การต่อหมายเลขโทรศัพท์โดยใช้เสียง

DIALING TELEPHONE NUMBER BY SPEECH RECOGNITION

โดย

นางสาวกรุณา แก้วสมศรี รหัส 36014007

นางสาวระพิกร นนท์จุมจัง รหัส 36014344

นางสาวโสภิณ ประมาคะเต รหัส 36014531

อาจารย์ที่ปรึกษา

ดร.ไกรสิน ส่งวัฒนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การต่อหมายเลขโทรศัพท์โดยใช้เสียง

DIALING TELEPHONE NUMBER BY SPEECH RECOGNITION

ผู้จัดทำ

1.นางสาวกรุณา แก้วสมศรี 36014007

2.นางสาวระพิกร นนท์จุมจัง 36014344

3.นางสาวโสภิณ ประมาคะเต 36014531

_____ อาจารย์ที่ปรึกษา

(ดร.ไกรสิน ส่งวัฒนา)

การต่อหมายเลขโทรศัพท์โดยใช้เสียง

(DIALING TELEPHONE NUMBER BY SPEECH RECOGNITION)

โดย นางสาวกรุณา แก้วสมศรี 36014007
นางสาวระพิกร นนท์จุมจัง 36014344
นางสาวโสภิณ ประมาคะเต 36014531

อาจารย์ที่ปรึกษา ดร.ไกรสิน ส่งวัฒนา

บทคัดย่อ

ปริญญานิพนธ์นี้มีจุดประสงค์เพื่อเพิ่มความสามารถในการติดต่อระหว่างมนุษย์กับคอมพิวเตอร์ด้วยการใช้เสียงพูดแทนการกดคีย์บอร์ด โดยให้สามารถต่อโทรศัพท์ด้วยเสียงได้ โดยใช้หลักการของการรู้จำเสียงพูด(Speech Recognition) ซึ่งประกอบด้วย กระบวนการการรู้จำเสียงพูด (Linear Predictive Coding :LPC), เวคเตอร์ควอนไตซ์เซชั่น (Vector Quantization :VQ) และกระบวนการสร้างแบบจำลองเสียงแบบฮิดเดน มาร์คอฟ โมเดล (Hidden Markov Models :HMM)

ในที่นี้ได้มีการศึกษาโครงงานการรู้จำเสียงพูดตัวเลขภาษาไทยแบบคำโดด ไม่ขึ้นกับผู้พูด และ ส่วนที่เป็นแนวทางการประยุกต์ใช้งาน โดยใช้ภาษาซีเขียนโปรแกรมทั้งสองส่วน ซึ่งได้ทำการสร้างแบบจำลองของเสียงที่เป็นชื่อคน10ชื่อ และมีฐานข้อมูลของแต่ละชื่อเป็นหมายเลขโทรศัพท์แล้วทำการเขียนโปรแกรมสั่งงานโมเด็มต่อโทรศัพท์ให้ตามคำที่สามารถรู้จำได้

ABSTRACT

This project proposes to use an enabling technology that allows people to interact with computers by recognizing speech in place of pressing keyboard.This projection,it is able to dial telephone by voice. The theory of Speech Recognition is used,including Linear Predictive Coding (LPC), Vector Quantization (VQ) and Hidden Markov Models (HMM).

The thai numeral speech recognition is carried out based on independent speaker and task oriented application for automatic speech recognition part.Both are programmed by C language .For task oriented application,the models of 10 names with database of each name associate with telephone number have been created.To program in order to direct modem for dialing of telephone number has been done.

สารบัญ

บทที่1 บทนำ.....1

บทที่2 ทฤษฎีหรือหลักการ.....3

2.1 ทฤษฎีการรู้จำเสียง.....3

2.1.1 การวิเคราะห์เสียงเบื้องต้น.....3

2.1.2 การจัดระดับเวคเตอร์.....10

2.1.3 การสร้างแบบจำลอง.....15

2.2 การเชื่อมต่อส่วนการรู้จำเสียงกับระบบโทรศัพท์.....26

บทที่3 การคำนวณและการสร้าง.....29

3.1 ส่วนการวิเคราะห์เสียง.....29

3.2 ส่วนควบคุมการสั่งงานโมเด็ม.....31

บทที่4 การทดลองและผลการทดลอง.....32

4.1 การทดลอง.....32

4.2 ผลการทดลอง.....34

บทที่5 สรุปและวิจารณ์ผลการทดลอง.....46

5.1 สรุปผลการทดลอง.....46

5.2 วิจารณ์.....47

5.3 แนวทางพัฒนา.....47

ภาคผนวก

เอกสารอ้างอิง

กิตติกรรมประกาศ

สารบัญรูปภาพ

รูปที่ หน้า

2.1 บล็อกไดอะแกรมของการรู้จำเสียง....3

2.2 แสดงขั้นตอนการเตรียมสัญญาณในการวิเคราะห์....3

2.3 วงจรกรองความถี่สูงผ่าน....4

2.4 การแบ่งช่วงของสัญญาณ....4

2.5 แสดงวินโดว์แบบแฮมมิ่ง....5

2.6 แสดงขั้นตอนในการวิเคราะห์หาคุณลักษณะของเสียง....6

2.7 บล็อกไดอะแกรมแสดงโมเดลการสร้างสัญญาณเสียงพูดอย่างง่าย....7

2.8 แสดงการกระจายเฟรมของเสียงพูดแต่ละจุดแทนเฟรมของเสียง....11

2.9 การรวมกลุ่มของเฟรมเสียงเพื่อสร้างโค้ดบุค....11

2.10 แสดงตัวอย่างการหาโค้ดบุค....12

2.11 บล็อกไดอะแกรมของเวกเตอร์ควอนไตซ์....12

2.12 ขั้นตอนของเวกเตอร์ควอนไตเซชั่น....14

2.13 แสดงแบบจำลองต่างๆของ HMM....17

2.14 กระบวนการไปข้างหน้า....18

2.15 กระบวนการถอยหลัง....19

2.16 แสดงลำดับการคำนวณการเกิดค่าปรากฏร่วมซึ่งจะอยู่ที่สเตท i ที่เวลา t และอยู่ที่สเตท j ที่เวลา t+1....20

2.17บล็อกไดอะแกรมการรู้จำคำโดดด้วยแบบจำลองของมาร์คอฟ....25

4.1 แสดงขั้นตอนการเรียนรู้....32

4.2 แสดงขั้นตอนการวิเคราะห์....33

4.3 แสดงสัญญาณเสียงอินพุท 1 เฟรมของเสียง 1....35

4.4 แสดงสัญญาณที่ผ่านการพรีเอมฟาซีส 1 เฟรม ของเสียง 1....35

4.5 แสดงสัญญาณที่นำมาผ่านการวินโดว์ 1 เฟรม ของเสียง 1....36

สารบัญตาราง

ตารางที่ หน้า

2.1 รหัสผลลัพท์จากโมเด็ม.....28

4.1 แสดงค่าสัมประสิทธิ์ LPC ,เซปสตรัม และค่าพารามิเตอร์ที่เวทแล้ว.....37

4.2 ผลจากการทดสอบโดยให้ผู้พูดคนเดิมพูด 0-9 เสียงละ 5 ครั้ง.....38

4.3 ผลจากการทดสอบโดยใช้เสียงผู้อื่น (unknown) พูด 0-9 เสียงละ 1 ครั้ง.....38

4.4 ผลจากการทดสอบโดยให้ผู้พูดคนเดิมพูด 0-9 เสียงละ 5 ครั้ง.....39

4.5 ผลจากการทดสอบโดยใช้เสียงผู้อื่น (unknown) พูด 0-9 เสียงละ 1 ครั้ง.....39

4.6 ผลจากการทดสอบโดยให้ผู้หญิงคนเดิมทั้ง 7 คน พูด 0-9 เสียงละ 1 ครั้ง.....40

4.7 ผลจากการทดสอบโดยใช้เสียงผู้อื่น (unknown) พูด 0-9 เสียงละ 1 ครั้ง.....40

4.8 ผลจากการทดสอบโดยใช้ผู้พูดคนเดิมพูด 0-9 เสียงละ 1 ครั้ง.....41

4.9 ผลจากการทดสอบโดยใช้เสียงผู้อื่น (unknown) พูด 0-9 เสียงละ 1 ครั้ง.....41

4.10 ผลจากการทดสอบโดยใช้ผู้พูดคนเดิมพูดชื่อคน 20 ชื่อ.....42

4.11 ผลจากการทดสอบโดยใช้เสียงคนอื่น (unknown) พูดเสียงชื่อคน ชื่อละ 1 ครั้ง.....43

4.12 แสดงผลการเปรียบเทียบความถูกต้องของการทดสอบการรู้จำเสียงพูด.....44

บทที่1
บทนำ

ในอดีตมีกาวิเคราะห์ว่าการสอนให้คอมพิวเตอร์สามารถรู้จำเสียงพูดของมนุษย์นั้นทำไม่ได้ เนื่องจากการพูดของมนุษย์มีความซับซ้อนและมีความแตกต่างกันในแต่ละบุคคล ความเร็วในการพูด รวมถึงเสียงสูงต่ำของคนแต่ละคน แต่ก็มีการวิจัยเพื่อให้คอมพิวเตอร์สามารถรู้จำเสียงพูดเรื่อยมา จนในปัจจุบันการรู้จำเสียงพูดสามารถจะใช้งานได้ดี และมีการนำไปใช้กับอุปกรณ์ต่าง ๆ เช่น ATM (Automated Teller Machine) และโทรศัพท์มือถือ ถึงแม้คอมพิวเตอร์จะไม่สามารถรู้จำเสียงได้เท่ากับมนุษย์ แต่เราสามารถสอนให้คอมพิวเตอร์รู้จำได้ในขีดจำกัดระดับหนึ่ง โดยมีการกำหนดขอบเขตของการรู้จำ เช่นจำกัดคำที่จะสามารถรับรู้ได้

ปริญญานิพนธ์นี้ได้มีการศึกษาและทดลองนำเอาส่วนของการรู้จำเสียงที่เป็นทฤษฎีมาใช้งานจริง โดยทำในรูปการสั่งใช้เสียงสั่งคอมพิวเตอร์ให้หมุนเบอร์โทรศัพท์ให้ ซึ่งจะได้กล่าวต่อไป

วัตถุประสงค์ของปริญญานิพนธ์ มีดังนี้

- ศึกษาการวิเคราะห์เสียงพูดแบบไม่ขึ้นกับผู้พูด
- นำงานวิจัยเกี่ยวกับการรู้จำเสียงภาษาไทยที่ได้มีผู้พัฒนาไว้ในระดับหนึ่งแล้วมาทำการศึกษาและพัฒนาต่อ โดยจะเน้นพิจารณาและพัฒนาในด้านการนำไปใช้งานจริง
- เพื่อพัฒนาโปรแกรมส่วนต่างๆที่ทำให้คอมพิวเตอร์สามารถรู้จำเสียงพูด และนำโปรแกรมมาใช้งานได้
- ทำการปรับปรุงวิธีการทางทฤษฎีให้สามารถนำมาใช้ในทางปฏิบัติได้ เช่นหาวิธีที่จะช่วยเพิ่มความเร็วในการวิเคราะห์เสียง หรือปรับปรุงขั้นตอนบางอย่างเพื่อลดความผิดพลาดในการวิเคราะห์ เป็นต้น
- เพื่อเป็นพื้นฐานในการประยุกต์ใช้งานเพิ่มเติมต่อไป

ขอบเขตของปริญญานิพนธ์

เพื่อให้สามารถ "ใช้เสียงสั่งคอมพิวเตอร์ให้โทรศัพท์ผ่านโมเด็มไปยังบุคคลที่ต้องการได้" ซึ่งมีขอบเขตพอสังเขปดังนี้

1. รับเสียงพูดผ่านทาง Microphone ที่ต่อมาจาก Sound Card ของเครื่อง PC แล้วนำมาประมวลผล และส่งผลที่ได้ไปสั่ง Modem ให้ทำการโทรศัพท์ไปยังบุคคลนั้น

