Fast Two-Level-Dynamic-Programming Algorithm For Speech Recognition

Agbago, Akakpo; Barrière, Caroline

*Fast Two-Level-Dynamic-Programming Algorithm for Speech Recognition **

Agbago, A., and Barrière, C.
May 2004

* published at IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004). Montréal, Québec, Canada. May 17-21, 2004. NRC 48074.

FAST TWO-LEVEL-DYNAMIC-PROGRAMMING ALGORITHM FOR SPEECH RECOGNITION

Akakpo AGBAGO[1] *and Caroline BARRIERE* [1,2]
1. School of Information Technology and Engineering, University of Ottawa
2. Institute for Information Technology, National Research Council of Canada

ABSTRACT

A three-stage architecture for speech recognition is presented including pre-processing, phoneme recognition, and natural language post-processor. Within this context of phoneme-based utterance recognition, this paper focuses on the often problematic speed of the second stage and reengineers a standard Two-Level Dynamic Programming (TLDP) approach to achieve an increase in speed of 75%. Our Fast Two-Level Dynamic Programming Algorithm (FTLDP) uses a phoneme clustering technique to reduce the reference search space and silence detection to reduce the length of the utterance to recognize. An overview of the FTLDP algorithm is presented as well as some results.

1. INTRODUCTION

With the growing interest in Automatic Speech Recognition (ASR) applications[1], the work of researchers in this field in the recent years has focused mainly on increasing two numbers: the accuracy and processing speed of speech recognition. Since precise speech model boundaries help achieve high accuracy, most researchers use isolated or connected words as topology [2] and run Hidden Markov Model (HMM) [9] to build valid output sentences. When boundaries are unclear (continuous speech case), Two-Level Dynamic Programming (TLDP) techniques [5, 10] can be used to find them quite well.

These techniques are highly time consuming due to distance, likelihood, probability observation computations especially at phone unit level (phoneme, diphone, etc.) since the progress step becomes small and thus increases the number of iterations. Despite that factor, our objective is to do recognition with an acoustic-phonetic approach for the benefit of its finite and small set of phonetic units to be used as reference. The recognized phonemes will need to be processed into valid words, a step where HMM can help. Therefore, we propose a three-stage architecture shown in Figure 1.1. Its stages can be described as follow:

- Stage 1: a pre-processor using Digital Signal Processing (DSP) to enhance the input signal.
- Stage 2: a recognition processor that will generate multiple strings of phonemes as solutions.
- Stage 3: a Natural Language Processing (NLP) module that refines stage 2's solutions into valid words. Unlike other systems that fuse HMM in stage 2, we suggest it's moved here to run on strings (simple matching of phonemes) rather than utterances (complex distances).

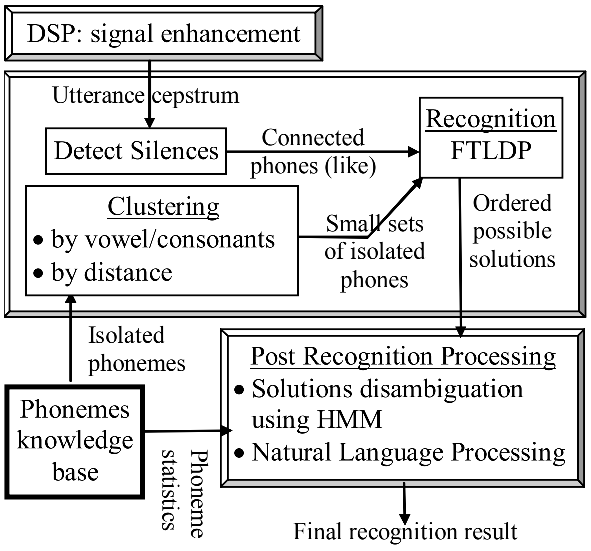

Figure 1.1: Block diagram for fast speech recognition

Without doubting the accuracy or effectiveness of Stage 1 (DSP), the focus of this paper is to have the second stage running a fast TLDP (to help find phone boundaries [8]), but without using HMM process. The TLDP has two levels of processing with computational cost in time depending on the size of the reference models and the length of the utterance to recognize though its first level is the most expensive. We revisit these two levels and suggest some improvements in an algorithm that we call Fast Two Level Dynamic Programming (FTLDP). To do so, we propose: (Technique 1) at the first level, to use clustering to reduce the number of reference phoneme models, (Technique 2) at the second level, (a) to improve the DP algorithm to skip over unnecessary evaluations, (b) to reduce the depth of the process by analyzing the variation of the distortion slope and (Technique 3) to detect silences to reduce the utterance's length for both levels. The result is up to 75% faster than the original as presented by Rabiner in his book [8].

