NOTE TO USERS

This reproduction is the best copy available.

UMI®

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Akakpo AGBAGO

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

Department of Electrical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Investigating Speed Issues in Acoustic-Phonetic Models for Continuous Speech recognition

C. Barrière

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

M. Bouchard

R. Goubran

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

INVESTIGATING SPEED ISSUES IN ACOUSTIC-PHONETIC MODELS FOR CONTINUOUS SPEECH RECOGNITION

Akakpo AGBAGO

A Thesis submitted to the faculty of Graduate Studies and Postdoctoral Studies in partial fulfillment of the requirements for the degree of Master of Applied Science, Electrical Engineering

February 2004

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

© Akakpo AGBAGO, 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-494-01394-X
Our file Notre référence
ISBN: 0-494-01394-X

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Keywords: Three-Stage-Architecture, ParrallelRecognizer, FTLDP, Cepstrum Gain Envelop Profile, ASR, fast

Thesis summary

Automatic Speech Recognition applications face two challenges: accuracy and speed. For good accuracy, Dynamic Programming and Hidden Markov Model algorithms are widely used despite their heavy computational load. To solve the speed problem, this thesis uses a Three-Stage-Architecture (TSA) in which Stage.1 is to enhance and extract features from the input speech signal, Stage.2 does a phonetic-acoustic level recognition to output strings of phonemes to Stage.3 that completes the recognition into valid words using HMM on strings rather than utterances processing.

We designed two algorithms for Stage.2: Fast Two-Level Dynamic Programming (FTLDP) that is 20 times faster than a standard Two-Level DP and ParrallelRecognizer that performs 320 times faster than the standard Two-Level DP. Both algorithms are combined with a heuristic feature called Cepstrum Gain Envelop Profile (CGEP) based Silence Detection to shorten the input speech and clustering to reduce the search space in the reference phonetic models.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to the following individuals:

- My mother, Mrs. Iwuimadi DJODJUNE, whose love, moral help and sound advice helped me make the decision to pursue this degree and follow it through to completion.
- My supervisor, Dr Caroline BARRIERE, who introduced me to possibility of graduate studies, offered me her supervision, and guided me through this research with talent, experience, and a great deal of knowledge of the domain;
- Dr Martin Bouchard, my key advisor in the area of Digital Signal Processing (DSP), who provided me access to the TIMIT database (phonetic-acoustic realizations database);
- Groupe de Recherche en Ingénierie de la Langue (GRIL, www.gril.uottawa.ca), and in particular Dr Lise DUQUETTE, who hosted me in GRIL laboratory, and provided me with equipment for my experiments. The team spirit of the research group made me feel at home and provided me with the support I needed to overcome challenges.
- The memory of my deceased father, M Komadan AGBAGO, who taught me how to find strength in times of difficulty. This allowed me to pull through on numerous occasions.
- To my friend Izabella SOWA who assisted me in editing this thesis.

Abstract

Automatic Speech Recognition (ASR) has been a research area for around forty years. In the last ten years, it has found its way to the scope of business therefore it has gained the interest of commercial applications. It has then become a focus of intensive research and development.

The research community has come up with different solutions answering different challenging problems of this technology. Among these challenges are accuracy and speed issues. In majority, the solutions suggested in the literature have a focus on accuracy and they use two-stage system architecture as follow:

Stage 1. Input speech signal enhancement and feature extracting: the speech signal is processed through a Digital Signal Processing unit to make it robust to noise and eliminate artifacts. It is then converted into an encoded form (speech feature extracting) that may be either spectrums (frequency contents) or cepstrums (LPC analysis) or Ensemble Interval Histogram (EIH, an auditory-based spectral analysis).

Stage 2. A comparison technique is run on the features from the first stage to output solutions that are the conversion of the speech signal into text or commands. These techniques are based on entities matching paradigm and they commonly use, solely or a combination of Hidden Markov Models (HMM), Dynamic Programming (DP) or neural network algorithms. These techniques work with word-based reference model or phonetic-acoustic-based model.

In this thesis we suggest a model that consists of three-stages and called Three-Stage Architecture (TSA). Compared to the description above, this approach has an identical Stage 1 but a lighter Stage 2 and an additional Stage 3 where the use of HMM helps validate the primary recognition solution from Stage 2 into linguistically valid words.

The paradigm of the approach is based on the use of phonetic-acoustic reference language models. In Stage 2, a phonetic level recognition is done (low-level linguistic recognition) to output strings of phonemes and in the third stage, the strings of phonemes are assembled into valid words in the chosen language (our default language used in this thesis is English).

The speed issues for this process are mainly located at Stage 2. Therefore, this thesis focuses on the development of algorithms to improve its performance. Since this stage can be subdivided into 3 components:, (a) data bank of phonetic-acoustic models, (b) input signal, (c) time-sequence comparison algorithm between (a) and (b), we explore performance increase possibilities for each component. In relation to (a) and (b), we respectively exploit the concepts of clustering and silence removal in the processing of the speech signal. In relation to (c), we first suggest a modified version of a standard algorithm called Two-Level Dynamic Programming (TLDP) proposed by Lawrence Rabiner to make it 20 times faster. The modified version is called Fast Two-Level Dynamic Programming (FTLDP). We also design a new and very fast algorithm called ParrallelRecognizer that performs 4 to 16 times faster than FTLDP or 320 times faster than the standard Two-Level DP (20-FTLDP x 16- ParrallelRecognizer = 320).

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER-1

INTRODUCTION

Automatic Speech Recognition (ASR) is a technology that allows a computer to identify the words that a person speaks into a voice-capturing device (microphone or telephone). Its objective is to achieve 100% recognition accuracy of all words intelligibly spoken, to have unlimited vocabulary size, to be insensitive to environmental or transmission channel noise in case of online applications, and to be independent to speaker characteristics (accents and dialects). Ideally, ASR is also expected to perform in real-time.

Of course, this goal is far from being achieved and there are many problems still existing in today's applications. In this chapter, we will first try to define what Speech Recognition is, and then we give a brief history of it. We will then focus on types of systems that exist today, showing their strengths and limitations. This will allow us to position ourselves and illustrate what we wish to contribute. In such a large field, it is important to find a particular issue to explore, and attempt at showing improvement in it.

1.1. Definitions of Speech Recognition on the Web

The definition of speech recognition can be found at several sources from the web and in encyclopedias. We report some of the definitions below. As we report definitions, we ought to keep them in their original wording as they are found at the referred sources. In these definitions, beyond different wording, the main point is that speech recognition is about giving the capability to a computer to understand human language naturally spoken to either convert it into text or react upon it as commands.

a) Speech or voice recognition is the ability of a machine or program to recognize and carry out voice commands or take dictation. In general, speech recognition involves the ability to match a voice pattern against a provided or acquired vocabulary. Usually, a limited vocabulary is provided with a product and the user can record additional words. More sophisticated software has the ability to accept natural speech (meaning speech as we usually speak it rather than carefully-spoken speech) [WEB1].

In a more technical assessment of the definition of speech recognition, we can quote:

b) Speech or voice recognition is the identification of spoken words by a machine. The spoken words are digitized (turned into sequence of numbers) and matched against coded dictionaries in order to identify the words. Most systems must be "trained," requiring samples of all the actual words that will be spoken by the user of the system. The sample words are digitized, stored in the computer and used to match against future words. More sophisticated systems require voice samples, but not of every word. The system uses the voice samples in conjunction with dictionaries of larger vocabularies to match the incoming words. Yet other systems aim to be "speaker-independent", i.e. they will recognize words in their vocabulary from any speaker without training. Another variation is the degree with which systems can cope with connected speech. People tend to run words together, e.g. "next week" becomes "neksweek" (the "t" is dropped). For Speech recognition system to identify words in connected speech it must take into account the way words are modified by the preceding and following words. It has been said (in 1994) that computers will need to be something like 1000 times faster before large vocabulary (a few thousand words), speaker-independent and connected speech voice recognition will be feasible. [WEB2]
c) Automated speech recognition (ASR) is a technology that allows users of information systems to speak entries rather than punching numbers on a keypad. ASR is used primarily to provide information and to forward telephone calls.

Basic ASR systems recognize single-word entries such as yes-or-no responses and spoken numerals. This makes it possible for people to work their way through automated menus without having to enter dozens of numerals manually with no tolerance for error. In a manual-entry situation, a customer might hit the wrong key after having entered 20 or 30 numerals at intervals previously in the menu, and give up rather than call again and start over. ASR virtually eliminates this problem.

Sophisticated ASR systems allow the user to enter direct queries or responses, such as a request for driving directions or the telephone number of a hotel in a particular town. This shortens the menu navigation process by reducing the number of decision points. It also reduces the number of instructions that the user must receive and comprehend.

For institutions that rely heavily on customer service, such as airlines and insurance companies, ASR makes it possible to reduce the number of human call-center employees. Those people can then be trained for other jobs that are more profitable and interesting, such as complaint resolution, customer retention, or sales.

The technology of speech recognition has been around for some time. It is improving, but problems still exist. An ASR system cannot always correctly recognize the input from a person who speaks with a heavy accent or dialect, and it has major problems with people who combine words from two languages by force of habit. Marginal cell-phone connections can cause the system to misinterpret the input. And, although the cost is gradually diminishing, ASR systems are still too expensive for some organizations [WEB4].

1.2. History of Speech Recognition

1.2.1. The pioneer in ARS was a Dog called "Radio Rex" [WEB5]

The first success story in the field of automatic speech recognition took place decades before major research in the area was even considered. It was a dog called "Radio Rex" manufactured in 1920. This dog was built by a toy company and could respond to its name [WEB5] or to almost any sound with sufficient 500-Hz energy. With almost no computation power compared to today's devices, Radio Rex was just a simple electromechanical device using electromagnet principles (see Figure 1.2.1.1).

Figure 1.2.1.1: Radio Rex was just a simple electromechanical device using electromagnet principles to respond to its name

The dog was held down in its house by an electromagnet that was energized through a circuit bridge. The bridge was sensitive to 500 Hz acoustic energy, which could be generated by the energy of the vowel sound of the word "Rex". So, calling the dog by its name "Rex" caused the bridge to vibrate, break the electrical circuit, and allow a spring on which the dog is sitting to push Rex out of its house[1].

1.2.2. U.S. Department of Defense sponsored the first academic research

In late 1940's, in an effort to intercept and process Russian messages, the U.S. Department of Defense sponsored the first academic research in speech recognition to

[1] Other source : Berkeley class literature, http://www.icsi.Berkeley.edu/eecs225d/spr95/

develop an automatic language translator. The first step in developing such a translator was to build a system able to recognize speech. Though the project was a failure, it has created or opened a new path for research. This resulted in the creation and funding of the Speech Understanding Research (SUR) program at Carnegie Mellon University, MIT, and some select commercial institutions by the government. The agency that funded the research later became known as the Defense Advanced Research Project Agency (DARPA, http://www.darpa.mil).

A decade later, in 1952, as funds increased for the research in the domain, Bell Laboratories developed an automatic speech recognition system that successfully identified the digits 0 to 9 spoken to it over the telephone. The success was followed by major finding at MIT where in 1959, a vowel sounds was successfully identified with 93% accuracy followed by a successful test of a 50 words vocabulary in 1966. In the early 1970's, a complete sentence with a limited of range of grammar structures could be recognized by a system called HARPY, developed by the SUR program. However, the system took up 50 contemporary computers computing power to process a recognition channel. The research pointed out for the first time, the three key obstacles in the domain: computing power, speaker independency, and continuous speech[2].

1.2.3. Early Commercial Applications

Speechworks and Dragon Systems were two of the major companies that achieved substantial reductions in the processing power needed by speech recognition systems (see Figure 1.2.3.1), such that the needed processing power equaled the available computing systems around 1993.

[2] Other source : kekoura, http://www.nexus.carleton.ca/~kekoura/history.html

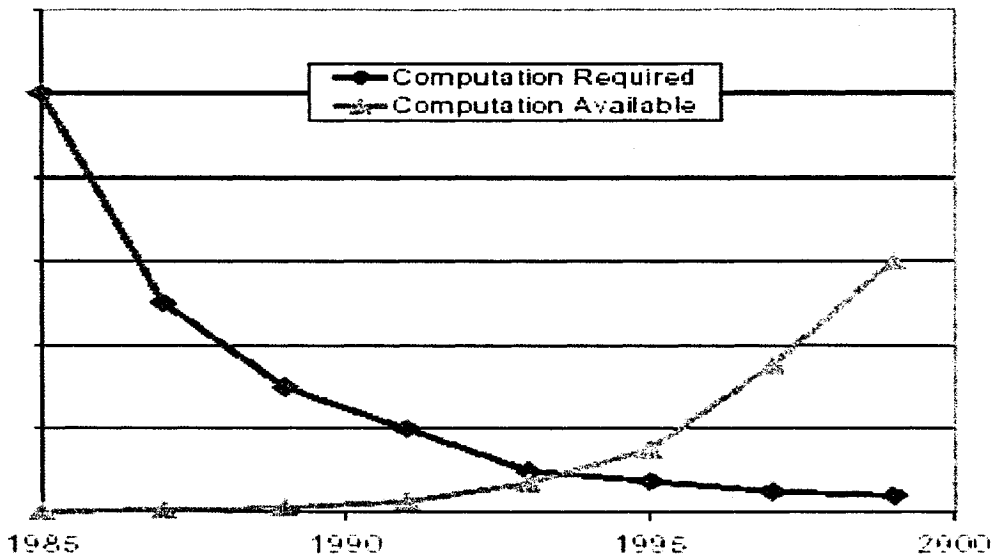


Figure 1.2.3.1 Computation required and Computation available over time [WEB6]

1.3. Today's Opportunities in Automatic Speech Recognition

With the benefit of the advanced technology in telephony with a prediction of 1.1 billion users by 2005[3] [APPENDIX D], and significant progress[4] in speech applications such as Automatic Speech Recognition (ASR) and its counterpart Text-To-Speech (TTS) technologies, a new generation of applications has emerged in Internet and is called *voice portal application*. Voice portal applications allow callers to access e-mail, traffic information, news, stocks and many other leisure and business information. The

[3] TMA Associates, a speech industry analyst firm in Southern California, has forecast $1.8 billion market for the use of speech technology in telecom industry. Automatic Speech Recognition has become reliable enough such that many most pragmatic business managers are interested in (Bell Canada customer services is an example).
Source: TMA Associates, Tarzana, CA, 1999, www.tmaa.com

[4] Recognition rates in some specific business environment show 95-97% accuracy and higher. This is a remarkable achievement as it rivals the accuracy of live human agent.
Source: Voices Portals – The heart of the Voice Web, pp 2,
http://www.stanford.edu/~jmaurer/VoicePortals_WP.pdf

combination of rapid growth in voice applications technology on the web and the increasing growth of end users[5] that are showing big excitement about the products[6], show that it is most likely that research and commercial interest in speech applications will grow. Many commercial markets might find great opportunity in the success of voice applications. Some of them are:

- Call Center Automation
- Enterprise Customer services systems
- Widespread Internet access
- Mobile Internet
- Applications for the Disabled
- Security-voice verification

As ASR technology is improving, some of the opportunities that attract businesses are:

- Increased Reach of consumers: Voice portals will extend enterprise data and consumer information to all mobile and landline phones given opportunity to consumers without computers to access the Internet, to do so via their traditional phone. With the extension of Internet to wireless PCs, digital mobile phones, and analog mobile phones, the reach to the consumers will be even larger.
- Use of Mobile Agent Features: Voice portals will help design devices that will facilitate hands-free operation, and thanks to wireless networks, ubiquitous access to enterprise data and services to mobiles consumers be facilitated.
- User-friendly and Natural Interface: no matter how small and easy to use portable devices become, the handling of the dialog through a keypad and a screen cannot compete with a headset (microphone for ASR, and earphone for TTS). Users will

[5] The Kelsey Group forecasts that by 2005 there will be over 128 million voice portal users with related revenues of $12.3B.
Source: Datacomm Research Company, http://www.datacommresearch.com/old/mvppr.html

[6] Recent research conducted by Nuance Communications and Evans Research shows that 87% of the users of speech systems are satisfied with their interaction, often preferring the speech systems to touch tone navigation menus (DTMF) or even human operators. Instead of wading through menus and submenus only to listen to a monotone, pre-recorded message, these first-time users are impressed by the efficiency and quality of the seemingly human-interaction they find.
Source: 2000 Speech User Scorecard, Nuance Communications.

never be fully satisfied with the tiny keypads and screens found on mobile handsets. This voice interface is most natural, fast and less tiring way of manipulating data.

- Services for Disabled Users: Voice access of web content greatly simplifies accessing the Internet for disabled people without the use of their limbs.
- New winning horizon: developers have in speech recognition, new fields in creating new winning applications.
- Reduces Cost of Operating Phone-based Applications: Voice menus have significant advantages over traditional phone menu systems.
- Complements Other Telecommunications Trends: Increased voice portal traffic can help carriers achieve scale for Voice over IP deployments.

1.4. Types of systems: evaluation of strength and limitations

The most important and widely quoted performance metric for speech recognition systems seems to be accuracy [2]. In many cases, the high accuracy evaluations are achievable with acceptable speed performance if the designer can pay a price for increase in computational power and increase in storage. Recent studies show the increase interest of consumers in speech applications over Internet as we showed in the previous section. In this case, telecommunication applications commonly deal with real-time process running on small devices where electrical power consumption, computational power and storage are difficult if not impossible to get in big scale.

A big latency in response to input speech is also unacceptable in telecommunications as it causes customers dissatisfaction, therefore loss of interest of service providers. So, if success in speech applications is most likely to come from the orientation of the research field toward applications in communication systems, speed will become the main issue to deal with. If we look far ahead in telecommunication market, we can assume that multimedia applications carried over wireless systems and online services will embed

speech recognition applications (called voice portals) and video applications. These two features have huge computation, storage and bandwidth consumption.

On these bases, the metric system for speech recognition systems appreciation should not mainly rely on accuracy but:

a) Computational complexity (CPU usage)
b) And response latency.

This change in the evaluation parameter is also justified by the study done by Richard P. Lippmann in 1997 on the comparison of speech recognition by machines and humans [12]. The study is to determine how far recent dramatic advances in technology have progressed towards the goal of human-like performance, Automatic Speech Recognition (ASR) versus Human Speech Recognition (HSR).

As Amit Juneja [19] referred to it in his thesis, Lippmann's comparison is still valid today given only incremental improvements to HMM based ASR have been made since that time. Lippmann's results suggest that the human–machine performance gap can be reduced by basic research on improving low-level acoustic-phonetic modeling, on improving robustness with noise and channel variability, and on more accurately modeling spontaneous speech.

Lippmann showed that humans perform approximately 3 to 80 times better than machines using word error rate (WER) as the performance measure. He noted that the performance of ASR on a continuous speech corpus drops from 3.6% WER to 17% WER when the grammar information is not used (equivalent to all the words in the corpus have equal probability) while the HSR performance's drop was from 0.1% to 2%.

The conclusion is that ASR is much more dependent on high-level language information than HSR. This result implies that ASR systems need not to waste time on low-level processing. According to Lippmann, human spectrogram reading performance is close to ASR performance although it is not as good as HSR.

So the acoustic-phonetic approach, inspired partially from spectrogram reading, is a valid option for ASR. Also, it is known that in continuous speech, people tend to slur one word into another. This situation makes the automatic delimitation of big speech units such as words difficult in ASR. Therefore recognition engines for continuous speech and large vocabulary systems work on smaller speech units so that each word or sentence becomes a sequence of smaller units and thus reducing the slurring effect.

The consequence of this smaller units approach is huge on speed dimension. Most recognition systems use techniques (conventional DP, HMM, classifiers, etc.) whose complexity and processing speed depends on the length in speech unit (phone, phoneme, word, etc.) of utterances to compare. Obviously the length of an input speech utterance expressed in phoneme is several tens times higher than if it was expressed in words. So, speed is going to be the most important dimension on which speech recognizers will be evaluated; the accuracy factor has already been relatively improved.

Looking furthermore in a comparison to human capabilities, Fletcher perceptual experiments [20] bring in more evidence that humans rely highly on accurate phoneme level recognition. In fact, he found a recognition error of 1.5% on clean speech over the phones in nonsense consonant-vowel-consonant (CVC) syllables though the machine performance is unknown on nonsense CVC syllables. It also showed that the probability of correct recognition for a syllable is the product of the probability of correct recognition of the constituent phones.

When Allen [21, 22] reviewed Fletcher's report, he inferred that individual phones must be correctly recognized for a syllable to be recognized correctly. Allen's interesting conclusion is that it is unlikely that context is used in the early stages of human speech recognition and that *the focus in ASR research must be done on phone recognition.*

1.5. Where we stand

Based on Lippmann's conclusions, ASR depends on high level language information and the acoustic-phonetic approach is a valid option for ASR and Allen's conclusion is that ASR research must be done at phone level. This fact is good for speaker independent recognition system because, by going to linguistic lower level the effect of context is weaker. These reasons justify our choice to work on an ASR system that uses the acoustic-phonetic approach and that does recognition on two levels: a low-level that processes phonetic features and high-level that processes the high-level language information. ***Our objective is to speed up the low-level.***

The time of restriction of speech recognition applications to isolated or connected word or digits recognitions has past. As shown in the section on opportunities, the market tends to orient toward continuous or natural speech cases unless the application scope is reduced to commands or voice recognition. For this reason, we choose to work on continuous speech recognition where our choice of acoustic-phonetic approach might help reduce the problem of slurring words into others by speakers.

Obviously in stage 1, the time requirement to encode input speech data and vocabulary templates into features (spectrums, LPC, cepstrums, time derivatives of cepstrums, etc.) cannot be largely changed except the effect of an efficient implementation. The parts where some work can be done to gain on time are the techniques of speech/non-speech classification, DP matching and the validity of the outputs with respect to valid chains of vocabulary word.

Multimedia applications are abundantly used in mobile environments nowadays. This implies that a speech recognition application might run in a rapidly changing background noise environment. Therefore the system must be able to readjust its heuristic thresholds easily either automatically or manually.

1.6. Goals and approaches

Throughout the research and experimentations, we try to find a way to implement a fast, continuous and speaker independent speech recognizer. Our objectives are:

Goal.1 Based on the recommendation made by Allen [21, 22] and to permit speaker independent recognition system that also makes the setup of the recognizer easy prior to its use, we orient our effort toward the design of an acoustic-phonetic approach ASR [33]. To achieve this goal, we should build a language knowledge base that is necessary for a recognizer.

Goal.2 Find a way of doing the continuous speech recognition fast. This is the speed issue that is the main objective in this thesis (reduce the computation power)

As we said it many times, literature on ASR shows that the work of researchers in the recent years has focused mainly on increasing two numbers: the accuracy[7] and processing speed of speech recognition though accuracy has always been the main focus. 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, 36, and 37] 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 phoneme unit level (phoneme, diphone, etc.) since the progress step becomes small, thus increases the number of iterations. Despite that factor, our objective is to do recognition with an acoustic-phonetic approach (Goal.1) 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. In fact, there exist standard databases that provide the a priori observation

[7] Recognition rates in some specific business environment show 95-97% accuracy and higher. *The heart of the Voice Web, pp 2, http://www.stanford.edu/~jmaurer/VoicePortals_WP.pdf*

probability values to all the phonemes known in a language (English for example). HMM techniques use this information to build chains of valid phonemes or words.

Also, studies have shown that a speech recognition process cannot be completely and accurately done at the signal processing level [34] due to the fact that we cannot yet distinctively cluster vowel sounds. This is the same observation made by Lippmann [12]. The implication of this insufficiency is that we need, at the signal processing level, to produce several alternatives solutions to a higher level that might be a Natural Language Processing Unit. Therefore, we propose a Three-Stage Architecture (TSA) shown in Figure 3.3.1 to achieve speaker-independent, continuous speech recognition. 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 [36], we suggest it's moved in this third stage to run on strings (simple matching of phonemes) rather than on utterances (complex distances).

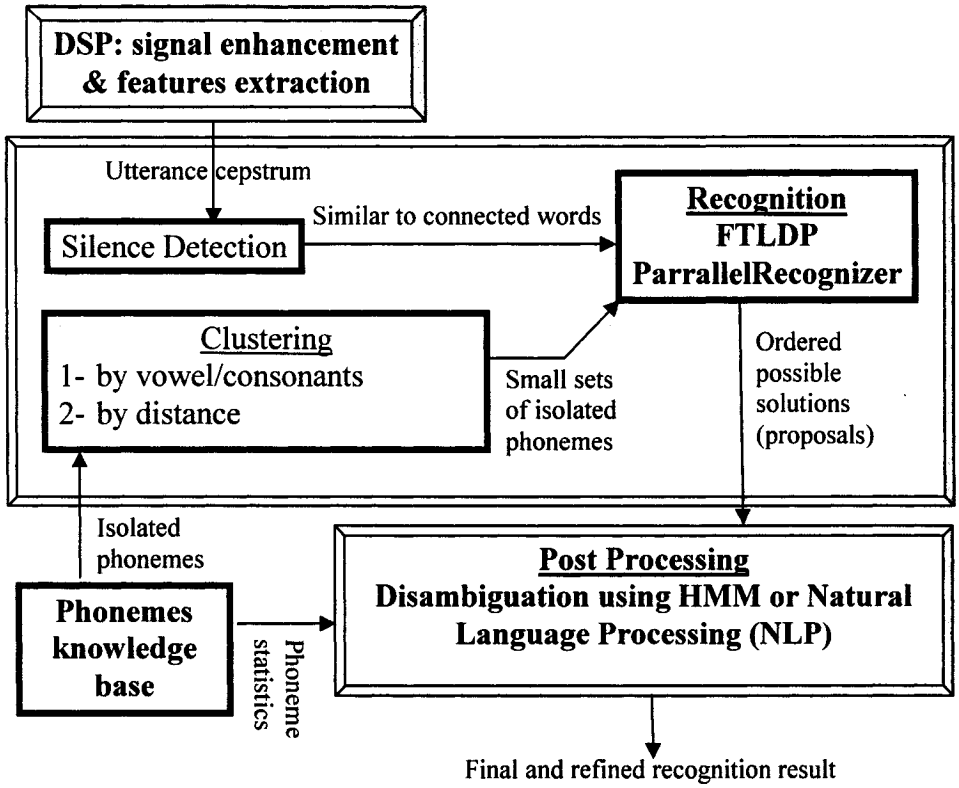


Figure 1.6.1: Block diagram for a fast speech recognition

As with the suggested approach of Figure 1.6.1 the processing of the strings of phonemes found at Stage 2 has to be done at Stage 3, we need to find algorithms capable of outputting several solutions (strings of phonemes) at Stage 2. This gives a larger degree of freedom to the NLP module in Stage 3 (possibly using HMM) to generate valid words. The constraints lead us to direct our design toward multiple output algorithms [38]. In this perspective, we worked on two recognition algorithms as solutions that are as follow:

Solution.1 We have modified a standard Two-Level Dynamic Programming Algorithm (TLDP) to make it very fast using Clustering Approach to reduce the reference search space and Silence Detection to reduce the length of the utterance to recognize.

Solution.2 Working on a different approach, we design an algorithm that we call ParrallelRecognizer that works on the basis of best-fit principle. The

principle is a position competition open to all phonemes in the knowledge base unless we add a heuristic layer to the process to narrow down the number of competitor phonemes.

To achieve our goals 1&2 using solutions 1&2, the ASR system needs to have a language knowledge base. In our case of acoustic-phonetic paradigm, this reference language database is the set of all the phonetic units in English.

1.7. Presentation of the thesis

The remaining of this thesis will be as follow. Since we wish to explore different strategic places for speed improvement in a phonetic-based continuous speech recognition system, we will present each effort independently with its related pertinent literature review.

In our Three-Stage Architecture (TSA) shown in Figure 1.6.1, we specifically wish to work on speed improvement of stage 2. Subdividing stage 2 into 3 components, we have (a) a data bank of phonetic units, (b) an input signal, and (c) a comparison approach of the first two. Our proposed system will incorporate different improvements, but it is important that we first show them independently.

Therefore, Chapter 2 will focus on the databank of phonetic units. Chapter 3 will focus on different heuristics for reducing the input signal. Chapter 4, the heart of this thesis, will present 2 different comparison algorithms, first a FTLDP algorithm, and second a ParallelRecognizer .

We'll show in Chapter 5 some experiments and results. Chapter 6 will conclude and present directions for future work.

CHAPTER-2

ORGANIZING A DATABASE OF PHONEMES

The speech recognition techniques or algorithms that we present in this thesis assume the existence of an acoustic-phonetic database that is the reference source of language knowledge for a recognizer application. The quality of this resource is important for the performance of the comparison algorithms in term of accuracy of their output results.

As shown in Figure 1.6.1 of chapter 1, this module may have several roles in the Three-Stage Architecture (TSA) that we propose. It should provide models individually for every phoneme in English language (our arbitrary choice of language). The models are required for the stage 2 of the architecture and also for the stage 3. In Stage.2 where the techniques are supposed to operate only at a low linguistic level, the models provided by this resource should be available in sampled signal form. In Stage.3, the models should rather have a statistical form to serve in HMM processes.

In this chapter, we present details of **Goal.1** that is to provide a Reference Phoneme Knowledge Base (RPKB) that will be the Language Knowledge Base for the ASR system. With RPKB, we expect to make the recognizer application easy to use and speaker-independent. Next in this chapter, we describe how a complete phonetic utterance database has been created from TIMIT database. In this phase, we will create only utterance models that we intend to use in the Stage.2 of the TSA.

The TIMIT database that we will use to create the RPKB will generate a huge amount of realization of each language unit. Therefore we will use clustering techniques to refine and narrow down the results. We will also be reported works that have been done similar to this clustering module by other authors.

