

國立交通大學

電信工程學系

碩士論文

利用加速演算法之大詞彙連續語音辨識系統

A Fast Large Vocabulary Continuous Speech
Recognition System

研究生：杜明桓

指導教授：陳信宏 博士

中華民國九十八年七月

利用加速演算法之大詞彙連續語音辨識系統

A Fast Large Vocabulary Continuous Speech
Recognition System

研究生：杜明桓

Student : Ming-Huang Tu

指導教授：陳信宏 博士

Advisor : Dr. Sin-Horng Chen



Submitted to Department of Communication Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Communication Engineering

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

利用加速演算法之大詞彙連續語音辨識系統

研究生：杜明桓

指導教授：陳信宏 博士

國立交通大學電信工程學系碩士班



本論文主要探討如何建構大詞彙連續語音辨識系統的加速演算法及其應用。第一部份先對語音辨識系統各部份的加速演算法做一系列的研究與實做分析，主要由聲學模型、語言模型和搜尋演算法三方面去做加速，而使得辨識率的下降在極小的範圍內。在Treebank語料實驗中，可減少50%以上辨識時間，其辨識率幾乎無衰退；在非特定語者TCC300語料的實驗中，辨識時間可降低22~44%左右，而辨識率只下降在1%範圍內。

其次，第二部份對於辨識系統建立不同的應用，設計出有文法規則的辨識系統，並且使該系統更具彈性。可藉由簡易的方式去調整系統的模型與參數，方便往後使用者去實做出不同需求的語音辨識系統。

A Fast Large Vocabulary Continuous Speech Recognition System

Student: Ming-Huang Tu Advisor: Dr. Sin-Horng Chen

Department of Communication Engineering
National Chiao Tung University



Abstract

This thesis can be divided into two parts. In the first part, large vocabulary continuous speech recognition (LVCSR) by speedup algorithms is constructed. The thesis describes some effective algorithms that reduce the computation of the acoustic model (AM), language model (LM) and search space. In the outside of Treebank, the system recognition speed can be accelerated by more than 50%, and maintain the same recognition accuracy; besides, in the TCC300, the recognition speed also can be accelerated by more than 22%, and the character accuracy just decreases by less than 1%. Therefore the system is capable of the speaker independent recognition.

In the second part of the thesis, a flexible LVCSR for the different applications is built. The user can not only tune up the system's parameter and on line, but also be easy to design the grammar-ruled word net, and compile the language model which the recognition system can read in.

致謝

本篇論文能完成，必須感謝陳信宏老師與王逸如老師的教誨與指導。兩位老師利用各自的觀點帶給自己兩年來充實的研究生涯，也讓這篇論文有了不同的生命。兩位老師帶著親切卻不失嚴謹的態度也讓自己從一位懵懂無知的大學生轉變至今完成一篇碩士論文的研究生，真正感激兩位老師的辛勞與付出。

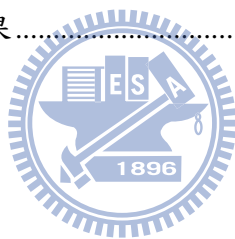
其次，感謝實驗室的各個優秀博班學長。感謝有如小小老闆讓自己問無數問題的性獸；感謝當初拉把我進實驗室的志合；感謝讓我最後能夠心無旁騖作辨識系統的阿德；感謝苦心勸我們愛護地球的希群學長；感謝指導我如何寫程式的巴金學長；還有不時就會恐嚇我們畢業的輝哥，因為您們的經驗傳承使得自己在研究以及生活上無助時有供尋求幫助、安慰的地方。當然也感謝這兩年一起奮鬥的好夥伴：宅中之宅的好男人宋哥；很 Rock & Roll 的普烏；還有帥到不行的小帥哥，以及健身房常客的湯哥，雖然一路上有歡笑有淚水(有嗎!?)，但是我們仍然順利完成碩士學業。當然也感謝碩一的學弟妹，宥余、承燁、舒姐、依玲、皓翔、小卡、嘆馬，因為你們讓實驗室熱鬧許多，也祝福你們可以平平安安、歡歡樂樂地在實驗室度過每一天每一刻。此外也感謝 Apple 姐與胖胖兩位可愛的專題生，這一年多也是一起努力過。最後還要感謝小廣時常抽空從聯發科回來實驗室與我們吃飯跟聽我們的煩惱。最後要特別感謝李庚達學長，因為他建立起這麼好的辨識系統基礎，讓自己接下此系統時能夠快速修改相關演算法與程式碼，讓這篇論文能夠順利地完成。

此外要感謝我的父母跟兄長，對於我所做的每一個決定都非常支持，從我決定離開台南到外地讀書一直到選擇工作都持續在背後支撐著我。讓我雖在外地讀書卻能感受到家中的溫暖。感謝惠卿的鼓勵，這份榮耀亦與妳一同分享。

目錄

中文摘要.....	II
Abstract.....	III
致謝.....	IV
目錄.....	V
表目錄.....	VII
圖目錄.....	VIII
第一章 緒論.....	1
1.1 研究動機.....	1
1.2 文獻回顧.....	2
1.3 章節概要.....	5
第二章 語音辨識與相關演算法介紹.....	6
2.1 背景知識.....	6
2.2 聲學模型 (Acoustic Model).....	7
2.3 發音詞典 (Pronunciation Lexicon).....	9
2.4 語言模型 (Language Model).....	10
2.5 語言模型預查(LMLA).....	11
2.6 維特比演算法(Viterbi Algorithm).....	13
2.7 刪除演算法(Pruning Algorithm).....	14
第三章 即時辨識系統演算法.....	16
3.1 基本概論.....	16
3.2 聲學模型.....	16
3.3 語言模型.....	19
3.4 刪除演算法與其他加速法.....	21
第四章 實驗與分析.....	24
4.1 實驗準備.....	24

4.2 基準實驗測試.....	24
4.3 聲學模型.....	27
4.4 語言模型.....	31
4.5 刪除演算法與其他加速法.....	32
4.6 TCC300.....	36
第五章 建構語音辨識系統.....	39
5.1 系統展示.....	39
第六章 結論與未來展望.....	42
6.1 結論.....	42
6.2 未來展望.....	42
參考文獻.....	44
附錄 A、TCC300 辨識相關結果.....	47



表目錄

表 3.1 使用 PDE 於快速 GMM 計算.....	17
表 3.2 使用 DGS 於快速 GMM 計算.....	18
表 3.3 快速部份排序法.....	22
表 3.4 聲學與統計式刪除法之合併.....	23
表 4.1 原始實驗測試結果.....	25
表 4.2 改良後的基準實驗結果.....	26
表 4.3 利用 Treebank 語料估計各部份占辨識系統時間百分比.....	27
表 4.4 利用最近點概算的實驗結果.....	28
表 4.5 Average Distortion 最高的前十維.....	30
表 4.6 利用最近點概算配合上 FCR 與 BMP 方法辨識結果.....	31
表 4.7 利用單連語言模型預查來快速建立雙連語言模型預查的辨識結果.....	32
表 4.8 每兩個音框延展一次詞轉移的辨識結果.....	33
表 4.9 每三個音框延展一次詞轉移的辨識結果.....	33
表 4.10 每四個音框延展一次詞轉移的辨識結果.....	34
表 4.11 每五個音框延展一次詞轉移的辨識結果.....	34
表 4.12 經由加速演算法後各部份占辨識系統時間百分比.....	35
表 4.13 利用 TCC300 語料估計各部份占辨識系統時間百分比.....	36
表 4.14 TCC300 語料的基準實驗結果.....	36
表 4.15 經由加速演算法後 TCC300 的辨識結果.....	37
表 4.16 經由加速演算法後 TCC300 各部份占辨識系統時間百分比.....	38
表 A.1 TCC300 語料的基準實驗結果.....	48
表 A.2 TCC300 利用最近點概算配合上 FCR 與 BMP 方法辨識結果.....	48
表 A.3 TCC300 單連語言模型預查來快速建立雙連語言模型預查的辨識結果..	48
表 A.4 TCC300 每兩個音框延展一次詞轉移的辨識結果.....	48
表 A.5 TCC300 每三個音框延展一次詞轉移的辨識結果.....	49
表 A.6 TCC300 每四個音框延展一次詞轉移的辨識結果.....	49

圖目錄

圖 1.1 以動態規劃法進行語音辨識.....	2
圖 1.2 以有限狀態機作語音辨識為例.....	3
圖 1.3 兩階段語音辨識示意圖.....	4
圖 1.4 兩階段式 GMM 計算示意圖.....	4
圖 2.1 大詞彙連續語音辨識架構簡圖.....	6
圖 2.2 HMM 示意圖.....	8
圖 2.3 (A) 線性詞典範例.....	10
圖 2.3 (B) 樹狀詞典範例.....	10
圖 2.4 樹狀詞典與雙連語言模型例圖.....	11
圖 2.5 樹狀詞典與雙連語言模型預查分數例圖.....	12
圖 2.6 (A) 音節內 HMM 狀態轉移示意圖.....	13
圖 2.6 (B) 字元間 HMM 狀態轉移示意圖.....	13
圖 2.6 (C) 詞轉移的 HMM 狀態轉移示意圖.....	14
圖 2.6 (D) 樹狀詞典轉移示意圖.....	14
圖 3.1 語言模型查詢的架構圖.....	19
圖 3.2 語言模型預查暫存表.....	20
圖 3.3 雙連語言位置與範圍查詢表.....	20
圖 3.4 利用單連語言模型預查建立雙連語言模型預查的圖示.....	21
圖 4.1 Initial MFCC v.s. Average Distortion 圖示.....	29
圖 4.2 Final MFCC v.s. Average Distortion 圖示.....	29
圖 4.3 Silence MFCC v.s. Average Distortion 圖示.....	29
圖 4.4 基準實驗與經加速演算法後實驗的比較圖.....	35
圖 4.5 TCC300 基準實驗與經加速演算法後實驗的比較圖.....	37
圖 5.1 辨識系統介面展示圖.....	39
圖 5.2 辨識系統實際 Demo 圖之一.....	40
圖 5.3 辨識系統實際 Demo 圖之二.....	41

第一章 緒論

1.1 研究動機

大詞彙連續語音辨識(Large Vocabulary Continuous Speech Recognition; LVCSR)已發展三十多年，至今仍然是語音辨識的重要課題之一。在前人的研究成果累積以及電腦設備的進步推進之下，大詞彙連續語音辨識的相關研究有相當的突破。不僅是辨識的詞彙量增多，對聲學模型特徵參數發展出多種抽取方法，以及利用更複雜的語言模型，例如加入特定領域的 subword 與傳統 N 連語言模型(N-gram Language Model)結合來增進詞的覆蓋率 (Converge Rate)；甚至在其他相關課題，如語者調適(Speaker Adaptation)、雜訊消除(Noise Cancellation)、或是在辨識過程利用多階段的重計分(Rescoring)方法等，亦有相當多的研究與討論。

中文的大詞彙連續語音辨識與英文的語音辨識有不同之處，主要為中文的詞(word)並不存在很精確的邊界，導致可能發生斷詞錯誤的問題；另外，未在詞典收錄的詞(Out of Vocabulary; OOV)無法辨識，會影響最終的辨識結果。而聲學特性方面，中文發音是屬於一個字(Character)對應一個音節(Syllable)，所以聲學模式常使用音節模型(Syllable Model)或是右相關聲韻母模型(Right Context Dependent Initial/Final Model; RCDIF)。近年來由於電腦記憶體的增加，亦開始利用三連音素模型(Triphone)作為大詞彙連續語音辨識的聲學模型。

大詞彙連續語音辨識系統其中有三個課題是不斷地受人討論的：語音辨識率 (Recognition Accuracy)、辨識速度 (Recognition Speed)、以及記憶體的管理 (Memory Management)，此三者是在大詞彙語音辨識系統所需要做妥協(Trade off)。本論文著重於聲學模型、語言模型以及詞轉移三方面，利用演算法與程式最佳化來加快辨識速度，而控制辨識率的損失在可容許的範圍，並且將辨識系統設計成有彈性(Flexible)，可依據不同的應用，去讀取具文法規則(Grammar Rule)形式的語言模型，以建構語音辨識系統。

1.2 文獻回顧

語音辨識的目的為「將一段語音信號轉換成文字」，然而從系統實作角度去觀察，如何利用語音信號的各種特徵如聲學信號、頻譜、韻律等，配合上自然語言的文字、文法、語義等特性去找出最符合的文字解答，才是語音辨識技術的關鍵；概括而論，結合各項語音和文字資訊，設計出有效率的演算法來產生辨識解答，乃為語音辨識系統的核心內容。

語音辨識系統的設計軟體最廣泛被使用的為劍橋大學 S. J. Young 等人所發表的 HTK，所使用演算法名為 Token Passing 的演算法[1]，想法上使用維特比演算法(Viterbi Algorithm)，在 token 上紀錄回溯資訊，以便最後找出最佳辨識結果。

西元 2000 年前後，有關大詞彙連續語音辨識的系統演算法相繼被提出討論，常見的可分成以下三類：

(A) H. Ney 與 S. Ortman 所提出利用動態規劃法(Dynamic Programming)[2]實作大詞彙連續語音辨識，根據聲學模型、發音辭典和語言模型三者，以動態規劃搜尋出最佳辨識結果。此論文提出語言模型預查(Language Model Look Ahead; LMLA)演算法[3]，而相關研究陸續提出進一步的改良，如利用 Hash Table 加速 LMLA 查詢時間[4]，或是利用結合 Phoneme Look-ahead 和 Language Model Look-ahead 加速辨識時間[5]等。

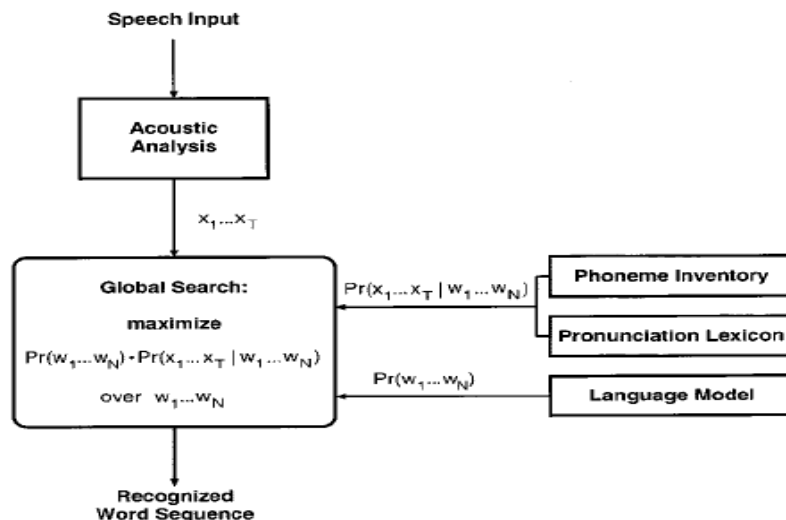


圖 1.1 以動態規劃法進行語音辨識(Copyright: H. Ney, et. al., 2000 [2])