[1] TMA Associates (*www.tmaa.com*) has forecast in 1999, $1.8 billion market for the use of speech technology in telecom industry and Kelsey Group forecasted over 128 million voice portal users by 2005.
http://www.datacommresearch.com/old/mvppr.html

Rafid, Shawn and Anand [2] have proposed a similar approach that uses: 1) silence skipping, 2) silence-based pruning of DP and 3) early decision to reduce the computational load in speech recognition. The technique resulted in 13% reduction of computational complexity on connected digit task and 6.7% on the company name task (connected words topology).

A similar attempt by S. Nkagawa [11] resulted in a reduction by a factor of 4 to 6 the time needed to compute local distances in the improved DP algorithm proposed by Sakoe [5]. Our FTLDP technique differs from [2] by working on the size of the set of reference models and it uses TLDP rather than Tree Structure Vector Quantization (TSVQ) approach [3].

Next chapters will present detail on the suggested techniques used to design FTLDP with the improvements made to the two levels of the TLDP, the results of a test of an implementation of FTLDP and the conclusion.

2. FAST TWO-LEVEL DYNAMIC PROGRAMMING ALGORITHM

The automatic recognition of an utterance T into text is a problem of solving Equ.2.1 for the approximation solution R^{S*} that corresponds to the smallest distortion D^*.

$$D^* = \min_{R^S} D\left(R^S, T\right) \textit{ best distortion of } R^S \textit{ to } T \quad (Equ.2.1)$$

It can be shown [8] that, the solution of the equation is to find (Equ.2.2) an optimal length L^* for R^S, that would be the concatenation of the best reference phoneme utterance models that match consecutive segments of T according to a Dynamic Time Warping (DTW) path function w(m) and whose indexes in a V references library C_V form the optimal sequence $q^*(1), q^*(2), .., q^*(L)$.

$$D^* = \min_{L_{\min} \le L \le L_{\max}} \min_{\substack{q(1),..,q(L) \\ 1 \le q(i) \le V}} \min_{w(m)} \sum_{m=1}^{M} d\left(t(m), r^S(w(m))\right) \quad (Equ.2.2)$$

T and R^S utterances are expressed as in Equ.2.3.

$$\left.\begin{aligned} T &= \{t(1), t(2), t(3), .., t(M)\} = \{t(m)\}_{m=1}^{M} \\ R^S &= \left\{U_{q(1)}, U_{q(2)}, U_{q(3)}, .., U_{q(L)}\right\} = \left\{r^S(m)\right\}_{m=1}^{N^S} \end{aligned}\right\} (Equ.2.3)$$

To cut down the computational cost of Equ.2.2 that is $O\left(M * L * V^L\right)$, two levels of processing is used as shown in next sections.

2.1 FTLDP Level 1

The Level 1 of TLDP consists in finding the best indexes $v^* = q^*(i)$ in Equ.2.2 by solving Equ.2.4 for v^* over all possible segments [b, e] of T scanning all v in C_V.

$$\hat{D}(v, b, e) = \min_{w(m)} \sum_{m=b}^{e} d\left(t(m), r_v(w(m))\right) \quad (Equ.2.4)$$

It results in building matrices $\tilde{D}(b,e)$ for distortions and $\tilde{N}(b,e)$ for the corresponding indexes as in Equ.2.5.

$$\left.\begin{aligned} \tilde{D}(b,e) &= \min_{1 \le v \le V} [\hat{D}(v,b,e)] \textit{ best distortions} \\ \tilde{N}(b,e) &= \arg\min_{1 \le v \le V} [\hat{D}(v,b,e)] \textit{ best indexes} \end{aligned}\right\} (Equ.2.5)$$

We can see that the computation of these two matrices for this Level 1 should be expensive in time and is a function of V, the size of the library C_V, and M, the length of T (Equ.2.6). DTW is the average cost of time warping [8].

$$O_1 = V * M * \overline{DTW} \quad (Equ.2.6)$$

To reduce this cost, our algorithm FTLDP compresses V (Technique 1) and M (Technique 3) parameters by:

a) Clustering C_V to reduce V, the size of the search space,
b) Removing silence segments from T to reduce M.

Technique 1: The clustering technique is very interesting, for the clusters are built only once at the start of the application. It might even have a two nodes structure where at the first node, vowel or consonant decision is made, and at the second node the space is narrowed to a specific phoneme models cluster. FTLDP implements only one node that is clusters of phoneme models. Any clustering technique will work in FTLDP. We used the Modified Binary Split K-Mean Clustering algorithm [6] because it can be setup to generate a specific number of clusters (suitable to shape the structure into clusters of phoneme realizations) or for a maximal intra-cluster distance (favor accuracy). The detail of the clustering will not be shown in this paper for it is a standard problem [7].

