 Comparison with HMM

Tables 23 and 25 show the same results for some HMM cases (with 3 states and 1, 3 or 5 mixtures). From Table 23, we can see that the results concerning the percentages of good boundaries increase when the mergin increases from 0 to 3. It reaches the same value as our method but for margin equals to 0 or 1 our method gives much better results.

From Table 25, we can see that the percentages of good phonemes are very similar for these three models (1, 3 or 5 mixtures). As for the boundaries, in each case, the mergin set to 3 improves the results compare to those obtained with the values 0 or 1. As far as phonemes are concerned, our system obtains a percentage of good phonemes better than HMM cases even with the margin set to 3. By comparing our results with the best results obtained with HMM (3 mixtures), we can see that our method is much more successful. We obtain 73.68% as percentage of good phonemes and the accuracy is 44.62% whereas the HMM only reaches the 61.04% of good phonemes and the accuracy is 19.35%.

9 Final Results

Examples of all result files are in the Appendix B.

9.1 Results obtained

As in this part we are able to identify the short segment (2 frames) (for the theory, see previous paragraph 7.5.2), we should obtain better results than before.

Now, there are two different approaches to consider the landmarks of the short segments:

- One, it to consider all landmarks as main landmarks, so the final path with the identification of the phonemes will go through all landmarks.
- The other possibility is to consider the landmarks of the short segments as second ones. In this case, if the maximum of Likelihood is increased by going through those nodes then they will be landmarks of identified phonemes.

In the next tables, the first approach is characterized by the label "main only" and the second one by the label "main and second".

9.1.1 Effectiveness of duration probability

First, we check the fact that the option -D set to -1 for the Latdec program gives better performance than the value 0. To know the duration of the phonemes helps makes easier their recognition. Table 26 illustrates this point for the full set of testing data with the threshold set to 0.3 (because we already know that with only one threshold the maximum performance is obtained around this value).

	Option -D	
Nb. Hidden states	0	-1
20 (main only)	37.98	39.55
	43.16	44.70
20 (main and second)	38.51	40.04
	42.83	44.47
30 (main only)	38.54	40.14
	43.84	45.37
30 (main and second)	39.08	40.66
	43.43	45.10

Table 26: Accuracy (first line) and Percentage of Good phonemes for the full set of testing data with the newest version of the latdec program (recognition of short segments) with the threshold set to 0.3

More, this table lets us see that the best performances (accuracy and percentage of good phonemes) are reached with the RNN which has 30 hidden states (10 forward and 10 backward states) and by considering main and second landmarks (second ones for the short segments).

9.1.2 Effectiveness of short segment evaluation

The recognition of short segments improves the results very slightly.

As Table 26 shows the best results are obtained with 30 hidden states, the option -D set to -1 and considering main and second landmarks. With such parameters, we are going to compare the results of this latest program with the previous one (Table 27).

		Threshold					
	Version	0.1	0.2	0.3	0.4	0.5	0.6
Accuracy	Previous Version	26.85	38.09	40.70	39.66	37.53	33.17
	Latest Version	26.49	38.15	41.03	40.34	38.15	33.73
Perc. Good Phonemes	Previous Version	47.23	45.74	43.81	41.32	38.48	34.08
	Latest Version	47.88	46.66	44.82	42.33	39.28	34.83

Table 27: Accuracy and Percentage of Good phonemes of the testing data of the directory dr1 for the previous and latest versions (30 hidden states, 1000 iterations, -D -1, main and second landmarks for the latest version). The latest version corresponds to the one with the recognition of short segments, the previous version is the one just before without the recognition of short segments.

Considering only one threshold to demarcate the possible boundaries, the increase of the best value of the accuracy is not very significant (about 0.3%, from 40.70% to 41.03%, for directory dr1 of the testing data) whereas, for the same threshold (0.3) the percentage of the good phonemes increases by about 1% (from 43.81% to 44.82%.

We hope biggest increases for those values but one fact is that it is more difficult to identify short segments because they have less time (only two frames) to "point out" their differences and to identify them with one another.

9.1.3 Two thresholds

In part 3.2, two thresholds were used to obtain better results regarding the possible boundaries of the phonemes. It is time to see if this kind of pre-processing may improve the final performance.