2. เสียงที่ใช้สั่งไม่ขึ้นกับผู้พูด เสียงของคำที่มีความหมายเดียวกันจากผู้พูดต่างบุคคลก็จะได้ผลการรับรู้เหมือนกัน โดยขั้นต้นจะมีการเก็บตัวอย่างเสียงของคำนั้นๆ ที่มากและหลากหลายพอไว้ก่อน เพื่อผลการรับรู้ที่ถูกต้อง

3. เป็นการวิเคราะห์เสียงพูดแบบไม่ต่อเนื่อง ช่วงวิเคราะห์สัญญาณครั้งละ 1 คำ หรือประมาณ 0.5 - 1.0 วินาที

4. ในโครงงานนี้จะใช้ตัวอย่างให้สามารถรับรู้เสียงที่แตกต่างกันได้ประมาณ 10 เสียง หรือคือสามารถสั่งให้ Modem โทรศัพท์ไปยังบุคคลที่แตกต่างกันได้ 10 คน

ในการศึกษาเรื่องการสั่งงานคอมพิวเตอร์ด้วยเสียง ส่วนของขั้นตอนต่างๆ ที่ใช้ในการวิคราะห์เสียงและการควบคุมอุปกรณ์ต่างๆ นั้น จะต้องมีความรู้พื้นฐานพอสมควรในเรื่องต่างๆ ดังต่อไปนี้

- ลักษณะต่างๆ ของเสียงพูดมนุษย์ เช่น ความถี่ , ความดัง , ความแตกต่างของเสียงชายและหญิง , เสียงที่เปล่งออกมาทางปากหรือออกมาทางจมูก , เสียงรบกวนที่เกิดขึ้น และกราฟต่างๆ ของเสียง เป็นต้น

- พื้นความรู้ทางด้านคณิตศาสตร์ที่สำคัญ เช่น ฟังก์ชันการถ่ายโอน , การประมาณเชิงเส้น , ผลรวมกำลังสองของความคลาดเคลื่อน , เวคเตอร์หลายมิติ และเมตริกซ์ เป็นต้น

- พื้นความรู้ทางด้านสถิติ เช่น ความน่าจะเป็น , ความน่าจะเป็นแบบมีเงื่อนไข

- ลักษณะการทำงานของ Sound Card ว่าเก็บเสียงมาได้อย่างไร มีการแปลงสัญญาณจากสัญญาณอนาลอกเป็นดิจิตอลอย่างไร สัญญาณเสียงที่ได้มีการจัดเก็บและนำมาใช้งานได้ อย่างไร

- ลักษณะการทำงานของ Modem การสั่งงาน Modem

- หลักการเขียนโปรแกรมภาษา C หรือ C++

ขั้นตอนในการศึกษาและดำเนินงานมีดังนี้

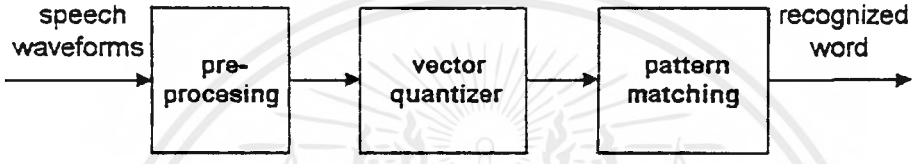
1. ศึกษารูปแบบการวิเคราะห์เสียงพูดแบบไม่ขึ้นกับผู้พูด และไม่ต่อเนื่อง
2. ศึกษาขั้นตอนการประมาณเชิงเส้นของสัญญาณเสียงพูด โดยวิธีออโตคอร์รีเลชัน เพื่อ นำมาใช้ในการวิเคราะห์เสียงพูดมนุษย์
3. วิเคราะห์การหาค่าพารามิเตอร์ LPC และ เซปสตรัม (Cepstrum) ของเสียง
4. สร้างแบบอ้างอิง (Codebook) โดยใช้การจัดระดับเวกเตอร์ (Vector Quantization) ของพารามิเตอร์ที่ได้จากการประมาณเชิงเส้น
5. สร้างแบบจำลองของ มาร์คอฟ (Hidden Markov Models) เพื่อใช้ในการรู้จำเสียงพูด
6. เขียนโปรแกรมตามรูปแบบการวิเคราะห์เสียงที่ได้ศึกษามา
7. ศึกษาและเขียนโปรแกรมติดต่อกับ Sound Card เพื่อนำสัญญาณเสียงเข้ามาวิเคราะห์
8. ศึกษาและเขียนโปรแกรมติดต่อกับ Modem เพื่อนำผลที่ได้จากการวิเคราะห์ไปสั่งงาน Modem
9. ทดสอบและปรับปรุงโปรแกรม
10. ทำเอกสารประกอบโครงงาน

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 ทฤษฎีการรู้จำเสียง

จากการศึกษารูปแบบการรู้จำเสียงพูดแบบคำเดี่ยว (Isolated word recognition) โครงสร้างพื้นฐานของระบบการจดจำเสียงพูด แบ่งออกเป็น 3 ส่วน ได้แก่

1. การวิเคราะห์เสียงเบื้องต้น (Speech Pre - processing)
2. การควอนไตเซชั่นแบบเวคเตอร์ (Vector Quantizer)
3. การกำหนดแบบจำลองการรู้จำเสียง (Pattern matching)

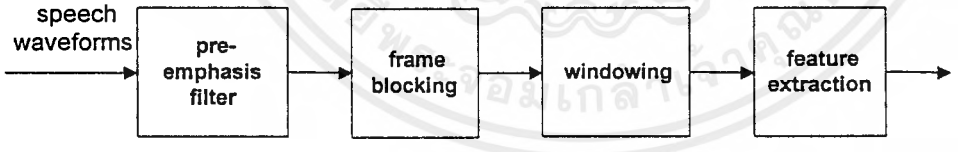


รูปที่ 2.1 บล็อกไดอะแกรมของการรู้จำเสียง

สัญญาณเข้าเป็นคลื่นเสียง ส่วนสัญญาณออกเป็นคำที่วิเคราะห์ได้จากสัญญาณเข้า ซึ่งผ่านขั้นตอนต่างๆ ดังต่อไปนี้

2.1.1 การวิเคราะห์เสียงเบื้องต้น (Speech Pre - processing)

ในการวิเคราะห์สัญญาณเสียง เราจำเป็นต้องมีการเตรียม ข้อมูลในการวิเคราะห์ก่อน ซึ่งขั้นตอนในการเตรียมสัญญาณเข้าเป็นดังรูปที่ 2.2



รูปที่ 2.2 แสดงขั้นตอนการเตรียมสัญญาณในการวิเคราะห์

การเตรียมสัญญาณเสียงในการวิเคราะห์ มีขั้นตอนดังต่อไปนี้

การพรีเอมฟาซีส (Preemphasis)

เนื่องจากสัญญาณเสียงพูดของมนุษย์ จะมีองค์ประกอบส่วนใหญ่อยู่บริเวณความถี่ต่ำเมื่อเทียบกับความถี่ปฏิบัติงานไม่เกิน 5 kHz ดังนั้น เพื่อทำให้อัตราส่วนของสัญญาณเสียงต่อสัญญาณรบกวน (SNR) มีค่าคงที่ตลอดช่วงความถี่ปฏิบัติงานจึงทำการพรีเอมฟาซีส โดยเน้นให้ความถี่สูงมีขนาดสูงขึ้น

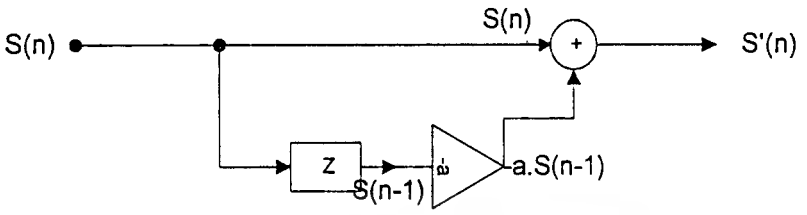
การพรีเอมฟาซีส คือ การกรอง สัญญาณด้วยวงจรกรองความถี่สูงผ่าน (High Pass Filter) ซึ่งนิยมใช้วงจรอันดับหนึ่ง ซึ่งเป็นตัวกรองเชิงเลขแบบง่ายที่สุด มีรูปแบบสมการดังนี้

$$y(n)=a_0x(n)-b_1y(n-1) \quad (2.1)$$

มีฟังก์ชันถ่ายโอนเป็น

$$H(z)=1-a.z^{-1} \; ; \; 0.9<a<1.0 \quad (2.2)$$

สมมติว่าสัญญาณเดิมเป็น S(n) เมื่อประมาณค่าสัญญาณแล้วจะเป็น S'(n)



รูปที่ 2.3 วงจรกรองความถี่สูงผ่าน

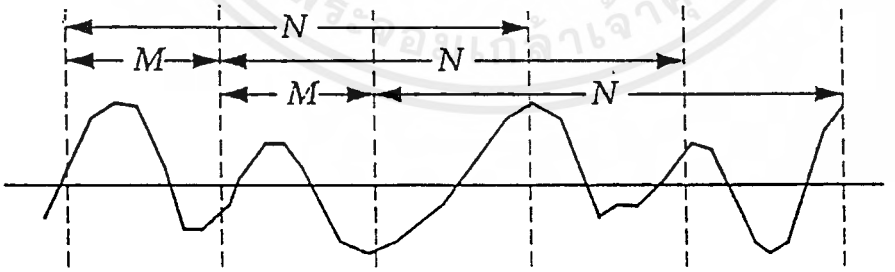
จะได้ว่า

$$S'(n) = S(n) - a.S(n-1) \quad (2.3)$$

ยิ่งค่า a เข้าใกล้ 1 เท่าใด ความถี่สูงก็จะถูกขยายมากขึ้นเท่านั้น ค่า a ที่นิยมสำหรับใช้ในการหาพารามิเตอร์ของ LPC คือ ค่า15/16 = 0.9375

การแบ่งช่วงสัญญาณ (Frame Blocking)

สัญญาณที่ผ่านการพรีเอมฟาซีสแล้ว ถูกตัดมาวิเคราะห์ทีละเฟรม เฟรมละ N ตัวอย่างสัญญาณ ในการวิเคราะห์ ทีละช่วงของแต่ละ N ตัวอย่างสัญญาณนั้น จะวิเคราะห์ โดย เลื่อนไปเป็นระยะ M ช่วงสัญญาณ จนหมดสัญญาณเสียงที่นำมาวิเคราะห์ ดังรูปที่ 2.4 ซึ่ง ระยะ N ที่เหมาะสมจะต้องมีค่าน้อยกว่า M จะทำให้วิเคราะห์สัญญาณได้แม่นยำ แต่ถ้าค่า N น้อยเกินไปการวิเคราะห์จะล่าช้า



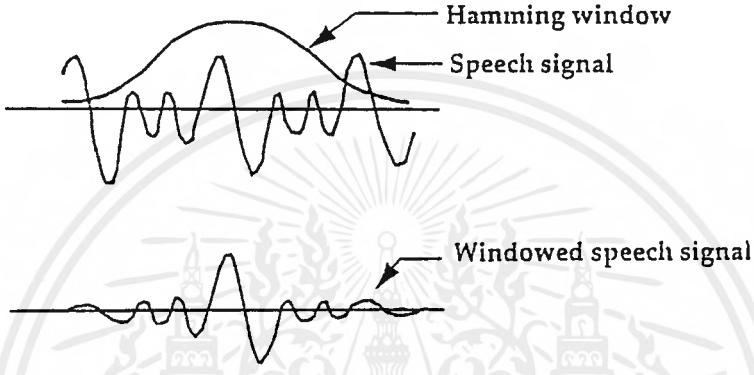
รูปที่ 2.4 การแบ่งช่วงของสัญญาณ

เนื่องจากสัญญาณเสียงมีการเปลี่ยนแปลงคุณสมบัติตามเวลา แต่ถ้าแบ่งเสียงออกเป็นเฟรมสั้นๆ เสียงจะมีการเปลี่ยนแปลงคุณสมบัติน้อยมาก ทำให้การวิเคราะห์ทำได้ง่ายขึ้น ในการทดลองนี้จึงทำการแบ่งสัญญาณเสียงออกเป็นเฟรมๆ ละ M แซมเปิ้ล (sample) และทำการเลื่อนเฟรมไปเรื่อยๆ จนหมด