Technique 3: Considering only the energy features of cepstrums and computing the first order cepstrum derivative, Figure 2.1 shows patterns that can help cut off silence segments. Our present goal being to present the idea of using silence detection (a standard problem), we will not further detail this approach. Recently, many Voice Activity Detection (VAD) techniques have been used to perform this task successfully [4, 7].

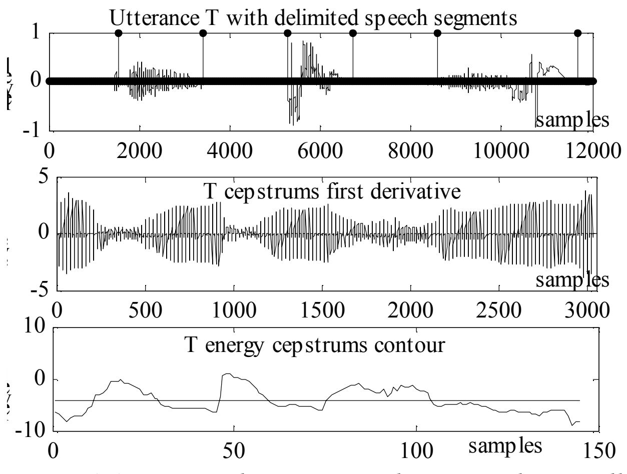

Figure 2.1: First order cepstrum derivative shows nulls for speech regions.

2.2 FTLDP Level 2

This Level 2 completes the resolution of Equ.2.2 in designing the best approximation R^{S*}, by solving for L^*, the optimal length of R^S that becomes R^{S*} using Dynamic Programming (DP) techniques as in Equ.2.7 over the whole utterance T. This involves a recursive process that Rabiner [8] expressed as shown in Equ.2.8-10.

$$\bar{D}_l(e) = \min_{1 \le b \le e}\left[\tilde{D}(b,e) + \bar{D}_{l-1}(b-1)\right] \quad (Equ.2.7)$$

Recursion as proposed by Rabiner's book

Step 1: Initialization

$$\bar{D}_0(0) = 0, \quad \bar{D}_l(0) = \infty, \quad 1 \le l \le L_{\max} \quad (Equ.2.8)$$

Step 2: String of one model, Loop on e for l = 2,..., L_{Max}

$$\bar{D}_1(e) = \tilde{D}(1,e), \qquad 2 \le e \le M \quad (Equ.2.9)$$

Step 3: more than one model, Loop on e for l = 2,..., L_{Max}

$$\left.\begin{array}{l} \bar{D}_2(e) = \min\limits_{1 \le b \le e}\left[\tilde{D}(b,e) + \bar{D}_1(b-1)\right], 3 \le e \le M \\ \bar{D}_3(e) = \min\limits_{1 \le b \le e}\left[\tilde{D}(b,e) + \bar{D}_2(b-1)\right], 4 \le e \le M \\ \bar{D}_l(e) = \min\limits_{1 \le b \le e}\left[\tilde{D}(b,e) + \bar{D}_{l-1}(b-1)\right], l+1 \le e \le M \end{array}\right\} (Equ.2.10)$$

Finale solution

$$\bar{D}^* = \min_{1 \le l \le L_{\max}}\left[\bar{D}_l(M)\right] \quad (Equ.2.11)$$

Equ.2.11 shows that, it is after building L_{max} R^S candidates (length 1 to L_{max} phoneme models) that R^{S*} can be found. Each R^S candidate involves scanning up to M, the length of T. Therefore, the cost in time of this Level 2 depends on M and L_{max} (maximum length guessed for R^S).

To reduce this cost, our algorithm FTLDP compresses the scope of the scan ending at M (Technique 2.a and 3) and tries, if possible, to cut short the process after L_{cut}<L_{max} iterations (Technique 2.b).

Technique 3: As presented earlier in Level 1, silence removal makes M small for this Level 2 too.