Table 28 illustrates these tries for different values of the 2 thresholds, for the data of a RNN with 30 hidden states considering "main only" and "main and second" landmarks (with two thresholds, the files created with "main only" landmarks have second landmarks also, those located between the two landmarks).

Table 28 shows that, with two thresholds, the results are better with "main only" than with "main and second". This can be explained by the fact that between the two thresholds the local maxima are second boundaries.

Another fact is that the use of two thresholds seems to improve the recognition because in the next table (Table 27) the maximum of the accuracy 41.03% was reached for the threshold 0.3 and "main and second" landmarks, whereas with the two thresholds 0.25 and 0.6 and "main only" landmarks the maximum of accuracy is 41.92%.

By trying the values 0.25 and 0.6 for the thresholds on the full testing data with "main only" boundaries, the results are:

* Accuracy = 41.33%

Lower thres.	*Upper thres.*	*Main only*	*Main and Second*
0.2	0.2	37.29	38.15
0.2	0.5	41.00	40.82
0.2	0.6	41.06	40.82
0.2	0.7	40.97	40.82
0.25	0.55	41.89	41.65
0.25	0.6	41.92	41.68
0.25	0.65	41.80	41.68
0.3	0.3	40.88	41.03
0.3	0.6	41.68	41.41
0.4	0.4	40.34	40.34

Table 28: Accuracy of the testing data (dr1) with two thresholds and two kinds of data "main only" or "main and second". Those data were obtained with a RNN with 10 forward and backward states and 30 hidden states.

* Percentage of good phonemes = 45.32%

These results are better than with only one threshold, indeed the maximum of accuracy for the whole testing set was 40.66%; so the improvement is small (about 0.7%) and the computational time may larger.

9.2 Best results obtainable

By running the latdec program on the outputs of the RNN with the training data as inputs, we can see the maximum performance that we could hope to reach with the testing data.

Table 29 shows that this maximum of performance (accuracy) is around 40% depending on the structure of the RNN and of the structure of the data. The difference between "main only" and "main and second" sections consists in the fact that in "main and second" section the landmarks separated by only one landmark are considered as second ones whereas in "main only" section they are main landmarks. Moreover, in each case, the option -D set to -1 gives a better result than the value 0. The maximum values seem to be obtained with the RNN build with 30 hidden states with main and second landmarks (these values are calculated from dr1 of the training data only).

Table 29 also underlines the fact that the maximum of good phonemes will not exceed a value between 45% and 47%.

In Table 29, the results are given for the threshold 0.3, but we have to be aware of the fact that they may be slightly better for another value of the threshold (0.25, 0.35...), or with two different thresholds.

9.3 Comparison with HMM

The aim of this work is to improve the performance of a Speech Recognizer based on Polynomial Segment Model compared to the performance obtained with the Hidden Markov Model.

9.3.1 Computational requirement comparison

From Table 30, we can see that one problem of PSM compared to HMM is that HMM is about three times faster than PSM.

	Option -D	
Nb. Hidden states	0	-1
20 (main only)	38.63	40.43
	43.19	44.88
20 (main and second)	39.09	40.74
	42.81	44.44
30 (main only)	39.01	40.79
	43.80	45.50
30 (main and second)	39.63	41.31
	43.57	45.29

Table 29: Accuracy (first line) and Percentage of good phonemes (second line) for training data (directory dr1 only) depending on the -D option and the number of hidden states in the RNN of 10 forward and backward states. The threshold value was set to 0.3.

	HMM (3 states)	PSM
System	7.56 sec.	198.67 sec.
User	748.3 sec.	2186.32 sec.

Table 30: Time requirements for running HMM and PSM on the testing data

9.3.2 Accuracy and Percentage of good phonemes

	Nb of States		
Data	1	2	3
Full testing data	16.27	40.08	40.86
	42.74	44.68	42.58
testing data of dr1	14.92	38.92	40.52
	41.92	44.14	42.36

Table 31: Accuracy (first line) and Percentage of Good phonemes (second line) of the testing data obtained with the HMM method with 1, 3 or 5 states, and 1 mixture per state.

The results obtained with the HMM-based recognizer (Table 31) seem to be approximately the same results as those obtained with the PSM-based recognizer (Table 26). Even the PSM-based recognizer is a bit better but it takes more time to make it run on the full set of testing data than the HMM-based one.