การวินโดว์ (Windowing)

เนื่องจากที่เราตัดสัญญาณเสียงมาวิเคราะห์ทีละเฟรมทำให้ที่ขอบของเฟรมเกิดความไม่ต่อเนื่องของสัญญาณ ถ้ามองในโดเมนความถี่สูง ก็จะมีความถี่สูงเกิดขึ้น ดังนั้นเพื่อที่จะลดองค์ประกอบทางความสูงเหล่านี้ เราจะคูณด้วยฟังก์ชันวินโดว์เพื่อลดความไม่ต่อเนื่องของสัญญาณที่ขอบ และไม่ทำให้สเปคตรัมของสัญญาณที่ช่วงความถี่ต่ำเปลี่ยนไปมากนัก ในที่นี้จะใช้ฟังก์ชันวินโดว์แฮมมิ่ง (Hamming window function) ซึ่งนิยามโดยสมการ

$$w(n) = 0.54 - 0.46\cos(2\pi n/(N-1)) \quad \text{เมื่อ } n = 0,1,2, ..., N-1 \qquad (2.4)$$



รูปที่ 2.5 แสดงวินโดว์แบบแฮมมิ่ง

ในการวิเคราะห์เสียงโดยใช้ฟังก์ชันวินโดว์ จะพบว่าสัญญาณเสียง ที่ผ่านการกรองโดยวินโดว์นั้นจะมีการแกว่งขึ้นลงมากน้อยขึ้นกับช่วงเวลาของวินโดว์ (ความกว้างของวินโดว์) คือถ้าช่วงเวลาของการวินโดว์สั้นจะมีการแกว่งขึ้นลงอย่างรวดเร็ว และถ้าช่วงเวลาของการวินโดว์มากจะมีการแกว่งขึ้นลงช้าๆ ดังนั้นในการเลือกช่วงเวลาของการวินโดว์ ต้องให้อยู่ในช่วงที่เหมาะสม คือไม่ให้เอาท์พุทของสัญญาณแกว่งช้าหรือเร็วเกินไป อยู่ในช่วงระหว่าง 10 - 30 ms

เมื่อคูณกับฟังก์ชันวินโดว์แล้ว ก็จะได้

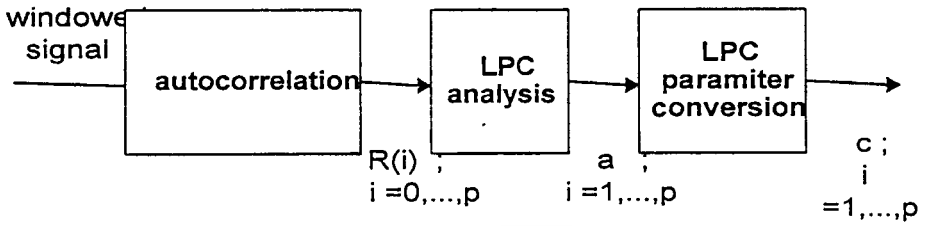
$$x'(n)=w(n)x(n) \qquad (2.5)$$

การวิเคราะห์หาคุณลักษณะของเสียง (Feature extract)

การวิเคราะห์หาคุณลักษณะของเสียง เป็นเทคนิคในการแปลงข้อมูลที่มีอยู่มากมาย ให้เป็นส่วนเล็กๆ ซึ่งส่วนเล็กๆนี้จะแสดงคุณสมบัติของคลื่นเสียงนั้นๆ โดยจะใช้วิธีการป้อนสัญญาณผ่านวงจรกรอง (Filter) และการประมาณเชิงเส้น (Linear Predictive Coding)

หลักการพื้นฐานของการประมาณเชิงเส้นคือ การประมาณสัญญาณเสียงจากผลรวมเชิงเส้นของสัญญาณเสียงในอดีต โดยอาศัยเกณฑ์กำลังสองของสัญญาณความคลาดเคลื่อนมีค่าต่ำสุดในการหาสัมประสิทธิ์การประมาณเชิงเส้น กล่าวคือหลังจากการทำพรีเอมฟาซีสและวินโดว์ครบทุกตัวในหนึ่งเฟรมแล้ว

จะทำการคำนวณออโตคอรีเลชั่น เพื่อที่จะทำการ หาค่า สัมประสิทธิ์ LPC และอัตราขยาย G มีขั้นตอนแสดงดังรูปที่ 2.6



รูปที่ 2.6 แสดงขั้นตอนในการวิเคราะห์หาคุณลักษณะของเสียง

การคำนวณออโตคอรีเลชั่น (Autocorrelation)

สมมติว่าสัญญาณเดิมเป็น S(n) การประมาณ ค่าสัญญาณเป็น S'(n) ดังนั้นสามารถอธิบายการประมาณเชิงเส้นด้วยสมการต่อไปนี้

$$s'(n)=\sum_{k=1}^{p} \alpha_k \, s(n--k) \qquad (2.6)$$

เมื่อ α_k เป็นค่าคงที่ เรียกวิธีการนี้ว่าการประมาณเชิงเส้นอันดับ p โดยมีเงื่อนไขว่า ค่า α_k ที่ใช้ในการประมาณจะต้องทำให้ ผลรวมของกำลังสองของความคลาดเคลื่อน $\{s(n)-s'(n)\}^2$ มีค่าน้อยที่สุด นั่นคือ $\sum e^2(n) = \sum \{s(n) - s'(n)\}^2$ มีค่าต่ำที่สุด ซึ่งจะใช้การประมาณเชิงเส้นวิธีออโตคอรีเลชั่น (Autocorreiation Method) หรือ วิธีอัตตสัมพันธ์

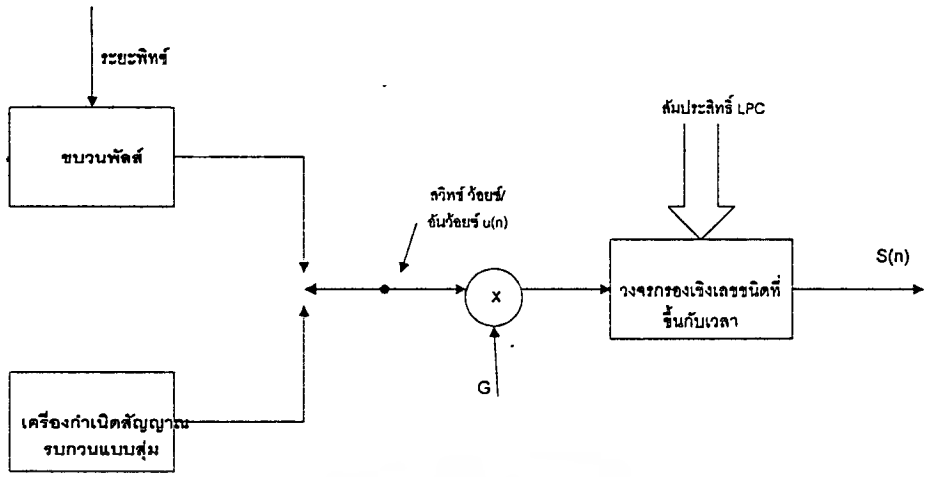
การคำนวณออโตคอรีเลชั่น เป็นวิธีการ หาสัมประสิทธิ์ LPC โดยฟังก์ชั่นออโตคอรีเลชั่น ซึ่งเป็นการเปรียบเทียบสัญญาณกับสัญญาณของตัวมันเองที่ถูกเลื่อนไปตามแกนเวลา ที่ใช้วิธีนี้ เนื่องจากเป็นการคำนวณที่มีการแก้สมการที่น้อยกว่าวิธีอื่นๆ และมีความแน่นอนในด้านเสถียรภาพ อีกทั้งมีการเก็บข้อมูลที่น้อยกว่า

คุณสมบัติของฟังก์ชั่นออโตคอรีเลชั่น

1. เป็นฟังก์ชั่นคู่ R(k) = R(-k)

2. จะให้ค่าสูงสุดเมื่อเป็นการเปรียบเทียบสัญญาณของตัวมันเองที่ตำแหน่งทางแกนเวลาเดียวกัน คือ R(0) = max

จากหลักการพื้นฐานของการประมาณเชิงเส้นและแบบจำลองระบบสร้างสัญญาณเสียง สามารถเขียนบล็อกไดอะแกรมการทำการประมาณเชิงเส้นมาสร้างสัญญาณเสียงพูดได้ดังรูปที่ 2.7



รูปที่ 2.7 บล็อกไดอะแกรมแสดงโมเดลการสร้างสัญญาณเสียงพูดอย่างง่าย

จากรูปที่ 2.7 สามารถเขียนสมการได้เป็น

$$s(n)=G*u(n)+\sum_{k=1}^{p} a_k\, s(n-k) \qquad (2.7)$$

การประมาณเชิงเส้นโดยใช้สัมประสิทธิ์ $\{ \alpha_k \}$ คือ

$$s'(n)=\sum_{k=1}^{p} \alpha_k\, s(n-k) \qquad (2.8)$$

ดังนั้น ความคลาดเคลื่อน คือ

$$e(n) = s(n) - s'(n)$$

$$e(n) = s(n)- \sum_{k=1}^{p} \alpha_k\, s(n-k) \qquad (2.9)$$

ฟังก์ชั่นถ่ายโอนระหว่าง e(n) และ s(n) คือ

$$A(z)=E(z)/S(z)$$

$$=1-\sum_{k=1}^{p} \alpha_k z^{-k} \qquad (2.10)$$

จากสมการ (2.7) และ (2.9) จะเห็นได้ว่า ถ้า $\{ \alpha_k \} = \{a_k\}$ แล้ว

$$e(n)=G*u(n) \qquad (2.11)$$

ดังนั้น ค่าผลรวมของกำลังสองของความคลาดเคลื่อน

$$E_n = \sum_m e_n^2 \ (m)$$

$$E_n = \sum_m [s(m) - s'(m)]^2 \qquad (2.12)$$

โดยที่ n คือ ช่วงที่ n ของสัญญาณที่ใช้คำนวณ เพราะฉะนั้นเพื่อให้ได้ค่า E_nต่ำที่สุดจะต้องมีเงื่อนไขว่า

$$\frac{\partial E_n}{\partial \alpha_i} = 0 \quad \text{เมื่อ } i=1,2,3,\ldots,p$$

จากสมการ (2.12)

$$\frac{\partial E_n}{\partial \alpha_i} = -2s_n(m-i)\sum_m [s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k)] \quad \text{เมื่อ i=1,2,3,....,p}$$

$$= -2[\sum_m s(m)s(m-i) - \sum_{k=1}^{p}\sum_m \alpha_k s(m-k)s(m-i)]$$

$$\frac{\partial E_n}{\partial \alpha_i} = 0 \quad \text{ก็ต่อเมื่อ}$$

$$\sum_{k=1}^{p} \alpha_k \sum_m s_n(m-k)s_n(m-i) = \sum_m s_n(m)s_n(m-i) \quad \text{เมื่อ i=1,2,3.....,p} \quad (2.13)$$

ถ้าเรากำหนดให้ $\phi_n(i,j)$ = $\sum s_n(m-k)\, s_n(m-i)$ เพราะฉะนั้น

$$\sum_{k=1}^{p} \alpha_k \phi_n(i,j) = \phi_n(i,0) \quad (2.14)$$

โดยสมการ (2.12)-(2.13)จะได้ว่า

$$E_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \alpha_k \sum_m s_n(m)s_n(m-k)$$