(B) M. Mohri 與 M. Riley 提出利用有限狀態機(Weighted Finite State Transducers; WFSTs)[6]概念實作連續語音辨識系統，預先將聲學模型、發音辭典和語言模型三者建構在狀態

機上，以語音為輸入，文字辨識結果為輸出，建立起語音辨識系統。其優點為可以更複雜的語音相關資訊建立起有限狀態機，並且只需規定出適當的輸入以及輸出即可利用結合(Composition)方法與原本的有限狀態機結合成一個更複雜的有限狀態機，而無需更動整體系統的資料結構。然而建立起有限狀態機系統往往會因所需存入狀態數總量過多導致系統記憶體使用過於龐大，因此近年來有許多在辨識時做狀態結合以降低其數量至接近最佳值的演算法問題被討論[7-9]。

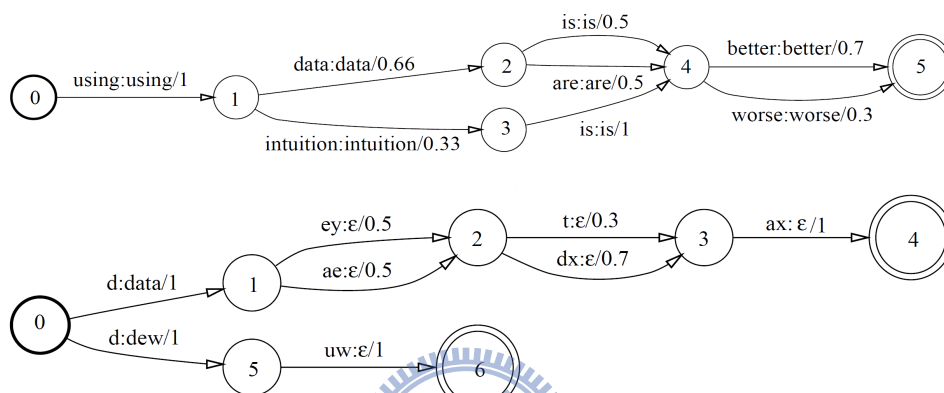


圖 1.2 以有限狀態機作語音辨識為例(Copyright: Mohri, Mehryar, et. al., 2000 [6])

(C) Asynchronous Stack Decoding 方法，是根據經驗搜尋法(Heuristics Search Method, known as A* principle)去實作出大詞彙連續語音辨識，近年相關研究較少[10]。

除了辨認演算法研究之外，尚有其他語音辨識系統細部分析與演算法改良等相關論文：

(A) 利用語音的知識來源(Knowledge Source)對辨識系統作 Two-pass 實行重新計算分數(Rescoring)動作。如利用中文聲調(Tone)[11-12]或是利用類神經網路(Artificial Neural Network; ANN)作屬性偵測(Attribute Detection)[13]等語音特性，對辨識系統重新計分動作。

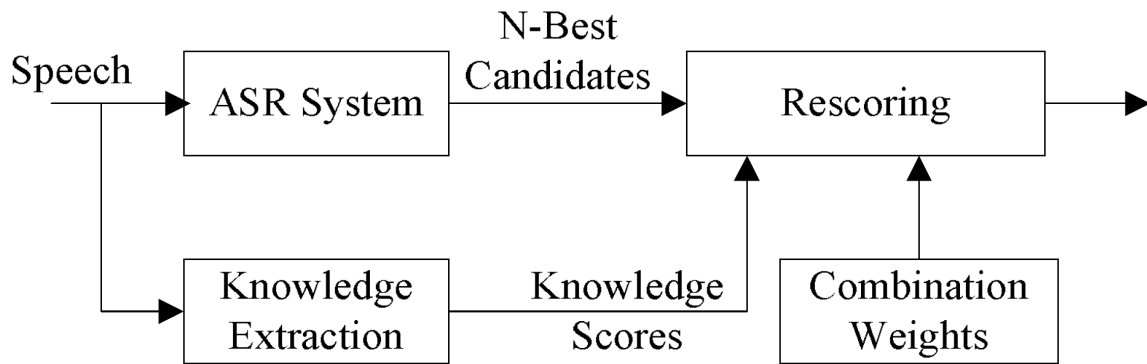


圖 1.3 兩階段語音辨識示意圖(Copyright: Jinyu Li, et. al., 2005 [13])

(B) 針對於如何快速計算 GMM Likelihood 以增進辨識速度方面，主要提出兩個方向：(a) 藉由相鄰音框的語音特性相近性質[14]，或將 GMM 做分群(Gaussian Clustering) [15]，進而減少計算 GMM Likelihood 的計算量；(b)利用兩階段式計算 GMM Likelihood，先使用粗略比對(Coarse Match)去除不可能的 hypothesis 之後再利用細部比對(Detail Match)去計算出真正 Likelihood 值[16-17]；利用發聲確認(Utterance Verification)作聲學快速比對(Acoustic Fast-match)[18]用來加快辨識速度。

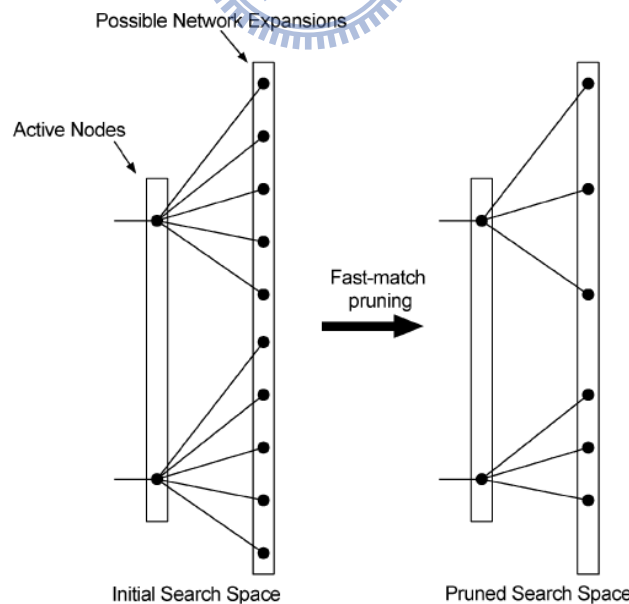


圖 1.4 兩階段式 GMM 計算示意圖(Copyright: Mohamed Afify, et. al., 2005 [18])

語言模型方面，除了 LMLA 相關演算法之外，對於壓縮詞典樹並降低語言模型系統記憶體的使用量也有相關討論[19][20]；或是藉由建立低階的 LMLA 再去部分更新高階的 LMLA 使得電腦 CPU 的運算量能夠降低[21]。

1.3 章節概要

本論文共分為六章：

- 第一章 緒論：介紹本論文之研究動機與相關文獻回顧。
- 第二章 語音辨識與相關演算法介紹：簡介大詞彙連續語音辨識的基本知識以及基礎常用演算法。
- 第三章 辨識系統加速演算法：介紹辨識系統各個部份的加速演算法。
- 第四章 實驗和分析：驗證前一章的加速演算法，並分析與探討實驗結果。
- 第五章 建構語音辨識系統：運用本論文的相關知識並建立一套可多功能應用與具有彈性之語音辨識系統。
- 第六章 結論與未來展望。



第二章 語音辨識與相關演算法介紹

2.1 背景知識

本章針對大詞彙語音辨識系統做背景知識說明，並介紹目前交通大學語音處理實驗室所使用的語音辨識系統。

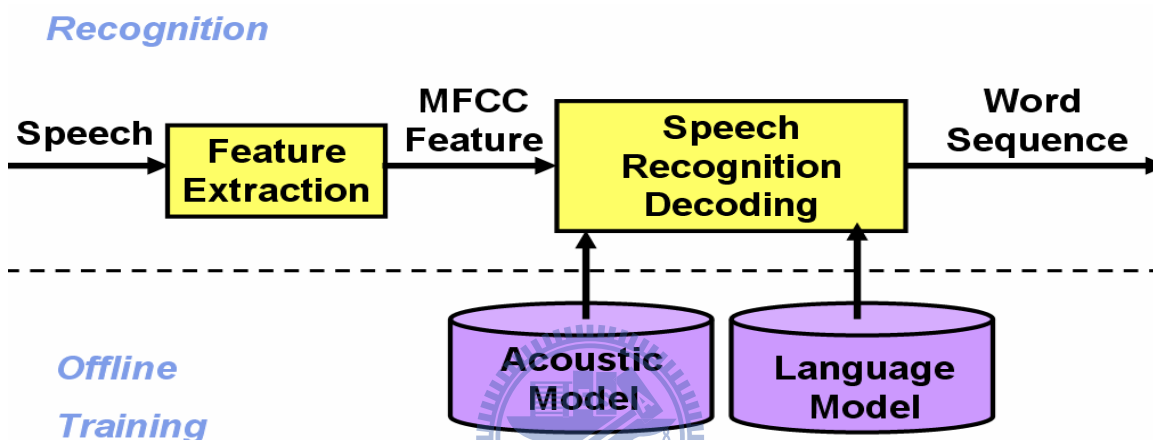


圖 2.1 大詞彙連續語音辨識架構簡圖

圖2.1為語音辨識系統的架構，在語音辨識中通常不直接將語音的類比訊號作數位取樣作語音辨識，由於語音的變化是緩慢且連續，因此取而代之作法為取一小段語音音框(Frame)，而相鄰的音框會有互相重疊的部份，並將音框抽取特徵向量(Feature Vector)來作語音辨識。常用來抽取的特徵向量參數為梅爾倒頻譜係數(Mel-Frequency Cepstral Coefficient; MFCC)。之後所得到的特徵向量經過辨識系統的解碼(Decoding)後即可得出辨識結果。

直覺上語音辨識系統即是「這段輸入語音訊號最像、最有可能為哪段字句」的問題，在數學上引進機率統計的概念，藉由統計式方法(Statistical Method)將“像”、“可能”轉換成機率的觀念來表示。如此即可將前面敘述的問題轉換成「找出輸入語音訊號對應機率最大的字句」的問題。由數學式子為表示：

$$\hat{W} = \arg \max_{w \in W} P(w|O) \quad (2-1)$$

其中 W 為所有可能的詞串(Word Sequence)， w 為集合 W 中的某一詞串； O 是輸入的語音(Utterance)， $P(w|O)$ 為在 O 發生情況下，詞串 w 的事後機率(Posterior Probability)。

式2-1可利用貝氏定理(Bayes' Theorem)將 $P(w|O)$ 進一步處理：

$$P(w|O) = \frac{P(O|w)P(w)}{P(O)} \quad (2-2)$$

$P(O|w)$ 代表詞串 w 產生語音 O 的可能(Likelihood)機率，一般採用機率分佈或機率模型來表示，在語音辨識系統中主要以聲學模型(Acoustic Model)描述； $P(w)$ 為詞串 w 的事前機率(Prior Probability)，若詞串 w 中含有 M 個詞 w_1, w_2, \dots, w_M ，則 $P(w) = P(w_1, w_2, \dots, w_M)$ 是 w_1, w_2, \dots, w_M 的聯合機率(Joint Probability)，在語音辨識系統中是以語言模型(Language Model)描述；另外由於 $P(O)$ 與 w 無關，故可將此項機率值由式2-1拿掉。整理後可得到式2-3：

$$\hat{W} = \arg \max_{w \in W} P(O|w)P(w) \quad (2-3)$$

實際語音辨識系統中，運用隱藏式馬可夫模型(Hidden Markov Model; HMM)建構聲學模型；而語言模型則是採用 N 連(N -gram)語言模型。下一節分別解釋聲學和語言模型與其演算法知識。

2.2 聲學模型 (Acoustic Model)

首先在建立聲學模型，首先對中文語音結構作簡單介紹：中文每個字的發音皆可對應一或多的音節(Syllable)，一般中文音節可歸類411類(不含Tone的資訊)，而每個音節又分為聲母(Initial)以及韻母(Final)的結合。本論文辨識系統時是採用音節模型(Syllable Model) 或右相關聲韻母模型(RCDIF Model)建構聲學模型。右相關聲韻母模型是利用右相關聲母模型 (Right Context Dependent Initial Model) 及前後文獨立韻母模型 (Context Independent Final Model)，共有100類聲母跟40類韻母所組成。

語音辨識中建立聲學模型應用最廣為由左至右的隱藏式馬可夫模型，此模型架構主要以狀態(State)和狀態轉移(Transition)所構成，HMM每次的轉移皆有轉移機率(Transition Probability)，每個狀態中亦有觀測機率(Observation Probability)。一般選擇高斯混合模型

(Gaussian Mixture Model; GMM)描述。本系統利用HMM去模擬Syllable Model時，每個音節利用八個狀態去描述；模擬RCDIF Model時，聲母利用三個狀態、韻母利用五個狀態去建構，因此每個音節皆由八個狀態組成；此外限制每個狀態的機率轉移規定只能前進至下一個或是停留在本身狀態。如圖2.2所示。經由數學運算後，驗證發現到不同轉移機率值的差距會遠小於HMM狀態之間的觀測機率，因此為了降低運算上的複雜度，則直接假設所有合法的轉移機率皆為相同機率值。故聲學模型分數即等於觀察機率的對數機率值(Log Likelihood)。

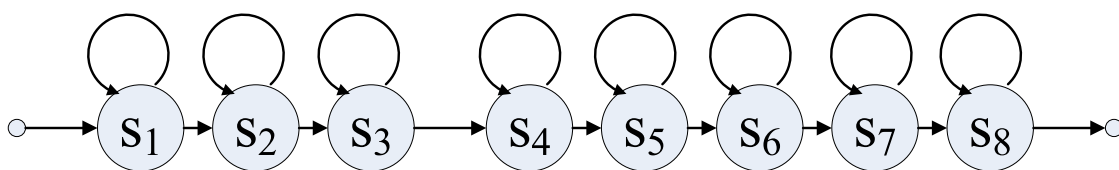


圖 2.2 HMM 示意圖

計算聲學模型時，先對輸入語音以音框為單位求取參數，本系統採用12維的梅爾倒頻譜參數 (MFCC)、12維的delta-MFCC、1維的delta-Energy、12維的delta-delta-MFCC以及1維的delta-delta-Energy，共計38維作為參數值；再計算HMM中的各個狀態的相似度 (Likelihood)。

$$\begin{aligned}
 p(\mathbf{x} | \lambda_s) &= \sum_k c_k N(\mathbf{x}; \mu_k, \Sigma_k) \\
 &= \sum_k c_k \cdot \frac{1}{\sqrt{(2\pi)^N |\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right) \quad (2-4)
 \end{aligned}$$

