Department of Computer Science and Engineering

The Chinese University of Hong Kong

Finial Year Project Report

2001-2002 Second Term

LYU0103

Speech Recognition Techniques for Digital Video Library

Supervisor:	Professor Michael R. Lyu
Prepared by:	Gao Zheng Hong (98795603)
Date of Submission:	17th, March 2002

Table of Contents

Abstract

Automatic Speech Recognition (ASR) is the process by which computers (or other type of machines) identifies spoken words and recognize what you are saying. It has been an interest of research by nearly four decades. It is amazing because it has a wide area of applications. If eventually computers can get 100 percent of recognizing words, it will bloom a new era of computer technology and even change people's daily life. However, there are many difficulties to overcome and speech recognition still has a long way to go.

This project is to apply multiple speech recognition techniques to build a speech information processor for audio information retrieval and speech recognition.

Chapter 1 : Introduction

1.1 Project Overview

Our project about speech recognition techniques is a part of a mother project called VIEW, which is short for “Video over InternEt and Wireless” and supervised by Professor Michael R. Lyu. This project is to develop a multilingual digital video content hub for culture exchange and commercial deployment. This project includes a digital video library, which uses the outcome tool of our project to produce captions for the videos in it. In order to specify the role of our project, we will give a brief introduction to the VIEW project. For more detailed information, please visit http://www.viewtech.org/.

1.1.1 VIEW Background

The “Multilingual Digital Video Content Hub” and “VIEW”(Video over InternEt and Wireless) project is funded by Innovation and Technology Commission (ITF) Fund and Area of Excellent. The planning time of funding is from September 2000 to September 2002. The Principal Investigator is our supervisor Prof. Michael R. Lyu in department of Computer Science and Engineering, Chinese University of Hong Kong.

The main objective of the VIEW project is to develop a multilingual digital video content hub with emphasis on Chinese and English languages for cultural exchange and commercial deployment.

And also, it aims at to deliver software-enabling schemes for content creation, information summarization, multi-model searching and presentation.

The logical framework of the VIEW project consists of three major components:

- Back-end
  - Video Information Processing Engine
  - Indexing and Searching Engine
- Front-end
  - Visualization and Presentation Engine

1.1.2 Video Information Processing

There is a video information extraction engine that extracts multi-modal information from video including: transcript, key frames, characters, face, etc.

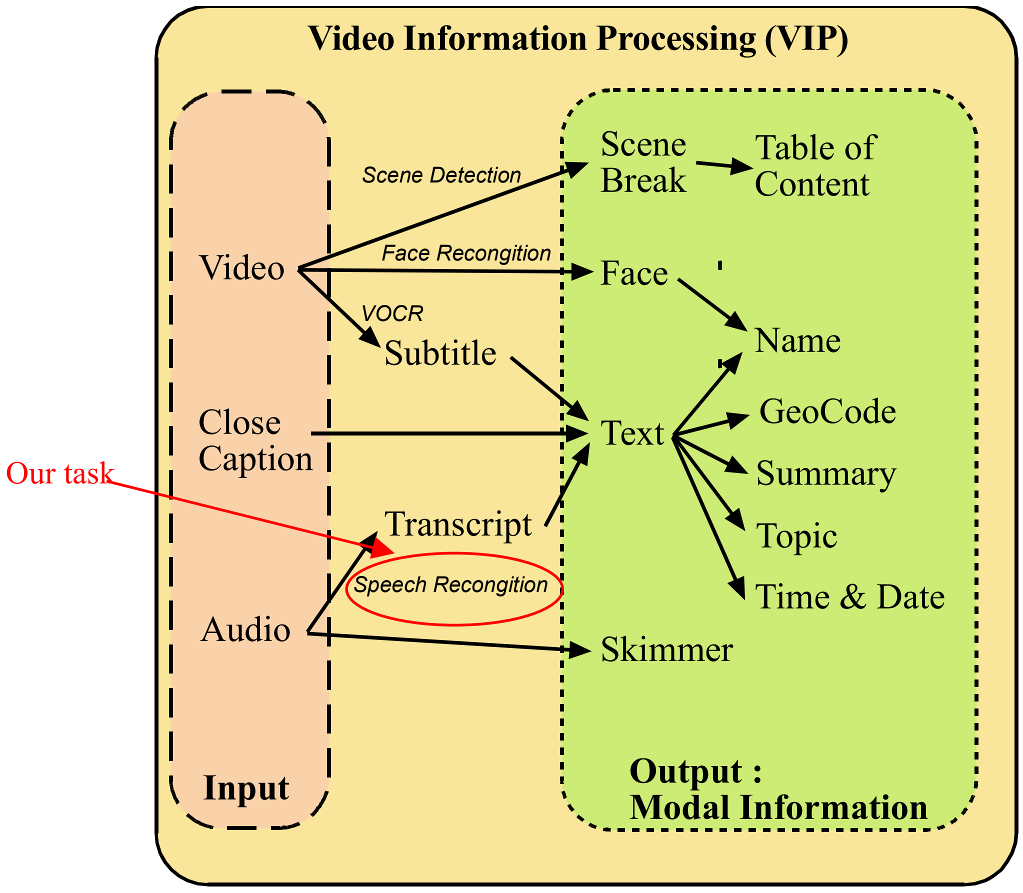


Figure 1.1 Video Information Processing

The information processing has major components including: speech recognition, video OCR, scene detection, face recognition, etc.

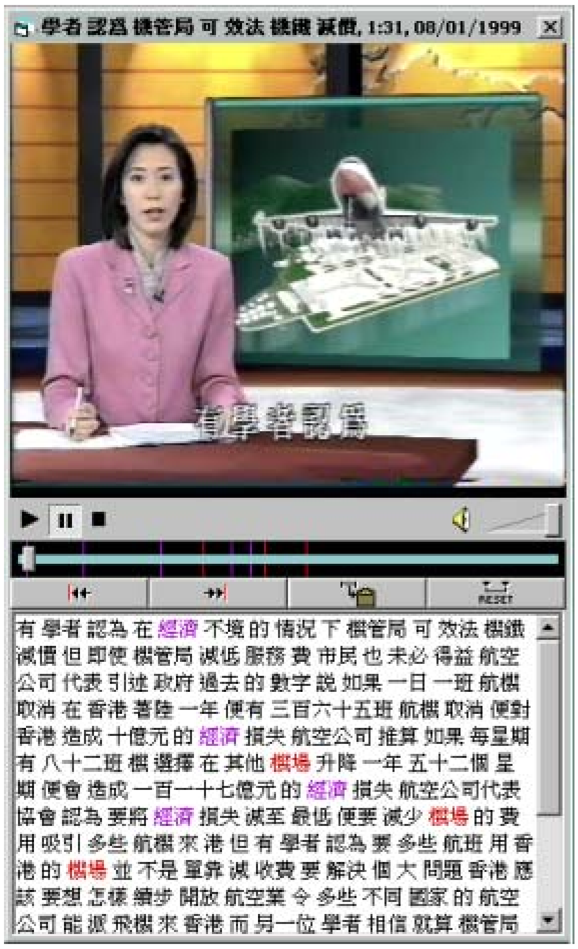
1.1.3 Speech Recognition Engine

A speech recognition engine is a software component, which is able to handle low-level speech recognition tasks. The nature of our project requires a state-of-the-art high quality speech recognition engine, which can dictate the speech in a large volume of video segments and produce text with an acceptable accuracy. Usually, a speech engine is a big component, which requires large amount of investment and human recourses to develop. Therefore, we are not going to develop an engine ourselves. Instead, we use other commercial engines to provide a programmable low-level speech API for us to work on. There are several commercial speech engines available, such as Dragon Systems, Microsoft SAPI, and IBM ViaVoice. After a comparison study, we chose IBM ViaVoice.

1.2 Project Objectives

After introduce the Multilingual Digital Video Content Hub, our task in the project is not hard to see: apply speech recognition techniques for the video data obtained from digital video library to retrieve certain useful information, including the text of the speech, the timing of every word spoken, the gender of the speaker, and so on.

The speech information will enable the video information processor to produce captions for videos and highlighting the word, which is currently being spoken, at correct time while playing videos. The speech processor should be able to retrieve other useful audio information, such as audio scene change, which may further improve our understanding of speech. In order to achieve our goal, we need to perform multi-lingual speech recognition, timing information retrieval, real-time dictation, recognition text editing, audio scene change detection, audio segment classification, and gender classification. The ultimate goal is to display the video as well as showing the text of speaker's voice, as shown in figure 1.2.



有 學者 認為 在 經濟 不境 的 情況 下 機管局 可 效法 機鐵
減價 但 即使 機管局 減低 服務 費 市民 也 未必 得益 航空
公司 代表 引述 政府 過去 的 數字 說 如果 一日 一班 航機
取消 在 香港 著陸 一年 便有 三百六十五班 航機 取消 便對
香港 造成 十億元 的 經濟 損失 航空公司 推算 如果 每星期
有 八十二班 機 選擇 在 其他 機場 升降 一年 五十二個 星
期 便會 造成 一百一十七億元 的 經濟 損失 航空公司代表
協會 認為 要將 經濟 損失 減至 最低 便要 減少 機場 的 費
用 吸引 多些 航機 來 港 但 有 學者 認為 要 多些 航班 用 香
港 的 機場 並 不是 單靠 減 收費 要 解決 個 大 問題 香港 應
該 要想 怎樣 續步 開放 航空業 令 多些 不同 國家 的 航空
公司 能 派 飛機 來 香港 而 另一位 學者 相信 就算 機管局

Figure 1.2 Video with Speech Recognition Processing

Since speech recognition is heavily used in our project, we will give a brief introduction to it in the next chapter.

Chapter 2 : Introduction to Speech Recognition

Automatic Speech Recognition (ASR) is a process by which a computer (or an equivalent type of machine) identifies spoken words from an input voice device. Basically, SR means talking to your computer, AND having it correctly recognize what you are saying.

2.1 Terminologies in SR System

In order to understand SR technology, we have to learn some terms first:

2.1.1 Utterance

An utterance is the vocalization (speaking) of a word or words, which represents a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

2.1.2 Speaker Dependence Vs. Speaker Independence

A speaker-dependent system is a system that must be trained on a specific speaker in order to recognize accurately what has been said (We will explain what means "training" later in the section). To train a system, the speaker is asked to record predefined words or sentences that will be analyzed and whose analysis results will be stored. This mode is mainly used in dictation systems where a single speaker is using the speech recognition system.

On the contrary, any speaker without any training procedure can use speaker-independent systems. Those systems are thus used in applications where it is not possible to have a training stage (telephony applications, for example).

It is also clear that the accuracy for the speaker-dependent mode is better compared to that of the speaker-independent mode. Adaptive system usually starts as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy.

2.1.3 Vocabularies

Vocabularies (or dictionaries) are a list of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. It is because that the SR system has more opportunities to make some errors in larger size. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g." Wake Up"), while very large vocabularies can have a hundred thousand or more!

A good SR system will therefore make it possible to adapt its vocabulary to the task it is currently assigned to. For example, enable a dynamic adaptation of the vocabulary.

2.1.4 Grammar

Grammars in SR system refer to the rules of the language model. Without grammar, every word will have an equal probability of occurrence at any point in utterance. As a consequence, it makes continuous speech recognition impossible. In fact, grammar places constraints on the sequence of words allowed, and the possible next word choice. That gives the system fewer choices at each point, improving the performance of recognition.

2.1.5 Isolated Word Recognition Vs. Continuous Speech Recognition

Isolated word recognition is the simplest speech recognition mode and it requires less CPU resources compared with continuous speech recognition mode. Each word is surrounded by a silence so that word boundaries are clearly justified. The system does not need to find the beginning and the end of each word in a sentence. The word is compared to a list of words models, and the model with the highest score is retained by the system. This kind of recognition is mainly used in telephony application to replace traditional DTMF methods.

Continuous speech recognition is much more natural and user-friendly. It assumes the computer is able to recognize a sequence of words in a sentence, or a sequence of sentences. But this mode requires much more CPU and memory, and the recognition accuracy is really lower compared with the preceding mode. Here lists the major problems faced with continuous speech recognition:

a. Speakers' pronunciation is less careful
b. Word boundaries are not necessarily clear
c. Speaking rate is less constant
d. There is more variation in stress and intonation.
e. Additional variability is introduced by the unconstrained sentence structure
f. Co articulation is increased both within and between words
g. Speech is mixed with hesitations, partial repetition, etc.

2.1.6 Difficulty Level

Based on speech recognition mode, we classify SR difficulty in 10 levels (Figure 2.1). The first level is most simple one, which means that it is speaker-dependent, able to recognize isolated words, and using a small vocabulary. 10 correspond to the most difficult level, which is speaker-independent continuous speech in a large size of vocabulary. State-of-the-art speech recognition systems with acceptable error rates are somewhere in between these two extremes.

	Isolated Words	Continuous Speech
Speaker Dependent	Small Vocabulary 1	Small Vocabulary 5
	Large Vocabulary 2	Large Vocabulary 7
Multi-Speaker	Small Vocabulary 2	Small Vocabulary 6
	Large Vocabulary 4	Large Vocabulary 7
Speaker Independent	Small Vocabulary 3	Small Vocabulary 8
	Large Vocabulary 5	Large Vocabulary 10

Figure 2.1 Difficulty level of Speech Recognition

2.1.7 Accuracy

The ability of a SR system can be measured by examining its accuracy. It means that how well it will recognize the utterances, which includes not only correctly identifying an utterance, but also identifying if the spoken utterance is not in its vocabulary. Good ASR system has an accuracy of more than 90%. The acceptable accuracy of a system mainly depends on the application.

2.1.8 Training

As mentioned before, some of the SR system has the ability to adapt to a specific speaker. When the system has this ability, it can apply the training process. The ASR system is trained by letting that specific speaker to record predefined phrases and sentences and adjusting its comparison algorithm to match that particular speaker's voice. So consequently, training increases the accuracy of ASR system.

Training can also be used by speakers who have problems in speaking or pronouncing certain words. As long as the speaker can consistently repeat an utterance, ASR systems with training should be able to adapt it.

2.2 Speech Recognition Process

2.2.1 Process Diagram

The speech recognition process can be divided into different components, which is illustrated in figure 2.2:

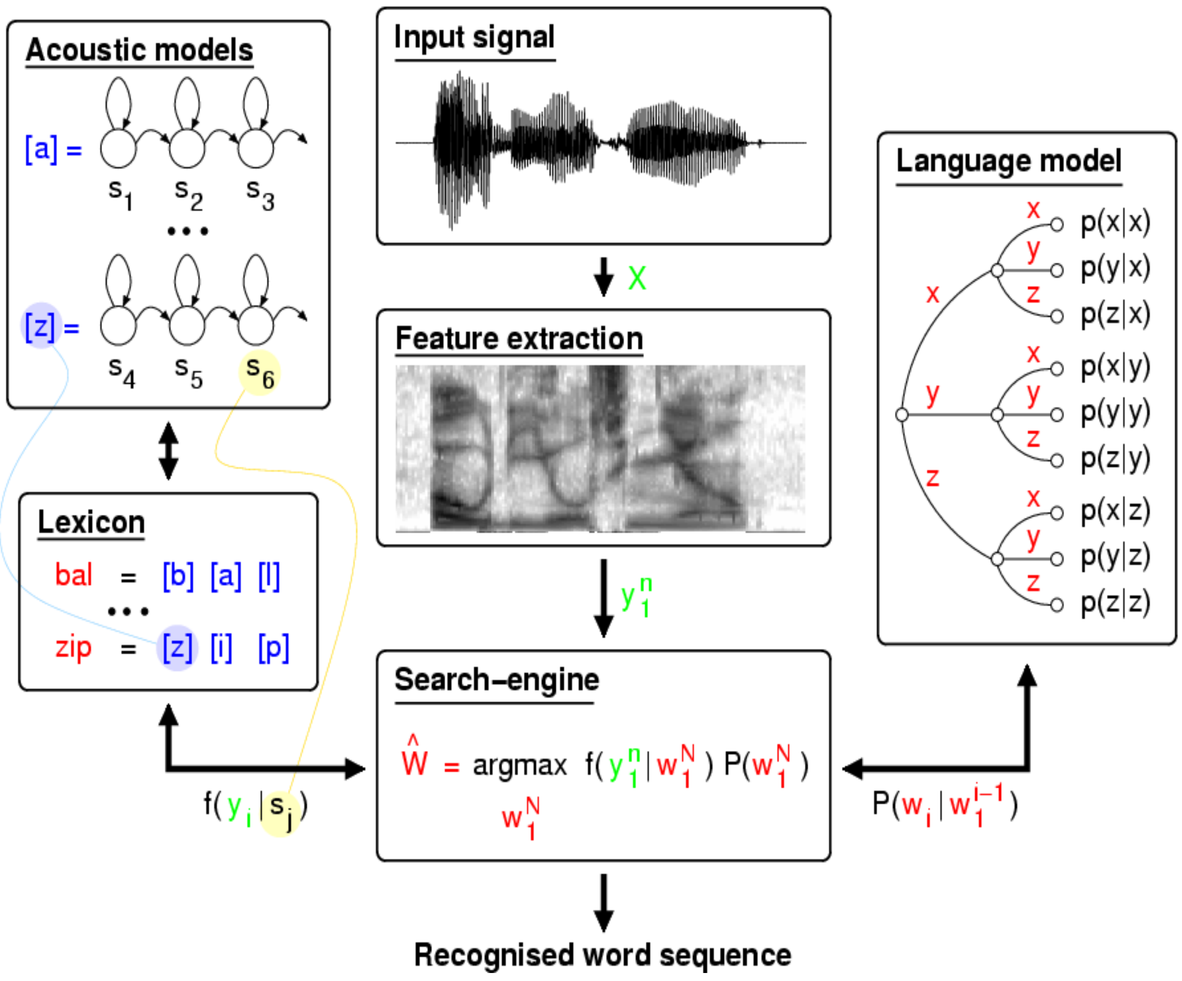


Figure 2.2 Speech recognition process

2.2.2 Process Explanation

2.2.2.1 Input Signal

First, we get the speaker's voice from an input device and save it as speech signals. The commonly used input device is a microphone. Note that the quality of the input device can influence the accuracy of SR system very much. The same applies to acoustic environment. For instance, additive noise, room reverberation, microphone position and type of microphone can all relate to this part of process.

2.2.2.2 Feature Extraction

The next block, which shows the feature extraction subsystem, is tried to deal with the problems created in first part, as well as deriving acoustic representations. The two aims are separate classes of speech sounds, such as music and speech, and effectively suppress irrelevant sources of variation

2.2.2.3 Search Engine

The search engine block is the core part of speech recognition process. In a typical ASR system, a representation of speech, such as spectral or cepstral representation, is computed over successive intervals, e.g., 100 times per second. These representations or speech frames are then compared with the spectra frames, which were used for training. It is done by using some measurement of distance of similarity.

Each of these comparisons can be regarded as a local match. The global match is a search for the best sequence of words, in the sense that it is the best match to the data. And it is determined by integrating many local matches. The local match does not usually produce a single hard choice of the closet speech class, but rather a group of distances or probabilities corresponding to possible sounds. These are then used as part of a global search or decoding to find an approximation to the closest sequence of speech classes, or ideally to the most likely sequence of words.

2.2.2.4 Acoustic Model

The acoustic model is the recognition system's model for the pronunciation of words, crucial to translating the sounds of speech to text. In reality, the type of speaker model used by the recognition engine greatly affects the type of acoustic model used by the recognition system to convert the vocalized words to data for the language model (Context / Grammar) to be applied.

There are a wide variety of methods to build the pattern models, the three major types are:

a. Vector Quantization

b. Hidden Markov Models (HMM)

c. Neural Networks

Due to the limitation of the size of this document, we are not going to present an in-depth discussion about the above methods here.

2.2.2.5 Language Model & Lexicon

Language model is another major component of the speech recognition process. It provides the knowledge source for the engine and helps to predict the next word.

The first component of the language model is the lexicon, which consists of the vocabulary. The vocabulary, as explained before, contains all of the possible words in the voice system that may be encountered.

The second component of the language model is the grammar. It defines the structure and format of the text allowed at any point in the utterance. Without grammar, every word in the lexicon would have an equal likelihood of occurrence at any point within the utterance, requiring algorithm that are too complex, making continuous speech impossible. These systems use the grammar to constrain the word choice at any point, which is more efficient.

2.3 Uses and Applications

Speech recognition system has a wide area of application. Here we just list some to get some points in mind:

2.3.1 Dictation

Dictation is the most commonly usage for ASR system. As a form of input, it is especially useful when someone's hands or eyes are busy. It allows people working in active environment such as hospitals to use computers. With ASR system, typing is not a necessary skill for using a computer. If we can successful get 100% accuracy of speech recognition, it would make computers accessible to people who do not want to learn the technical details to use them or just lazy to type words.

2.3.2 Command And Control

Imagine that in the near future, if we want to open a Netscape browser or click the start menu on Windows operating system, what we need to do is just say a word to the computer. People's hands are got free from mouse clicking forever.

2.3.3 Telephony

With the usage of ASR system, users can say the command in the telephone instead of pressing the buttons. It is more natural and convenient to do so.

2.3.4 Medical/Disabilities

Many people have difficulties in typing the words because of physical disabilities, such as blindness, muscular dystrophy and so on. Other people may have difficulties to hear the voice. With the ASR system, they can overcome these problems by talking to the engine or convert the caller's speech to text.

2.3.5 Embedded System Applications

It can also be used in embedded systems, for example a cell phone in the future can listen the command like "Call Jenny". Or a car can automatically drive itself by just hearing the voice form its host.

2.4 Challenges And Difficulties

Although SR is an amazing task, it has some challenges: mapping from a continuous time signal, the speech signal, to a sequence of discrete entities, such as words and sentences is the primary challenge of speech recognition processing.

Voice recognition is complex processing, for both discrete and continuous speech. The difficulties affect the process in the construction of the engine, acoustic and language models, the training of the system by users, and in practical application of the system, the major challenges are:

2.4.1 Linguistic Variability

Several factors will affect the input audio signal. Although these factors may only change the voice signal slightly, it can also complicates the speech recognition ability to accurately interpret the signal. These factors are:

a. Phonetics

b. Phonology

c. Syntax

d. Semantics

e. Discourse

2.4.2 Speaker Variability

The speaker variability not only refers to the variability between two different speakers, but also refers to the variability in speech, which can occur by an individual. The factors that affect the output include:

a. Tempo

b. Pronunciation

c. Inflection

d. Fatigue

e. Stress

f. Upper Respiratory Infection

2.4.3 Channel Variability

Even the quality and position of microphone and background environment will affect the output of ASR system dramatically. So we have to deal with mechanical disruption and outside interference of voice signal itself, which includes:

a. Background Noise

b. Transmission Channel

c. Quality and position of microphone

2.4.4 Coarticulation

Speech sounds of words are known as *Phonemes*. It is the characteristics of these phonemes, which allows listeners to identify the words. However, phonemes that make up words are not said in isolation. Every sound used for speech is affected by the sounds both proceeding and following it. Thus, no phoneme is an "island". Every phoneme will be influenced by the phoneme before and after it.

Coarticulation then is referred to the contextual effects of the phonemes on other phonemes:

a. The articulation of each sound blends with the articulation of the neighboring sounds (before and after)
b. It is known that the phonetic contextual effects may span multiple phonemes, but most often involves the closest neighboring phonemes.

Chapter 3 : Speech Recognition Engines

During the summer holidays in 2001, we investigated different speech recognition software such as *CMU Sphinx*, *Microsoft SAPI* and *IBM ViaVoice*. And we also visited ***IBM Research Lab China*** and ***Microsoft Research Institute of China*** in Beijing together with our supervisor Professor Michael R. Lyu this summer.

With the background knowledge of speech recognition in hand, and also the investigation and visiting experience, it is ready for us to do a summary on what we have seen and what we have learned.

We would introduce different speech recognition software in this chapter, which have their own advantages and drawbacks, and give a brief comparison at the end.

3.1 CMU Sphinx

3.1.1 Introduction to CMU Sphinx

Sphinx is originated at Carnegie Mellon University (CMU) and developed by a group of faculty members and students in department of computer science in that university. It is funded by DARPA (Defense Advanced Research Projects Agency).

Since the born the Sphinx Group is dedicated to release a set of reasonably mature, world-class speech components that provide a basic level of technology to anyone interested in creating speech-using applications without the once-prohibitive initial investment cost in research and development; the same components are open to peer review by all researchers in the field, and are used for linguistic research as well.

3.1.2 Sphinx2 Vs. Sphinx3

In early 2000, the Sphinx Group release Sphinx2, a real-time, large vocabulary, speaker-independent speech recognition system as free software. It is the engine used in the Sphinx Group's dialog systems that require real-time speech interaction, such as a many-turn dialog for travel planning. The pre-made acoustic models include American English and French in full bandwidth. Sphinx 2 is a decent candidate for handheld, portable, and embedded devices, and telephone and desktop systems that require short response times.

At the same year, Sphinx 3, a slower, yet more accurate recognizer is released. It is used, for example, broadcast news transcription. An Acoustic Trainer, allowing anyone to create acoustic models for conditions, channels, and domains, is released as well.

3.1.3 SphinxTrain

SphinxTrain is the acoustic training environment for CMU Sphinx (both sphinx2 and sphinx3), which was first publicly released on June 7th, 2001. It is a suite of programs, script and documentation for building acoustic models from data for the Sphinx suite of recognition engines. With this contribution, people should be able to build models for any language and condition for which there is enough acoustic data.

3.2 Microsoft SAPI

Microsoft® Speech API Version 5.0

3.2.1 Introduction to Microsoft SAPI

The Microsoft Speech API (SAPI) is a software layer allowing speech-enabled applications to communicate with speech engines (Speech Recognition (SR) engines and Text to Speech (TTS) engines). SAPI includes an Application Programming Interface (API) and a Device Driver Interface (DDI). Applications communicate with SAPI via the API layer and speech engines communicate with SAPI via the DDI layer.

3.2.2 Responsibilities of SAPI and SR Engine

A speech-enabled application and an SR engine do not directly communicate with each other -- all communication is done via SAPI.

SAPI takes responsibility for a number of aspects of a speech system, such as:

- Controlling audio input, whether from a microphone, files, or a custom audio source; and converting audio data to a valid engine format.
- Loading grammar files, whether dynamically created or created from memory, URL or file; and resolving grammar imports and grammar editing.
- Compiling standard SAPI XML grammar format, and conversion of custom grammar formats, and parsing semantic tags in results.
- Sharing of recognition across multiple applications using the shared engine. All marshalling between engine and applications is done by SAPI.
- Returning results and other information back to the application and interacting with its message loop or other notification method. This allows an engine to have a much simpler threading model than in SAPI 4, as SAPI 5 does much of the thread handling.
- Storing audio and serializing results for later analysis.
- Ensuring that applications do not cause errors -- preventing applications from calling the engine with invalid parameters, and dealing with applications hanging or crashing

The SR engine is responsible for:

- Using SAPI grammar interfaces and loading dictation.
- Performing recognition.
- Polling SAPI in order to learn about grammar and state changes.
- Generating recognitions and other events to provide information to the application.

3.2.3 SAPI SR Objects and Interfaces

In order for an SR engine to be a SAPI 5 engine, it must implement at least one COM object. Each instance of this object represents one SR engine instance. The main interface this object must implement is the ISpSREngine interface. SAPI calls the engine using the methods of this interface to pass details of recognition grammars and tell the engine to start and stop recognition etc. SAPI itself implements the interface ISpSREngineSite. A pointer to this is passed to the engine and the engine calls SAPI using this interface to read audio, return recognition results etc.

ISpSREngine is the main interface that needs implementing, but there are other interfaces that an engine may implement. The SR engine can implement the ISpObjectWithToken interface. This provides a mechanism for the engine to query and edit information about the object token in the registry used to create the engine.

There are two other interfaces than the engine can also implement. These each need to be implemented in separate COM objects, because SAPI needs to create and delete them independently of the main engine. These interfaces are:

- ISpSRAlternates, which can be used by the SR engine to generate alternates for dictation results. It is possible to generate alternates without this interface, but this interface allows alternates to be generated off-line, after the result has been serialized.
- ISpTokenUI allows engines to implement UI components that can be initialized from an application. These can be used to perform user training, add and remove user words, calibrate the microphone etc.

The engine can also implement another COM object allowing engine-specific calls between the application and the engine. This object can implement any interfaces, which the application is able to QueryInterface for.

3.3 IBM ViaVoice

3.3.1 What is IBM ViaVoice?

In 1994, IBM was the first company to commercialize a dictation system based on speech recognition. Since then IBM devoted to develop the most advanced technology in this field.

ViaVoice is the speech recognition system that developed by IBM and it supports 13 different languages. In September 1997, ViaVoice Chinese version came into being and it draw much attention to the people who concerned. It successfully solves the problems like there are many characters with the same pronunciation and different meaning in Chinese, Chinese language has different tones, etc.

3.3.2 System Characteristics

Since the first Chinese version of ViaVoice speech recognition system was born, IBM devoted to increase the accuracy of system. In 1999, they enhanced the system to be a user independent, no vocabulary constraint, and continuous speech recognition system with high recognition accuracy. And now they have three different versions including Mandarin, Taiwanese and Cantonese.

3.3.3 ViaVoice Native Architecture Overview

The heart of a speech recognition system is known as the speech recognition engine. The speech recognition engine recognizes speech input and translates it into text that an application can understand. Application can access the engine through a speech recognition API. For ViaVoice, this API is known as Speech Manager API, or SMAPI, for short. SMAPI is a conventional API, meaning that the API is defined as a part of resource; in ViaVoice, SMAPI is defined as part of the speech engine. With an API, speech becomes a resource to all applications.