และจาก $\phi_n(i,j)$ = $\sum s_n(m-k)\, s_n(m-i)$

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k)$$

$$E_n = \phi_n(0,0) - \phi_n(0,k) \quad (2.15)$$

สมมุติว่าใน 1 เฟรม ของสัญญาณ ที่ตัดมาคำนวณมี N ตัวอย่าง คือ $S_n(0)$, $S_n(1)$, $S_n(2)$,..., $S_n(N-1)$ ในที่นี้เราให้ $S_n(m) = 0$ เมื่อ $m < 0$ หรือ $m > N-1$ เพราะฉะนั้น

$$\phi_n(i,j) = \sum_m s_n(m-k)s_n(m-i)$$

$$= \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k) \qquad 0 \le k \le p, 1 \le i \le p$$

ให้ $$R_n(k) = \sum_m^{N-1-k} s_n(m)s_n(m+k) \qquad \text{เมื่อ k=0,1,2,....,p} \quad (2.16)$$

ดังนั้น จากสมการ (2.5) และ (2.16) จะได้ว่า

$$R_n(k) = \sum_{m=0}^{N-1-k} x'(m)x'(m+k) \quad (2.17)$$

จากสมการที่ (2.14) จะได้ว่า

$$\sum_{k=1}^{p} \alpha_k R_n(|i-k|) = R_n(i) \qquad \text{เมื่อ i=1,2,3,....,p}$$

จากสมการ เขียนให้อยู่ในรูปเมตริกซ์ได้เป็น

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

หรือ

$$R_n \cdot \alpha = r_n \tag{2.18}$$

เมื่อ $R_n = \begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix}$, $\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix}$ และ $r_n = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$

การหาพารามิเตอร์ LPC

พารามิเตอร์ LPC ได้แก่ สัมประสิทธิ์ α และอัตราการขาย G

เมื่อได้ค่า $R_n(0)$, $R_n(1)$, $R_n(2)$, ..., $R_n(p)$ จากสมการ (2.18) แล้ว ก็สามารถหาค่า α นั่นคือ

$$\alpha = R_n^{-1} \cdot r_n$$

และจากสมการ (2.7) และ (2.9) จะได้ว่า

$$e(n)=G^*u(n)$$

$$\therefore E_n = \sum_{m=0}^{N-1} e^2(m) = G \sum_{m=0}^{N-1} u^2(m) \tag{2.19}$$

จาก สมการ (2.15) จะได้ว่า

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k)$$

$$= R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k) \tag{2.20}$$

และจากสมการ (2.19) เราสามารถหาค่า G โดยตรงจาก

$$G^2 = \frac{R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k)}{\sum_{m=0}^{N-1} u^2(m)} \tag{2.21}$$

ขั้นตอนนี้ ก็ได้วิเคราะห์สัญญาณไปหนึ่งเฟรมแล้ว โดยในหนึ่งเฟรมนี้จะได้ค่าสัมประสิทธิ์ทั้งหมด 12 ค่า (p=12) ; $\alpha_1, \alpha_2, \ldots, \alpha_p$

การเปลี่ยนพารามิเตอร์ LPC เป็น สัมประสิทธิ์เซปสตรัม

หลังจากที่หาสัมประสิทธิ์ LPC และ Gain ใน 1 เฟรมแล้ว จะเปลี่ยนให้เป็น สัมประสิทธิ์เซปสตรัม เนื่องจากการรู้จำเสียงพูดนั้น สัมประสิทธิ์เซปสตรัมนี้เป็นพารามิเตอร์ที่มีลักษณะน่าเชื่อถือได้ดีกว่าสัมประสิทธิ์ LPC ทั้งยังมีความสัมพันธ์ใกล้ชิดกับการรับรู้เสียงตามความรู้สึกของมนุษย์โดยแท้จริง สัมประสิทธิ์เซปสตรัมสามารถหาได้โดยตรงจากสัมประสิทธิ์ LPC ดังนี้

$C_0 = \ln G$ เป็นสัมประสิทธิ์ตัวแรก ซึ่งเป็นเกน

$Q \approx \frac{3}{2} p$ โดย p = 12 ดังนั้น Q = 18 รวมกับเกน (C_0)

จะได้ว่า สัมประสิทธิ์เซปสตรัมใน 1 เฟรม = 19 ตัว

$$C_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) C_k a_{m-k} \quad ,1 \le m \le p$$

$$C_m = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) C_k a_{m-k} \quad ,m > p$$

การเวทค่าพารามิเตอร์ (Parameter Weighting)

เนื่องจากสัมประสิทธิ์เซปสตรัมที่ได้นั้น ช่วงลำดับต้นๆ และลำดับท้ายๆ ของเฟรมที่นำมาวิเคราะห์จะเกิดความคลาดเคลื่อนมากกว่าบริเวณส่วนอื่น เพราะฉะนั้นจึงทำการถ่วงน้ำหนักเพื่อลดค่าความคลาดเคลื่อนดังกล่าวนี้ ด้วยฟังก์ชั่นเวทติ้ง ดังนี้คือ

$$W_m = [1 + \frac{Q}{2} \sin(\frac{\pi m}{Q})] \quad ,1 \le m \le Q$$

จะได้พารามิเตอร์สุดท้าย คือ

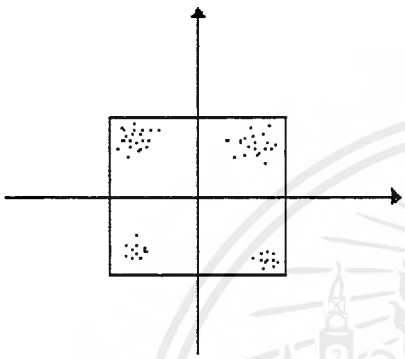
$$C'_m = C_m * W_m$$

จากนั้นก็พิจารณาให้ครบทุกเฟรมของข้อมูล เมื่อพิจารณาเรียบร้อยแล้วก็จะนำไปจัดกลุ่มเสียง และสร้างแบบจำลองเสียงเพื่อใช้ในการเปรียบเทียบต่อไป

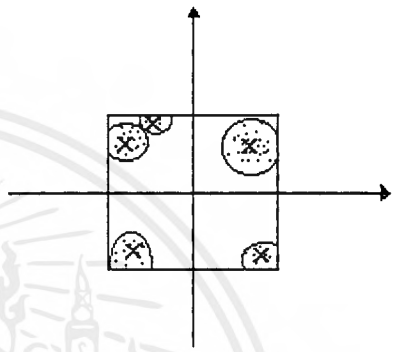
2.1.2 **การจัดระดับเวคเตอร์** (Vector Quantization)

เวคเตอร์ควอนไตซ์เซชัน เป็นวิธีการลดไดเมนชั่น (Dimension) หรือจำนวนของข้อมูล เวคเตอร์อินพุท หรือ เซตเทรนนิ่ง หรือ พารามิเตอร์ที่ได้จากขั้น LPC จะถูกเลือกมากลุ่มหนึ่งซึ่งใช้เป็นตัวแทนของข้อมูลจำนวนหนึ่งหรือเรียกว่าการหา Codebook อินพุทที่เข้ามาจะถูกทำการเปรียบเทียบกับ codebook ที่มีอยู่ โดยจะ

พิจารณาว่าอินพุทที่เข้ามานั้นห่างจาก codebook ใดน้อยที่สุด อินพุทดังกล่าวจะถูกแทนด้วยเวคเตอร์โค้ด (index) นั้น อินพุททุกตัวที่เป็นสมาชิกของเวคเตอร์โค้ดใดๆ จะถูกนำมาหาจุดศูนย์กลางร่วมใหม่ และนำจุดศูนย์กลางที่ได้นี้ไปทำการหาความคลาดเคลื่อนกับสมาชิกทุกตัว ถ้าค่าความคลาดเคลื่อนที่ได้มีค่ามากกว่าค่าที่กำหนดไว้ค่าหนึ่งหรือค่าที่ยอมรับได้ ก็จะนำศูนย์กลางใหม่นั้นไปเป็น codebook แทน และจะทำการจัดกลุ่มอินพุทเข้ากับ codebook ใหม่ที่ได้และหาความคลาดเคลื่อนอีกครั้งทำอย่างนี้ซ้ำๆ จนกระทั่งค่าความคลาดเคลื่อนมีค่าน้อยถึงค่าที่ยอมรับได้ ก็จะถือว่าได้ codebook ที่ดีที่สุดที่จะเป็นตัวแทนของอินพุททั้งหมด จะสังเกตได้ว่าทุกครั้งที่มีการหา codebook ใหม่นั้น ค่าความคลาดเคลื่อนที่ได้จะมีค่าลดลงทุกครั้งด้วย

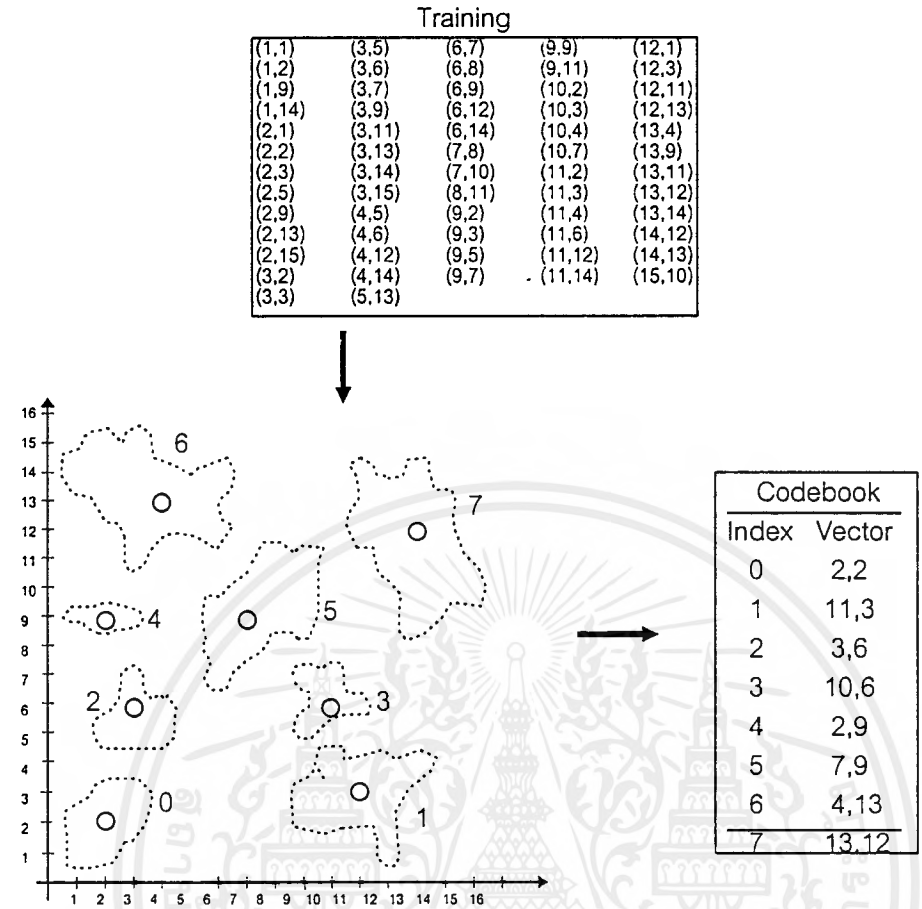


รูปที่ 2.8 แสดงการกระจายเฟรมของเสียงพูดแต่ละจุดแทนเฟรมของเสียง

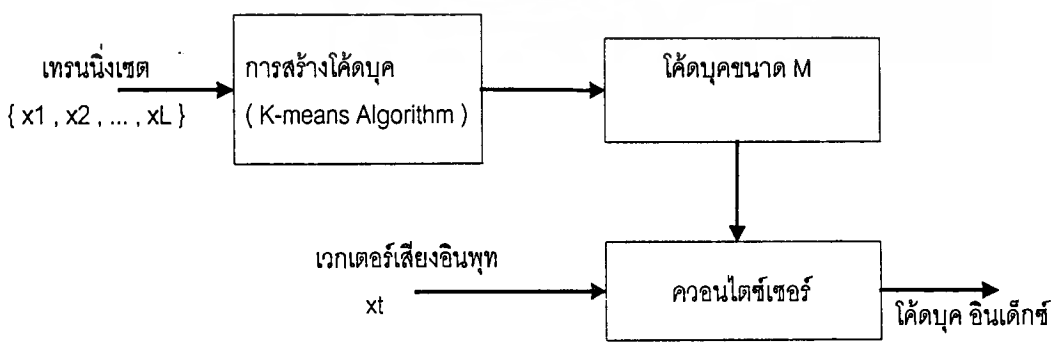


รูปที่ 2.9 การรวมกลุ่มของเฟรมเสียงเพื่อสร้างโค้ดบุค X

ตัวอย่างเวกเตอร์ควอนไตซ์เซชั่น สมมุติให้เวกเตอร์แต่ละตัวมี 2 มิติ และทำการหา Codebook ขนาด 8 เวกเตอร์ทั้งหมดจะถูกจัดเข้าไปในกลุ่มของ Codebook ต่างๆ แล้วทำการหาจุดศูนย์ใหม่โดยการเฉลี่ยค่าเวกเตอร์สมาชิกทุกตัวที่อยู่ในกลุ่มเดียวกัน ผลที่ได้คือ Codebook 8 ตัวที่เป็นตัวแทนของเวกเตอร์ทั้งหมด