式2-4為利用GMM表示HMM裡狀態的相似度。其中 \mathbf{x} 為特徵向量， λ_s 為某一HMM狀態， $\mu_k, \Sigma_k, c_k, \Sigma_k$ 各是GMM的平均向量(Mean Vector)、變異數向量(Variance Vector)、Mixture Weight、以及共變異矩陣(Covariance Matrix)， N 為MFCC維度總數(=38)。而 p 為相似度，實際系統運作若將 p 機率值直接相乘，因小數互相乘積，導致機率值很快地過小而超過浮點數可表示的範圍(因underflow)而失真，故先將 p 取對數值後再做相加運算以避免此問題，則式2-4改寫如下：

$$\begin{aligned}
\log(p(\mathbf{x} | \lambda_s)) &= \log\left(\sum_k c_k N(\mathbf{x}; \mu_k, \Sigma_k)\right) \\
&= \log\left(\sum_k c_k \cdot \frac{1}{\sqrt{(2\pi)^N |\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right)\right) \\
&= \log\left(\sum_k C_k \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right)\right) \quad (2-5)
\end{aligned}$$

$$C_k = c_k \cdot \frac{1}{\sqrt{(2\pi)^N |\Sigma_k|}} \quad \text{is a constant for each density.}$$

實際系統運算中，共變數矩陣 Σ_k 常使用對角共變數(Diagonal Covariance)替代全矩陣(Full Matrix)以節省GMM運算量，最後整理可得到：

$$\log(p(\mathbf{x} | \lambda_s)) = \log\left\{\sum_k C_k \exp\left[\sum_{j=1}^N \frac{(x_k(j) - \mu_k(j))^2}{-2\sigma_k^2(j)}\right]\right\} \quad (2-6)$$

2.3 發音詞典 (Pronunciation Lexicon)

中文語音辨識在聲學模型得出聲學相似度分數後，必須藉由發音詞典對應到有意義的詞彙，最後才能輸出令人通曉瞭解的文句；發音詞典建構方式則會確立辨識系統中的搜尋空間(Search Space)。而資料結構大致可分為兩種類型：線性詞典(Linear Lexicon)和樹狀詞典(Tree Lexicon)。

線性詞典結構如下頁圖2.3(A)，每一個詞條皆獨立存入字串陣列中，其優點為該資料結構單純，作語音辨識時進入該詞條後即可立即找出相對應的語言模型分數，然而線性詞典缺點為需儲存龐大的HMM狀態使得記憶體浪費。樹狀詞典是以線性詞典作為基礎，進而採用樹狀結構來儲存詞典以節省儲存詞典的空間。將相同前綴(Prefix)的詞條合併並儲存在同一節點(Node)，次音節若有相同前綴即以同樣的作法類推。最後用鏈結串列(Linking List)的資料結構將所有詞條建構成一顆龐大的樹狀詞典，如下頁圖2.3 (B)。然而樹狀詞典最大缺點當在語音辨識時進入詞典時無法立即決定語言模型分數，一直要到詞尾(Word End)時才可確定為哪個詞條，對此問題之後可利用語言模型預查(Language Model Look Ahead;

LMLA)方法去提前預估語言模型分數。

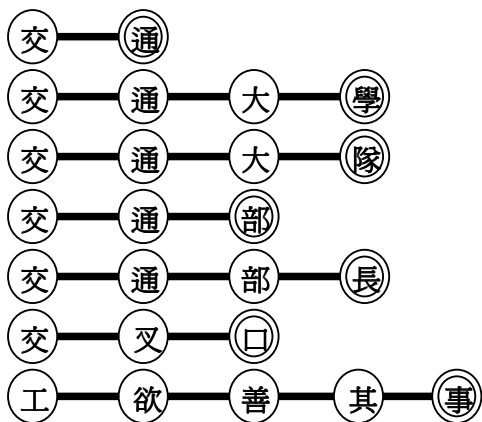


圖 2.3(A) 線性詞典範例

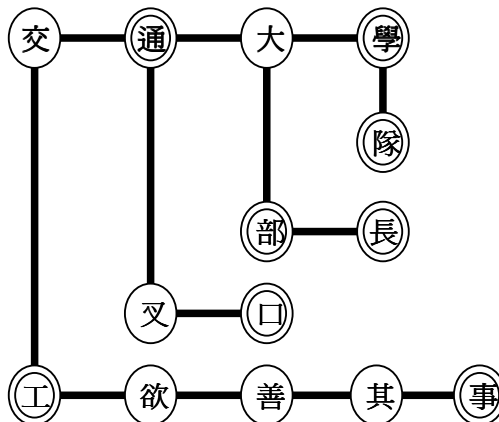


圖 2.3 (B) 樹狀詞典範例

2.4 語言模型 (Language Model)

語音辨識系統中，語言模型常使用 N 連語言模型來模擬文句的機率。 N 連語言模型運用 $N-1$ 階馬可夫假設($N-1$ Order Markovian Assumption)簡化，換言之該詞條的條件機率部份僅與前 $N-1$ 階前詞(Predecessor)有相關。因此語言模型機率 $P(w_1, w_2, \dots, w_M)$ 可簡化成下式：

$$P(w_1, w_2, \dots, w_M) = \prod_{i=1}^M P(w_i | w_1, \dots, w_{i-1}) = \prod_{i=1}^M P(w_i | w_{i-N+1}, \dots, w_{i-1}) \quad (2-7)$$

N 連語言模型優點為只需存 $N-1$ 個前詞歷史資訊，一般語音辨識系統中常用雙連(Bigram)語言模型或三連(Trigram)語言模型，然而使用 N -gram語言模型時會因訓練語料不足發生有未出現的詞串組合導致無法獲得 $P(w_i | w_{i-N+1}, \dots, w_{i-1})$ 的機率，此時利用後撤平滑化(back-off smoothing)方法[22]解決此問題，主要概念是採用低一階的語言模型機率乘上後撤加權值去調整此詞串的機率值，因此 $P(w_i | w_{i-N+1}, \dots, w_{i-1})$ 可改寫成下式。

$$P(w_i | w_{i-N+1}, \dots, w_{i-1}) = P(w_i | w_{i-N+2}, \dots, w_{i-1}) \times b(w_{i-N+2}, \dots, w_{i-1}) \quad (2-8)$$

有了語言模型機率分數，配合聲學模型機率分數會一起加回存活路徑的hypothesis。實際系統中會將語言模型機率分數乘上一個權重值後才會加入hypothesis，使得聲學與語言模型兩者機率分數值對於hypothesis的影響能夠平衡。由辨識結果可發現設定不同的語言模型權重值會導致辨識率與辨識速度有不同的差異。

2.5 語言模型預查(LMLA)

回到樹狀詞典的議題，當hypothesis要加上語言模型機率分數，必定當作詞轉移才能知道本身hypothesis是為哪個詞 w_i ，再由語言模型中求得該機率分數 $P(w_i | w_{i-N+1}, \dots, w_{i-1})$ ；以圖2.4為例：共有“交通”、“交通部”和“交通大學”三個詞(雙層圈代表詞尾)。當hypothesis進入詞首‘交’時，由於有三種詞的可能選擇，因此無法立即給于語言模型分數，即使進入‘通’仍無法判斷直到hypothesis從‘通’最後一個狀態離開時才能加上 $w_i =$ ‘交通’的語言模型分數。而‘交通部’和‘交通大學’最快需要進入‘部’和‘大’才能加入它們的語言模型分數。

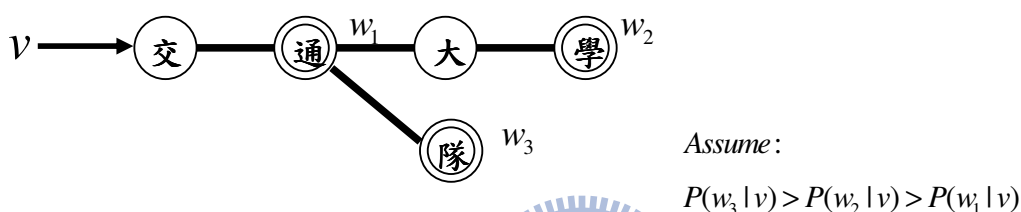


圖 2.4 樹狀詞典與雙連語言模型例圖

因此若能對剛進入各個字首或樹狀詞典分支點時，提供一個適當的語言模型分數給hypothesis，憑藉該可靠的語言模型分數資訊，盡早刪除語言模型機率分數較低的hypothesis，降低整體搜尋空間。如圖2.4為例：在進入詞首“交”便會給于部份的語言模型分數，之後進入到“通”以此類推，直到最後離開詞尾時才補償回原本該詞 w_i 完整的語言模型分數。

而語言模型預查的基本概念相當的直觀，當hypothesis進入樹狀詞典中的字元節點 c 上，對於此字元 c 為根節點的所有子樹內的詞彙，尋找出一個最高語言模型機率詞的分數，代表進入此節點後最高分數的上限。若此時hypothesis仍被詞轉移刪除演算法(Word End Pruning)刪除，代表該子樹內的詞彙語言模型分數皆過低；反之若此hypothesis存活下來，代表該子樹至少有一個詞其語言模型分數足以保護該hypothesis。最後將所有節點皆計算出對應的最高分數，並在詞尾加回原本詞彙的語言模型機率分數。

以上頁圖2.4為例：當hypothesis進入“交”時得到最大的語言模型預查機率分數為 $P(w_3 | v)$ ；而進入“通”時若此hypothesis需要作詞轉移則語言模型分數等於 $P(w_1 | v)$ ；再往下走分別為“大”和“部”，語言模型預查機率分別為 $P(w_2 | v)$ 以及 $P(w_3 | v)$ 。如圖2.5所示。

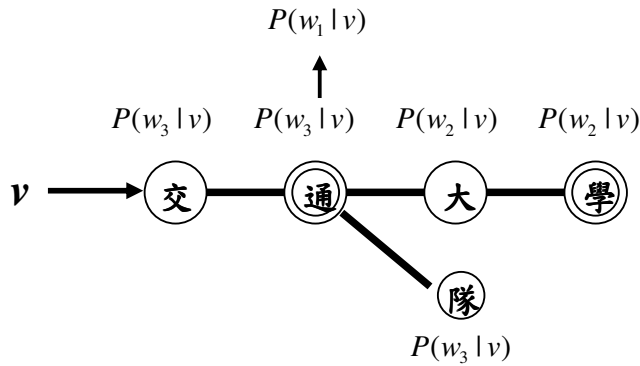


圖 2.5 樹狀詞典與雙連語言模型預查分數例圖

一般語音辨識系統採用單連或雙連語言模型預查方法。單連語言模型預查分數優點為單連語言模型只與本身詞的機率分數有關，而無需前詞資訊，故在辨識前建立一顆單連語言模型預查樹，辨識期間即可直接查詢該預查樹的資訊得到預查分數。實做上 N 連語言模型可配合單連語言模型預查使用，只需在詞尾補償回原本的 N 連語言模型分數即可。

然而 N 連語言模型使用單連語言模型預查會無法獲得某一節點的真正機率分數上限(由於無確切的 N 連語言模型分數資訊)，若設定詞轉移刪除演算法門檻值過高有可能導致某些在單連語言模型預查分數較低的詞將永遠被刪除掉。因此採用 N 連語言模型預查替代單連語言模型預查來改良此問題。做 N 連語言模型預查在辨識途中需使用時才動態建立，每當前詞不同時，則必須重新建立一顆預查樹，往後遇到相同前詞時可直接查表獲得語言模型預查分數。然而實際系統中因記憶體有限，故無法毫無限制地儲存，因此會設定儲存值的上限門檻，當儲存筆數達到上限時便移除一筆舊資料。

需要移除舊資料時，移除方式可利用不同方法去實踐。最簡易的方式是採用佇列(Queue)去實現，當佇列填滿時則刪除最舊的一筆前詞資料，即為先進先出(First in, First out; FIFO)的概念。另外可考慮更進一步的方法，如移除最久以前被用過的一筆資料或是雙連機率個數最少的資料等相關有意義的方法，亦能減少語言模型預查重複計算的次數。另外對於儲存個數上限多寡會直接影響到重複建立的頻率，若上限設定過低，則越容易重複建立相同前詞的預查樹。

以上為語音辨識各層架構的簡介，仍需配合維特比演算法(Viterbi Algorithm)和刪除演算法(Pruning Algorithm)控制整體搜尋空間。以下簡略介紹此兩種演算法的運作原理。

2.6 維特比演算法(Viterbi Algorithm)

根據聲學模型以及語言模型所組成的Trellis，同一個時間點當有兩條狀態轉移路徑重合至同一個狀態，可使用維特比演算法的概念保留機率分數較高的hypothesis即可。而根據HMM的轉移的狀況不同可分為“音節內HMM狀態轉移”、“字元間HMM狀態轉移”以及“詞轉移HMM狀態轉移”三種情況。

圖2.6(A)為音節內的HMM狀態轉移圖，其中兩條不同來源的路徑都連結到同一個 S_{t+1} 狀態，由於 $t+1$ 時間的觀察機率皆在同一個狀態，狀態的轉移機率、和 $t+1$ 時間以後產生的路徑皆完全相同，唯一差異點在 t 時間以前兩條路徑所累積的機率分數，因此運用維特比演算法概念保留累積分數較佳路徑。而圖2.6(B)為字元間HMM狀態轉移圖，其中有箭頭方向為詞內某字元的第八個狀態，下一個時間點除了會回到自己本身狀態之外，亦產生進入下一個字元的第一個狀態的hypothesis。