3.3.4 ViaVoice Speech Engine Architecture

The speech engine has a rather complex task to handle, that is, taking the raw audio input and translating it to recognized text that an application understands.

The audio input source module encapsulates the methods used by the engine to retrieve the audio input stream. By default, the engine retrieves its audio input from the standard microphone input device in the system. A developer can write a custom audio library so that the input of the engine would be a custom piece of hardware.

The acoustic processor takes raw audio data and converts it to the appropriate format for use. The acoustic processor consists of two components: the signal processor and the labeler.

In the ViaVoice engine, audio input picked up by the microphone is analyzed by the signal processor. This raw audio data is captured at 22 kHz by default, but 11 kHz and 8 kHz sampling is also supported. It contains both speech data and background noise.

3.3.5 Application Programming Interfaces

ViaVoice SDK provides several programming interfaces that developers can use to incorporate speech into their application.

- *Speech Manager APIs (SMAPI):* IBM speech recognition engine APIs
- *SMAPI Grammar Compiler API:* APIs used to compile grammars used by the speech recognition engine.

3.3.5.1 SMAPI

There is significantly more function in the ViaVoice engine beyond raw recognition of spoken words, including dynamic vocabulary handling, database functions to query and select installed users, languages, and domains, and the ability to add new words to the user's vocabulary. SMAPI supports:

- Verifying the API version
- Establishing a database session to query system parameters

SMAPI is provided as a library, which is linked into an application. The engine is a separate executable. The ViaVoice architecture supports many speech applications through a single engine, connected to one microphone.

3.3.5.2 SMAPI Grammar Compiler API

The ViaVoice SKD provides the ability for developers to compile grammars programmatically. This function is know as the SMAPI Grammar Compiler API and supports:

- Specifying parameters to the grammar compiler
- Compiling a grammar
- Obtaining error messages from the grammar compiler

The SMAPI Grammar Compiler APIs are provided as a library, which is linked into an application.

3.4 Comparisons of Different SR Systems

After introducing three different SR systems, we want to do a comparison to show their characteristics and drawbacks.

	CMU Sphinx	Microsoft SAPI	IBM ViaVoice
Characteristics	a. Open source b. Free software, no need for initial investment c. Good for researchers and developers d. With relatively high quality	a. Applications and SR engine communicate with SAPI b. DDI and API remove implementation details making speech synthesis and SR engine and application convenient	a. SMAPI is define as a part of SR, which becomes the source of application b. Support 13 languages, including Cantonese and Chinese c. Developers can write audio library to handle input d. Support for Grammar Compiler APIs e. Support Dynamic vocabulary handling, database querying, add new words to the user's vocabulary
Drawbacks	a. Limited and hard to read documentation b. No Chinese version, although SphinxTrain can be used to build any language model, it needs large volume of acoustic data and investigation of the system c. SphinxTrain is only available in Unix system d. Acoustic build process can take many days (or longer)	a. Has to implement COM objects and interfaces for SR engine to be a SAPI 5 engine b. Limited language version c. Do not support grammar compiler	a. Constrained input audio data format b. Relatively low accuracy without training

Figure 3.1 Comparison of different SR systems

3.5 Why Choose ViaVoice?

As you can see from above table of comparison, IBM ViaVoice is the most distinguishable SR system. It has a high accuracy of dictation if fully trained. It uses 150,000-word base dictionary and user can add up to 64,000 words of their own. What's more important, it provide both Cantonese and Chinese version, which enable us to integrate it as a part of VIEW project.

When we visited IBM Research Lab China in Beijing this summer, we were really impressed by the outstanding performance of their product such as real-time dictation and telephony system, as well as the intelligence of the people there.

3.6 Our Objective With ViaVoice

In late September, VIEW lab got the ViaVoice SR system from IBM China. Our first task is to get the SR engine to work. And then, based on the knowledge we learned and SMAPI developer's guide, we are trying to increase the accuracy of the speech recognition engine as much as possible and obtain the timing information of speech. We apply some techniques such as audio extraction, segmentation and training to the raw audio data.

Although we have get some work done, SR is not an easy task. We meet some difficulties when doing the project. It will be describe at the end of our report. In the next a few chapters, we will describe the implementation detail of our project.

Chapter 4 : Visual Training Tool

Our project and also the speech recognition engine mainly deal with audio data. But in the digital video library, most data are stored as multimedia files, a mixture of both video and audio data. Therefore, the first step we need to take is to extract audio data from these multimedia files. Further more, we also need to apply some speech recognition techniques on these audio data. Thus, it is more convenient for us to store these data on some persistent storage rather than real-time extracting them out when needed. The IBM ViaVoice engine supports only monotonic, 22/16/8 kHz ACM data. Hence, we decided to store these audio data in monotonic, 22 kHz wave format.

4.1 Audio Extraction

Under Win32 environment, Microsoft DirectX provides a convenient multimedia library. We have developed our program for audio extraction right on the top of DirectX API. In the next section, we are going to introduce DirectX in more detail.

4.1.1 Microsoft DirectX

Microsoft® DirectX® is a set of low-level application programming interfaces (APIs) for creating games and other high-performance multimedia applications. Microsoft® DirectX® 8.1 is made up of the following components: DirectX Graphics, DirectX Audio, DirectInput, DirectPlay, DirectShow, and DirectSetup. Among these components, Microsoft DirectShow® provides for high-quality capture and playback of multimedia streams. It is the right component for us to achieve our goal. In order to write DirectShow program, we have to understand Microsoft DirectShow first.

DirectShow Overview

The Microsoft® DirectShow® application programming interface is a media-streaming architecture for the Microsoft Windows® platform. It supports a wide variety of formats, including Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), Audio-Video Interleaved (AVI), MPEG Audio Layer-3 (MP3), and WAV files.

DirectShow is based on the Component Object Model (COM). Most provided components, including filters and graphs, are in the form of COM-compliant object. DirectX programmers need to know at least how to use existing COM object.

DirectShow Application Programming

At the heart of DirectShow services is the modular system of pluggable components called filters, arranged in a configuration called a filter graph. A component called the filter graph manger oversees the connection of these filters and controls the stream's data flow.

Filter

The basic building block of DirectShow is a software component called a filter. A filter generally performs a single operation on a multimedia stream. For example, there are DirectShow filters that

- Read files
- Get video from a video capture device
- Decode a particular stream format
- Pass data to the graphics or sound card

Filters receive input and produce output. For example, if a filter decodes MPEG-1 video, the input is the MPEG-encoded stream and the output is an uncompressed RGB video stream.

Filter Graph

To perform a given task, an application connects several filters so that the output from one filter becomes the input for another. A set of connected filters is called a *filter graph*. As an illustration of this concept, the following diagram (figure 4.1) shows a filter graph for playing an AVI file:

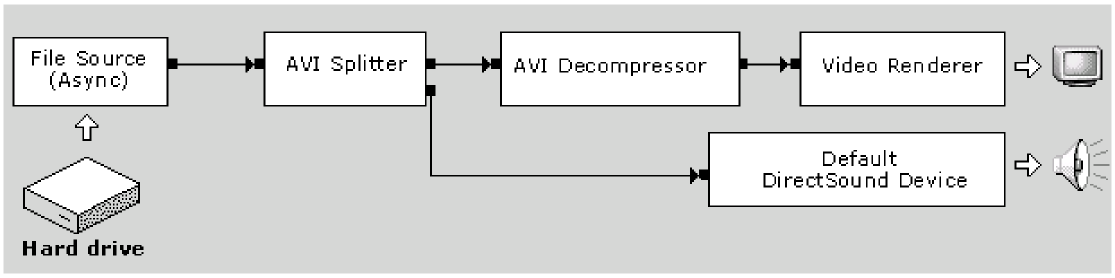


Figure 4.1 DirectShow Filter Graph

Filter Graph Manager

DirectShow provides a high-level component called the *Filter Graph Manager*. The Filter Graph Manager controls the flow of data through the graph. Our application makes high-level API calls such as "Run" (to move data through the graph) or "Stop" (to stop the flow of data). If we require more direct control of stream operations, we can access the filters directly through COM interfaces. The Filter Graph Manager also passes event notifications to the application, so that our application can respond to events, such as the end of a stream.

Writing a DirectShow application

A typical DirectShow application performs three basic steps, as illustrated in the following diagram.

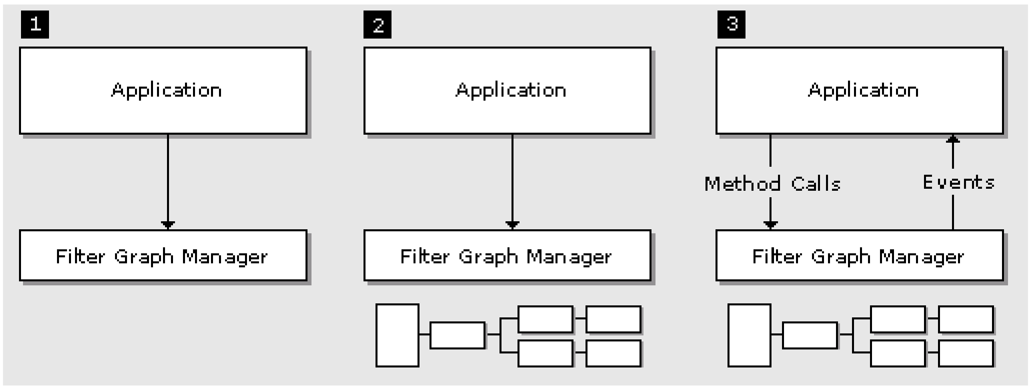


Figure 4.2 Process of Writing DirectShow Applications

1. Creates an instance of the Filter Graph Manager, using the **CoCreateInstance** function.

2. Uses the Filter Graph Manager to build a filter graph. (Other DirectShow helper components can be used here as well.)
3. Controls the filter graph and responds to events.

4.1.2 Implementation of Audio Extractor

The implementation of audio extractor employs some special filters to construct a filter graph, which is capable of converting other wave formats to the format desired by ViaVoice engine. This graph is also capable of storing wave streams into files.

Special Filters Employed in Our Program

Since our purpose is a little bit special, we need some special filters other than standard playback filters. There are two kinds of special filters needed. First, a transform filter is needed to produce the desired wave format. Second, a wave file writer is needed to write the extracted stream onto persistent storage. In our program, an ACM wrapper filter handles the transformation; a WavDest filter and a File Writer filter works together to handle the stream saving task.

ACM Wrapper Filter

The ACM Wrapper filter enables codes that use the Audio Compression Manager (ACM) to join a filter graph. It can act either as a decompression filter or as a compression filter. Decompression is only to PCM audio. It is used only for playback or wave format conversion.

In our program, we use it to convert the input PCM stream to monotonic, 22.05 kHz format. Since decompression is only to PCM audio, we connect the upstream to an audio decoder, which can produce PCM data. In DirectX, wave format is defined as a structure **WAVEFORMATEX**. The definition is

```
typedef struct {
  WORD  wFormatTag;        // Should be WAVE_FORMAT_PCM
  WORD  nChannels;         // Number of channels in the waveform-audio data
  DWORD nSamplesPerSec;    // Sample rate, in samples per second (hertz)
  DWORD nAvgBytesPerSec;   // required average data-transfer rate, in bytes per second
  WORD  nBlockAlign;       // the minimum atomic unit of data, in bytes
  WORD  wBitsPerSample;    // Bits per sample, usually equals 8 or 16
  WORD  cbSize;            // Size, in bytes, of extra format information appended
} WAVEFORMATEX;   // defines the format of waveform-audio data
```

In our program, the format information is store in a `WAVFORMATEX` structure pointed by `pFormat`. The core part of conversion is the following two lines of code:

```
pFormat->nSamplesPerSec = 22050;
pFormat->nChannels = 1;
```

To reserve the validity of the format, we need to update other fields accordingly:

```
pFormat->nBlockAlign = pFormat->nChannels * pFormat->wBitsPerSample / 8;
pFormat->nAvgBytesPerSec = pFormat->nBlockAlign * Format->nSamplesPerSec;
```

Now the upstream PCM audio data should be converted to monotonic, 22.05 kHz format.

WavDest Filter & File Writer

The WavDest filter multiplexes an audio stream to produce a stream, which is capable of being written to a file. It takes a single audio stream as input, and its output pin must be connected to the File Writer filter, which is capable of writing a multiplexed stream to a file. These two filters are used to save the stream produced by the ACM wrapper into a file.

Filter Graph of Audio Extractor

Now we use an example to illustrate the flow of the audio extractor visually. Since nowadays the most popular video format is MPEG, we choose an MPEG-I file, called "*tvbnews.mpg*", as the target to convert. The normal playback filter graph of the target file is shown in figure 4.3. On the top of the graph, a source filter is responsible for grabbing media stream from input file. Then the `MPEG-I Stream Splitter` divides the obtained media stream into two separate streams: video and audio. After that, the `MPEG Audio Decoder` and `MPEG Video Decoder` decode the corresponding stream data and pass them to output devices, namely `Video Renderer` and `DirectSound Device`, to render out.

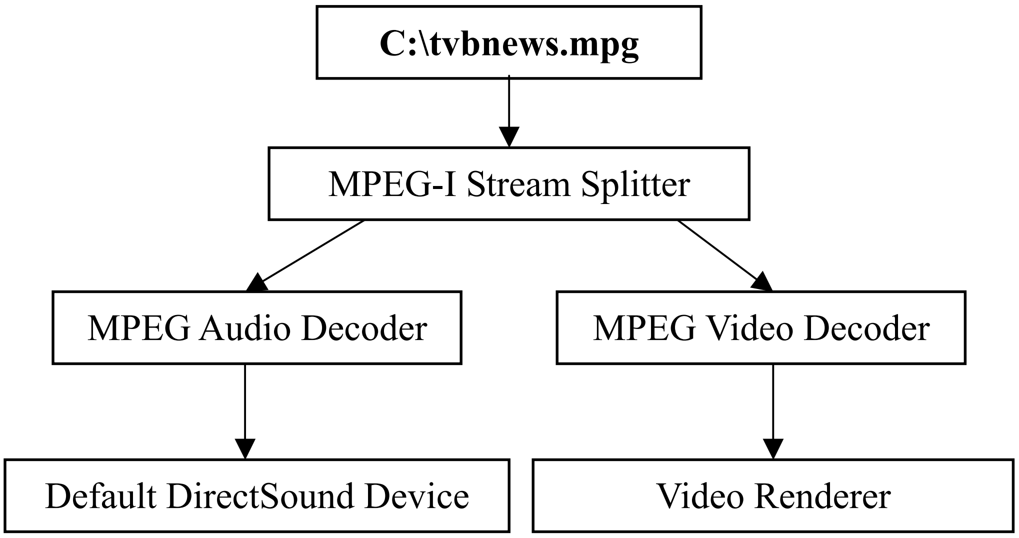


Figure 4.3 Filter Graph for Displaying Video Files

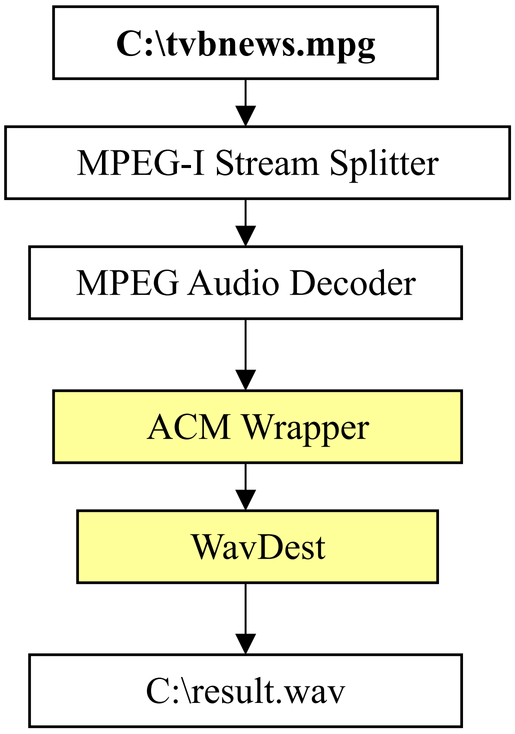


Figure 4.4 Filter Graph of Audio Extractor

The filter graph of audio extractor is shown in figure 4.4. The video data is no longer needed. So the video decoder is removed. And there is no need to render the streams to users; the sound and video render devices are also removed. Instead, the output stream of the MPEG Audio Decoder is directed to an ACM Wrapper, which convert the wave format and passes the stream to a WavDest Filter, connected with a File Writer, to save the output to a file.

4.1.3 Dictation by Untrained ViaVoice Engine

After audio extraction, the extractor generated a wave file named "*tvbnews.wav*", which is in the right format (monotonic, 22 kHz), from the original MPEG file "*tvbnews.mpg*" (A frame of this file is shown in figure 4.4). Now the ViaVoice speech engine could process the speech data in "*tvbnews.wav*" using a customized audio library named AudWav. (For more information on AudWav audio library, please refer to Chapter 6, Section 6.2.9.) Figure 4.5 is a screen shot of the dictation interface.

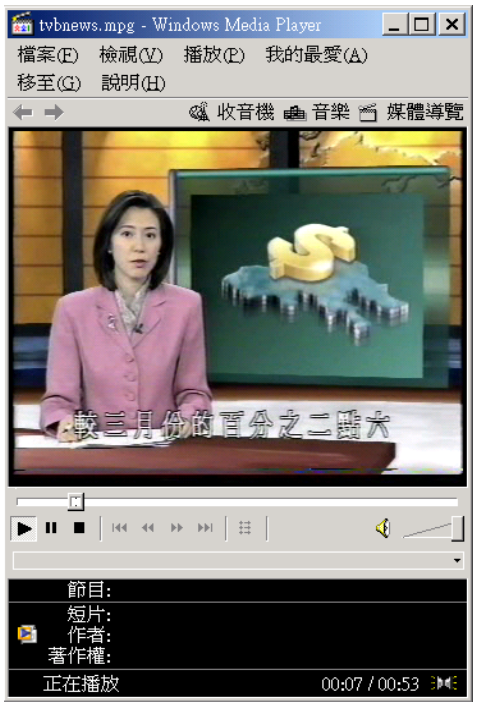
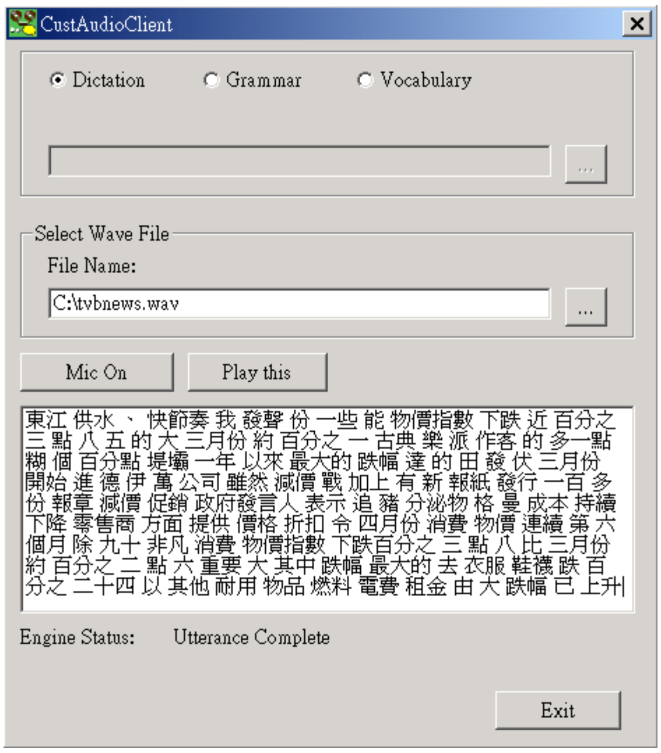


Figure 4.4 Media File "*tvbnews.mpg*"



Select Wave File
File Name: C:\tvbnews.wav

```
東江 供水 、 快節奏 我 發聲 份 一些 能 物價指數 下跌 近 百分之
三 點 八 五 的 大 三月份 約 百分之 一 古典 樂 派 作客 的 多一點
糊 個 百分點 堤壩 一年 以來 最大的 跌幅 達的 田 發 伏 三月份
開始 進 德 伊 萬 公司 雖然 減價 戰 加上 有 新 報紙 發行 一百 多
份 報章 減價 促銷 政府發言人 表示 追 豬 分泌物 格 曼 成本 持續
下降 零售商 方面 提供 價格 折扣 令 四月份 消費 物價 連續 第 六
個月 除 九十 非凡 消費 物價指數 下跌百分之 三 點 八 比 三月份
約 百分之 二 點 六 重要 大 其中 跌幅 最大的 去 衣服 鞋襪 跌 百
分之 二十四 以 其他 耐用 物品 燃料 電費 租金 由 大 跌幅 已 上升
```

Engine Status: Utterance Complete

Figure 4.5 Dictation Result of "*tvbnews.wav*"

The results are shown in figure 4.6. There are 165 characters out of 249 characters that are recognized correctly. Thus the accuracy of untrained ViaVoice engine is approximately 66.3% relative to the speech in "*tvbnews.wav*". We can see that the accuracy is not very high. Therefore, we need to do something to improve the performance of ViaVoice engine.

The next a few sections will introduce the techniques we used to improve the accuracy of the ViaVoice speech engine.

本港通縮的情況係持續惡化，四
月份的消費物價指數下跌了百分
之三點八。比較係三月份的百分
之二點六，係再下跌多一點二個
百分點。係八一年以來最大的跌
幅。羅佩瓊報道。三月份開始，
長途電話公司出現減價戰，加上
有新報紙發行，引發多份報章減
價促銷。政府發言人表示，隨著
本地物價同埋成本持續下降，零
售商普遍提供價格折扣，令到四
月份的消費物價連續第六個月出
現收縮。四月份消費物價指數下
跌百分之三點八，比三月份的百
分之二點六仲要低，其中跌幅最
大的係衣服鞋襪，跌百分之二十
幾。而其它耐用物品、燃料、電
費、租金都有好大跌幅。而上昇
的只有煙酒同埋交通費。亞洲電
視記者羅佩瓊。

東江 供水 、 快節奏 我 發聲 份
一些 能 物價指數 下跌 近 百分之
三 點 八 五 的 大 三月份 約 百
分之 一 古典 樂 派 作客 的 多一
點 糊 個 百分點 堤壩 一年 以來
最大的 跌幅 達 的 田 發 伏 三月
份 開始 進 德 伊 萬 公司 雖然
減價 戰 加上 有 新 報紙 發行 一
百 多 份 報章 減價 促銷 政府發
言人 表示 追 豬 分泌物 格 曼 成
本 持續 下降 零售商 方面 提供
價格 折扣 令 四月份 消費 物價
連續 第 六個月 除 九十 非凡 消
費 物價指數 下跌百分之 三 點 八
比 三月份 約 百分之 二 點 六 重
要 大 其中 跌幅 最大的 去 衣服
鞋襪 跌 百分之 二十四 以 其他
耐用 物品 燃料 電費 租金 由 大
跌幅 已 上升 約 只有 煙斗 爲 簡
化 亞洲 電視 記者 劉 佩 瓊

(a) Result of Human Recognition (b) Result of ViaVoice Engine

Figure 4.6 Recognition Results by Human and ViaVoice Engine

4.2 Speech Segmentation

As shown above, the dictation result produced by an untrained ViaVoice engine is full of errors. So we need to apply some speech recognition techniques to improve the accuracy of the engine. One of these techniques is to train certain number of models for the engine. It is being expected that such training will offer a perceptible increase in dictation accuracy.

4.2.1 Why To Do Speech Segmentation

Before actually train the engine, we need to some preprocessing on the audio data. The first one is to use speech segmentation and classification techniques to achieve silence removal.

There are mainly three reasons to do speech segmentation.

First, storage consumption is considerable for multimedia files. The silence in speech does consume disk space. Silence removal could save some storage.

Second, the training of ViaVoice engine is not done randomly. We need to provide some guidance to the engine. We plan to compare the real texts and the dictation result of the engine and pick up only those words, which are recognized correctly, to feed to the engine during the training process. Most of these real texts will be human input. Due to the usually long duration of media files, it will be more convenient to the speech interpreter, which is a human but not a machine, if the whole speech is segmented sentence by sentence and played back a single sentence once.

Third, during the training, we need to compare the real texts and the result produced by the speech engine to retrieve the words, which are recognized correctly. Such a comparison is done by a string alignment algorithm. The outcome of such a string alignment algorithm will be better if the length of the compared strings is shorter. And also the computation time increased with the length of input strings dramatically. Therefore, it will be better to do this string alignment sentence by sentence rather than on the whole lengthy speech.

4.2.2 Overview of Different Approaches

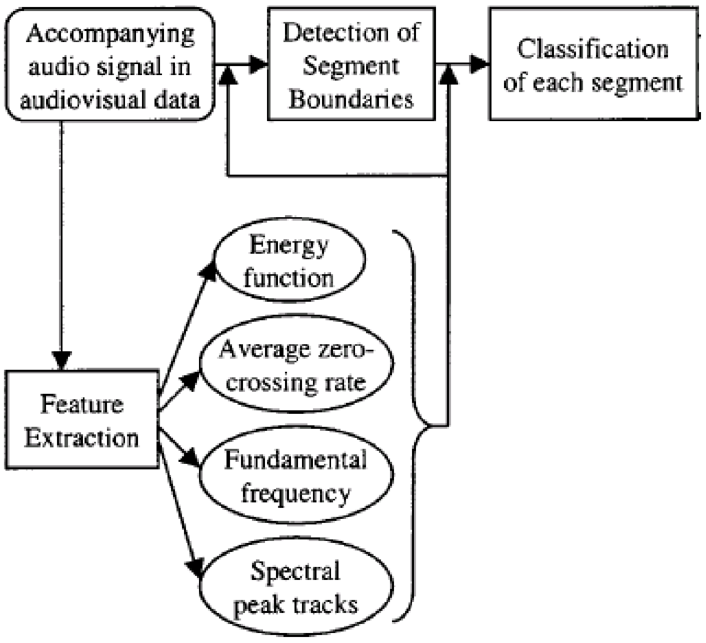

Accompanying audio signal in audiovisual data
Detection of Segment Boundaries
Classification of each segment
Feature Extraction
Energy function
Average zero-crossing rate
Fundamental frequency
Spectral peak tracks

Figure 4.7 Speech Segmentation Using Feature Extraction

The basic idea of speech segmentation is to use certain features of the boundary data to determine the boundary position of each segment. This is called feature extraction. Figure 4.7 shows the flow of speech segmentation procedure using feature extraction. As shown in the above figure, there are different approaches to detect the boundaries of speech segments.

- *Energy function:* define a short-time energy function of an input audio signal. It provides a convenient representation of the amplitude variation over time.
- *Average zero-crossing rate:* for discrete-time signals, a zero-crossing is said to occur if successive samples have different signs. The rate at which zero-crossing occur is a simple measure of the frequency content of a signal. It can be used as another measure to distinguish between voiced and unvoiced speech because unvoiced speech usually has much higher ZCR values than voiced ones.
- *Fundamental frequency:* a harmonic sound consists of a series of major frequency components including the fundamental frequency and those which are

integer multiples of the fundamental one. We may divide sounds into two categories, i.e., harmonic and nonharmonic sounds. Whether an audio segment is harmonic or not depends on its source. The speech signal is a harmonic and nonharmonic mixed sound, since voiced components are harmonic while unvoiced components are nonharmonic. We can also use this characteristic to separate them.

4.2.3 Frame Energy Analysis

Upon the current phase of our project, the feature extraction target is mainly silence. The frame energy model is better at detecting silence than other approaches. Hence, we choose it for the implementation of this phase.

This approach works because the energy for unvoiced speech components is in general significantly smaller than those of the voiced components, as shown in Figure 4.9. The low energy parts in the graph show the position of silences in the speech. This feature provides a basis for distinguishing voiced speech components from unvoiced ones.

How to measure the energy of speech? In fact, there is some relationship between energy and wave amplitude of speech. Figure 4.8 shows the waveform of the same speech as in figure 4.9. It is not hard to see that the lower the amplitude, the higher the energy. And the amplitude data can be retrieved from the wave file. Hence, we established a threshold on wave amplitude to determine silence in the speech. If the amplitude of certain sample is below the threshold, which means its energy is considerably low, then it is classified as silence.

