PARALLEL RECOGNIZER ALGORITHM FOR AUTOMATIC SPEECH RECOGNITION

Akakpo AGBAGO, Caroline BARRIÈRE
Institute for Information Technology, National Research Council of Canada,
Pavillon Lucien Brault, 101 rue St-Jean-Bosco, Gatineau, Québec, J8Y 3G4, Canada
{Akakpo.Agbago, Caroline.Barriere}@nrc-cnrc.gc.ca

1. INTRODUCTION

Research in Automatic Speech Recognition (ASR) has been very intense in recent years with focus given to accuracy and speed issues. To achieve good accuracy, the employed techniques usually rely on heavy computations. Agbago and Barrière [2] earlier defined a Three-Stage Architecture (TSA) framework for ASR composed of (1) pre-processing stage, (2) phoneme recognition stage, and (3) natural language post-processor stage. Within that TSA framework, our present focus is to improve the speed of Stage 2 which looks specifically at the comparison of low level speech units. It is different from several systems that include HMM processes in this Stage (e.g. Shawn's [5]). We present a new algorithm called Parallel Recognizer that is 320 times faster than a standard Two-Level Dynamic Programming (TLDP) [3]. In comparison, working on speed at low-level, Nkagawa [4] got a reduction factor of 4 to 6 the time needed to compute local distances in the improved DP algorithm of Sakoe [6].

2. PARALLEL RECOGNIZER

A Knowledge Base of Reference Phonemes (KBRP) provides English speakers' phoneme models. To identify the phonemes which are part of a speech segment, our Parallel Recognizer algorithm uses a principle of best-fit in an open competition for all phonemes in the KBRP. It can also embed heuristics that might reduce the search space such as clustering KBRP into a lesser number of models.

2.1 Principle

Parallel Recognizer must take an input speech segment (hereafter T) and recognize it as a sequence of phonemes. These phonemes, already encoded into cepstrums, are selected from the reference base KBRP. In an attempt to reduce the search space during comparison task, heuristics are applied to pre-select some of the units (disqualify unvoiced facing voiced, etc.) that could match parts of segment T. The Parallel Recognizer's originality is that, every reference unit competes and scores a distance value with respect to any possible part of T. This sounds like processing a DP algorithm but it differs by the number of combinations to process that is normally far less than the binomial combinations employed by TLDP.

The algorithm opens a matching competition to fill *segments* of T starting from every frame of it (see section 2.2 for segment definition). It fixes the starting frame but gives the freedom to every competitor reference unit to find the optimal length of T it can best match from that frame (position). Consequently, the number of combination in the comparison is reduced from n^2 to almost n (plus the optimal length computation overhead) as shown in Figure 2.1 where the computation for every node of a matrix space is reduced to a vector space (every row).

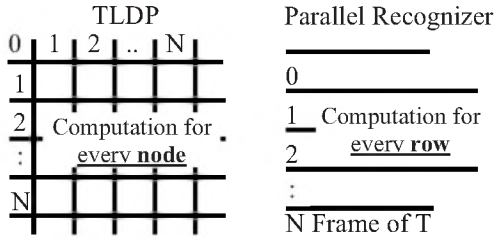

Figure 2.1: Computation combination comparison over the frames of the input utterance T.

The matching process generates a sorted pile for every frame position to the end of T. Using the piles as a lattice of segments we can form numerous chains of segments (hypotheses) by concatenating the best of any pile that comes in a consecutive order. The final result is the best hypothesis of these, based on conditions of our choice: best total chain distortion, fewer/larger number of composing units, etc. The hypotheses can be converted into strings of the ARPABET symbols of the reference speech units.

2.2 Segment Object definition and ordering Principle

Parallel Recognizer relies on an entity we call segment. It has a beginning and an ending position or a length but it encapsulates a third distortion parameter that represents the distance scored from the matching of this segment X in its entire length to a part of another segment Y starting from frame STARTPOS for a length LENGTH. So, we define a segment object (entity) as:
SegO=SegmentObject(X, Y, StartPos, Length, Distortion);

During our comparison process, reference units are successively replacing X and Y is the input segment T. As the resulting pile of segment objects needs sorting, a comparator rule should be defined upon them. The rule could be anything reasonable as how two segments should be ranked. For our experiments, we used the following:

- Order 1 (default): based on starting frame position of segments; used for forward chaining of hypotheses.
- Order 2: based on ending frame of segments to backtrack position to the beginning of a hypothesis.
- Order 3: based on distance to find the closest hypothesis.

The choice of a rule may influence accuracy but not the speed and since our focus is on the latter, we did not evaluate the impact of these different options on the former.

2.2 Strengths and weaknesses

The possibility to choose parameters on which hypotheses are sorted makes Parallel Recognizer very flexible. The segment pile technique *generates plenty and full length strings of phonemes* (no guess of length) as compared to TLDP that *outputs length from 1 unit to a given pre-guessed length L*. This is an advantage for an NLP post processing stage (Stage 3 of TSA) to complete the task. The use of heuristics to reduce search spaces can become weaknesses if they get too heavy in computations just like if naïve methods were used to perform the sorting process.