It is interesting to see the difference of performance according the phonemes. This shows whether these two approaches are better for the same phonemes or different ones. Table 32 gives the number of well recognized phonemes for some of them (those for which the difference between HMM and PSM methods is the most significant).

When the PSM results are not so good compared to the HMM results as for phonemes axr, b, s and y (Table 32), two explanations are possible:

- Some phonemes have a waveform (that leads to cepstral coefficients) badly balanced as in Figure 22 (left). So it is very difficult for a second order curve to fit the original one.

- Other phonemes can have very different waveforms as in Figure 22 (right). So it is impossible to identify all of them with only one model curve.

	Recognizer	
Phonemes	HMM	PSM
ax	353	450
axr	456	380
b	227	153
dh	275	57
dx	176	274
eh	307	387
sil	2293	1981
ih	363	483
ix	696	831

	Recognizer	
Phonemes	HMM	PSM
l	333	488
n	678	870
p	384	487
pcl	725	595
q	126	233
s	1615	1504
tcl	396	489
y	245	150

Table 32: Number of phonemes correctly identified by the two methods for the phonemes whose difference of performance between HMM and PSM methods is the biggest.

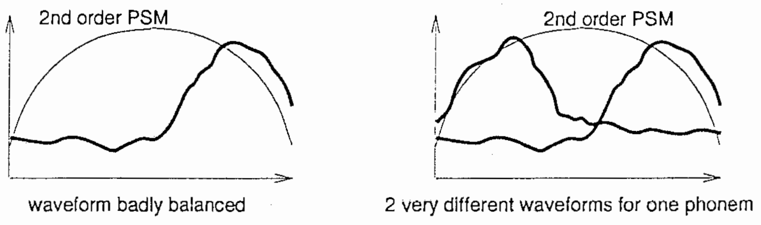


Figure 22: Left: Waveform badly balanced for a 2nd order PSM model; Right: 2 very different waveforms for the same phoneme

As the vowels are more stable than other phonemes, that means they have more regular waveforms (in terms of repartition in time and of repetition), they should be better identified with the PSM-based recognizer than with the HMM-based one. The results obtained (Table 33) are not so obvious.

- For 4 vowels (ae, ey, iy and oy), the HMM-based recognizer is better.
- For 5 vowels (aw, ay, uh, uw, ux), the two systems seem to be approximately equivalent.
- For 7 vowels (aa, ah, ao, eh, er, ih, ow) , the PSM-based recognizer gives better results.

	Recognizer	
Phonemes	HMM	PSM
aa	317	335
ae	430	409
ah	239	258
ao	309	339
aw	123	122
ay	420	422
eh	307	387
er	315	345

	Recognizer	
Phonemes	HMM	PSM
ey	544	508
ih	363	483
iy	1156	1074
ow	178	196
oy	90	46
uh	41	48
uw	77	69
ux	160	161

Table 33: Number of vowels correctly identified by the two methods.

10 Conclusion

In this work, we try to see if the performance of a speech recognizer based on RNN and PSM could reach the performance of HMM.

Our results are very similar compared to HMM results; but one point which is not so good is the fact that the computational requirements of RNN and PSM are higher than those needed for HMM case.

A further improvement of our system could be to use variance trajectory model [9] in order to make the system much efficient and reliable.

References

[1] M. Bacchiani and M. Ostendorf. Software for design and use of a speech recognition system based on automatically derived non-uniform units. Technical report, ATR, Interpreting Telecommunications Research Labs., Japan, 1996.

[2] H. Gish and K. Ng. A segment speech model with applications to wordspotting. *ICASSP*, 1993.

[3] J. Chang J. Glass and M. McCandless. A probabilistic framework for feature-based speech recognition. *ICSLP*, 1996.

[4] L.Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall Signal Processing Series, Alan V. Oppenheim, Series Editor, New Jersy, USA, 1993.

[5] M. Biacchiani M. Ostendorf and Y. Sagisaka. Speech recognition system design using automatically learned non-uniform segmental units. Technical report, ATR, Interpreting Telecommunications Research Labs., Japan, 1996.

[6] O. A. Kimball M. Ostendorf, V. Digalakis. From hmms to segment models: A unified view of stochastic modeling for speech recognition. Technical report, EDICS 1.6.3, May 22, 1995.

[7] P. Woodland S. Young and W. Byrne. Htk: Hidden markov model toolkit v1.5 reference manuel. *Cambridge University Engineering Departement Speech Group and Entropic Research Laboratories Inc.*, 1993.