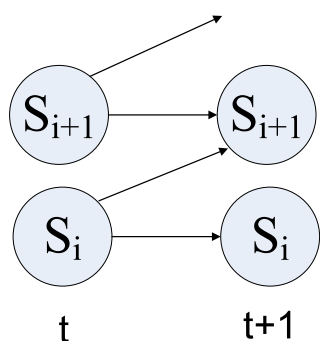


圖2.6(A)音節內HMM狀態轉移示意圖

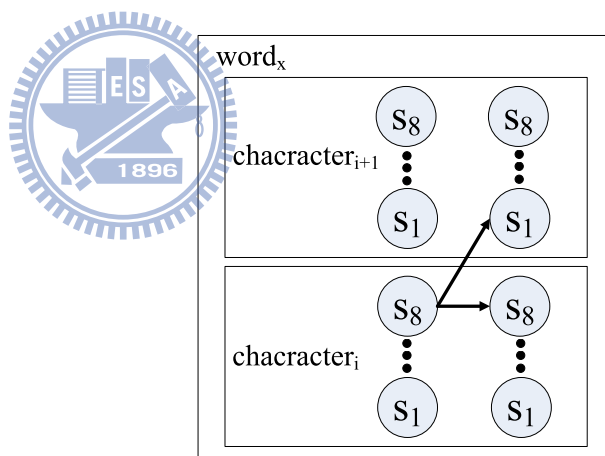


圖2.6(B)字元間HMM狀態轉移示意圖

若第八個狀態所在的字元已是一個詞的詞尾，此狀態除了回到本身狀態之外，會延展出新的詞，換言之即進入新詞的詞首第一個狀態。由圖2.6(C)可發現此時hypotheses會瞬間增加許多，以大詞彙詞典包含六萬詞為例，若採用線性詞典會產生六萬個不同的hypotheses，即使將線狀詞典合併成樹狀詞典仍會產生數千種hypotheses。可將詞轉移情況思考成圖2.6(D)，各個詞尾最後一個狀態皆連接至一條空狀態(Null State)，此空狀態代表所有詞首第一個狀態，即為重新進入樹狀詞典(Reentrant)。這種快速增加的情況，可藉由語言模型預查以及詞轉移刪除法篩選分數較佳的hypotheses。相對地所耗費的運算量也會大幅增加。之後會改善辨識系統在此處耗費的運算量。

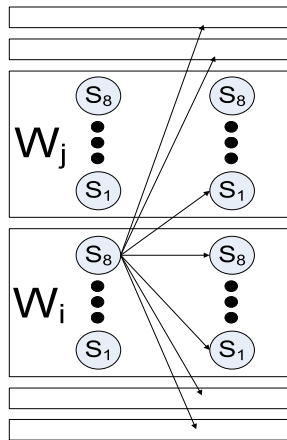


圖2.6 (C) 詞轉移的HMM狀態轉移示意圖

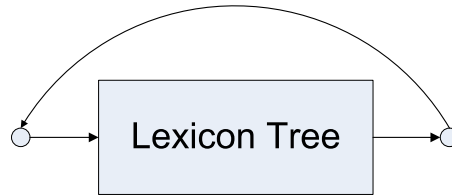


圖2.6 (D) 樹狀詞典轉移示意圖

2.7 刪除演算法(Pruning Algorithm)

進行語音辨識中，hypothesis的數量會隨著音框前進而數量會變得相當龐大。尤其考慮 N 連語言模型時增加的速度更為快速，因此系統實做上利用刪除演算法將分數較低的hypothesis刪除。原理為辨識途中分數落後太多的hypothesis能再度將分數重新超前的機率極小，因此在每個時間點留下足夠數量的hypothesis即確保與完全不刪除的辨識結果接近甚至相同。且可大幅降低辨識的搜尋空間，縮短辨識所耗費的時間。因此如何設定各種刪除演算法優化整體系統是重要且必要的一個步驟。

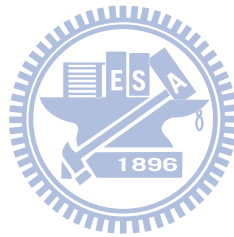
本系統採用三種刪除方法：聲學刪除法(Acoustic Pruning)、統計式刪除法(Histogram Pruning)以及詞轉移刪除法(Word End Pruning或Language Model Pruning)。

執行聲學刪除法，先找出當時音框所有hypotheses最高分的hypothesis，再設定刪除分數的門檻(Threshold)，若音框裡hypothesis與最高分差距超過門檻值，則將此hypothesis刪除，一般稱此演算法為光束搜尋演算法(Beam Searching Algorithm)。聲學刪除法優點為只需對所有hypotheses作一次整體的刪除即篩選出分數高的hypotheses，然而缺點為當每個hypotheses分數過於接近則無法有效刪除，使得無法確保系統的搜尋空間以及辨識時間的範圍。

統計式刪除法作法與聲學刪除法相似，運作原理為設定一個存活總數為 N 值，讓每個時間點僅留下分數最高前 N 個hypotheses於下一個音框使用。其刪除法優點為可控制hypotheses最大總量，然而缺點為需作排序(sorting)的動作時使得系統的運算量增加。在實

做上兩種方法可配合一起使用，先採用聲學刪除法去除分數較低的hypotheses，當剩餘hypotheses仍過多時再採用統計式刪除法留下固定個數的hypotheses。

詞轉移刪除法僅在詞轉移時才作用，當詞轉移時，因單一hypothesis會延展數千個新的hypotheses導致計算量大增，因此藉由語言模型預查配合詞轉移刪除法的機制，可刪除語言模型分數較低進而篩選出較佳的hypothesis。降低詞轉移所快速增加的hypotheses。



第三章 即時辨識系統演算法

3.1 基本概論

前一章已介紹大詞彙語音辨識系統相關知識以及演算法，此章再針對系統各個部份作深入探討，找出加速辨識系統的演算法。如第一章所提至，一個語音辨識系統有三個部份需作妥協：辨識速度、辨識率、與系統記憶體的使用量。本系統開發介面為個人電腦並非嵌入式版(Embedded)，故記憶體的考量可放寬鬆一些，因此可將系統所需的資訊，事前存入記憶體中換取辨識時的計算量；另外在容許損失少許辨識率範圍的情況，計算數學運算式可使用概算(Approximation)去近似，亦能降低計算量。最後針對電腦程式語言作最佳化的技巧，如使用整數運算(Integer Operation)取代浮點數計算(Floating Operation)以及迴圈展開(Loop Unrolling)技巧皆可節省系統程式的運算。以下為分別詳述各部份的演算法。



3.2 聲學模型

對於辨識系統而言，計算聲學模型GMM相似度機率值占整體系統時間高達約30~70%左右[26]，因此如何快速地估計GMM相似度機率值成為加速辨識系統重要課題之一。在快速計算高斯混合模型(以下簡稱GMM)方面，本論文參考Bryan L. Pellom等人在2001年所提出的方法[23]。回顧第二章的2-6式：

$$\log(p(\mathbf{x} | \lambda_s)) = \log \left\{ \sum_k C_k \exp \left[\sum_{j=1}^N \frac{(x_t(j) - \mu_k(j))^2}{-2\sigma_k^2(j)} \right] \right\} \quad (2-6)$$

利用最近點概算(Nearest-Neighbor Approximation)方法，從所有mixture中找出相似度機率最高的mixture得出的概算取代計算所有mixture總和值。因此式2-6可改寫成如下式。

$$\log(p(\mathbf{x} | \lambda_s)) \approx \max_{1 \leq k \leq K} \left(\log(C_k) - \frac{1}{2} \sum_{j=1}^N \frac{(x_k(j) - \mu_k(j))^2}{\sigma_k^2(j)} \right) \quad (3-1)$$

由式3-1的概算法，可大幅降低計算量。因為 $\log(C_k)$ 是一固定值可在辨識前計算並

存入記憶體備用，而 $\frac{(x_k(j) - \mu_k(j))^2}{\sigma_k^2(j)}$ 此項計算結果必為一正值，故 $\log(p(\mathbf{x}|\lambda_s))$ 等於

$\log(C_k)$ 減去 MFCC 每項 $\frac{(x_k(j) - \mu_k(j))^2}{\sigma_k^2(j)}$ 值時會越減越小，故實做上可儲存已計算的

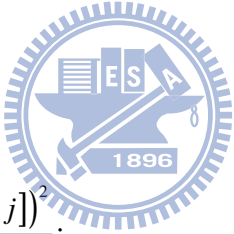
mixture 中最大值，爾後計算其他 mixture 時，當其 mixture 已小於最大值，則停止計算剩餘 MFCC 的維度，以節省 GMM 的計算量。這種方法稱為部份距離刪除法 (Partial Distance Elimination; PDE)，演算法如表 3.1。另外可搭配迴圈展開 (Loop Unrolling) 方法對於表 3.1 的 for 迴圈作程式最佳化方式即能將削減計算量。

表 3.1 使用 PDE 於快速 GMM 計算

```

int sum = 0, Mixturebest = -∞;
for all k Mixtures
{
    sum = log(Ck);
    for all MFCC j Dimensions
    {
        sum =  $\frac{(Feature[j] - Mean[j])^2}{Variance[j]}$ ;
        if(sum < Mixturebest)
            break;
    }
    if(sum > Mixturebest)
        Mixturebest = sum;
}
return Mixturebest;

```



此外如何提前找出 *bestMixture* 以及儘早刪除非最大值的 Mixture，則提出兩種方法：最佳 mixture 的預測 (Best Mixture Prediction; BMP) 以及特徵參數重新排序 (Feature Component Reordering; FCR)；前者是將前一刻時間在該狀態所得到最佳的 mixture 套用於現在時間點，先計算該 mixture，將它設為暫定 *bestMixture*，之後再去計算剩餘的 mixture；後者是藉由 offline 測試過後，將 MFCC 影響相似度機率值大小的維度重新排序後，讓影響成份較大的維度排序在前，以方便儘早刪除非最大值的 mixture。在下一章會針對這些作法做進一步的分

析與實驗。

表3.2 使用DGS於快速GMM計算

```
compute  $Mixture_{BMP}$ ;  
 $Mixture_{approx} = Mixture_{best} = Mixture_{BMP}$ ;  
for all  $k$  Mixtures except  $Mixture_{BMP}$   
{  
  int  $sum = \log(C_k)$ ;  
  for all MFCC  $j$  Dimensions  
  {  
     $sum = \frac{(Feature[j] - Mean[j])^2}{Variance[j]}$ ;  
    if( $sum < Mixture_{best}$ ) break;  
  }  
  if( $j > Q_{thresh}$ ) //  $Q_{thresh}$  is a threshold.  
  {  
    complete calculate all MFCC Dimensions to  $sum$ ;  
     $Mixture_{approx} = \log(\exp(Mixture_{approx}) + \exp(sum))$ ;  
  }  
  if( $sum > Mixture_{best}$ )  $Mixture_{best} = sum$ ;  
}  
return  $Mixture_{approx}$ ;
```

這兩年亦有學者針對此最近點概算法作相關討論與改進[24]，指出使用機率最大的mixture的分數僅占計算所有mixture分數的85%，因此該GMM分數的解析度(Resolution)不足，易使整體辨識率受到衰減。故提出動態高斯選擇(Dynamic Gaussian Selection; DGS，演算法如表3.2)合併前述BMP和FCR的方法，利用經驗知識(Heuristic Knowledge)可在計算途中設定不同門檻定義值而找出數個較佳mixture加入相似度機率值，讓得到的數值相似度更接近原本計算所有的mixture的相似度機率值。

3.3 語言模型

語言模型預查查詢(LMLA lookup)和建立兩者都是主要都採取以記憶體換取計算時間的方法，本系統是利用雙連語言模型和雙連語音模型預查作語音辨識，圖3.1為雙連語言模型查詢與建立預查資訊的完整架構圖：

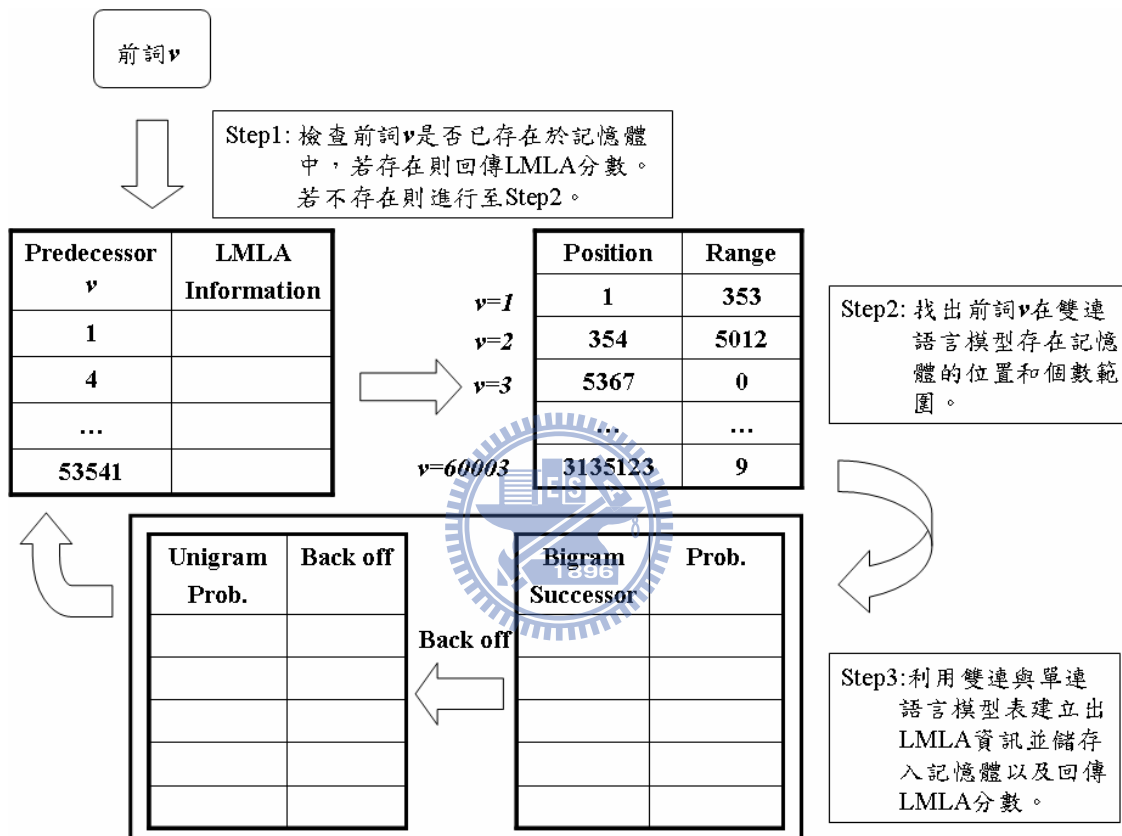


圖 3.1 語言模型查詢的架構圖

圖3.1架構圖主要概念是藉由語言模型機率分數去建立預查資訊並將所得到LMLA分數資訊儲存於記憶體裡供往後重複查詢使用，直到儲存上限到達門檻值再運用刪除機制清除。該架構可便於說明辨識期間，需要LMLA分數時能夠快速攫取(Fetch)與建立。以下針對此架構各個部份作介紹。