รูปที่ 2.10 แสดงตัวอย่างการหาโค้ดบุค



รูปที่ 2.11 บล็อกไดอะแกรมของเวคเตอร์ควอนไตซ์

- **การทำงานของการควอนไตซ์แบบเวคเตอร์** แบ่งเป็น 2 ขั้นตอนดังนี้

1. การสร้างโค้ดบุค (codebook) **โดยใช้วิธี** K-means

จากขั้นตอนการประมาณเชิงเส้นของเสียงตัวอย่างจำนวนมากจะได้เทรนนิ่งเซตซึ่งประกอบด้วยเวคเตอร์สเปคตรัมจำนวน L เฟรม ; $x = \{ x_i , 1 \leq i \leq L \}$ เฟรมละ P มิติ ; $x = [x_1, x_2, \ldots, x_p]$ แล้วนำข้อมูลที่ได้มาทำการสร้างเป็นกลุ่มของแบบอ้างอิง

ในระบบการรับรู้เสียงพูดแบบต่างบุคคล จะใช้แบบอ้างอิงของคำหนึ่งๆ จากผู้พูดจำนวนมาก เพื่อที่จะได้ครอบคลุมถึงความแปรปรวนต่างๆ ที่เกิดขึ้นระหว่างผู้พูดแต่ละคน เนื่องจากถ้าใช้แบบอ้างอิงจำนวนมาก เวลาที่ใช้ในการตอบสนองจะมาก เนื้อที่ในหน่วยความจำสำรองที่ใช้เก็บแบบอ้างอิงจะเพิ่ม และเมื่อเพิ่มแบบอ้างอิงไปจนถึงระดับหนึ่ง ความถูกต้องในการรับรู้จะเริ่มคงที่ ดังนั้นจึงมีการจัดกลุ่มของแบบอ้างอิงใหม่เพื่อให้ได้แบบอ้างอิงที่เหมาะสม และสามารถใช้เป็นตัวแทนของแบบอ้างอิงที่มีอยู่ทั้งหมดได้ อัลกอริทึมที่ใช้ได้แก่ K-means Algorithm ซึ่งขั้นตอนที่ใช้ในการสร้างโค้ดบุคมีดังนี้

1) **นำเทรนนิ่งเซตมาใช้ในการสร้างโค้ดบุค**

ขนาดโค้ดบุคของการควอนไตซ์แบบเวคเตอร์ คือ $M = 2^B$ เวคเตอร์ (B - bit codebook) และเพื่อที่จะหาเซตของ M โค้ดบุคที่ดีที่สุด จำนวนเวคเตอร์อินพุทจะต้องมากกว่าขนาดโค้ดบุคม่าก ($L >> M$)

2) **การสุ่มค่าเริ่มต้น**

การสุ่มค่าเริ่มต้น เป็นวิธีหนึ่งในการออกแบบโค้ดบุค ซึ่งคือ การเลือกค่าเริ่มต้นของโค้ดบุค เรียกโค้ดบุค

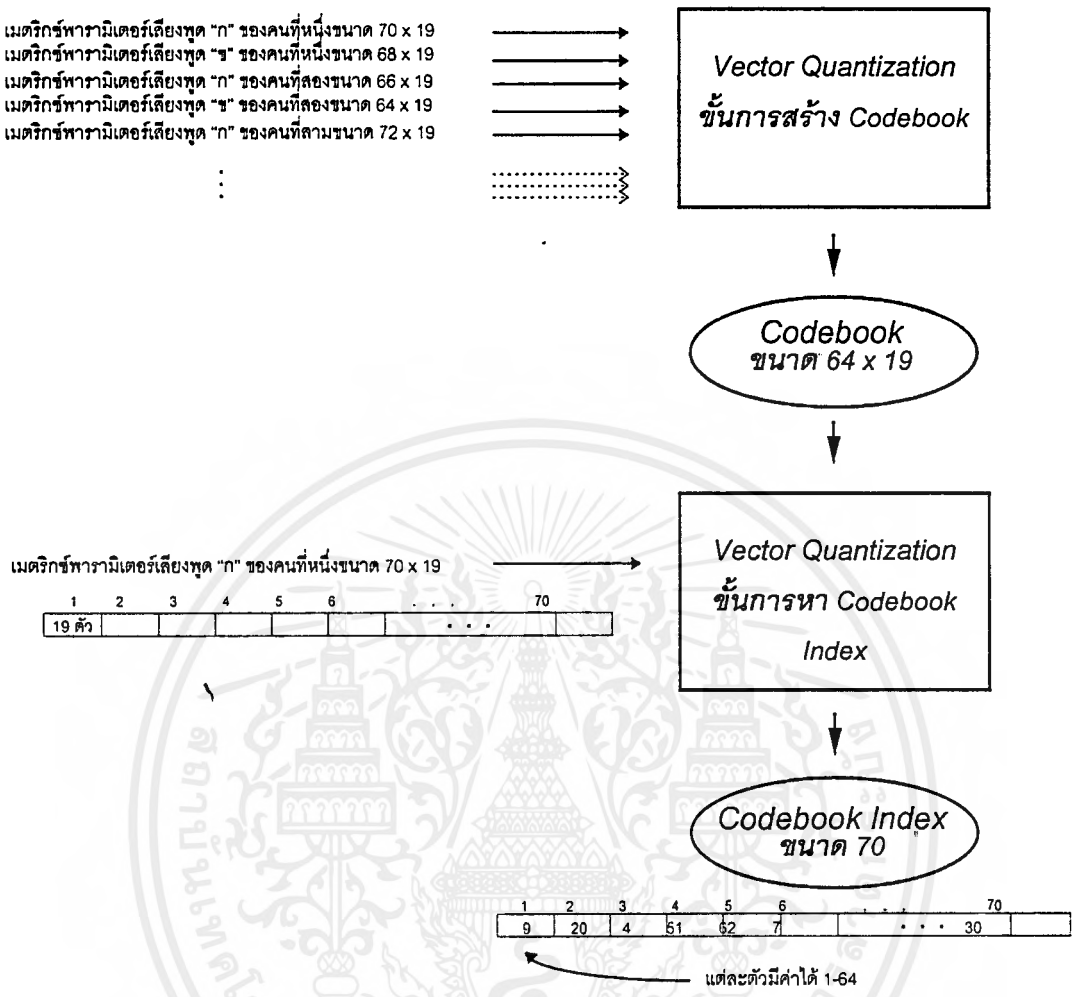
ที่ได้จากการสุ่มค่าเริ่มต้นนี้ว่า แรนดอมโค้ดบุค (random codebook) ถึงแม้วิธีนี้จะไม่ใช่วิธีที่ดีนัก แต่โค้ดบุคที่ได้จากการสุ่มก็ให้ผลเป็นที่ยอมรับ

3) **การหาความคลาดเคลื่อน**

การวัดความคลาดเคลื่อน เป็นส่วนที่จำเป็นและเป็นประโยชน์ต่อการออกแบบโค้ดบุค สมการ ทางพีชคณิตที่ใช้ในการหาระยะทาง มีหลายวิธี แต่วิธีที่นำมาใช้ คือ การหาความคลาดเคลื่อนกำลังสองรวม (Total square error) ซึ่งเป็นวิธีการคำนวณที่ง่ายและรวดเร็ว

ถ้าสัญญาณมี P มิติ สามารถหาระยะห่างระหว่างสัญญาณอินพุท (x) กับเวคเตอร์โค้ด (y) โดยสมการ

$$d(v_1,v_2) = \| v_1 - v_2 \|^2 = \sum_{i=0}^{k-1} (x_i - y_i)^2 \qquad (2.14)$$



รูปที่ 2.12 ขั้นตอนของเวคเตอร์ควอนไตเซชั่น

4) **การจัดกลุ่ม** (classification) **และการหาจุดศูนย์กลางของกลุ่ม** (center cluster)

การจัดกลุ่มเป็นการแบ่งเวคเตอร์อินพุทเข้าไปตามกลุ่มต่างๆ ของแรนดอมโค้ดบุค โดยพิจารณาระยะทาง หรือความคลาดเคลื่อนน้อยที่สุด ของแต่ละเวคเตอร์อินพุท x กับเวคเตอร์ โค้ดบุค y ซึ่งเป็นโค้ดบุค

จากนั้นจะทำการหาค่าเฉลี่ยของแต่ละกลุ่ม เพื่อเป็นค่ากลางของกลุ่มนั้นๆ จะได้

$$\overline{Y} = \frac{1}{L}\sum_{i=1}^{L} x_i$$

$\overline{Y}$ เป็นจุดศูนย์กลางซึ่งเป็นเวคเตอร์ที่อยู่ตรงกลางของ $\{x_i\}_{i=1}^{L}$ ซึ่งแต่ละมิติจะไม่ขึ้นแก่กันหมายความว่า แต่ละ y_k เป็นค่ากลางของ $\{x_i\}_{i=1}^{L}$

ทำ 2 ขั้นตอนนี้ซ้ำ จนกว่าจะเกิดการลู่เข้า (convergent) โดยความคลาดเคลื่อนรวมจะต่ำกว่าค่าหนึ่งๆ ซึ่งค่าความคลาดเคลื่อนรวมจะลดลงทุกครั้งที่มีการคำนวณซ้ำใหม่ จึงขึ้นกับค่าที่กำหนดว่าต้องการให้

ความคลาดเคลื่อนรวมน้อยเท่าใด ค่ากลางดังกล่าวของแต่ละกลุ่มจะถูกเก็บเป็นเวคเตอร์โค้ดจะได้ว่า y เป็นควอนไตซ์ของค่า x

$$y = q(x)$$

โดย q(.) เป็นโอเปอร์เรเตอร์ของควอนไตซ์ y ถูกเรียกว่าเอาท์พุทเวคเตอร์ของค่า x โดย y เป็นค่าใดค่าหนึ่งใน $Y = \{ y_i , 1 \leq i \leq M \}$ โดย $y_i = [y_{i1} \; y_{i2} \; ... \; y_{iP}]$ Y เป็นเซ็ตของโค้ดบุด M เป็นขนาดของโค้ดบุค และ $\{ y_i \}$ เป็นเซ็ตของเวคเตอร์โค้ด y_i อาจเรียกว่าเป็นโค้ดอ้างอิง และ M อาจเรียกว่าจำนวนระดับขั้น จะทำการแบ่งเวคเตอร์ x ไปใน M เซล $\{ C_i , 1 \leq i \leq M \}$ เมื่อ x อยู่ในเซล C_i

$$q(x) = Y_i \quad \text{ถ้า} \quad x \in C_i$$

2. ขั้นตอนการเปรียบเทียบ

เวคเตอร์ควอนไตเซชั่นที่ใช้เพื่อการออกแบบการรับรู้เสียงพูดนั้น มีจำนวนควอนไตเซอร์ M ตัวซึ่งหมายถึง มี M ระดับเสียงเพื่อการรับรู้ แต่ละระดับเสียงพิจารณามาจากเซตของข้อมูลเทรนนิ่ง ซึ่ง M เป็นดัชนีระดับ แต่ละเซตของเทรนนิ่งในแต่ละระดับจะเก็บเสียงที่อยู่ในระดับเดียวกัน