[8] M. Schuster. Learning out of time series with an extended recurrent neural network. *NNSP*, 1996.

[9] Y. Sagisaka T. Fukada and K. Paliwal. Model parameter estimation for mixture density polynomial segment models. *ICASSP*, 1997 (to appear).

Part IV
Appendix

11 Appendix A: How to use all the programs

All the programs have the same organization, they use the same data files. These are the reasons why we have chosen to present them all together.

11.1 Data files

There are two kinds of data files: the original data files with the true boundaries and the estimated data files with the estimated boundaries (obtained after the RNN network).
The extension of the original data files must be ".bnd" and the extension of the estimated data files must be ".bnd.nn_out". Otherwise, these have to be changed in the program.

The original data files and the estimated data files have the same structure.
The data files have to written in ASCII, with one the first line NN_ascii_data, on the second line there should be the number of frames, one the third line the number of column (in general 1) and then a blank line.
At the end of each data file, it must not be a blank line.

The presentation is as follows:

```
NN_ascii_data
4
1

1.000000e+00
7.500000e-01
5.000000e-01
2.500000e-01
```

11.2 List file

To run all the programs, it is necessary to create a file whose content is the name of all the data files. Those names are written without any extension.

The presentation is like this:

```
s0000
s0001
```

s0000 and s0001 are the two names of files whose the program is going to compare the estimated files with the original ones (i.e. s0000.bnd.nn_out with s0000.bnd, and s0001.bnd.nn_out with s0001.bnd).

11.3 To run a program

The command to make one program run is very simple:

```
program_name [options] list_filename
```

The options can be multiple and they are explained in the next part.

11.4 Options

- -t: only in the program "all_eval", correspond to -t_upper (default 0.8).
- -t_upper: value of the upper threshold (default 0.8).
- -t_lower: value of the lower threshold (default 0.5).
- -m: margin value (default 1). The margin is one the left and one the right of each true boundary.
- -s: step (default 1). The value 0 has no meaning. With the value 1, all points above the upper threshold are considered; otherwise, there is a ratio 1/step.
- -d: distance (default 0). This value is the minimum of frames that need to separate two local maxima above the upper threshold so that they are considered as main boundaries, otherwise they become second boundaries.
- -extension (default lat). To change the extension of the output file.

11.5 Programs

11.5.1 Concerning Boundaries

- all_eval.c:
  - Use: all_eval [options] filename_list
  - Function:It calculates the accuracy, the number of deleted and inserted boundaries for a list of files. The estimated boundaries are the points above the threshold. It is used in the paragraph 3.1.
  - Options: -t, -m.

- all_find.c:
  - Use: all_find [options] filename_list
  - Function: It calculates the accuracy, the number of deleted and inserted boundaries and the percentage of good boundaries for a list of files. The estimated boundaries are the points above the upper landmark and the local maxima between the two landmarks. It is used in the paragraph 3.2.
  - Options: -t_upper, -t_lower, -m

- landmark_step1.c:
  - Use: landmark_step1.c [options] filename_list
  - Function: It calculates the same values and the reducing rate. The estimated boundaries are the points above the upper threshold separated by "step-1" frames when they are following points above the threshold and the local maxima between the two thresholds. It is used in the paragraph 3.3.
  - Options: -t_upper, -t_lower, -m, -s

- main_landmarks.c:
  - Use: main_landmarks [options] filename_list
  - Function: It creates a list of the main landmarks (points local maxima above the upper threshold) and a list of the second landmarks (the other points above the upper threshold and the local maxima between the two thresholds).
  - Options: -t_upper, -t_lower, -m, -s

- perf_main_landmarks.c:
  - Use: perf_main_landmarks [options] filename_list
  - Function: It creates the same lists as the previous program, and it evaluates the performance of the way to establish the boundaries compare to the true ones.
  - Options: -t_upper, -t_lower, -m, -s

11.5.2 Concerning Phonemes

- distance_boundaries.c:
  - Use: [options] filename_list
  - Function: It has the same outputs as perf_main_landmark.c but it considers that when two main boundaries are not separated by more than "distance" frames, then they are considered as second boundaries. It is used in the paragraph 4.2. For the previous paragraph (paragraph 4.1), the program used was an old version of distances_boundaries; now we use this latter with the parameter distance set to 0.
  - Options: -t_upper, -t_lower, -m, -s, -d