Predecessor v	LMLA Information
1	
4	
...	
53541	

圖 3.2 語言模型預查暫存表

圖3.2為語言模型預查暫存表。因辨識時在一段固定時間內，相同前詞的LMLA分數會持續被使用，因此當建立該前詞的LMLA資訊時則將它存入暫存表記憶體中，爾後若需要該前詞的LMLA分數則可直接攫取而無須重新建立計算預查資訊。對於暫存表需要存多少前詞資訊才足夠，本實驗室李庚達[25]的實驗成果發現存100筆前詞資訊與存1000筆前詞資訊，兩者在辨識時間上約差距3%，然而後者所需記憶體卻比前者要多約500MB。因此在設計上採用100筆前詞資訊的暫存表即足夠辨識系統的查詢。

當暫存表資訊已存滿必須移除資訊時，最簡易方式是採用佇列方法移除，此外藉由移除最久沒被使用的前詞，或移除機率最低的前詞亦可有效地管理暫存表、減少重複建立相同前詞。

	Position	Range
$v=1$	1	353
$v=2$	354	5012
$v=3$	5367	0

$v=60003$	3135123	9

圖 3.3 雙連語言位置與範圍查詢表

需建立前詞的LMLA資訊時，是憑藉著辨識前已存在記憶體的單連與雙連語言模型機率去建立；雙連語言機率模型分數是依序前詞順序將該前詞所有雙連機率分數存於記憶體中，然而每個前詞的雙連語言模型機率並非為固定個數，故需另外建立雙連語言位置與範圍查詢表，如圖3.3所示，由該表可迅速對應該前詞的雙連語言模型機率位置與範圍，配合單連語言模型機率資訊建立出該前詞的預查資訊。此外建立過程會花費許多的加減運算量，因此可將語言模型分數乘上某一倍數，使浮點數運算轉換成整數運算，簡化建立預查

分數的運算量。再與聲學模型分數作結合時，除上相同倍數的語言模型權重值即能恢復原本的語言模型機率分數。

另外，利用offline預先建立的單連語言模型預查資訊，節省辨識時所需建立的雙連語言模型預查資訊的計算量，主要觀念為建立雙連語言模型預查樹時，若該前詞的雙連語言模型機率個數較少，即直接套用事前已建立的單連語言模型預查資訊作為基礎，再針對該前詞的少量雙連語言模型機率去更新(Update)預查資訊，節省建立預查時的計算量，該作法近似[21]。圖3.4為演算法的基本概念。

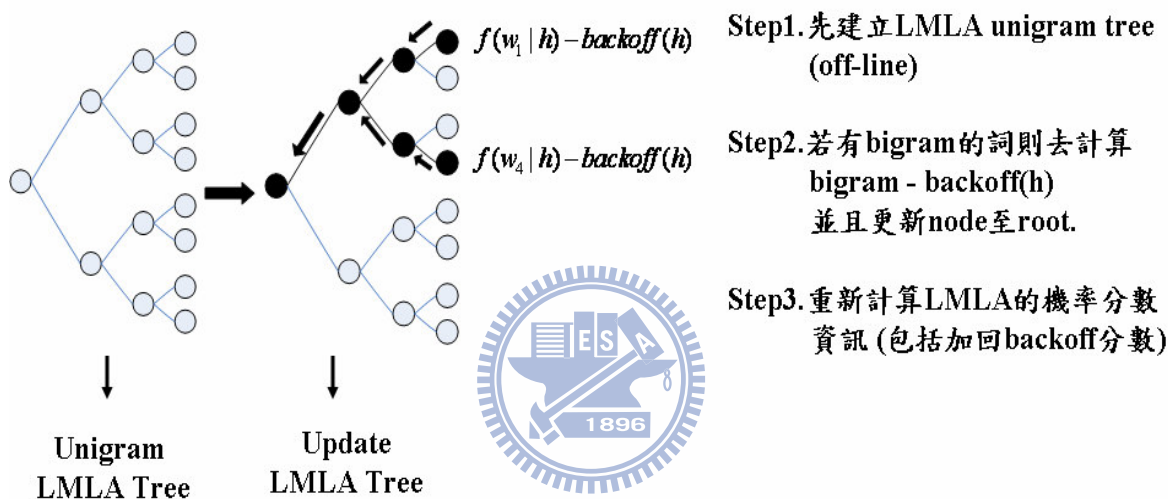


圖 3.4 利用單連語言模型預查建立雙連語言模型預查的圖示

3.4 刪除演算法與其他加速法

刪除演算法方面，前章有提至統計式刪除法最大的缺點為必須花費計算量於排序上，然而事實上，統計式刪除法為只保留前N筆機率最高的資料，無需對全部資料作大小完整排序，因此作排序時只需確保找出前N筆最高機率的hypotheses即可。實做系統上採用以快速排序法(Quick Sort)為基礎改良的排序法，稱為快速部份排序法(Quick Partial Sort)如表 3.3；主要概念為僅做半邊的快速排序法，排序出前N筆的hypotheses的資料，而無大小排序的順序。

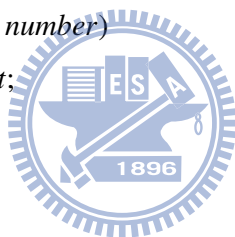
表 3.3 快速部份排序法

```
input : All of hypotheses;
assign Histogram number
int i = first position of hypotheses;
int j = last position of hypotheses;
function QuickPartialSort(int i, int j)
{
    To do Quicksort for all of hypotheses;
    Check final index after Quicksort;
    if(index > Histogram number)
        to do QuickPartialSort(int i, int index);
    else(index < Histogram number)
        to do QuickPartialSort(int index+1, int j);
    else
        return The former Histogram number of hypotheses.
}
```

系統實做上，聲學刪除法與統計式刪除法各有其優缺點，因此本辨識系統採取兩者並行的方法，先以聲學刪除法對於所有hypotheses進行刪除，若一旦發現存活的hypotheses數量過多時則再利用統計式刪除法留下前N筆機率最大的hypotheses。表3.4為刪除演算法之實踐。

表 3.4 聲學與統計式刪除法之合併

```
input: All of hypotheses;
int Survivor = 0;
function pruning()
{
  for all of hypotheses
  {
    if (hypothesis < Beam threshold)
      to do pruning.
    else
    {
      hypothesis can survive;
      Survivor ++;
      if (Survivor > Histogram number)
        to do QuickPartialSort;
    }
  }
}
```



隨著hypotheses數量而對於聲學、統計式的設定門檻可設計出不同應對的刪除法。實驗結果發現不同設計概念會使得辨識時間與辨識率的些許差距。

最後在語音辨識過程中，詞尾會延展出數千個新詞，導致大幅增加計算量和辨識時間。因此實做上，假設相信在鄰近數個音框所延展的詞幾乎相同，故可採用數個音框才容許做一次詞轉移取代每個音框時間點都允許詞轉移，雖然該方法會造成辨識率有些許衰減影響，卻能節省詞轉移所產生的龐大運算量。

第四章 實驗與分析

4.1 實驗準備

本實驗第一部份以Treebank作為測試語料，該語料由一位專業的女性播音員讀稿錄製而成；Treebank的前380個音檔是訓練語料，從編號397號的音檔開始，選取相差13號的音檔作為測試語料，共有60個音檔，總音長為0.437小時(1574.71秒)，共計有5681個字元，組成3109個詞。而Treebank的HMM模型為本實驗室先前由HTK toolkit[26]所訓練的右相關聲韻母模型(聲母為三個狀態，韻母為五個狀態)，每個模型的mixture數量並不固定，依照資料的需要而有所不同，最高與最低的數量為101與26 mixtures，平均為66.81個，short pause為129 mixtures。

第二部份使用TCC300中的成功大學所錄製的116個音檔作為測試語料，共計有十個語者(男女各半)，共13160個字元，總音長為1.01小時(3636.563秒)。TCC300的HMM模型亦為由HTK toolkit所訓練出來的音節模型(每個音節使用八個狀態)，每個模型的mixture數量亦因資料需要而不固定，最高與最低的數量各為22和3個mixtures，平均為9.97個，short pause則為64 mixtures。此部份主要目的為測試本辨識系統對於非特定語者做加速快演算法後辨識速度以及辨識率會有何影響。

所有實驗的聲學模型的狀態皆採用38維MFCC建立GMM，以及使用相同樹狀詞典，詞彙數量皆為六萬詞；辨識時間皆在同一台電腦上做測試，CPU型號為Intel® Core™2 Extreme X9650，3.00GHz，內裝8G記憶體，作業系統為Microsoft® Windows Server™ 2003 Standard Edition SP2。

最後實驗結果以辨識率和辨識時間兩方面為重心，並搭配李庚達[25]論文的相關實驗結果為參考比較。

4.2 基準實驗測試

表4.1為李庚達所做出最佳的辨識率以及辨識時間，語言模型權重為13，每個音框能夠留下的詞數固定為6個，並僅採用聲學刪除法。實驗結果可發現當Beam Width的門檻過於

寬鬆時，在辨識期間會產生大量hypotheses存活導致搜尋空間過大、辨識時間拉長；另外一方面當Beam Width小於125時，辨識率會大幅衰減，故不列入測試考量，往後實驗僅列出Beam Width從125至250的實驗結果。

表4.1原始實驗測試結果(李庚達,2008[25])

Beam Width	Acc(%)		Real-time(s)	RTF
	syllable	character		
50	13.22	7.75	162.404	0.103
75	53.95	35.93	319.859	0.203
100	85.2	67.91	548.629	0.348
125	92.94	82.56	792.332	0.503
150	93.91	84.81	1213.238	0.77
175	94.09	85.07	1605.7	1.02
200	94.19	85.39	2524.262	1.603
250	94.12	85.16	4460.558	2.833

本論文之辨識系統除了多考慮利用適當的統計式刪除法配合聲學刪除法之外，其餘包括語言模型權重、每個音框所能留下的詞數等相關系統設定都與李庚達學長最佳設定相同。基準實驗測試先將語言模型機率分數從浮點數運算量化成整數運算；另外針對原本系統的建立語言模型預查的方法和統計式刪除法的排序法作微幅修正，成為基準實驗(Baseline)。

表 4.2 改良後的基準實驗結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF
		syllable	character		
125	NoUse	93.61	82.42	772.771	0.491
125	4000	93.61	82.42	766.325	0.487
125	5000	93.61	82.42	770.642	0.489
150	NoUse	94.68	84.69	1071.557	0.68
150	5000	94.72	84.62	1063.786	0.676
150	6000	94.68	84.6	1068	0.678
175	NoUse	94.7	84.83	1463.676	0.929
175	5000	94.68	84.65	1358.317	0.863
175	6000	94.74	84.91	1378.997	0.876
175	7000	93.14	83.4	1378.206	0.875
175	8000	94.84	84.93	1422.44	0.903
200	NoUse	94.37	84.05	2000.164	1.27
200	6000	94.84	85	1646.949	1.046
200	7000	94.86	85	1699.834	1.079
200	8000	94.82	85.02	1723.557	1.095
200	9000	94.91	85.25	1975.928	1.255
225	6000	94.74	84.6	2020.621	1.283
225	7000	94.81	84.53	2144.188	1.362
225	8000	94.84	84.86	2169.373	1.378
225	9000	94.86	84.86	2314.754	1.47
250	6000	94.79	84.7	1977.412	1.256
250	7000	94.86	84.7	2312.175	1.468
250	8000	94.82	84.76	2396.07	1.522
250	9000	94.94	84.81	2248.758	1.428

由表4.2實驗結果，最高的辨識率等於85.25%，與表4.1的最佳辨識率85.39%有微幅的差異，其原因為微幅修正語言模型預查的機制所導致。另外可發現採用統計式刪除法配合聲學刪除法使用，可限制每個音框所存活hypotheses的最大總數，讓整體的搜尋空間不會無限的擴張。相對於僅使用聲學刪除法，當光束門檻設定過寬時，每個時間點皆會留下過多的hypotheses，使得辨識時間拉長。另外利用整數運算取代浮點數運算確實能改善計算時間，由表4.2的辨識結果與表4.1作比較可發現在辨識時間上有所加快。最後由於表4.2的相關數據數量過多，因此僅針對於數個辨識率最佳的數據作延續實驗，觀察是否經過一系列的加速演算法後會對辨識速度與辨識率產生何種影響。

針對整體語音辨識系統各個部份所花費的時間百分比做進一步的分析。使用所有Treebank測試音檔所做出的結果如表4.3。GMM的機率分數計算約占整體系統38%的辨識時間；另外維特比演算法(包括狀態轉移、詞轉移和語言模型預查)占辨識時間57%左右，其中建立和得到語言模型預查機率分數與詞尾作詞轉移都約占20%，所以針對這三部份做改善，則有助於整體辨識速度的上升。

表 4.3 利用 Treebank 語料估計各部份占辨識系統時間百分比

Viterbi	Likelihood	Pruning	Else	LMLA	Word End
57.16%	38.21%	2.48%	2.46%	20.24%	20.55%

4.3 聲學模型

$$\log(p(\mathbf{x} | \lambda_s)) \approx \max_{1 \leq k \leq K} \left(\log(C_k) - \frac{1}{2} \sum_{j=1}^N \frac{(x_t(j) - \mu_k(j))^2}{\sigma_k^2(j)} \right) \quad (3-1)$$

首先利用3-1式的最近點概算(Nearest-Neighbor Approximation)方法取代計算所有高斯混合模型並配合部份距離刪除法(Partial Distance Elimination; PDE)提早剔除非最大值機率的mixture，以及迴圈展開(Loop Unrolling) 程式最佳化方法減少計算for迴圈數量。相關實驗結果如下表：

表 4.4 利用最近點概算的實驗結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	93.8	82.5	593.763	0.377	77.48%
150	5000	94.7	84.76	849.081	0.539	79.82%
175	6000	94.79	84.93	1131.925	0.719	82.08%
200	6000	94.93	85.2	1345.885	0.855	81.72%
200	9000	94.98	85.46	1477.593	0.938	74.78%
225	8000	94.91	84.93	1674.301	1.063	77.18%

$$\text{Ex. time} = \frac{\text{Proposed Recognition Time}}{\text{Baseline Recognition Time}}$$

表4.4實驗結果可發現，系統的辨識速度明顯的提昇，比起表4.2的基準實驗約增快17至25%，證實GMM計算在整體辨識系統佔相當重的時間比例。最近點概算配合部份距離刪除法確實可簡化GMM的計算量進而加速辨識時間。此外針對辨識率方面，不但沒有下降的趨勢，反而有微幅上升的現象，其原因可能因Treebank語料是由單一語者錄製，故雖然僅用概算方法仍能描述出該語者的語音特性，因此辨識率未有衰減的情況。