2.1. Models for every phoneme in English language

There are approximately 50 phonemes in English language. How can they be modeled to provide good representation of all the accents in English? How to create this resource to meet the expectation to serve ASR systems for good speed and accuracy? These are some of the challenging questions that we will try to answer in this section. The simplest approach is to record the phoneme but it will be wrong if they are not spoken in context. In case that they are spoken in context, thus in sentences, the challenge is to extract them with precision. This is known as segmentation problem. Another fundamental question is how to store the models? Should silence precedes and terminates a phoneme model or not?

Adding silence to the beginning and to the end of a phoneme utterance will increase the length of the elements, thus the computational load during matching operations. With this justification, we can opt to record only the part of a phoneme that corresponds to a significant energy level. This brings up the challenge of a clear determination of the start and the end point if the phoneme was recorded in context and thus it is in the midst of other phonemes.

In real life, people utter a given (English) phoneme in different ways (sounds); we call this dialect accents. There are many dialects around the world for English language alone. Also, in addition to the dialects, there are differences in the uttering speed of a given phoneme depending on the context. Therefore, ideally we want the Knowledge Base to take into account, all of these factors.

To resolve the challenge of the variability in the way that a given phoneme is uttered, we might want to have multiple realizations of the phoneme in the Knowledge Base. These realizations should correspond to the differences we have described above. In a technical expression, we should build clusters of phonemes. In theory, the larger and well-populated clusters we dispose in the database, the better is the probability to make good matches during matching processes. This is a way that we can achieve the goal of speaker-independent that we have set to begin.

The challenge in having clusters of phonemes, and more over, large clusters, is the overlapping problem. The improvement of the recognition accuracy factor will be made easy if ideally we can get the clusters clearly disjoint; this means that there is no overlapping. Unfortunately, this is not the case as Rabiner shows it in the Figure 2.1.1 in his book [25].

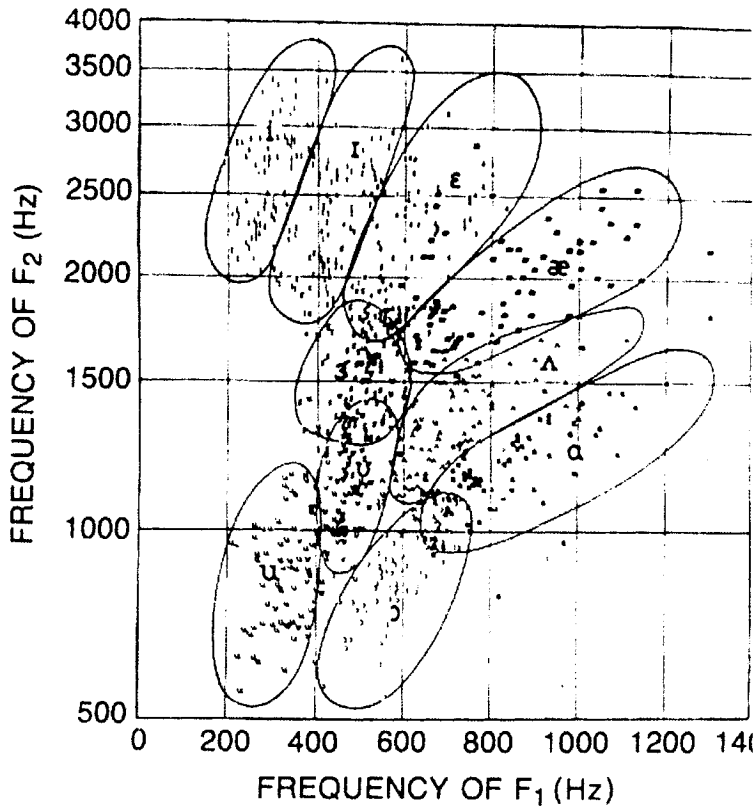


Figure 2.1.1: Measured frequencies of first and second formants (F1 & F2) for a wide range of talkers for several vowels (after Pertson & Braner [24])

We also have to notice that the figure just shows the case of vowels. Vowels are known to have formants (voiced sounds due to vocal cords vibrations) but some consonants are models by just white noise (unvoiced). The confusion or the overlap of clusters should be higher in the case of such consonants.

So the question is to decide on what to do:

- Should we design very tight (very concentrated) clusters of phonemes? In this case, it is equivalent to say that we select the centroid of each cluster in Figure 2.1.1 and build a circle of very small diameter around it. This way, it is most likely that the clusters should not overlap. However, should this configuration represent well all the varieties we have talked about?
- Should we design relatively large clusters of phonemes? In this case, we will have to process a disambiguation at the recognition level. A NLP module at Stage.3 in the TSA that we propose can do the disambiguation. How much effective are the disambiguation tools that we might have and how much expensive would they be in term of computational load?

Some of these questions found answers in the resource provided by TIMIT database that we will present in the next section. TIMIT is a production of the National Institute of Standards and Technology (NIST).

2.2. TIMIT phonetic corpus

To use the acoustic-phonetic approach, we need to have a database of sample phoneme utterances of good quality and a wide variety of them to reinforce the speaker independency of the ASR system.

For this part, we used TIMIT read speech corpus to generate the knowledge base. Hereafter, we will refer to this speech corpus with only its name TIMIT for short. TIMIT is an acoustic-phonetic read speech corpus that was published in October 1990. It was designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. Under the sponsorship of the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO), TIMIT was recorded by Texas Instruments (TI), transcribed at Massachusetts Institute of Technology (MIT), has been maintained,

verified, and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST), and with the joint effort of Stanford Research Institute (SRI).

TIMIT contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers (males and females) from 8 major dialect regions of the United States (see Appendix C). There have been 3 types of sentences used to build the corpus:

- SA sentences: 3960 sentences consisting of 2 dialects "shibboleth" designed at SRI to expose the dialectal variants;
- SX sentences: 450 phonetically-compact sentences designed at MIT to provide a good coverage of pairs of phones with extra occurrences of phonetic contexts thought to be either difficult or of particular interest;
- SI sentences: 1890 phonetically diverse sentences selected at TI to add diversity in sentence types and phonetic contexts (variety of allophonic contexts).

The corpus is composed of the utterance of the sentences, their segmentation and labeling in phonemes and in words. For further details about the architecture organization of the corpus, see the Appendix C.

The labeling of the utterances is done using ARPABET, the standard transcription of the English phonetic symbols into ASCII letters extended with some special symbols to handle particular cases such as stops. The labeled utterances are in the transcription part of the corpus that also encloses a lexicon of words transcribed into phonemes.

The transcription module can help in ASR by providing reference samples for words and phonemes. In this case, the ASR application might use words as reference speech units (pattern recognition paradigm) or phonemes (phonetic-acoustic paradigm) to decode unknown utterances. Once the decoding is done in term of speech units, the lexicon might be employed to get the appropriate words corresponding to the string of phonemes of the word.

The lexicon is a resource useful for a Text-To-Speech (TTS), in which case, the application converts words into phonetic segments and then matches the found phonemes to reference utterances.

The next section of this chapter will provide the detail of the set of the phonetic symbols used in TIMIT.

2.3. Phonemic and phonetic symbols in TIMIT

All of the phonemic and phonetic symbols used in the TIMIT lexicon and in the phonetic transcriptions are reported in this section and can found in Table 2.3.1. The set includes the stress markers {1, 2} found only in the lexicon and the following symbols, which occur only in the transcriptions:

a) The closure intervals of stops, which are distinguished from the stop release. The closure symbols for the stops b, d, g, p, t, k are bcl, dcl, gcl, pcl, tck, kcl, respectively. The closure portions of jh and ch, are dcl and tcl.

b) Allophones that do not occur in TIMIT lexicon. The use of a given allophone may be dependent on the speaker, dialect, speaking rate, and phonemic context, among other factors. Since the use of these allophones is difficult to predict, they have not been used in the phonemic transcriptions in the lexicon.

- flap dx, such as in words "muddy" or "dirty"
- nasal flap nx, as in "winner"
- glottal stop q, which may be an allophone of t, or may mark an initial vowel or a vowel-vowel boundary
- voiced-h hv, a voiced allophone of h, typically found intervocalically
- fronted-u ux, allophone of uw, typically found in alveolar context
- devoiced-schwa ax-h, very short, devoiced vowel, typically occurring for reduced vowels surrounded by voiceless consonants

c) Other symbols include two types of silence; pau, marking a pause, and epi, denoting epenthetic silence which is often found between a fricative and a semivowel or nasal, as in "slow", and h#, used to mark the silence and/or non-speech events found at the beginning and end of the signal.

Table 2.3.1: Phonetic symbols used in TIMIT corpus (Source: TIMIT)

POSSIBLE PHONETIC		
Symbol	Example	Word Transcription
stops:		
b	bee	bcl b iy
d	day	dcl d ey
g	gay	gcl g ey
p	pea	pcl p iy
t	tea	tcl t iy
k	key	kcl k iy
dx	muddy, dirty	m ah dx iy, dcl d er dx iy
q	bat	bcl b ae q
Affricates:		
jh	joke	dcl jh ow kcl k
ch	choke	tcl ch ow kcl k
Fricatives:		
s	sea	s iy
sh	she	sh iy
z	zone	z ow n
zh	azure	ae zh er
f	fin	f ih n
th	thin	th ih n
v	van	v ae n

dh	then	dh e n
nasals:		
m	mom	m aa m
n	noon	n uw n
ng	sing	s ih ng
em	bottom	b aa tcl t em
en	button	b ah q en
eng	washington	w aa sh eng tcl t ax n
nx	winner	w ih nx axr
semivowels and		
glides:		
l	lay	l ey
r	ray	r ey
w	way	w ey
y	yacht	y aa tcl t
hh	hay	hh ey
hv	ahead	ax hv eh dcl d
el	bottle	bcl b aa tcl t el
vowels:		
iy	beet	bcl b iy tcl t
ih	bit	bcl b ih tcl t
eh	bet	bcl b eh tcl t
ey	bait	bcl b ey tcl t
ae	bat	bcl b ae tcl t
aa	bott	bcl b aa tcl t
aw	bout	bcl b aw tcl t
ay	bite	bcl b ay tcl t
ah	but	bcl b ah tcl t

ao	bought	bcl b ao tcl t
oy	boy	bcl b oy
ow	boat	bcl b ow tcl t
uh	book	bcl b uh kcl k
uw	boot	bcl b uw tcl t
ux	toot	tcl t ux tcl t
er	bird	bcl b er dcl d
ax	about	ax bcl b aw tcl t
ix	debit	dcl d eh bcl b ix tcl t
axr	butter	bcl b ah dx axr
ax-h	suspect	s ax-h s pcl p eh kcl k tcl t
others:		
Symbol	**description**	
pau	pause	
epi	epenthetic silence	
sil	begin/end marker (non-speech events)	
1	primary stress marker	
2	secondary stress marker	

The Table 2.3.1 gives the summary of all the symbols defined for the phonetic units in the English language according to ARPABET standards. The symbols are unique but the realizations of the phonetic sound associated to them are not exactly the same. Therefore a symbol will represent a class of sounds similar enough to be grouped under a unique symbol that we call clusters. The next section will describe how we derive clusters of phonemes from TIMIT database.

2.4. Derivation of phonemic and phonetic clusters from TIMIT

The format used in each TIMIT corpus to represent the speech units (phonemes or words) is inappropriate or is not ready to be for use in a recognizer application though the segmentation and the labeling that have been done in the corpus are the hardest work in the design of the database. Refer to Appendix C for detail. In fact, at the phonemes level, there are hundreds of thousands of samples of phonemes available in the corpus but scattered all over the corpus.

Another problem with TIMIT corpus is the non-standard format of the wave files that contain the utterances. For example, a file SA1.WAV which contains the audio samples for the corresponding labeling string of phoneme symbols and their segmentation bounds in SA1.PHN cannot be read as a standard wave file. The reason is that, the *.wav file has a private TIMIT type of header padded with the audio samples. Therefore to extract the samples, one must be able to decode the header first and then read out the utterance samples.

What we did is to fetch wherever they exist in the corpus, every utterance segment that is a realization of a given phoneme, gather all of them and cluster them under the ARPABET symbol of the phoneme. We use Wilpon and Rabiner's clustering technique known as "A modified K-Means clustering algorithm" [6] although different clustering methods [7] could be used. The regrouping operation is distinctively done for male realizations and female realizations. The result of the lengthy operation is the construction of around 61 clusters containing each, several thousands of realizations of the underlined phoneme for the cluster.

Each cluster is organized as a directory (folder) under the underlined phoneme symbol. The directory contains all of the realization of the phoneme as wave files as shown in Figure 2.4.1. This time, the wave files as formatted in the standard form (see Appendix B) so that any multimedia file editor utility based on WAVE standards would be able to read the utterances.

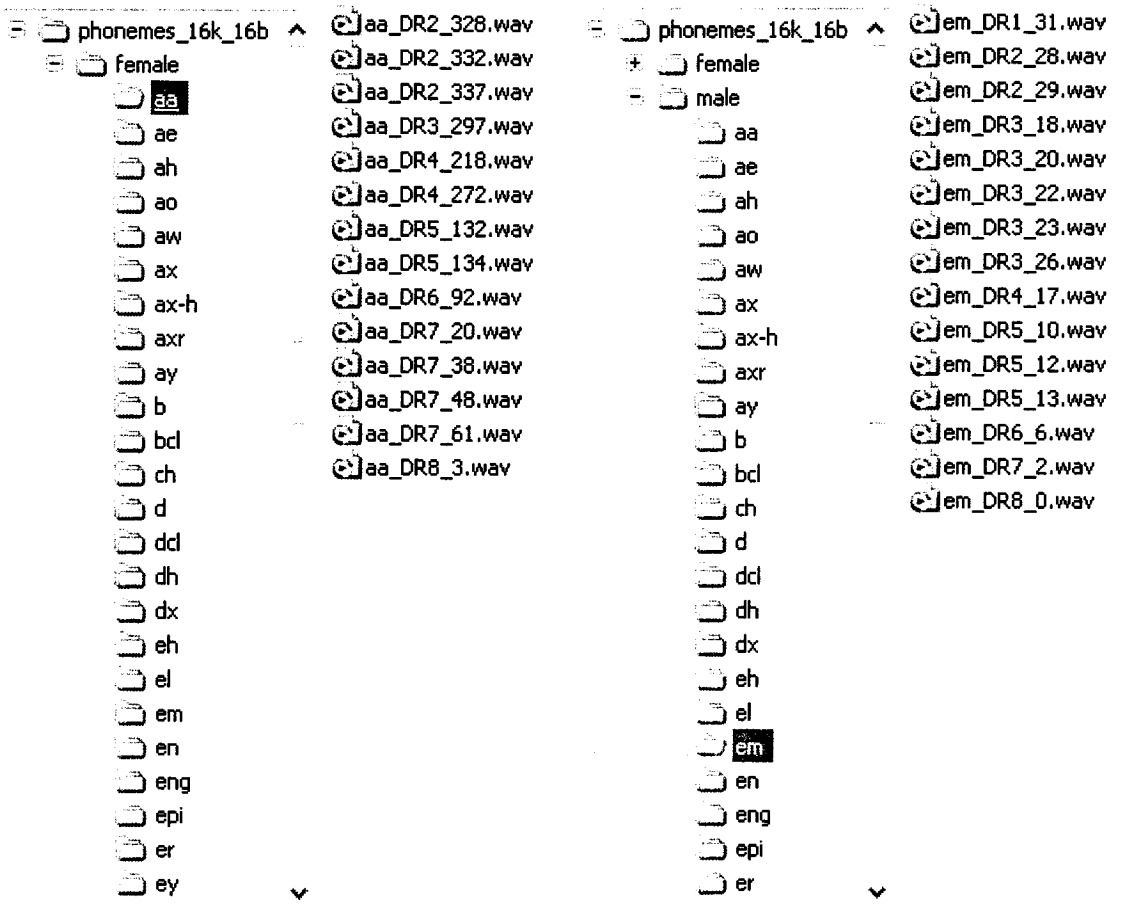


Figure 2.4.1: Phoneme based restructuring of TIMIT database. The phonemes are represented in their ARPABET symbols

The conversion of TIMIT speech utterance database into clusters of phonemes was not done without raising some problems. The next sections will present these major problems and possible solutions for them.

2.4.1. Problems arising from the derivation of clusters from TIMIT

Pro.1. The restructuring operation results in several thousands of realizations for each utterance. As said earlier, the realizations were made with different types of sentences (SA, SI, and SX) to exhibit particular characteristics of the same unit in

different circumstances. The fact is that, it would be impossible in terms of storage (memory), processing time and efficiency, to use a whole cluster as a model in the recognizer. It needs truncating, but how to select the members of the reduced representative sub cluster?

Pro.2. The realization of the utterances were made with a sampling frequency F_s = 16 kHz. This is suitable for online telephone applications whose frequencies are 8 kHz or 16 kHz. The question is what happens in the case of workstation applications where multimedia applications use sampling frequencies ranging from 11.025 kHz to 44.1 kHz for CD quality?

2.4.2. Solutions for the clusters derivation problems

Sol.1. To the problem Pro.1, we use clustering technique guided with progressively tightened constraints (intra-cluster distortion and population) to single out a compact (dense) subset of realizations small enough in size (e.g. 15 realizations maximum) to be easily handled by a recognizer application. The algorithm is applied to every phoneme cluster to generate a representative subset.

How is phonemes similarity evaluated?

We evaluated the distortion among phoneme realizations using cepstral coefficient features. Therefore the evaluation of the similarity is based on distance measurement between the acoustic cepstrums. Other methods use different features such as formant extraction. In the case of formant extraction, the computation requires voiced/unvoiced segmentation, the location of vowel core and the extraction of the formant values. This means, for the recognizer application, the computation of different other features than the cepstrums that are already available.

Sol.2. To the problem Pro.2, the API of the recognizer should provide a way to convert (resample up or down) either the Language Knowledge Base to the user's

preferred sampling rate (one time operation at startup of the application as advantage) or the continuous speech of the user to the Language Knowledge Base signal characteristics (real time applications).

2.5. Foreseen speed improvements

We said that the clusters shown in Figure 2.4.1 are generated with very small diameters (highly concentrated). However, we can see that there is a very good mixture of dialects in the representative selection (see Figure 2.4.1). The resulting mixture of dialects will help provide to an ASR system that uses this clustering module, a large variety of accents, thus a good speaker-independency capability in the recognition tasks.

The use of the results (clusters) of this technique is also expected to reduce the possibility of having clusters overlapping each other. As we mentioned it earlier, it is an important factor to consider in the task of making the disambiguation of the recognition results simple and light.

By finding the first few central realizations of phonemes, the results of this clustering module are expected to drastically reduce the search space while comparing utterances during recognition processes. From several thousands, this module reduces the phonetic models to few hundreds. Such reduction makes a big difference for a DP algorithm whose computation cost is a power of the size of the search space.

2.6. Other usage of the data bank

This functionality of the language resource should be able to provide statistics and probabilities associated to the language units (phoneme or word possibly). This feature is important for the use of HMM modules in the third stage of the model that we suggest. The HMM module requires the a priori probabilities of language units preceding or following each other. This information is not provided in TIMIT database but is available

as a product of research; we have not come across it yet but we do believe it is available possibly at the Center for Spoken Language Understanding (CSLU).

The phoneme database resulting form the design of this clustering module may also be of good interest for the scientific community. In fact, from this work, we dispose a large bank of phoneme utterances from eight different dialects in English, grouped sex and their ARPABET symbols. Depending on the task, this phoneme bank can be used with in data mining techniques to train systems. It can also serve as a resource for acoustic-phonetic scientific studies to find interrelationship between the phonemes, group of phonemes, statistic features, etc.

2.7. Related work

In the International Congress of Phonetic Science 1999 (ICPhS99), Philippe Boula de Mareüil presented a technique [29] to help cluster automatically phonemes from 3 languages (French, English, German, Spanish, called IDEAL corpus). The motivation for his work is that he too believes that it is possible to replace several phonetic recognition systems by a single, common system for all the languages as inferred in [30] by Corredor-Ardoy.

As we did in our case, he also used cepstral coefficients to evaluate the similarities or the distortions between the phonemes. Unlike our method that uses a progressive cluster tightening method, Mareüil's method uses a hierarchical clustering algorithm. Doing similar work, the hierarchical clustering algorithm was also used by Berkling [31] on six languages with the use of similarity measure between acoustic vectors, Köhler [32] on three languages with the use of acoustic likelihood. The difference between hierarchical clustering algorithm and the progressive cluster tightening method reside only in the concept. Using similarity measure the former algorithm starts with every phoneme as a cluster and progressively group clusters of maximum likelihood. So the concept is from numerous small clusters to few big clusters. Using distance, the latter does the inverse by starting with one cluster containing every phoneme and progressively split it into small

clusters. Both algorithms can do the job intended in this thesis, in fact any clustering algorithm can. On the comparison point, we have no data for these clustering algorithms because our task is not to design a clustering technique. Also, we will show that the clustering task has just a fix computational cost in our solutions. The clustering is used offline to prepare the RPKB only at the start of the recognizer.

Mareüil designed his phoneme hierarchical clustering algorithm for a multi-lingual purpose. However as his pointed it out, when applied within a mono-lingual framework, the hierarchical clustering algorithm may assist in analyzing phonetic classes, in defining phone sets for speech recognition, and in grouping a large number of Markov models in context. These application domains are the one our method presented in this thesis targets, especially the definition of phone sets for speech recognition.

CHAPTER-3

HEURISTICS AT THE INPUT SIGNAL

The heuristics techniques that we present in this chapter are intended to reduce the cost of either the search space in Reference Phoneme Knowledge Base (RPKB) or the length of the input utterance T to recognize. Certainly, there are many other techniques that can do this task, however we will present only two techniques that we have successfully implemented and tested. They are:

- Cepstrum Gain Envelop Profile (CGEP): this feature can be used as a fast heuristic to find similarity between two utterances.
- Silence detection: this feature can serve as way to make short an input speech utterance to gain time in the comparison process.

The primary goal in this chapter is to exploit as much as possible the speech features (cepstrums) that we dispose for the comparison algorithms in the recognition process to predict or guess the solutions easily. Therefore, we focus on what characteristics we can draw out from the cepstrums that are the speech features we dispose in the case of this thesis.

3.1. Cepstrum Gain Envelop Profile (CGEP)

The Cepstrum Gain Envelop Profile is features that can be easily extracted during the conversion of utterances from raw signal waveform to cepstrums. In fact, in the computation of the cepstrums, we evaluate the autocorrelation of each frame of the speech signal, thus the energy of each frame of the utterance is known through the first coefficient of the autocorrelation. Moving one step further, we can access the values of

the LPC gain features ($G^2 = E_{min}$ = minimum energy) from which the first coefficient (c_0) of the cepstrums for each speech frame is derived using the expression $c_0 = \ln(G^2)$.

Since the frames are normally short enough in time to confer stationary characteristic to the signal, by tracking the cepstrum gain of every frame, we can construct a profile that reflects the envelop of the variation of the gain of LPC features of the cepstrums of the utterance. We call these features CGEP.

This heuristic features are thus a byproduct of the computation of the cepstrums, yet very useful. They need no extra computation except a little overhead for storage and decisions. It is more interesting to realize that CGEP is a feature that is very short in size. In fact, we need to keep only the first cepstrum coefficient per frame. For example, if a speech signal is made of 3000 samples and it is framed into 100 samples, the CGEP would be an array of 30 values only. See Figure 3.1.1 for illustration.

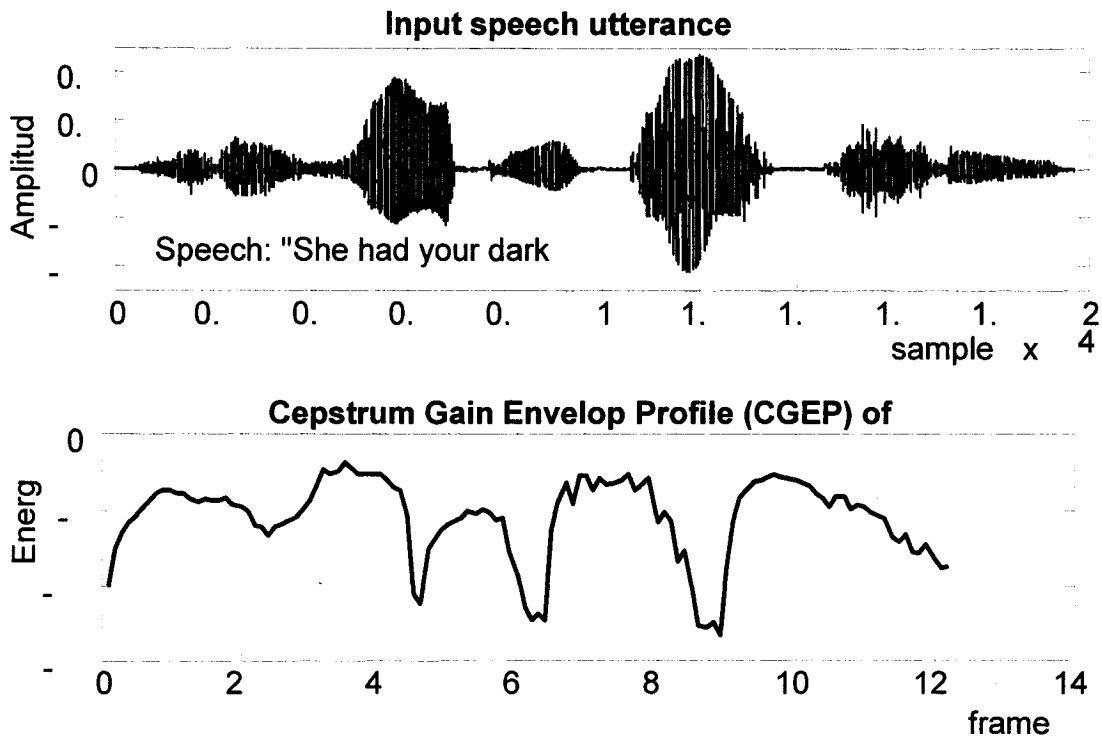


Figure 3.1.1: Cepstrum Gain Envelop Profile for the utterance of the sentence "She_had_your_dark_suit" extracted from TIMIT database.

Possible ways of using CGEP

CGEP is features that can be handled just like any other feature in speech recognition. This means, to find similarity between two utterances X and Y, we can do several things:

- Compute the LPC gain envelops' distortion between X, Y;
- Compare the average of envelops;
- Undertake some statistic evaluations.

Depending on the amount of computation overhead that a system can tolerate, more complicate operations can be done with the CGEP features.

In the case of our thesis, the CGEP features will be used in FTLDP and ParrallelRecognizer algorithm that we propose and that will be detailed in chapter 4. It will be used to test the qualification of language units (phonemes) for a comparison computation.

3.2. Silence detection

Silence detection is a technique that has for objective to delimit and remove parts of a speech utterance that are not relevant to the decoding of the underlined utterance into text. These parts correspond normally to noise, artifacts from voice capturing devices such as microphone, or communication lines. Ideally, the parts to remove should correspond to the segments of the utterance that might not have a matching phoneme in RPKB.

There exist several algorithms to achieve this task [23, 26, 27, 1, and 4]. The algorithms are called Voice Activity Detection algorithms. The techniques used by several of theses algorithms are very costly computationally. For example, the entropy contrast method employed by Khurram is expensive in the computation of a LOG function over the entire speech waveform samples. Khurram method will be detailed further down in this chapter in the section on related work.

Instead of using quasi-stationary points in the speech waveform as in Khurram method, we use the CGEP features and high order spectral features (second cepstrum derivatives) to detect the silence segments. These techniques are called CGEP or Cepstrum derivative based silence detection. Wilpon, Lee, and Rabiner did something similarly to this approach in their research published in 1991 [28] on connected digit recognition except that they added the use of HMM method to refine their solutions. As Figure 3.2.1 shows, the use of CGEP features can help to delimitate quite well the speech regions of an utterance. The second order cepstrum derivatives can support the results from the use of CGEP to make them precise or serve as features on their own for silence detection.

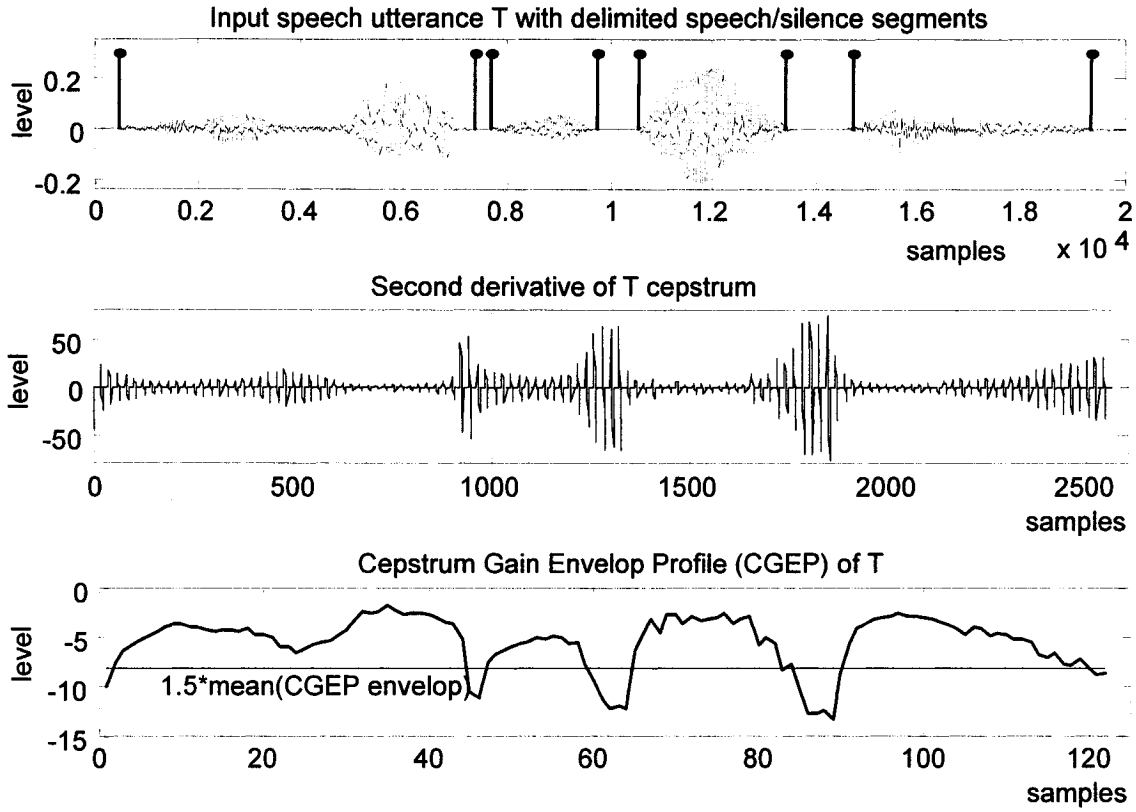


Figure 3.2.1: Silence/Speech delimitation using CGEP heuristic information. The first graph shows speech regions delimited by heuristic information provided by the CGEP shown in the third graph. The second graph shows the second order cepstrum derivative, which shows nulls for speech regions. The null regions can be used to detect silence/speech regions.