11.5.3 For the latdec program

- create_lattice_file.c:
  - Use: create_lattice_file [options] filename_list
  - Function: The input is one file whose extension is "bnd.nn_out" and it creates the lattice file from the list of the second and main boundaries (the second boundaries are all local maxima between the two thresholds and the points above the upper threshold which are not local maxima - with the parameter "step"). It is possible to test the files with only one threshold by giving the same value to both upper and lower thresholds.
  - Options: -t_upper, -t_lower, -s, -d

- create_lattice_file_major.c:
  - Use: create_lattice_file_major [options] input_file output_file (the last 3 characters are removed to be changed into the extension).
  - Function: It has the same aim as the previous program but it considers as second boundaries only the local maxima between the two thresholds. It is possible to test the files with only one threshold by giving the same value to both upper and lower thresholds.
  - Options: -t_upper, -t_lower, -extension

The goal of these programs create_lattice_file.c and create_lattice_file_major.c is to create a list of all the possible boundaries of phonemes and a list of the links that may exist between them.

From the probabilities of the frames to be boundaries, such programs determine which of them are possible boundaries, called nodes. There are two kinds of nodes: main and second ones according to their probabilities to be boundaries.

The links are created from one node to another one according some rules. One link can never go over a main node. For instance, if 65 and 87 are the frames of second nodes and there is a main node at the frame 72, then the links 65-72 and 72-87 exist but the link 65-87 does not exist.

The programs create_lattice_file and create_lattice_file_major are quite similar because they create lattice file from the outputs of the RNN network and the mfcc files coresponding. Their difference is in the approach to consider the frame probabilities and the way to choose some of them as nodes.

- Inputs:

  The inputs are two kinds of files. The less important is the mfc file because it is only used to give back to the file being treated its right name. The output file of the RNN network, called in the remaining of this part "sXXXX.bnd.nn_out", is the only file really useful. It has the following structure:

> NN_ascii_data *type of data*
> number of data vectors (int, ASCII)
> 1 *number of outputs*
> *blank line*
> values (between 0 and 1)
> ⋮

- Outputs:

The output is only one file, whose name will be the name of the mfc file with its last three letters of its name changed into "lat". There is one lattice file for each couple of mfc file and sXXXX.bnd.nn_out file.
The nodes are defined by a number. The lattice file lists the number of nodes, the number of links, the nodes with their attributed number and the time they correspond to in the sentence (in seconds), the structure of all links (a number attributed, the start node, the end node, the WORD IDentifiation). The word identification corresponds to the number of the phoneme tested for this link. As we are working with a set of 61 phonemes, there will be 61 tests for each link.
A lattice file has the following structure:

> NODES=*number of nodes in this sentence*
> LINKS=*number of links between second, main and second, and main*
>
> I=1 *number of this node*
> time=*translation in seconds of the number of this frame*
> I=
> time=
> ⋮
>
> J=1 *number of this link*
> START=*number of the nodes in which this link starts*
> END=*number of the node in which this link ends*
> WORDID=*identification number of a phoneme of the data*
>
> J=
> START=
> ⋮

This structure is required to use the latdec software made by M. Bacchiani and M. Ostendorf.

- Difference between these two programs:

The only difference between the programs create_lattice_file and create_lattice_file_major is the way to consider the nodes as main boundaries or second ones.

The program *create_lattice_file_major* considers only the local maxima above the upper threshold as main boundaries (i.e. main nodes) and the local maxima between the two thresholds as second boundaries (second nodes).

The program *create_lattice_file* considers the same main nodes but the second nodes are the frames which are local maxima between the two thresholds and the points above the upper thresholds with the notion of "step". The "step" parameter allows the user to choose if he wants to have all these points, one half, one third of these points.

- For several files:

In order to make create_lattice_file_major run on several files (a directory), I have made shell script whose name is **phonem_detector_major.s**. This shell script not only runs the create_lattice_file_major program for several files, but it also runs the latdec program and it gives th performance of the system in terms of accuracy and percentage of good phonems thanks to the tool HResults from the HTKtest tolls.

The architecture of the files should be as it is represented in Figure 23 in order to keep the structure of the "foreach" loops of the schell program.