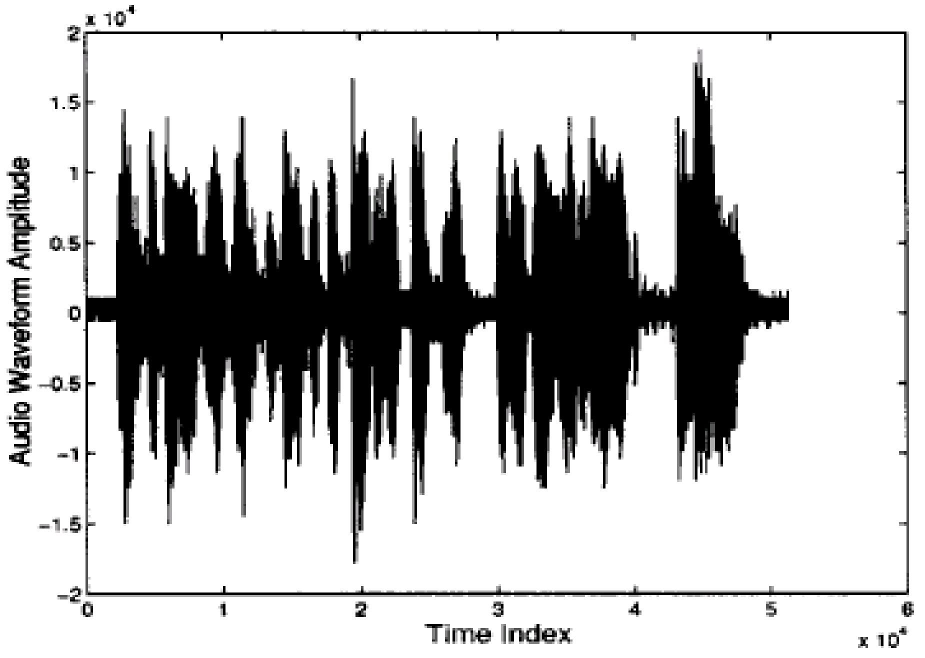


Figure 4.8 Waveform of a Speech Segment

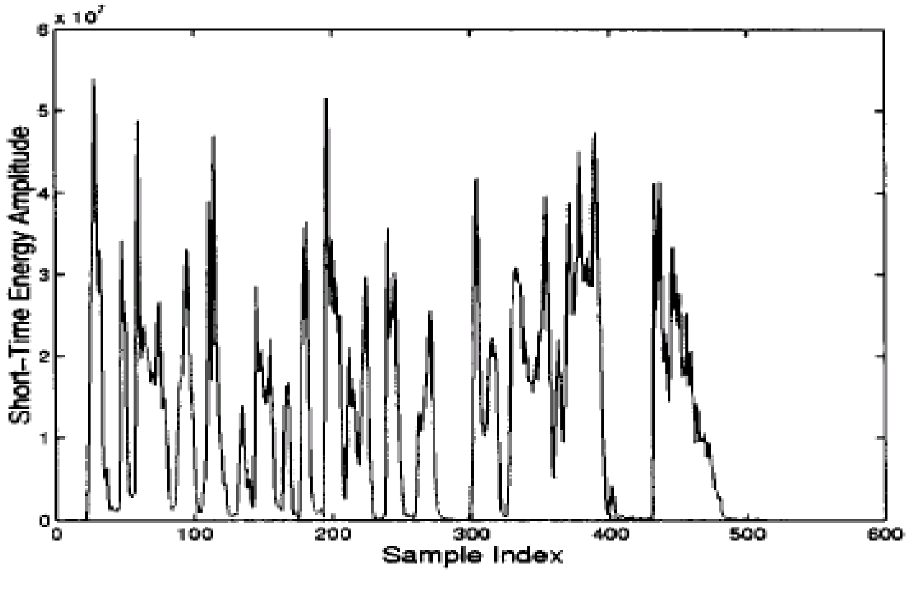


Figure 4.9 Frame Energy Graph of a Speech Segment

4.2.4 Segmentation Result

We implement the segmentation program using Matlab. A C/C++ version of this program is currently under development. Once again we come back to the old sample file "*tvbnews.mpg*" (figure 4.4 in Chapter 4), which is used as an example in Section 4.3. After extracting the audio data and saving in a file named "*tvbnews.wav*", we segment the speech in this file into different parts based on the frame energy of the wave. The result is shown visually in figure 4.10. The small green dots in the wave diagram indicate silence, which, in turn, segments the speech into different parts. Every part of the green line on the top of the graph represents a segment of the original speech. There are 24 such segments.

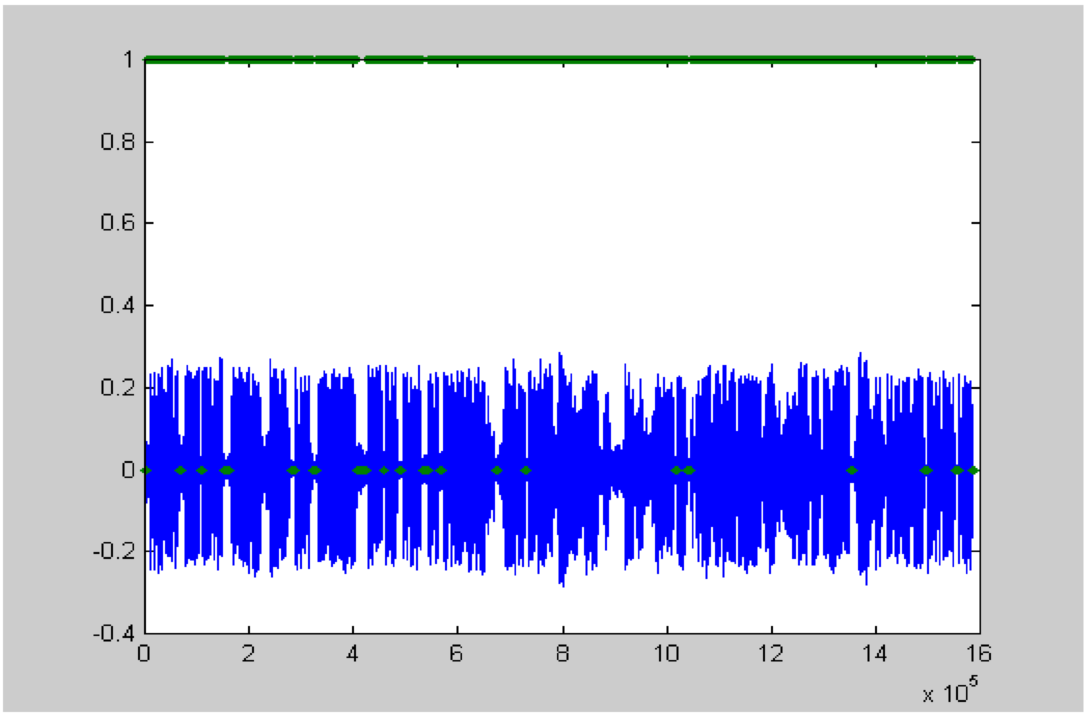


Figure 4.10 Result of Segmentation on file "*tvbnews.wav*"

The result is stored in a file named "*tvbnews.txt*", which records the start and end sample index of each segment. For example, the first two numbers stored in "*tvbnews.txt*" are 9901 and 133100. This means the first segment begins at the 9901st wave sample and ends at the 133100th wave sample. And also the 24 segments are saved separately as 24 small wave files for further manipulation.

4.3 Visual Training Tool

As mentioned before, training of a speech recognition system means letting the specific speaker to read a predefined paragraph of text and adjusting the language model to match that particular speaker's voice. However, in our system, we cannot employ the speaker to record his/her voice beforehand. For example, when we get the audio file from TVB news, it is impossible to find the speaker and ask him/her to do the training.

So in order to increase the accuracy of the ViaVoice system, our solution is training the system "manually". After preprocessing, the speech data are not ready for training yet. We need to compare the output of the speech engine with the correct texts and find those correctly recognized words. Before we can do this comparison, we need to collect large amount of text data. Such input work will require considerable effort under the condition that lack of the help of certain appropriate tool. Thus, we decided to implement a visual training tool to help us train the ViaVoice engine.

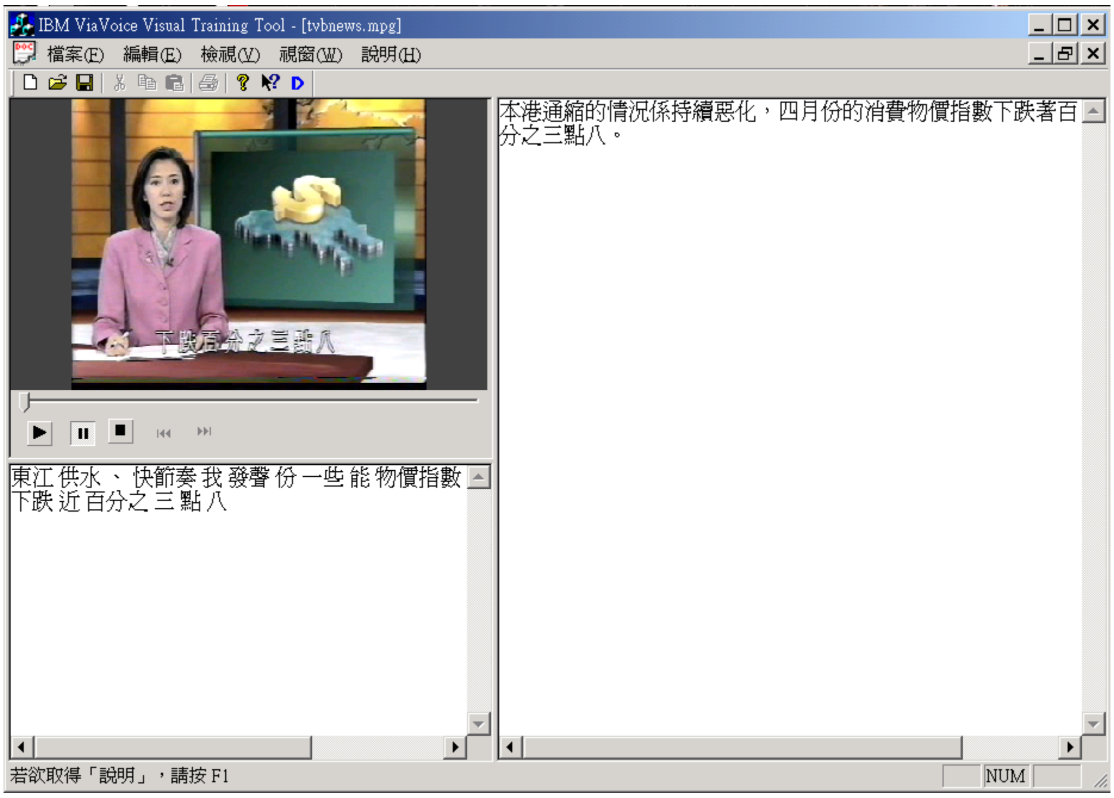
First, the tool displays the video, which contains that speaker's voice we want to train. When the user hears what the speaker is saying, he/she can type in the texts of what they have heard. In addition, the dictation output of IBM ViaVoice engine is also shown in the interface.

In order to let user catch up what the speaker is saying and record it correctly, we display the video sentence by sentence.

Second, we should use some algorithm to compare the two outputs (one by ViaVoice engine and one by user input). For those characters that are recognized as correct by speech recognition engine, we record them and use the data to do the training.

This tool is named WinTrain, which was developed using Microsoft Visual C++. The next section is going to introduce the features of the WinTrain tool.

4.3.1 WinTrain Features


IBM ViaVoice Visual Training Tool - [tvbnews.mpg]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
下跌百分之三點八
NUM

東江 供水 、 快節奏 我 發聲 份 一些 能 物價指數 下跌 近 百分之 三 點 八

本港通縮的情況係持續惡化，四月份的消費物價指數下跌著百分之三點八。

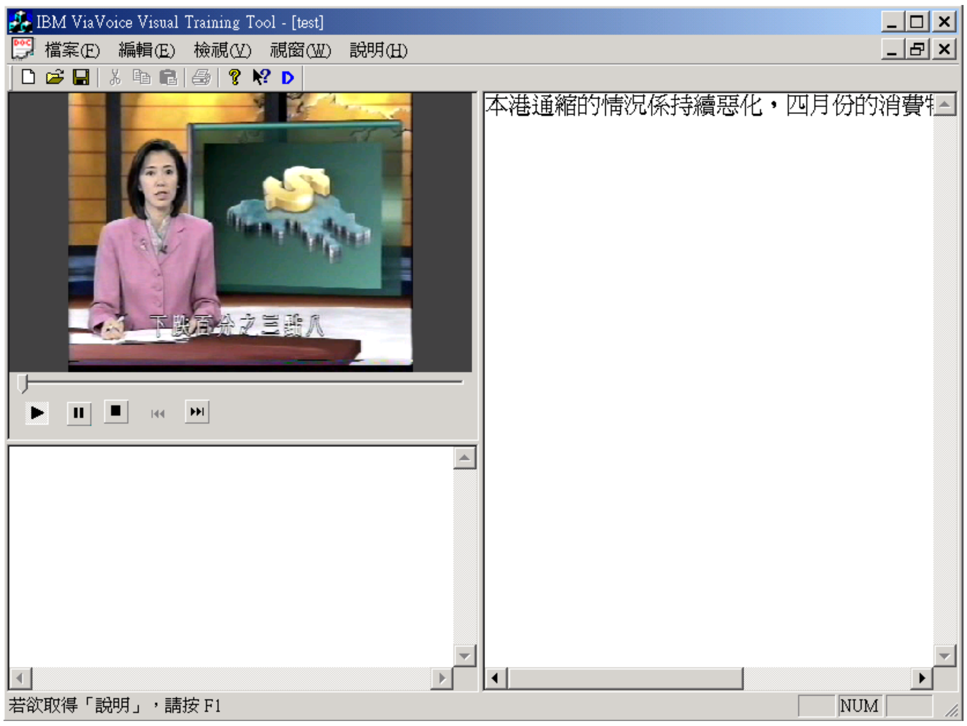
若欲取得「說明」，請按 F1

Figure 4.11 Main Interface of WinTrain

This tool is running on Microsoft Windows with DirectX and IBM ViaVoice Runtime installed. The main window is split into three parts, as shown in figure 4.11. The left upper window is the video window, where the opened media file is played. The window just below the video window is the dictation window, where the result text of speech recognition is displayed. The right window is a text editor. It is for the user to input captions of the media file. After the document is saved, user input captions are also saved.

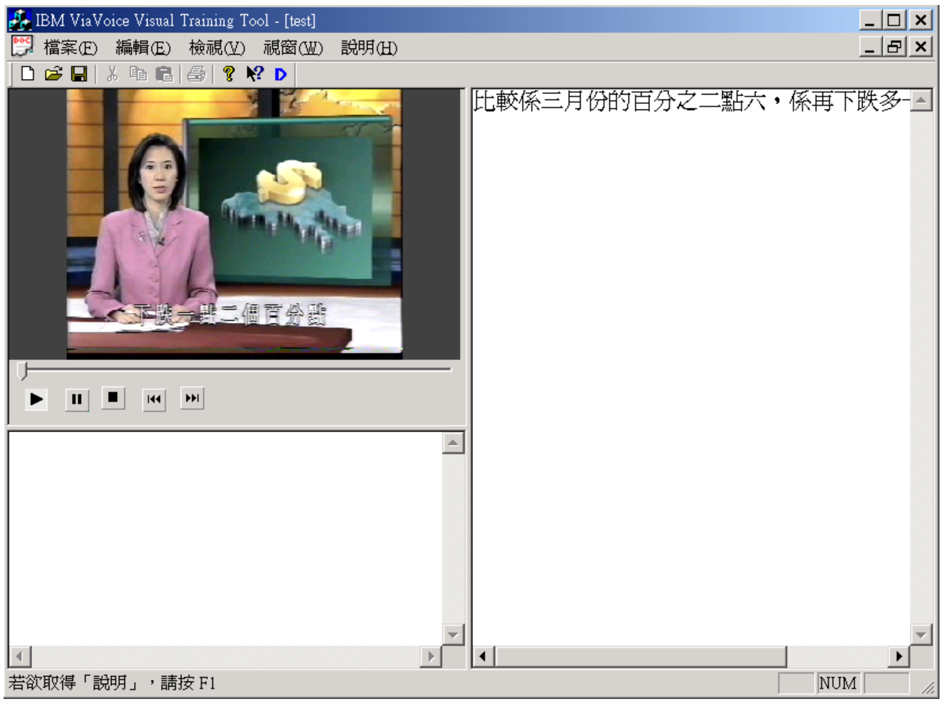
This tool can also process the speech segmentation information. Prior to open a media file, the tool will search for a text file with the same name of the media file. For example, if the media file is "*tvbnews.mpg*", the expected segmentation information file is "*tvbnews.txt*". If this file is present and is in a valid format, the tool will split the whole media content into segments, whose positions and durations are specified in the segment file. If there is more than one segment in a media file, the previous and next button in the video window will be available and the video will be played one segment each time. At the end of each segment, the player will stop automatically until the user presses the next button. If the user presses the previous button, the currently playing segment will be stopped and the last segment will be load and played. The texts in the text editor are refreshed automatically according to the segment that is currently played. Only the texts corresponding to the currently played segment are shown in the editor.

Figure 4.12 (a through c) shows a multi-segment media being played by the tool. The segment being played in (a) is the first segment of the media. After the user presses the next button, the currently played segment will be switched to the second segment, which is shown in figure 4.12(b). If the user presses the next button again, the third segment will be played, as shown in figure 4.12(c). The user could go back to (b) from (c) or go back to (a) from (b) by pressing the previous button. Note that the editor text is also refreshed corresponding to the currently played segment.


IBM ViaVoice Visual Training Tool - [test]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
若欲取得「說明」，請按 F1
NUM

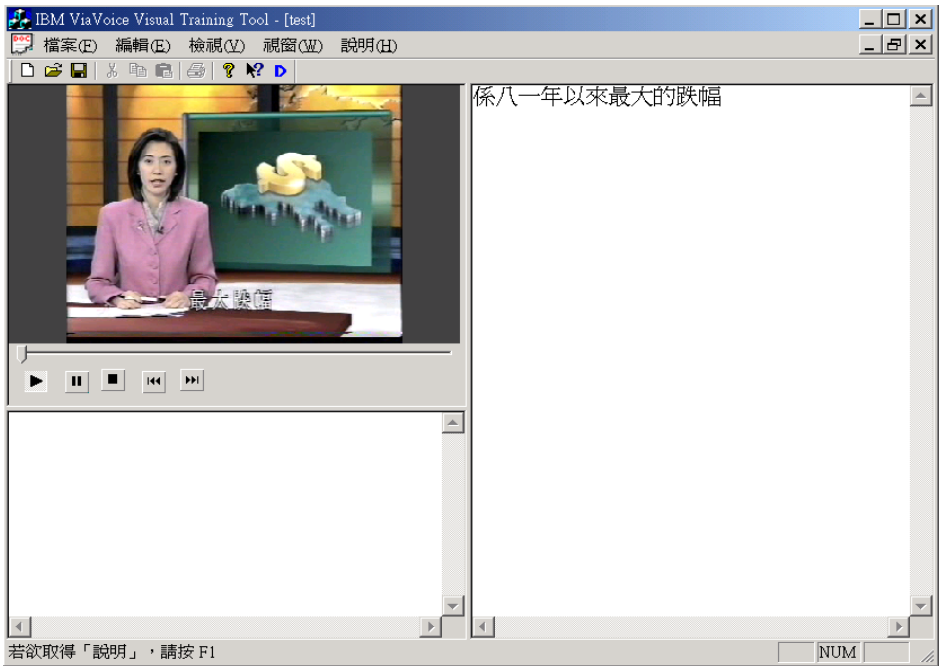
本港通縮的情況係持續惡化，四月份的消費

(a) First Segment


IBM ViaVoice Visual Training Tool - [test]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
下跌一點二個百分點
若欲取得「說明」，請按 F1
NUM

比較係三月份的百分之二點六，係再下跌多一

(b) Second Segment


IBM ViaVoice Visual Training Tool - [test]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
最大跌幅
若欲取得「說明」，請按 F1
NUM

係八一年以來最大的跌幅

(c) Third Segment

Figure 4.12 (a ~ c) Demonstration on Video Segment Switching

There is a dictation button in the toolbar of the main window, as shown in figure 4.13. After this button is pressed, the speech of the currently played segment will be passed to the ViaVoice speech recognition engine, and the result will be shown in the dictation window. The result will be used further with the string alignment algorithm.

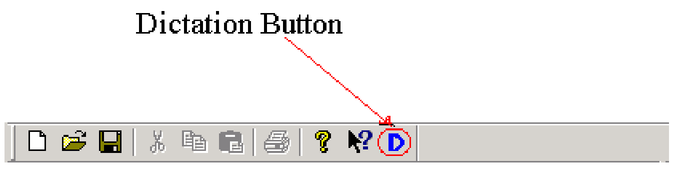


Figure 4.13 Dictation Button

The visual train tool is also capable of processing multiple media files simultaneously, as shown in figure 4.14.

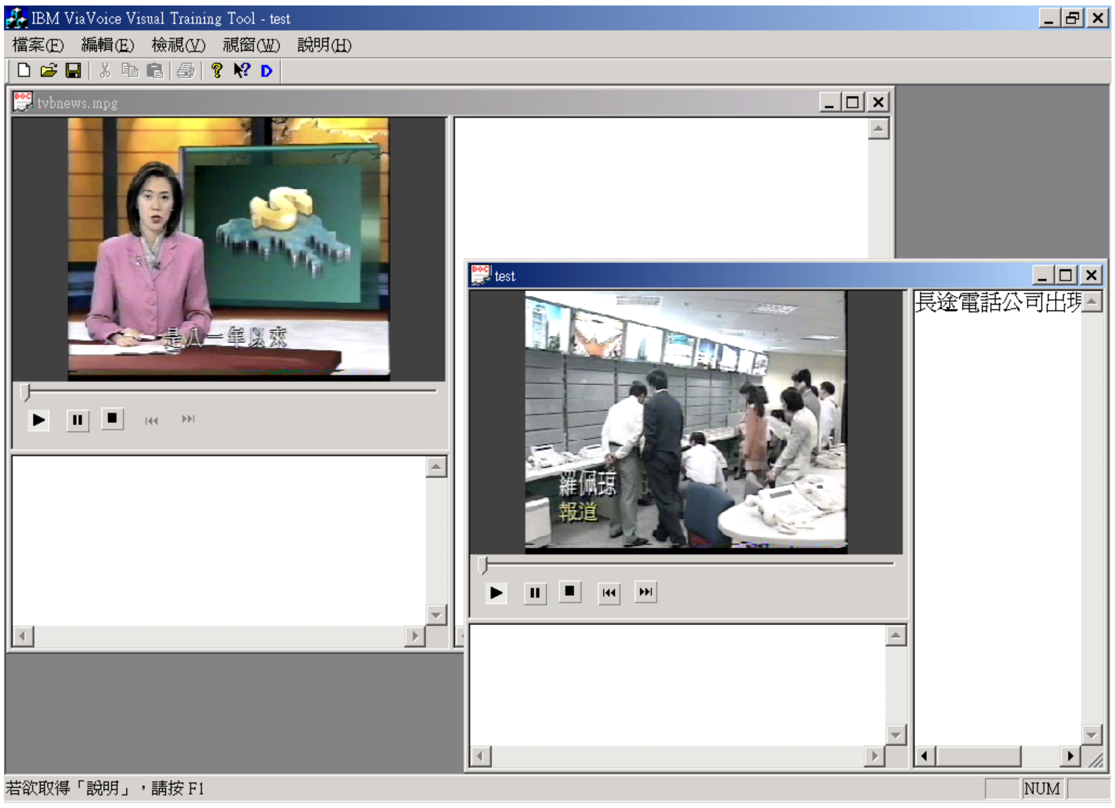

IBM ViaVoice Visual Training Tool - test
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
tvbnews.mpg
test
長途電話公司出現
若欲取得「說明」，請按 F1
NUM

Figure 4.14 Multiple Documents Demonstration

Now we are going to discuss the implementation of the visual training tool.

4.3.2 Implementation of The Visual Training Tool

The visual training tool is developed using Microsoft Foundation Classes. The document/view architecture is adopted.

Document and View

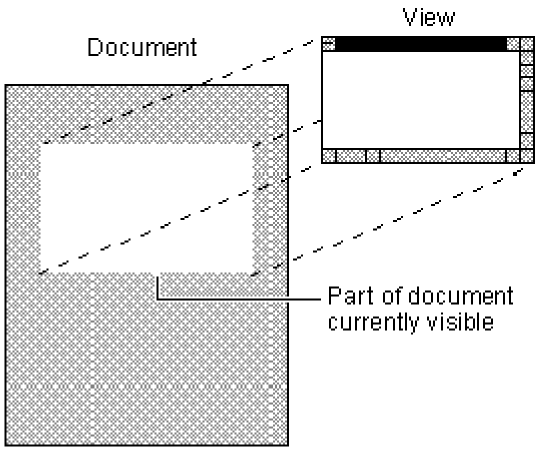


Figure 4.15 Document and View Architecture

The document/view implementation separates the data itself from its display and from user operations on the data. All changes to the data are managed through the document class. The view calls this interface to access and update the data.

Documents, their associated views, and the frame windows that frame the views are created by a document template. The document template is responsible for creating and managing all documents of one document type.

The key advantage to using the MFC document/view architecture is that the architecture supports multiple views of the same document particularly well. In the visual training tool, there are three separated views, including the video view, the dictation view, and the editor view. All these three views would be associated with a single document object. The document stores the data. All views access the document and display the data they retrieve from it. This scenario would be difficult to code without the separation of data from view, particularly if the views stored the data themselves.

Object Diagram

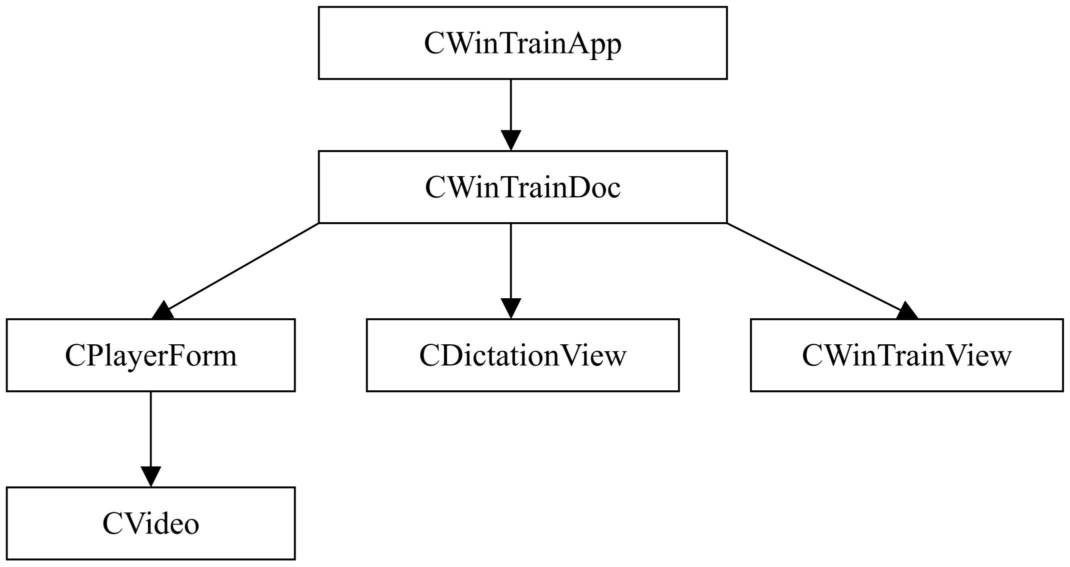


Figure 4.16 Object Diagram of The Visual Training Tool

CWinTrainApp

This is the class of the main application. There will be one and only one CWinTrainApp object. This object maintains a document template, which manages the interaction between the document and its associated views. The prototype of this class is

```
class CWinTrainApp : public CWinApp
{
public:
    CWinTrainApp();
    virtual BOOL InitInstance();
        virtual int ExitInstance();

// Implementation
    afx_msg void OnAppAbout();
    DECLARE_MESSAGE_MAP()
private:
    BOOL m_bATLInited;
    BOOL InitATL();
}
```

CWinTrainDoc

This is the one and only one type of document in the visual training tool. It manages the document data, including the media file, the segmentation information, and the user-entered texts. All views are updated according to the document object. The prototype of this class is

```
class CWinTrainDoc : public CDocument
{
protected: // create from serialization only
    CWinTrainDoc();
    DECLARE_DYNCREATE(CWinTrainDoc)
// Overrides
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
    virtual BOOL OnSaveDocument(LPCTSTR lpszPathName);
// Implementation
public:
    void UpdateTextView(LONG index);
    void SetTextAt(LONG index);
    void SetTextAt(LONG index, CString text);
    CList<Entry, Entry&> & GetEntryList();
    CString GetMediaPath();
    virtual ~CWinTrainDoc();
protected:
    BOOL GetSegmentInfo(CString fileName);
    CList<Entry, Entry&> m_EntryList;
    CPlayerForm *GetPlayerForm();
    CEditView *GetEditView();
    CArray <CString, CString> m_TextArray;
    CFile m_textFile;
    CString m_textFileName;
    CString m_mediaFileName;
    afx_msg void OnAppDictate();
```

```
        DECLARE_MESSAGE_MAP()
    }
```

CPlayerForm

This form is the video view of the document. One of its responsibilities is to lay out the video window and control panel. There is a `CVideo` object embedded. The user inputs, such as play and stop and so on, are passing to it. This form is also responsible for update the `CVideo` object upon changed made to the document. The prototype of this class is

```
    class CPlayerForm : public CFormView
    {
    protected:
        CPlayerForm();          // protected constructor used by dynamic creation
        DECLARE_DYNCREATE(CPlayerForm)
    // Form Data
    public:
        enum { IDD = IDD_PLAYER_FORM };
        CBitmapButton m_StopButton;
        CBitmapButton m_PrevButton;
        CBitmapButton m_NextButton;
        CBitmapButton m_PauseButton;
        CBitmapButton m_PlayButton;
        CStatic  m_Screen;
    // Operations
    public:
        LONG GetCurrentEntry();
        void SetMediaPath(CString path);
    // Overrides
        public:
        virtual void OnInitialUpdate();
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    // Implementation
    protected:
        CBitmap m_PlayBitmap;
        CVideo * m_Video;
        virtual ~CPlayerForm();

        //message map functions
        afx_msg void OnPlayButton();
        afx_msg void OnPauseButton();
        afx_msg void OnStopButton();
        afx_msg void OnNextButton();
        afx_msg void OnPreviousButton();
        DECLARE_MESSAGE_MAP()
    }
```

CDictationView

This view implements the dictation view of the visual training tool. If a user presses the dictation button, the currently played speech will be fed to ViaVoice engine to produce the text. And the result is displayed on this view. The prototype of this class is

```
    class CDictationView : public CEditView
    {
    protected:
        CDictationView();           // protected constructor used by dynamic
    creation
        DECLARE_DYNCREATE(CDictationView)

    // Overrides
        protected:
        virtual void OnDraw(CDC* pDC);      // overridden to draw this view

    // Implementation
    protected:
        virtual ~CDictationView();
        DECLARE_MESSAGE_MAP()
    }
```

CWinTrainView

This view implements a text editor, which is used for a user to type down what he hears from the media speech. The whole texts are also segmented, as the same way as the speech. Only texts related to the currently played speech segment are displayed. The prototype of this class is

```
class CWinTrainView : public CEditView
{
protected: // create from serialization only
    CWinTrainView();
    DECLARE_DYNCREATE(CWinTrainView)

// Attributes
public:
    CWinTrainDoc* GetDocument();

// Overrides
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);

// Implementation
public:
    virtual ~CWinTrainView();
}
```

CVideo

This class implements a customized media player, which supports segmentation of the media content. The segmentation information is provided during construction. If no such information supplied, the default segmentation is to regard the whole media content as a single segment. All operations other than previous and next, including play, pause, and stop, are relative to the current active segment. A user can switch between segments only by pressing the previous or the next button. The concept of media segment is encrypted inside a class named CMediaSegment. A list of CMediaSegment and the index of current active segment are maintained in class CVideo in order to perform the switching between segments. This player requires Microsoft DirectShow to work. All formats supported by DirectShow are therefore supported by this player. The prototypes of class CMediaSegment and CVideo are

```
class CMediaSegment
{
public:
     CMediaSegment();
     CMediaSegment(REFTIME start, REFTIME stop);
     ~CMediaSegment();

public:
     REFTIME GetStartTime();
     REFTIME GetStopTime();
     void SetStartTime(REFTIME start);
     void SetStopTime(REFTIME stop);
};

typedef CArray<CMediaSegment, CMediaSegment&> MediaEntries;

class CVideo
{
public:
     CVideo(CString file, CWnd * pWnd, MediaEntries & entries, CRect rect =
     CRect(0,0,0,0));
     ~CVideo();

public://function
     LONG GetCurrentEntry();
     LONG GetPreviousEntry();
     LONG GetNextEntry();
```

```
        void Previous();
        void Next();
        int Seek(LONGLONG i);
        double GetAvgTimePerFrame(){return sPF;};
        LONGLONG GetWidth(){return Width;};
        LONGLONG GetHeight(){return Height;};
        LONGLONG GetLengthInFrames ();
        void Initial(CRect);
        void ReleaseDirect();
        HBITMAP  GetCurrentBitmap(int);
        char*     GetImagePtr (long seek_to_frame);
        void Play();
        void Stop();
        void ToEnd();
        void ToStart();
        void Pause();
        void SpeedUp();
        void SpeedDown();
        void SetVolume(long);
        long GetVolume();
        void SetWindowPosition(CRect rect);
        long GetDuration();
        long GetPosition();
        void GetWindowPosition(long* left,long*top,long*,long*);
        long GetVideoStatus();
        LONGLONG SetPosition(GUID,LONGLONG);
    };
```

Customized Audio Library

The dictation button in the toolbar of the main window of the visual training tool enables real time dictation. This task requires passing audio data to the speech engine, getting the dictation result, and displaying the texts in the dictation window.