The delimitation shown in Figure 3.2.1 has been done using the Algorithm 3.2.1. In this algorithm, we assume the computation of the cepstrum is done externally.

Algorithm 3.2.1: *Silence/Speech delimitation using CGEP heuristic information*
Input: waveform samples of the input speech
Output: pairs of positions or sample indexes in the input speech waveform samples
- define x as the waveform of the input utterance; - define N as the length of frames of x; - define M as the shift step between 2 consecutive frames of x; %---------- Cepstrums computation ---------- - Cm = compute the array of cepstrums of x corresponding to each frame. That is a matrix whose columns correspond to cepstrums per frame. %---------- Extract the envelop of the cepstrum ---------- - CGEP = array of first coefficient of cepstrums in Cm. That is an array formed from the elements in the first row of Cm. %------- Set an heuristic threshold for the decision on silence/speech segments --------- - Set NOISE_FACTOR as correction factor, say NOISE_FACTOR = 1.5. - Threshold = NOISE_FACTOR * mean of CGEP. %------- Find the segments of the utterance that contains noticeable speech ---------- - Segments = detect Silence Segments using CGEP, Threshold, and N, M to recover the correct position of the segments in x. - Return segments;

The Algorithm 3.2.1 use CGEP features to find the segments. As the Figure 3.2.1 shows it too, we can use the derivative of the cepstrums to locate the silence or speech regions using Algorithm 3.2.2. However, the latter algorithm needs computations that are heavier than the former because the derivative features are far larger in size (Number of frames * LPC order).

Algorithm 3.2.2: Silence/Speech delimitation using cepstrums derivative
Input: - waveform samples of the input speech - order for cepstrums derivative; default value ORDER=2, second derivative
Output: pairs of positions or sample indexes in the input speech waveform samples
- define x as the waveform of the input utterance; - define N as the length of frames of x; - define M as the shift step between 2 consecutive frames of x; %---------- Cepstrums computation ---------- - Cm = compute the array of cepstrums of x corresponding to each frame. That is a matrix whose columns correspond to cepstrums per frame. %---------- Compute the derivatives of the cepstrum ---------- - dCm = differentiate the cepstrums to ORDER. %------- Set an heuristic threshold for the decision on silence/speech segments ------- - Set NOISE_FACTOR as correction factor, say NOISE_FACTOR = 0.5. - Threshold = NOISE_FACTOR * mean of dCm. %---- Find the segments of the utterance that contains noticeable speech segments ----- - Segments = detect Silence Segments using CGEP, Threshold, and N, M to recover the correct position of the segments in x. - Return segments;

Possible ways of using silence detection

We said earlier that the features that we dispose for the comparison algorithms for the recognition process were cepstrums. The computation of cepstrums usually covers the whole length of its underlined utterance waveform samples. Consequently, all of the parts of the waveform that corresponds to silence are also involved in the computation of the cepstrums and will also be carried to the comparison processes where their effect of time waste will be noticed.

A possible way to avoid having these useless parts is to remove them. But in this case, the time alignment of the speech features might be lost. For example, if the vowel "aa" was located at 5sec in the speech utterance, it might have shifted down to 3.75sec (the numbers are arbitrary). So, if the time alignment should be of some importance, we might use a labeling technique to indicate the silence parts and track the offsets so that we can reconstitute the original positions later. Therefore the comparison algorithms should use this labeling information to skip the part, thus shortening the overall length of the utterance.

3.3. Related work

In the literature, we found two major works, which look into the reduction of the length of the input signal. We present them hereafter and compare them to what we have implemented.

3.3.1. Reducing computational complexity and response latency through the reduction of content less frames [2]

The approach described in this section, is the one proposed by Rafid A. Sukkar [2] in IEEE ICASP 2000 to provide a solution to the speed issue in speech recognition. The method employs a Tree Structure Vector Quantization (TSVQ) classifier that discriminates between silence and non-silence frames. To reduce the computational complexity and latency in response of the ASR system, the method used three techniques: 1) silence skipping, 2) silence-based pruning of dynamic programming network and 3) early decision.

Though the method seems to be adaptable to continuous speech recognition, Rafid did his experiments on connected digit task and a large vocabulary company name task but not in continuous speech paradigm. The results showed a possible reduction of ASR response latency by more than 82%. On the computational reduction side, results showed 13.6% reduction on the connected digits task and 6.7% on the company name task.

Rafid also sees accuracy as the most important and widely quoted performance metric for speech recognition systems. Guided by reasons relative to the use of speech recognition in telecommunication applications, he found speed as the important issue for the actual stage of research in speech recognition. Relying on research efforts on reducing the computational complexity of real-time applications and efficient computation achieved by either reducing the complexity of HMM state likelihood computation [13, 14] or the optimization of decoding networks through pruning, Rafid oriented his method in a way that gives emphasis to ASR response latency.

The method manipulates the decoding network to simultaneously lower computational complexity and accurately determine the end of the input utterance in order to achieve low response latency. For Rafid, latency response of a recognizer depends, to large extent, on when the ASR system determines the input speech has stopped relative to the true end of the utterance. In other words, the processing of meaningless or content-less portions of an utterance is useless and causes response latency. This means, the ASR system needs a front-end unit able to determine and remove the useless parts of input utterances (parts that should not have matches in the back-end lexicon). To be efficient, the task of removing the underlined parts must be computationally inexpensive. Therefore Rafid, used Tree-Structured Vector Quantization (TSVQ) of the feature vector. Finding all of the content-less frames in an utterance using TSVQ should be too difficult, therefore Rafid focus on a single type of such parts that is silence. So Rafid's method presented in this section employs a TSVQ-based silence/non-silence frame classifier and consists of the three techniques mentioned earlier that are 1) silence skipping, 2) silence-based pruning of dynamic programming network and 3) early decision that will not be considered for the theme of this chapter.

The baseline recognition system used by Rafid employs an energy-based Voice Activity Detector (VAD) to determine the beginning and end points of the input utterance and consists of two units:

- A front-end to generate feature vectors consisting of 12 LPC-derived cepstral coefficients, normalized log energy, and the first and second order time differences of the cepstral and log energy parameters.
- A back-end unit that uses HMM and a single-pass Viterbi decoder in which the DP network is expanded and released in a way said frame-synchronously [15]. The decoder also uses a phone tree and a beam search algorithm to perform likelihood-based DP network pruning.

The front-end performs the silence classification using a TSVQ encoder that operates on the cepstrums coefficients features. The encoder is designed offline with data from a speech database containing phonetically balanced sentences (including silence). The principle is to mark the leaf nodes of the TSVQ, which corresponds to silence so that during classification, any frame whose feature vector resides in a marked cell is considered to be a silence frame. The task of the TSVQ is then to identify silence frames between the VAD start and end points into silence/speech segments. The energy-based VAD is supposed to chunk speech utterances from VAD start to VAD end. To limit errors (processing non-silence as silence frame in the back-end), the method requires that N_{skip} consecutive frames be classified as silence in the front-end before the middle frame of the group is tagged as silence and skipped by the back-end.

The pruning of the DP network is performed by the back-end on the tagged frames output by the TSVQ of the front-end. The first type of pruning is silence-based and happens on frames tagged as silence by TSVQ and the second type of pruning happens with the beam search on the basis of likelihood scores.

This technique that Rafid suggested computes cepstrums, time derivatives of cepstrums and log energy features of the speech utterance as it is common to every standard speech recognition system. However, the use of TSVQ seems difficult to handle and to update. Despite the fact that it is built offline, it has to be trained too. This implies a quasi-impossibility to adjust or to reconfigure the system in response to dynamic changes in the recognizer environment.

Knowing that ASR technology is becoming popular for highly variable environment applications such as ASR application in mobile devices, the ability to quickly adapt to changes is an important factor to care about in the design phase.

3.3.2. A robust algorithm for detecting speech segments using an entropic contrast [23]

Many studies show that the performance of any HMM or DP based speech recognition system depends on the length of the utterances to process. The more content-less parts there are in an utterance, the worst waste of time the system will generate. It will also impair its accuracy. So, it is always good to be able to remove all possible nonsense portions of the utterances to recognize. Such portions are noise and, with relative consideration, silences. For isolated word recognition in a limited vocabulary, the problem of content-less parts removal become the determination of correct boundaries for the words and the rejection of speech artifacts such as breath, mouth and lip clicks etc. In the case of connected speech, the problem is to remove intra-word silences and the speech artifacts. In continuous speech situation, efficient and precise automatic speech segmentation pre-processor can lower the computation load for the system.

The detection of speech presence in audio processing systems is known as Voice Activity Detection VAD. Lot of work has been published in this field. The common techniques used are energy-based, entropy-based and time derivative of cepstral coefficients. The techniques often use statistical methods to determine a threshold for the speech/non-speech classification.

In this section, we present the entropy-based approach proposed by Khurram Waheed and Kim Weaver in 2002 [23]. The entropy-based contrast exhibits more stable characteristics as compared to the energy-based methods. Experimental results from the study show that this algorithm outperforms the energy-based algorithms. Its accuracy is much better than that of other algorithms in high and medium signal-to-noise ration (SNR from 10 to

15dB) while for lower SNR, the improvement depends on the type of noise involved; the worst case is the white noise.

As Waheed said, conventional short time or spectral energy based endpoint detection algorithms are very sensitive to speech artifacts and their performance drops badly in the presence of noise. These algorithms usually exploit pitch and duration information, adaptive thresholds, zero crossover rates etc. Waheed suggests a method that computes the entropy of the speech directly in time domain.

The computation of the entropy (Equation 2.3.1) involves the evaluation of the probability distribution within each individual speech signal frame. To do so, a histogram of N bins should be built and normalized to satisfy the statistical properties of the cumulated distribution function (CDF).

$$H = -\sum_{k=1}^{N} p_k \log p_k \quad (2.3.1)$$

This process generates an entropy profile $\xi = [H_1\ H_2 \cdots H_m]$ for the complete speech data available assuming the speech is framed into m total number of frames. The entropy profile can be used as in Equation 2.3.2 to find a convenient speech/noise decision threshold γ for the entire available speech data. The μ factor helps to set the threshold a little higher than the mean entropy profile.

$$\gamma = \frac{\max(\xi) - \min(\xi)}{2} + \mu * \min(\xi); \quad \mu > 0 \qquad (2.3.2)$$

The minimum of the profile corresponds to remnant noise floor. Therefore the threshold γ as formulated in Equation 2.3.2 minimizes excessive influence of the background noise. Using the threshold γ, the entropy profile of the entire speech is clipped so that any portion of the speech signal corresponding to entropy higher than γ is considered as speech region and the other as noise and set to zero.

The result of this process should not be accurate unless a correction is made to cancel the effect of vocal and background speech artifacts. Knowing that humans do not generally generate very short duration sounds, every segment shown as speech in the clipped entropy profile whose length is shorter than a certain minimum length λ_i can be considered as artifact and zeroed. λ_i can be the length of the shortest phone or phoneme in the reference library.

The second anomaly correction regards the very often-accidental split of speech into two segments due to the pronunciation particularity of the phoneme or phone. In this case, using a series of criteria, the segments can be merged.

The results of K.Waheed's method to find speech boundaries show a very good accuracy though he gave no overall ASR system efficiency results. By computing the entropy in time domain on the available speech data, the method induces a huge computation load especially when the sampling frequency become high the time requirement for the log function will become significant. A possible way of cutting down such load is to reduce the number of samples to be processed.

CHAPTER-4

COMPARISON ALGORITHMS

This chapter is at the heart of our thesis, and presents two major investigations into speed improvement that is applying directly to the comparison process.

The first technique investigated is about the use of the dynamic programming algorithm to compare two utterances. The purpose of this first investigation is to learn the common way this comparison task has been done by others. Doing this, we found some corrections that might increase the processing speed in the standard algorithm. The result of the investigation is the Fast Two-Level Dynamic Programming Algorithm (FTLDP) that we describe in detail in the next section. Strengthened by the experience with TLDP, we come up with a completely different paradigm to do the same comparison task. We call the new technique ParrallelRecognizer that uses a "position competition for all" philosophy to qualify phonemes for the recognition output result. We will detail the latter algorithm in section 4.2 and in section 4.3 we will report some related work.

4.1. Fast Two-Level Dynamic Programming Algorithm (FTLDP)

FTLDP technique is designed to run in Stage.2 of TSA. Therefore, without doubting the accuracy or effectiveness of Stage 1 (DSP) in the TSA, the focus of Solution.1 is to have Stage.2 running a fast TLDP (to help find phoneme 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 Level 1 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 20 times faster than the original Two-Level DP algorithm as presented by Rabiner [8] and will be shown in Chapter 5 on experiments.

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

4.1.1. Definitions of parameters

Let T be a spectral sequence given as a test pattern and expressed as in Equation 4.1.1.1.

$$T = \{t(1), t(2), t(3), \cdots, t(M)\} = \{t(m)\}_{m=1}^{M} \qquad (4.1.1.1)$$

In Equation 4.1.1.1 t(m) can be an encoding of a condensed speech data through a filter bank (spectrums), a LPC (cepstrums), etc.

Given another spectral sequences U_i as reference patterns of speech units (say phonemes), such that U_i is expressed as in Equation 4.1.1.2 where N_i is the duration in frames of the i^{th} reference speech unit in a reference collection or vocabulary.

$$U_i = \{u_i(1), u_i(2), u_i(3), \cdots, u_i(N_i)\} \qquad (4.1.1.2)$$

Let us call the vocabulary C_V, containing V phoneme models and let it be expressed as in Equation 4.1.1.3.

$$C_V = \{U_1, U_2, U_3, \cdots, U_V\} \qquad (4.1.1.3)$$

Let us call R^S a string (concatenation) of phoneme models. We can express R^S made of L phoneme models long as in Equation 4.1.1.4 with q(l) a function that finds an optimal index v in the vocabulary C_V, and N^S is the total duration of the concatenated reference pattern R^S.

$$\left.\begin{aligned} R^S &= \left\{U_{q(1)} \oplus U_{q(2)} \oplus U_{q(3)} \oplus \cdots \oplus U_{q(L)}\right\} = \left\{u^S(n)\right\}_{n=1}^{N^S} \\ &\textit{where } U_{q(l)} \textit{ is the optimal word in the vocabulary for the position } l \\ &\textit{in the string and } q(l) \in [1, V] \end{aligned}\right\} \qquad (4.1.1.4)$$

The automatic recognition of an utterance T into text is a problem of solving Equation 4.1.1.5 for the approximation solution R^{S*} that corresponds to the smallest distortion D^*.

$$D^* = \min_{R^S} D\left(R^S, T\right) \textit{ the best distortion of } R^S \textit{ to } T \qquad (4.1.1.5)$$

It can be shown [8] that, the solution of the equation is to find (Equation 4.1.1.6) 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 $q^*(1), q^*(2), .., q^*(L)$.

$$D^* = \min_{L_{\min} \le L \le L_{\max}} \min_{\substack{q(1),\ldots,q(L) \\ 1 \le q(i) \le V}} \min_{w(m)} \sum_{m=1}^{M} d\left(t(m), r^S\left(w(m)\right)\right) \qquad (4.1.1.6)$$

T and R^S utterances are expressed as in Equation 4.1.1.7.

$$\left.\begin{aligned} T &= \{t(1), t(2), t(3), .., t(M)\} = \{t(m)\}_{m=1}^{M} \\ R^S &= \left\{U_{q(1)}, U_{q(2)}, U_{q(3)}, .., U_{q(L)}\right\} = \left\{r^S(m)\right\}_{m=1}^{N^S} \end{aligned}\right\} \qquad (4.1.1.7)$$

To cut down the computational cost of Equation 4.1.1.5 that is $O(M*L*V^L)$, two levels of processing is used as we will show it next.

4.1.2. FTLDP Level 1

The Level 1 of TLDP consist in finding the best indexes $v^*=q^*(i)$ in Equation 4.1.1.5 by solving Equation 4.1.2.1 for v^* over all possible segments [b, e] of T scanning all v in C_V.

$$\hat{D}(v,b,e) = \min_{w(m)} \sum_{m=b}^{e} d(t(m), r_v(w(m))) \quad (4.1.2.1)$$

It results in building matrices $\tilde{D}(b,e)$ for distortions and $\tilde{N}(b,e)$ for the corresponding indexes as in Equation 4.1.2.2.

$$\left.\begin{aligned} \tilde{D}(b,e) &= \min_{1\le v\le V} [\hat{D}(v,b,e)] \; best\ distortions \\ \tilde{N}(b,e) &= \arg\min_{1\le v\le V} [\hat{D}(v,b,e)] \; best\ indexes \end{aligned}\right\} \quad (4.1.2.2)$$

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 (Equation 4.1.2.3). DTW is the average cost of a time warping [8] operation.

$$O_1 = V * M * \overline{DTW} \quad (4.1.2.3)$$

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 of the reference phoneme models known as RPKB,
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, voiced or unvoiced phonemes decision might be made, and at the second node the space is narrowed down to a specific phoneme models cluster as illustrated in Figure 4.1.2.1.

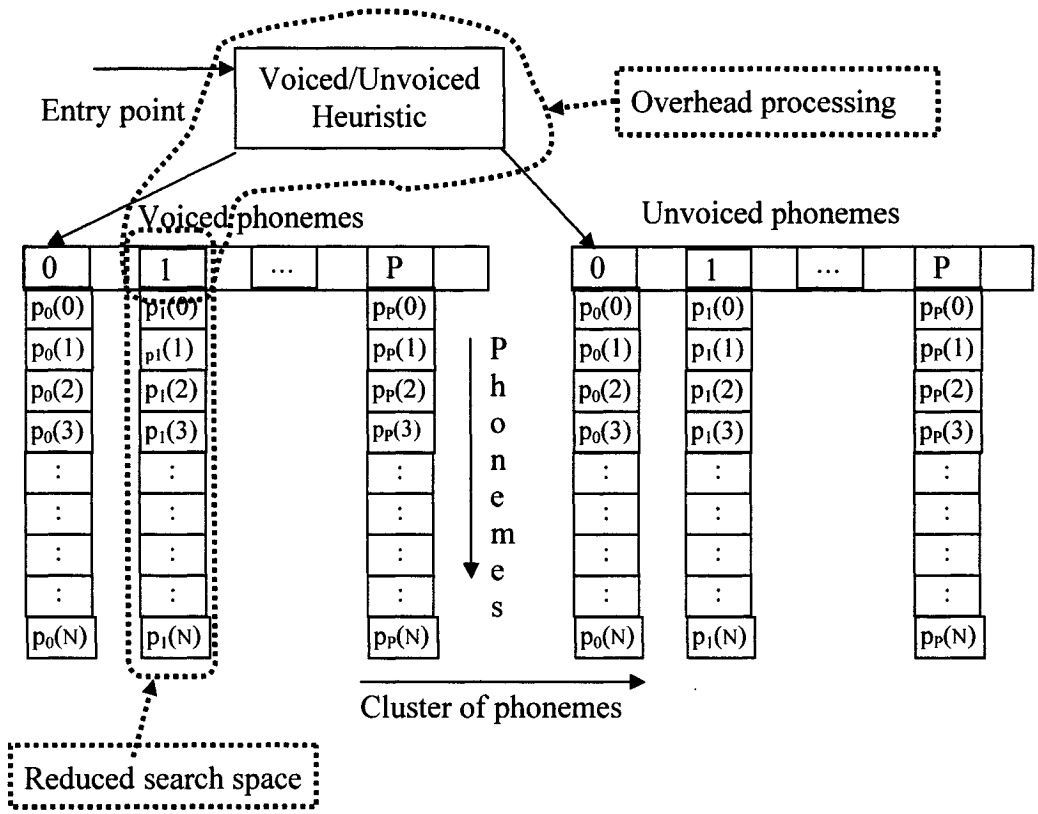


Figure 4.1.2.1: Illustration of the use of clustering to reduce the search space during matching processes.

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 (to favor accuracy). The detail of the clustering will not be shown in this section as it has been mentioned in Chapter 2 and it will be found in [6].

Technique 3: Considering only the energy features of cepstrums and computing the second order cepstrum derivative, Figure 4.1.2.2 shows patterns that can help cut off silence segments. Our present goal is to present the idea of using silence detection (a standard problem) that we have detailed as CGEP or cepstrum derivative based silence detection in the chapter on heuristics (Chapter 3) to shorten the length of an utterance. So we will not repeat detail of this technique in this section. Recently, many Voice Activity Detection (VAD) techniques have been used to perform this task [23, 26, 27, 1, and 4].

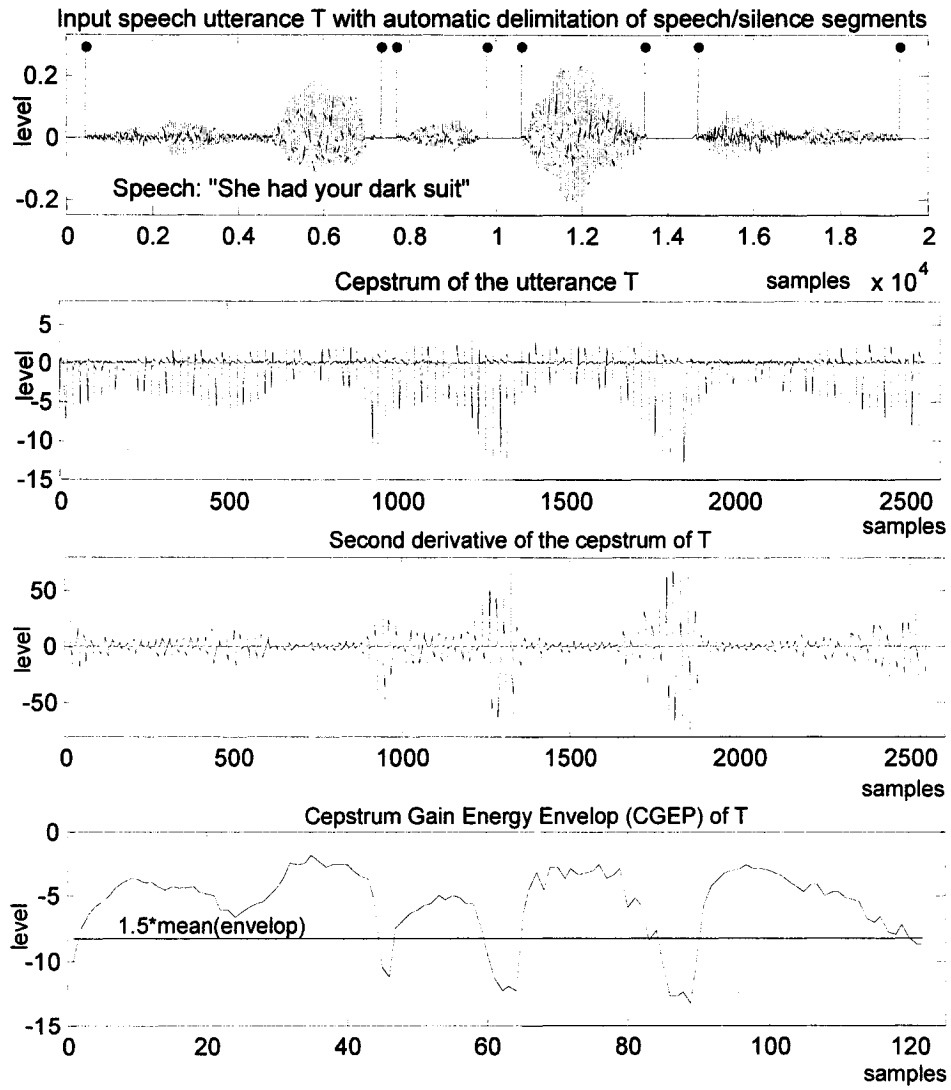


Figure 4.1.2.2: Silence/Speech delimitation using CGEP or cepstrum derivative heuristic information. The Second order cepstrum derivative shows nulls for speech regions. The null regions can be used to detect silence regions.

At this point, the Level 1 of this algorithm has exploited two heuristics (clustering and CGEP) to essentially reduce the search space, thus the size of the huge matrices $\tilde{D}(b,e)$ and $\tilde{N}(b,e)$. The processing of these matrices are the major sources for the computational load of the algorithm. Therefore, the action of the Level 1 is the heart of the FTLDP algorithm relative to speed issue in the ASR system. Next, we present the processing at Level 2. Level 2 does not use any heuristic but it gives a correction to the formulation of the DP algorithm as Rabiner expressed it in his book [8]. We refer to this correction as Technique 2.a.

4.1.3. FTLDP Level 2

The processing at Level 2 completes the resolution of Equation 4.1.1.5 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 Equation 4.1.3.1 over the whole utterance T. This involves a recursive process that Rabiner [8] expressed as in Equation 4.1.3.2-4.

$$\overline{D}_l(e) = \min_{1 \le b \le e} \left[\tilde{D}(b,e) + \overline{D}_{l-1}(b-1) \right] \quad (4.1.3.1)$$

Recursion as proposed by Rabiner's book

Step 1: Initialization

$$\overline{D}_0(0) = 0, \quad \overline{D}_l(0) = \infty, \quad 1 \le l \le L_{\max} \quad (4.1.3.2)$$

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

$$\overline{D}_1(e) = \tilde{D}(1,e), \qquad 2 \le e \le M \quad (4.1.3.3)$$

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

$$\left.\begin{aligned}
\overline{D}_2(e) &= \min_{1\le b\le e}\left[\tilde{D}(b,e)+\overline{D}_1(b-1)\right], 3\le e\le M \\
\overline{D}_3(e) &= \min_{1\le b\le e}\left[\tilde{D}(b,e)+\overline{D}_2(b-1)\right], 4\le e\le M \\
\overline{D}_l(e) &= \min_{1\le b\le e}\left[\tilde{D}(b,e)+\overline{D}_{l-1}(b-1)\right], l+1\le e\le M
\end{aligned}\right\} \quad (4.1.3.4)$$

Finale solution

$$\overline{D}^* = \min_{1\le l\le L_{\max}}\left[\overline{D}_l(M)\right] \quad (4.1.3.5)$$

Equation 4.1.3.5 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[8] distortion values resulting from the recursion in Equation 4.1.3.4. In fact, if the path distortion at a step l is invalid, because the path distortion at a step l +1 is a function of the value at step l, then the resulting path distortion value will also be invalid. Equation 4.1.3.6 illustrates the fact stated above.

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

In Equation 4.1.3.1,

$$\left.\begin{array}{l} \forall b_0, \textit{if } \overline{D}_{l-1}(b_0 - 1) = \infty \\ \quad \Rightarrow \overline{D}_l(e_0 = b_0 - 1) = \infty \ \forall (e_0 \le b_0) \end{array}\right\} \quad (4.1.3.6)$$

Equation 4.1.3.6 revels that, for every $\overline{D}_l$ involving invalid values as shown in Figure 4.1.3.1, at least b_0 evaluations can be skipped using a jump function J(l) to determine the *starting value* e_0 where the distortion becomes a valid value (Equation 4.1.3.7).

e	$\overline{D}_1(e)$	$\overline{D}_2(e)$	$\overline{D}_3(e)$
3	9	-	
4	10	-	-
5	13	-	-
6	17	$6+\overline{D}_1(4)$	-
7	22	$4+\overline{D}_1(4)$	-
8	25	$6+\overline{D}_1(4)$	-
9	29	*	$15+\overline{D}_2(6)$
10	33	*	$15+\overline{D}_2(7)$
11	37	*	$15+\overline{D}_2(8)$

where
- *computation of invalid value*
- ** some real number*

Figure 4.1.3.1: Illustration of invalid values impact in the computation of FTLDP Level 2

As illustrated in the Figure 4.1.3.1, it's needless to compute the "-" values though Equation 4.1.3.4 requires them to be evaluated. In fact as Table 4.1.3.1 shows below, we can determine the computation range using a memory function J(l).