3. EXPERIMENTS AND RESULTS

In Agbago and Barrière [2], we have presented a Fast Two-Level Dynamic Programming (F-TLDP) approach which already was 5 to 20 times faster than standard TLDP. So, we tested Parallel Recognizer against F-TLDP using an input speech T ("she had your dark suit") and KBRP speech units (a teen phoneme models for every ARPABET symbol) that we derived from TIMIT. Only the execution time of the algorithms was considered excluding the overhead time to compute cepstrums features. Figure 3.1&2 show their contrast with respect to clustering KBRP and the length of the input T. Using the analysis of Figure 3.1, the test for Figure 3.2 is done in the case of no clustering (N=1) and the worst performance of Parallel Recognizer (N=80) compared to F-TLDP. Parallel Recognizer is 4 to 16 times faster than F-TLDP or an overall of 320 times faster than TLDP. The peak in the performance of Parallel Recognizer corresponds to the valley in the performance of F-TLDP [2] which meant better performance of F-TLDP.

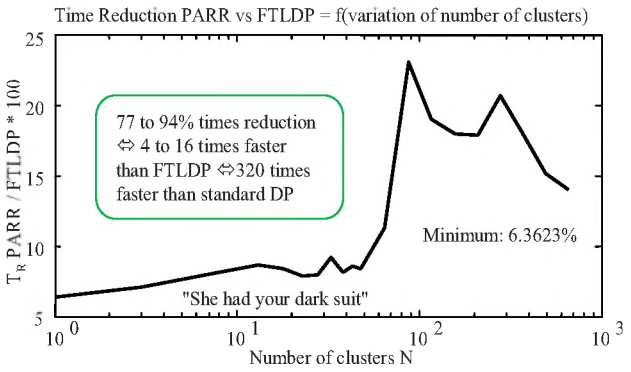

Figure 3.1: Parallel Recognizer vs. F-TLDP as a function of N the number of phoneme clusters in KBRP. PARR = Parallel Recognizer and T_R = time reduction

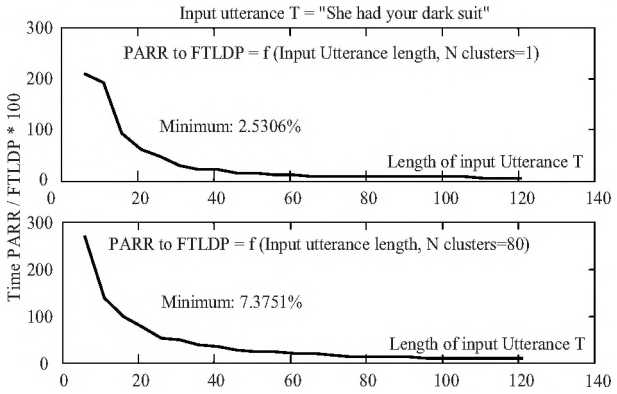

Figure 3.2: Comparison graph between FTLDP and Parallel Recognizer as a function of the length L of the input speech utterance T. PARR = Parallel Recognizer

4. CONCLUSION

With the concept of concurrency (parallel) matching of phonemes in KBRP to an input T using our defined segment entities, we succeeded to convert the matrix computation space of TLDP to a vector space (Figure 2.1). It corresponds to a significant compressing from a computation order of n^2 towards n. The result is 16 times speed increase of Parallel Recognizer over F-TLDP with the advantage of numerous hypotheses output for Stage 3 of the TSA. The speed result is independent of the segment ranking chosen but the choice may have an impact on accuracy.

Agbago has detailed elsewhere [1] that within the Stage 2 (presented here) of the overall Three-Stage Architecture framework, the accuracy of the results depends on an autonomous unit that evaluates the distortion between utterances (segments in this case). In this paper, focus has not been accuracy but speed improvement and it shows quite interesting results.

REFERENCES

[1] Akakpo Agbago, (May 2004) "Investigating Speed Issues In Acoustic-Phonetic Models For Continuous Speech Recognition", Master thesis at the University of Ottawa, SITE.

[2] Akakpo Agbago, Caroline Barriere, (2004) "Fast Two-Level-Dynamic-Programming Algorithm For Speech Recognition", IEEE 2004, May 17-21, Montreal, Canada.

[3] Lawrence R. and Biing-Hwang J., (1993) "Fundamentals of Speech Recognition", Prentice hall signal processing series, Englewood Cliffs New Jersey 07632, pp. 395

[4] Nakagawa S., (1982) "Continuous speech recognition methods by Constant Time Delay DP matching and O(n) DP matching", Trans. Committee Speech Research, ASJ, S82-17.