As introduced in Chapter 3, Section 3.3.4, the audio input source module encapsulates the methods used by the engine to retrieve the audio input stream. By default, the engine retrieves its audio input from the standard microphone input device in the system. A developer can write a custom audio library so that the input of the engine would be a custom piece of hardware. The audio library functions must be packaged as a library, On Windows this is a Dynamically Linked Library (DLL), on Unix based systems and Macintosh it is a shared library. The implementations of the audio library functions are determined by the requirements of the application developer. The audio library will be loaded by the engine upon the first client connection. The engine will then resolve audio library exported function addresses. The audio library will be unloaded by the engine when the last client disconnects and a new client connects specifying a different audio source. The engine will also unload the audio library when the engine is terminating.

The acoustic processor takes raw audio data and converts it to the appropriate format for use. The acoustic processor consists of two components: the signal processor and the labeler.

The following 10 functions must be implemented in an audio library.

```
AudioConnect Function
AudioCreate Function
AudioDestroy Function
AudioDisconnect Function
AudioGetPCM Function
AudioPutPCM Function
AudioStartPlayback Function
AudioStartRecording Function
AudioStopPlayback Function
AudioStopRecording Function
```

In our case, audio sources are media files. Therefore, we need to a customized audio source library, which takes raw audio data from media files and converts these data into formats that will be accepted by the IBM ViaVoice engine.

Currently, The visual training tool uses an audio library named AudWav. This library is capable of retrieving audio data from wave files only. Thus we need to store every segment of a media file as a single wave file. These small wave files may not be needed for other tasks. Another audio library, namely AudMedia, is under development. This library will support arbitrary media files, given that they are supported by DirectShow, and will be capable of retrieving audio data from media files directly.

Chapter 5 : Speech Recognition Experiments

Using our virtual training tools, we can get both the output text of ViaVoice SR engine and the correct transcript input by user. We are now ready to have a test on the performance of IBM ViaVoice.

Before doing the experiments, we have

- Selected 77 news video clips, each of which is about 30 minutes long, to form our video collection for speech recognition;
- Employed our audio extraction module to extract audio channels (stereo 44.1 kHz) from selected video files into wave files (mono 22 kHz);
- Used our audio segmentation module to segment the extracted wave files into sentences by detecting its frame energy with a median filter to eliminate the false segment;
- Employed 7 student helpers to produce transcripts of the selected 77 news video clips using our visual training tool.

After these preparations have been done, we performed our experiments in early Spring 2002.

5.1 Four Kinds of Experiments

In order to measure how well speech preprocessing and trained speech model can improve the performance of IBM ViaVoice engine, we have conducted the speech recognition experiments under different situations. Thus we performed four kinds of experiments: baseline measurement, trained model measurement, slow-down measurement, and indoor news measurement.

5.1.1 Baseline Measurement

In this experiment, we measured the raw accuracy of ViaVoice engine. The speech model was not trained specifically, so that it was the same as when the engine package was shipped to us. We selected 10 video clips from our video collection for this experiment. Then we extracted the audio channels form the selected clips and stored them as wave files. These wave files were not supplied to ViaVoice directly. We preprocessed them and segmented them into sentences. The visual training tool provided a neat interface for users to produce transcripts for segmented audio. With this tool, our student helpers could easily produce the transcripts for our video collection. The baseline of word recognition accuracy was measured by the Hidden Markov Model Toolkit (HTK).

5.1.2 Trained Model Measurement

This experiment is to measure the accuracy of ViaVoice engine with trained speech recognition models.

IBM ViaVoice SDK supports speech model training. This SDK provides programmers with the necessary tools to develop applications that incorporate speech. It includes a robust set of application programming interfaces (APIs) that allows an application to access speech resources. It contains several utilities that enable developers to define and manage speech within an application. The heart of a speech recognition system is known as the speech recognition engine. The speech recognition engine recognizes speech input and translates it into text that an application understands. Applications can access the speech recognition engine through a speech recognition API. For ViaVoice, this API is known as the Speech Manager API, or SMAPI, for short. In this experiment, we use the SMAPI function `SmWordCorrection` to train the speech model. `SmWordCorrection` corrects a misrecognized word. This function notifies the speech recognition engine that an incorrectly recognized word or sequence of words was corrected by the user. The user corrects the word by providing the correct spelling. This call specifies whether the pronunciation for this word is to be added to the user's personal pronunciation pool and whether the corrected spelling is to be added to the user's personal text vocabulary extension. The speech recognition engine uses this information to assist future recognition.

With this method of training speech model within ViaVoice engine, we performed this experiment as follows. First, we selected 10 video clips in the video collection as our training set. The clips were preprocessed to produce segmented audio clips. Then, these audio clips were fed to ViaVoice engine. After dictation, we used the correctly dictated words to train the ViaVoice engine. Then, we repeated the procedures of the baseline measurement on this trained speech model to get the accuracy measurement.

In order to increase the statistical significance of this experiment, we repeated it using another 20 video clips.

5.1.3 Slow-Down Measurement

We noted that fast speed of speech might decrease the accuracy of a speech recognition engine. Hence, in this experiment, we slowed down the speech manually and fed it to the engine to see whether or not the accuracy could be improved.

First, we re-sampled the segmented wave files in the testing set by the ratio of 1.05, 1.1, 1.15, 1.2, 1.3, 1.4 and 1.6. If this ratio is greater than 1, then the re-sampled wave file is longer and slower than the original one. For example, if the ratio is 1.1, then a 1-minute long segment in the original wave file will turned into a 1.1-minute long segment in the re-sampled wave file. In other words, the audio is slowed down by the ratio of 1.1.

Then we repeated the procedures of baseline measurement to measure the corresponding recognition performance at different slow-down ratios.

5.1.4 Indoor News Measurement

As described in early chapters, noise could decrease the performance of speech recognition dramatically. In this experiment, we tried to measure the significance of performance decrease introduced by noise within speech.

After segmented the video clips into sentences, we selected those sentences with an indoor environment. Then we dictated these indoor sentences using the untrained speech model. We used the same procedure in baseline measurement to measure the dictation performance. Finally, we repeated this experiment using the trained speech model.

5.2 Experimental Results

There are ten training and testing video clips with 54162 Cantonese words. Within the testing set, there are 10109 Cantonese words being indoor news reporter speech. The overall results are shown in Table 7.1.

Experiment	Word Recognition Accuracy (Max. Performance)
Baseline	25.27%
Trained Model	25.87% (with 20 video clips trained)
Slow down Speech	25.67% (max. at ratio = 1.15)
Indoor Speech (untrained model)	35.22%
Indoor Speech (trained model)	36.31% (with 20 video clips trained)

Table 7.1 Overall Recognition Results using ViaVoice with TVB News

Table 7.1 shows that the highest accuracy is obtained when the recognition is performed indoor with the trained model. The trained model performed a little bit better than untrained one. It also shows that slowing down the audio channel with the ratio 1.15 can increase the performance.

The results of trained model measurement experiment are shown in Table 7.2. From these results, we can see that training with correct recognized word can increase the performance. However, the accuracy will be saturated after a number of training video clips. After we trained the engine using the first ten video clips, the accuracy increased from 25.27% to 25.82%. This means that we achieved a 0.55% improvement on the accuracy for the first ten video clips. But when we continued to train the model using another ten video clips, the recognition accuracy only increased from 25.82% to 25.87%. This means that we only got a 0.05% accuracy improvement for the second ten video clips.

Trained Video Number:	Untrained	10 video clips	20 video clips
Word Recognition Accuracy (W.R.A.)	25.27%	25.82%	25.87%

Table 7.2 Result of trained model with different number of training videos

Table 7.3 shows the results of slow-down measurement. We noted that the ratio between 1 and 1.2 could increase the performance. After ratio 1.2, the accuracy is degraded a lot.

Ratio	1	1.05	1.1	1.15	1.2	1.3	1.4	1.6
W.R.A.	25.27%	25.46%	25.63%	25.67%	25.82%	17.18%	12.34%	4.04%

Table 7.3 Result of using different slow down ratio

5.3 Conclusions

The speech recognition accuracies in these experiments are low. After these experiments, we forwarded our results to IBM China. In their reply, they expressed their opinions on the reason why the performance is not good. There are mainly four reasons:

- Language model mismatch. The text is not so familiar to the model. Some specific name is included in it.
- Voice channel mismatch. The models in ViaVoice are trained with microphone data, which is different from the data from broadcast
- The broadcast is very fast and some characters are not so clear because the stress varied much.
- The voice of video clips is too loud.

The first two reasons are the most critical ones.

The accuracy of indoor speech recognition is much higher than that of outdoor speech recognition. This means that noise plays an important role for accuracy decrease.

The results also showed that there was only about 1% word recognition improvement after training the model of ViaVoice engine using SmWordCorrection. The similar result is got when the speech is slow down by the ratio 1.15. This implied these two approaches could not achieve our objectives of this project. In the IBM reply, they also provided some comments on our future work.

First, we cannot do much acoustic model training with the SMAPI API. Since we can only use word correction to train the model (only new words are added and Language Model is adjusted a little bit), the recognition will be improved a lot only for these trained words. But because the news clips always have new words, such a training scheme will not help much to improve the overall accuracy.

Second, after trained a model with one speaker's voice, the model will only have good accuracies for that particular speaker and perform poorly for other speakers.

Third, one of the most critical causes for the low accuracy is the great difference between the news audio and the training speech for ViaVoice. Our speech is from MPEG Video. It is not known whether or not the feature extracted from MPEG video will have significant difference from that of ViaVoice mode. Second, the reading style is also much different. The ViaVoice models are trained with microphone data. These speeches have different styles from speeches of new reporters.

Forth, if we have a lot of transcriptions of broadcast news, the language model can be adapted accordingly. IBM will have a tool to adapt our existing acoustic model to a target new channel/source, and a tool to adapt our language model to a new domain (such as broadcast news). These should be the essential way to solve our problems. But such tools are still under development.

Hence, in order to have correct transcripts of news video clips, we have to change our approach. The adaptation of existing language models will not help us much.

We have also tried the Mandarin speech recognition with Mandarin video news clips and the estimated accuracy of baseline is about 70%. There is a large difference of baseline accuracy between Mandarin (~70%) and Cantonese (~36%) speech recognition. We think the Cantonese speech has various oral words, which do not appear in the models training stage. Thus, the acoustic and language models have certain mismatch with the incoming speech from news video.

Chapter 6 : Speech Information Processor

On the basis of the visual training tool, we developed the Speech Information Processor (SIP for short). This is an integrated speech-processing tool. SIP is both a flexible speech recognition editor and an audio information retrieval unit. This software component has the following main features: media playback, real-time dictation, firm word index, word time index, dynamic recognition text editing, audio scene change detection, audio segments classification, and gender classification.

6.1 System Chart

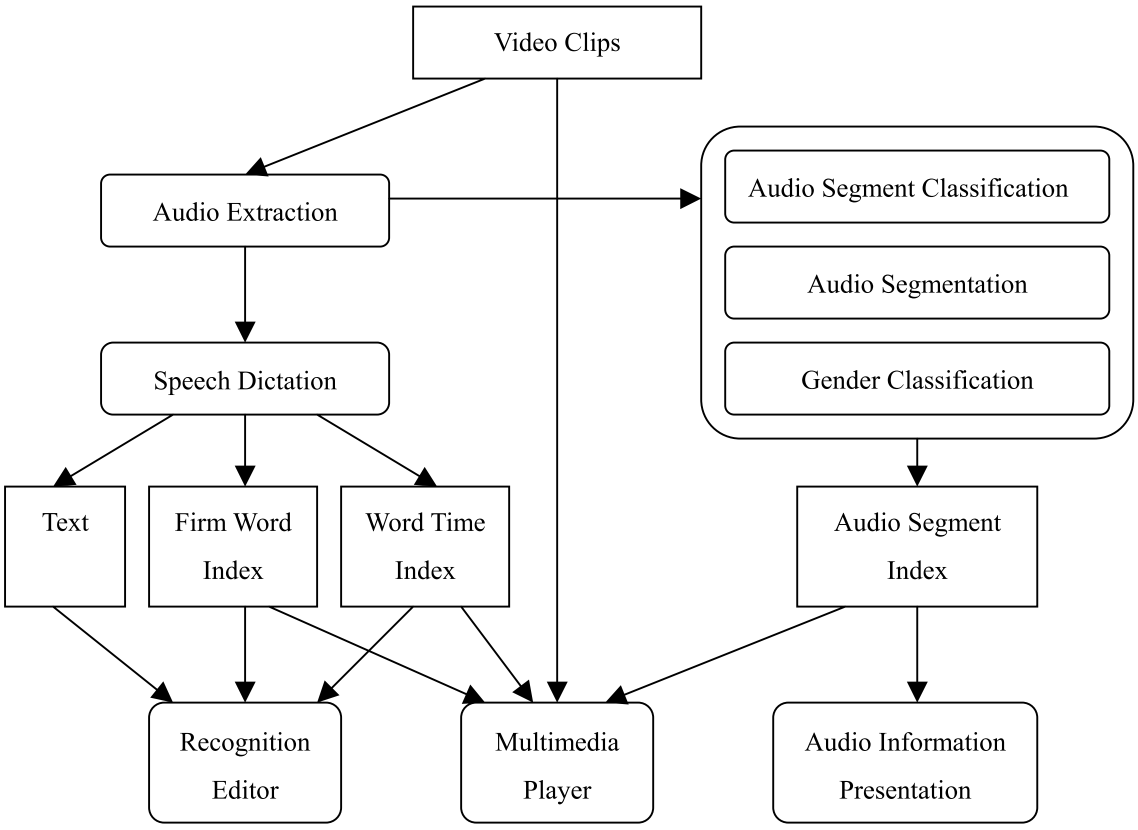


Figure 6.1 Overall System Chart

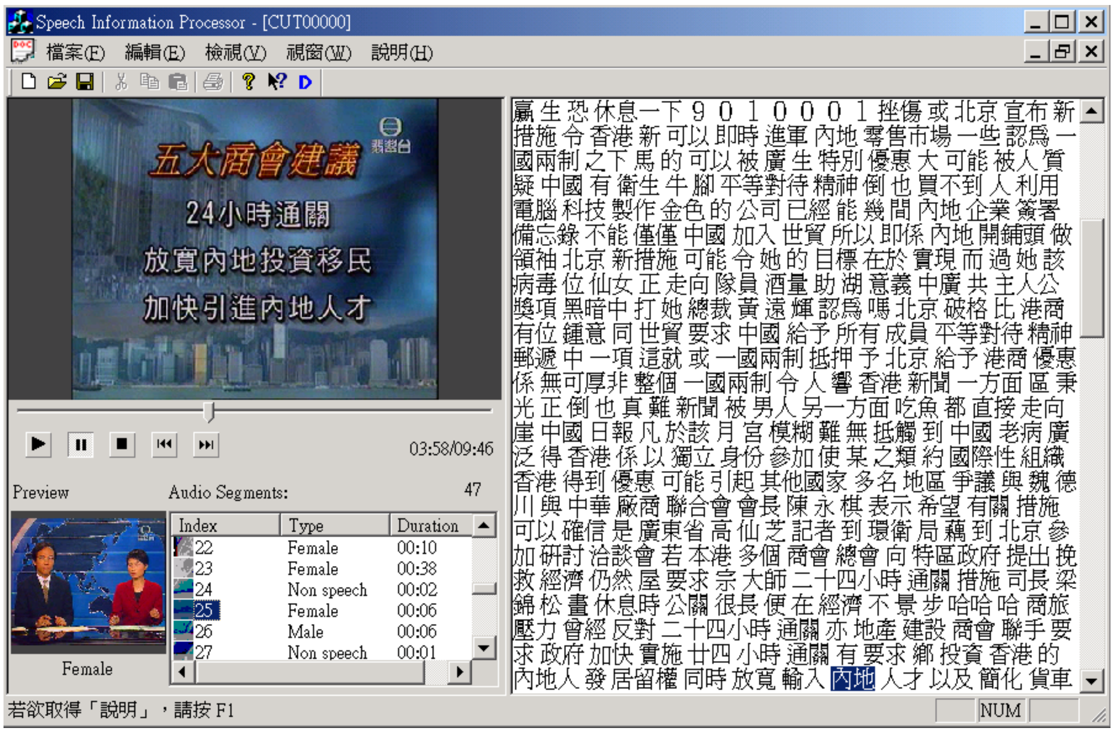
The inputs to the system are multimedia files, mainly new video clips. The audio channel of the input clip will be extracted and stored in a wave file (mono, 22KHz).

Then the wave file is passed to a speech recognition engine to produce the text of the speech, firm word index, and word time index. The recognition editor will display the text. The word time index is used to highlight the corresponding word during video playback. When the user edits the dictation text, the recognition editor will update the firm word index dynamically.

The audio channel is processed further, including audio segment classification, audio segmentation, and gender classification. The output of these audio processing will be stored in an audio segment index. The audio information presentation panel will use this index to present the audio information to the user in a user-friendly interface.

The system also includes a media player. The produced firm word index, word time index, and audio segment index will be forwarded to this player. The player is able to understand these indices, and use them to realize many speech-related features.

6.2 Main System Features



Index	Type	Duration
22	Female	00:10
23	Female	00:38
24	Non speech	00:02
25	Female	00:06
26	Male	00:06
27	Non speech	00:01

贏 生 恐 休息一下 9 0 1 0 0 0 1 挫傷 或 北京 宣布 新
措施 令 香港 新 可以 即時 進軍 內地 零售市場 一些 認爲 一
國兩制 之下 馬 的 可以 被 廣 生 特別 優惠 大 可能 被人 質
疑 中國 有 衛生 牛 腳 平等對待 精神 倒 也 買不到 人 利用
電腦 科技 製作 金色 的 公司 已經 能 幾間 內地 企業 簽署
備忘錄 不能 僅僅 中國 加入 世貿 所以 即係 內地 開鋪頭 做
領袖 北京 新措施 可能 令 她 的 目標 在於 實現 而 過 她 該
病毒 位 仙女 正 走向 隊員 酒量 助 湖 意義 中廣 共 主人公
獎項 黑暗中 打 她 總裁 黃 遠 輝 認爲 嗎 北京 破格 比 港商
有位 鍾意 同 世貿 要求 中國 給予 所有 成員 平等對待 精神
郵遞 中 一項 這就 或 一國兩制 抵押 予 北京 給予 港商 優惠
係 無可厚非 整個 一國兩制 令 人 響 香港 新聞 一方面 區 秉
光 正 倒 也 真 難 新聞 被 男人 另一方面 吃魚 都 直接 走向
崖 中國 日報 凡 於該 月 宮 模糊 難 無 抵觸 到 中國 老病 廣
泛 得 香港 係 以 獨立 身份 參加 使 某 之類 約 國際性 組織
香港 得到 優惠 可能 引起 其他國家 多名 地區 爭議 與 魏 德
川 與 中華 廠商 聯合會 會長 陳 永 棋 表示 希望 有關 措施
可以 確信 是 廣東省 高 仙 芝 記者 到 環衛 局 藕 到 北京 參
加 研討 洽談會 若 本港 多個 商會 總會 向 特區政府 提出 挽
救 經濟 仍然 屋 要求 宗 大師 二十四小時 通關 措施 司長 梁
錦 松 畫 休息時 公關 很長 便 在 經濟 不 景 步 哈哈 哈 商旅
壓力 曾經 反對 二十四小時 通關 亦 地產 建設 商會 聯手 要
求 政府 加快 實施 廿四 小時 通關 有 要求 鄉 投資 香港 的
內地人 發 居留權 同時 放寬 輸入 內地 人才 以及 簡化 貨車

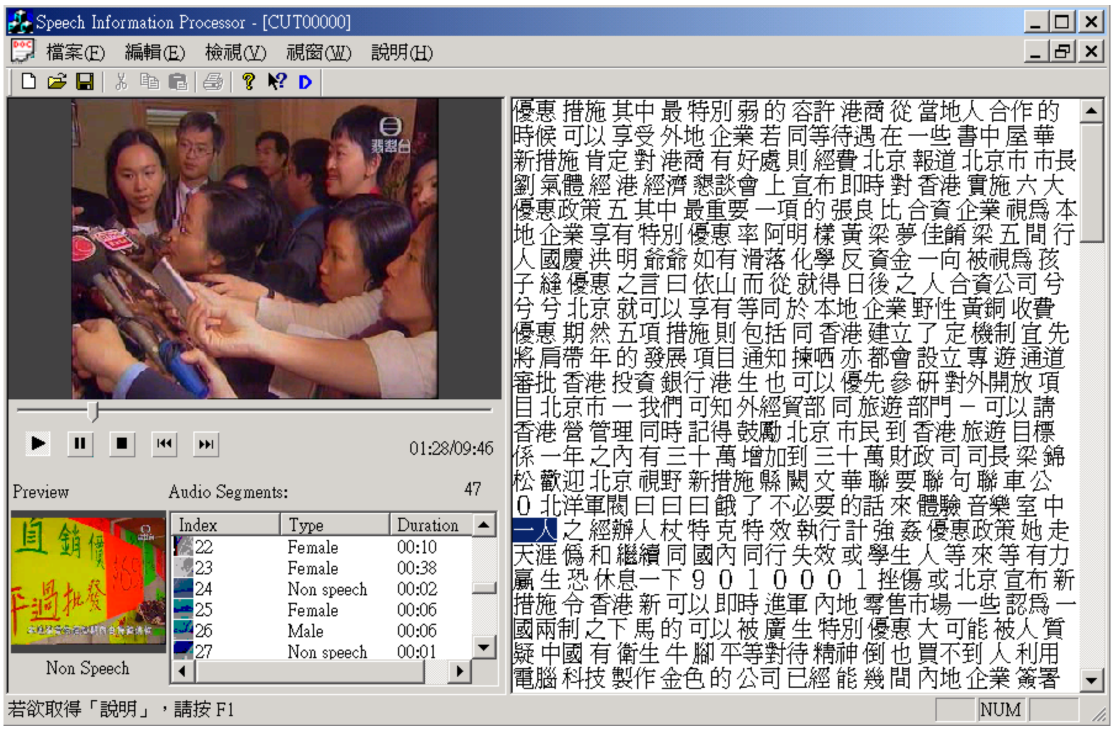
Figure 6.2 Main Interface of SIP

As shown in figure 6.2, the recognition editor is on right hand side. The sub-window on left hand side is divided into two parts. The upper half part is the media player, while the lower half part is the audio information representation panel. The panel consists of a segment list and a segment preview window.

6.2.1 Media Player

The media player supports audio segmentation. There is a previous button ⏮ and a next button ⏭ on the player interface.

After the user presses the previous button, the video playback will jump back to the starting position of the previous segment if there exists some segment that precedes the current segment. If the next button is pressed, the playback will jump forward to the starting position of the next segment. These segments are divided according audio scene change. That means a single audio scene change will divide the clip into two different segments, while two consecutive audio scene changes will divide the clip into three different segments. The detail of audio segmentation will be discussed in the next chapter.