之後，運用特徵參數重新排序(Feature Component Reordering; FCR)的概念對於MFCC維度做進一步分析，去探討MFCC各個維度的機率分數影響程度的差異，作法為在語料中任意挑選四到五個的音檔，計算所有的右相關聲母、前後文獨立韻母以及Silence的MFCC的38個維度Log 機率值(稱為Average Distortion)的平均值。結果為圖4.1至圖4.3。

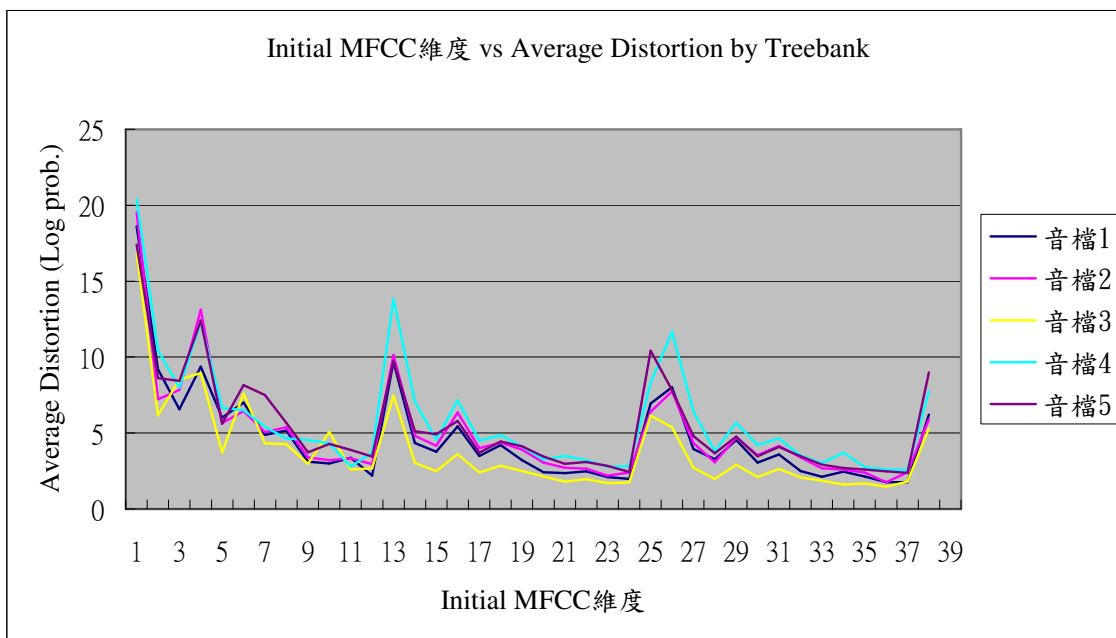


圖 4.1 Initial MFCC v.s. Average Distortion 圖示

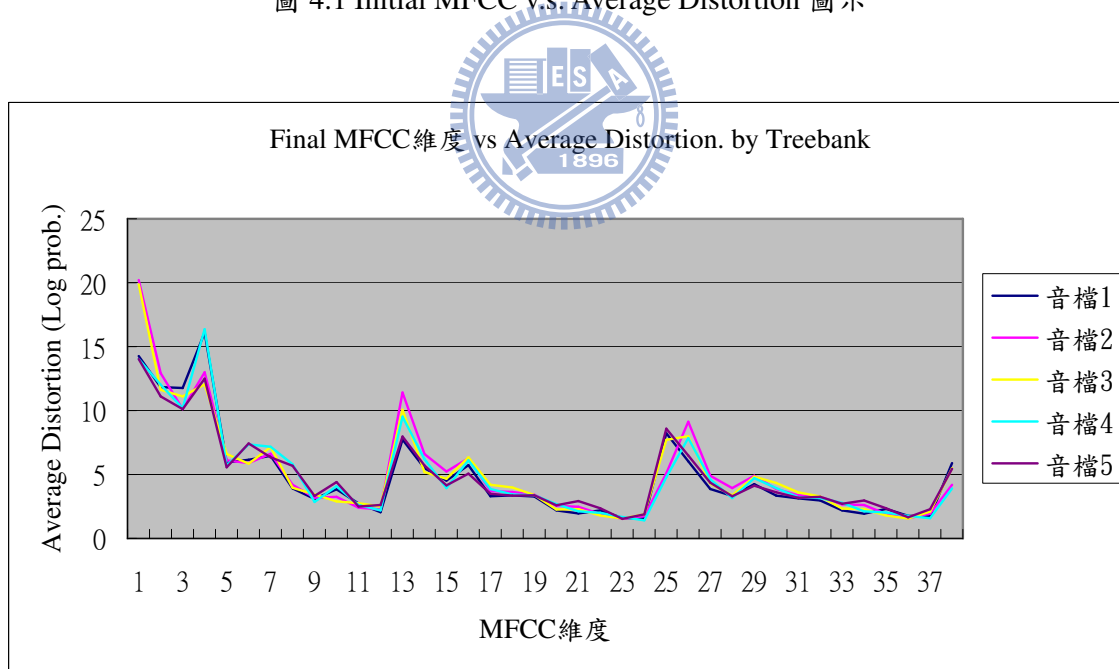


圖 4.2 Final MFCC v.s. Average Distortion 圖示

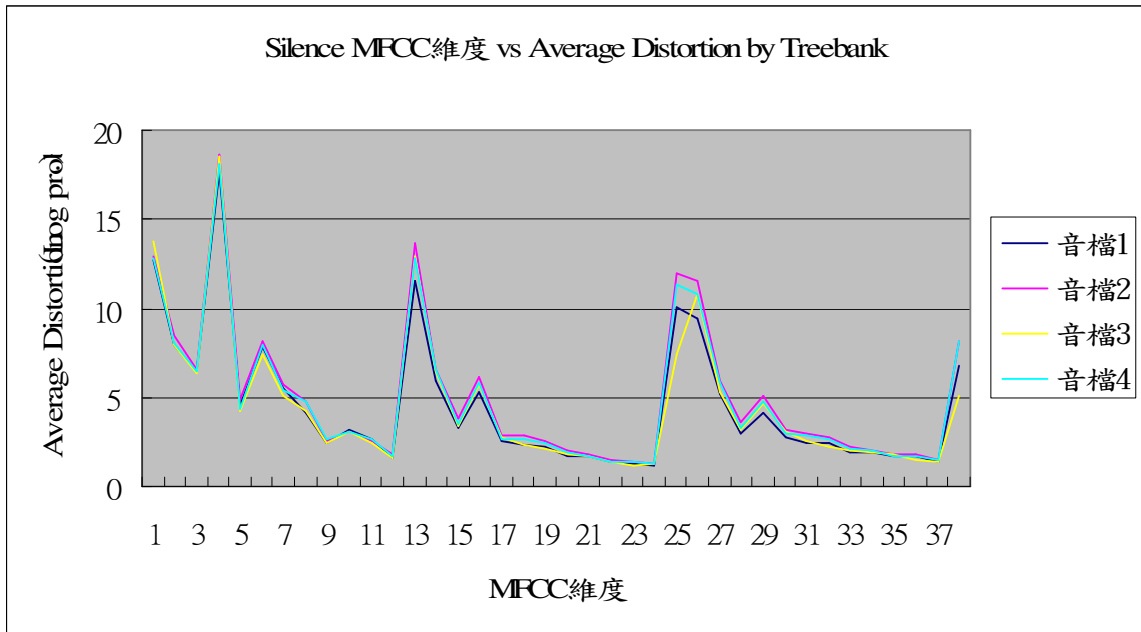


圖 4.3 Silence MFCC v.s. Average Distortion 圖示

圖4.1至圖4.3可發現MFCC的前幾維、delta前幾維和delta-delta的第一維以及delta和delta-delta的energy維度的Average Distortion相對於其他維度高許多。且聲母的五個音檔所畫出來的軌跡差異較大，韻母與Silence的音檔所繪製出來的軌跡較相似，可推斷出聲母的變異性比起韻母與Silence相對較高。然而三者的Average Distortion分佈高低仍存在固定規律。表4.5列出Average Distortion平均最高的前十個維度。

表4.5 Average Distortion最高的前十維

Initial	1	4	13	2	26	3	25	6	38	16
Final	1	4	2	3	13	26	25	7	6	5
Silence	4	1	13	26	25	2	6	38	3	14

表4.5發現Average Distortion最高前十維在聲母、韻母與Silence共同出現的MFCC維度總共有八個，尤其前7名維度幾乎相同，代表各音素狀態的MFCC維度對於Average Distortion的影響大小順序約固定。因此實做系統上對於聲學模型各個狀態只需統一建立單一個維度查詢表供給在辨識期間GMM計算使用即可。配合上最佳mixture的預測(Best Mixture Prediction; BMP)的方法，紀錄目前音框在某個狀態最佳的mixture，當下一個音框有遇到相同狀態時，則先計算前一刻最佳的mixture機率值再計算剩餘的mixture機率值。最後實驗結果如表4.6。

表4.6 利用最近點概算配合上FCR與BMP方法辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	93.8	82.5	585.685	0.372	76.43%
150	5000	94.74	84.79	833.792	0.523	78.38%
175	6000	94.75	84.97	1103.636	0.701	80.03%
200	6000	94.91	85.2	1331.026	0.845	80.82%
200	9000	94.9	85.28	1452.322	0.922	73.50%
225	8000	94.93	85.07	1650.562	1.048	76.08%

表4.6實驗結果發現到辨識速度會稍微進步，然而對整體系統增加快僅約1~2%，其原因可能為因前幾維MFCC的Average Distortion本來就相當高，故經過重新排序所制定的維度查詢表，與原本未經排序的計算，實質上這兩種演算法所最先計算到的MFCC維度非常相似，因此加速效果必然相當有限。另外表4.4與表4.6雖然數學式相同卻發生辨識率產生不同的結果，是由於浮點數計算時本身有四捨五入的機制，導致經由加減運算順序不同，會產生誤差導致最後辨識率的差異。

4.4 語言模型

語言模型方面主要是以加快建立雙連語言模型預查資訊為主。根據3.3節的概念，利用單連語言模型預查的資訊作為基礎，再藉由雙連語言模型機率分數去更新資訊去取代原始採用單連與雙連語言模型機率分數重新建立預查資訊，以降低建立時計算量。另外一方面前詞資訊已滿溢而需要刪除時，找出擁有雙連語言模型機率個數最少的前詞資訊，並將它刪除(然而仍需保證固定時間以內受到保護的條件，防止剛建立的前詞資訊由於個數過少而立即被刪除)，而寫進新的一筆前詞的資訊。實驗結果如下。

表 4.7 利用單連語言模型預查來快速建立雙連語言模型預查的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	93.8	82.5	520.197	0.33	67.88%
150	5000	94.74	84.79	757.591	0.481	71.22%
175	6000	94.75	84.97	1018.917	0.647	73.89%
200	6000	94.91	85.2	1248.383	0.793	75.80%
200	9000	94.9	85.28	1373.66	0.872	69.52%
225	8000	94.93	85.07	1554.196	0.987	71.64%

表4.7實驗結果發現利用單連語言模型預查的資訊可降低建立雙連模型語言預查的所耗費的計算量進而辨識時間增快許多。進一步觀察不同的刪除門檻所做出的實驗，發現各個測試結果所得到加快的時間約在一定範圍之間(約RTF增快 0.05 ± 0.01)，原因可能為因辨識系統中已經固定每個音框所能留下的詞個數為6個，故整個音檔能夠延展的詞總數大致相似，所以整體辨識時間的加快也約在固定時間範圍。此外因前詞資訊已滿溢而需要刪除所採用刪除雙連語言模型機率個數較少的前詞來取代佇列(Queue)，實驗發現對於辨識速度進步有限，原因可能因暫存的前詞個數已足夠，因此僅有少量前詞會被重複建立，經由實驗結果計算僅約加快3~8秒左右，此差距亦有可能為電腦本身處理的誤差範圍，故在此省略其實驗結果。

4.5 刪除演算法與其他加速法

辨識系統辨識途中，當詞尾作詞轉移時會延展出數千個hypotheses，導致計算量瞬間增加許多，因此若數個音框才做一次詞轉移應可節省龐大的計算量，相對地風險即是辨識率會產生若干損失，關於詞轉移的實驗結果如下。

表 4.8 每兩個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	92.59	80.94	361.411	0.23	47.16%
150	5000	94.56	84.39	554.681	0.352	52.14%
175	6000	94.79	84.77	811.562	0.515	58.85%
200	6000	94.81	84.81	1021.326	0.649	62.01%
200	9000	94.75	85	1121.528	0.712	56.76%
225	8000	94.77	84.79	1319.772	0.838	60.84%

表 4.9 每三個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	92.96	79.79	294.417	0.187	38.42%
150	5000	94.56	84.25	472.941	0.3	44.46%
175	6000	94.9	84.77	701.371	0.445	50.86%
200	6000	94.98	84.97	919.738	0.584	55.84%
200	9000	94.97	85.2	1014.028	0.644	51.32%
225	8000	95	84.99	1212.68	0.77	55.90%

表 4.10 每四個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	92.15	78.38	258.205	0.164	33.69%
150	5000	94.21	83.45	414.968	0.264	39.01%
175	6000	94.47	84.19	641.95	0.408	46.55%
200	6000	94.67	84.44	844.728	0.536	51.29%
200	9000	94.74	84.67	917.075	0.582	46.41%
225	8000	94.7	84.51	1133.067	0.72	52.23%

表 4.11 每五個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	90.85	76.86	238.573	0.152	31.13%
150	5000	93.96	82.7	377.023	0.239	35.44%
175	6000	94.37	84	580.57	0.369	42.10%
200	6000	94.58	84.33	789.387	0.501	47.93%
200	9000	94.77	84.56	851.651	0.541	43.10%
225	8000	94.61	84.32	1062.9	0.675	49.00%

表4.8至表4.11的實驗結果發現到數個音框才做一次詞轉移確實能讓系統辨識速度有明顯的提昇，在兩或三個音框延展一次的情況下，辨識速度大幅地加速而辨識率仍能維持水準；然而隨著音框增加而辨識速度加快幅度也逐漸縮小，相對地辨識率卻隨著音框增加而有明顯地下降。故最後本系統利用三個音框做延展一次的方式使得辨識效果最佳。