เมื่อมีเสียงที่เราไม่ทราบ (unknown) ; x_t เข้ามา จะเป็นอินพุทเข้าไปยังทุกๆ ควอนไตเซอร์ ค่าดัชนีระดับ (index) ที่ถูกเลือก จะเป็นระดับที่มีความคลาดเคลื่อน เฉลี่ยน้อยที่สุด D (c) เมื่อ l = 1,2,...,M ซึ่งความคลาดเคลื่อนเฉลี่ยนั้นหาได้จากการวัดระยะทาง โดยใช้วิธีการหาความคลาดเคลื่อนกำลังสองรวม

2.1.3 การสร้างแบบจำลอง (Pattern Matching)

หมายถึง การกำหนดรูปแบบขึ้นเพื่อนำมาเปรียบเทียบกับสิ่งที่เราไม่ทราบ โดยการจัดกลุ่มของสัญญาณที่ไม่รู้จัก(unknow signal)ซึ่งเป็นสัญญาณที่ไม่คงที่ให้อยู่ในกลุ่มใดกลุ่มหนึ่ง วิธีที่เลือกใช้ในการสร้างและหารูปแบบของคำพูดที่เหมือนมีอยู่สองแนวทาง

- Dynamic time warping (DTW)
- Hidden Markov Models (HMM)

วิธีที่เลือกใช้ในการสร้างแบบจำลองคือ Hidden Markov Model (HMM) ซึ่งเป็นวิธีการสร้างแบบจำลองที่ถูกนำมาประยุกต์ใช้ในการรู้จำเสียงพูด แบบจำลองมาร์คอฟ แบ่งออกเป็น 2 ประเภทคือ แบบต่อเนื่อง (Continuous) และแบบไม่ต่อเนื่อง (Discrete-time) ในที่นี้จะเลือกใช้แบบไม่ต่อเนื่องเพราะเป็นวิธีการที่ซับซ้อนน้อยกว่าและใช้ได้ดีกับคำพูดสั้นๆ

Hidden Markov Model(HMM)

HMM เป็นวิธีการที่ใช้ทฤษฎีความน่าจะเป็นมาอธิบายการเกิดเหตุการณ์ของลำดับของเหตุการณ์ 2 ลำดับ คือ สเตท(state) และค่าปรากฏ(observation) โดยเหตุการณ์ที่สังเกตุเห็นจะเห็นได้เพียงเอาท์พุทของแต่ละสเตทในลักษณะที่เป็นค่าปรากฏเท่านั้น จะไม่ทราบแน่ชัดว่าอยู่ที่สเตทใด

ส่วนประกอบของแบบจำลอง HMM

1. N คือจำนวนสเตทในแบบจำลอง โดยสามารถย้ายจากสเตทหนึ่งไปยังอีกสเตทหนึ่งได้ เราให้เซตของสเตทเป็น { 1,2,...,N } และสเตทที่เวลา t ใดๆเป็น q_t

2. M คือจำนวนของค่าปรากฏต่อหนึ่งสเตท แทนด้วยสัญญลักษณ์ $V = \{ V_1, V_2, ..V_M \}$

3. $A = \{a_{ij}\}$ คือความน่าจะเป็นในการเปลี่ยนสเตทที่ $a_{ij} = P\{ q_t=j \mid q_{t-1}=i \}$ เมื่อ $1 \le i,j \le N$

4. $B = \{b_j(k)\}$ คือ ความน่าจะเป็นของการเกิดค่าปรากฏที่สเตทใดๆ $b_j(k)=P\{ O_t=V_k|q_t=j \}$ เมื่อ $j=1,2,...,N$

5. $\pi = \{ \pi_i \}$ คือ ความน่าจะเป็นที่แต่ละสเตทจะเป็นสเตทเริ่มต้น เมื่อ $\pi_i = P(q_i$ ที่เวลา $t=1)$

โครงสร้างของแบบจำลองHMM

แบ่งตามลักษณะการเปลี่ยนสเตท(transition)ของเมตริกซ์ A

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

1. แบบ Egordic Model หรือ Fully Connected Model แบบจำลองนี้ทุกสเตทสามารถเปลี่ยนสเตทไปยังสเตทอื่นๆได้ทุกๆสเตท ดังรูปที่ 2.10(a) เป็นตัวอย่างของแบบจำลองที่มี N=4

2. แบบ Left - Right Model หรือ Bakis Model แบบจำลองนี้ การเปลี่ยนสเตทจะเปลี่ยนจากซ้ายไปขวา มีคุณสมบัติการเปลี่ยนสเตทดังนี้

2.1 $a_{ij}=0, j<1$ หมายความว่า เมื่อผ่านสเตทใดไปแล้วจะไม่มีการย้อนกลับมายังสเตทนั้น

2.2 $\pi_i = \{ 0$ เมื่อ $i \neq 1; 1$ เมื่อ $i=1\}$ หมายความว่า ลำดับของสเตทต้องเริ่มต้นที่สเตทที่ 1 สเตทที่เหลือจึงมีความน่าจะเป็นที่จะเป็นสเตทเริ่มต้นเท่ากับศูนย์

Left-Right Model นี้มีกฎข้อบังคับการเปลี่ยนสเตทดังนี้

$a_{ij}=0$ เมื่อ $i > i+\Delta i$ โดยค่าของ $\Delta i=2$ หมายความว่า การเปลี่ยนสเตทจะสามารถกระโดดข้ามเปลี่ยนไปยังสเตทข้างหน้าได้ไม่เกิน 2 สเตท

จะได้เมตริกซ์การเปลี่ยนสเตทเป็น

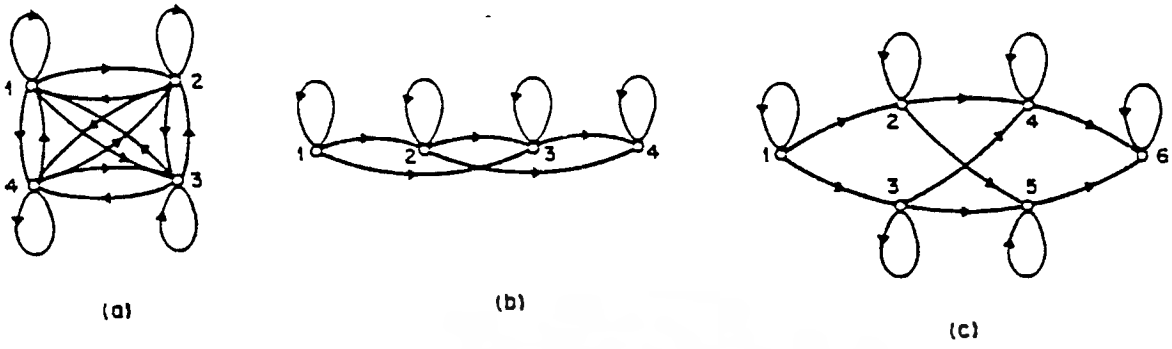
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

จะเห็นว่าสเตทสุดท้ายมีสัมประสิทธ์การเปลี่ยนสเตทเป็น $a_{NN}=1$

$a_{Ni} = 0$ เมื่อ $i<N$

แบบจำลองนี้เหมาะกับสัญญาณที่มีการเปลี่ยนแปลงอย่างต่อเนื่องเช่น คำพูด

3. แบบ Parallel Left-Right Model มีคุณสมบัติการเปลี่ยนสเตทคล้ายแบบที่ 2 แต่มีความยืดหยุ่นมากกว่า ดังรูปที่ 2.13c)



รูปที่ 2.13 แสดงแบบจำลองต่างๆของHMM

ปัญหาของHMM

ปัญหาของ HMM มี 3 ข้อซึ่งต้องใช้อัลกอริทึมวิธีต่างๆในการคำนวณเพื่อแก้ปัญหานี้

ปัญหาที่1 เมื่อลำดับของค่าปรากฏ $O=\{O_1,O_2,...,O_T\}$ และมีโมเดล $\lambda=(A,B,\pi)$ เราจะคำนวณหาค่า$P(O|\lambda)$ ของลำดับของค่าปรากฏได้อย่างไร

ปัญหาที่2 เมื่อมีลำดับของค่าปรากฏ $O=\{O_1,O_2,...,O_T\}$ และแบบจำลอง $\lambda=(A,B,\pi)$ เราจะหาลำดับสเตท $q=\{q_1q_2...q_T\}$ ที่เหมาะสมในการให้ค่าปรากฏนั้นได้อย่างไร

ปัญหาที่3 จะหาแบบจำลอง $\lambda=(A,B,\pi)$ ที่ให้ค่า$P(O|\lambda)$ มากที่สุดได้อย่างไร

ลำดับของค่าปรากฏที่ใช้ปรับค่าพารามิเตอร์ A,B และπเพื่อให้ได้แบบจำลองที่ดีที่สุดนั้นเรียกว่าลำดับเทรนนิ่ง(training sequence)

การคำนวณเพื่อแก้ปัญหาของHMM

1.**การแก้ปัญหาที่1** เป็นการคำนวณว่าแบบจำลองλให้ความน่าจะเป็นที่จะได้ลำดับค่าปรากฏมากน้อยเพียงใด มีวิธีการเพื่อช่วยแก้ปัญหาโดยใช้กระบวนการต่อไปนี้

1.1 กระบวนการไปข้างหน้า (Forward Procedure)

เมื่อกำหนดให้ตัวแปรไปข้างหน้า(forward variable)

$$\alpha_t(i)=P(O_1,O_2,...,O_T,q_t=i|\lambda)$$

หมายถึง ความน่าจะเป็นของการเกิดลำดับค่าปรากฏ $O_1,O_2,...,O_T$ ที่จะอยู่ที่สเตท i ณ เวลา t โดยมีแบบจำลองเป็นλโดยสามารถหา $\alpha_t(i)$ ได้ดังนี้

1.1.1 การเริ่มต้น (initializa tion) เมื่อกำหนด $\alpha_t(i)=\pi_i b_i(O_t)$ที่เวลาเริ่มต้น t=1 และเหตุการณ์เริ่มต้น O_1 เมื่อ $1\le i\le N$

1.1.2 การเหนี่ยวนำ (induction)

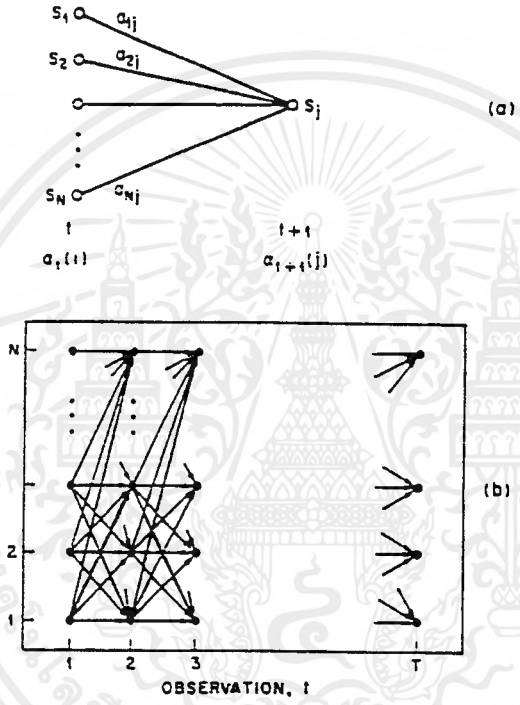
$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}]\, b_j(o_{t+1}) \qquad (2.15)$$