Speech Information Processor - [CUT00000]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
01:28/09:46
Preview
Non Speech

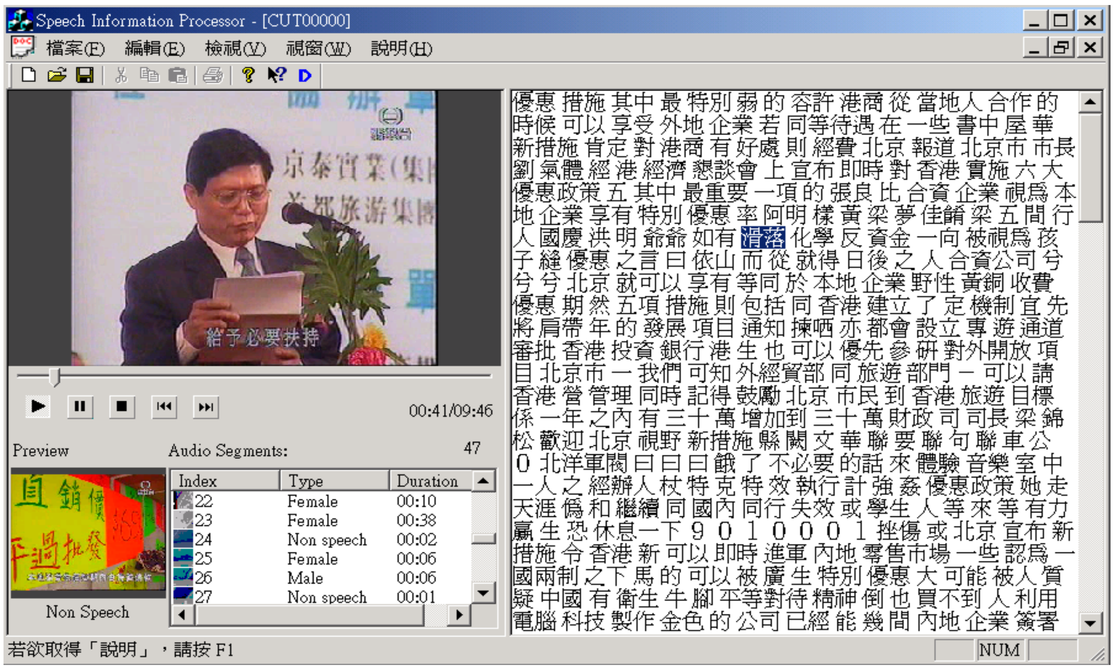
Audio Segments: 47

Index	Type	Duration
22	Female	00:10
23	Female	00:38
24	Non speech	00:02
25	Female	00:06
26	Male	00:06
27	Non speech	00:01

優惠 措施 其中 最 特別 弱 的 容許 港商 從 當地人 合作的
時候 可以 享受 外地 企業 若 同等待遇 在 一些 書中 屋 華
新措施 肯定 對 港商 有 好處 則 經費 北京 報道 北京市 市長
劉 氣體 經 港 經濟 懇談會 上 宣布 即時 對 香港 實施 六 大
優惠政策 五 其中 最重要 一項 的 張良 比 合資 企業 視爲 本
地 企業 享有 特別 優惠 率 阿明 樣 黃 梁 夢 佳餚 梁 五 間 行
人 國慶 洪 明 爺爺 如有 滑落 化學 反 資金 一向 被視爲 孩
子 縫 優惠 之言 曰 依山 而 從 就得 日後 之 人 合資公司 兮
兮 兮 北京 就可以 享有 等同 於 本地 企業 野性 黃銅 收費
優惠 期 然 五項 措施 則 包括 同 香港 建立 了 定 機制 宜 先
將 肩帶 年 的 發展 項目 通知 揀晒 亦 都會 設立 專 遊 通道
審批 香港 投資 銀行 港 生 也 可以 優先 參 研 對外開放 項
目 北京市 一 我們 可知 外經貿部 同 旅遊 部門 一 可以 請
香港 營 管理 同時 記得 鼓勵 北京 市民 到 香港 旅遊 目標
係 一年 之內 有 三十 萬 增加到 三十 萬 財政 司 司長 梁 錦
松 歡迎 北京 視野 新措施 縣 閥 文 華 聯 要 聯 句 聯 車 公
０ 北洋軍閥 曰 曰 曰 餓 了 不必要 的話 來 體驗 音樂 室 中
一人 之 經辦 人 杖 特 克 特 效 執行 計 強 姦 優惠政策 她 走
天涯 僞 和 繼續 同 國內 同行 失效 或 學生 人 等 來 等 有力
贏 生 恐 休息一下 ９ ０ １ ０ ０ ０ １ 挫傷 或 北京 宣布 新
措施 令 香港 新 可以 即時 進軍 內地 零售市場 一些 認爲 一
國兩制 之下 馬 的 可以 被 廣 生 特別 優惠 大 可能 被人 質
疑 中國 有 衛生 牛 腳 平等對待 精神 倒 也 買不到 人 利用
電腦 科技 製作 金色 的 公司 已經 能 幾 間 內地 企業 簽署

若欲取得「說明」，請按 F1 NUM

(a) Current Segment


Speech Information Processor - [CUT00000]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
Preview
Non Speech
NUM

00:41/09:46

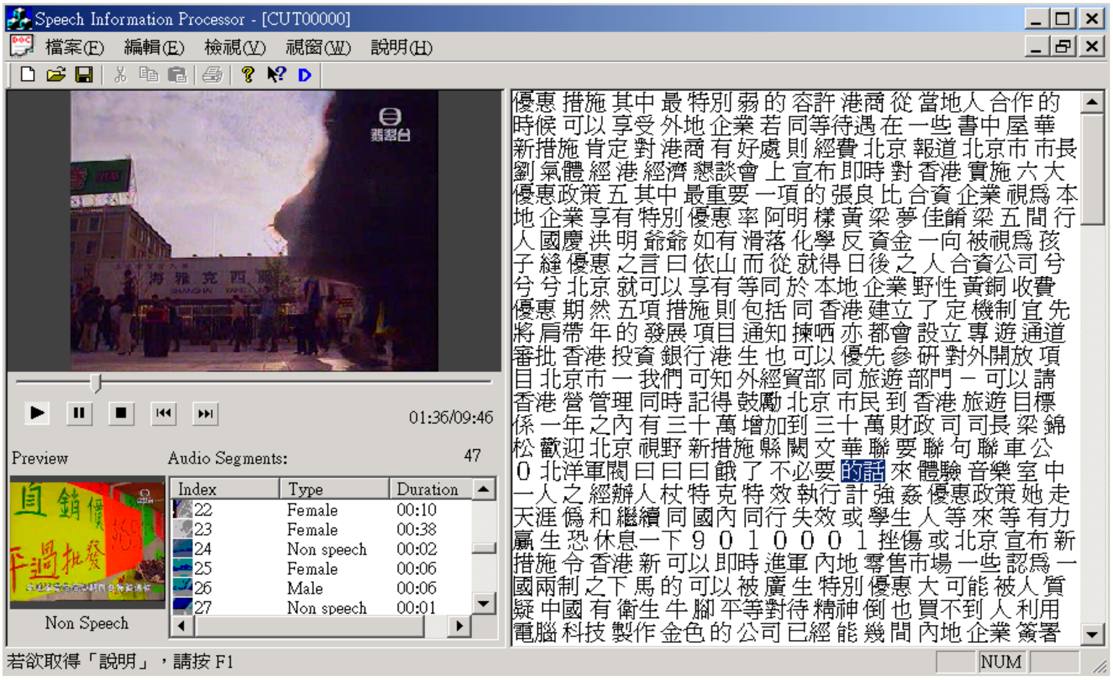
Audio Segments: 47

Index	Type	Duration
22	Female	00:10
23	Female	00:38
24	Non speech	00:02
25	Female	00:06
26	Male	00:06
27	Non speech	00:01

優惠 措施 其中 最 特別 弱 的 容許 港商 從 當地人 合作的
時候 可以 享受 外地 企業 若 同等待遇 在 一些 書中 屋 華
新措施 肯定 對 港商 有 好處 則 經費 北京 報道 北京市 市長
劉 氣體 經 港 經濟 懇談會 上 宣布 即時 對 香港 實施 六大
優惠政策 五 其中 最重要 一項 的 張良 比 合資 企業 視爲 本
地 企業 享有 特別 優惠 率 阿明 樣 黃 梁 夢 佳餚 梁 五 間 行
人 國慶 洪 明 爺爺 如有 滑落 化學 反 資金 一向 被視爲 孩
子 縫 優惠 之言 曰 依山 而 從 就得 日後 之 人 合資公司 兮
兮 兮 北京 就可以 享有 等同 於 本地 企業 野性 黃銅 收費
優惠 期 然 五項 措施 則 包括 同 香港 建立 了 定 機制 宜 先
將 肩帶 年 的 發展 項目 通知 揀哂 亦 都會 設立 專 遊 通道
審批 香港 投資 銀行 港 生 也 可以 優先 參 研 對外開放 項
目 北京市 一 我們 可知 外經貿部 同 旅遊 部門 一 可以 請
香港 營 管理 同時 記得 鼓勵 北京 市民 到 香港 旅遊 目標
係 一年 之內 有 三十 萬 增加到 三十 萬 財政 司 司長 梁 錦
松 歡迎 北京 視野 新措施 縣 鬩 文 華 聯 要 聯 句 聯 車 公
0 北洋軍閥 曰 曰 曰 餓 了 不必要 的話 來 體驗 音樂 室 中
一人 之 經辦人 杖 特 克 特 效 執行 計 強 姦 優惠政策 她 走
天涯 僞 和 繼續 同 國內 同行 失效 或 學生 人 等 來 等 有力
贏 生 恐 休息 一下 9 0 1 0 0 0 1 挫傷 或 北京 宣布 新
措施 令 香港 新 可以 即時 進軍 內地 零售市場 一些 認爲 一
國兩制 之下 馬 的 可以 被 廣 生 特別 優惠 大 可能 被人 質
疑 中國 有 衛生 牛 腳 平等對待 精神 倒 也 買不到 人 利用
電腦 科技 製作 金色 的 公司 已經 能 幾 間 內地 企業 簽署

若欲取得「說明」，請按 F1

(b) Previous Segment


Speech Information Processor - [CUT00000]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
Preview
Non Speech
NUM

01:36/09:46

Audio Segments: 47

Index	Type	Duration
22	Female	00:10
23	Female	00:38
24	Non speech	00:02
25	Female	00:06
26	Male	00:06
27	Non speech	00:01

優惠 措施 其中 最 特別 弱 的 容許 港商 從 當地人 合作的
時候 可以 享受 外地 企業 若 同等待遇 在 一些 書中 屋 華
新措施 肯定 對 港商 有 好處 則 經費 北京 報道 北京市 市長
劉 氣體 經 港 經濟 懇談會 上 宣布 即時 對 香港 實施 六大
優惠政策 五 其中 最重要 一項 的 張良 比 合資 企業 視爲 本
地 企業 享有 特別 優惠 率 阿明 樣 黃 梁 夢 佳餚 梁 五 間 行
人 國慶 洪 明 爺爺 如有 滑落 化學 反 資金 一向 被視爲 孩
子 縫 優惠 之言 曰 依山 而 從 就得 日後 之 人 合資公司 兮
兮 兮 北京 就可以 享有 等同 於 本地 企業 野性 黃銅 收費
優惠 期 然 五項 措施 則 包括 同 香港 建立 了 定 機制 宜 先
將 肩帶 年 的 發展 項目 通知 揀哂 亦 都會 設立 專 遊 通道
審批 香港 投資 銀行 港 生 也 可以 優先 參 研 對外開放 項
目 北京市 一 我們 可知 外經貿部 同 旅遊 部門 一 可以 請
香港 營 管理 同時 記得 鼓勵 北京 市民 到 香港 旅遊 目標
係 一年 之內 有 三十 萬 增加到 三十 萬 財政 司 司長 梁 錦
松 歡迎 北京 視野 新措施 縣 鬩 文 華 聯 要 聯 句 聯 車 公
0 北洋軍閥 曰 曰 曰 餓 了 不必要 的話 來 體驗 音樂 室 中
一人 之 經辦人 杖 特 克 特 效 執行 計 強 姦 優惠政策 她 走
天涯 僞 和 繼續 同 國內 同行 失效 或 學生 人 等 來 等 有力
贏 生 恐 休息 一下 9 0 1 0 0 0 1 挫傷 或 北京 宣布 新
措施 令 香港 新 可以 即時 進軍 內地 零售市場 一些 認爲 一
國兩制 之下 馬 的 可以 被 廣 生 特別 優惠 大 可能 被人 質
疑 中國 有 衛生 牛 腳 平等對待 精神 倒 也 買不到 人 利用
電腦 科技 製作 金色 的 公司 已經 能 幾 間 內地 企業 簽署

若欲取得「說明」，請按 F1

(c) Next segment

Figure 6.3 Video playback on previous and next segments: (a) current segment; (b) previous segment (after the previous button being pressed); (c) next segment (after the next button being pressed)

Figure 6.3 (a) – (c) gives an example of the functionalities of the previous and the next button. After the previous button is pressed, the video playback jumps from the state in (a) into the state in (b). The difference between the state in (a) and that in (c) shows the effect of the next button.

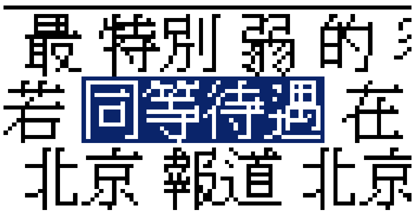
Word time index is another feature of the media player. As we can see from 6.3, the output text is not organized directly on the basis of characters. It is divided into many words. The boundaries of the words are divided by spaces. For example, the highlighted word "滑落" is treated as a single unit, not two separate characters. The word time index records timing information of all the words. In the index, each entry is associated with a word, and each word is associated with an entry. Thus, there is one-to-one correspondence between entries in the word time index and words of dictation result. Each entry in the index records the time and position of the corresponding word. The player is able to parse the index, so that it can highlight the corresponding word during playback. This feature is also shown in figure 6.3. As in figure 6.3 (b), the person is currently saying the word "滑落". So that we can see, in the recognition text edition, the word "滑落" is highlighted.

6.2.2 Recognition Text Editor

As described in the previous chapter, the performance of IBM ViaVoice engine is not that good for new clips. And there is little to do to further improve its accuracy significantly. But one of the objectives of our project is to provide complete and correct transcripts for the video clips in the digital video library. Hence, we must change our approach. The solution we used for this problem is to let ViaVoice produce the text first and then manually correct the mis-recognized words. The main functionality of the recognition text editor is to realize this manual correction. In order to facilitate the work of correction, this editor provides many special features.

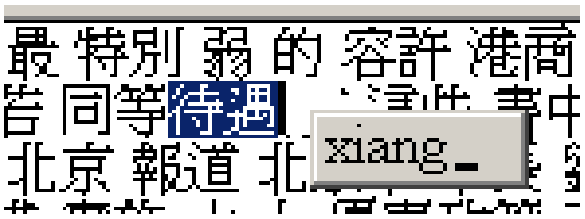
The editor supports word index. It is able to understand that the recognition text consists of many words. Each word is separated by the spaces around them. If the user double clicks on a word in the editor, the corresponding audio segment will be played by the media player. This feature is useful to the user. Because when the user realizes that some word is mis-recognized, he/she needs to correct it. But it is difficult to tell where the word is in the original clip. So in order to listen to the speech segment corresponding to the mis-recognized word, the user has to go through the whole clip to locate it. But with the word index functionality, the user could simply double click the mis-recognized word, and the corresponding clip segment will be played again.

The editor also supports dynamic index alignment. Since the indexed words are in the editor, so it is possible that the user changes the structures of words during editing. For example, the user may add characters to a word, delete characters from a word, insert new words, or delete a whole word. Under such situation, the index must be updated accordingly, so that the system can associate each entry in the index with the correct time and position of the word.



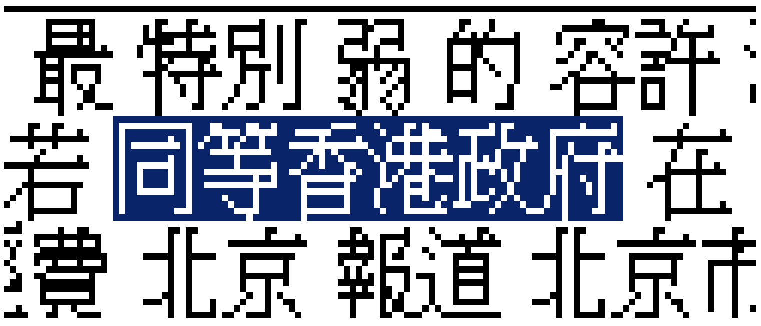
最 特別 弱 的
若 同等待遇 在
北京 報道 北京

(a) Before editing



最 特別 弱 的 容許 港商
同等待遇
xiang
北京 報道

(b) Being edited



最 特別 弱 的 容許
若 同等香港政府 在
北京 報道 北京

(c) After editing

Figure 6.4 Word alignments during editing

As an example, figure 6.4 (a) shows the word “同等待遇” before it is being edited. So double clicking on any part of this word, the entire four characters consisting the word will be highlighted. This shows that the system treats the four characters as a single word. Figure 6.4 (b) shows the word “同等待遇”being edited. The user edited this word using Microsoft Pinyin Input Method. Figure 6.4 (c) shows the word after being edited, the two characters “待遇”now change into “香港政府”. Now double clicking on the any part of “同等香港政府” will highlight the whole six characters. This shows that the system has aligned the index of the original four-character word into the new six-character word.

6.2.3 Audio Information Panel

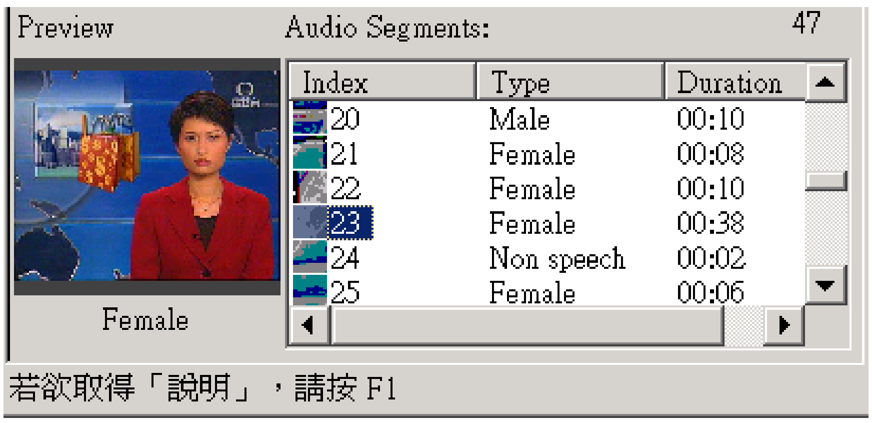
Since SIP is a speech information processor, we provide not only the dictation and editing functionalities in it, but also other useful audio information retrieval features, including audio scene change detection, audio segment classification, and gender classification. The underline algorithms and implementation of audio information retrieval is rather complicated, so we preserve the next chapter for it. We only present the interface features here.

After audio scene change detection, the detected change points divide the whole clip into many segments. Then, we will apply audio segment classification and gender classification on these segments to classify them into three categories, including male voice, female voice, and non-speech. The audio information panel is reserved to present the result of this process to the user.

Figure 6.5 (a) – (c) shows a screen shot of audio information panel when processing a news video clip. On the right half of the panel, there is a segment list displaying information of all segments of the clip. The line above the list shows the total number of segments. In figure 6.5, there are totally 47 segments of the clip. The list has three columns. The first column includes the segment indices and icons. For each segment, its icon is the frame of the original video clip at the starting position of the segment. The icon may be too small to see clearly. A larger preview of the first frame is on left hand side. When a segment is selected, its first frame will be shown in the preview window.

The second column is the type of segments. The type "Male" means that the segment contains only male speech, while the type "Female" means that the segment contains only female speech. The other type "Non speech" means that the segment contains other sound rather than speech. This may be noise, music, or silence. The type of the selected segment is also shown below the preview window.

The third column is duration. This column shows the duration of each segment. For example, in figure 6.5 (b), the duration of the selected is 13 seconds long.



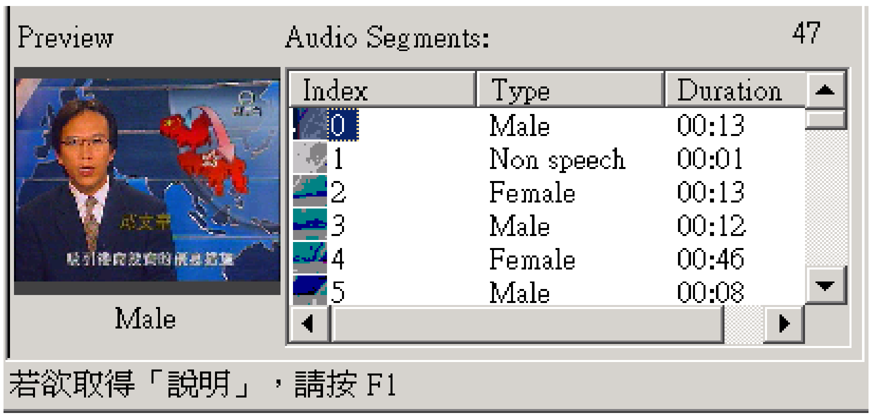
Preview Audio Segments: 47

Index	Type	Duration
20	Male	00:10
21	Female	00:08
22	Female	00:10
23	Female	00:38
24	Non speech	00:02
25	Female	00:06

Female

若欲取得「說明」，請按 F1

(a) Female segment selected



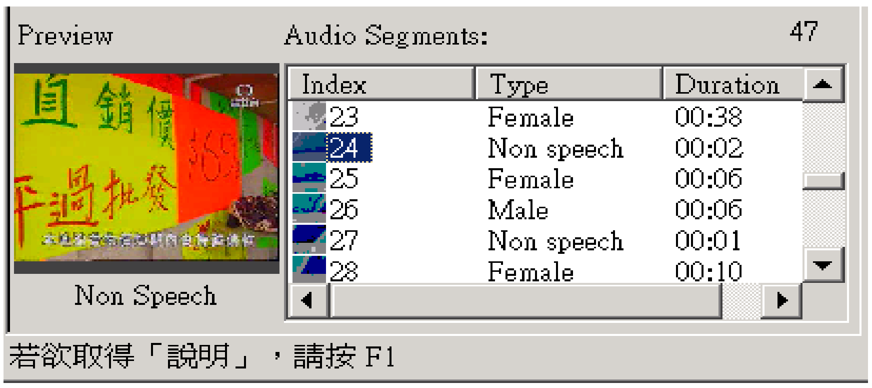
Preview Audio Segments: 47

Index	Type	Duration
0	Male	00:13
1	Non speech	00:01
2	Female	00:13
3	Male	00:12
4	Female	00:46
5	Male	00:08

Male

若欲取得「說明」，請按 F1

(b) Male segment selected



Preview Audio Segments: 47

Index	Type	Duration
23	Female	00:38
24	Non speech	00:02
25	Female	00:06
26	Male	00:06
27	Non speech	00:01
28	Female	00:10

Non Speech

若欲取得「說明」，請按 F1

(c) Non-speech segment selected

Figure 6.5 Audio information presentation panel: (a) female segment selected; (b) male segment selected; (c) non-speech segment selected

6.2.4 Multi-lingual Support

The audio information retrieval and text editing is not related to particular languages. But the dictation part is sensitive to different languages. Different languages require different speech models. In SIP, IBM ViaVoice serves as the speech recognition engine. Currently we have the Cantonese and Mandarin versions of ViaVoice. And furthermore, there is only one single API for different language versions of ViaVoice. Therefore, SIP has multi-lingual capacity. Now SIP support Cantonese and Mandarin, separately, but not mixed language of the two. SIP will support English, as soon as the English version of IBM ViaVoice is available.

In the remaining sections of this chapter, we discuss the implementation details of these features.

6.3 Timing Information Retrieval

In order to build the word time index, timing information of every word recognized must be retrieved. The timing information is supplied by the speech recognition engine, which is ViaVoice in SIP. SMAPI is the application programming interface for applications to communicate with ViaVoice engine.

6.3.1 SMAPI

For IBM ViaVoice, all speech engine functions can be accessed from the Speech Manager API, SMAPI. API calls from the application to the engine can either by synchronous or asynchronous. For synchronous calls, SMAPI blocks locally, waits for the engine to reply, and returns a message through a function parameter. For asynchronous calls, SMAPI returns control to the application, the engine sends the reply message via a platform dependent transport mechanism, and notifies the application. On Windows this notification is through a posted Windows message.

Calls from the speech engine to the application are only asynchronous, so the application needs to establish a message-handling procedure for these. Many events from the engine are delivered and handled through the same mechanisms (case/callbacks) as asynchronous replies to SMAPI function calls. One example of such an event from the engine would be recognized words. These are inherently asynchronous, because they depend on when the user speaks. The `SmGet` data access functions are used to retrieve data from reply messages.

Some operations take several seconds to complete (such as the first application to load and initialize the speech engine). Synchronous calls can lock up an application. We can solve this problem by using asynchronous calls. SMAPI calls fall into three phases: initialization, recognition, and termination. In the initialization phase, the application sets items that determine how recognition will be performed, such as the user ID, the enrollment ID, and the speech domain. During the recognition phase, the application interacts with the speech engine to set vocabularies and grammars from which recognition takes place, processes recognized speech, controls the microphone, and performs other tasks related to speech recognition itself. During the termination phase, the application makes sure that its session with the speech engine ends properly.

The speech engine allows multiple concurrent connections to it, even from within the same application. Two concurrent sessions from separate applications are referred to as shared sessions, and two concurrent sessions within the same application are referred to as parallel sessions. The API is not guaranteed to be thread-safe, so any individual session should be restricted to a single thread. All SMAPI function calls for a session should be made from the same thread.

Although SMAPI is a C language interface, it is written in an object-oriented style. For example, the reply structure `SmReply` should be thought of as an object with `SmGet` access methods. Keeping this in mind will help us understand the API.

The speech recognition engine is a separate process from the application. Communication between a SMAPI speech application and the speech engine is in the form of messages passed between the two processes via a transport protocol. This transport protocol is transparent to the application.

As mentioned early, function calls to the speech engine can be processed synchronously or asynchronously. When a synchronous call is made, SMAPI sends a request message to the engine, and waits until the engine has processed the request and sent a reply message. When an asynchronous call is made SMAPI sends the request message to the engine and returns immediately to the caller without waiting for the engine to process the request and send a reply. The engine sends the reply after processing the request and the reply message is later dispatched explicitly by the application or through a registered callback.

The engine processes requests the same way, whether they are made synchronously or asynchronously. Also, the engine processes requests in the order that they are received.

The application can use callbacks to process messages sent by the speech engine after a speech session has been established. To use this method, the application must register callbacks prior to issuing the function call. The previously registered callbacks are automatically dispatched when a message is sent by the engine. Multiple callbacks are permitted for a given message. These callbacks are processed in the order that they were registered.

There are special speech API functions to add and remove functions from callback lists. However, such lists must be defined carefully. Because the list of callbacks is processed sequentially, the procedure sequence must guarantee proper processing so that changes of state by early routines do not unintentionally disrupt processing by later routines in the list. The functions include error checking for some conditions that could cause problems in callback list processing.

Callback procedures are defined as returning a particular data type and having the following parameters:

```
SmHandler Callback_Handler ( SM_MSG reply, void *client_data, void *call_data );

reply
    - The message to be handled

client_data
    - Data supplied by the speech-aware application when registering the callback

call_data
    - Dynamic data to be supplied
```

Callbacks are registered by using the `SmAddCallback` function, which specifies the type of callback, the name of the callback routine, and any user data. The type of callback is sometimes referred to as the attribute name for the callback. User data is application specific information that is passed along to the callback procedure when it is called. One procedure can be registered per `SmAddCallback` function.

6.3.2 SmGetWordTimes

Most of speech related tasks are programmed in the source file *speech.cpp*. In this file, the function `RecoTextCB` is registered as a call back function. It is called whenever the engine has recognized something from the text (dictation) vocabulary.

There are two types of words that the engine might generate, firm word and infirm word. Firm words are those words which the engine has completed decoding. Once the engine has declared a word "firm," it is finished processing that word. The application should display all firm words to the user as they are recognized by the engine. In firm words, on the other had, are those that the engine is still decoding. One or more infirm words might eventually become a single firm word, and a single infirm word might result in one or more firm words. Hence, inside the callback function `RecoTextCB`, we must determine the type of words that the engine has just recognized. And then we use the function `SmGetWordTimes` to retrieve the start and end times of every firm word the engine has recognized.

`SmGetWordTimes` retrieves the most likely start and end times for recognized words from the reply structure. During recognition, the speech recognition engine computes the most likely start and end times for each recognized word. These times are used for playback and for build the time index.

The recognition engine keeps track of word start and end times in frame time offsets since the beginning of the utterance. The frame time is converted to wall time before returning the times to the application. On Windows the engine uses the Windows function `GetTickCount` to create the timestamp for the beginning of the utterance.

In order to determine a word's start or end time offset from the beginning of the utterance, the application must subtract the word start or end time from the time returned in the SM_SPEECH_START engine state message. Therefore, we registered another callback function `EngineStateCB` to record the start time of the utterance.

For each firm word, there is a `FirmWord` entry associated with it. This structure is defined as

```
typedef struct firm_word {
    unsigned long stime, etime;
    int start, end;
} FirmWord;
```

The member `stime` and `etime` are the start time and end time of the firm word, respectively, while `start` and `end` record the position of the firm word.

After the timing information of a firm word is successfully retrieved, a `FirmWord` entry will be created for it. And this entry will be added into an array to form the firm word index.

6.4 Dynamic Index Alignment

When the user edits the recognition text, the original word structure might not be preserved. Therefore, we need to update the word index accordingly. First, we need to capture the event that the user edits the text.

Applications written for Microsoft Windows are "message driven." In response to events such as mouse clicks, keystrokes, window movements, and so on, Windows sends messages to the proper window. Windows applications do not make explicit function calls (such as C run-time library calls) to obtain input. Instead, they wait for the system to pass input to them. The system passes all input for an application to the various windows in the application. Each window has a function, called a *window procedure*, which the system calls whenever it has input for the window. The window procedure processes the input and returns control to the system.

In MFC, applications process Windows messages like any other application for Windows. But the framework also provides some enhancements that make processing messages easier, more maintainable, and better encapsulated. In MFC, a dedicated *handler* function processes each separate message. Message-handler functions are member functions of a class.

In our case, we want to capture the event that the user is editing the text. That is the user has entered something into the editor window. Therefore, we added a message handler `OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)`. In this function, the changed text is analyzed again to see is there any change to some word in the index. If there is some change, the index will be modified according. Since this must be done every time the user has input something, the performance issue becomes important here. In our implementation, we deployed binary search on the index, and the resulting performance is satisfiable for this application.

6.5 The Player Form

The player form is the left half window of the main interface of SIP. It consists of the player window and the audio information presentation panel. One of its responsibilities is to lay out the video window and control panel. There is a CVideo object embedded. The user inputs, such as play and stop and so on, are passing to it.

There is a `CListCtrl` object embedded in this form. This object is the segment list. The preview window is also in this form. The form is responsible to manage the behavior of the left window on the main interface. When the selection of the segment list has changed, the list forwards this message to the form. And then form tells the preview window to update the frame to the first frame of the newly selected segment. When the user clicks an item in the segment list, the double click message will be forwarded to the form. And the form will then tell the video player to play the double clicked segment of the clip.

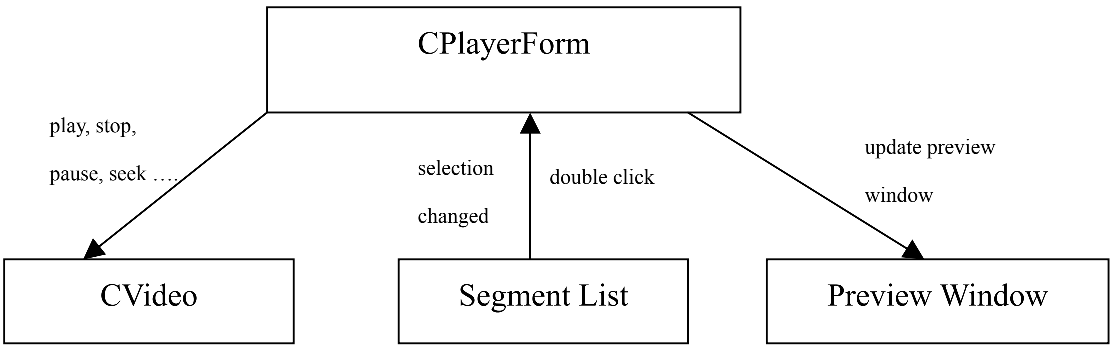


Figure 6.6 Object diagram of the player form

In next chapter, we will discuss the underline theory and algorithms for audio informaton retrieval.