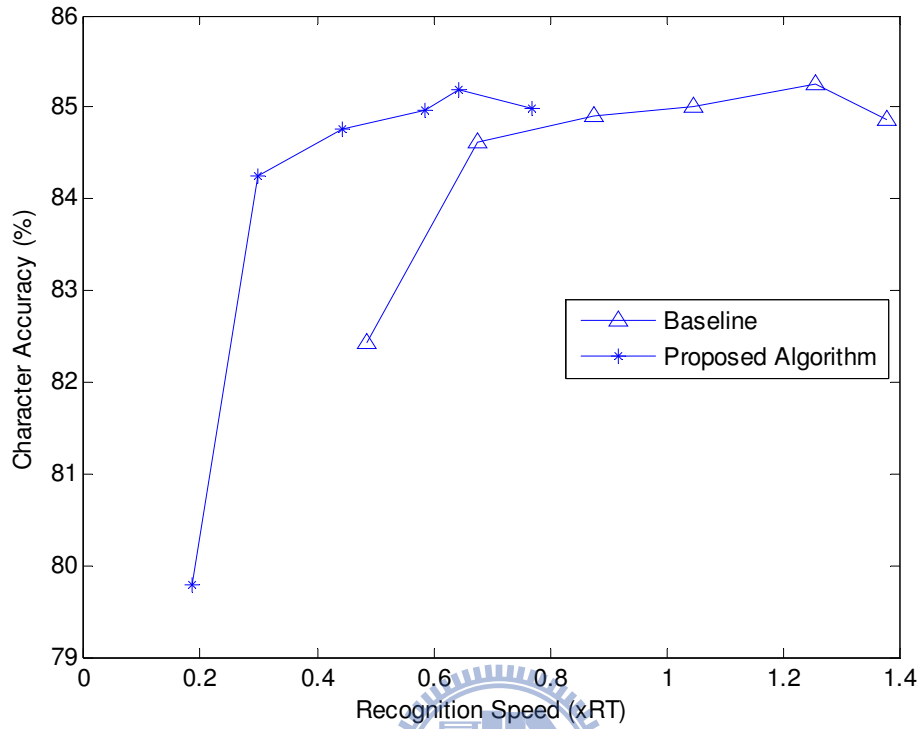


圖 4.4 基準實驗與經加速演算法後實驗的比較圖

藉由上述相關加速演算法的實現，圖4.4的基準實驗與經加速演算法後實驗比較圖相比，最終結果與基準實驗比較起來辨識速度節省40%以上，而整體辨識率的下降皆約在1%以內。進一步針對表4.12經由加速演算法後的各部份占整體辨識時間做分析，發現語言模型預查機率分數與詞尾作詞轉移兩部份所佔的時間比例下降許多。維特比演算法經由加速演算法後辨識時間亦有削減，故計算GMM的機率分數的時間所佔的比重只微幅降低。

表4.12 經由加速演算法後各部份占辨識系統時間百分比

Viterbi	Likelihood	Pruning	Else	LMLA	Word End
62.24%	31.32%	2.44%	4.00%	12.74%	13.77%
↑ 5.08%	↓ 6.89%	↓ 0.04%	↑ 1.54%	↓ 7.50%	↓ 6.78%

4.6 TCC300

如同第一部份，對於所有TCC300 NCKU語料做辨識並且觀察各部份占系統時間的比例如表4.13。結果發現GMM的機率分數計算約在32%，比起Treebank語料所占的比例略為降低，原因為因TCC300所使用聲學模型的mixture數較少，因此所需花費在GMM的計算量比例較低，相對地維特比演算法所佔比例較高。然而計算聲學模型GMM分數、建立與查詢語言模型預查機率分數以及做詞尾作詞轉移這三者依然是在辨識期間中所最耗費時間的部份。

表 4.13 利用 TCC300 語料估計各部份占辨識系統時間百分比

Viterbi	Likelihood Pruning	Else	LMLA	Word End
64.99%	29.87%	2.94%	2.20%	21.90%

表4.14是利用TCC300語料所作的基準實驗，如同前面Treebank第一部份的實驗，在TCC300語料中亦僅挑選數個辨識率較佳的數據做延伸實驗(詳細實驗數據於附錄A供參考)，字元最高辨識率等於72.66%，之後觀察在非特定語者語料中因加速演算法導致辨識率降低以及是否有與第一部份相同的加速程度。

表 4.14 TCC300 語料的基準實驗結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF
		syllable	character		
125	4000	80.17	70.14	2032.272	0.559
150	7000	82.3	72.17	3101.336	0.853
175	8000	82.51	72.64	4332.455	1.191
200	9000	82.67	72.62	5497.665	1.512
225	8000	82.71	72.66	5885.097	1.618

表4.15為經由一系列加速快演算法所得到的辨識結果(其他相關辨識結果於附錄A僅供參考)，辨識速度約增快22~44%，辨識率仍損失在1%以內。與Treebank語料所做出的辨識結果相比，辨識速度提昇較少，其主因可能為TCC300的聲學模型mixture數較少，因此在聲學模型所節省的GMM計算量相對較少，使得最終辨識速度的改善幅度未如同Treebank

語料部份來的大。

表 4.15 經由加速演算法後 TCC300 的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	80.65	69.34	1149.866	0.316	56.58%
150	7000	82.12	71.87	1970.832	0.542	63.55%
175	8000	82.22	72.2	3065.089	0.843	70.75%
200	9000	81.94	71.85	4114.897	1.132	74.85%
225	8000	82.45	72.22	4599.251	1.265	78.15%

圖4.5比較圖觀察發現，經過加速演算法之後，與第一部份Treebank語料所得到結果相比則有些差異。TCC300語料對於辨識率較高的部份因加速演算法會使得辨識率的下滑，導致辨識速度雖有加快卻與原本基準實驗有相似辨識率的部份的辨識速度差距不大。(如圖4.5右半部份)；然而在可容許辨識率微幅下降的(與最高辨識率相差1%)情況下，仍發現辨識系統仍可得到速度改善(如圖4.5左半部份)。因此對於非特定語者在容許的錯誤率下，加速演算法仍能幫助有效加快系統的辨識速度。

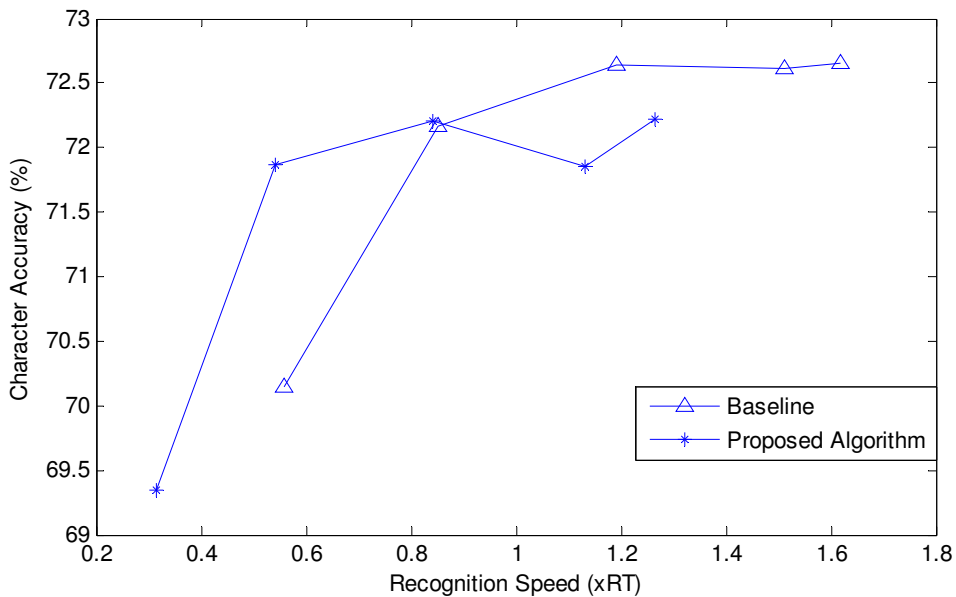


圖 4.5 TCC300 基準實驗與經加速演算法後實驗的比較圖

最後表4.16各部份占辨識時間百分比所得出來結果與Treebank語料做出的結果相似，在此不再贅論。

表 4.16 經由加速演算法後 TCC300 各部份占辨識系統時間百分比

Viterbi	Likelihood	Pruning	Else	LMLA	Word End
67.38%	27.12%	2.51%	2.99%	14.05%	15.28%



第五章 建構語音辨識系統

5.1 系統展示

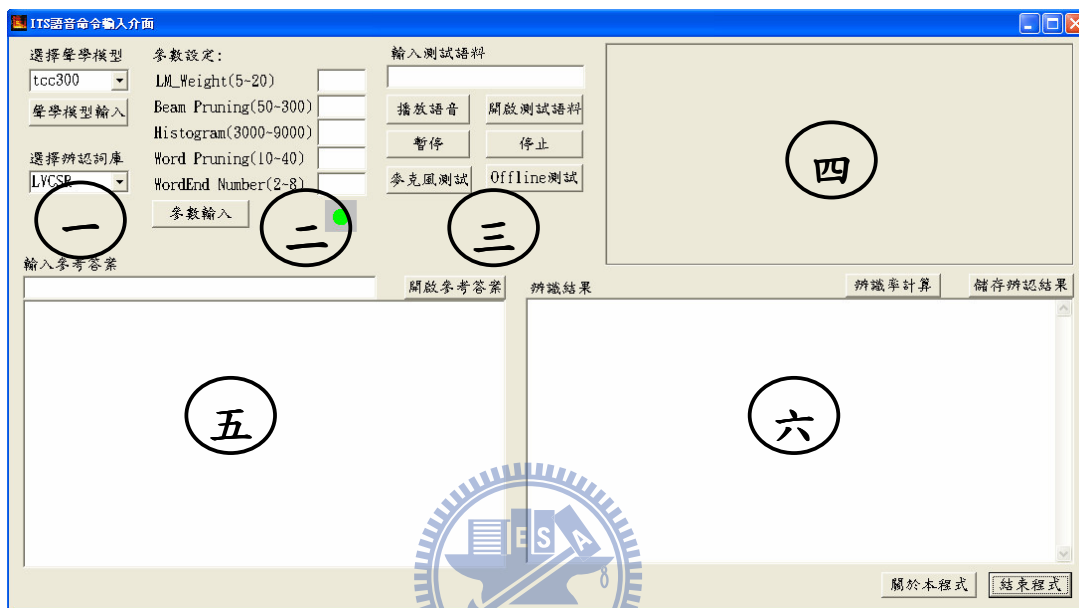


圖 5.1 辨識系統介面展示圖

本語音辨識系統介面是利用MFC(Microsoft Foundation Classes)程式所撰寫，可將系統大致分為六部份，詳細介紹如下：

- 一、聲學與詞典擷取：此部份功能為可利用事前所建立的聲學模型以及語言模型，在程式開啟後仍能替換聲學模型。
- 二、參數設定：此部份功能為調整刪除演算法的門檻值、語言模型權重等相關語言參數設定。
- 三、輸入測試語料：此部份功能為定義輸入測試，可為HTK所產生的mfc檔案、pcm音檔或是直接利用麥克風輸入的選擇。
- 四、波形與音檔資訊：此部份功能為可顯示音檔的波形，並且列出音檔波形時間以及辨識所花費的時間。
- 五、文字檔案輸入：此部份功能為可輸入純文字檔案並且顯示在辨識系統介面上，方便使用者念稿。

六、辨識結果以及辨識率計算：此部份功能為將語音辨識結果顯示在介面上，並且可藉由事先準備好的解答去計算辨識率。若在第五部份有輸入文字檔時，可進一步逐一去比對辨識結果與文稿的相同與差異性。

圖5.2和圖5.3為語音辨識系統的實際Demo範例。左上角先將語音系統相關的模型和參數設定輸入，之後開啟純文字檔作為使用者念稿準備，利用麥克風測試鍵錄下使用者語音，再經由本語音辨識系統處理過後即可輸出語音波形以及辨識解答，最後再按下辨識率計算，即可逐一比對預備文稿和以及辨識結果的對照，並計算音節與字元的正確率，最後可儲存辨識結果將所有資訊儲存至純文字檔案。此外可按播放語音即可將使用者的聲音輸出至喇叭。