Technique 2.a: FTLDP modifies Rabiner's algorithm [8] to make it computationally efficient exploiting the fact that there are fairly long ranges of invalid[1] distortion values resulting from the recursion in Equ.2.10. In fact, if the path distortion at a step l is invalid, because the path distortion at step l +1 is a function of the value at step l, then the resulting path distortion value will also be invalid. Equ.2.12 illustrates the fact stated above.

$$\left.\begin{array}{l} \text{In } Equ.2.7, \forall b_0, \text{if } \bar{D}_{l-1}(b_0 - 1) = \infty \\ \quad \Rightarrow \bar{D}_l(e_0 = b_0 - 1) = \infty \ \forall (e_0 \le b_0) \end{array}\right\} (Equ.2.12)$$

[1] A distortion value is considered invalid if it is a non-real, undefined or infinite number, which implies something unnatural between the utterance arguments that participate in the computation; they are not comparable.

Equ.2.12 revels that, for every $\bar{D}_l$ involving invalid values as shown in Figure 2.2, at least b_0 evaluations can be skipped using a jump function J(l) (Equ.2.13) to determine the *starting value e_0* where the distortion becomes a valid value.

$$\left[\begin{array}{cccc} e & \bar{D}_1(e) & \bar{D}_2(e) & \bar{D}_3(e) \\ 3 & 9 & - & \\ 4 & 10 & - & - \\ 5 & 13 & - & - \\ 6 & 17 & 6+\bar{D}_1(4) & - \\ 7 & 22 & 4+\bar{D}_1(4) & - \\ 8 & 25 & 6+\bar{D}_1(4) & - \\ 9 & 29 & * & 15+\bar{D}_2(6) \\ 10 & 33 & * & 15+\bar{D}_2(7) \\ 11 & 37 & * & 15+\bar{D}_2(8) \end{array}\right] \mathit{where} \left\{\begin{array}{l} -\ \mathit{computation} \\ \mathit{of\ invalid} \\ \mathit{value} \\ \\ *\ \mathit{some\ real} \\ \mathit{number} \end{array}\right.$$

Figure 2.2: Illustration of invalid values impact.

$$J(l) = e_0 \ \textit{such that} \ D_{l-1}(e_0) \ne \infty \quad (Equ.2.13)$$

With a light offset computation overhead, J(l) also helps save memory by keeping only valid ranges in $\bar{D}_l$ matrix. So the correction made to Rabiner's algorithm is shown in Equ.2.14 where J(l) is used to control the iteration ranges.

$$\left.\begin{array}{l} \bar{D}_l(e) = \min\limits_{(J(l-1)+1) \le b \le e}\left[\tilde{D}(b,e) + \bar{D}_{l-1}(b-1)\right], \\ J(l-1) + 2 \le e \le M \ \mathit{with} \left\{\begin{array}{l} J(0)=0 \\ J(l)=\mathit{first} \arg \bar{D}_l(e) \ne \infty \end{array}\right. \end{array}\right\} (Equ.2.14)$$

Technique 2.b: Assuming that the unit which evaluates the distortion between R^S and T is a stable function with respect to the length L of R^S, we can expect a single minimum over the range $1 \le L \le L_{max}$ which corresponds to L^*. Therefore, by analyzing the return values of $\bar{D}_l$, FTLDP is able to find a length L_{cut} such that $L^* + \Delta L \le L_{cut} \ll L_{max}$ where ΔL is the slope study margin. This saves computation and also memory if storage for $\bar{D}_l$ is dynamically allocated.

3. TEST RESULTS

To verify the relative speed performance of FTLDP to TLDP, we have implemented it in java, tested it and got some results that are shown in this section. To smooth out unequal OS influence on processing times, we got the average of 10 processing times for each point of the resulting characteristic graphs in Figure 3.1&2. The first graph shows the performance of the Level 1 of FTLDP, which is 75% less than TLDP time for a structure of 15 clusters and up containing an average of less than 10 models of reference phonemes each. This performance should guarantee good satisfaction for an English language recognizer using nearly 50 clusters of 5 models for each English phonetic unit. For the performance of the Level 2 of FTLDP, Figure 3.2 shows that FTLDP becomes faster than TLDP as T's length becomes large while it costs no more than TLDP for short Ts. This is important for low-level linguistic unit based systems that are likely to deal with large value of L (i.e. long chain of phonemes for the parameter T).