Table 4.1.3.1: Illustration of possible jumps or computation skipping in the Level 2 of FTLDP

	Ranges required from algorithm	**Effectively needed ranges**	**Modification so that** $\overline{D}_l(e_0 = b_0 - 1) \neq \infty$	**Number of skipped evaluation using J(l) function**
For L=2	$1 \leq b < e$; with e = [3..11]	$4 \leq b < 11$; with e = [5..11] because $\overline{D}_1(0..2) = \infty$	J(1) = 3, J(1)+1 ≤ b < 11 ⇨ e = [J(1)+2 ..11] ⇨ $\overline{D}_2(0..3) = \infty$ guaranteed	3
For L=3	$1 \leq b < e$; with e = [4..11]	$7 \leq b < 11$; with e = [8..11] because $\overline{D}_2(0..5) = \infty$	J(2) = 6, J(2)+1 ≤ b < 11 ⇨ e = [J(1)+2 ..11] ⇨ $\overline{D}_2(0..5) = \infty$ guaranteed	6

We can express the memory function as in Equation 4.1.3.7.

$$J(l) = e_0 \text{ such that } D_{l-1}(e_0) \neq \infty \quad (4.1.3.7)$$

With a light offset computation overhead, J(l) also helps save memory by keeping only valid ranges in $\overline{D}_l$ matrix.

So the correction made to Rabiner's algorithm would be as shown in Equation 4.1.3.8.

$$\left.\begin{aligned} \overline{D}_l(e) &= \min_{(J(l-1)+1)\le b\le e}\left[\widetilde{D}(b,e)+\overline{D}_{l-1}(b-1)\right], \\ J(l-1)+2 \le e \le M \; &with \begin{cases} J(0)=0 \\ J(l)=first \arg \overline{D}_l(e)\neq\infty \end{cases} \end{aligned}\right\} \quad (4.1.3.8)$$

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 $1 \le L \le L_{Max}$ that corresponds to L^*. Therefore, by analyzing the return values of $\overline{D}_l$, FTLDP is able to find a length L_{cut} such that $L + \Delta L \le L_{cut} \le L_{Max}$ where ΔL is the slope study margin. This saves computation and also memory if storage for $\overline{D}_l$ is dynamically allocated. Figure 4.1.3.2 illustrates the detection of a local minimum.

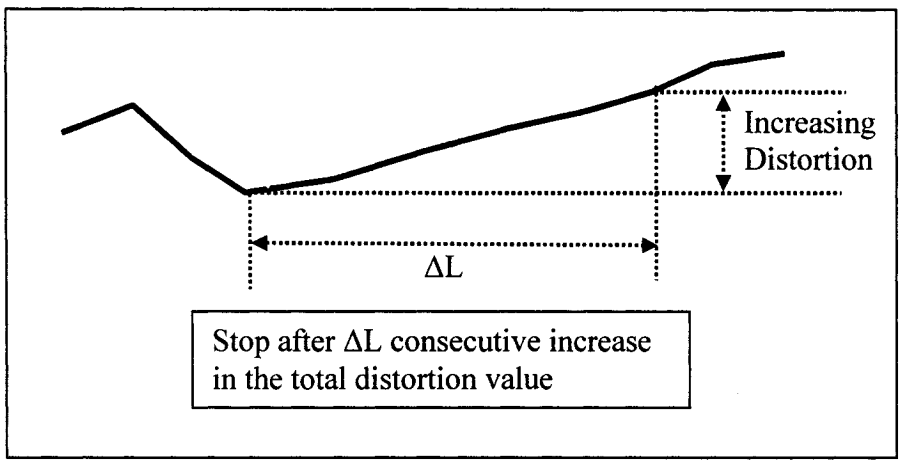


Figure 4.1.3.2: Graph showing a local minimum in $\overline{D}_l$ for $1 \le L \le Lmax$ in the computation of FTLDP Level 2. The detection of the minimum can help truncate the computation far shorter than Lmax iteration.

With the analytical detail of the FTLDP algorithm, we now give a summary of the principles used to design it.

4.1.4. Summary of the functionality of FTLDP as a revised TLDP

After introducing the concept of clustering, silence removal and jumps function, the Algorithm 4.1.4.1 shows the main lines of the comparison processing for FTLDP.

Algorithm 4.1.4.1: *Principle of FTLDP algorithm*
Input: - RPKB organized as an array of phoneme utterance clusters with clusters centroids accessible; a) A cluster is an array of phoneme utterances with one element (phoneme utterance) known as the centroid of the group. b) The length of the longest phoneme utterance in RPKB is known as PHONEME_MAX_LEN - Input speech utterance T
Output: SOLUTION_ARRAY, an array of ARPABET string symbol chains of recognized phoneme symbols
- Shorten the input speech utterance T using Silence Removing technique **//---------- 1st level of Two-Level DP: ----------** - Compute an inter frame distances triangular (lower) matrix by finding the best distortions (smallest distances) for phoneme models in a "Portion of RPKB" to all possible segments of T. Using the clustering technique to determine that "Portion of RPKB" which should be a winner cluster of phoneme models among other in RPKB. - Store this best distortion in matrix $\tilde{D}(begining\ frame, ending\ frame)$ - Store the index of the corresponding phoneme model in RPKB in matrix $\tilde{N}(begining\ frame, ending\ frame)$ **//---------- 2nd level of Two-Level DP: ----------**

- Using matrices $\tilde{D}(begining\ frame, ending\ frame)$ and $\tilde{N}(begining\ frame, ending\ frame)$, compute the matrix $D_{chain\ length}(ending\ frame)$ for all possible chains' distortion. The chains are built from length ChainLength = [1 to a maximum MAX_CHAIN_LENGTH] using the Jump Function and the Slope Analyzer to skip unnecessary computations.

//--- Build the resulting string chains of ARPABET symbols of phoneme models ----

- Build the resulting string chains of ARPABET symbols

- return the string chains of ARPABET symbols of phoneme models;

With the combination of the effect of clustering that reduces the size of the search space in the reference phoneme models (RPKB), the reduction of the size of the matrices $\tilde{D}(b,e)$ and $\tilde{N}(b,e)$ by shortening the input speech utterance as a consequence of removing silence segments, and the use of the jump function to skip unnecessary computation, we expect a faster algorithm compared to FTLDP. In Chapter 5, we will conduct some tests to very this expectation.

After working through FTLDP which is an algorithm derived from a standard one, we have gained insight into ASR systems and now we move to the next step which is to present our original algorithm that is called ParrallelRecognizer.

4.2. ParallelRecognizer

In the previous section we presented Solution.1 that is the reengineering of a standard DP algorithm to make it faster. In this section, we present Solution.2 which is our original suggestion called ParrallelRecognizer. It is an algorithm that works on the basis of best-fit principle. The principle is a position competition open to all phonemes in the

knowledge base or a section of the knowledge database if we involve heuristics in the process. Compared to DP algorithm of Solution.1, ParrallelRecognizer is capable of generating a large number of output phoneme chains in a very short time. This algorithm generates complete sentences compared to DP which produces sentences from length 1 to N among which, only one sentence has the preferred length.

4.2.1. Principle

This algorithm assumes the availability of a Reference Phoneme Knowledge Base (RPKB) whose units are already encoded in cepstrums and the possibility of getting some characteristics of these utterances to do some heuristic works. In fact at this point in the recognizer application, the RPKB is loaded (at the start of the application) and the template or utterance T to recognize should have been already encoded. Therefore, the main task of this algorithm is to find the best matches from RPKB to parts of the input utterance T in a consecutive way from the beginning frame to the ending frame of T. This description is a common principle to many algorithms. The originality of this algorithm is as follow.

In its standard form (without heuristics), the matching of every segment of the input utterance T is competed by the entire population of the RPKB. This means, every phoneme has a chance to score its distance or distortion to any possible part of T. This sounds like processing a DP algorithm but it's not. In fact, the term "every segment" and "any possible part of T" are not driven by statistical combination principle because it will degenerate into thousands of possible segments to be matched to hundreds units in the RPKB.

The algorithm opens a matching competition for segments of T starting from every frame of T (see Figure 4.2.1.1). A segment is defined from a starting frame to an ending frame [start, end]. This definition can also be seen as parameterized by a starting frame and the length of the segment [start, length]. Using the latter definition, the algorithm dictates (fixes) the starting frame but gives the freedom to every competitor phoneme to find the best value for the length of the segment. The advantage is that as units in RPKB are very

different in length, imposing a length for the segment is certainly at the disadvantage of all the competitors except the only one best unit for the underlined segment. However, there is no guarantee that the underlined segment should in reality correspond to a unit; in reality, it may be a segment overlapping at least two phoneme units. This is what is wrong in standard DP algorithms, the forceful matching process computation over every possible segment. Consequently, a huge amount of segment combination is cut down in the computation process compared to DP.

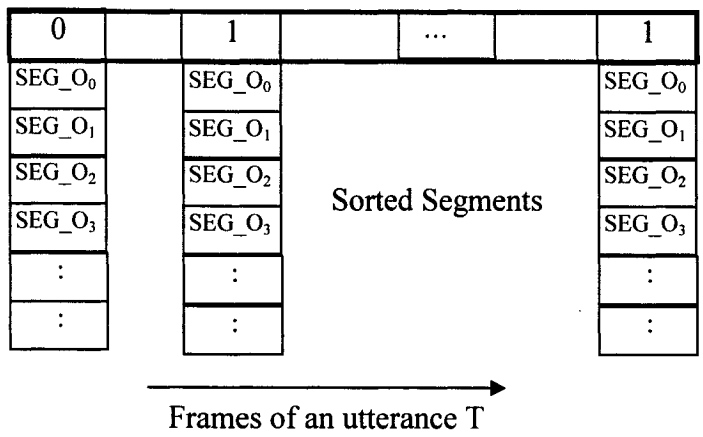


Figure 4.2.1.1: Battery of sorted segments corresponding to each frame of an utterance T. The segments should be sorted based on their scores relative to an ordering system.

The length of a segment is not unbound for the sake of efficiency. It ranges from 0.5 to 2 times the length of the competitor phoneme's own length [8]. This is the best range where a Dynamic Time Warping (DTW) can be usefully done.

An important advantage of this algorithm is that, unlike in FTLDP, there is no threshold value to guess regarding the length of the result phoneme string. ParrallelRecognizer produces strings of different length. The length of the string output result is automatically determined by the concatenation of the segment objects that form the string.

Unlike in this ParrallelRecognizer algorithm, FTLDP needs an initial guess for the length of the output phoneme string. The guessing of a threshold requires a heuristic evaluation, which may overestimate the length (thus waste of computation) or underestimate it (thus incomplete utterance recognition). In the case of overestimation, the cost of the useless computations caused by the added extra length is far too heavy. In fact, the size of the block of equations in Equation 4.1.3.4 becomes larger.

Heuristic: Using heuristics, it is possible to further reduce the computational load of the algorithm. In fact, with a light overhead computation, some heuristics can be defined to eliminate, before the process reaches the distance computations level, competitor phonemes that are "obviously not fit". For example, if a phoneme unit is determined as voiced but its corresponding preferred segment of the input utterance T is found to be unvoiced, there is no need for that unit to continue the competition. Similar heuristic based on energy, pitch or formants can be used as long as information is available to work them out without too much overhead.

As shown in this section, ParrallelRecognizer is based on a fundamental entity that we call segment. In the next section, we will define the segment entity and provide extensive detail about it.

4.2.2. Segment Object definition

ParrallelRecognizer algorithm heavily relies on an entity described in the section above and that we have called *segment*. In this algorithm, a segment has the same definition as in geometry. A segment has a beginning position and an ending position. It can also be expressed as a beginning position and a length. The particularity of the segments in this algorithm is that they are associated with a third parameter called *distortion*. The distortion parameter represents the value of the distance score got from a matching process between two utterance entities. Therefore, we define a segment object class (entity) called SEGMENT_OBJECT which binds the geometrical parameters to the distortion value as follow:

SEG_O = SEGMENT_OBJECT(X, Y, START_POS, END_POS, DISTORTION);

In words, SEG_O is a segment object resulting from the matching of the reference utterance X in its entire length to a part of the utterance Y starting from START_POS and ending at END_POS. The matching has scored a distortion or distance value of DISTORTION.

A second version of the definition, useful for some cases, is as follow:

SEG_O = SEGMENT_OBJECT(X, Y, START_POS, LENGTH, DISTORTION);

The interest of this class of segment objects is to encapsulate the geometrical segment with the distortion value that is the incremental output result of the recognition process and ease the chaining or the concatenation of best results afterward.

Now that we have defined the entity segment, it is important to notice that the principle of ParrallelRecognizer algorithm requires that an ordering rule be defined upon these objects. We tackle the ordering principle of the segment objects in the next section.

4.2.3. Segment Object ordering principle

ParrallelRecognizer algorithm uses sorting processes to work. The sort is performed on the segments determined by the matching process. Therefore, it is important to define an order among segments. Three comparison orders have been defined as follow:

Segment.Order 1. This is the default order to compare two segments objects. It is based, in priority order, on:

- The *starting position* (or speech frame) of the segments,
- *Their lengths,*
- And *their distortion scores.*

The pseudo-code in Algorithm 4.2.3.1 shows the detail of the priority among the segment objects parameters and it is illustrated in figures in Table 4.2.3.1.

Algorithm 4.2.3.1: *Segment.Order 1, the default order principle to compare two segments objects based on the starting frame position*

Input:

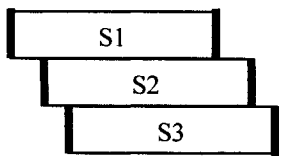
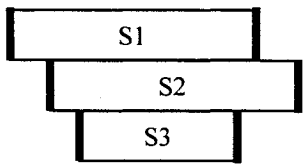
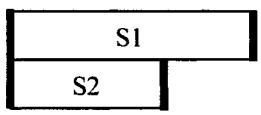
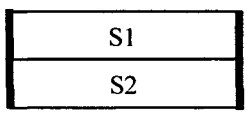
- SEGMENT_OBJECT S1;
- SEGMENT_OBJECT S2;

Output: an order index: LESS_THAN, GREATER_THAN xor EQUAL

```
If (S1.startFrame < S2.startFrame)
      order <- S1 LESS_THAN S2;
Else if (S1.startFrame > S2.startFrame)
      order <- S1 GREATER_THAN S2;
Else (Case equal: then decide upon length and then distortion)
{  If ( Length(S1) equals Length(S2) then decide upon distortion)
   {  If (S1.distortion < S2.distortion)
      {   order <- S1 LESS_THAN S2;
      }Else if (S1.distortion > S2.distortion)
       {   order <- S1 GREATER_THAN S2;
      }Else (case S1.distortion equals S2.distortion)
      {   order <- S1 EQUAL S1;
      }//End if
   }Else if(Length(S1) >  Length(S2) then priority to distortion relative to length)
   {  If (S1.distortion < ACCEPTABLE_DISTORTION_MARGIN *
S2.distortion)
      {   order <- S1 LESS_THAN S2;
      }Else
      {   order <- S1 GREATER_THAN S2;
      }//End if
```

```
}Else (Length(S2) > Length(S1))
{  If (ACCEPTABLE_DISTORTION_MARGIN * distortion > S2.distortion)
   {   order <- S1 GREATER_THAN S2;
   }Else
   {   order <- S1 LESS_THAN S1;
   }//End if
}//End if
}//end if
```

Table 4.2.3.1: *Illustration of the segment objects comparison techniques according Segment.Order 1 principle*

 	S1 < S2 < S3
	S1 < S2 iff S1.Distortion <= S2.Distortion * ACCEPTABLE_DISTORTION_MARGIN
	S1 == S2 iff S1.Distortion == S2.Distortion

This default definition helps in the forward tracking of recognized units (phonemes)

Segment.Order 2. In this case, the *ordering is mainly based on the ending position* (ending speech frame) of the segment. This definition is useful for a backward chaining or backtracking of recognized units (phonemes). It is similar to Segment.Order 1 principle except that, instead of using the starting frame position, it uses the ending frame position then the length and the distortion parameters follow in priority order. See Algorithm 4.2.3.2 the detail and Table 4.2.3.2 for the illustration.

Algorithm 4.2.3.2: *Segment.Order 2, the order principle to compare two segments objects based on the ending frame position*

Input:

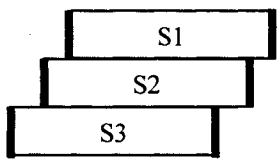
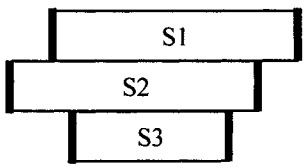
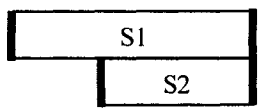
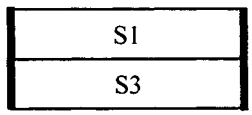
- SEGMENT_OBJECT S1;
- SEGMENT_OBJECT S2;

Output: an order index: LESS_THAN, GREATER_THAN xor EQUAL

```
If (S1.endFrame > S2.endFrame)
      order <- S1 LESS_THAN S2;
Else if (S1.endFrame < S2.endFrame)
      order <- S1 GREATER_THAN S2;
Else (Case equal: then decide upon length and then distortion)
{  If ( Length(S1) equals Length(S2) then decide upon distortion)
   {  If (S1.distortion < S2.distortion)
      {   order <- S1 LESS_THAN S2;
      }Else if (S1.distortion > S2.distortion)
       {   order <- S1 GREATER_THAN S2;
      }Else (case S1.distortion equals S2.distortion)
      {   order <- S1 EQUAL S1;
      }//End if
   }Else if (Length(S1) >  Length(S2) then priority to distortion relative to length)
   {  If (S1.distortion < ACCEPTABLE_DISTORTION_MARGIN * S2.distortion)
      {   order <- S1 LESS_THAN S2;
```

```
        }Else
        {   order <- S1 GREATER_THAN S2;
        }//End if
    }Else (Length(S2) >  Length(S1))
    {  If (ACCEPTABLE_DISTORTION_MARGIN * distortion > S2.distortion)
        {   order <- S1 GREATER_THAN S2;
        }Else
        {   order <- S1 LESS_THAN S1;
        }//End if
    }//End if
}//end if
```

Table 4.2.3.2: *Illustration of the segment objects comparison techniques according to Segment.Order 2 principle*

	S1 < S2 < S3
	
	S1 < S2 iff S1.Distortion <= S2.Distortion * ACCEPTABLE_DISTORTION_MARGIN
	S1 == S2 iff S1.Distortion == S2.Distortion

This definition helps in the backward tracking of the recognized units (phonemes). It is also useful to group and give easy access to all the units that end their preferred lengths at some specific frame position.

Segment.Order 3. The third and last definition of the order of a segment object *works with the distortion parameter only* (regardless the length and positions of the segments) as detailed in Algorithm 4.2.3.3.

Algorithm 4.2.3.3: *Segment.Order 3, the order principle to compare two segments objects based on the distortion value*
Input: - SEGMENT_OBJECT S1; - SEGMENT_OBJECT S2;
Output: an order index: LESS_THAN, GREATER_THAN xor EQUAL

If (S1.distortion < S2.distortion)
{ order <- S1 LESS_THAN S2;
}Else if (S1.distortion > S2.distortion)
{ order <- S1 GREATER_THAN S2;
}Else (case S1.distortion equals S2.distortion)
{ order <- S1 EQUAL S1;
}//End if

This definition helps in the case we would want to get the closest (smallest distance) solution by chaining the best recognized units (phonemes).

At this point, we have defined the principle of ParrallelRecognizer, its fundamental object called segment and the ordering principle upon it. We now present detail of the ParrallelRecognizer algorithm.

4.2.4. ParrallelRecognizer Algorithm

Let us consider an unknown input speech utterance T to recognize using a Reference Phoneme Knowledge Base RPKB. The algorithm to do this using ParrallelRecognizer principle is as listed in Algorithm 4.2.4.1.

Algorithm 4.2.4.1: ParrallelRecognizer principle
Input: - RPKB organized as an array of phoneme utterance clusters with clusters centroids accessible; c) A cluster is an array of phoneme utterances with one element (phoneme utterance) known as the centroid of the group. d) The length of the longest phoneme utterance in RPKB is known as PHONEME_MAX_LEN - Input speech utterance T
Output: SOLUTION_ARRAY, an array of ARPABET string symbol chains of recognized phoneme symbols
Initialization - Create a empty pile of segment object; - Create an array of the centroids in RPKB and define it as CENTROIDS; they represent their respective clusters; - Define symbol LEN as the geometric length of a segment of T; - Define symbol SEG as a geometrical meaning of a segment [POS, LEN] of T; //---------- //--- Populate the pile with segment objects resulting from the distortion values scored //--- by eligible units of RPKB. The competition is opened for positions from left to //--- the end frame of T. //---------- **For each position POS from the beginning frame of T to its ending Frame** { //Open the distortion score competition for every unit of RPKB

- Let LEN be guessed as PHONEME_MAX_LEN; thus form SEG = [POS, LEN] from T;
- Find the nearest centroids in CENTROIDS to SEG; thus the closest phoneme cluster of RPKB to SEG; call it CLUSTER;

For each unit or phoneme U in CLUSTER

{ - determine a more accurate segment length LEN of T preferred to the current phoneme U starting from POS. This is the determination the length of a segment SEG as a function of the length of U or not.
- Heuristic process if available to determine if the candidate U is eligible to continue the competition. This heuristic judgment is to determine if SEG is similar to U.
- If not similar, continue with the next U in CLUSTER.
- Compute the distortion DIST between U and SEG. DIST = getDistortion(U, SEG).
- Create a segment object as SEG_O = SEGMENT_OBJECT(U, T, POS, LEN, DIST).
- Store SEG_O in the pile of segments.

}//end for on U

}//end for on POS

// At this point an array of all winning phonemes possibly for all positions (frames)
// is available.

//---

//---- To provide a direct access to the segment objects later in this algorithm in forward
//---- and back tracking of the segments, let us build a battery of sorted segments based
//---- on their startFrame then on their endFrame members.

//---

- sort (PILE using rule Segment.Order.1);
- START_POINT_PILE = get the best segment objects based on Segment.Order.1 for every position from the beginning of T to its end.
- sort (PILE using rule Segment.Order.2);

```
- END_POINT_PILE = get the best segment objects based on Segment.Order.2 for
  every position from the beginning of T to its end.

//------------------------------------ Post treatment ------------------------------------
//--- Build recognized output phrases
//----------------------------------------------------------------------------------------
  - Consider LEADING_ELEMENT_PILE = START_POINT_PILE;
  - Define SOLUTION_ARRAY as an array of chains of phoneme symbols;
For up to the maximum number of output Phrases required MAX_PHRASE
{  - Peak the next element E on LEADING_ELEMENT_PILE
    - Using START_POINT_PILE and END_POINT_PILE,
        - Backtrack a chain of the best consecutive segments toward the beginning of T
          from E.START_POS
        - Forward track the chain of the best consecutive segments toward the ending of
          T from E.END_POS
    - Evaluate the total distortion of the chain obtained from the tracking.
    - Store the chain as an output recognized solution phrase in SOLUTION_ARRAY.
}//End for
Return SOLUTION_ARRAY.
```

As the Algorithm 4.2.4.1 shows, ParrallelRecognizer has three blocks:

- build a vector of segment objects from the already available input speech and template (reference phoneme) utterances and compute the distances,
- use sorting to order the vector using the defined rules,
- and chain the best segments such that we form logical connectivity of these from the beginning frame of the input utterance T to its end.

ParrallelRecognizer can also optionally combine the use of heuristics to reduce the number of template competitors. This approach is completely different from FTLDP on the following points:

- It uses vector rather than matrices. This is important because the handling of matrices by Operation Systems is far more complex than vector. In the worst case, the handling of huge matrices in memory can trigger the use of complex virtual memory processes that obviously slow down the overall computation.
- ParrallelRecognizer does more comparisons than evaluations compared to FTLDP either in the sorting or in the distance computation. In fact, the distance computation is optimized to a limited range of segment length unlike in FTLDP where it was required to evaluate the distance over unrealistic ranges.
- A fundamental flexibility of this algorithm is that, after the sorting block we can change the way the chaining block can be operated.
    - We can starting with the best segment from the frame zero and chain forward to the end frame of the input utterance T,
    - We can start with the overall best segment and continue the chaining both forward to the end frame of T and backward to frame zero.
    - We can start with the best segment ending at the end frame of T and backtracking to frame zero.
    - We can build the chain using a sorted vector with the rule Segment.Order.3 (distortion only) and forward and backward tracking. This will give the best distortion (smallest) results.

As we will show in Chapter 5 on the experimentations, the above factors help ParrallelRecognizer to gain tremendously in speed compared to FTLDP.

We have presented the two algorithms (FTLDP and ParrallelRecognizer) as announced in the introduction as solutions for fast comparison process in ASR systems. The two algorithms are very different in concept but achieve the same goal, the recognition of phoneme units in Stage.2 of TSA. We now present in next section a technique suggested by Ohno to operate the comparison process in ASR to give us an idea of how other authors have resolved this problem.

4.3. Related work

In comparison to the two algorithms developed in this chapter, we find in the literature the work of Rafid, Shawn and Anand [2], which we have partially presented in chapter 3. Rafid, Shawn and Anand have proposed an utterance comparison approach that uses DP combined with HMM and an early decision method to reduce the computational load for speech recognition. The technique, combined with the silence detection (presented in chapter 3) resulted in 13% reduction of computational complexity on connected digit task and 6.7% on the company name task (connected words topology).

The concept that is special in their work compared to ours if the use of the early decision technique in the utterance comparison module. The goal of the early decision technique is to report the recognition answer significantly sooner than the end response indicated by the Voice Activity Detector (VAD) endpoint (see chapter 3, section 3.3.1). This technique can be expensive, as Rafid feared it. Using a Tree Structure Vector Quantization (TSVQ) technique, the method consists in checking for non-final nodes among the surviving DP nodes after each DP network pruning. If all word paths reside in final nodes when silence-based DP pruning is triggered, we can assume that we have reached the end of sentence for the utterance. Consequently, the word sequence with the current best likelihood score is taken as the recognition answer.

A similar attempt by S. Nkagawa [11] to quicken the comparison process 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 a TSVQ approach [3].

We now give a summary (but with some detail) of the Method for Efficient DP Matching in Spoken Word Recognition [18].

Ohno presented in 1994 the method that we are going to present in this section for spoken words recognition. This method dealt with speed issue as it investigated the reduction of

processing time in the conventional recognition method that is Dynamic Programming (DP matching) algorithm. He found that the performance of the new and widely used statistical methods using Hidden Markov Models for spoken word recognition are not quite good or comparable (does not quite parallel, he stated) to that of the conventional DP method based on template matching. Assuming that word-size templates are available, the only disadvantage of the conventional DP method is the much greater amount of computation it needs over HMM-based methods. Therefore, he did not include any statistical method such as Hidden Markov Model in his work. The results of his recognition experiments on a 50-word vocabulary indicate that the method can reduce the computation time to 36 % of the time required for a conventional DP matching method.

The method is based on the detection and utilization of points along the time axis where the acoustic features of the speech signal are quasi-stationary. In other words, it is about finding good delimitation points to chop the input speech utterance and the reference phoneme templates into segments. These quasi-stationary points are called QS-points. They are determined for both the input speech and for the stored templates, and are used for:

2- Pre-selection of candidate templates for DP matching. This process helps reduce the candidates population and thus, the overall matching process time.
3- Reduction of the search space for piecewise DP matching between corresponding intervals of the input signal and a candidate template. In fact, DP methods normally require time alignment process without which, the useless but searchable space becomes huge.

The analytical feature that helps detecting the QS-points is the norm of the regression coefficient vector calculated over several frames of the speech signal as the measure for the degree of stationery of the signal [16]. The QS-points are defined as the point in time where the norm of the regression coefficient vector is at a local minimum below a certain chosen threshold. The stability of this detection method resides in a careful selection of the smoothing interval (frame overlapping steps) and the threshold of detection. If these conditions are successfully met, the QS-Point is generally found within each segment

with a relatively longer duration such as a vowel, a nasal murmur, a fricative consonant, and the closure interval of a plosive consonant, etc.

The norm $\Delta(t)$ of the regression coefficient vector using 12 FFT cepstrum coefficients $c_i(t)$ for i = 1,...,12, is expressed as in Equation 4.3.1.

$$\Delta(t) = \sum_{i=1}^{12} \left| \frac{\sum_{n=-N_r/2}^{N_r/2} n * c_i(t+n)}{\sum_{n=-N_r/2}^{N_r/2} n^2} \right|^2 \qquad (4.3.1)$$

The coefficient $c_i(t)$ is calculated over a frame of 25.6ms with 7ms steps in his experiments that used speech signals sampled at 10kHz with 12-bit precision. The QS-points were detected as the local minima of the magnitude of the Δ-cepstrum over N_r = 21 frames (147ms). The threshold for detecting the QS-points was set to 1.0 and a total number of templates Nt were selected per word using the k-means clustering technique [6].

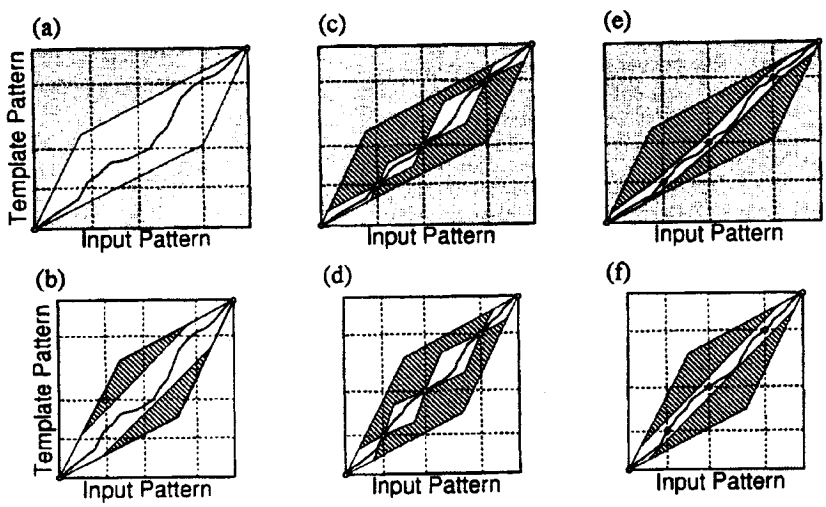
The occurrence of QS-points is expected to be similar, if not exactly the same, for an input speed utterance and the corresponding template of the same word. Also, for time alignment factor, a range of [0.4, 2.5] is used for the possible ratio of durations of the corresponding intervals of the signal and a template. Therefore, considering only templates that satisfy the alignment duration ratio requirements for further selection, the pre-selection method qualifies only templates whose QS-points form acceptable pairs with QS-points in the input utterance. If pairing the endpoints of the input and a template terminates successive detection of QS-point pairs, the template is selected as a candidate for further DP matching.