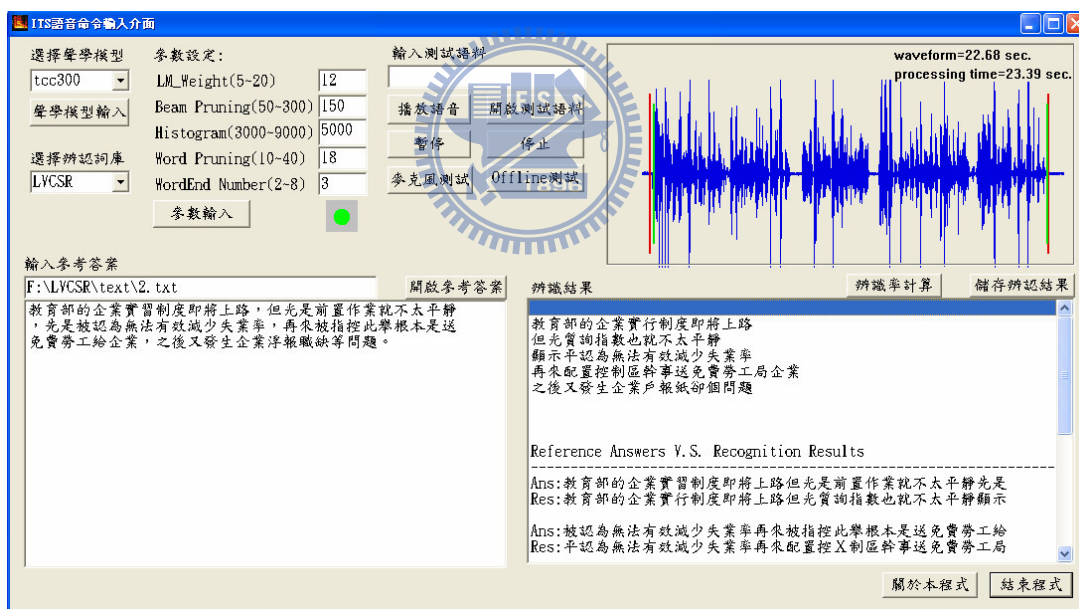


圖 5.2 辨識系統實際 Demo 圖之一

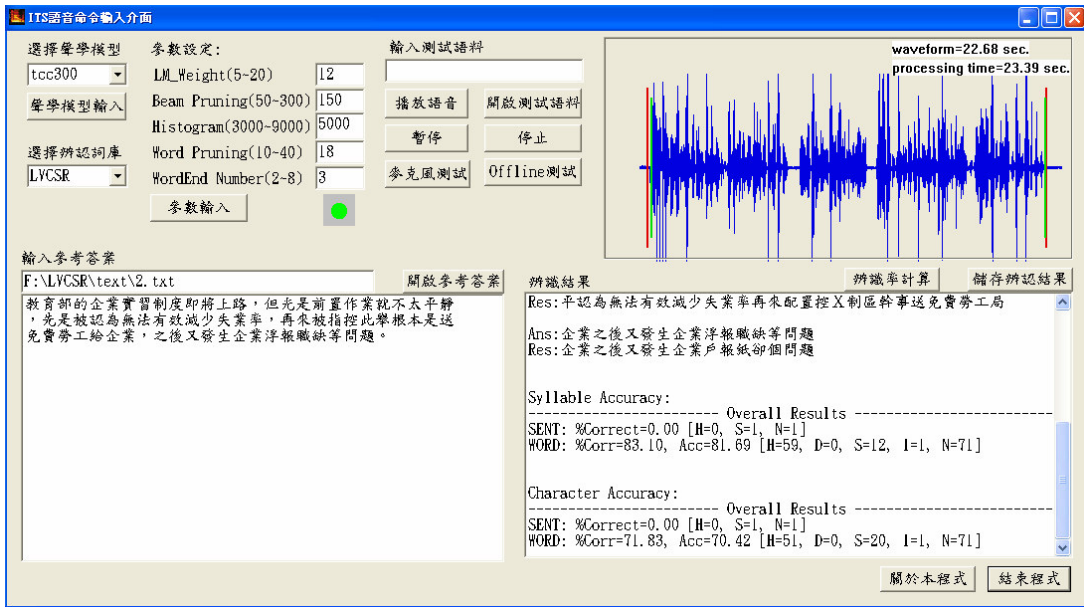


圖 5.3 辨識系統實際 Demo 圖之二

另外該語音辨識系統可因不同的應用而可設計出不同語法的詞典，例如將所有詞條建立一顆節點龐大的樹供系統辨識時搜尋；或是利用文法的結構再利用HTK的HParse指令將該文法建立出詞網(Word Net)，再經由本系統相關程式修改成近似雙連語言模型形式的語言機率模型，將合法路徑的語言模型機率設成1，而將不合法路徑的語言模型機率設成0，如此一來可藉由刪除演算法將非法路徑的詞條立刻刪除，即可達成文法架構的語音辨識系統。

第六章 結論與未來展望

6.1 結論

本論文主要分為兩個部份，在論文的前半段對於大詞彙連續語音辨識系統相關能夠加快速率的演算法做一系列的探討與實驗，從利用最近點概算等方法簡化聲學模型裡的GMM計算，其次利用單連語言模型預查的資訊去簡化建立雙連語言模型預查所需要的計算量，最後去對於詞尾的詞轉移做經過數個音框才延展的方式等相關演算法，在單一語者Treebank的語料下，使得整體辨識速度增快了40%以上。而針對不定語者TCC300的語料亦有辨識速度的提昇。

論文後半部對於語音辨識系統也改進至具有彈性(Flexible)的系統，可藉由介面的呈現，方便使用者去動態調整模型與相關參數。此外也將此辨識系統添增類似語音辨識的互動系統，可藉由文字內容與語音辨識結果的比對，方便使用者能夠清楚瞭解兩者文字相異的地方。最後對於不同應用而所需具有文法規則的語言模型，亦可透過編輯之後嵌入至本辨識系統中。



6.2 未來展望

本系統速度提昇之後，進一步可以去使用更複雜的模型去做此辨識系統。例如使用三連音素模型(Tri-phone Model)或是三連語言模型(Trigram LM)等模型來提昇整體辨識率；或是建構出詞網(Word Lattice)之後再用其他語音相關知識做再計分(Rescoring)的動作以提高辨識率。

對於語音辨識能夠真正落實於真實世界環境除了語音辨識本身相關搜尋演算法之外，對於由麥克風端至PC端的處理亦為同等重要，例如使用VAD(Voice Activity Detector)技巧有效地偵測出語音的內容並且進一步抵抗環境背景的雜訊，可有效地決定語音辨識的斷點偵測(Endpoint Detection)位置。

此外該系統是針對於字元基礎(Character-based)作為詞典架構，使得該系統無法處理中文破音字的問題，若將該系統修改成詞基礎(Word-based)的詞典架構應能進一步處理中文破

音字的問題。

最後可利用該系統的即時辨識特性與系統的彈性設計，加入語音合成相關功能，可進而修改成多功能應用的對話系統(Dialogue System)，更能讓學術實現於日常生活。畢竟科技始終來自於人性，雖然語音辨識仍然是一門深不可測的學問，然而如何讓這項技術成為一般人的生活中便利的工具，仍需依靠後續的傑出研究人才努力克服了。



參考文獻

- [1] S. Young, N. Russell, and J. Thornton, "Token passing: A Conceptual Model for Connected Speech Recognition Systems," Technical Report F INFENG/TR38, Cambridge University, 1989.
- [2] H. Ney and S. Ortmanns, "Progress in Dynamic Programming Search for LVCSR, " *Proceedings, IEEE*, Aug. 2000, Vol. 88, pp. 1224 - 1240.
- [3] Ortmanns, S., Ney, Hermann, and Eiden, A. "Language-model Look-ahead for Large Vocabulary Speech Recognition," In *ICSLP-1996*, 2095-2098.
- [4] A. Cardenal-Lopez, F.J. Dieguez-Tirado, and C. Garcia-Mateo, "Fast LM Look-ahead for Large Vocabulary Continuous Speech Recognition Using Perfect Hashing," in *Proc. ICASSP*, May 2002, pp. 705–708.
- [5] Ortmanns, S., Ney, H., Eiden, A. (1996): "Language-Model Look-Ahead for Large Vocabulary Speech Recognition", In *Proc. of the Int. Conf. of Spoken Language Processing*, pp. 2091–2094, Philadelphia, PA, October 1996. Ortmanns,
- [6] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted Finite-state Transducers in Speech Recognition," *Computer Speech and Language*, 16(1):69–88, 2002.
- [7] D. Caseiro and I. Trancoso, "A Specialized On-the-fly Algorithm for Lexicon and Language Model Composition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1281–1291, July 2005.
- [8] Hori, Takaaki / Hori, Chiori / Minami, Yasuhiro (2004): "Fast On-the-fly Composition for Weighted Finite-state Transducers in 1.8 Million-word Vocabulary Continuous Speech Recognition", In *INTERSPEECH-2004*, 289-292.
- [9] O. Cheng, J. Dines, M.M. Doss, "A Generalized Dynamic Composition Algorithm of Weighted Finite State Transducers for Large Vocabulary Speech Recognition," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* , vol.4, no., pp.IV-345-IV-348, 15-20 April 2007
- [10] D. Willett, C. Neukirchen, G. Rigoll, "DUcoder-the Duisburg University LVCSR Stack Decoder," *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on* , vol.3, no., pp.1555-1558 vol.3, 2000

- [11] Lei, Xin / Siu, Manhung / Hwang, Mei-Yuh / Ostendorf, Mari / Lee, Tan (2006): “Improved Tone Modeling for Mandarin Broadcast News Speech Recognition,” In *INTERSPEECH 2006*, paper 1752-Tue3A2O.4
- [12] H. L. Wang, Y. Qian , F. K. Soong, J. L. Zhou, et al, “Improved Mandarin Speech Recognition by Lattice Rescoring with Enhanced Tone models,” in *Proc. of ISCSLP*, pp. 445-443, 2006
- [13] Jinyu Li, Yu Tsao, Chin-Hui Lee, “A Study on Knowledge Source Integration for Candidate Rescoring in Automatic Speech Recognition,” *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on* , vol.1, no., pp. 837-840, March 18-23, 2005
- [14] Nicolás Morales, Liang Gu, Yuqing Gao, “Fast Gaussian Likelihood Computation by Maximum Probability Increase Estimation for Continuous Speech Recognition,” *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on* , vol., no., pp.4453-4456, March 31 2008-April 4 2008
- [15] E. Bochieri, “Vector Quantization for The Efficient Computation of Continuous Density Likelihoods,” *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on* , vol.2, no., pp.692-695 vol.2, 27-30 Apr 1993
- [16] Hoon Chung, Jeon Gue Park, Yun Keun Lee and Ikjoo Chung, “Fast Speech Recognition to Access a Very Large List of Items on Embedded Devices,” *Consumer Electronics, IEEE Transactions on* , vol.54, no.2, pp.803-807, May 2008
- [17] M. Afify, Feng Liu, Hui Jiang, O. Siohan, “A New Verification-based Fast-match for Large Vocabulary Continuous Speech Recognition,” *Speech and Audio Processing, IEEE Transactions on* , vol.13, no.4, pp. 546-553, July 2005
- [18] M.K. Ravishankar, “Efficient Algorithms for Speech Recognition,” *Ph.D. thesis*, School of Computer Science, Carnegie Mellon University, 1996.
- [19] E. Whittaker and B. Raj, “Quantization-Based Language Model Compression,” *Proc. Eurospeech*, pp.33–36, Aalborg, Denmark, 2001.
- [20] B. Raj , E. U. D. Whitaker “Lossless Compression of Language Model Structure and Word Identifiers,” *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on* , vol.1, no., pp. I-388-I-391 vol.1, 6-10 April 2003

- [21] Langzhou Chen and K. K. Chin, "Efficient Language Model Look-ahead Probabilities Generation Using Lower order LM Look-ahead Information," *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, vol., no., pp.4925-4928, March 31 2008-April 4 2008
- [22] R Kneser, H Ney, "Improved Backing-off for M-gram Language Modeling," *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol.1, no., pp.181-184 vol.1, 9-12 May 1995
- [23] B. L. Pellom, R. Sarikaya, and J. H. L. Hansen, "Fast Likelihood Computation Techniques in Nearest-Neighbor Based Search for Continuous Speech Recognition," *IEEE Signal Processing Letters*, Vol. 8, No. 8, pp. 221-224, August 2001.
- [24] Jun Cai, Ghazi Bouselmi, Yves Laprie, Jean-Paul Haton, "Efficient Likelihood Evaluation and Dynamic Gaussian Selection for HMM-based Speech Recognition," *Computer Speech & Language*, Volume 23, Issue 2, April 2009, Pages 147-164, ISSN 0885-2308, DOI: 10.1016/j.csl.2008.05.002.
- [25] 李庚達, "雙連語言模型預查之大詞彙連續語音辨識系統," 交通大學電信工程系碩士論文, 2008。
- [26] M. J. F. Gales, K. M. Knill, and S. J. Young, "State-Based Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMM's," *IEEE Trans. on Speech and Audio Processing*, Vol. 7, No. 2, pp. 152-161, March 1999.
- [27] Cambridge university Engineering Dept. (CUED), "Hidden Markov Model Toolkit (HTK)," <http://htk.eng.cam.ac.uk>, version 3.4.



附錄A、TCC300辨識相關結果

表 A.1 TCC300 語料的基準實驗結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF
		syllable	character		
125	3000	79.41	69.95	2023.3	0.556
125	4000	80.17	70.14	2032.272	0.559
125	5000	81.2	70.12	2042.344	0.562
150	4000	82.29	71.9	2883.832	0.793
150	5000	82.24	72.06	2972.664	0.817
150	6000	82.28	72.15	3071.205	0.845
150	7000	82.3	72.17	3101.336	0.853
150	8000	82.26	72.17	3130.998	0.861
175	5000	82.07	71.94	3805.176	1.046
175	6000	82.57	72.55	4020.465	1.106
175	7000	82.62	72.54	4179.545	1.149
175	8000	82.51	72.64	4332.455	1.191
175	9000	82.58	72.55	4430.934	1.218
200	5000	82.53	72.35	4369.244	1.201
200	6000	82.51	72.42	4686.33	1.289
200	7000	82.54	72.32	4983.183	1.37
200	8000	82.61	72.61	5249.656	1.444
200	9000	82.67	72.62	5497.665	1.512
225	6000	82.54	72.36	5107.056	1.404
225	7000	82.64	72.46	5512.399	1.516
225	8000	82.71	72.66	5885.097	1.618
225	9000	82.61	72.28	6265.196	1.723
250	6000	82.51	72.15	5369.691	1.477
250	7000	82.62	72.45	5846.051	1.608
250	8000	82.56	72.38	6256.837	1.721
250	9000	82.72	72.6	6491.626	1.785
250	10000	82.47	71.95	7175.987	1.973

表 A.2 TCC300 利用最近點概算配合上 FCR 與 BMP 方法辨識結果表

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	81.02	70.09	1851.812	0.509	91.12%
150	7000	82.38	72.17	2841.321	0.781	91.62%
175	8000	82.38	72.21	4043.854	1.112	93.34%
200	9000	82.61	72.47	5194.134	1.428	94.48%
225	8000	82.57	72.51	5464.691	1.503	92.86%

表 A.3 TCC300 利用單連語言模型預查來快速建立雙連語言模型預查的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	81.02	70.09	1647.975	0.453	81.09%
150	7000	82.38	72.17	2562.473	0.705	82.62%
175	8000	82.38	72.21	3733.213	1.027	86.17%
200	9000	82.61	72.47	4830.917	1.328	87.87%
225	8000	82.57	72.51	5174.652	1.423	87.93%

表 A.4 TCC300 每兩個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	80.65	69.34	1149.866	0.316	56.58%
150	7000	82.12	71.87	1970.832	0.542	63.55%
175	8000	82.22	72.2	3065.089	0.843	70.75%
200	9000	81.94	71.85	4114.897	1.132	74.85%
225	8000	82.45	72.22	4599.251	1.265	78.15%

表 A.5 TCC300 每三個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	79.88	68.02	936.84	0.258	46.10%
150	7000	81.85	71.42	1683.203	0.463	54.27%
175	8000	82.19	71.82	2737.514	0.753	63.19%
200	9000	82.27	71.91	3823.671	1.051	69.55%
225	8000	82.39	71.88	4321.441	1.188	73.43%

表 A.6 TCC300 每四個音框延展一次詞轉移的辨識結果

Beam Width	Histogram	Acc(%)		Real-time(s)	RTF	Ex. time(%)
		syllable	character			
125	4000	78.22	66	809.614	0.223	39.84%
150	7000	81.15	70.08	1516.094	0.417	48.89%
175	8000	81.44	70.53	2530.352	0.696	58.40%
200	9000	81.68	70.85	3630.771	0.998	66.04%
225	8000	81.7	70.79	4146.647	1.14	70.46%