Chapter 7 : Audio Information Retrieval

7.1 Flow Diagram

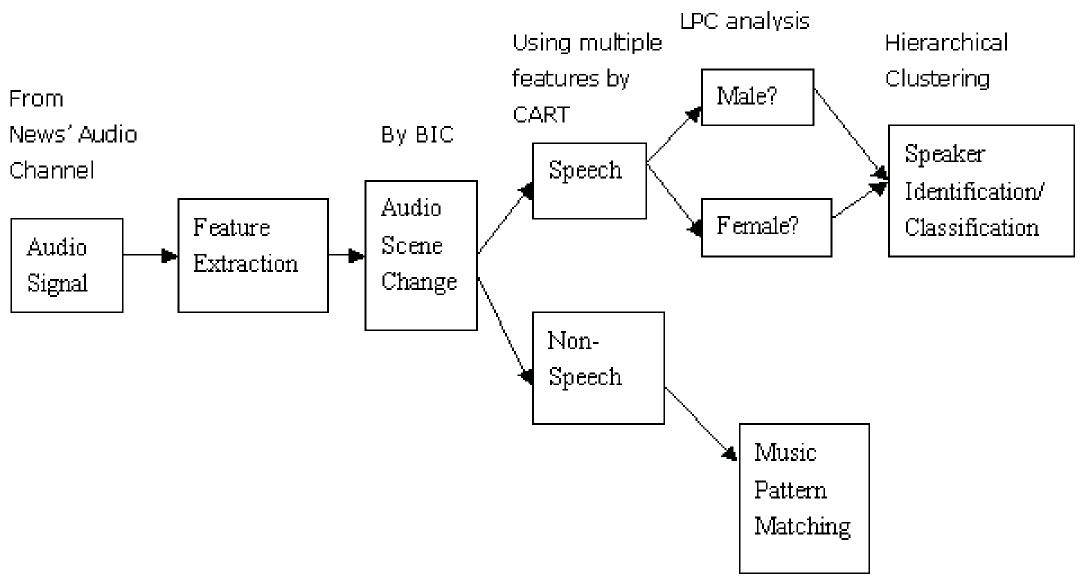


Figure 7.1 Flow diagram of audio information retrieval system

Here is the flow diagram of our audio information retrieval system. We take sample audio data as input and produce the clusters of speech segments as output. The major steps of processing raw audio signals are:

- Feature Extraction: first we extract multiple features from input audio stream
- Audio Scene Change Detection: we use the BIC audio scene change detection algorithm to find the change points of the audio clips. It is done by maximizing the likelihood function. The basic idea is to penalize the system by model complexity.
- Classifying speech/non-speech parts: we use multiple features from input audio to classify speech from non-speech signals. And then we use CART (Classification and Regression Tree) to make a decision.
- Gender Classification: it is done by LPC (Linear Predictive Coding) analysis. We have studies two algorithms and choose the better one from testing results.

- Speaker Identification/Classification: It is done by hierarchical clustering audio segments.

The details of steps we implemented will be introduced later in following chapters.

7.2 Audio Scene Change Detection and Clustering Via the Bayesian Information Criterion (BIC)

In our system we are trying to detect changes in speaker identity, environmental condition and channel condition, which is called the problem of *acoustic change detection*. The audio input stream is modeled as a Gaussian distribution in the cepstral space. We apply the Maximum Likelihood learning principle to detect turns of a Gaussian process, and the decision of a change is made by the Bayesian Information Criterion (BIC), which is a model selection criterion well known in statistics.

The basic idea of BIC is that the two nodes can only be merged to one only if the merging increases the BIC value. From our experiment it shows that it can successfully detect acoustic changes, and the clustering algorithm can generate pure clusters for each speaker.

7.2.1 Introduction

Automatic segmentation of an audio stream and automatic clustering of audio segments based on speaker identities, environmental changes and channel conditions are drawn much attention in speech recognition field of study. For instance, in TV news, the audio data contains clean speech, music segments, telephone speech, and noise, etc. The problem is that there are not explicit cues for the sudden changes in speaker identity, environment condition and channel condition. And it is also possible the same speaker may appear multiple times in the stream. In order to process and transcribe the speech content in the audio stream of the nature, we need:

- *Segmenting* the audio stream into homogeneous regions based on speaker identity, environmental condition and channel condition so that regions of different properties can be handled differently: for example, regions of background music and noise can be rejected or filtered out.

- *Clustering* speech segments into homogeneous clusters based on speaker identity, environment and channel. Unsupervised learning principle can then be applied to each cluster to improve the accuracy of recognition.

  A number of segmentation algorithms are proposed in the field, and it is shown as the following:

a) Decode-guided segmentation.

For decode guided segmentation method, it first decodes input audio stream and then cuts the input at the location of silence to get the desired segments. Other information such as the gender information could also be utilized in the process.

However, this method it not very successful in detecting acoustic changes of the input data. It is because the algorithm only places decision boundaries at the silence location,, which in general is not precise and has no direct connection in detecting acoustic changes in the data.

b) Model-based segmentation.

The Model-based segmentation method proposes to build different models, e.g. Gaussian mixture models, for a fixed set of acoustic classes, such as pure speech, background music, etc. From a training set, the input audio stream is classified by maximum likelihood selection criterion over a sliding window. The segmentation is made where there is a change in the acoustic class.

c) Metric-based segmentation

Metric-based approach proposes to segment the audio stream at maximum of the distances between neighboring windows placed at every sample. The distance uses one of the measures like KL distance, the generalized likelihood ratio distance, etc.

d) Disadvantages of the algorithms

As mentioned above, the algorithm a) is not accurate in the sense that it does not have direct connection is change detection. The algorithm b) and c) are not so good in detection the acoustic changes in the data either. It is because both model-based segmentation and metric-based segmentation depended on threshold of measurements, so we have to choose a good threshold in order for the algorithm to work well. Hence, they lack stability and robustness. Besides, the model-based segmentation does not utilize unseen acoustic conditions.

- **Hierarchical clustering**

Clustering of audio segments is often done by hierarchical clustering. And it consists of the following steps:

Step 1: a distance matrix is computed, the common distance measure includes KL distance and generalized likelihood ratio. Each audio segment is considered as a Gaussian in the space.

Step 2: the bottom-up hierarchical clustering can be performed to generate a clustering tree. They main difficulty of implementing the algorithm is that: it is hard to determine the number of clusters. One may solve the problem by heuristically pre-determine the number of clusters, and then correspondingly go down the tree to obtain desired clusters.

Another heuristic solution is to threshold the distance measures during the hierarchical process, and the threshold value is tuned on the training set of the data.

7.2.2 Our Approach For Audio Scene Change Detection

In this section, we will describe our approach to detect the changes of speaker, environmental and channels. We will first introduce the model selection criterions we used in the statistics literature, and then explains the maximum likelihood method applied for acoustic change detection. Finally, we will mention the clustering algorithm based on BIC.

(a) Model Selection Criteria and BIC

I. Overtraining problem and model selection

The major problem with model identification is how to choose one among a number of candidate models so that it can describe a given data set best. Usually we have a set of candidate models with different number of parameters. Apparently the more parameters we have, the more likelihood we can train the data well. However, when the number of parameters in the model increases, the model becomes more complex, and what's more, we encounter the problem of *overtraining*, which means that the underlying distribution can estimate the output of training set pretty well on current input data set, but cannot predict the input data well in the future. So we need to propose an efficient algorithm that can solve the dilemma between the training performances and the number of model parameters. It is called a *model selection*.

II. Bayesian Information Criterion (BIC) for model selection

Several model selection criteria have been proposed in the statistical literature, ranging from non-parametric methods such as cross-validation, to parametric methods such as the Bayesian Information Criterion (BIC).

Bayesian Information Criterion (BIC) is a likelihood criterion. The main principle is to penalize the system by the model complexity: the number of parameters in the model:

Let $\chi = (x_1, x_2, \ldots x_N)$ be the input data set we are modeling. Let $M = (m_1, m_2, \ldots m_k)$ be the candidate models with desired parameters. If we maximize the likelihood functions separately for each model *M* and obtain, say, L (χ ,*M*). Let #(*M*) denotes the number of parameters in the model *M*. The BIC criterion is defined as:

$$BIC(M) = \log L(\chi, M) - \lambda \frac{1}{2} \#(M) * \log(N)$$

where λ is the weight of the penalty and it is equal to 1 in our system. Actually we can adjust the weight λ to be different values from 0 to 1, and it determines how the system penalizes the complexity of the model.

From above formula, we can see that BIC tries to choose the best candidate considering two factors: the likelihood function and the complexity of the model. It is applied by selecting the model with maximum BIC values.

This procedure can be derived as a large-sample version of Bayes procedures in the condition that the sample data is independent, identically distributed and linear.

III. Application of BIC

The BIC criterion is a well-known statistical literature and it has been widely used for model identification in the modeling, time series, linear regression, etc. In the engineering study it is known as *minimum description length (MDL)*. It has wide range of application in the speech recognition area, for example, speaker adaptation.

(b) Detection of changes using BIC

In this section, we introduce the maximum likelihood approach we applied for acoustic change detection, and it is based on BIC criterion.

Let use denote $\chi = \{x_1, x_2, ... x_N\}$ as a sequence of vectors extracted from the entire audio stream. Assume *x* is drawn from an independent multivariate Gaussian distribution:

$$x_i \sim N(\mu_i, \Sigma_i)$$

where μ_i is the mean and Σ_i is the covariance matrix of the distribution.

I. Detection a change of single point

Let's first examine the simplest case: assume that there is only one change point in Gaussian process. If the change happened at time *i,* we define:

$$H_0 : x_1, x_2, ... x_N \sim N(\mu, \Sigma)$$

to be the whole sequence without change where as:

$$H_1 : x_1, x_2, ... x_i \sim N(\mu_1, \Sigma_1)\,;\;\; x_{i+1}, x_{i+2}, ... x_N \sim N(\mu_2, \Sigma_2)$$

is the hypothesis that change occurring at time *i*.

The maximum likelihood ratio is defined as:

$$R(i) = N \log |\Sigma| - N_1 \log |\Sigma_1| - N_2 \log |\Sigma_2|$$

where Σ, Σ_1 and Σ_2 are the sample covariance matrices from all the data, from $\{x_1, x_2, \ldots x_i\}$ and from $\{x_{i+1}, x_{i+2}, \ldots x_N\}$ respectively. Now we can obtain the maximum likelihood function of changing point:

$$\hat{t} = \arg\max_i R(i)$$

As a matter of fact, we can view the hypothesis testing as a problem of model selection. We should select from two models: one with the data coming from two Gaussians; the other with data coming from one Gaussian.

The difference between the BIC values of these two models can be expressed as:

$$BIC(i) = R(i) - \lambda P$$

where the likelihood ratio $R(i)$ is as described above and the penalty *P* is defined as:

$$P = \frac{1}{2}(d + \frac{1}{2}d(d+1))\log N$$

here we choose the penalty weight λ to be 1 for simplicity and we can modify it in our system for further adjustment. In the formula *d* is the dimension of space.

If BIC value is positive, it means that two Guassian models are favored than one. So we decide there is a change if

$$\max_i BIC(i) > 0$$

It is evident that the maximum likelihood estimation of the changing point can be expressed by:

$$\hat{t} = \arg\max_{i} BIC(i)$$

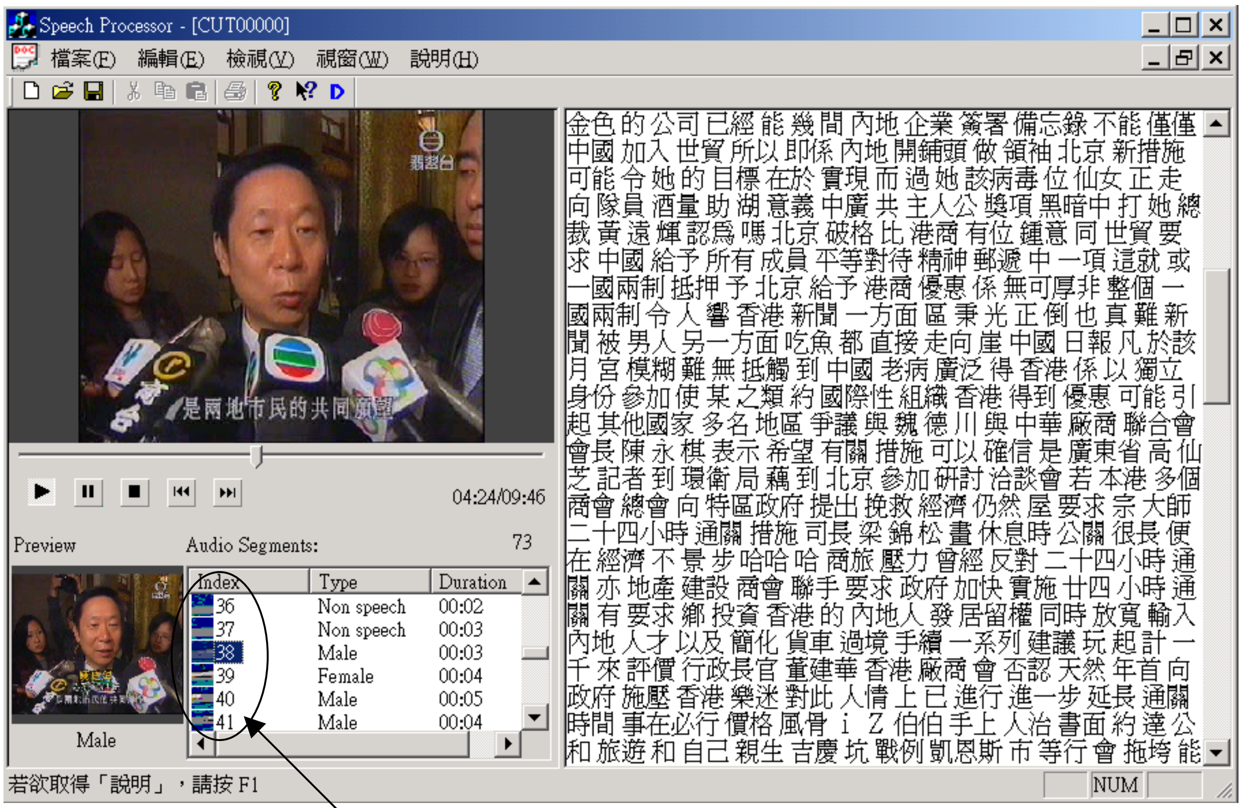
II. Detection changes of multiple points

Extending from the detection of change for a single point, we propose the algorithm to sequentially detect the multiple changing points in the Guassian process:

```
a. Initialize the interval [a, b] with a=1, b=2
b. Detect if there is one changing point in interval [a, b] using
   BIC
c. If (there is no change in [a, b])
        let b= b + 1
   else
        let $\hat{t}$ be the changing point detected
        assign a = $\hat{t}$ +1; b = a+1;
   end
d. go to step (b) if necessary
```

By expending the window [a, b], the final decision of a change point is made based on as much data points as possible. The advantages of that is it is more robust than decisions based on distance between two adjacent sliding windows of fixed size, but the disadvantage is that it costs more.

Below is the interface of our system showing the detected audio scene changes by BIC. Each segment is represented by the key frame image of that segment. And it is sorted according to the timing information obtained from speech recognition engine. In addition to scene change information, we can also get the duration of the clip, gender of the speaker and speech/non-speech attribute of the audio data; we will introduce the methods to handle these later.



Index	Type	Duration
36	Non speech	00:02
37	Non speech	00:03
38	Male	00:03
39	Female	00:04
40	Male	00:05
41	Male	00:04

```
金色的 公司 已經 能 幾間 內地 企業 簽署 備忘錄 不能 僅僅
中國 加入 世貿 所以 即係 內地 開鋪頭 做 領袖 北京 新措施
可能 令 她的 目標 在於 實現 而 過 她 該病毒 位 仙女 正 走
向 隊員 酒量 助 湖 意義 中廣 共 主人公 獎項 黑暗中 打 她 總
裁 黃 遠 輝 認爲 嗎 北京 破格 比 港商 有位 鍾意 同 世貿 要
求 中國 給予 所有 成員 平等對待 精神 郵遞 中 一項 這就 或
一國兩制 抵押 予 北京 給予 港商 優惠 係 無可厚非 整個 一
國兩制 令 人 響 香港 新聞 一方面 區 秉 光 正 倒 也 真 難 新
聞 被 男人 另一方面 吃魚 都 直接 走向 崖 中國 日報 凡 於該
月 宮 模糊 難 無 抵觸 到 中國 老病 廣泛 得 香港 係 以 獨立
身份 參加 使 某 之類 約 國際性 組織 香港 得到 優惠 可能 引
起 其他國家 多名 地區 爭議 與 魏 德 川 與 中華 廠商 聯合會
會長 陳 永 棋 表示 希望 有關 措施 可以 確信 是 廣東省 高 仙
芝 記者 到 環衛 局 藕 到 北京 參加 研討 洽談會 若 本港 多個
商會 總會 向 特區政府 提出 挽救 經濟 仍然 屋 要求 宗 大師
二十四小時 通關 措施 司長 梁 錦 松 晝 休息時 公關 很長 便
在 經濟 不 景 步 哈哈 哈 商旅 壓力 曾經 反對 二十四小時 通
關 亦 地產 建設 商會 聯手 要求 政府 加快 實施 廿四 小時 通
關 有 要求 鄉 投資 香港的 內地人 發 居留權 同時 放寬 輸入
內地 人才 以及 簡化 貨車 過境 手續 一系列 建議 玩 起 計 一
千 來 評價 行政長官 董建華 香港 廠商 會 否認 天然 年首 向
政府 施壓 香港 樂迷 對此 人情 上 已 進行 進一步 延長 通關
時間 事在必行 價格 風骨 ｉ Ｚ 伯伯 手上 人治 書面 約 達 公
和 旅遊 和 自己 親生 吉慶 坑 戰例 凱恩斯 市 等行 會 拖垮 能
```

Audio Scene Change Detection

Figure 7.2 Audio Scene Change Detection

III. Advantages of BIC approach

Compared with metric-based segmentation algorithm mentioned before, the BIC procedure has some advantages:

- *Robustness*

  Metric-based algorithm measure the variation at location *I* as the distance between a window to the left and a window to the right. Generally the size of window is short, e.g. two seconds, and the distance can be chosen to be the log likelihood ratio distance or the KL distance. Such measurements are often noisy and not robust, because it includes only a limited number of samples in two short windows.

  In contrast, our BIC criterion is rather robust since it computes the variation at time *I* using as many samples as possible. Figure 1 demonstrate the robustness of our system:

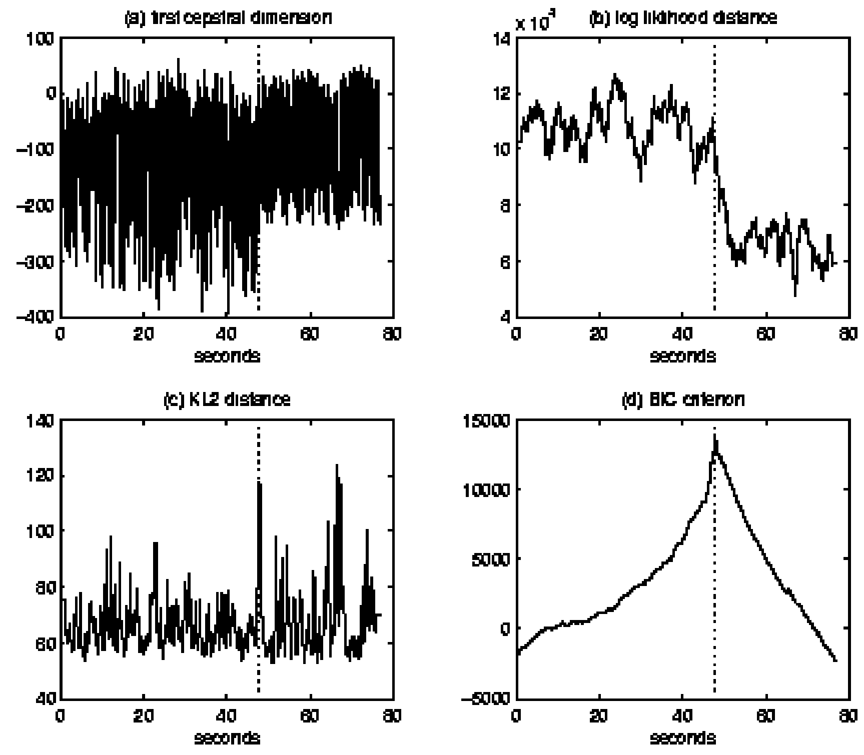


Figure 7.3 Detection of a single changing point

We experimented on a speech signal of 70 seconds and it contains two speakers.

Diagram (a) plots the first dimension of the cepstral vectors. The dotted line in it indicates the location of change. From observation we can clearly notice the changing behavior around the changing point. We calculate both the log likelihood ratio distance and the KL distance between two adjacent sliding windows of 100 frames.

Diagram (b) shows the log likelihood distance: it gets the local maximum at the location of change. However, we notice that it has several maximum, which do not correspond to any changing points and it seems to be rather noisy.

Similarly diagram (c) shows the KL distances: there is a sharp spike at the location of change; however, there are several other spikes, which do not correspond to any changing points.

Diagram (d) displays the BIC criterion; it clearly predicts the changing point.

- *Thresholding-free*

  One of the major advantages of BIC is that it automatically performs model selection, whereas other algorithms are based on thresholding.

  As shown in Figure 1(b) and (c), it is difficult to set a threshold value to pick the changing points. Whereas figure 1(d) indicates there is a change since the BIC value at the detected changing point is positive.

- *Optimality*

  Our procedure is derived from the maximum likelihood learning principle and the model selection theory. It can be shown that it converges to the true changing point (global maximum) as the sample set increases.

  The performance of our procedure heavily depended on the amount of data available for each of the two Gaussian models separated by the true changing point. Here we define the *detectability* of a changing point at time *t* as:

  $$D\ (t) = \min\ (t, N\text{-}t).$$

  Usually the BIC procedure is les accurate as the detectability decreases, which can be illustrate by the following experiment:

  If we place the multiple windows of the same size around a speaker changing point in an audio stream, with each window having different detectability. Within each window, the BIC procedure is applied to detect the change. Figure 2 plots the BIC value versus the detectability of the sampling:

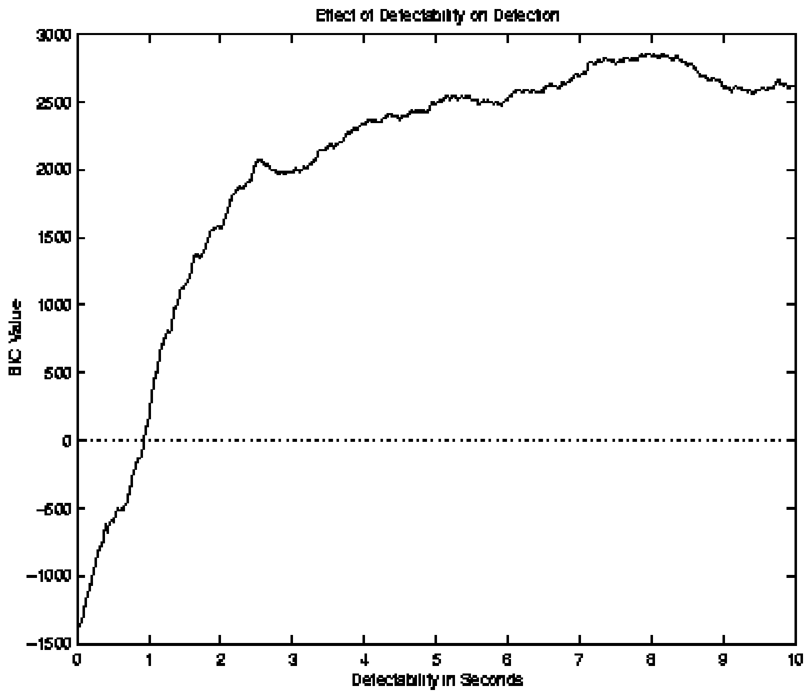


Figure 7.4. Detectability vs BIC values

From the figure we can see that the BIC value starts as negative, meaning that there is no speaker change for the audio data. But as the detectability increases, the BIC value also increases dramatically, it is well above zero for more than 2 seconds, strongly supports the change point hypothesis.

IV. Conclusion of BIC

The BIC criterion can be regarded as thresholding the log likelihood distance, with the threshold level automatically chosen as $\lambda \frac{1}{2}(d + \frac{1}{2}d(d+1))\log N$ where *N* is the size of the decision window and *d* is the dimension of feature space.

The accuracy of our system depends on the detectabilities of the true changing points. Let T={*ti*} be the true changing points, the detectability is defined as :

$$D(t_i) = \min(t_i - t_{i-1} + 1,\ t_{i+1} - t_i + 1)$$

When the detectability is low, the current changing point is often missed. What's more, the error accumulates to the next Gaussian model, so it affects of detection of the next changing point.

Finally, our algorithm works with quadratic complexity.

(c) Clustering algorithm via BIC

In this section, we introduce how to apply the BIC criterion to do the clustering of audio data.

I. Basic Concept

Let S={s1, s2… SM} denote the collection of signals we want to classify, and each of the signal is associated with a sequence of independent random variables $X^i = \{x_1^i, x_2^i ... x_n^i\}$. In the context of speech clustering, S is a collection of audio segments, and X^i is the cepstral vectors extracted from the I'th segment. Let $N = \sum_i n_i$ denote the total sample size of the vectors X^i.

Let $C_k = \{c_1, c_2 ... c_k\}$ denote the clustering that has k clusters. Each of the cluster is modeled as a multivariate Gaussian distribution $N(\mu_i, \Sigma_i)$ where μ_i is the mean of the sample vectors and Σ_i can be estimated as the sample covariance matrix. So we can imply that the number of parameters of each cluster is:

$$d + \frac{1}{2} d(d+1)$$

Let *ni* denote the numver of samples in cluster I, so the BIC value of the cluster is:

$$BIC(c_k) = \sum_{i=1}^{k} \{-\frac{1}{2} n_i \log |\Sigma_i|\} - \lambda P$$

where P is penalty factor:

$$P = \frac{1}{2}(d + \frac{1}{2} d(d+1)) \log N$$

Again we use weight coefficient λ =1 in our implementation. And we select the clustering that maximizes the BIC values.

II. Hierarchical Clustering using greedy BIC criterion

One problem with searching of BIC criterion is that it costs so much to find a global optimum solution. Since clustering has to be performed to obtain different numbers of clusters. So we improve it by a hierarchical clustering algorithm, which is possible to optimize the BIC criterion in a greedy fashion.

The bottom-up methods start with each signal as on initial node, and then it successively merge two nearest modes according to a distance measure. Let S={s_1, s_2… s_k} be the current set of nodes, and s_1 and s_2 are the candidate nodes to merge. Let s denote the new node merge from s_1 and s_2. Thus we want to compare the current clustering S with a new one S'={s, s_3… s_k}. Each node is modeled as a multivariate Gaussian distribution $N(\mu_i, \Sigma_i)$. So the increase of the BIC value by merging s_1 and s_2 is:

$$BIC = n \log|\Sigma| - n_1 \log|\Sigma_1| - n_2 \log|\Sigma_2| - \lambda P$$

where $n=n_1+n_2$ is sample size of merged node. Σ is the sample covariance matrix of the merged node, the penalty is :

$$P = \frac{1}{2}(d + \frac{1}{2}d(d+1))\log N$$

Our BIC termination procedure is that two nodes should not be merged together if the BIC value is negative. Because the BIC value increases at each merging. Note that we are searching for an optimal clustering tree by optimizing the BIC criterion in a greedy fashion.

Here we only use BIC criterion for termination. It is possible to use it as the distance measure in the bottom-up process. But it is better to use a more sophisticated measure in our applications. And the method is also applied for top-down design.

7.2.3 Summary of BIC

In our system we use maximum likelihood approach to detect changing points in an independent Gaussian process; the decision of a change is made on the BIC criterion. The key features of our method are:

- Instead of making local decision based on distance between two adjacent sliding windows of fixed size, we expand the decision window as much as possible so that our final decision of changing points are more *robust*
- Our approach is thresholding-free. The BIC criterion can be regarded as thresholding the log likelihood distance, it automatically chooses the thresholding level as $\frac{1}{2}(d+\frac{1}{2}d(d+1))\log N$ where *N* is the size of the decision window and *d* is the dimension of the feature space.

We also proposed to apply the BIC criterion as a termination criterion in the hierarchical clustering. The change detection algorithm can successfully detect acoustic changing points with reasonable detectability (greater than 2 seconds). And it is also able to choose the number of clusters according to the intrinsic complexity present in the data set and produce clustering solution with high purity.

One point to note is that the penalty weight parameter λ can be tuned to obtain various degrees of segmentation and clustering. It is a value between 0 and 1. The smaller weight will result in more changing points and more clusters. However, we use the value of 1 in our system implementation.