The result of the preliminary experiment indicates that the pre-selection procedure can reduce the number of candidates to approximately 60 %.

The DP matching process is to be done on piecewise basis. This reduces the length of the utterances to match as the computation time of a DP process also depends on the length of the input utterance. So the method defines segments corresponding to intervals over which piecewise DP matching can be performed using the correspondence between QS-points of the input utterance and a candidate template.

Compared with the conventional word level DP matching, this piecewise DP matching can reduce the amount of computation to a large extent.

Ohno investigated several slope limitation methods (time alignment) to reduce the search area. From the conventional range [0.5, 2] through the use of allowable time difference window ±W/2 and time axis equalization, he came up with the idea of the use of a search area that will have a certain degree of allowance on the correspondence between QS-points as shown in Figure 4.3.1.



(a) Conventional DP matching with the slope limitation [0.5, 2].
(b) Introduction of a time window of width W to procedure (a).
(c) Piecewise DP matching without piecewise time axis equalization.
(d) Piecewise DP matching with piecewise time axis equalization.
(e) Introduction of a time window to allow for approximate correspondence of the QS-points on the input and a template.
(f) Procedure (e) applied after piecewise time axis equalization.

Figure 4.3.1: Various procedures for reducing the search space in DP matching.

The time has past for the restriction of speech recognition applications to isolated or connected word or digits recognitions. As mentioned in the section on opportunities, the market tends to orient toward continuous speech cases unless the scope of the application is reduced to commands or voice recognition. Ohno used piecewise technique to reduce DP computation field, and pairing of end points to reduce the search space for isolated words. We should think of extending this to continuous speech case.

4.4. Conclusion of the development of FTLDP and ParralelRecognizer

In this Chapter 4 we have developed the two algorithms that correspond to Solution.1 and Solution.2 announced in the introduction in Chapter 1. As shown with extensive detail, the techniques (algorithms) embed several principles that make us expect good speed performance as a result.

In summary, we have worked to reduce the size of the search in RPKB for both algorithms, reduce the size of distance computation matrices for FTLDP and eliminate matrix processing and unrealistic comparison for ParrallelRecognizer.

As the theory about these algorithms is completed, we now move to experiments and testing to verify our speed performance expectations.

CHAPTER-5

EXPERIMENTS AND RESULTS

To test the performance of the two solutions (Solution.1 & 2) that we have suggested in Chapter 4, we have written a code in java where both techniques have been implemented in addition to some modules necessary to evaluate the results. In this chapter, we will show the results and the conditions in which they are obtained.

For both techniques, the input data is the same:

- A speech utterance of the phrase "she had your dark suit in greasy wash water all year". Hereon, this utterance will be referred to as input utterance T. It is a very clean utterance provided by TIMIT database. The waveform of the utterance is shown in Figure 5.1 where we can see that the recording was noiseless.

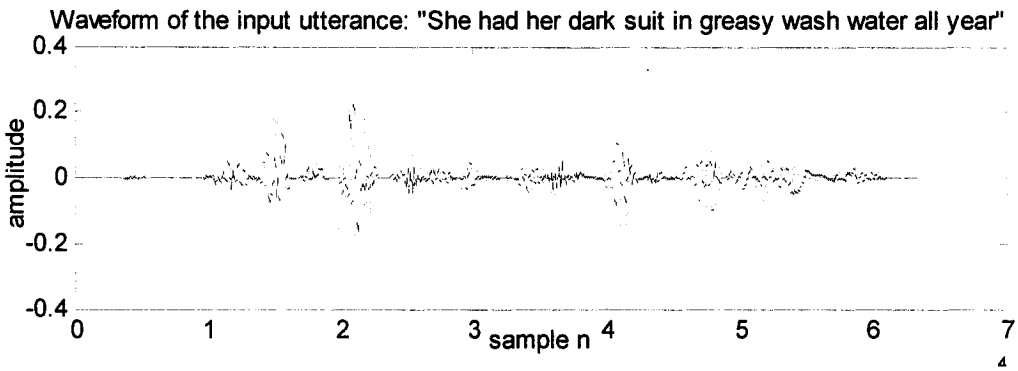


Figure 5.1: Waveform of the input utterance: "she had your dark suit in greasy wash water all year" in TIMIT database.

- A Reference Phoneme Knowledge Base (RPKB) as presented in chapter 8.

Also, the results are interpreted and evaluated in the same manner:

- The speed performance from the start of the underlined technique to the end of its execution. This implies, the overhead time required to prepare the input speech raw signal into features (cepstrums, LPC gain envelops, etc.) does not count as they are fix time costs and they do not depend on the technique to be used during the comparison.
- The percentage score of the recognized phonemes (accuracy)

5.1. Preprocessing tasks

These tasks regard the conversion of the input speech raw signal into features (cepstrums, LPC gain envelops, etc.). It can include any technique useful in enhancing the speech signal (noise elimination, pre-emphasing, framing, windowing, etc.). These tasks are in two folds: the preparation of RPKB and the preparation of every input utterance.

- The preparation of RPKB is done one for the life time of the recognizer application unless some reconfigurations have been done on some key parameters related to these tasks such as frame size, sampling frequency, LPC analysis order, a choice of clustering/no-clustering mode, etc. In that case, the system updates the RPKB only once until next changes.
- The preparation of input speech utterance is done every time a new utterance is to be recognized. The utterances are usually not long and the tasks are fast. Since our goal here is not the design of a highly real time constrained system, the handling of these tasks in continuous speech recognition can be done through the use of a queue of utterances to recognize. The speech signal can be captured using a voice-capturing device such as microphone in the case of a continuous ASR. However, for simplicity and to efficiently evaluate the speech feature that is the main goal of this thesis, we have opted for a speech input from an audio wave file. To smooth out unequal Operation System influence on processing times, the use of audio files as speech source can help us perform some average

testing using the same conditions for all tests. Also, the same conditions can be used to test techniques of Solution.1 and Solution.2 fairly.

The preprocessing tasks consist of operations shown as blocs in the general schema in Figure 5.1.1 as Rabiner proposed it [35]. More technical detail about these operations is given in the Technical Report on our experimentations [39] and can also be read in the general literature on ASR.

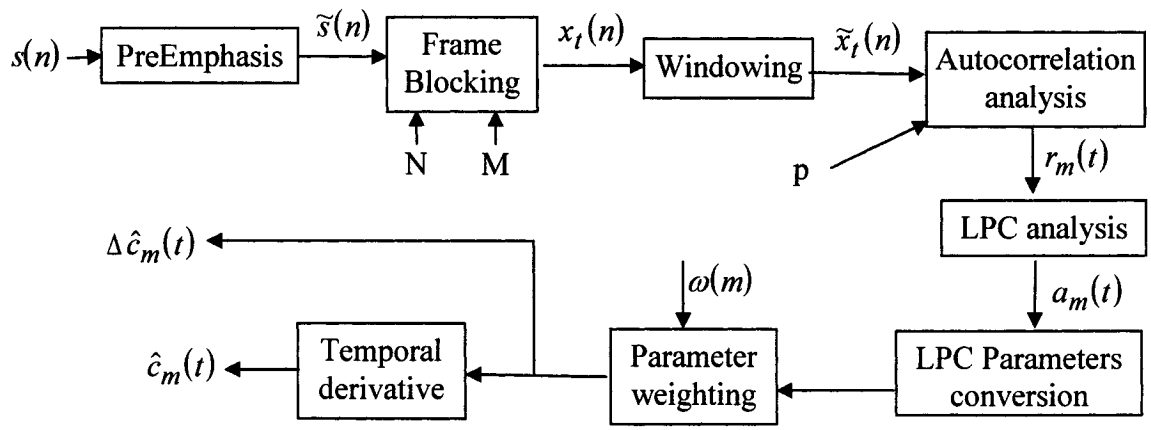


Figure 5.1.1: Block diagram of LPC processor for speech recognition

The goal is to make the speech signal a stationary process so that DSP analysis tools can work with good satisfaction. To fulfill this requirement, the technique used is to chop the speech signal into "frames" that are small enough (about 20msec) to be considered stationary signals. The result of the segmentation of the signal called framing is as shown in Equation 5.1.1.

$$\begin{matrix} \textit{There is one frame every } M \textit{ samples, or } Fs/M/\text{sec} \\ \begin{bmatrix} x(0) \\ \vdots \\ x(M) \\ \vdots \\ x(n) \end{bmatrix} \textit{convert to frames} \begin{bmatrix} x_0(0) & \cdots & x_0(m) & \cdots & x_0(L-1) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ x_M(0) & \cdots & x_M(m) & \cdots & x_M(L-1) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_n(0) & \cdots & x_n(m) & \cdots & x_n(L-1) \end{bmatrix} \end{matrix} \quad (5.1.1)$$

- **Pre-emphasis**: This module flattens the speech signal using Equation 5.1.2 to make it robust to finite precision effects, to reduce the effects of the glottal pulses and radiation impedance, and also to take the focus to the spectral properties of the vocal tract.

$$H(z) = 1 - a * z^{-1} \quad where \quad 0.9 \le a \le 1.0 \qquad (5.1.2)$$
$$\tilde{s}(n) = s(n) - a * s(n-1) \quad in\ time\ domain$$
$$Usually,\ a = 0.95.$$

- **Frame blocking**: provides stationary speech signal. The generic expression for Equation 5.1.1 is Equation 5.1.3 where N is the total number of samples in the original signal s(n), M is the number of samples shift in between consecutive frames and L is the total number of frames that the original signal is segmented into.

$$x_l(n) = \tilde{s}_p(M * l + n), \quad where\ n = 0,1,2,\ldots,N-1, \quad l = 0,1,2,\ldots,L-1 \qquad (5.1.3)$$

- **Windowing**: smooth out the edges of the frames as to minimize the discontinuities at the beginning and at the end of the signal frame using a window function ω(n) and it is provided by Equation 5.1.4. The Table 5.1.1 gives some examples of window function ω(n). Hamming window is the commonly used window function for it has a very good attenuation of its first lobes.

$$\tilde{x}_l(n) = x_l(n)\omega(n), \quad 0 \le n \le L-1 \quad (5.1.4)$$

Table 5.1.1: Analytical expressions and characteristics of window functions.

Window type	Expression
Rectangular	$\omega_1(n) = u(n) - u(n-L)$

Hamming and Hanning	$\omega_2(n) = \omega_1(n)\left(\beta_1 - 2\beta_2 \cos\left(\frac{2\pi}{L-1}n\right)\right)$ *For Hanning*: $\beta_1 = 0.5, \quad \beta_2 = 0.25$ $\omega_2(n) = 0.5 - 0.5\cos\left(\frac{2\pi}{L-1}n\right)$ (5.1.4) *For Hamming*: $\beta_1 = 0.54, \quad \beta_2 = 0.23$ $\omega_2(n) = 0.54 - 0.46\cos\left(\frac{2\pi}{L-1}n\right)$ (5.1.5)
Hamming is the commonly used window	

The length L of the window depends on the needs of the system in terms of number of formants to resolve for. To resolve for F_0 (the pitch), the width of the main lobe ($2F_s$) should not exceed F_0. For example using Hamming window, we might solve for L in the Equation 5.1.6.

$$\frac{8\pi}{L} \leq \frac{2\pi F_0}{F_s} \quad \Leftrightarrow \quad L \geq 4\frac{F_s}{F_0} \qquad (5.1.6)$$

Usually, L is chosen to be equal to N, the length of the frames generated in frame blocking module.

- **Autocorrelation, Linear Predictive Coding (LPC) and LPC Parameter Conversion to Cepstral Coefficients**: extensive explanation of these modules can be seen in [39] and in the literature. The objective is to convert the framed speech signal into cepstrum using an autocorrelation method in a LPC technique. Basically, the autocorrelation matrix R(n) of the speech samples is computed using Equation 5.1.7, then LPC parameters are calculated using the algorithm expressed in Equation 5.1.8 and finally the cepstrums are derived from the LPC parameters with the Equation 5.1.9 as summarized in Table 5.1.2.

Table 5.1.2: Analytical expressions for Autocorrelation, LPC and Cepstrums.

<table>
<tr><td colspan="2">Autocorrelation matrix of the speech samples</td></tr>
<tr><td colspan="2">$$r_l(k) = \sum_{n=0}^{N-1-k} \tilde{x}_l(n) * \tilde{x}_l(n+m), \quad where\ k = 0,1,\ldots,p \qquad (5.1.7)$$</td></tr>
<tr><td>LPC parameters derivation</td><td>Cepstrums derivation from LPC parameters</td></tr>
<tr><td>$$E^0 = r(0),$$
$$for(1 \le i \le p) \qquad (5.1.8)$$
{
$$k_i = \frac{\left\{ r(i) - \sum_{j=1}^{i-1} \alpha_j^{i-1} * r(i-j) \right\}}{E^{(i-1)}}$$
$$\alpha_i^i = k_i$$
$$for(1 \le j \le i-1)$$
$$\left\{ \alpha_j^i = \alpha_j^{i-1} - k_i * \alpha_{i-j}^{i-1} \right\}$$
$$E^{(i)} = (1 - k_i^2) * E^{(i-1)}$$
}</td><td>$$\left.\begin{aligned} G^2 &= E_{\min} \\ c_0 &= \ln(G^2) \qquad 1 \le m \le p \\ c_m &= a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k c_{m-k} \quad for \quad 1 \le m \le p \\ c_m &= \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k c_{m-k} \quad for \quad m > p \end{aligned}\right\} (5.1.9)$$</td></tr>
</table>

- **Typical LPC analysis parameters**: the values provided in the Table 5.1.2 that are given in Rabiner's book [35] can also be analytically determined as show in [39]

Table 5.1.3: Typical values for LPC parameters

Parameter	F_s = 6.67kHz	F_s = 8kHz	F_s = 10kHz	Description
N	300 (45 msec)	240 (30 msec)	300 (30 msec)	Number of samples in the analysis
M	100 (15 msec)	80 (10 msec)	100 (10 msec)	Number of samples shift between frames
P	8	10	10	LPC analysis order

Q	12	12	12	Dimension of LPC derived cepstral vector
K	3	3	3	Number of frames over which cepstral time derivatives are computed
L	2*P = 16	2*P = 20	2*P = 20	Truncation level of cepstral distance computation

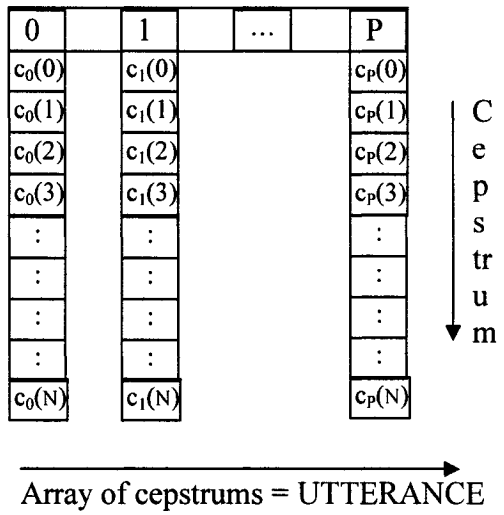
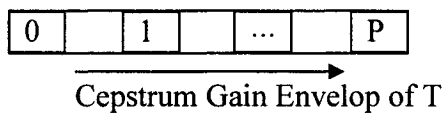
- **Preprocessing output features format**: every speech utterance that has gone through the above operations is converted into an array of cepstrums corresponding to its frames. The result is encapsulated into an object that we called UTTERANCE. All phonemes units in RPKB and every input utterance to recognize are converted into UTTERANCE objects. An UTTERANCE object has the following format in ObjectDefinition 5.1.1 and looks like the illustration given in Table 5.1.2.

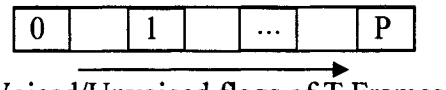
ObjectDefinition 5.1.1: Encapsulation of UTTERANCE object

```
UTTERANCE
{
  /** cepstrum[0..N][1..P] where cepstrum[n] == X[t]
   * that is a cepstrum realized a time t and P is the
   * LPC order. */
   double[1..N] cepstrum[1..P];

  /** The Cepstrum Gain Envelop of this utterance. */
   double cepstrumGainEnvelopProfile[1..N] = null;
```

```
  /** an array of 0=unvoiced or 1=voiced corresponding
   * the frames in this utterance. */
   int voicedUnvoiced[1..N];

   int voicedFlag = UNDERTEMINED_VOICE;
   /** A short name tag for this phoneme model. */
   public String label = "";
}
```

Table 5.1.4: Utterance object

UTTERANCE	
Field	Illustration
cepstrum[0..P][1..N] that is a cepstrum realized a time t and P is the LPC order.	
cepstrumGainEnvelopProfile: the energy envelop of this utterance.	

voicedUnvoiced: an array of 0=unvoiced or 1=voiced corresponding the frames in this utterance.	 Voiced/Unvoiced flags of T Frames
Label: A short name tag for this utterance.	Example: "ae"

With the speech converted into UTTERANCE object feature, the recognition algorithm can be run to do the matching operations using distance evaluation between segments of the input utterance and RPKB units.

Assuming that the features from the preprocessing operation are ready, let us now look at how the proposed algorithms work. In other words, let us move to simulations or testing of the algorithms.

5.2. Simulation strategy for both FTLDP and ParrallelRecognizer

Having identical conditions to test both algorithms via the use of audio files as speech input source, the only source of fluctuation to avoid is the Operation System influence on processing times. In fact, a process can run twice on the same computer with the same configuration but may not result in the same exact processing time value. To do this, we run the algorithms on the same exact conditions several times (`MAX_SIM_ITERATION` defaults to 10 times) to get the average value for the processing time parameter.

We have two techniques (algorithms) but three components to test for speed analysis:

- Fast-Two Level Dynamic Programming (FTLDP). It has two levels (Level 1&2) that were reengineered in Chapter 4.
  - The speed of its Level 1 to evaluate the impact of clustering units in RPKB.
  - The speed of its Level 2 to evaluate the correction of Rabiner's algorithm.

- ParrallelRecognizer

  - The speed of this technique relative to FTLDP.

With the described protocol, we now test the algorithms individually: FTLDP first and ParrallelRecognizer second.

5.3. Test of Fast-Two Level Dynamic Programming (FTLDP)

5.3.1. The speed of the Level 1 of FTLDP to evaluate the impact of clustering units in RPKB

The test in this part consists in, first, running FTLDP on non-clustered RPKB (all the reference phonemes participate in every matching competition) and record its average processing time. Secondly, we cluster RPKB into a fixed number of clusters, run the test and we record the average processing times. The testing algorithm is described as in Algorithm 5.3.1. Note that the first block of Algorithm 5.3.1 is for the variation of the number of clusters (parameter N) which regards Level 1 only and that is presented in this section. The second block is for the variation of the length of the input speech utterance (parameter L) which regards Level 2 only and that will be presented in the next section.

Note: A DP algorithm consists of two levels (Level 1 & 2): Level 1 generates a matrix of best distance or distortion between candidate units of RPKB and the input utterance. Level 2 exploits the matrix to determine the best chains for various chain lengths. Therefore it depends only on the length of the input utterance and not on the size of RPKB unlike the Level 1. For this reason, the test that we have performed is restricted to the Level 1 only, the second level being of a fix cost because we use the same input utterance always.

Algorithm 5.3.1: *Speed performance test algorithm for FTLDP*

Input:

- RPKB as group of all known phoneme utterances;
- Input speech utterance T;

Output:

Time comparison data to plot graphs on FTLDP:

- Clustering effect on FTLDP LEVEL 1 vs TLDP LEVEL 1;
- Effect of shorting utterance T's length on FTLDP LEVEL 2 vs DP LEVEL 2;

```
MAX_SIM_ITERATION = 10;
//-------------------------------------------------------
//--------------------- LEVEL 1 TESTING ------------------
//-------- Variation of the number of clusters -----------
//-------------------------------------------------------

//--------------------- No clustering --------------------
 -  Evaluate the processing time of the Level 1 DP for
    MAX_SIM_ITERATION times;
 -  Get the time average value;

//------ With clustering number [Max, Max-5, .., 5, 1] ---

for different number of clusters
{
   - Cluster(RPKB);
        ▪ Evaluate the processing time of the Level 1 DP for
          MAX_SIM_ITERATION times;
        ▪ Get the average value;
}

//-------------------------------------------------------
//--------------------- LEVEL 2 TESTING ------------------
//-------- Variation of input utterance's length ---------
//-------------------------------------------------------
```

```
for different utterance length
{
   - Compute Level 2 FTLDP and
        ▪ Evaluate the processing time of Level 2 DP for
          MAX_SIM_ITERATION times;
        ▪ Get the average value;

   - Compute Level 2 Rabiner DP and
        ▪ Evaluate the processing time of Level 2 DP for
          MAX_SIM_ITERATION times;
        ▪ Get the average value;
}
```

After running the simulation without and with clustering switch in the first and second step (first block of Algorithm 5.3.1), the third step compares the clustering based technique performance to a non-clustering based technique performance. The comparison is done as expressed in Equation 5.3.1.

$$Time\ \text{Re}duction,\ T_R = \frac{Time\ Level\ 1\ with\ N\ clusters}{Time\ Level\ 1\ without\ clusters} * 100 \qquad (5.3.1)$$

The result obtained is as shown in Figure 5.3.1 where the graph displays the performance as a function of the number of clusters.

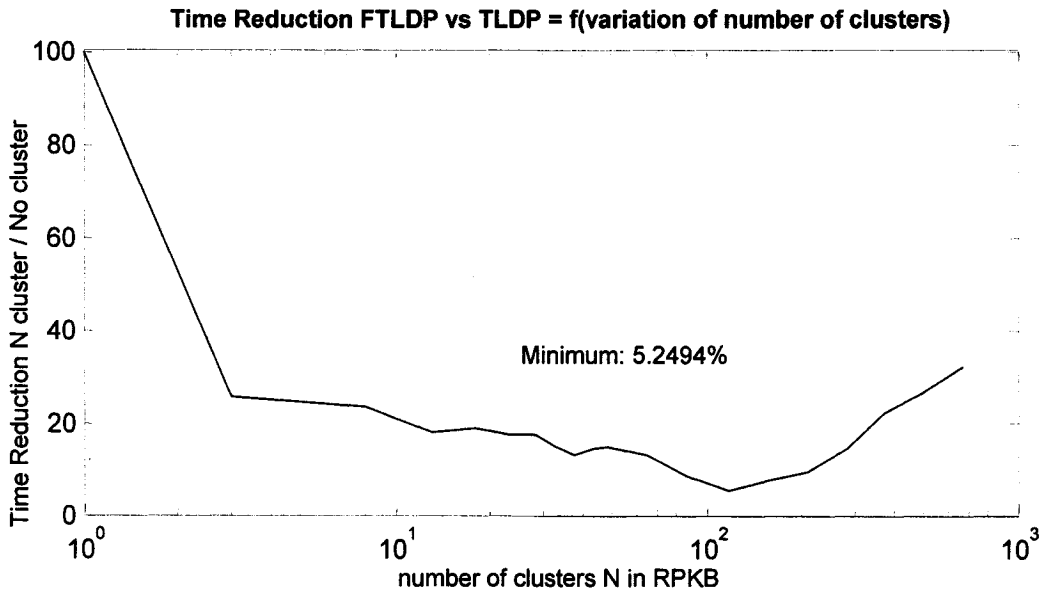


Figure 5.3.1: Results for Level 1 showing processing time reduction as a function of Clustering (number of clusters).

The result of Figure 5.3.1 shows that, the performance of FTLDP-with-clustering is better than without clustering because there is a significant reduction in the processing time of Level 1 when using clustering. The reason is that the clustering structure helps eliminate candidate phoneme models by block of several elements rather than one by one. This makes a significant reduction of the population of the search space thus far smaller than the entire RPKB. The consequence is fewer distance evaluations to process during the comparison.

The analysis of the graph shows that, for a structure of 10 to 100 clusters, this clustering based technique can be between 80 to 95% reduction in the processing time. This equals to $T_R \in [0.05, 0.20]$ ***in processing time. Therefore, Level 1 with a clustering technique is 5 to 20 times faster than without clustering.***

How about the population of the clusters?

Depending on the clustering algorithm used or implemented by a user of our method, the clusters may have difference population. Our suggestion is to have an average population that corresponds to N_{best} even if the number of cluster chosen by the user is different from N_{best}. In the case of our experiments, the graph shows N_{best} = 120 clusters. So, with a total RPKB population of 666 models, we have an average of 6 (=666/120) for the population of the clusters. Because after a cluster is selected among other in the RPKB, the comparison process has a linear cost, it is then reasonable to assume that the optimum population for the clusters is less than 10 models of reference phonemes.

This performance should guarantee a good satisfaction for an English language ASR using nearly 50 clusters of 10 models corresponding to each English phonetic unit.

Fear:

Having too few numbers of clusters does not make significant increase to the speed performance but worst, it can result in large cluster overlapping areas that will lead to difficulties in getting good accuracy performance. On the other hand, using too numerous clusters brings the system out of the optimum zone as the cost of selecting a winning cluster becomes big. In fact, the larger N becomes, the more the configuration looks like having one single cluster (equivalent to no-clustering). This fact is illustrated by the graph in Figure 5.3.1 through the increasing slope for larger N. So, the optimum range can be set as 10 to 100 clusters where a user who will implement our method should choose a value depending on his constraints.

The use of clustering has resulted in a very high increase in speed of Level 1 of FTLDP. Now tackling Level 2, we present in the next section, the result of the correction to Rabiner's Two-Level DP formulas as expressed in Equation 4.1.3.8.

5.3.2. The speed of the Level 2 of FTLDP to evaluate the impact of the correction of Rabiner's algorithm

As we pointed it out earlier, this part of the technique depends only on the length of the input speech utterance to recognize. It does not mean that the Level 1 of a Two-Level DP algorithm does not depend on this length parameter, indeed it does. The reason why our test is restricted to this Level 2 is that, this is the part of the Two-Level DP algorithm proposed in Rabiner's book where we have made some corrections. Therefore it is more convenient to test for the outcome performance of the modifications only.

The result of the test using the second block of Algorithm 5.3.1 and the same input speech utterance T gives the graph of Figure 5.3.2.

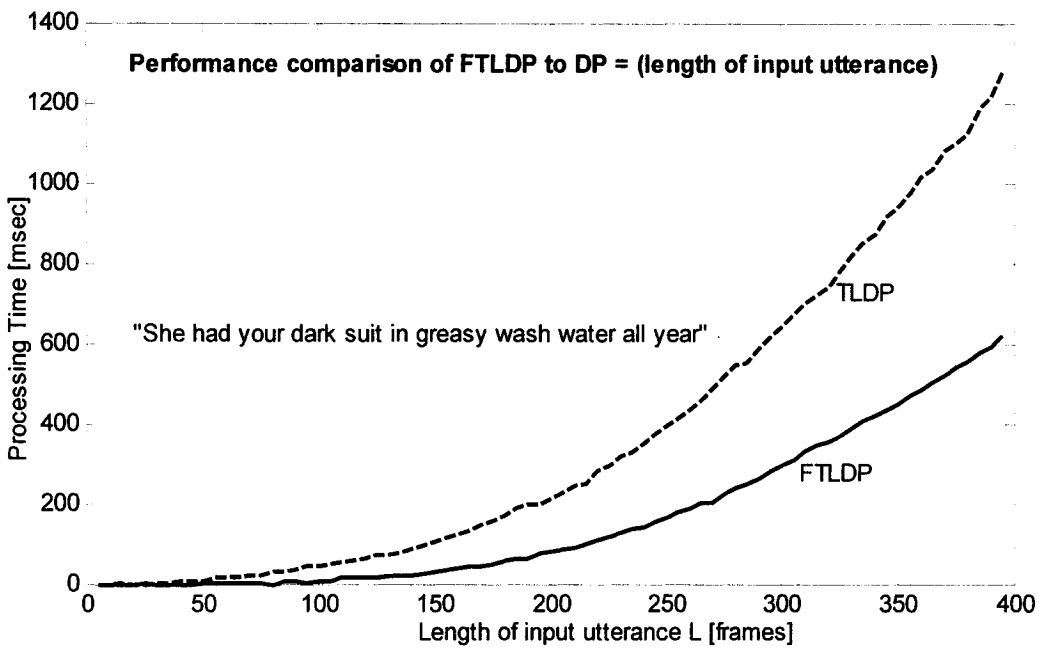


Figure 5.3.2: Results for Level 2 showing the out-performance of FTLDP over TLDP when using the Jump and Slope Analyzer.

As the Figure 5.3.2 shows, the performance of the Level 2 of FTLDP becomes better than TLDP (Rabiner original algorithm) as the length L of the input utterance T becomes

large. Also, it costs no more than TLDP for short input utterances. In fact, for short utterances, there is almost no significant jump to make while it becomes noticeable for long utterances. This result is important for low-level linguistic unit based systems that are likely to deal with large value for the length L (i.e. long chain of phonemes for the utterance T.)

In this chapter, we have presented the Solution.1 using 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 20 times that of the original version (Two-Level DP). This improvement in speed allows us to view as a realistic model the 3-stage processing presented in Figure 1.6.1 as TSA 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 with the gain in speed of the Stage.2 will certainly make the overall ASR system using FTLDP a lot faster. In fact, there is a gain in speed in Stage.3 because it processes string matching rather than utterance distance computation.