เมื่อ$1 \le t \le T-1$ และ$1 \le j \le N$ หมายถึง ความน่าจะเป็นของสเตท j ที่เวลา t+1 ได้มาจากสเตท i ที่เป็นไปได้ถึง N สเตท ที่เวลา t ดังรูปที่ 2.14

1.1.3 การสิ้นสุด (termination)

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \qquad (2.16)$$

ความน่าจะเป็นของลำดับค่าปรากฏ O ได้จากผลรวมของ $\alpha_t(i)$ จากทุกๆ สเตท เมื่อ $1 \le j \le N$



รูปที่ 2.14กระบวนการไปข้างหน้า

1.2 กระบวนการย้อนกลับ (Backward Procedure)

เมื่อกำหนดให้ตัวแปรย้อนกลับ(backward variable) $\beta_T(i) = P(o_{t+1}o_{t+2}....o_T, q_t=i\ |\lambda)$ หมายถึงความน่าจะเป็นของลำดับค่าปรากฏส่วนหลัง จากเวลา t+1 ไปจนจบ โดยกำหนดว่าต้องอยู่ที่สเตท i ความน่าจะเป็นของลำดับค่าปรากฏส่วนหลัง จากเวลา ที่เวลา t และมีแบบจำลองเป็น λ เราจะคำนวณหา $\beta_T(i)$ ได้ดังนี้

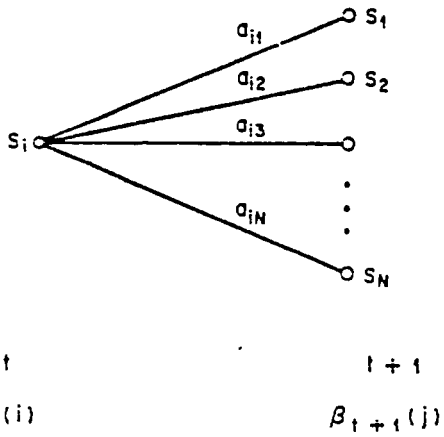
1. การเริ่มต้น (initialization)

$\beta_T(i) = 1$ เมื่อ $1 \le j \le N$

2. การเหนี่ยวนำ(induction)

$$\beta_T(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\beta_{t+1}(i) \qquad (2.17)$$

เมื่อ $t=T-1,T-2,...1$, $1 \le i \le N$



รูปที่ 2.15 กระบวนการถอยหลัง

รูปที่ 2.15 แสดงถึงค่าปรากฏที่จะอยู่ที่สเตท i ที่เวลา t โดยคำนึงถึงลำดับค่าปรากฏจากเวลา t+1 ซึ่งต้องพิจารณาสเตท j ที่จะเป็นไปได้ทั้งหมด ณ เวลา t+1 โดยจะขึ้นอยู่กับค่า a_{ij} และ $b_j(o_{t+1})$

2. การแก้ปัญหาที่ 2 เพื่อหาลำดับสเตทที่เหมาะสม

เราจะใช้**วิธี** วิทเทอร์บี อัลกอริทึม (Viterbi Algoritym) เพื่อหาลำดับสเตทที่ดีที่สุด ณ เวลา t หนึ่ง ๆ เมื่อกำหนดลำดับเหตุการณ์ $O = (o_1o_2....o_t)$ โดยนิยามให้

$$\delta_t(i) = \max_{q1,q2,....,qi} P[q_1q_2....q_{t-1},\ q_t=i,\ o_1o_2....o_t|\lambda]$$

หมายถึง ความน่าจะเป็นสูงสุดของเส้นทาง (path) ณ เวลา t ซึ่งเริ่มนับจากเหตุการณ์ที่เวลาเริ่มต้น จนถึงเวลา t ที่สเตท i และ โดยการอาศัยคุณสมบัติการเหนี่ยวนำ (induction) เราจะได้

$$\delta_{t+1}(i) = [\max_i \delta_t(i)\, a_{ij}]\, b_j(o_{t+1})$$

เราสามารถหาลำดับสเตทที่ดีที่สุดได้โดยใช้กระบวนการต่อไปนี้ เมื่อกำหนดให้ $\psi_t(i)$ เป็น อาเรย์ (array)

1. การเริ่มต้น (initialization)

$$\delta_t(i) = \pi_i b_i(o_1) \text{ เมื่อ } 1 \le j \le N$$

$$\psi_t(i) = 0$$

2. การย้อนกลับ (recursion)

$$\delta_t(j) = [\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}]\, b_j(o_t) \text{ เมื่อ } 2 \le t \le T,\ 1 \le j \le N$$

$$\psi_1(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i)\, a_{ij}] \quad \text{เมื่อ } 2 \le t \le T,\ 1 \le j \le N$$

3. การสิ้นสุด (termination)

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \max_{1 \le i \le N} [\delta_T(i)]$$

4. เส้นทางเดินย้อนกลับ (Path backtracking)

$$q_t^* = \psi_{t+1}(q^*_{t+1}) \text{ เมื่อ } t=T-1,T-2,...,1$$

3.การแก้ปัญหาที่3 เพื่อหาโมเดลที่จะให้ผลตามลำดับค่าปรากฏหนึ่งๆโดยเลือกค่าพารามิเตอร์ A,B,π ที่ดีที่สุด โดยใช้ กระบวนการทำซ้ำ (Iterative) วิธีที่เราเลือกใช้ คือวิธี บาม-เวลช์ (Baum-Welch) หรือ EM (Expectation-Maximization)

เมื่อนิยามให้

1. $\gamma_t(i) = P(q_t=i \mid O,\lambda)$

หมายถึง ความน่าจะเป็นที่จะอยู่ที่สเตท i ณ เวลา t โดยกำหนดลำดับเหตุการณ์ O และแบบจำลอง λ ให้ สามารถแสดงค่า $\gamma_t(i)$ ได้ดังนี้

$$\gamma_t(i) = \frac{p(O,q_t=i\mid\lambda)}{p(O\mid\lambda)}$$

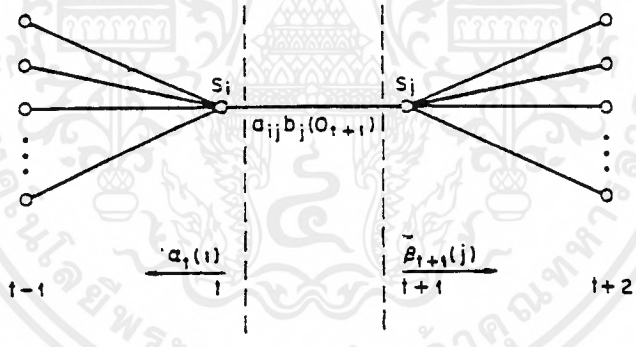
$$= \frac{p(O,q_t=i\mid\lambda}{\sum_{i=1}^{N} P(O,q_t=i\mid\lambda)}$$

เนื่องจาก $P(O,q_t=i \mid\lambda)$ มีค่าเท่ากับ $\alpha_t(i)\beta_t(i)$ จึงได้

$$\gamma_t(i) = \frac{\alpha(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha(i)\beta_t(i)} \qquad (2.18)$$

2. $\varepsilon_t(i,j) = P(q_t=i, q_{t+1}=j \mid O,\lambda)$

หมายถึง ความน่าจะเป็นที่จะอยู่ที่สเตท i ที่เวลา t และสเตท j ที่เวลา t+1 เมื่อกำหนดแบบจำลอง และลำดับค่าปรากฏให้



รูปที่ 2.16 แสดงลำดับการคำนวณการเกิดค่าปรากฏร่วมซึ่งจะอยู่ที่สเตท i ที่เวลา t และอยู่ที่สเตท j ที่เวลา t+1

ซึ่งจากนิยามของตัวแปรไปข้างหน้าและตัวแปลย้อนกลับ สามารถนำมาสัมพันธ์กับ $\varepsilon_t(i,j)$ ได้ดังนี้

$$\varepsilon_t(i,j) = \frac{P(q_t=i, q_t+1=j, O\mid\lambda)}{P(O\mid\lambda)}$$

$$= \frac{\alpha(i)aijbj(o_t+1)\beta_t+1(j)}{P(O\mid\lambda)}$$

$$= \frac{\alpha(i)aijbj(o_t+1)\beta_t+1(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha(i)aijbj(o_t+1)\beta_t+1(j)} \qquad (2.19)$$

และจะได้ความสัมพันธ์ของ $\gamma_t(i)$ กับ $\varepsilon_t(i,j)$ ดังนี้

$$\gamma_t(i) = \sum_{j=1}^{N} \varepsilon_t(i,j) \qquad (2.20)$$

และ

$\sum_{t=1}^{T-1} \gamma_t(i)$ = จำนวนของการเปลี่ยนสเตทออกจากสเตท i ในลำดับค่าปรากฏ O

$\sum_{t=1}^{T-1} \varepsilon_t(i,j)$ = จำนวนของการเปลี่ยนสเตทจากสเตท i ไป j ในลำดับค่าปรากฏ O

ดังนั้นสามารถหาค่าพารามิเตอร์ได้ดังนี้

$$\pi_i' = \gamma_1(I) \text{ เมื่อ } 1 \le i \le N \qquad (2.21)$$

$$a_{ij}' = \frac{\sum_{t=1}^{T-1} \varepsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad (2.22)$$

$$b_j(k) = \frac{\sum_{t=1, O_t=v_k}^{T} n(j)}{\sum_{t=1}^{T} n(j)} \qquad (2.23)$$

จากกระบวนการข้างต้น ถ้าเราจะคำนวณซ้ำๆ โดยให้ $\lambda'=(A',B',\pi')$ แทน $\lambda=(A,B,\pi)$ซึ่งเป็นแบบจำลองเริ่มต้นแล้ว จะทำให้ความน่าจะเป็นของการเกิดลำดับค่าปรากฏ O ดีขึ้น จนกระทั่งถึงจุดวิกถต ซึ่งเราจะได้จุดวิกฤตของฟังก์ชันความน่าจะเป็นในกรณีที่ $\lambda'=\lambda$ หรือถ้า λ' มีความน่าจะเป็นมากกว่าแบบจำลอง λ ในลักษณะที่ $P(O|\lambda')>P(O|\lambda)$ นั่นก็คือ เราก็จะได้ แบบจำลอง λ' ใหม่ ที่น่าจะทำให้เกิดลำดับค่าปรากฏ 0 ได้ดีกว่า

การปรับค่าพารามิเตอร์ของ HMM

1. การสเกลลิ่ง (Scaling)

เนื่องจาก $\alpha_t(i)$ จะประกอบด้วยผลรวมของเทอมจำนวนมาก ซึ่งก็คือ

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(O_s) \right)$$

และเนื่องจากแต่ละเทอมของ a และ b มีค่าน้อยกว่า 1 อยู่แล้ว เมื่อพิจารณาผลรวมของการคูณค่ายิ่งน้อยลงเรื่อยๆ แสดงว่าเมื่อ t มากขึ้น แต่ละเทอมของ $\alpha_t(i)$ จะเข้าสู่ศูนย์ ทำให้Dynamic Range ของการคำนวณ $\alpha_t(i)$ เกิน Range ของคอมพิวเตอร์ ทำให้ค่าที่ได้ไม่ถูกต้อง จึงได้มีการสเกลลิ่งขึ้นเพื่อทำให้ $\alpha_t(i)$ อยู่ภายใน Dynamic Range ของคอมพิวเตอร์ การสเกลลิ่งทำได้โดยการคูณ $\alpha_t(i)$ โดยสัมประสิทธิ์

การสเกลลิ่ง ซึ่งสัมประสิทธิ์นี้ไม่ขึ้นกับ i การสเกลลิ่ง $B_t(i)$ ก็เช่นเดียวกัน หลังการคำนวณค่า การสเกลลิ่ง ก็จะตัดกันหมดไปเอง พิจารณาจาก สมการ