7.3 Gender Classification by Voice

7.3.1 Motivation and Purpose of Gender Classification

a. Background

People can always recognize and categorize the patterns of the acoustic signal from a person's speech without any difficulty. For example, we can convert the signals into linguistic information, information concerning the speaker's identity and speaker's personality, and also we can often discern the speaker's emotional state, age, dialect, and health, etc. However, automatic speech recognition (ASR) is far from accomplishing these tasks. Progress is being made on isolated word and continuous speech recognition. But the speaker must usually train the speech recognition engine and word vocabularies are generally in small size. Under well-constrained situations, computerized speaker recognition system can identify or verify a talker, but much less capably than a listener.

b. Motivation and Purpose

People feel that speech recognition and speaker identification or verification would be assisted if we could automatically recognize a speaker's gender. This would allow different speech analysis algorithms for each gender, facilitating speech recognition by cutting the search space in half. It also helps us to build gender-dependent recognition model and better training of the system. The synthesis of high quality speech would benefit since acoustic features for synthesizing speech for either gender would be identified. And finally, we presumed that the results of gender classification would provide new guidelines for future research to develop quantitative measures of speech quality and to develop new methods to identify acoustic features related to dialect and speaking style.

7.3.2 Algorithms for Gender Classification

In our project, we tried to study and understand different algorithms for automatically recognizing the gender of a speaker. Here we introduce two of them. It uses acoustic parameters extracted from the speaker's voice. The audio input used for developing the algorithms were taken from a large data set. Only acoustic parameters for vowels and fricatives were used to develop and test.

Algorithm I

Algorithm I uses LPC (Linear Predictive Coding) analysis and template pattern recognition. The LPC analysis conditions are:

- Analysis frame: 265 points/frame
- Analysis window: hamming
- Analysis: Pitch asynchronous, autocorrelation
- Data set for coefficient calculations: six frames total, the autocorrelation (or cepstral) coefficient were obtained for each frame and then average.

- Filter order: 16 coefficients
- Frame overlap: none
- Speech pre-emphasis factor: 0.95

Reference and test templates are prepared using various coefficient sets, including the following: LPC, cepstral autocorrelation (ARC), reflection (RC), and Mel-frequency cepstral (MFCC).

It also uses a feature vector that consists of the frequencies, bandwidths, and amplitude of the first through fourth formants and the fundamental frequency of voicing (F_0). For this feature vector F_0 is calculated using a modified cepstral algorithm. Formants are calculated by a peak-picking technique. This feature vector is denoted as (FFF) for formant and fundamental frequency.

Two training-testing procedure are employed:

One is called exclusive, which is the leave-one-out or jack-knife method. The procedure uses all the data, except the data for one subject, which is left out, to train the classifier. The data left out becomes the test template.

The other procedure is inclusive, also known as the re-substitution method, which means that all the data is used to develop the reference template and then one template becomes the test template.

We also examine five distance metrics:

- Euclidean distance (EUC)
- Weighted Euclidean distance (WEUC)
- Probability Density Function (PDF)
- Log Likelihood (LLD)
- Cepstral (CD)

The Euclidean distance is

$$\underset{EUC}{D(X,Y)} = [(X-Y)^T(X-Y)]^{\frac{1}{2}}$$

where X and Y are the test and reference vectors, respectively, and T denotes transpose.

The weighted Euclidean distance is defined as:

$$\underset{WEUC}{D(X,Y)} = [(X-Y)^T W^{-1} (X-Y)]^{\frac{1}{2}}$$

where we have the symbols defined above and W denotes the covariance matrix obtained from the reference vector.

The likelihood distance uses the probability density function, and is defined as:

$$\underset{PDF}{D(X,Y)} = [\frac{1}{2\pi}]^{n/2} \frac{1}{|W|^{1/2}} \exp[\frac{-(x-y)^t w^{-1}(x-y)}{2}]^{1/2}$$

The LPC log likelihood distance is the Itakura distortion measure:

$$D_{LPC}(\hat{a},a) = [\frac{aRa'}{\hat{a}R\hat{a}'}]$$

where a and $\hat{a}$ are the LPC coefficient vectors of the reference and test data and R is the matrix of autocorrelation coefficients of the test data.

The cepstral distortion measure is:

$$DCD(\hat{c},c) = \sum_{i=-N}^{N} (C_i - C_i')^2$$

where $C_0=0$ and $\{c_i\}$ and $\{c_i'\}$ are the cepstral coefficient sequences for the reference and test data, respectively.

Results:

The following table shows the classification result by using ARC, LPC, MFCC, FFF, RC and CC for vowels, unvoiced fricatives and voiced fricatives. The data is analyzed using the Euclidean and probability density distance metrics for both exclusive and inclusive procedures.

	Percent Correct	
Sustained Vowels	LPC (PDF)	85.30
	FFF (PDF)	91.15
	RC (EUC)	93.00
Unvoiced Fricatives	RC (EUC)	80.77
	MFCC (EUC)	82.69
	CC (EUC)	84.62
Voiced Fricatives	LPC (EUC)	89.23
	RC (EUC)	91.15
	LPC (LLD)	91.15
	CC (EUC)	93.08

TABLE 7.1 Most effective features and distance metric combinations for speaker gender classification

Algorithm II

The same data set is used to develop this algorithm. But it uses only one sustained vowel, /i/, as in beet. The approach uses LPC analysis to find the spectral features. The thresholds are developed for these features rather than templates. Gender classification is achieved by comparing the test data features to the appropriate threshold values using the if-then algorithm.

The LPC analysis conditions are:

- Analysis frame: 1 pitch period
- Analysis window: hamming
- Analysis: Pitch synchronous, covariance
- Data set for coefficient calculations: varied the number of frames, using 1,5, 10, 50, and 100. The data is normalized on a pitch period basis using the squared energy and the mean is moved.
- Filter order: 10 coefficients
- Frame overlap: none

- Speech pre-emphasis factor: 0.90

The feature selection is preformed using data for the sustained vowel /i/ for 15 subjects only. This is done using a vector quantization technique. For a given number of data frames (successive pitch periods), the LPC coefficients are computed for each frame. The centroid of these LPC coefficients is determined. The spectrum of this centroid set of coefficients is calculated and the following three features are noted to distinguish male from female speakers:

- In the frequency interval from zero frequency to the first formant, either the slope of the spectrum is positive or it is first negative and then becomes positive. The former feature is typical for males and the latter one is typical feature for females.
- The frequency intercept of the spectrum following the first formant with the zero dB line is greater for female than male speakers.
- The amplitude different, between the spectrum level at zero frequency and the spectrum level of the first formant varied depending on the gender of the speaker. The large amplitude difference, in dB, is noted for male speakers and a small value denoted female speakers.

The three features are parameterized for gender classification based on these observations, and critical frequency *fc* is empirically determined to be 256 KHz for analysis conditions.

Below shows the plot of histograms for female/male speakers. It is evaluated on the frequency count and value domain. We can see that the amplitude difference between male and female speakers is evident and it is the third features to distinguish male from female as mentioned above.

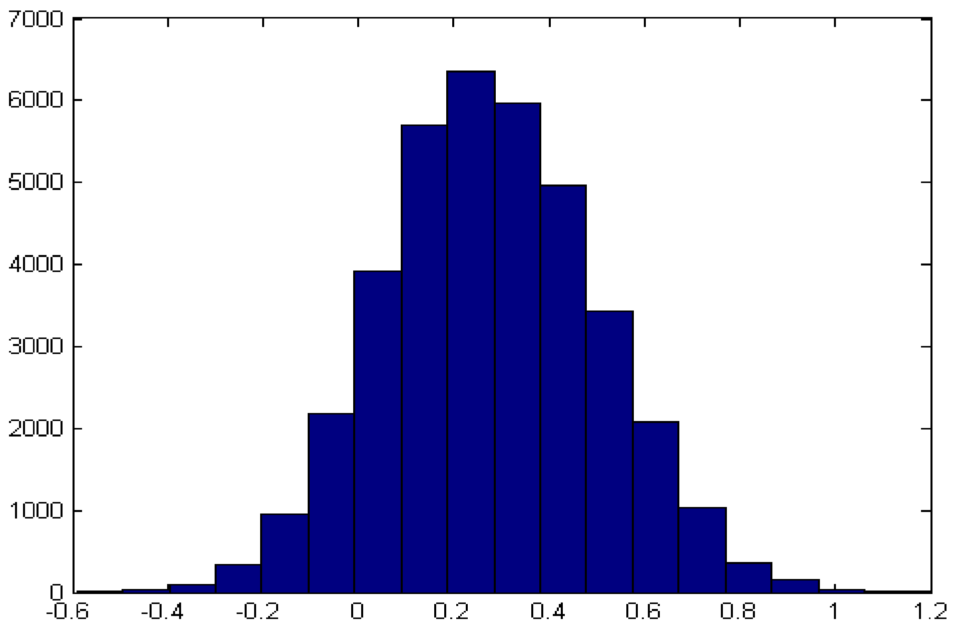


Figure 7.5 Female histogram

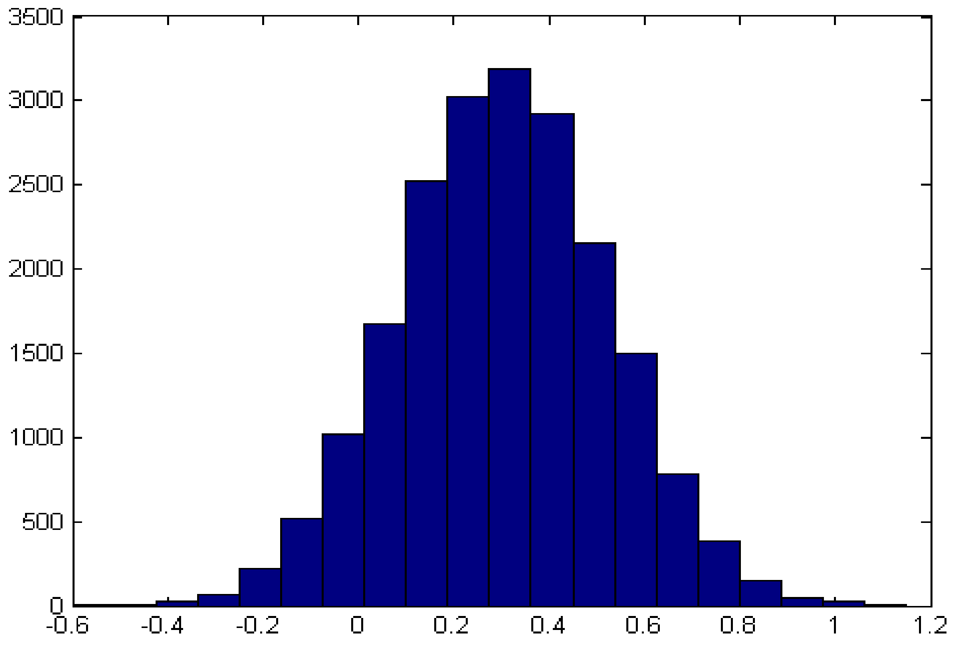


Figure 7.6 Male histogram

Gender classification is then performed as the followings:

Algorithm for gender recognition

Step 1: Obtain N sets of pitch synchronous LPC coefficients for the sustained vowel /i/

Step 2: Find the centroid vector of N sets of LPC vectors

Step 3: By examining the log spectrum of the centroid vector, extract the following features:

- If there exists a spectral slope sign change in the frequency interval from zero to the first formant, then set SCHAN=1, otherwise SCHAN = -1.
- If the LPC spectrum at frequency *fc* is greater than zero, then set FRTHR=1, otherwise FRTHR=-1.
- If the amplitude difference, between zero frequency and the first formant frequency is larger than 10 dB, then set ADIFF=1, otherwise ADIFF=-1.

Step 4: Let FACTOR = SCHAN+FRTHR

Step 5: If FACTOR>0, then the subject is FEMALE

If FACTOR<0, then the subject is MALE

If FACTOR=0, then go to step 6

Step 6: If ADIFF>0, then the subject is MALE

If ADIFF<0, then the subject is FEMALE.

A modified form of the Itakura-Saito distortion measure was used to compute the centroid of the LPC vectors, it is:

$$D(X,Y) = (X-Y)^T C_x (X-Y)$$

where X and Y represent the LPC vectors for the test and reference vectors, respectively and C_x is the covariance matrix for the test vector.

Results

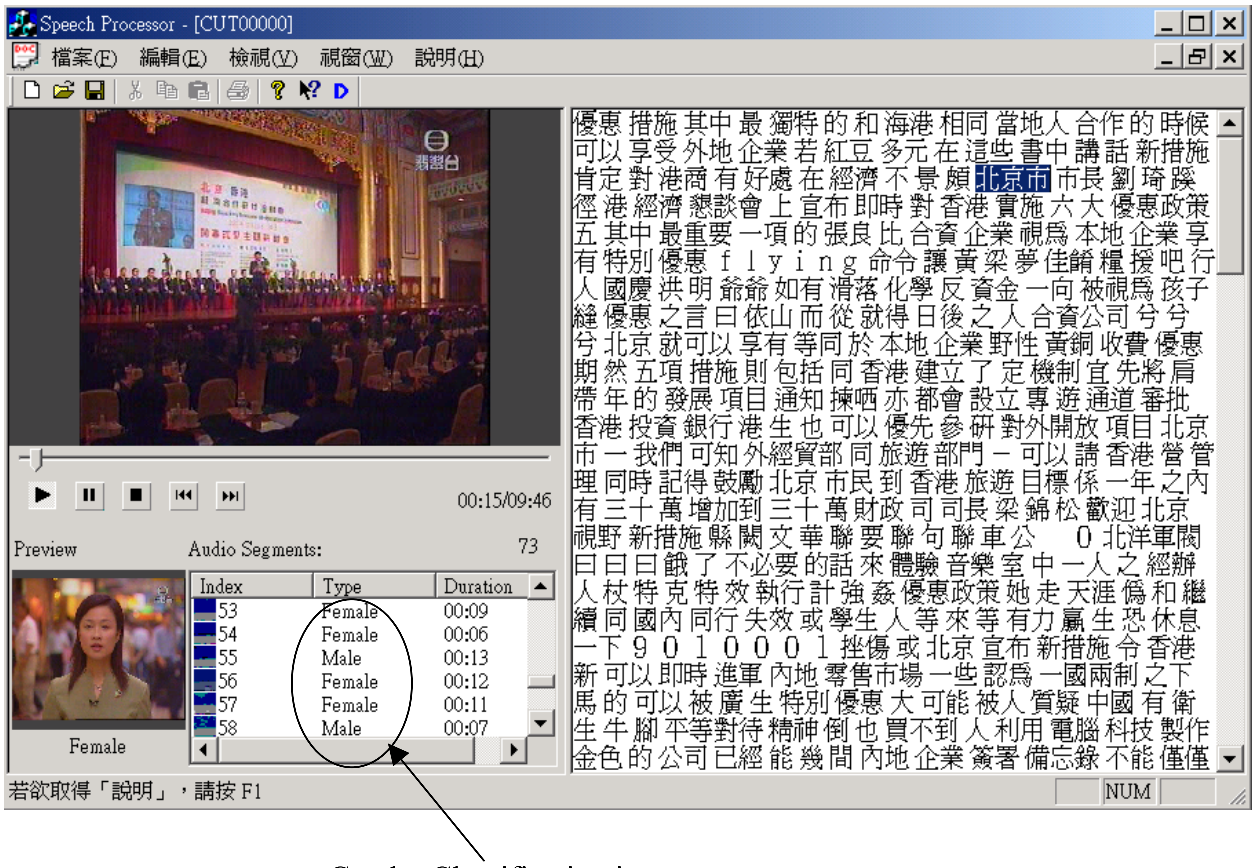
Table 2 represents the gender classification results for this algorithm. If 100 successive pitch periods are used, then 92% correct classification can be achieved.

Number of pitch periods	Percent Correct		
	Male	Female	Total
N=1	85.0	87.2	86.1
N=5	88.0	89.0	88.5
N=10	89.0	90.5	89.75
N=50	91.0	89.0	90.0
N=100	91.0	93.0	92.0

Table 7.2 Algorithm II: sustained vowel /i/

7.3.3 Implementations

Here we can see that the second algorithm for classifying male from female speakers works better than the first one. It utilizes more features of the input audio stream and it gets more than 90 percent accuracy for all the numbers of pitch period. So in our implementation we choose the second approach. And the major factor we considered for distinguishing male from female speakers is the difference of amplitude between the two.


Speech Processor - [CUT00000]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
00:15/09:46
Preview
Audio Segments: 73
Female
若欲取得「說明」，請按 F1
NUM
Gender Classification in our system

Index	Type	Duration
53	Female	00:09
54	Female	00:06
55	Male	00:13
56	Female	00:12
57	Female	00:11
58	Male	00:07

```
優惠 措施 其中 最 獨特 的 和 海港 相同 當地人 合作 的 時候
可以 享受 外地 企業 若 紅豆 多元 在 這些 書中 講話 新措施
肯定 對 港商 有 好處 在 經濟 不 景 頻 北京市 市長 劉 琦 蹊
徑 港 經濟 懇談會 上 宣布 即時 對 香港 實施 六 大 優惠政策
五 其中 最重要 一項 的 張良 比 合資 企業 視爲 本地 企業 享
有 特別 優惠 ｆｌｙｉｎｇ 命令 讓 黃 梁 夢 佳餚 糧 援 吧 行
人 國慶 洪 明 爺爺 如有 滑落 化學 反 資金 一向 被視爲 孩子
縫 優惠 之言 曰 依山 而 從 就得 日後 之 人 合資公司 兮 兮
兮 北京 就可以 享有 等同 於 本地 企業 野性 黃銅 收費 優惠
期 然 五項 措施 則 包括 同 香港 建立 了 定 機制 宜 先將 肩
帶 年 的 發展 項目 通知 揀晒 亦 都會 設立 專 遊 通道 審批
香港 投資 銀行 港 生 也 可以 優先 參 研 對外開放 項目 北京
市 一 我們 可知 外經貿部 同 旅遊 部門 － 可以 請 香港 嘗 管
理 同時 記得 鼓勵 北京 市民 到 香港 旅遊 目標 係 一年 之內
有 三十 萬 增加到 三十 萬 財政 司 司長 梁 錦 松 歡迎 北京
視野 新措施 縣 閣 文 華 聯 要 聯 句 聯 車 公   ０ 北洋軍閥
曰 曰 曰 餓 了 不必要 的話 來 體驗 音樂 室 中 一人 之 經辦
人 杖 特 克 特 效 執行 計 強 姦 優惠政策 她 走 天涯 僞 和 繼
續 同 國內 同行 失效 或 學生 人 等 來 等 有力 贏 生 恐 休息
一下 ９ ０ １ ０ ０ ０ １ 挫傷 或 北京 宣布 新措施 令 香港
新 可以 即時 進軍 內地 零售市場 一些 認爲 一國兩制 之下
馬 的 可以 被 廣 生 特別 優惠 大 可能 被人 質疑 中國 有 衛
生 牛 腳 平等對待 精神 倒 也 買不到 人 利用 電腦 科技 製作
金色 的 公司 已經 能 幾 間 內地 企業 簽署 備忘錄 不能 僅僅
```

Figure 7.7 Gender Classification in our system

7.4 Speech/Non-speech Classification By Multiple Features

7.4.1 Motivation

The problem of distinguishing speech signals from non-speech signals such as music has become increasingly important as automatic speech recognition (ASR) systems are applied to more and more "real-world" multimedia domains. For example, in VIEW project; we have a pool of TV news with speech and non-speech signals mixed together. It is desirable if we can distinguish speech from non-speech signals of audio input data. Then we can build an index of segmented audio clips. The users can select the favorable segments to browse by skipping the ones they do not like to see.

7.4.2 Major Cause of Errors

One major cause of errors in the speech recognition systems is the inaccurate detection of the beginning and ending boundaries of test and reference patterns. Especially, the performance of endpoint detection severely degrades in noisy environments. Some endpoint detection algorithms have been proposed which are based on features of short-time energy and zero-crossing rate. However, these features do not work well in noisy environment. For example, in the TV news, the speaker reads the news script with some background noise in the real environment.

To improve the speech recognition performance in noisy environment, there have been many studies on new features for robust endpoint detection, such as linear prediction error energy, pitch, and band energy, etc. But each of these features does not always guarantee the reliable endpoint detection in the various kinds of environments. And it is necessary to use these parameters together to improve the accuracy of the system.

Since it is very difficult to design the rules that combine multiple features efficiently, statistical approach is considered. In our implementation, we use CART algorithm for speech/non-speech classification of each frame using multiple features.

We performed speech/non-speech classification experiments on the TV news clips. We also investigated the usefulness of various features for speech/non-speech classification in noisy environments.

7.4.3 Structure of Endpoint Detection Algorithm

The conventional endpoint detection process is shown on the diagram below:

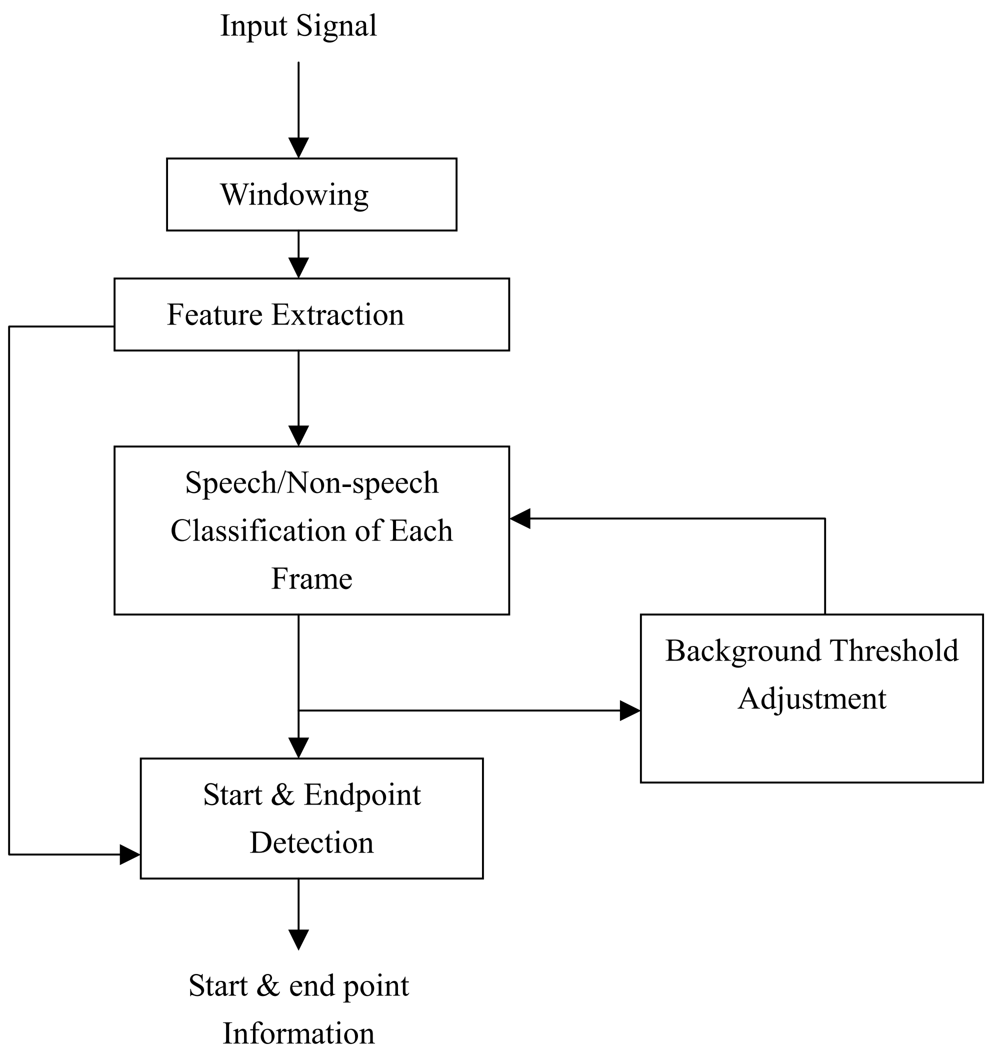

Input Signal
Windowing
Feature Extraction
Speech/Non-speech Classification of Each Frame
Background Threshold Adjustment
Start & Endpoint Detection
Start & end point Information

Figure 7.8: the block diagram of general endpoint detection algorithm

As in the block diagram, input speech signals are segmented into frames using window function and feature parameters such as energy, and zero-crossing rates are extracted. During the non-speech period of several mili-seconds, a few background thresholds are calculated, which are adjusted continuously using the following non-speech frames. Each frame is classified into speech or non-speech using the background thresholds. Endpoint detection logic extracts the start and end point information using the speech/non-speech class information of each frame, the number of successive frames of each class, and the parameter values.

Below shows one method we used to deal with the feature of audio input: pitch tracking. It is evaluated by plotting frequency and number of frames in the domain, and illustrates that speech has more continue contour than non-speech signals. So that we can distinguish them by counting how much percent in clips that have continuous contour. If it has higher percentage of continuous contour, it is more likely to be a speech.

From practical experiment, we obtain that speech clip has 30%-55% of continuous contour, whereas silence or music has 1%-15%. We choose a threshold of more than 20 percent to be speech.

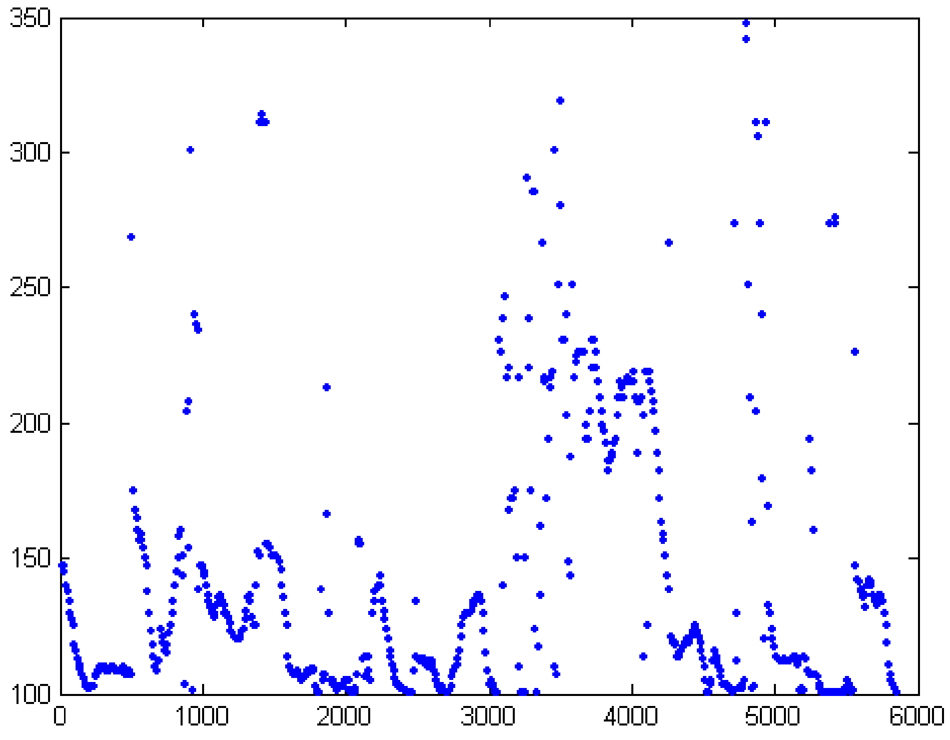


Figure 7.9 Speech pitch tracking

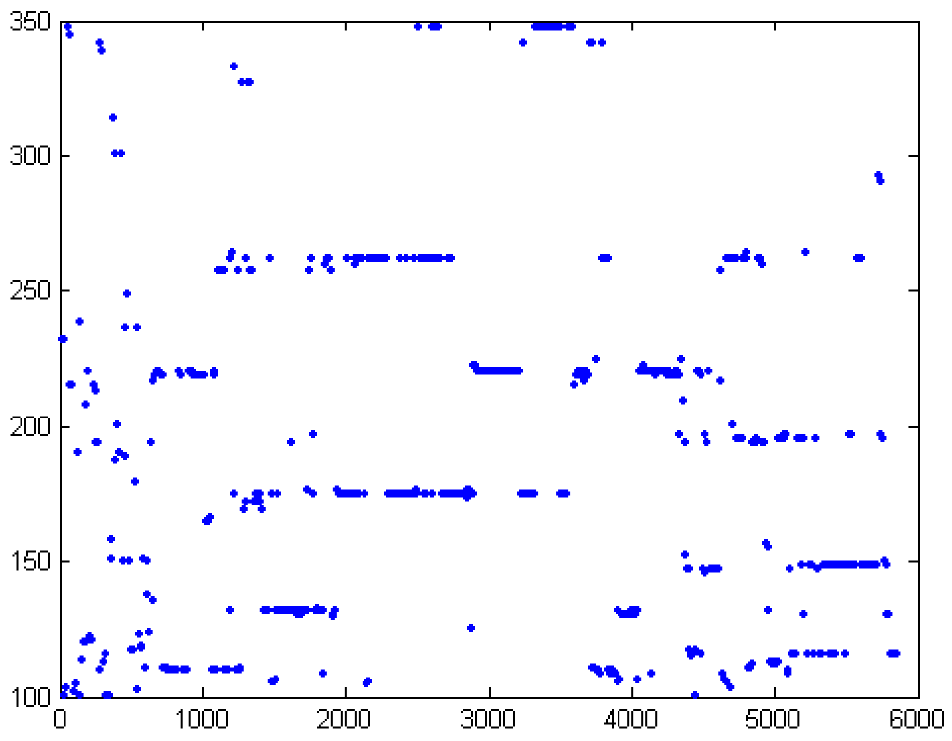


Figure 7.10 Non-speech pitch tracking

7.4.4 Speech/Non-speech Classification Using Multiple Features

Introduction to CART

- It stands for Classification and Regression Trees
- It builds a binary tree by splitting the records at each node according to a function of a single input field
- In CART, the impurity metric is defined by the Gini index. For a node *n*, with possible classes, this is defined as:

$$Gini(n) = 1 - \sum_i p(c_i \mid n)^2$$

where $p(c_i \mid n)$ is the probability of a class given a particular node, that is, the percentage of the observations of that class in the node

- This is minimized when a node contains a single class, i.e, $p(c_i \mid n)=1$ for some *Ci,* and maximized when all classes are equally probable
- Gini can be interpreted as the expected error rate if the class label is chosen randomly at the node
- One problem with growing a tree is to know when to stop. If the tree is too small,

The classifier will have the effect of under-fitting, while a tree that is too large (at the extreme, with leaf nodes containing single observations) will have the effect of over-fitting.