Now we continue in the next section with the testing of the second algorithm that we have suggested in Chapter 4 and that is know as ParrallelRecognizer.

5.4. Test of ParrallelRecognizer

The test for this algorithm consists in its comparison to FTLDP. First, we do the test on non-clustered RPKB (all the reference phonemes participate in every matching competition) and we record its average processing time. Secondly, we do the test on clustered RPKB into fixed number of clusters. The testing algorithm is described as in Algorithm 5.4.1.

Algorithm 5.4.1: *Speed performance test algorithm for ParrallelRecognizer*

Input:

- RPKB as group of all known phoneme utterances;
- Input speech utterance T;

Output:

Time comparison data on FTLDP and ParrallelRecognizer for plotting graphs;

```
MAX_SIM_ITERATION = 10;

//--------- No clustering ------------
// Run FTLDP and ParrallelRecognizer MAX_SIM_ITERATION times
// for plotting

- Evaluate the processing time for FTLDP and ParrallelRecognizer
  MAX_SIM_ITERATION times;
- Get their average values;

//--------- Variation of the number of clusters ---------
//With clustering number as [Max, Max-5, .., 5, 1]
//Variation of clusters number

for different number of clusters
{
    - Evaluate the processing time for FTLDP and ParrallelRecognizer
      MAX_SIM_ITERATION times;
    - Get their average values;

}

//--------- Variation of T utterance length ------------
//Variation of T utterance length

for some chosen number of clusters (1, 20, 60, 80)
{
    - Cluster RPKB into N clusters.
```

```
    for different input speech utterance length
    {
      for MAX_SIM_ITERATION times
      {  - Evaluate processing time for FTLDP
         - Evaluate processing time for ParralleleRecognizer
      }
      Get their average values as their performance for this length;
    }
}
```

Running Algorithm 5.4.1, we got the results plotted in Figure 5.4.1&2. With respect to variation of the number of clusters N, Figure 5.4.1 shows good performance.

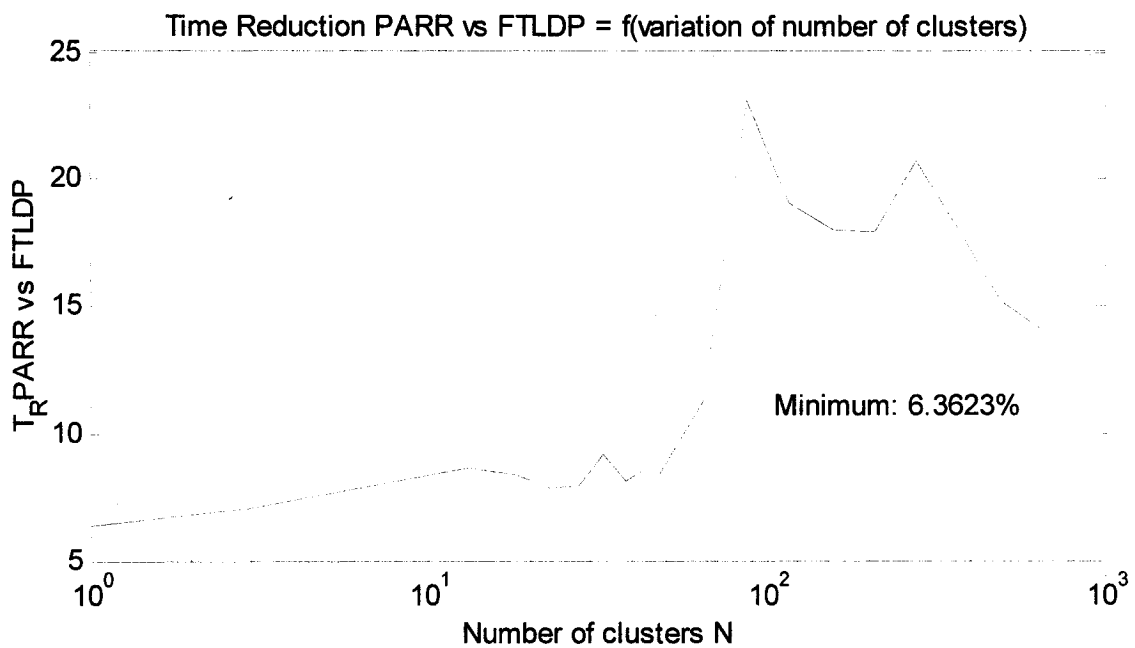


Figure 5.4.1: Comparison graph between FTLDP and ParrallelRecognizer as a function of N the number of phoneme clusters in RPKB. PARR = ParrallelRecognizer and T_R = time reduction.

The graph in Figure 5.4.1 shows that ParrallelRecognizer algorithm has a processing time reduction of 77 to 94% compared to FTLDP on the range of number of clusters from 1 cluster (no-clustering) to 100 clusters. This is 4 to 16 times faster than FTLDP. Therefore, the overall speed increase compared to the standard Two-Level DP algorithm is 320 times as shown in Equation 5.4.1.

$$\left.\begin{aligned} T_{R_{Parra-TLDP}} &= \frac{Time\ Parrallel\ \mathrm{Re}cognizer}{Time\ FTLDP} \\ &= \frac{Parra}{FTLDP} * \frac{FTLDP}{TLDP} = 0.05 * 0.0636 = 0.00318 \\ &= 99.7\%\ time\ reduction \\ &\Leftrightarrow 0.05^{-1} * 0.0636^{-1} \approx 20 * 16 = \quad 320\ faster \end{aligned}\right\} \quad (5.3.1)$$

The reason of this performance is that the clustering structure helps eliminate candidate phoneme models by block of several elements rather than one by one. This makes a significant reduction of the population of the search space thus far smaller than the entire RPKB. The consequence is fewer distance evaluations to process during the comparison. This result also shows that with no clustering (or small number of clusters, thus larger cluster populations), ParrallelRecognizer is extremely fast compared to FTDLP. This is because the performance of FTLDP improves as N gets larger thus the clusters population becomes smaller (See Figure 5.3.1). The ending decreasing slope of the graph (Figure 5.4.1) corresponds to the increasing slope shown in the performance of FTLDP in Figure 5.3.1. So the peak shown the performance of ParrallelRecognizer corresponds to the valley in the performance of FTLDP in Figure 5.3.1 which meant better performance of FTLDP.

With respect to shorting the length of the input speech utterance (a consequence of removing silence segments), Figure 5.4.2 shows a good performance in general though the values depend on the number of clusters.

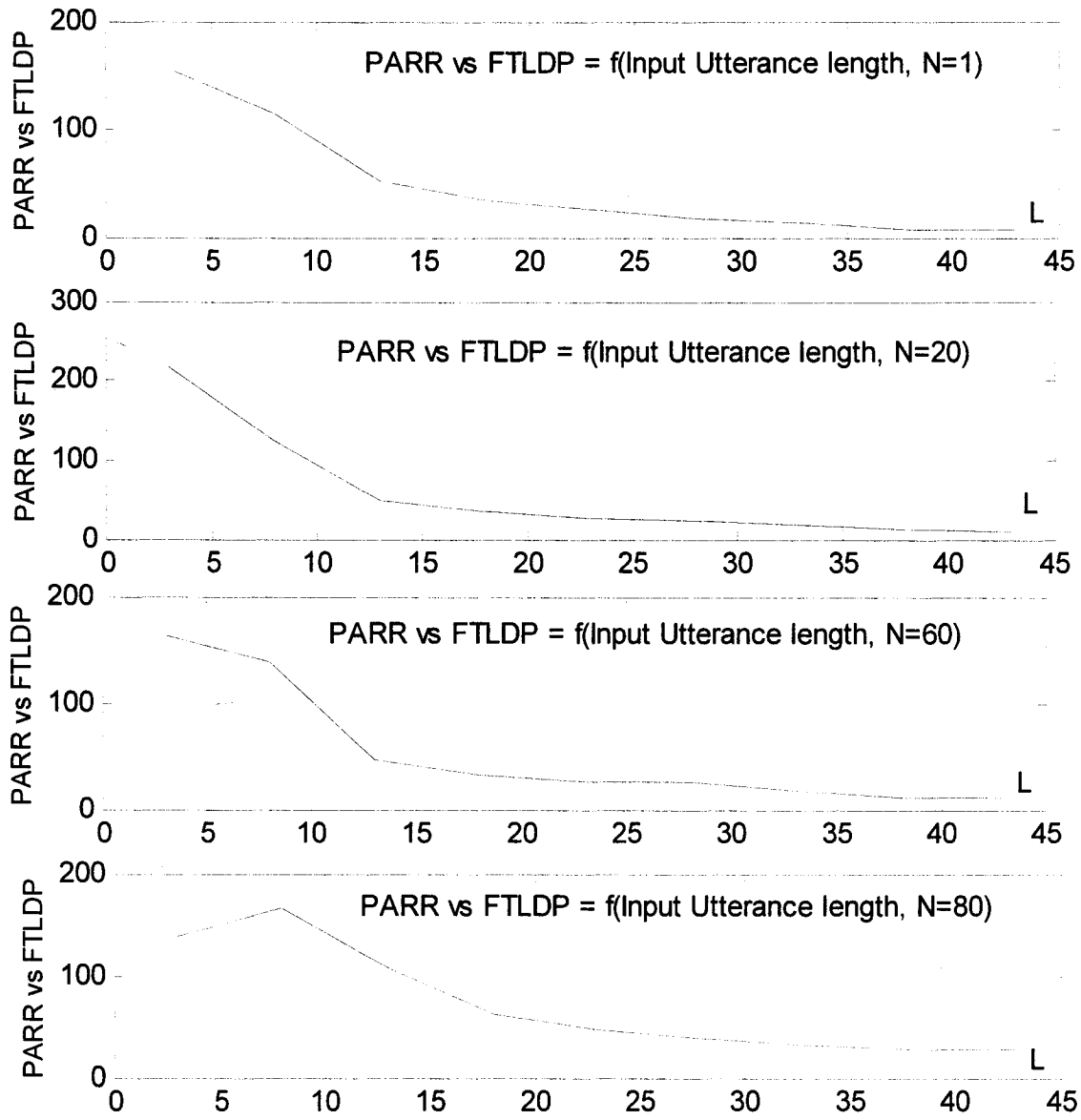


Figure 5.4.2: Comparison graph between FTLDP and ParrallelRecognizer as a function of the length L of the input speech utterance T. PARR = ParrallelRecognizer.

As shown in Figure 5.4.1, the performance of ParrallelRecognizer depends on the number of clusters. For this reason, we graph the characteristic of this algorithm for different number of clusters as shown in Figure 5.4.2 to better appreciate how it behaves. The analysis of the resulting graphs shows that FTLDP is preferable to ParrallelRecognizer

for very short input utterances but ParrallelRecognizer becomes quickly efficient with increase in the length of the input utterance. As predicted in Chapter 4, the longer the input speech utterance T is, the larger the processing of the matrices $\tilde{D}(b,e)$ and $\tilde{N}(b,e)$ get as their size becomes big.

5.5. Conclusion of the experiments

To this point of the thesis, we have created a Reference Phoneme Knowledge Base (RPKB) and we have provided detail about the two techniques or algorithms that are the main contribution of this work. Also, we suggested some heuristic methods that were used in the above-mentioned algorithms. The results of the test of these algorithms have shown great performance in speed as this chapter has shown them.

Exploiting the concept of Clustering, Jump Function and Slope Analyzer, the FTLDP algorithm is 5 to 20 times faster than the Two-Level DP as formulated by Rabiner [8]. This performance is valid for a number of clusters in the range 10 to 100 clusters assuming that the ASR system uses our designed RPKB.

Using the concept of clustering combined with CEGP heuristics, the ParrallelRecognizer algorithm is 4 to 16 times faster than the Two-Level DP as formulated by Rabiner [8]. ParrallelRecognizer is guaranteed to be faster than FTLDP for any number of clusters. In fact, it has its best performance over FTLDP when clustering is not used. The weakest performance of ParrallelRecognizer over FTLDP corresponds to the range of the number of clusters (30 to 300) where FTLDP scores its best performance.

Using a different reference phoneme database (RPKB larger or smaller in population) should not change the global shape of the performance except that the performance numbers may vary slightly.

The next step is the discussion of the paradigm of our work and the conclusion of the thesis.

CHAPTER-6

DISCUSSION AND CONCLUSION

We have situated ourselves within a framework of phonetic-based continuous speech recognition. We have established one overall 3-stage-architecture that we called Three Stage Architecture (TSA) and we have set one general goal of speed improvement. That improvement can be at different places within the process, and we have explored many of these places.

One issue we have not addressed and that we wish to come back to is the notion of accuracy. Since we did mention it in the introduction that much effort is done in the research community on accuracy, we will talk about it here in section 6.1. In 6.2 we will summarize how we have achieved our goals and restate our contributions. In 6.3 we look at future work. We finish in section 6.4 by looking at the challenges awaiting in developing speech recognition.

6.1. Accuracy of the suggested techniques

The two techniques (FTLDP and ParrallelRecognizer) that we have suggested are shown to be good for reducing the computational load of the recognition processes thus they help to speed them up. The important question is what happens to the accuracy factor. As it is common in engineering to trade off a parameter in the benefit of another one, the wonder is to know if the accuracy factor has been sacrificed for speed in these suggested algorithms.

The answer is no. In fact, the question of accurately decoding an input utterance into text depends on two factors:

1- The capability to delimitate boundaries for a segment of the input utterance that is appropriate to match singularly the best phoneme unit in the language knowledge database (RPKB).
2- The availability of tools that can clearly differentiate distances or distortions among linguistic entities in their encoded-speech-features (spectrums or cepstrums). In other words, the quality of the comparison evaluation.

The schema in Figure 6.1.1 shows the different modules that cooperate to do the decoding of an input speech utterance into text as we conceive it throughout this thesis.

- The modules "DSP & features extraction" encode the raw input speech signal or reference phoneme signal into features (cepstrums). This corresponds to the Stage.1 of the TSA structure.
- The module "Pairing Operator" does the pairing operation between the different units in RPKB and all the possible segments of the input speech utterance T using the illustrating parameters Tp (segment of T) and U (model in RPKB). The variation of these two parameters can help, in theory, do all possible pairing based on combination principle from T to RPKB. We will describe more this module further down in this section.
- The module "Tp and U generator" provides to the module "Pairing Operator" numerical values for Tp and U. Normally, Tp and U are independent parameters. However, this module can establish a certain relationship between them for the sake of optimizing the algorithm. Mainly, the function is to intelligently skip useless pairs.
- The module "Comparator: distance (Tp, U) evaluation" is the one responsible to accurately distinguish the similarities among pair of utterances. This is the module whose performance has a heavy weight on the overall ASR system's accuracy. The more this module can clearly determine the similarity between utterances, the more accurate the "Decision Maker" module output will be. This similarity might not use solely distance between utterances but other parameters too. However, in this thesis we used only distance as parameter.

- The module "Decision Maker" uses the comparison result (distance value) output by "Comparator: distance (Tp, U) evaluation" module to select the appropriate phoneme unit in RPKB as the winner.

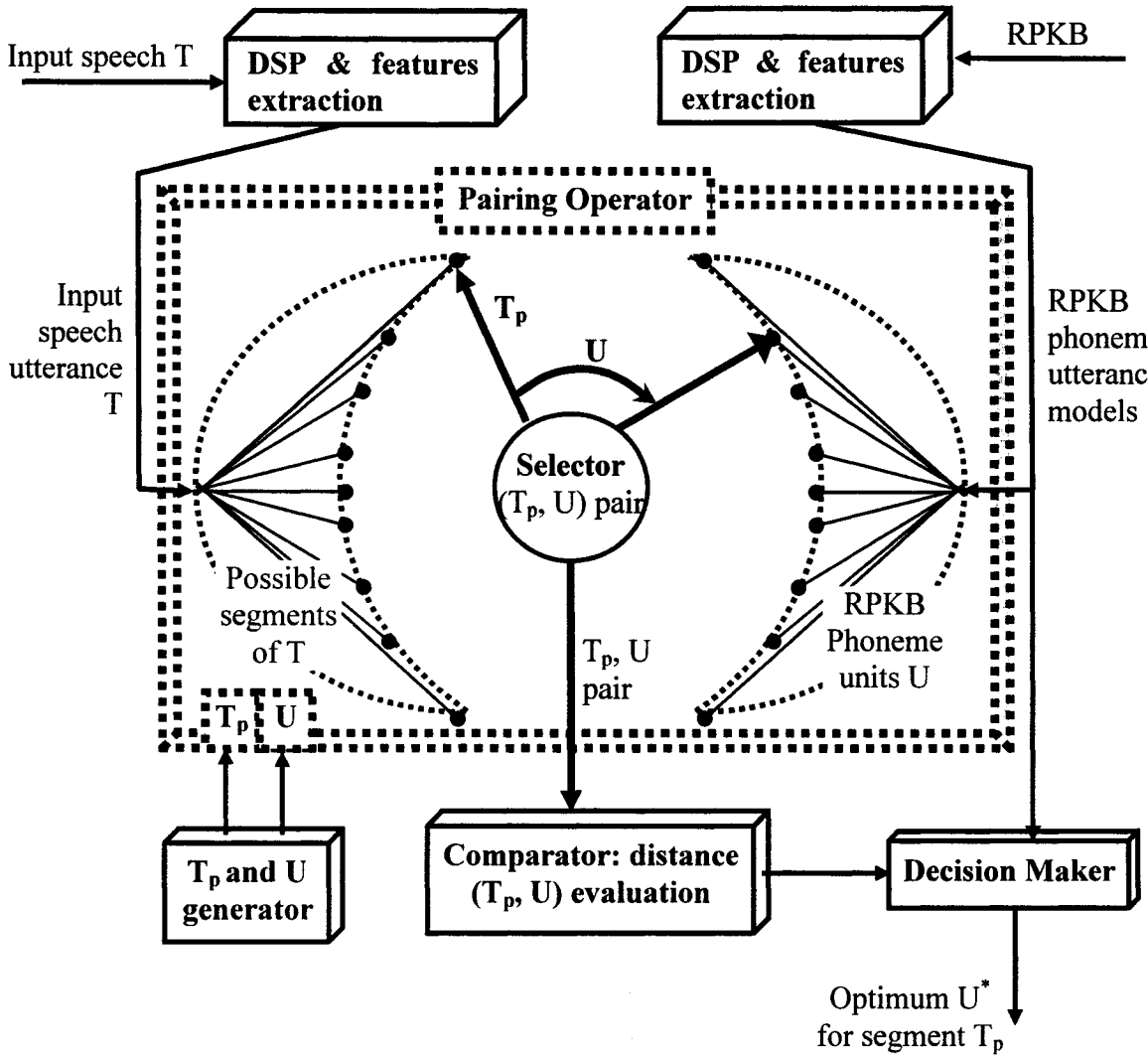


Figure 6.1.1: Illustration of an ASR machine using TSA with detail of Stage.2.

In the diagram 6.1.1, all the modules except "DSP & features extraction" are components of Stage.2 of the TSA. "DSP & features extraction" belong to Stage.1. Components in Stage.3 are not shown in this schema (see section on future works in this chapter).

Now, let us come back to more explanation of the module "Pairing Operator".

In the illustration that this diagram provides, the "Selector" reacts to the values of Tp and U to pair a segment of the input speech utterance T with a phoneme unit U in the RPKB. In a very exhaustive way, the module "Tp and U generator" is not needed since the selector can just operate all possible combinations. This is what a Two-Level DP does, consequently so does FTLDP.

Certainly, it is not all of the possible combination pairs that are relevant. In fact, lots of them are not. Therefore the question is how to efficiently generate only the relevant pair values (Tp, U)? This is where the generator becomes important because it encapsulates the generation of values to these two parameters. It can host a function that knows how to relate Tp to U efficiently in the sense that only useful pairs are generated. For example, as described in the section on SEGMENT Objects, the length of the segment Tp might be a function of U. In this case, a whole lot of other possible segments are disqualified thus the search space is reduced. This Tp and U generator" module is the heart or the secret of the performance of the two techniques we have suggested, FTLDP and ParrallelRecognizer.

In FTLDP, the clustering technique does the job of the generator function and in ParrallelRecognizer, the automatic determination of the length of SEGMENT object's corresponds to the generator function in addition to clustering.

Assuming that this diagram gives the best illustration of the components in the whole architecture of Stage.2, let us return to the problem of accuracy factor.

A careful analysis of the diagram should point out that all the machinery made of "Tp and U generator" and "Pairing Operator" modules should have almost no effect on accuracy. The module responsible for this is "Comparator: distance (Tp, U) evaluation" where every pair built in the previous modules scores a distance value. If this comparison score can singularly quantify the similarity and closeness of Tp to U, thus it is possible to get a very good accuracy performance.

It is difficult to compare our performance to other scientists' results on speed and accuracy. If for the accuracy factor, it is possible to compare with another result how many phonemes have been correctly recognized, for the speed factor, it is irrelevant unless the two results compare respectively algorithm X and algorithm Y to a same baseline algorithm B. Ideally, it should be interesting to develop, implement and test a the speed performance of a new algorithm with respect to a baseline algorithm that would be one used in the best (state of the art) commercial application. Unfortunately, these commercial applications have proprietary techniques that are difficult to get, work on and make them public.

For the accuracy factor, we can give some results as we got them in the experimentations of Chapter 5. We tested the ASR system that we have implemented with several continuously spoken sentences taken in TIMIT and we got good recognition results with the reference phoneme models RPKB. The results are appreciated as good based on two methods that we have used to evaluate the accuracy performance of the algorithms.

First, we use audio file reconstruction to evaluate by audition the similarity between the results and the input speech. Using the phoneme models of the recognition output (string of phoneme symbols), we were able to reconstruct by concatenation, an audio WAVE file that we can play and evaluate the quality of the synthesis.

Secondly, we compare the original transcription of the input speech into string of phoneme symbols to the output results of our algorithms. In this process, we try to get some percentage numbers but the values were irrelevant without the processing of this string of symbols into words. Even if we get these percentage numbers, it doe not make sense to compare them to other researchers' work unless we have the same method of judging the results. In fact, if we were using words as models, it is easy to count the correct recognized words, insertion, deletions and inversions. At phoneme level, the problem is not simple due to numerous insertions and inversions. In fact, if we were to recognize the phrase "dirty joke" whose phonemic transcription is "dcl d er dx iy SIL dcl j how kcl k", we have 11/2 * 100 = 550% chance to do wrong than if we were to recognize 2 words only.

The only way, an acceptable accuracy evaluation can be gotten is to have the Stage.3 run its linguistic processing to convert the string of phonemes into words by detecting and eliminating insertions and possibly recovering deletions. Nonetheless, we show some results in the Table 6.1.1. We judge the quality of these results by simply analyzing the patterns in them, to discover how close the sound of a given string is to the original speech.

Table 6.1.1: Some typical recognition output results of our testing ASR
Phrase: "She_had_your_dark_suit"
Expected transcription
SIL - sh - iy - hv - ae - dcl - d - y - er - dcl - d - aa - r - kcl - k - s - uw – dx – SIL
Results obtained using ParrallelRecognizer, 80 clusters.
Phrase 0: zh - jh - ah - pcl - hh - epi - uh - epi - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 1: zh - jh - ah - pcl - hh - epi - uh - epi - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 2: zh - jh - ah - pcl - hh - epi - uh - epi - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 3: zh - jh - ah - pcl - hh - epi - en - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 4: zh - jh - ah - pcl - hh - epi - ng - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 5: zh - jh - ah - pcl - hh - epi - l - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 6: zh - jh - ah - pcl - hh - epi - ux - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 7: zh - jh - ah - pcl - hh - epi - l - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 8: zh - jh - ah - pcl - hh - epi - uh - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g Phrase 9: zh - jh - ah - pcl - hh - epi - l - b - epi - ow - b - aa - ah - g - k - q - p - ch - pau - en - g
Results obtained using ParrallelRecognizer, No clustering.
Phrase 0: zh - hh - pau - oy - ey - oy - pau - s - ey - g Phrase 1: zh - pau - oy - ey - gcl - oy - pau - s - ey – g Phrase 2: f - pau - oy - l - oy - pau - s - ey - g Phrase 3: zh - pau - oy - ey - oy - pau - s - ey - g Phrase 4: f - pau - hh - oy - gcl - oy - pau - s - ey - g

Phrase 5: f - pau - oy - l - oy - pau - s - ey - g

Phrase 6: f - pau - hh - oy - gcl - oy - pau - s - ey - g

Phrase 7: th - zh - pau - oy - ey - oy - pau - s - ey - g

Phrase 8: f - pau - hh - oy - gcl - oy - pau - s - ey - g

Phrase 9: f - pau - oy - l - oy - pau - s - ey - g

Results obtained using FTLDP, 80 clusters.

Phrase 0: zh - hh - ix - pcl - pcl - aa - ay - ey - q - aa - aa - g - k - q - epi - s - pau - q – g

Phrase 1: zh - hh - ix - pcl - pcl - aa - ay - ey - q - aa - aa - g - k - q - epi - s - pau - g

Phrase 2: zh - hh - ix - pcl - pcl - aa - ay - epi - ow - q - aa - aa - g - k - q - epi - s - pau - q - g

Phrase 3: zh - hh - ix - pcl - pcl - aa - ay - ey - q - aa - aa - g - k - q - epi - s - pau

Phrase 4: zh - t - hh - ix - pcl - pcl - aa - ay - epi - ow - q - aa - aa - g - k - q - epi - s - pau - q - g

Phrase 5: zh - hh - ix - pcl - pcl - aa - ay - ey - aa - aa - g - k - q - epi - s - pau

Phrase 6: zh - t - hh - ix - pcl - pcl - aa - ay - epi - ow - aa - aa - p - p - g - k - q - epi - s - pau - q - g

Phrase 7: zh - hh - ix - pcl - pcl - aa - ay - epi - aa - k - w - aa - aa - p - p - g - k - q - epi - s - pau - q - g

Phrase 8: zh - hh - ix - pcl - pcl - aa - ay - ey - aa - aa - k - q - epi - s - pau

Phrase 9: zh - t - hh - ix - pcl - pcl - aa - ay - epi - aa - k - w - aa - aa - p - p - g - k - q - epi - s - pau - q - g

Phrase 10: zh - hh - q - pcl - hh - ay - ey - aa - aa - k - q - epi - s - pau

Phrase 11: zh - hh - ix - pcl - pcl - aa - ay - ey - aa - aa - epi - s - pau

Phrase 12: zh - hh - q - pcl - hh - ay - ey - aa - aa - epi - s - pau

Phrase 13: zh - hh - q - pcl - hh - ay - ey - aa - epi - s - pau

Phrase 14: zh - ah - pcl - hh - ay - ey - aa - epi - s – pau

Phrase 15: zh - ah - pcl - hh - ay - ey - aa - ch - pau

Phrase 16: zh - ah - pcl - hh - ay - ey - ch - pau

Phrase 17: zh - hh - ay - ey - aa - ch - pau

Phrase 18: zh - hh - ay - ey - ch - pau

Phrase 19: zh - hh - ay - pau - pau

Phrase 20: zh - aa - pau - pau

Phrase 21: zh - aa - pau

Phrase 22: zh - hh

Results obtained using FTLDP, No clustering.

Phrase 0: zh - pau - pau - oy - oy - pau - s - oy

Phrase 1: f - pau - oy - dh - oy - oy - pau - s - oy

Phrase 2: f - pau - oy - oy - n - m - oy - pau - s – oy

Phrase 3: f - pau - oy - oy - oy - f - oy

Phrase 4: f - pau - oy - dh - ey - n - m - oy - pau - s – oy

Phrase 5: f - pau - oy - dh - dh - oy - n - m - oy - pau - s - oy

Phrase 6: zh - pau - pcl - oy - dh - dh - oy - n - m - oy - pau - s - oy

Phrase 7: zh - pau - pcl - oy - dh - dh - oy - n - m - oy - pau - s - ey - g

Phrase 8: zh - pau - pcl - oy - dh - dh - oy - n - m - oy - pau - s - ay - ix - g

Phrase 9: zh - pau - pcl - oy - dh - dh - oy - n - m - oy - pau - s - ah - uh - ix – g

Phrase 10: zh - pau - pcl - oy - dh - dh - oy - n - m - oy - k - k - pau - f - ey - ix - g

Phrase 11: f - pau - oy - oy - f - oy

Phrase 12: zh - zh - hh - v - pcl - oy - dh - dh - oy - n - m - oy - pau - s - ah - uh - ix - g

Phrase 13: zh - zh - hh - v - pcl - oy - dh - dh - oy - n - m - oy - k - k - pau - f - ey - ix - g

Phrase 14: zh - zh - hh - v - hh - ay - q - dh - dh - oy - n - m - oy - k - k - pau - f - ey - ix - g

Phrase 15: zh - zh - hh - v - pcl - hh - ay - ax - dh - oy - ax - m - gcl - aa - oy - f - pau - v - q - ix - g

Phrase 16: zh - zh - hh - v - hh - ay - q - dh - dh - oy - n - m - oy - k - k - pau - f - em - v - q - ix - g

Phrase 17: zh - zh - hh - v - pcl - hh - ay - ax - dh - oy - ax - m - gcl - oy - k - k - pau - f - em - v - q - ix - g

Phrase 18: zh - zh - hh - v - hh - ay - q - dh - dh - th - ah - ax - m - gcl - oy - k - k - pau - f - em - v - q - ix- g

Phrase 19: f - oy - oy - pau - pau

Phrase 20: pau - oy - pau - pau

Phrase 21: pau - oy - pau

Phrase 22: pau - pau

Phrase 23: pau -

To show the evidence of the strong dependency of the recognition accuracy on the precise evaluation of distances in the module "Comparator: distance (Tp, U) evaluation", we made a test that we report the results in the following lines. For this test, we use the same input speech utterance "She_had_your_dark_suit" that we cut into 5 pieces. We include these pieces in the RPKB and run the ARS for results using ParrallelRecognizer. The result of the functional sorted segment piles is shown in Table 6.1.2 followed by some output results in Table 6.1.3.