[5] Rafid A. S., Shawn M.H., Anand R.S. and Carl D.M., (2000), "Reducing computational complexity and response latency through the detection of contentless frames", IEEE-ASSP, Vol.6.

[6] Sakoe H., (1979), "Two-level DP-matching--A dynamic programming-based pattern matching algorithm for connected word recognition", IEEE-ASSP, vol. 27 no. 6, pp. 588 -595.

ON THE STOCHASTIC PROPERTIES OF THE NEURAL ENCODING MECHANISM OF SOUND INTENSITY

Liz C. Chang, Willy Wong
Edward S. Rogers Department of Electrical and Computer Engineering, University of Toronto, Ontario, Canada, M5S 3GS
Institute of Biomaterial and Biomedical Engineering, University of Toronto

1. INTRODUCTION

One of the simplest ways to model the activity in a neural spike train is to use a Poisson process. However, the conventional homogeneous Poisson process (HPP) is not compatible with the results collected from past studies. For example, the pulse-number distribution (PND) extracted from an HPP will follow a Poisson distribution with a mean-to-variance ratio of one. On the other hand, several studies have indicated that the ratio is not unity but is approximately equal to 2 across the dynamic range (e.g. Teich and Khanna, 1985). Additionally many features in the behaviour of real sensory neurons such as rate adaptation, rate-intensity dependence and a dead time in spike activity require that modifications be made to this model.

Based on these concerns, the HPP was modified and extended to give a more realistic stochastic model that expressed quantitatively the properties of auditory neurons. The predictions of the model with respect to the mean-to-variance ratio will be taken as an indication of whether the new model outperforms the conventional HPP-based model.

2. METHOD

In an HPP, the inter-event intervals $\mathbf{t}_{1,2,\ldots}$ which specify (in our case) the interval been spikes are governed by independent exponential random variables with a probability density function $f(t_n = x) = \lambda T e^{-\lambda T x}$ (Leon-Garcia, 1994). λ is a constant representing the spike count within a fixed time window **T**. The exponential distribution was used as a basis from which the new model was developed.

2.1 Firing Rate Modifications

We discarded the fixed spike rate λ and used in its place a rate function $\lambda(L, t)$, where **L** is sound intensity level and **t** is stimulus duration. This function was constructed based on the measurements of rate adaptation (Litvak, *et al.*, 2003) and intensity dependence (Yates, *et al.*, 2000; Smith, 1988). Please see Figure 1. With a non-constant firing rate, the process we have described is known as a non-homogeneous Poisson process (NHPP).

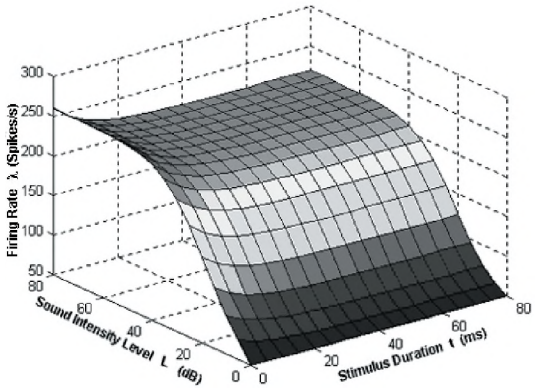

Figure 1. The idealised firing rate behaviour with respect to stimulus duration and sound intensity level for a peripheral neuron.

2.2 Dead Time Modifications

A fixed value τ was used to denote the dead time during which the neuron cannot be activated further. In our model, the inter-spike interval was set equal to the sum of the time value generated by the firing probability function and the dead time τ. This process is known as a dead-time-modified Poisson Process (DTMPP).

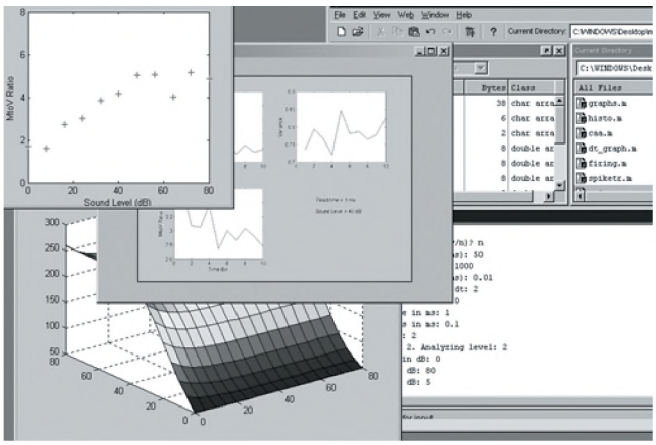

Figure 2. A screen shot of the program in MATLAB.

We implemented our stochastic model within MATLAB. To study the effects of the different components on the mean-variance ratio, a simple command program was written in MATLAB to control the different parameter values (Figure 2). A flowchart illustrating the difference between the various simulations is shown in Figure 3. Each