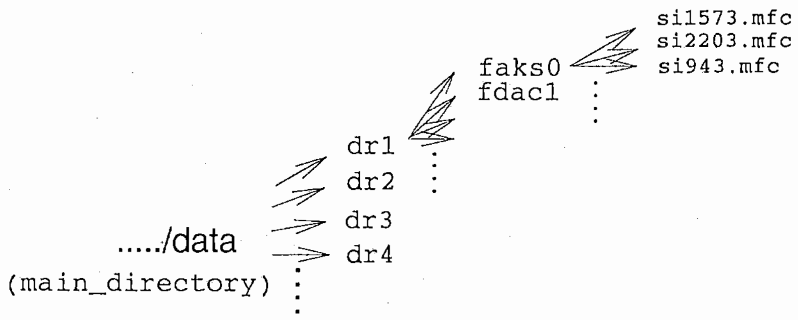


Figure 23: Organisation of the data files

The schell program is organized as follows:

- main_directory: sets the path to reach the first node of the data files as Figure 23 shows.
- file_bnd: sets the path to reach the ".bnd.nn_out" files.
- upper: sets the value of the upper threshold.
- lower: sets the value of the lower threshold.
- extension: sets the extension of the lattice file. By default, its name is the mfc file whose extension is changed into "lat".
- latdec: sets the extension given to the files which are the outputs of the latdec program.
- script: sets the name of the file in which the names of the latdec-files are written in order to realize the program result.s.

The only thing to be careful to change, if necessary, the script file used in the part results.s of the schell script phonem_detector_major.s. The file lists all the files to be read for the evaluation of the performances (by default, it contains lab_new files). If the upper and lower thresholds have the same value, it works as if it only was one threshold.

12 Appendix B: the different kinds of files

These files correspond to the "first" file of the testing data: /drl/fasks0/si1573.*.

The original sentence is (si1573.txt):
0 79565 His captain was thin and haggard and his beautiful boots were worn and shabby.

It has been separated into phonemes (dictionary: 61 phonemes) and the beginning and the end of each phoneme is expressed into milliseconds (si1573.lab). In Table 34, there are examples of such files obtained with HMM-based recognizer and PSM-based recognizer.

LAB file	*REC file*	*LAB_NEW file*
0 5575000 sil	0 300000 b	0 2178000 sil 203.207489
5575000 5956250 hh	300000 2100000 sil	2178000 5578000 pau 651.526611
5956250 6683750 ih	2100000 3200000 t	5578000 5878000 d 37.148560
6683750 7675625 z	3200000 5800000 pau	5878000 6578000 ih 126.724709
7675625 8081250 kcl	5800000 6700000 ih	6578000 7678000 s 142.326019
8081250 8568750 k	6700000 7700000 s	7678000 7978000 kcl 3.008018
8568750 9816875 ae	7700000 8000000 kcl	7978000 8478000 k 53.198250
9816875 10962500 pcl	8000000 8500000 k	8478000 9678000 ae 72.374687
10962500 11143750 t	8500000 9700000 aw	9678000 9978000 q -80.167198
11143750 11748750 ix	9700000 11100000 q	9978000 11078000 q -1.238128
11748750 12600625 n	11100000 11800000 eh	11078000 11678000 ih 15.260043
12600625 13158750 w	11800000 12700000 nx	11678000 12578000 n 74.577003
13158750 13603125 ax	12700000 13200000 w	12578000 13578000 oy 140.129807
13603125 15116875 s	13200000 13700000 ax	13578000 15078000 s 311.884491
15116875 15787500 th	13700000 15100000 s	15078000 15778000 th 127.877625
15787500 16775000 ih	15100000 15500000 pcl	15778000 16678000 iy 146.200058
16775000 17318750 n	15500000 15800000 g	16678000 17278000 nx 7.782028
17318750 17966875 ae	15800000 16600000 iy	17278000 17978000 eh 8.468667
17966875 18772500 n	16600000 17200000 nx	17978000 18678000 ng 9.960123
18772500 19723750 hv	17200000 18000000 eh	18678000 18978000 ix 9.853329
19723750 21435625 ae	18000000 18800000 ng	18978000 19678000 hv 151.826523

Table 34: Examples of labl files: LAB file contains the true frames (in time) of the boundaries, REC file is obtained for HMM, LAB_NEW file is obtained for PSM