Table 6.1.2: Top portion of the sorted Segment object piles in ParrallelRecognizer algorithm. *The remark is that the performance of the distance computation module is insufficient to clearly bring up to the top of a pile the appropriate piece for a given frame position.*

startFrame= 0	endFrame= 27	distortion= 0.05994323726146143	she
startFrame= 0	endFrame= 35	distortion= 16.445651813956733	suit
startFrame= 0	endFrame= 30	distortion= 18.28930735342259	oy
startFrame= 0	endFrame= 26	distortion= 23.384182194261847	dark
...			
startFrame= 26	endFrame= 28	distortion= 31.738925531845787	k
startFrame= 27	endFrame= 53	distortion= 14.664313197893277	dark
startFrame= 27	endFrame= 57	distortion= 17.67091160407614	oy
startFrame= 27	endFrame= 62	distortion= 21.335791779408552	suit
startFrame= 27	endFrame= 48	distortion= 13.129994478064592	had
startFrame= 27	endFrame= 46	distortion= 15.709804939707483	er
startFrame= 27	endFrame= 47	distortion= 16.79590040508181	oy
startFrame= 27	endFrame= 54	distortion= 24.521703310580126	she
startFrame= 27	endFrame= 43	distortion= 15.054094113848459	aw
...			
startFrame= 27	endFrame= 29	distortion= 34.22147261624421	k
startFrame= 28	endFrame= 58	distortion= 17.14300149913577	oy
startFrame= 28	endFrame= 63	distortion= 20.65404741154269	suit
startFrame= 28	endFrame= 54	distortion= 15.715629401390421	dark
startFrame= 28	endFrame= 49	distortion= 12.967708044619409	had
...			
startFrame= 47	endFrame= 49	distortion= 32.10079877761076	k
startFrame= 48	endFrame= 66	distortion= 5.881369222985097	uw
startFrame= 48	endFrame= 63	distortion= 4.969123416554527	your
startFrame= 48	endFrame= 67	distortion= 6.237660843556192	er
...			
startFrame= 52	endFrame= 53	distortion= 20.429296020287094	d
startFrame= 53	endFrame= 69	distortion= 5.896876652559351	ey
startFrame= 53	endFrame= 68	distortion= 5.605218146890687	uw
startFrame= 53	endFrame= 69	distortion= 5.961610473774912	ey
...			
startFrame= 62	endFrame= 64	distortion= 28.617963611196394	jh
startFrame= 63	endFrame= 65	distortion= 0.05177660401505797	nx
startFrame= 63	endFrame= 64	distortion= 0.04750689675875705	b
startFrame= 63	endFrame= 64	distortion= 0.054454794614570534	b
startFrame= 63	endFrame= 64	distortion= 0.059772816446918973	b
startFrame= 63	endFrame= 64	distortion= 0.09079555113082122	b
startFrame= 63	endFrame= 65	distortion= 0.13705376270809355	gcl
startFrame= 63	endFrame= 64	distortion= 0.10709028710158017	b
startFrame= 63	endFrame= 65	distortion= 0.28861159408385373	dx
startFrame= 63	endFrame= 84	distortion= 2.1948266985607923	had

```
startFrame= 63      endFrame= 65 distortion= 0.30132291022329666   gcl
...
startFrame= 63      endFrame= 65 distortion= 29.279151721820448    jh
startFrame= 64      endFrame= 65 distortion= 0.0656375932832746    b
startFrame= 64      endFrame= 85 distortion= 1.0059086101606294    had
startFrame= 64      endFrame= 65 distortion= 0.3747728786421937    b
startFrame= 64      endFrame= 65 distortion= 0.695471533912798     b
startFrame= 64      endFrame= 65 distortion= 0.697724828188054     b
startFrame= 64      endFrame= 65 distortion= 0.7105149687002803    b
startFrame= 64      endFrame= 65 distortion= 0.749739042765658     b
startFrame= 64      endFrame= 90 distortion= 13.244608200528498    dark
startFrame= 64      endFrame= 94 distortion= 18.149476326737886    oy
startFrame= 64      endFrame= 99 distortion= 23.379044758038603    suit
...
startFrame= 64      endFrame= 66 distortion= 27.436524116644808    jh
startFrame= 65      endFrame= 86 distortion= 7.57994989352738      had
startFrame= 65      endFrame= 91 distortion= 16.047350153826883    dark
startFrame= 65      endFrame= 95 distortion= 19.120871802238028    oy
startFrame= 65      endFrame= 100     distortion= 23.2599286467817      suit
...
startFrame= 65      endFrame= 66 distortion= 23.2286442643681      d
startFrame= 66      endFrame= 87 distortion= 11.747931145940322    had
startFrame= 66      endFrame= 84 distortion= 12.154329013480208    oy
startFrame= 66      endFrame= 96 distortion= 19.921800121227616    oy
startFrame= 66      endFrame= 101     distortion= 23.478036550876016    suit
startFrame= 66      endFrame= 79 distortion= 9.15201709013755      aw
startFrame= 66      endFrame= 92 distortion= 18.51152609333922     dark
...
startFrame= 68      endFrame= 70 distortion= 36.13490747161404     jh
startFrame= 69      endFrame= 87 distortion= 12.40361692611946     oy
startFrame= 69      endFrame= 104     distortion= 24.222216420165758    suit
startFrame= 69      endFrame= 82 distortion= 9.592415261513628     aw
...
startFrame= 88      endFrame= 89 distortion= 27.082631190907428    q
startFrame= 89      endFrame= 121     distortion= 11.302618283097162    suit
startFrame= 89      endFrame= 119     distortion= 18.347281354516717    oy
startFrame= 89      endFrame= 116     distortion= 17.174712673063908    she
startFrame= 89      endFrame= 115     distortion= 26.23230071290182     dark
...
startFrame= 90      endFrame= 91 distortion= 31.87506415778652     q
startFrame= 91      endFrame= 121     distortion= 15.30037313116736     suit
startFrame= 91      endFrame= 121     distortion= 16.33424449340813     oy
startFrame= 91      endFrame= 118     distortion= 16.142188051659847    she
startFrame= 91      endFrame= 117     distortion= 24.232379858289303    dark
startFrame= 91      endFrame= 104     distortion= 12.954534275032758    SIL
```

Using the connections in Table 6.1.2, we can deduct several recognition solutions with some small corrections as summarized in Table 6.1.3.

Table 6.1.3: Illustration of the reason why recognition accuracy rates are irrelevant in our speed paradigm unless we design a good distance computation module.

Piece	She	had	your	dx	dark	suit
Segments	0-27	27-48	48-63	63-65	65-91	91-121
Sorting adjustment to bring the piece to the top of a pile	0	+4	+1	0	+2	0
If the connections were parsed naturally (top of piles directly), we would get the following result that is our ASR's						
Piece	she	dark	ey	oy	suit	Accuracy = 60%
Sorting adjustment to bring the piece to the top of a pile	0	0	0	0	0	

The comments that we have presented in this section was to illustrate why it is irrelevant in the paradigm of the investigation we made on the speed issue of an ASR, to evaluate the accuracy factor. This is possible only if we had spent time on designing a good "Comparator: distance (Tp, U) evaluation" module (Figure 6.1.1) and if we have completed the TSA with the design of the Stage.3. The design of the Stage.3 would be necessary only if we were to investigate both accuracy and speed issues in ASR. Also, information in Table 6.1.1 shows that, using ParrallelRecognizer, Stage.3 has uncountable proposals to process for an output solution.

6.2. Summary of achievements and contributions

In this thesis, we have set two goals to achieve from the introduction in Chapter 1.

Goal.1 Design an ASR system that would be speaker-independent (no context parameter involved). The solution should use an acoustic-phonetic approach

to make the setup of the recognizer easy prior to its use (no training) and context-less.

Goal.2 Design an ASR that does continuous speech recognition very fast. This goal was set as the main objective in this thesis, the speed issue (reduce the computation power).

We have come to good and satisfactory achievements of these goals with the results of the different modules and techniques that we have presented in this thesis, especially in chapter 2, 3 and 4. The modules of the TSA in which we have made some contributions are shown in Figure 6.2.1. The modules are:

- The *Reference Phoneme Knowledge Base (RPKB)* built from clustering a trusted phonetic-acoustic database called TIMIT. The resource is a reference of phonetic-acoustic-phoneme models. It gives then the capability of context-independent to a baseline ASR system.

  The phoneme-clustering module of chapter 2 that was used to build the RPKB is capable of finding a small set of phonemes for each ARPABET phonetic unit that can be used as representative realizations for each phoneme. The interesting result of this module is the good dialects mixture that can be found in the resulting small cluster. This can relate well to other method [29] found in the literature that can be used to substitute ours and that are shown to successfully cluster phonemes from several languages such as French, English, German, Spanish (IDEAL corpus).

- The two heuristic methods called the *Cepstrum Gain Envelop Profile (CGEP)* and the *CGEP or Cepstrum derivative based Silence Detection.*

  The originality of our contribution to the field of Silence Detection is the use of our invented features called CGEP. In fact, for the purpose of detecting silent segments from the input speech, there is no need to involve huge and complicated calculations. Instead, CGEP features that are by-products of cepstrums computation are sufficient as shown in chapter 2 on heuristics. At worst, the cost of having noisy

speech inputs would be the failure of CGEP to reduce the length of the input speech, thus a small drop in speed performance.

The utterance comparison techniques that were presented as **Solution.1** and **Solution.2** are:

- The *Fast-Two-Level-Dynamic-Programming (FTLDP)* that is a modified version the standard Two-Level DP algorithm through the correction of its analytical formulation, the use of clustering, and the silence skipping. Two-Level-Dynamic-Programming (TLDP) being a widely used algorithm in ASR systems, the good speed performance that we have got hear is a major contribution in the domain. Any ASR system that uses TLDP can upgrade to FTLDP and make its low-level recognition unit 20 times faster. We have a paper to present on this FTLDP algorithm in the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004) that will be held in May 17-21, 2004 in Montreal, Canada.
- The *ParrallelRecognizer* is our original suggestion that is shown to be an extremely fast algorithm compared to FTLDP. This algorithm is the best of the contributions that we have made to the ASR world. It can substitute TLDP algorithm in any ASR and make it 320 times faster with the advantage of having numerous output solutions to be processed using NLP. It uses less memory than FTLDP. We plan to submit a paper on this algorithm in the Canadian Acoustics Association (CAA) meeting in Ottawa (6-8 October, 2004): www.caa-aca.ca/Ottawa2004/index.htm.

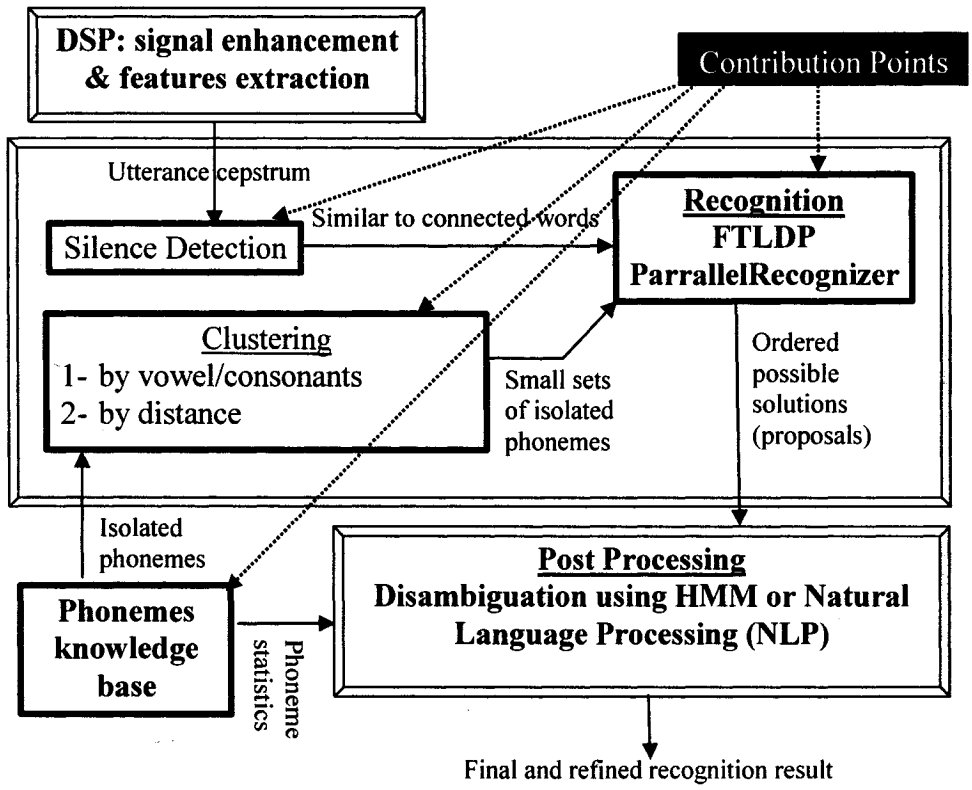


Figure 6.2.1: TSA block diagram with the contribution focus points highlighted.

On the speed issue, the FTLDP is shown to be more than 20 times faster than the standard Two-Level DP algorithm and the ParrallelRecognizer extremely fast compared to FTLDP (16 times) or 320 times compared to the standard Two-Level DP. Both of these techniques use cepstral coefficients as speech features to do the recognition as well as the heuristic modules addable to them.

The consequence of this single-feature set is that, any recognizer using our algorithms will start faster (no training and few features to evaluate) than many other pattern comparison commercial applications that we have tried (IBM ViaVoice and FreeSpeech of Philips).

The combination of the two techniques with the phoneme clustering module has the potential to be easily configured to process recognition operations in several languages if

we assume the point of view of Mareüil and Corredor-Ardoy that it is possible to replace several phonetic recognition systems by a single, common system for all language [30].

The investigation of this thesis should be viewed as an exploration of the whole scope of performance in speed of an ASR system but not as a specific implementation design of an ASR. In this sense, we have covered a large of possibilities and factors that might be used to increase the speed of an ASR. Therefore, we have found a range for the number of clusters to be used and also a minimum length for the input speech utterance to consider for recognition using our best algorithm, ParrallelRecognizer. So, it is up to anybody willing to use this technique to determine specific values for the parameters according to his constraints.

The accuracy factor depends on how efficient the clustering system implemented by a user of our method reduces the overlaps of clusters and the comparison evaluation as shown in "Comparator: distance (Tp, U) evaluation" in Figure 6.1.1. Any reverberance or any speech corruption problem at the input speech should be dealt with in the Stage.1 of the TSA. It is important to notice that any problem related to the accuracy factor should not impact largely the speed of our design for this Stage.2 except that the disambiguation at Stage.3 would become more difficult to assure the constancy of the accuracy factor.

6.3. Future work

This thesis has extensively presented the components that form the Stage.2 of the Three-Stage Architecture (TSA) that we suggest to help improve the speed of a phonetic-acoustic based ASR.

The components of Stage.1 are mainly the fundamental techniques known in DSP. Therefore we needed not to elaborate a lot about them. However, this does not mean the task at Stage.1 level is perfect. As an autonomous stage, we have assumed it unquestionable for it can be improved independently. A future study concerning this Stage.1 might be the enhancement of the input speech signal, the determination of

characteristics that are interesting enough (characteristics that we refer to throughout this thesis as features) and their efficient extraction. This is if we have to work on the accuracy factor for a recognizer.

In fact, the problem of overlap among clusters of phonemes affects the accuracy factor of a recognition application. This is why the choice of the number of clusters will depend on the avoidance of cluster overlaps. Cluster overlaps make the distance evaluation problem (as it is with frequencies or cepstrums) singular (means multiple solutions). This boils down to a tougher disambiguation problem for the Stage.3 of the TSA thus increasing its computation load. We don't want to have a heavy Stage.3 or else, the overall gain in speed will be lost. Therefore a solution might be, if possible, to come up with innovative ideas about features whose representation might reduce if not cancel the overlap of clusters. For now, the overlap shown in Figure 4.1.1 is a result of a representation that uses frequencies as analyzing parameter. Maybe the use of a different parameter might look different.

The main recommendation of this thesis concerns the Stage.3 of the TSA. In this stage, we assume that there will have a NLP module that will form linguistically valid words from the chains of phonemes from Stage.2. Therefore Stage.3 must exist. Throughout this document, we did not elaborate about the content of Stage.3 though the overall performance of an ASR application using our suggested approaches depends also on the performance of this stage.

The task at Stage.3 is not simple. Indeed, it involves complex algorithms such as HMM processes or can even use neural networks. It is a module that takes strings of phonemes as inputs and should output the most probable string of linguistically valid words. This task might involve work with a lexicon using search algorithms.

There exist several and very efficient algorithms to work on strings. This is the leading reason which brought us to opt for moving the HMM to Stage.3 to exclusively work on strings rather utterances. The task that remains to be achieved is the design of the modules in this Stage.3, a string-processing problem.

6.4. Challenges in automatic speech recognition

Several challenges await the commercial market of speech recognition for the next several years in technology, business and social aspects. Technology, business, and social challenges are:

- Speech Technology: noisy environments are still enemies of speech recognition applications though there is progress on that side. It's particularly inherent to mobile communication systems with losses and fading channels. On another hand, ASR and TTS become extremely complex when it comes to process diverse vocabularies and grammars of people worldwide.
- Standards Debate: standardizing the domain is still a problem and is an ongoing debate especially due to the confusion over Voice XML and SALT (Speech Application Language Tags).

  SALT is an open standard that gained instant credibility on October 15, 2001 when the SALT Forum (www.saltforum.org) was announced by a powerful consortium including Microsoft, Cisco, and Intel. The reaction of the industry players to the proposed standard was that SALT was designed to compete with the current leading standard that is Voice XML (VXML), a standard that has just been created in 1999 (Forum http://www.infoworld.com/articles/hn/xml/01/10/23/011023hnsalt.xml). The denial of it proponents in that SALT is intended to complement XML did not clear the fear of the opponents. (Source: "Microsoft Sprinkles Salt on Developers" http://www.infoworld.com/articles/hn/xml/01/10/23/011023hnsalt.xml).

  The debate is around how many standards are needed to support the different platforms on the market. Voice XML was designed to enable a single application for both PC and telephony-based platforms. Strongly promoted by Microsoft Corporation, SALT does not seem to convene its opponents about its capability to support different platforms although it is largely believed to be designed for different purposes. The ironic story about this confusion of the market about which standard to

use is that the World Wide Web Consortium (W3C) announced its support for VXML the same day Bill Gates announced his support for SALT.

So, until a standard is agreed upon, the full Internet will not be available to voice portal users. Therefore the market explosion that is predicted for the speech recognition application might just remain a dream.

- Real-Time Access: in the way Internet is design today, it's complicate to handle voice application over it due to bandwidth limitation and real-time protocols.
- Customer Acclimation: to begin with wide use of ASR, consumer population must learn how to interact with automated agents versus a real person, while waiting for technology improvement. Also, the automated agents must be able to accommodate themselves to ensure customers enjoy interacting with these new systems.
- Privacy and Security: if data transmission can be secured in the network, it's probably difficult to do so at the interface end-point. In fact, in an environment full of people, consumers may not be willing to read out loud their social security and credit card numbers.

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.

[12] R. P. Lippmann, "Speech Recognition by machines and humans", Speech Communication 22, 1997, pp. 1-15.

[13] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihood", ICASSP, Proc. 1996, vol. 2, pp 692-695, Apr. 1996.

[14] S. M. Herman and R. A. Sukkar, "Variable threshold vector quantization for reduced continuous density likelihood computation in speech recognition," Workshop on Automatic Speech Recognition and Understanding; Proc. 1997; pp. 331-338, Dec. 1997.

[15] E. Burhke, W. Chou, and Q. Zhou, "A wave decoder for continuous speech recognition," Proc. 1996 ICSLP, pp. 2135-2138, Oct. 1996.

[16] S. Furui: "Speaker-independent isolated word recognition using dynamic features of speech spectrum," IEEE Trans. Acoust., Speech, Signal Processing, ASSP-34, 1, pp. 52-59 (1986).

[18] Sumio Ohno, Hiroya Fujisaki, Keikichi Hirose; "A Method for Efficient DP Matching in Spoken Word Recognition"; International Symposium on Speech, Image Processing and Neural Networks, 13-16 April 1994, Hong Kong

[19] Amit Juneja, "Ph.D thesis", Department of electrical and Computer Engineering, University of Maryland, College Park, MD 20742, USA, October 31, 2003.

[20] H. Fletcher and J. C. Steinberg; "Articulation testing methods"; Bell System Technical Journal, vol 88, pp. 806-854, Oct. 1929.

[21] J. B. Allen, "How do humans process and recognize speech?", IEEE Trans. on Speech and Audio Proc., 2(4):567-577, October 1994.

[22] J. B. Allen, "From Lord Rayleigh to Shannon: How do humans decode speech?", http://auditorymodels.org/jba/PAPERS/ICASSP .

[23] K. Waheed, K. Weaver, M. F. Salam; "A robust algorithm for detecting speech segments using an entropic contrast"; 45th Midwest symposium on circuits and systems, Conference proceedings cat. No.02CH37378.2002: III-328-31, Vol.3

[24] G.E. Peterson and H. L. Barney, "Control Methods Used in a study of the Vowels," J. Acoust. Soc. Am., 24(2): 175-194, March 1952.

[25] R. Lawrence and J. Biing-Hwang, "Fundamentals of Speech Recognition", Prentice hall signal processing series, Englewood Cliffs New Jersey 07632, pp. 27, 1993.

[26] 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.

[27] K. Sartipi, K. Kontogiannis, "Component clustering based on maximal association", Proceedings of the Eighth Working Conference on Reverse Engineering, pp. 103 - 114, Oct. 2001.

[28] Wilpon, J.G.; Lee, C.-H.; Rabiner, L.R.; "Improvements in connected digit recognition using higher order spectral and energy features", Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on , 14-17 April 1991, Page(s): 349 -352 vol.1.

[29] P. B. de Maureüil, C. Corredor-Ardoy and M. Adda-Decker; "Multi-lingual automatic phoneme clustering", International Congress of Phonetic Science 1999.

[30] C. Corredor-Ardoy, J. L. Gauvain, M. Adda-Decker, L. Lamel; "Language Indentification with Language –Independent Acoustic Models"; Eurospeech'97, Rhodes.

[31] K. M. Berkling, E. Barnard; "Language Identification of six Languages Based on a Common Set of Broad Phonemes"; International Conference on Spoken Language Processing 1994.

[32] J. Kölher; "Multi-lingual Phoneme Recognition Exploiting Acoustic-Phonetic Similarities of Sounds"; International Conference on Spoken Language Processing 1996.

[33] Lawrence Rabiner and Biing-Hwang Juang; "Fundamentals of Speech Recognition", prentice hall signal processing series, 1993, pp. 42.

[34] Lawrence Rabiner and Biing-Hwang Juang; "Fundamentals of Speech Recognition", prentice hall signal processing series, 1993, pp. 27.

[35] Lawrence Rabiner and Biing-Hwang Juang, "Fundamentals of Speech Recognition", prentice hall signal processing series, 1993, pp. 113

[36] Keh-Yin, Su; Chin-Hui, Lee; "Robustness and discrimination oriented speech recognition using weighted HMM and subspace projection approaches", International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91), 14-17 April 1991, Vol. 1, pp: 541 -544

[37] Melnikoff, S.J.; Quigley, S.F.; Russell, M.J.; "Performing speech recognition on multiple parallel files using continuous hidden Markov models on an FPGA", IEEE International Conference on Field-Programmable Technology (FPT-2002), 16-18 December 2002, pp: 399 –402.

[38] Marzal, A.; Vidal, E.; "A N-best sentence hypotheses enumeration algorithm with duration constraints based on the two level algorithm", 11th IAPR International Conference on Pattern Recognition, Vol.III. Conference C: Image, Speech and Signal Analysis, 30 Aug.-3 Sept. 1992, pp: 619 -622.

[39] Akakpo AGBAGO, "Technical Report on implementation of an acoustic-phonetic based ASR system", OCIECE, April 2004, to be submitted.

[WEB1] SearchCRM.com Definitions, speech recognition: http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci213033,00.html

[WEB2] FOLDOC, speech recognition: www.nightflight.com/foldoc-bin/foldoc.cgi?speech+recognition

[WEB4] SearchMobileComputing.com Definitions, automated speech recognition: searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci786138,00.html

[WEB5] Brief History of Speech Recognition: www.pcai.com/Paid/Issues/PCAI-Online-Issues/16.6_OL/New_Folder/Sample_16.6/PCAI-16.6-Sample-pg.18-Art1.htm

[WEB6] NetByTel, History of Speech Recognition : www.netbytel.com/literature/e-gram/technical3.htm

[WEB7] Comparison of Speech Recognition Software: www.dyslexic.com/dictcomp.htm

SYMBOLS

API: Application Programmable Interface

ASR: Automatic Speech Recognition

DARPA: Defense Advanced Research Project Agency, www.darpa.mil

FTLDP: Fast-Two-Level Dynamic Programming.

HMM : Hidden Markov Models.

NLP: Natural Language Processing.

PARR: stands for ParrallelRecognizer in some figures.

RPKB : Reference Phoneme Knowledge Base

SUR : Speech Understanding Research program at Carnegie Mellon University, MIT

SALT : Speech Application Language Tags, http://www.saltforum.org/

TLDP: Two-Level Dynamic Programming.

TSA: Three-Stage Architecture

TTS: Text-To-Speech

T_R: Time Reduction

XML : Extensible Markup Language

APPENDIX A

The Timeline history of Speech Recognition

by (NetByTel, http://www.netbytel.com/literature/e-gram/technical3.htm)

1936	AT&T's Bell Labs produced the first electronic speech synthesizer called the Voder (Dudley, Riesz and Watkins). This machine was demonstrated in the 1939 World Fairs by experts that used a keyboard and foot pedals to play the machine and emit speech.
1968	The world-popular science fiction movie 2001: A Space Odyssey introduced the idea of speech recognition with the space ship computer, HAL.
1969	John Pierce of Bell Labs said automatic speech recognition will not be a reality for several decades because it requires artificial intelligence.
Early 1970's	The Hidden Markov Modeling (HMM) approach to speech recognition was invented by Lenny Baum of Princeton University and shared with several ARPA (Advanced Research Projects Agency) contractors including IBM. HMM is a complex mathematical pattern-matching strategy that eventually was adopted by all the leading speech recognition companies including Dragon Systems, IBM, Philips, AT&T and others.
1971	DARPA (Defense Advanced Research Projects Agency) established the Speech Understanding Research (SUR) program to develop a computer system that could understand continuous speech. Lawrence Roberts, who initiated the program, spent $3 million per year of government funds for 5 years. Major SUR project groups were established at CMU, SRI, MIT's Lincoln Laboratory, Systems Development Corporation (SDC), and Bolt, Beranek, and Newman (BBN). It was the largest speech recognition project ever.
1978	The popular toy "Speak and Spell" by Texas Instruments was introduced. Speak and Spell used a speech chip which led to huge strides in development of more human-like digital synthesis sound.
1982	Dragon systems was founded in 1982 by speech industry pioneers Drs. Jim and Janet Baker. Dragon Systems is well known for its long history of speech and language technology innovations and its large patent portfolio.
1984	SpeechWorks, the leading provider of over-the-telephone automated speech recognition (ASR) solutions, was founded.
1995	Dragon released discrete word dictation-level speech recognition software. It was the first time dictation speech recognition technology was available to consumers. IBM and Kurzweil followed a few months later.
1996	Charles Schwab is the first company to devote resources towards developing up a speech recognition IVR system with Nuance. The

	program, Voice Broker, allows for up to 360 simultaneous customers to call in and get quotes on stock and options... it handles up to 50,000 requests each day. The system was found to be 95% accurate and set the stage for other companies such as Sears, Roebuck and Co., and United Parcel Service of America Inc., and E*Trade Securities to follow in their footsteps.
1996	BellSouth launches the world's first voice portal, called Val and later Info By Voice.
1997	Dragon introduced "Naturally Speaking", the first "continuous speech" dictation software available (meaning you no longer need to pause between words for the computer to understand what you're saying).
1998	Lernout & Hauspie bought Kurzweil. Microsoft invested $45 million in Lernout & Hauspie to form a partnership that will eventually allow Microsoft to use their speech recognition technology in their systems.
1999	Microsoft acquired Entropic, giving Microsoft access to what was known as the "most accurate speech recognition system" in the world.
2000	Lernout & Hauspie acquired Dragon Systems for approximately $460 million.
2000	TellMe introduces first world-wide voice portal.
2000	NetBytel launched the world's first voice enabler, which includes an on-line ordering application with real-time Internet integration for Office Depot.

© Copyright 2000 NetByTel.com, Inc. All rights reserved.

APPENDIX B

Extraction of speech signal samples from a wave file

In the input gate of a speech recognizer, there is a speech signal. This signal can have different forms: electrical form (at very low level, the electrical signal is directly converted into samples and put into a buffer), audio stream resource or an audio file. The handling of an electrical signal or audio stream will be relatively difficult and unnecessary for the goal of this first phase in the design of a speech recognizer. Therefore, we choose to use audio files to test the tools of the process. For this, the speech utterances should be pre-recorded and saved into an audio file that will next be reloaded and the speech data or samples extracted from it. Though, several standards exist for audio files, we consider wave files for our research, though the code can easily be readjusted for other standards. Let see how a wave file is organized.

A wave file is a standard way of writing a digital audio data into a file. In general, an audio file contains a header that describes the structure of the file and the data as in Figure 1.