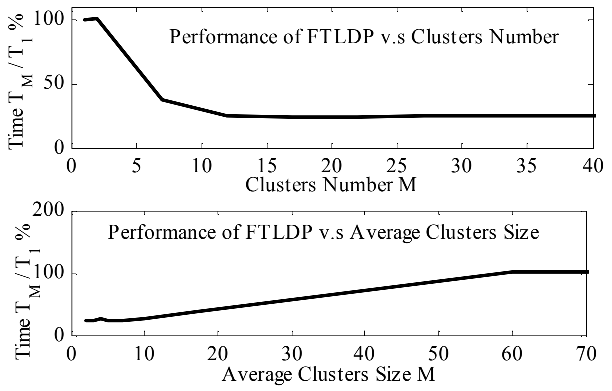

Figure 3.1: Results for Level 1(Clustering effect)

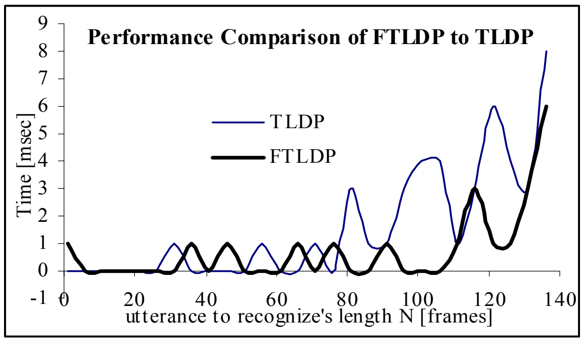

Figure 3.2: Results for Level 2(Jump and Slope Analyzer effects)

We tested the system with several continuously spoken sentences and got similar good speed results and also good recognition results with the reference phoneme models we had. The accuracy of the results of the implementation we used depends on an autonomous module that computes distortion between utterances and importantly, it depends on the quality of phoneme models we dispose as references. For this paper, the focus has not been accuracy but speed improvement of TLDP; therefore the efficiency of that module (black box) is irrelevant in testing FTLDP.

4. CONCLUSION

In this paper, we have presented four techniques (1, 2.a, 2.b, 3) to modify the standard Two-Level DP affecting both of its Levels 1&2. We used silence detection and removal to reduce the length of an utterance to recognize for both levels, clustering to narrow the reference search space for Level 1, and for Level 2, we used a jump function J(l) and a slope analyzer, to gain an overall increase in speed of up to 75% of the original version. This improvement in speed allows us to view as a realistic model the 3-stage processing presented in Figure 1.1 in which a continuous speech, phoneme based recognition approach is extremely flexible and allows the generation of multiple solutions for a further NLP stage. The gain in speed expected from stage 3 as it processes string matching rather than utterance distance computation, with the gain in speed of the stage 2, will certainly make the overall recognition system fast.

5. REFERENCE

[1] A. Sangwan, M.C. Chiranth, H.S. Jamadagni, R. Sah, P. Venkatesha and V. Gaurav, "VAD techniques for real-time speech transmission on the Internet", 5th IEEE International Conference on High Speed Networks and Multimedia Communications, pp. 46 -50, July 2002.
[2] A. S. Rafid, M.H. Shawn, R.S. Anand and D.M. Carl, "Reducing computational complexity and response latency through the detection of contentless frames", IEEE International Conference on ASSP, Volume: 6, 2000.
[3] Chin-Chen Chang; Fun-Chou Shiue; Tung-Shou Chen; "Tree structured vector quantization with dynamic path search", International Workshops on Parallel Processing, pp: 536 - 541, 21-24 Sept. 1999.
[4] C. Shi-Huang and W. Jhing-Fa, "A wavelet-based voice activity detection algorithm in noisy environments", 9th International Conference on Electronics, Circuits and Systems, vol.3, pp. 995 -998, Sept. 2002.
[5] H. Sakoe, "Two-level DP-matching--A dynamic programming-based pattern matching algorithm for connected word recognition", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 27 no. 6, pp. 588 -595, Dec. 1979.
[6] J.G. Wilpon, L.R. Rabiner, "A modified K-Means clustering algorithm for use in isolated word recognition", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 33, no.3, pp. 587-594, June 1985.
[7] K. Sartipi, K. Kontogiannis, "Component clustering based on maximal association", Proceedings of the Eighth Working Conference on Reverse Engineering, pp. 103 -114, Oct. 2001.
[8] R. Lawrence and J. Biing-Hwang, "Fundamentals of Speech Recognition", Prentice hall signal processing series, Englewood Cliffs New Jersey 07632, pp. 395, 1993.
[9] Su Keh-Yin and Lee Chin-Hui, "Robustness and discrimination oriented speech recognition using weighted HMM and subspace projection approaches", IEEE International Conference on ASSP, vol. 1, pp. 541-544, April 1999.
[10] S. Nakagawa, "A connected spoken word recognition method by O(n) dynamic programming pattern matching algorithm", IEEE International Conference on ASSP, vol. 8, pp. 296 -299, April 1983.
[11] S. Nakagawa, "Continuous speech recognition methods by Constant Time Delay DP matching and O(n) DP matching", Trans. Committee Speech Research, ASJ, S82-17, 1982.