- CART handles this by initially growing the tree too large and then pruning back
- CART uses the adjusted error rate to measure the error rate of a sub-tree (cost-complexity pruning).

Speech/non-speech classification using CART

To consistently extract speech from audio input signals, even in very noisy conditions, the proposed method uses the following multiple features for speech/non-speech classification. The features are full-band energy, band energy of audible frequency range (300-3700Hz) and higher frequency range (2-4kHz), peakyness, LPC-residual energy, and noise-filtered energy. The full-band and audible frequency energy is conventionally frequently. We use higher frequency range band energy to detect the consonants more accurately. But we do not use lower frequency band energy, since some noises such as car noise are centered on lower bands. The peakyness is useful for detection of the voiced part. LPC residual energy is robust to low-band noise such as car noise. Finally the noise-filtered energy is used to remove the influence of the background noise.

The figure below shows the structure of the proposed speech/non-speech classification process. There are multiple pre-classifiers using each feature independently. Each pre-classifier generates speech/non-speech class information for every frame. Since some features make good decision and some do not for each frame, it is necessary to design a rule to make a final decision using the multiple outputs of the pre-classifiers. CART is a convenient tool for designing decision logic, because it does not need a priori knowledge of the input features. Using the speech/non-speech class information from multiple pre-classifiers, CART makes a final decision whether the current frames is a speech or not.

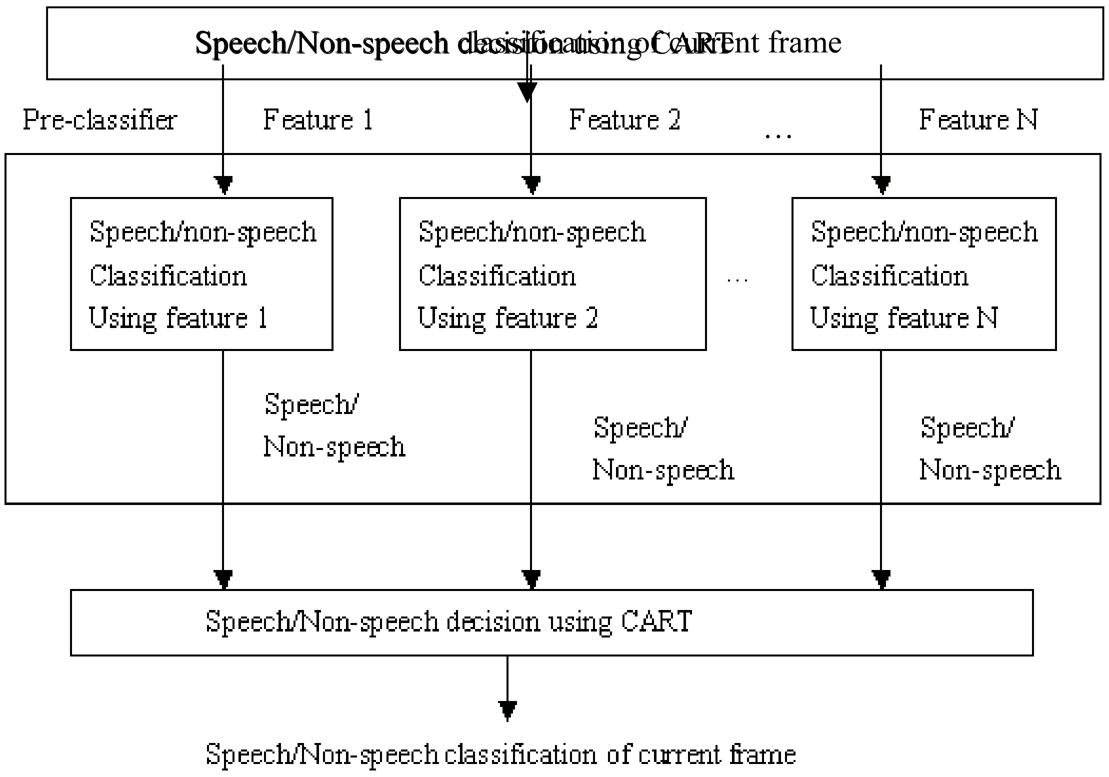

Speech/Non-speech classification of current frame
Speech/Non-speech decision using CART
Pre-classifier
Feature 1
Feature 2
…
Feature N
Speech/non-speech Classification Using feature 1
Speech/non-speech Classification Using feature 2
…
Speech/non-speech Classification Using feature N
Speech/ Non-speech
Speech/ Non-speech
Speech/ Non-speech
Speech/Non-speech decision using CART
Speech/Non-speech classification of current frame

Figure 7.11 Block diagram of the proposed speech/non-speech classification method

7.4.5 Experimental Results

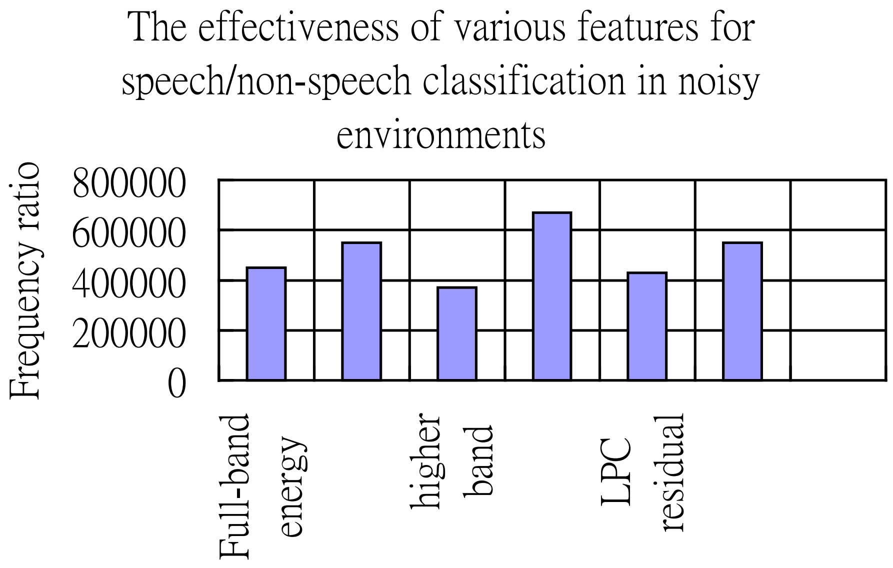
We evaluated speech/non-speech classification performance of the proposed method. And we use the audio data from TVB news as the test data set. For the comparison purpose, we also evaluated the classification performance using each feature separately. The speech/non-speech classification rate is calculated as follows:

Speech/non-speech classification rate = (number of frames coincide with manual classification) / (number of total frames)

The table below shows the proposed methods using multiple features performed better than using a single feature by about 4 to 10%. We can also find that it is more effective to use the information of the previous and the next frames together. The improvements in performance demonstrate that the CART performs well in combining multiple features for decision of speech or non-speech.

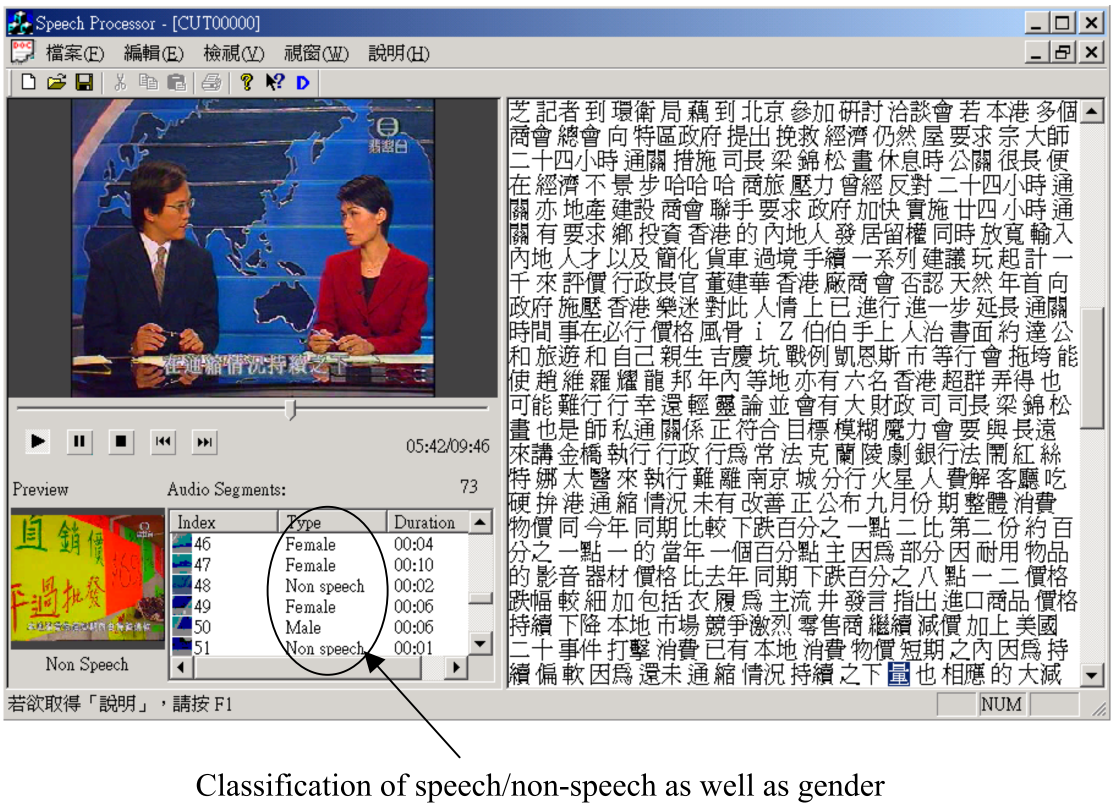
	Speech/non-speech classification rate (%)		
	5dB	0dB	-5dB
I	84.81	80.30	73.93
II	86.77	82.50	76.36
III	81.69	77.27	72.70
III	81.69	77.27	72.70
VI	88.85	84.07	76.80
V	84.35	79.52	74.13
VI	87.20	81.81	75.55
VII	89.45	84.97	78.21
VIII	91.11	87.73	81.16

Table 7.3 Used features and method: I: full-band energy, II: band energy of audible frequency range (300-3700Hz), III: band energy of higher frequency range (2-4kHz), VI: peakyness, V: LPC residual energy, VI: noise-filtered energy, VII: the proposed method (current frame only), VIII: the proposed method (including previous and next frames)



The data in diagram also show that the effectiveness of each feature on the speech/non-speech classification in various noise environments. The results show that the peakyness, the energy of audible frequency range, and the noise-filtered energy performed better than other features. The full band energy, the conventional feature for endpoint detection for clean speech, does not work well for noisy speech.

7.4.6 System Snapshot


Speech Processor - [CUT00000]
檔案(F) 編輯(E) 檢視(V) 視窗(W) 說明(H)
05:42/09:46
Preview
Audio Segments: 73
Non Speech
若欲取得「說明」，請按 F1
NUM
Classification of speech/non-speech as well as gender

Index	Type	Duration
46	Female	00:04
47	Female	00:10
48	Non speech	00:02
49	Female	00:06
50	Male	00:06
51	Non speech	00:01

芝 記者 到 環衛 局 藕 到 北京 參加 研討 洽談會 若 本港 多個
商會 總會 向 特區政府 提出 挽救 經濟 仍然 屋 要求 宗 大師
二十四小時 通關 措施 司長 梁 錦 松 晝 休息時 公關 很長 便
在 經濟 不 景 步 哈哈 哈 商旅 壓力 曾經 反對 二十四小時 通
關 亦 地產 建設 商會 聯手 要求 政府 加快 實施 廿四 小時 通
關 有 要求 鄉 投資 香港 的 內地人 發 居留權 同時 放寬 輸入
內地 人才 以及 簡化 貨車 過境 手續 一系列 建議 玩 起 計 一
千 來 評價 行政長官 董建華 香港 廠商 會 否認 天然 年首 向
政府 施壓 香港 樂迷 對此 人情 上 已 進行 進一步 延長 通關
時間 事在必行 價格 風骨 i Z 伯伯 手上 人治 書面 約 達 公
和 旅遊 和 自己 親生 吉慶 坑 戰例 凱恩斯 市 等行 會 拖垮 能
使 趙 維 羅 耀 龍 邦 年內 等地 亦有 六名 香港 超群 弄得 也
可能 難行 行 幸 還 輕 靈 論 並 會有 大 財政 司 司長 梁 錦 松
晝 也是 師 私通 關係 正 符合 目標 模糊 魔力 會 要 與 長遠
來講 金橋 執行 行政 行爲 常 法 克 蘭 陵 劇 銀行法 鬧 紅 絲
特 娜 太 醫 來 執行 難 離 南京 城 分行 火星 人 費解 客廳 吃
硬 拚 港 通縮 情況 未有 改善 正 公布 九月份 期 整體 消費
物價 同 今年 同期 比較 下跌 百分之 一點 二 比 第二 份 約 百
分之 一點 一 的 當年 一個 百分點 主 因爲 部分 因 耐用 物品
的 影音 器材 價格 比去年 同期 下跌 百分之 八 點 一 二 價格
跌幅 較 細 加 包括 衣 履 爲 主流 井 發言 指出 進口 商品 價格
持續 下降 本地 市場 競爭 激烈 零售商 繼續 減價 加上 美國
二十 事件 打擊 消費 已有 本地 消費 物價 短期 之內 因爲 持
續 偏 軟 因爲 還未 通縮 情況 持續 之下 量 也 相應 的 大減

Figure 7.12 Classification of speech/non-speech as well as gender

In our system we integrate the classification of male/female and speech/non-speech in one index. So that users are free to scroll up and down to browse the type information of the audio segments. When they click on the index of the segments, the corresponding image appears on the left corner of the screen, which facilitates the user to get brief content of that segment.

Chapter 8 : Individual Contribution

Background Study

In last summer, I have studied speech recognitions. Speech recognition is heavily used in our project. But I have little background on this field. Thus, the first task for me is to get some knowledge on speech recognition. I have read some books on this topic. This helped me a lot in my later work of this project.

After that, together with my partner, I have done a comparison study on different speech recognition engines, including CMU Sphinx, Microsoft SAPI, and IBM ViaVoice. And finally, we choose the ViaVoice speech recognition engine for our project.

In late September, last year, we got the IBM ViaVoice SDK. Then, I continued my study on speech recognition and especially ViaVoice. I read the ViaVoice documentation to obtain necessary knowledge for this project. I read carefully about the main application programming interface, speech management API (SMAPI). Through this API, I started to build a program to use ViaVoice to dictate speech in news video clips.

Audio Extraction

The first problem we encountered for this program is that the ViaVoice engine cannot directly take the input from video clips. After a careful study on audio libraries of SMAPI, I realized the engine could take audio input from wave files. But the format of the input wave files is restricted. It must be monotonic and the sampling rate must be 22kHz, 16kHz, or 8kHz. So I started to build an audio extractor. The main task of the audio extractor is to extract audio channels from video clips and store them into monotonic 22/16/8kHz ACM format. In order to achieve this, first I studied DirectX programming. I carefully read DirectX documentation, especially those of DirectShow. After that, I understood the concepts of filters, COM, and filter graphs. I used the two filters, ACM Wrapper and WavDest. The ACM Wrapper can convert the format of ACM data. With it, I can convert the audio data retrieved from video clips into monotonic 22/16/8 kHz format. The WavDest filter can write the converted ACM data into hard disk. With this filter, I can store the converted ACM streams as wave files.

After the audio extractor has been finished, my partner and I extracted some audio data from news video clips. Then, we performed an initial test on the recognition accuracy of the ViaVoice speech recognition engine. The accuracy is not high. So we decided to train the ViaVoice engine and see how much the accuracy can be improved. Before actually train the engine, we also did some preprocessing on the audio data. We segmented the audio data into sentences and removed the silence in these data.

Visual Training Tool

After preprocessing, the speech data are not ready for training yet. We need to compare the output of the speech engine with the correct texts and find those correctly recognized words. Before we can do this comparison, we need to collect large amount of text data. Such input work will require considerable effort under the condition that lack of the help of certain appropriate tool. Thus, we decided to implement a visual training tool to help us train the ViaVoice engine. The tool displays the video, which contains that speaker's voice we want to train. When the user hears what the speaker is saying, he/she can type in the texts of what they have heard. In addition, the dictation output of IBM ViaVoice engine is also shown in the interface.

In order to implement this tool, I studied the Microsoft foundation classes, visual C++ programming, especially the document and view architecture. I developed the following classes, `CWinTrainApp`, `CWinTrainDoc`, `CPlayerForm`, `CDictationView`, `CWinTrainView`, and `CVideo`. I have applied the document and view architecture in this visual training tool. So the interface supports three views, dictation view, player view, and editor view.

In order to integrate the IBM ViaVoice engine into the visual training tool for realtime dictation, I used SMAPI in the tool to invoke the ViaVoice engine. In this program, I used the asynchronous message handling mechanism. And asynchronous message handling is implemented by message handlers. I implemented the following handlers to handle engine callbacks: `ConnectCB`, `DisconnectCB`, `MicOnCB`, `MicOffCB`, `EnableVocabCB`, `RecoNextWordCB`, `RecoWordCB`, `RecoTextCB`, `MicOffReqCB`, `UtteranceCB`, `RecoPhraseCB`, `EngineStateCB`. I also added functions to deal with the recognition interface, including a button to start dictation and a editor that can display the dictated text produced by the engine.

I also implemented the video segment support into the CVideo class. I added two buttons into the player interface. The two buttons are the previous button and the next button. With these two buttons, the user can switch video playback between different segments.

I wrote about 2000 lines of source code for the visual training tool.

Speech Information Processor

After we finished the experiments on ViaVoice, we found out that the training process help us little. So we decided to modify our approach to build a more powerful speech processor.

On the basis of the visual training tool, the first feature I added to it is timing information retrieval. After carefully studying SMAPI, I retrieved firm word time information and built an index to store the times and positions of every firm words. Then, I studied windows timers. I customized the timer function of class CPlayerForm. The customized timer function first checks the current video playback position. And then it performs a binary search on the word index to retrieve the position of the firm word that the video is currently playing. Finally, this timer function forwards the position information to the dictation view to highlight the current firm word.

Because editing the text may change the structure of firm words, I implemented dynamic index alignment to avoid wrong word information. I studied the windows message handling mechanism. I overwrote the OnChar handler of the dictation view class to align word index during editing. When the user input something, the OnChar handler will be called. It first divides user actions into three categories according to the changing effect to the index. Then it will search for the words that are modified and then update them according. Since this align has to be performed every time the user input something, the word search process is invoked often. Thus, its performance is critical. In practice, I found that the binary search algorithm performs well.

We also included audio information retrieval functionalities into our speech processor. I built the interface to present the retrieved audio information to the user. I added a

`CListCtrl` to the `CPlayerForm` class. This list presents the index, type, and duration of each segment to the user. For each segment, I captured the first frame and store it as the icon of the segment entry in the list. I customized two more windows message handlers to respond to the item double click and item selection events.

I wrote about 3000 lines of C++ code for the speech processor.

Overall, I think my contribution to this project includes speech recognition study, audio information retrieval paper study, experimental results analysis, and programming implementation.

Chapter 9 : Summary

9.1 Accomplishment

We have obtained some knowledge of speech recognition. We did a study on different speech recognition engines, including CMU Sphinx, Microsoft SAPI, and IBM ViaVoice. And finally, we choose the ViaVoice speech recognition engine for our project.

After the ViaVoice SDK was shipped to VIEW lab, we performed a careful study on it and discovered that the speech recognition accuracy is a main issue of our project. Then we planed to train the ViaVoice engine to improve the accuracy. First, we implemented an audio extractor, which can extract audio channels from video clips and can convert the audio into a format that can be accepted by ViaVoice. Then, we segmented the audio data into sentences. This can make the training process easier. After that, we built a visual training tool to conduct the training experiment.

The outcome of the experiments showed that the training process brought only a very tiny improvement to the speech recognition accuracy. And the reply of IBM also indicated that the training could achieve little. Therefore, we modified our plan to let the user manually editing the recognition text produced by ViaVoice. Thus, we can get complete and correct transcripts of the video clips in the digital video library. For this purpose, we modified our visual training tool into a speech information processor. The features of this tool include media playback, real-time dictation, timing information retrieval, and dynamic recognition text editing. With these features, one can easily correct mis-recognized words of the raw text produced by ViaVoice. The speech information processor also includes the following features: audio scene change detection, audio segments classification, and gender classification. To implement these audio information retrieval functionalities, we read many related papers. And finally, we integrated the audio information into the interface of our speech information processor.

9.2 Possible Future Work

As shown in our experiments, the accuracy while recognizing indoor news clips is much higher than that of recognizing outdoor news. So if we can classify indoor news and outdoor news automatically, this might help us to make a better use of the IBM ViaVoice speech recognition engine.

As mentioned in IBM's reply for our accuracy problems, if we have a lot of transcriptions of broadcast news, the language model can be adapted accordingly. IBM will have a tool to adapt our existing acoustic model to a target new channel/source, and a tool to adapt our language model to a new domain (such as broadcast news). These should be the essential way to solve our problems. Thus, after these tools become available, we might be able to build our own language model especially for the digital video library. It is expected such a special model will work much better than the original models shipped with ViaVoice.

Acknowledgement

We would like to express our gratitude to Professor Michael R. Lyu, our project supervisor. He has given us many useful suggestions and directions throughout this project.

We would also like to express our thanks to Jackie Chun Keung CHAU, who has helped us to plan the tasks and give us useful suggestions, Edward Hon-hei YAU, who has offered us lot of helps on multimedia programming, and Min Cai, who offered many valuable suggestions for the implementation of this project.

We are fortunate to work in the VIEW lab directed by Professor Michael R. Lyu. Our team of colleagues also deserves our gratitude for helping us in numerous ways while we worked on this project.

Appendix A: References

A-I: Books

- *Fundamentals of Speech Recognition*; Lawrence Rabiner & Biing-Hwang Juang Englewood Cliffs NJ: PTR Prentice Hall (Signal Processing Series), c1993, ISBN 0-13-015157-2

- *Speech recognition by machine*; W.A. Ainsworth London: Peregrinus for the Institution of Electrical Engineers, c1988

- *Speech synthesis and recognition*; J.N. Holmes Wokingham: Van Nostrand Reinhold, c1988

- *Speech Communication: Human and Machine*, Douglas O'Shaughnessy; Addison Wesley series in Electrical Engineering: Digital Signal Processing, 1987.

- *Electronic speech recognition: techniques, technology and applications*, edited by Geoff Bristow, London: Collins, 1986

- *Readings in Speech Recognition*; edited by Alex Waibel & Kai-Fu Lee. San Mateo: Morgan Kaufmann, c1990

- *Hidden Markov models for speech recognition*; X.D. Huang, Y. Ariki, M.A. Jack. Edinburgh: Edinburgh University Press, c1990

- *Speech Recognition: The Complete Practical Reference Guide*; T. Schalk, P. J. Foster: Telecom Library Inc, New York; ISBN O-9366648-39-2; 377 pages; paperback only. Covers speech recognition in a telephony environment and wish to use call processing hardware based in PCs. It is written using Dialogic hardware as the example for the hardware.

- *Automatic speech recognition: the development of the SPHINX system*; by Kai-Fu Lee; Boston; London: Kluwer Academic, c1989

- *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition*, S. E. Levinson, L. R. Rabiner and M. M. Sondhi; in Bell Syst. Tech. Jnl. v62(4), pp1035--1074, April 1983

- *Automatic Speech and Speaker Recognition: Advanced Topics*, C.H. Lee, F.K. Soong and K.K. Paliwal (Eds.), Kluwer, Boston, 1996.

- *Review of Neural Networks for Speech Recognition*, R. P. Lippmann; in Neural Computation, v1(1), pp 1-38, 1989.

A-II: Papers

- *Audio Content Analysis for Online Audiovisual Data Segmentation and Classsification* , by Tong Zhang *Member IEEE*, and C. C. Jay Kuo, *Fellow, IEEE*
- *Construction and evaluation of a robust multi-feature speech/music discriminator* Eirc Scheirer, Malcolm slaney, Interval Research Corp, 1801-C Page Mill Road, Pal Alto, CA, 94304, USA
- *Speaker, environment and channel change detection and clustering via the Bayesian information criterion* P.S. Gopalakrishnan, Scott Shaobing Chen, IBM T,J Watson Rsearch Center
- *Pitch determination and voice quality analysis using subharmonic-to-hormonic ratio*, Xuejing Sun, Department of communication sciences and disorders, northwestern university, IL 60208 USA
- *Segregation of speaker s for speech recognition and speaker identification,* Herbert Gish, man-Hung Siu, and Robin Rohicek, BBN Systems and Technology, Cambridge, MA 02138
- *Speech/non-speech classification using multiple features for robust endpoint detection,* Won-Ho Shin, Byoung-Soo Lee, Yun-Keun Lee and Jong Seok Lee, Information Technology Lab, LG Corporate Institute of Technology
- *Improved methods for vocal tract normalization*, L. Welling, S. kanthak and H. Ney, RWTH Aachen-University of Technology, Aaceh, Germany
- *Automatic Recognition of gender by voice,* D.G.Childers, Ke Wu, K.S.Bae, Dept of Electrical Engineering, University of Florida, Gainesville, FL 32611

- *Audio Content analysis for online audiovisual data segmentation and classification,* Tong Zhang, Member IEEE, and C. C. Jay Kuo, Fellow, IEEE
- *Construction and evaluation of a robust multi-feature speech/music discriminator,* Eric Scheirer, Malcolm Slancy, Interval Research Corp, Pal Alto, CA, USA
- *Audio feature extraction and analysis for scene segmentation and classification,* Zhu Liu and Yao Wang, Polytechnic University, Brooklyn, NY 11201
- *Silent and voiced/unvoiced/mixed excitation (four-way) classification of speech,* D. G. Childers, M. Hahn, and J. N. Larar
- *Sound spotting-a frame-based approach,* Chirstian Spevak, Richard Polfreman, University of Herfordshired, College Lane, Hatfield, UK
- *A Robust and Fast Endpoint Detection Algorithm for Isolated Word Recognition,* **Yiying Zhang, Xiaoyuan Zhu, Yu Hao** State Key Laboratory of Intelligent Technology and Systems, Tsinghua Univeristy
- *The Carnegie Mellon University Distributed Speech Recognition System. Speech Technology* Adams,D. and Bisiani,R. 3(2), April, 1986.
- *BEAM: An Accelerator for Speech Recognition. In International Conference on Acoustics, Speech and Signal Processing.* Bisiani, R IEEE, May, 1989.
- *The use of recurrent neural networks in continuous speech recognition* Tony Robinson, Mike Hochberg and Steve Renals, Cambridge University Engineering Department, August, 1993
- *The application of dynamic programming to connected speech recognition* H.F. Silverman and D.P. Morgan, IEEE ASSP Magazine, July 1990
- *Connectionist probability estimators in HMM speech recognition* S.Renals, N.Morgan, IEEE Transactions on speech and audio processing, Jan, 1994
- *Automatic Speech Recognition: The Development of the SPHINX System.* Boston: Kulwer Academic Publishers, 1989
- *Speaker-independent isolated word recognition using dynamic features of speech spectrum* Furui, IEEE Transactions on Acoustics, Speech, and Signal Processing, Feb, 1986

A-III: Useful URL

- CMU Sphinx http://fife.speech.cs.cmu.edu/speech/
- IBM ViaVoice (Chinese version)

http://www-900.ibm.com/cn/ibm/crl/project/project1.html

- IBM ViaVoice (Cantonese speech recognition)
  http://www-900.ibm.com/cn/ibm/crl/project/cvp.html

- IBM ViaVoice (Putonghua speech recognition)
  http://www-900.ibm.com/cn/ibm/crl/project/vp.html

- Speech Recognition Introduction http://www.macalester.edu/~lkim/ai/sr.html

- Online Bibliography of Phonetics and Speech Technology Publications
  http://www.informatik.uni-frankfurt.de/~ifb/bib_engl.html

- MIT's Spoken Language Systems Homepage http://www.sls.lcs.mit.edu/sls/

- Speech Technology http://www.speechtechnology.com

- Speech Control (Speech Controlled Computer Systems: Microphones, headsets, and wireless products for ASR) http://www.speechcontrol.com/

- Microphone.com: Microphones and accessories for ASR
  http://www.microphones.com/

- Say I Can.com: "The Speech Recognition Information Source."
  http://www.sayican.com/

A-IV: Useful Newsgroups And Forums

Discussion forums dedicated to computer and speech

- US: http://www.speech.cs.cmu.edu/comp.speech/
- UK: http://svr-www.eng.cam.ac.uk/comp.speech/
- Aus: http://www.speech.su.oz.au/comp.speech/

Forums and newsgroups dedicated to users of speech software

- http://www.speechtechnology.com/users/comp.speech.users.html

- news:comp.speech.research

Newsgroup dedicated to speech software and hardware research

- news:comp.dsp

Newsgroup dedicated to digital signal processing

- news:alt.sci.physics.acoustics