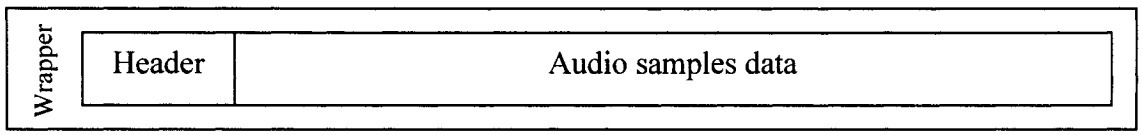
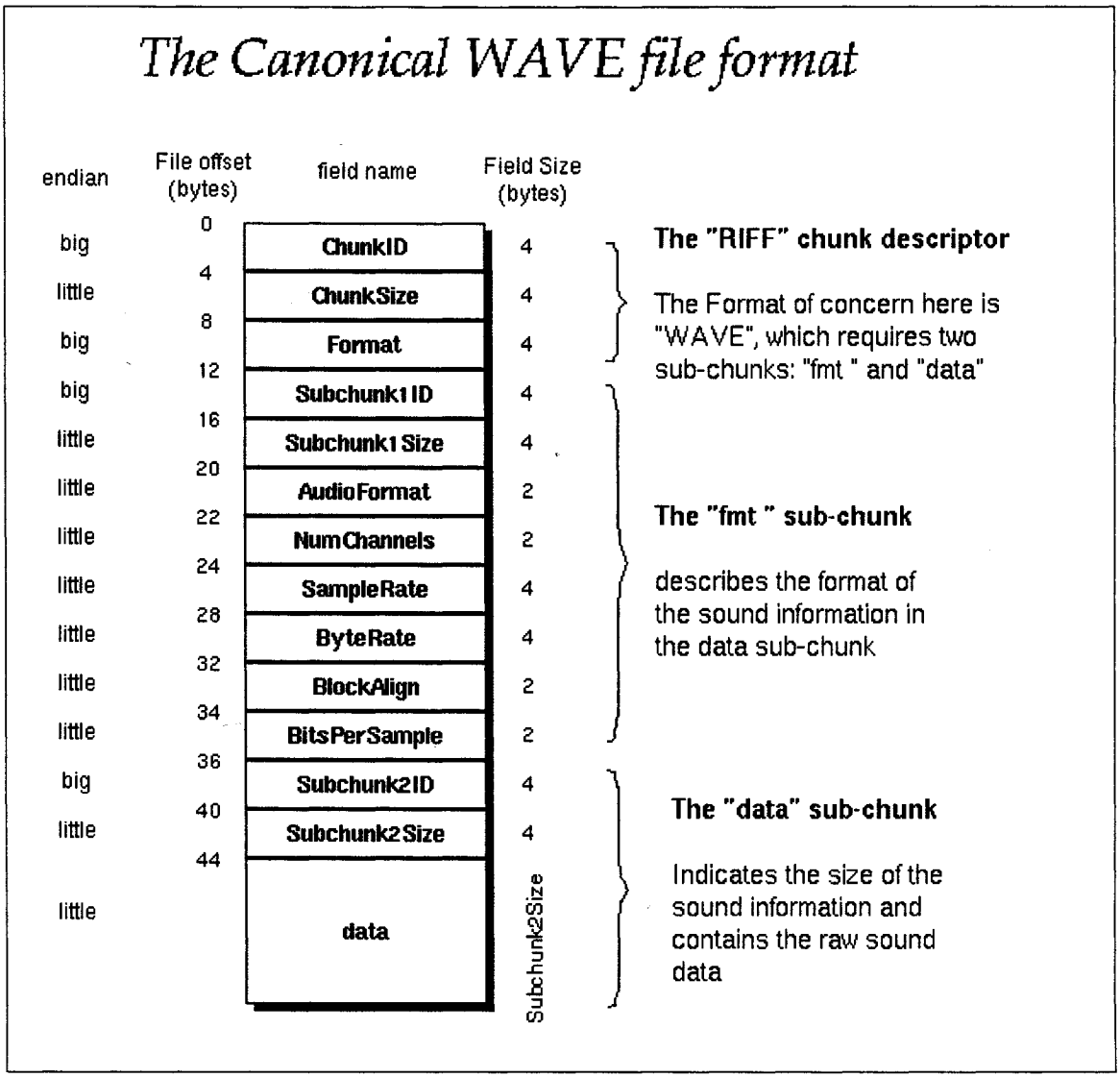


Figure 1: Audio file structure

A wave file has a format that is a subset of Microsoft's RIFF specification since a RIFF file contains a sequence of data chunks. Thus a WAVE file corresponds to a chunk RIFF file. The detail structure of a wave file header [1] is given in the figure 2



The canonical WAVE format starts with the RIFF header:

Offset	Size	Name	Description
0	4	**ChunkID**	Contains the letters "RIFF" in ASCII form(0x52494646 big-endian form).
4	4	**ChunkSize**	36 + SubChunk2Size, or more precisely: 4 + (8 + SubChunk1Size) + (8 + SubChunk2Size). This is the size of the rest of the chunk following this number. This is the size of the entire file in bytes minus 8 bytes for the two fields not included in this count: ChunkID and ChunkSize.
8	4	**Format**	Contains the letters "WAVE" (0x57415645 big-

			endian form).
The "WAVE" format consists of two sub chunks: "fmt " and "data": The "fmt " sub chunk describes the sound data's format:			
12	4	**Subchunk1ID**	Contains the letters "fmt " (0x666d7420 big-endian form).
16	4	**Subchunk1Size**	16 for PCM. This is the size of the rest of the Sub chunk, which follows this number.
20	2	**AudioFormat**	PCM = 1 (i.e. Linear quantization) Values other than 1 indicate some form of compression.
22	2	**NumChannels**	Mono = 1, Stereo = 2, etc.
24	4	**SampleRate**	8000, 44100, etc.
28	4	**ByteRate**	== SampleRate * NumChannels * BitsPerSample/8
32	2	**BlockAlign**	== NumChannels * BitsPerSample/8 The number of bytes for one sample including all channels. I wonder what happens when this number isn't an integer?
34	2	**BitsPerSample**	8 bits = 8, 16 bits = 16, etc.
	2	**ExtraParamSize**	if PCM, then doesn't exist
	X	**ExtraParams**	space for extra parameters
The "data" sub chunk contains the size of the data and the actual sound:			
36	4	**Subchunk2ID**	Contains the letters "data" (0x64617461 big-endian form).
40	4	**Subchunk2Size**	== NumSamples * NumChannels * BitsPerSample/8 This is the number of bytes in the data. You can also think of this as the size of the read of the sub chunk following this number.
44	*	**Data**	The actual sound data.

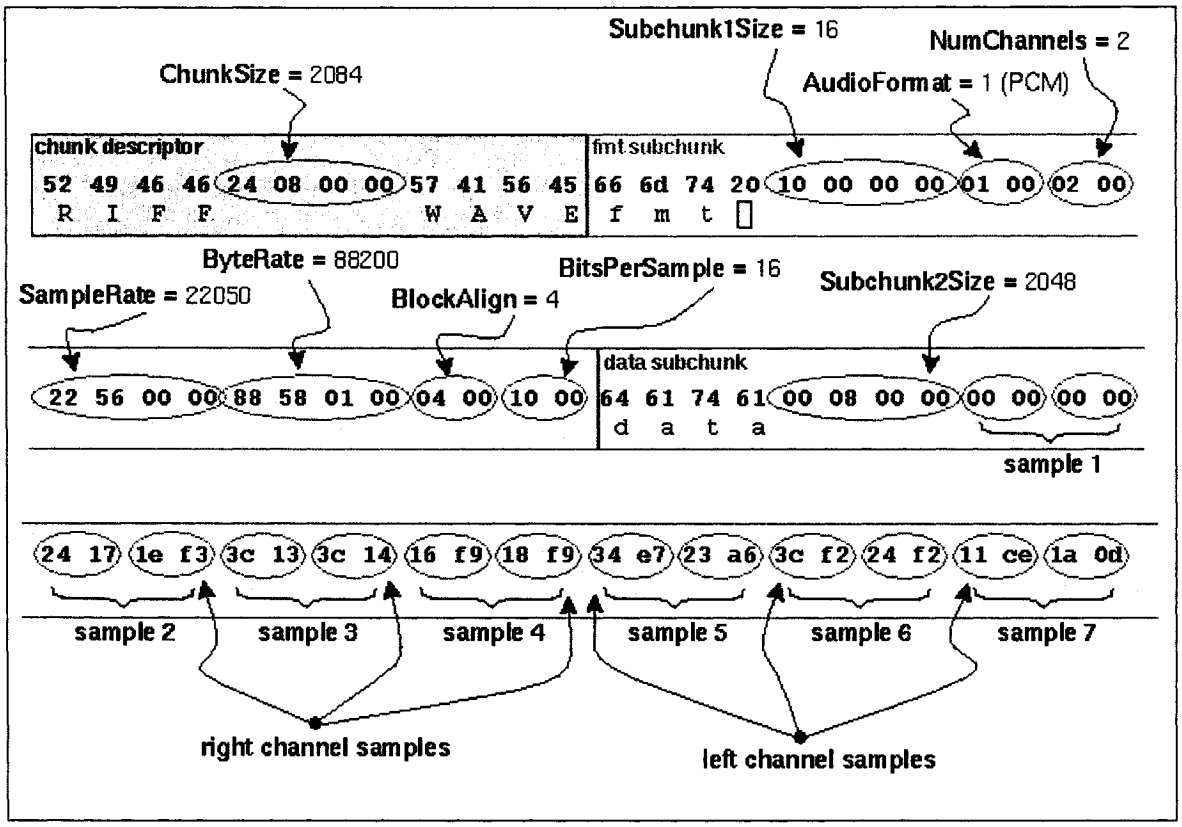
As an example, here are the opening 72 bytes of a WAVE file with bytes shown as hexadecimal numbers:

52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00

22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00 00

24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d

Here is the interpretation of these bytes as a WAVE sound file:



Notes:

- The default byte ordering assumed for WAVE data files is little-endian. Files written using the big-endian byte-ordering scheme have the identifier RIFX instead of RIFF.
- The sample data must end on an even byte boundary. Whatever that means.
- 8-bit samples are stored as unsigned bytes, ranging from 0 to 255. 16-bit samples are stored as 2's-complement signed integers, ranging from -32768 to 32767.
- There may be additional sub chunks in a Wave data stream. If so, each will have a char[4] SubChunkID, and unsigned long SubChunkSize, and SubChunkSize amount of data.
- RIFF stands for *Resource Interchange File Format.*

Implementation

We have written a class in java called WAVFile.java the reads the information in a *.wav file's header and its data content. The data in any channel of the media can be retrieved independently. Also, this class can write a wave file from audio data contained in a buffer.

APPENDIX C

TIMIT DATABASE

Table of content of the remaining of this document

Chapter 1. The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus

NIST Speech Disc CD1-1.1, October 1990 contains the complete TIMIT acoustic-phonetic speech corpus designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems.

TIMIT CD-ROM Contents

convert	Directory containing software (ESPRIT) for converting DARPA TIMIT continuous speech database files into the European Multi-lingual Speech input/output Assessment Methodology and Standardization (SAM) speech database format.

	SAM = Convert(TIMIT)
sphere	directory containing a new version (1.5) of the NIST Speech HEader REsources (SPHERE) software; SPHERE is a set of "C" library routines and programs for manipulating the NIST header structure prepended to the TIMIT waveform files.
Timit	Directory containing the TIMIT corpus as well as TIMIT-related documentation.

1. TIMIT (TI + MIT)

1.1. Corpus Speaker Distribution

TIMIT is a corpus of read speech, designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. Under sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO), TIMIT was recorded by Texas Instruments (TI), transcribed at Massachusetts Institute of Technology (MIT), has been maintained, verified, and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST), and with joint effort of Stanford Research Institute (SRI).

1.2. Corpus Speaker Distribution

TIMIT contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers (male and female) from 8 major dialect regions of the United States (see detail in \TIMIT\README.DOC).

1.3. Corpus Text Material

The text material in the TIMIT prompts (found in the file "prompts.doc") consists of 2 dialect "shibboleth" sentences (SA sentences) designed at SRI to expose the dialectal variants, 450 phonetically-compact sentences (SX sentences) designed at MIT to provide a good coverage of pairs of phones with extra occurrences of phonetic contexts thought to be either difficult or of particular interest, and 1890 phonetically-diverse sentences (SI sentences) selected at TI to add diversity in sentence types and phonetic contexts (variety of allophonic contexts).

1.4. Suggested Training/Test Subdivision

TIMIT corpus is subdivided into portions for training and testing according criteria described in the file "testset.doc".

1.5. TIMIT Directory and File Structure

The speech and associated data is organized on the CD-ROM according to the following hierarchy:
/<CORPUS>/<USAGE>/<DIALECT>/<SEX><SPEAKER_ID>/<SENTENCE_ID>.<FILE_TYPE>
Where,

CORPUS :== timit

```
     USAGE :== train | test
     DIALECT :== dr1 | dr2 | dr3 | dr4 | dr5 | dr6 | dr7 | dr8
                 (see Table 1 for dialect code description)
     SEX :== m | f
     SPEAKER_ID :== <INITIALS><DIGIT>

        where,
        INITIALS :== speaker initials, 3 letters
        DIGIT :== number 0-9 to differentiate speakers with identical
                  initials

     SENTENCE_ID :== <TEXT_TYPE><SENTENCE_NUMBER>

        where,

        TEXT_TYPE :== sa | si | sx
                (see Section 2 for sentence text type description)
        SENTENCE_NUMBER :== 1 ... 2342

     FILE_TYPE :== wav | txt | wrd | phn
                   (see Table 5 for file type description)

Examples:
     /timit/train/dr1/fcjf0/sa1.wav

     (TIMIT corpus, training set, dialect region 1, female speaker,
      speaker-ID "cjf0", sentence text "sa1", speech waveform file)


      /timit/test/df5/mbpm0/sx407.phn

      (TIMIT corpus, test set, dialect region 5, male speaker, speaker-
ID
       "bpm0", sentence text "sx407", phonetic transcription file)
```

1.6. File Types

```
The TIMIT corpus includes several files associated with each utterance.
In
addition to a speech waveform file (.wav), three associated
transcription
files (.txt, .wrd, .phn) exist.  These associated files have the form:

  <BEGIN_SAMPLE> <END_SAMPLE> <TEXT><new-line>
  .
  .
  .
  <BEGIN_SAMPLE> <END_SAMPLE> <TEXT><new-line>
     where,
       BEGIN_SAMPLE :== The beginning integer sample number for the
                        segment (Note: The first BEGIN_SAMPLE of each
                                file is always 0)

       END_SAMPLE :== The ending integer sample number for the segment
                     (Note: Because of the transcription method used,
                     the last END_SAMPLE in each transcription file
                     may be less than the actual last sample in the
```

```
                corresponding .wav file)

      TEXT :== <ORTHOGRAPHY> | <WORD_LABEL> | <PHONETIC_LABEL>

         where,

           ORTHOGRAPHY :== Complete orthographic text transcription
           WORD_LABEL :== Single word from the orthography
           PHONETIC_LABEL :== Single phonetic transcription code
                              (See "phoncode.doc" for description
                              of codes)
```

File Type Description

.wav - SPHERE-headered speech waveform file. (See the "/sphere" directory for speech file manipulation utilities.)

.txt - Associated orthographic transcription of the words the person said. (Usually this is the same as the prompt, but in a few cases the orthography and prompt disagree.)

.wrd - Time-aligned word transcription. The word boundaries were aligned with the phonetic segments using a dynamic string alignment program (see the printed documentation section "Notes on the Word Alignments" and the lexical pronunciations given in "timitdic.txt".)

.phn - Time-aligned phonetic transcription. (See the reprint of the article by Seneff and Zue (1988), in the printed documentation, and the section "Notes on Checking the Phonetic Transcriptions" for more details on the phonetic transcription protocols.)

```
Example transcriptions from the utterance in
"/timit/test/dr5/fnlp0/sa1.wav"

Orthography (.txt):
        0 61748 She had your dark suit in greasy wash water all year.

Word label (.wrd):
        7470 11362 she
        11362 16000 had
        15420 17503 your
        17503 23360 dark
        23360 28360 suit
        28360 30960 in
        30960 36971 greasy
        36971 42290 wash
        43120 47480 water
        49021 52184 all
        52184 58840 year

Phonetic label (.phn):
(Note: beginning and ending silence regions are marked with h#)
        0 7470 h#
        7470 9840 sh
```

```
9840 11362 iy
11362 12908 hv
12908 14760 ae
14760 15420 dcl
15420 16000 jh
16000 17503 axr
17503 18540 dcl
18540 18950 d
18950 21053 aa
21053 22200 r
22200 22740 kcl
22740 23360 k
23360 25315 s
25315 27643 ux
27643 28360 tcl
28360 29272 q
29272 29932 ih
29932 30960 n
30960 31870 gcl
31870 32550 g
32550 33253 r
33253 34660 iy
34660 35890 z
35890 36971 iy
36971 38391 w
38391 40690 ao
40690 42290 sh
42290 43120 epi
43120 43906 w
43906 45480 ao
45480 46040 dx
46040 47480 axr
47480 49021 q
49021 51348 ao
51348 52184 l
52184 54147 y
54147 56654 ih
56654 58840 axr
58840 61680 h#
```

2. SPHERE (SPeech HEader REsources)

Since the wave files (i.e SA1.waw) are not standard audio files but audio samples preceded by 1024 bytes object-oriented header, they cannot be read by standard audio players therefore SPHERE software is created to do the extraction.

SPHERE is a software package developed for use within the DARPA speech research
community containing:

1. a set of C functions that can be used to:
   a) create and modify TIMIT speech file headers (in memory)
   b) read (write) TIMIT speech file headers from (to) disk
2. a set of basic utility programs that use the functions

2.1. Usage: see detail in \timit\sphere\readme.doc

2.2. Synopsis of SPHERE functions

```
#include <header.h>
#include <sp.h>

struct header_t *sp_create_header()
struct header_t *sp_open_header(fp,parse_flag,error)
FILE *fp;
char **error;
int parse_flag;

int sp_close_header(h)
struct header_t *h;

int sp_clear_fields(h)
struct header_t *h;

int sp_get_nfields(h)
struct header_t *h;

int sp_get_field(h,name,type,size)
struct header_t *h;
char *name;
int *type, *size;

int sp_get_data(h,name,buf,len)
struct header_t *h;
char *name, *buf;
int *len;

int sp_add_field(h,name,type,p)
struct header_t *h;
int type;
char *name, *p;

int sp_delete_field(h,name)
struct header_t *h;
char *name;

int sp_change_field(h,name,type,p)
struct header_t *h;
char *name, *p;
int type;

int sp_write_header(fp,h,hbytes,databytes)
FILE *fp;
struct header_t *h;
long *hbytes, *databytes;

int sp_format_lines(h,fp)
struct header_t *h;
FILE *fp;

int sp_print_lines(h,fp)
FILE *fp;
struct header_t *h;
```

3. CONVERT (v 1.2)

SAM = Convert(TIMIT) was implementation at the Institut de la Communication Parlé Grenoble, France as result of cooperation of the National Institute of Standards and Technology (NIST) (U.S.A) with the E.E.C. European Strategic Project on Information Technology (ESPRIT) Project No. 2589 (SAM).

3.1. TIMIT speech file format: header of 1024 Byte

The NIST TIMIT speech file format is composed of a header which is 1024-byte
long, followed by the sampled speech. The header contains information about the file production. Some of this information is used by the CONVERT
software.

3.2. DARPA TIMIT file naming convention

```
\TIMIT\TRAIN\DR1\FCJF0\SA1.WAV

where      TIMIT: is the database identifier
           TRAIN: is the usage directory
             DR1: is the dialect region directory
           FCJF0: is the sex/speaker identifier directory
         SA1.WAV: is speech waveform file for the sentence-text, "sa1"
       with CJF0: speaker code (4 positions)
                1: dialect region code
              SA1: sentence code
              WAV: means WAVeform
```

3.3. SAM speech file format

The SAM speech file format consists only of sampled speech but is associated with a label file containing relevant information as clarified bellow.

3.4. SAM naming convention

```
      XXnnxxxx.SAS <-- matching information in --> XXnnxxxx.SAO

Where:
      XXnnxxxx.SAS: is the speech waveform file
      XXnnxxxx.SAO: is the associated label file
With:
      XX: speaker code (2 positions)
      nn: corpus code (2 positions)
    xxxx: unique file number (4 positions)
       S: means Sentence
       A: means American
       S: means Sampled speech
       O: means Orthographic time-aligned labelling
```

3.5. How to run CONVERT see \CONVERT\README.DOC for detail.

APPENDIX D

Market Assessment of Canada's Speech Recognition Software ISA990501

Source: http://strategis.ic.gc.ca/SSG/dd72404e.html

Industry Canada Industrie Canada

Canada

Français | Contact Us | Help | Search | Canada Site

strategis.gc.ca

Source: STAT-USA on the Internet
US Department of Commerce
(202) 482-1986

Previous | Home | Next

Market Assessment

Market Profile

Voice recognition is a relatively new input technology that allows users to communicate with their computer through voice recognition software application instead of a keyboard. Speech recognition technology enables a computer to respond to the human voice in place of a keyboard or mouse. It allows the user to interact with machines in the same way as they interact with people - through natural speech. Voice recognition has the potential to radically change the way people use computers. Bill Gates, Chairman of Microsoft Corporation, identified speech recognition as a key advancement to be emphasized in future development.

According to TMA Associates, an American research firm, 1998 worldwide revenues from automatic speech recognition (ASR) software are expected to surpass US$2 billion. This figure is expected to grow rapidly to nearly US$37 billion by 2003. The remarkable growth of ASR software in the short term is consistent with other patterns of technological development in the IT sector. New revolutionary technology is quickly becoming the standard. In the near future, use of voice user interface (VUI) is expected to become widespread. Consumer demand and increased competition will make ASR-supported VUIs an integral component of all computer systems.

Three major players, IBM, Dragon Systems, and Lernout and Hauspie (L&H) dominate the speech recognition software (SRS) market in Canada. According to industry experts, Dragon Systems holds approximately 40-45 percent market share, IBM 35-40 percent, followed by L&H with approximately 20-25 percent. Brampton, Ontario-based Northern Telecom is also getting into the market by forming alliances with business partners like Hewlett Packard, Intel and Microsoft.

Because of the small number of players in this sector, sales data are confidential and statistical data on market size, production and trade are not available. The lack of public information about the SRS market prevents industry players from releasing hard data or opinions on this sector's performance. Most activity in Canada related to SRS is related to the sale and customization of SRS for specific applications.

Nevertheless, industry experts agree that Canadian SRS market demand is growing very rapidly and that SRS will have a large impact on the future of computing and human-computer interactions.
To gauge SRS market potential, U.S. firms should be aware that the Canadian market is approximately one-tenth the size of the American market and that trends in Canada tend to mirror those in the United States. Canada is a prosperous country and one of the best-wired nations in the world. Canada's population is approximately 30 million and households number about 9 million.
In a recent survey A.C. Nielsen found that 58 percent of Canada's households have at least one personal computer. Canada's proliferation of computers capable of processing SRS applications efficiently in both the home/consumer and business markets speaks very strongly for increased use of SRS.
Canadians are receptive to efficiency enhancing technologies and have the computer and telecommunications infrastructure required to implement SRS rapidly. Canadian citizens and businesses understand the need for and support the implementation of information technology and SRS. As a result, the Canadian market is very receptive to SRS applications.
Several key factors can be cited to support the rapid growth of SRS market in Canada. Recent advances in computing power and speech technology have led to wide commercial availability of new phonetic-based recognition algorithms that enable more powerful speech based solutions. As a result, speech recognition application software now being used in Canada is relatively advanced, effective and affordable. The move from discrete recognition to continuous speech and natural language recognition, accompanied by exponential price decreases, has given speech recognition software wide acceptance in Canada.
Currently, in Canada, SRS technology is most prevalent in vertical markets, particularly in professional markets such as legal and medical. The indigenous industry is working to develop horizontal speech recognition applications for commercial and mass consumer applications. The computer industry sees speech as a natural interface for computers and appliances and industry experts predict that fully comfortable, transparent, unrestricted mainstream speech recognition software use is probably 3 to 5 years away.
The leveraging of state-of-the-art voice activated telephone applications, to serve customers and gain a competitive edge, will drive SRS market demand in Canada. Features such as very large vocabulary recognition and natural language understanding are contributing to telephone-based applications that allow service providers to give customers quicker, more convenient access to information. Speech recognition software is providing a competitive edge to users through increased efficiencies and cost reduction in Canadian businesses. Some users are realizing payback on their investment in SRS in less than a year.
Key applications, such as e-commerce will contribute to the rapid acceptance and market growth of SRS in Canada. Electronic commerce is revolutionizing the way consumers and businesses buy goods and services. It is projected that the value of e-commerce transactions in Canada will reach US$8.6 billion by the year 2002, with consumer transactions accounting for 20 percent of on-line sales and business-to-business sales accounting for the rest.
Speech recognition will become essential as e-commerce sites move to embrace the broader consumer market. Speech recognition companies see the telephone as an access device that can be used to synchronize the caller's request with a Website. Several technology companies, including Motorola, Inc., Visa International, BroadVision, Inc., Nuance Communications and others have formed the V-Commerce Alliance, a consortium which will enable telephone users to access electronic commerce applications using speech recognition technology.
On-line transaction security has been one of the impediments to rapid acceptance of e-commerce. Therefore, new speech recognition technologies continue to be developed to address the security concerns of the high-volume call-processing marketplace by offering voice-verification for automated telephony applications. With the ability to recognize a person's unique voice patterns, voice verification can significantly reduce the risk of fraud. This technology will also lower an organization's operating costs, by reducing the amount of customer service required to verify a caller's identity. Improvements in this technology will drive e-commerce and speech recognition software sales in Canada.
In large part, the future of speech technology lies in mobile communication whereby users' lives will be enhanced and made more efficient by a seamless, transparent speech-driven interaction with machines. Industry experts predict that this technology will make human-machine interface much more interactive. In the October/November 1998 issue of Speech Technology Magazine, Dr. Caroline Henton, Vice President of Strategic Technology at fonix Corp, stated that mobility will be the single most powerful driving force in the growth of speech technology. She said that, "handheld devices that incorporate cell-phone links, PDA

applications, handwriting entry, and infrared connectivity will be physical necessities for communication in the next millennium and will drive speech technology".
Canada is fertile ground for SRS with mobile applications. Canada's wireless telecommunications industry grew by an impressive 26.4 percent in 1998, adding more than 1.1 million new wireless telephone subscribers. Total wireless telephones in Canada now number more than 5.3 million. More than one in six Canadians have access to secure wireless telephones. Living in a communications-based society, Canadians expect nothing but the best when it comes to reliable and reasonably priced wireless services. Service pricing in Canada is among the best in the world. Over the next decade, mobile wireless usage for personal communications is predicted to rise from its current level of 17.5 percent of Canadians to almost 40 percent. The development of technical standards for voice technology as the user-machine interface will be key to the advancement of SRS technology. On April 13, 1999 leaders in speech recognition and mobile technologies announced the formation of the Voice Technology Initiative for Mobile Enterprise Solutions (VoiceTIMES). The VoiceTIMES alliance members include Dictaphone, Digital, IBM, Intel, Norcom Electronics, Olympus and Philips. VoiceTIMES' goal is to coordinate the technical requirements needed for companies to build and deploy solutions using voice technologies and handheld mobile devices. Common technical industry standards in this sector will help propel the growth of speech technology, its application and use in Canada.
Efforts to establish speech technology standards extend beyond mobile enterprise to all parts of the computing environment. A single group, or alliance, can not alone develop voice recognition, identification and verification solutions for a wide range of products and standards fast enough. As a result, the VoiceTIMES Alliance will complement other industry efforts by defining the structure of voice data to be used in enterprise solutions, including transmission over other wireless standards and will coordinate its activities with the other alliances as standards are developed.

Best Sales Prospects

Excellent sales prospects for U.S. SRS exist in Canada for voice to text application software; voice-to-command application software, telephony applications software, developers' kits and vertical market vocabularies/libraries. All of the following products fall under the HS Code 8524.90 (Recorded media for sound or other similar recorded phenomena)

Voice-to-Text Application
1) Continuous Speech to Text Recognition Software (Software that recognizes continuous speech and creates typed text)
2) Natural Language to Text Recognition Software (Use of intuitive "natural language" for creating, editing and formatting typed text)
3) Natural Language Understanding (Use of artificial intelligence to move natural language understanding beyond its first generation, which relies on constrained dialogues, command banks, and specific recognition techniques)

Voice-to-Command Application
Voice-to-command application software is where the user can command an action by using voice e.g. turn lights on in a room; or command the telephone to dial a specific telephone number. Voice ID applications for call centers also fall under this application.

Telephony Applications
Telephony applications include voice dialing, interactive voice recognition (IVR), call center and auto attendant applications. For example, by using the telephone, the caller can command a database search to retrieve information via an automated attendant. Telephony applications are often used in commercial environments for customer service or e-commerce.
In the United States the telephony market, including instruments using SRS technology and services provided by telecom companies, is expected to reach US$1.57 billion in 1999. The Canadian market is generally one-tenth the size of the U.S. market. By 2003, the U.S. telephony market is projected to be about US$22.6 billion.

Software Developers Kits
Software to enable developers to integrate speech recognition into their applications.

Vocabularies/Libraries
These include industry/market specific vocabularies for professional markets such as the legal or medical professions.

Publication Date: 1999-06-10
Author:U.S. Department of Commerce
Important Notices and Disclaimers
Privacy Statement

Canada
http://strategis.gc.ca