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (2.24)$$

เมื่อเราให้ $\alpha_t(i)$ แทน α ที่ยังไม่ได้สเกลลิ่ง

$\hat{\alpha}_t(i)$ เป็น α ที่สเกลลิ่งแล้ว

$\hat{\hat{\alpha}}_t(i)$ แทนเวอร์ชันของ α ก่อนการสเกลลิ่ง

เมื่อ t = 1 จะได้ $\hat{\alpha}_t(i) = c_1 \alpha_1(i)$

เมื่อ $c_1 = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$

เมื่อ $2 \leq t \leq T$ คำนวณ $\hat{\hat{\alpha}}_t(i)$ จาก สมการ (2.17) ในเทอมของ $\hat{\alpha}_t(i)$ ค่าก่อน

$$\hat{\hat{\alpha}}_t(i) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_i(O_t) \qquad (2.25)$$

เมื่อ สัมประสิทธิ์ การสเกลลิ่ง เป็น

$$c_t = \frac{1}{\sum_{i=1}^{N} \hat{\hat{\alpha}}(i)}$$

เมื่อให้ $\hat{\alpha}_t(i) = c_t \hat{\hat{\alpha}}_t(i)$

จากสมการ (2.25) จะเขียนได้ว่า

$$\alpha_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_j(O_t)} \qquad (2.26)$$

และโดยการเหนี่ยวนำ จะได้

$$\hat{\alpha}_{t-1}(j) = (\prod_{\tau=1}^{t-1} c_\tau) \alpha_{t-1}(i)$$

จะได้ว่า

$$\hat{\alpha}_t(i) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(i) (\prod_{\tau=1}^{t-1} c_\tau) a_{ij} b_i(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{t-1}(i) (\prod_{\tau=1}^{t-1} c_\tau) a_{ij} b_i(O_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)} \qquad (2.27)$$

นั่นคือจะสเกล $\alpha_t(i)$ ได้โดยหารด้วยผลรวมของ $\alpha_t(i)$ ทั้งหมด และสเกล $B_t(i)$ ด้วยค่าเดียวกันนี้

ในเทอมของการสเกลนี้สมการ (2.24) จะเป็น

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j)} \tag{2.28}$$

โดยแต่ละ $\hat{\alpha}_t(i), \hat{\beta}_{t+1}(j)$ จะได้เป็น

$$\hat{\alpha}_t(i) = \left[\prod_{s=1}^{t} c_s\right] \alpha_t(i) = C_t \alpha_t(i) \tag{2.29}$$

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=1}^{T} c_s\right] \beta_{t+1}(j) = D_{t+1} \beta_{t+1}(j) \tag{2.30}$$

ดังนั้นสมการ (2.28) จะเขียนได้เป็น

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)} \tag{2.31}$$

ซึ่งเทอม $C_t D_{t+1}$ จะเขียนได้ในเทอม

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T \tag{2.32}$$

ซึ่งไม่ขึ้นกับเวลา t ดังนั้น $C_t D_{t+1}$ จะถูกตัดทิ้ง ทั้งเศษและส่วนของสมการ (2.31) ซึ่งทำให้ได้สูตรการคำนวณซ้ำ ๆ (reestimate) เดิมกลับคืนมา กระบวนการ สเกลลิ่ง ดังกล่าวนี้สามารถใช้ได้กับสัมประสิทธิ์ β และ π ในการสเกลลิ่งนี้จะทำให้การคำนวณค่า $P(O|\lambda)$ เปลี่ยนไป เราจะไม่สามารถหาได้จากการรวมกับของเทอม $\hat{\alpha}_T(i)$ แต่จะหาจากคุณสมบัติ

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1$$

ดังนั้นจะได้

$$\prod_{t=1}^{T} c_t P(O|\lambda) = 1$$

$$P(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{2.34}$$

ทำให้อยู่ในรูป log ของ P เพื่อไม่ให้เกิน dinamic range ของคอมพิวเตอร์

$$\log[P(O|\lambda)] = \sum_{t=1}^{T} \log c_t \tag{2.35}$$

2. **ลำดับของค่าปรากฏหลายเหตุการณ์** (Multiple Observation Sequence)

ในการใช้แบบจำลองแบบ Left-Right นั้น การเทรนแบบจำลองต้องใช้หลาย ๆ เหตุการณ์ของลำดับค่าปรากฏเข้ามาเทรน เพื่อให้ได้ค่าพารามิเตอร์ที่ถูกต้องมากขึ้น

ถ้าให้เซตของ v ลำดับค่าปรากฏเป็น

$O=[O^{(1)},O^{(2)},.....,O^{(k)}]$

เมื่อ $O^{(v)}=O_1^{(v)}O_2^{(v)}.....O_{T_v}^{(v)}$ เป็นลำดับค่าปรากฏของเหตุการณ์ที่ v โดยให้แต่ละเหตุการณ์ เป็นอิสระต่อกัน จะได้

$$P(O|\lambda)=\prod_{v=1}^{V} P$$

$$=\prod P_v$$

นำเอาจำนวนเหตุการณ์ของการเกิดค่าปราฏแต่ละเหตุการณ์มารวมกันจะได้สูตรหาa_{ij},$b_j(k)$ เป็น

$$\overline{a_{ij}} = \frac{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1}^{T_v-1}\alpha_t^v(i)a_{ij}b_j(O_{t+1}^v)\beta_{t+1}^v(j)}{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1}^{T_v-1}\alpha_t^v(i)\beta_t^v(i)} \tag{2.36}$$

$$\overline{b_j}(k) = \frac{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1\ni O=k}^{T_v-1}\alpha_t^v(i)\beta_t^v(i)}{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1}^{T_v-1}\alpha_t^v(i)\beta_t^v(i)} \tag{2.37}$$

ส่วน π_iไม่ต้องคำนวณเนื่องจาก $\pi_1=1$, $\pi_i=0$,$i\neq1$

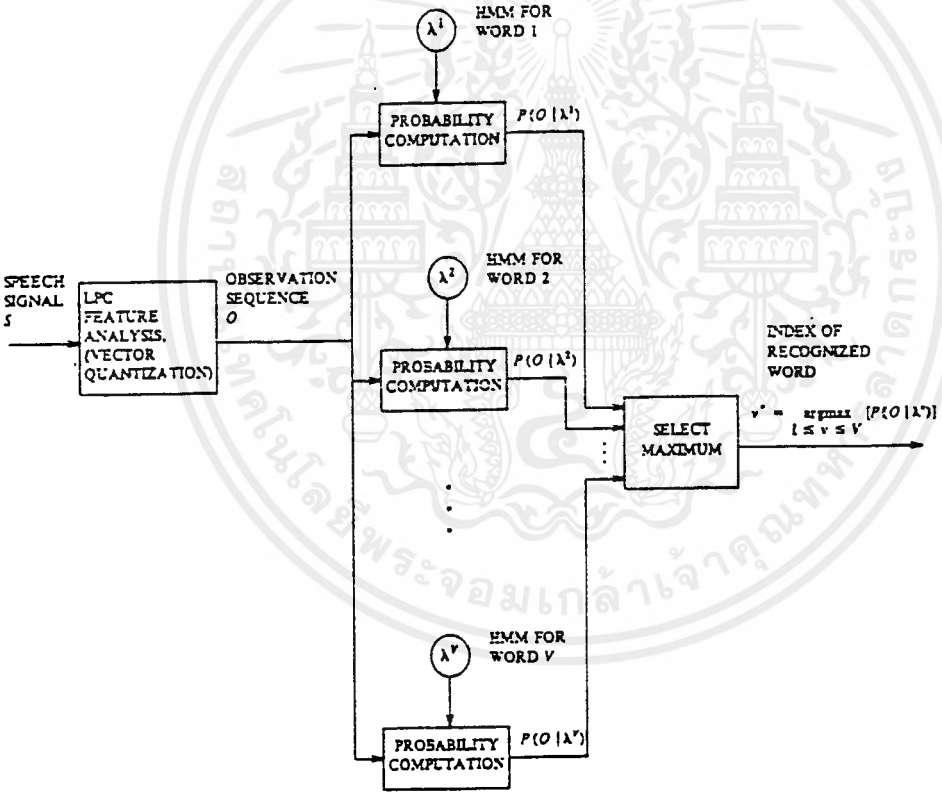
จะได้การสเกลลิ่ง ที่เหมาะสมของสมการ(2.36)-(2.37)

$$\bar{a}_{ij} = \frac{\sum_{v=1}^{V}\sum_{t=1}^{T_v-1}\hat{\alpha}_t^v(i)a_{ij}b_j(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)}{\sum_{v=1}^{V}\sum_{t=1}^{T_v-1}\sum_{j=1}^{N}\hat{\alpha}_t^v(i)a_{ij}b_j(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)} \quad (2.38)$$

$$\bar{b}_j(k) = \frac{\sum_{v=1}^{V}\sum_{t=1,O_t=k}^{T_v-1}\hat{\alpha}_t^v(i)a_{ij}b_j(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)}{\sum_{v=1}^{V}\sum_{t=1}^{T_v-1}\sum_{j=1}^{N}\hat{\alpha}_t^v(i)a_{ij}b_j(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)} \quad (2.39)$$

ระบบแบบจำลองมาร์คอฟ

เมื่อเรามีคำศัพท์อยู่ V คำ ในการทำการรู้จำได้ เราจะต้องทำการสร้างแบบจำลองของคำ แต่ละคำที่แตกต่างกัน คำแต่ละคำจะมี ลำดับเทรนนิ่งที่ได้มาจากคุณลักษณะเฉพาะของคำศัพท์นั้น ๆ การที่เราจะรู้จำคำพูดได้ต้องทำได้ตามบล็อคไดอะแกรมดังต่อไปนี้



รูปที่ 2.17 บล็อกไดอะแกรมการรู้จำคำโดดด้วยแบบจำลองของมาร์คอฟ

1. เมื่อมีคำศัพท์อยู่ V คำ เราต้องสร้างแบบจำลองมาร์คอฟ :λ_v ของแต่ละคำนั้น นั่นคือการหาค่า (A,B,π) ที่เหมาะสมกับลำดับเทรนนิ่งของคำนั้นๆ

2. ในการจะรู้จำคำศัพท์แต่ละคำ เราจะทำการหาค่า $P(O|\lambda)$ ของทุกๆ แบบจำลอง แล้วเลือกแบบจำลองที่มีค่าความน่าจะเป็นในการเกิดค่าปรากฏสูงสุด คือ

$$V^* = \arg\max P(O|\lambda_v)$$

คำศัพท์ที่สอดคล้องกับแบบจำลองดังกล่าวนี้ จะเป็นคำเดียวกับคำศัพท์ที่เราต้องการจะรู้จำนั่นเอง โดยขั้นตอนการคำนวณหาค่าความน่าจะเป็น จะใช้วิธีวิทเทอร์บี ดังที่ได้กล่าวแล้วในตอนต้น

2.2 การเชื่อมต่อส่วนการรู้จำเสียงกับระบบโทรศัพท์

การสื่อสารอาจเกิดขึ้นได้โดยตรงด้วยการเชื่อมต่ออุปกรณ์สองตัวด้วยสายสัญญาณ หรือโดยอ้อมด้วยสื่อกลาง ซึ่งมักจะเป็นโทรศัพท์ โดยใช้โมเด็ม (modem) เพื่อแปลงสัญญาณที่ปลายด้านหนึ่งให้เป็นสัญญาณที่เหมาะสมกับการส่งผ่านสายโทรศัพท์

ระบบเครือข่ายโทรศัพท์ที่ใช้กันอยู่ในปัจจุบันนั้นประกอบด้วยสายโทรศัพท์ซึ่งโยงใยไปทั่วและเชื่อมต่อกับอุปกรณ์สวิตชิ่งส่วนกลาง ในขณะที่ผู้ใช้ยกหูโทรศัพท์ วงจรโทรศัพท์ก็จะเชื่อมเข้าสู่ระบบสวิตชิ่งส่วนกลางทันที วงจรสวิตชิ่งนี้จะส่งสัญญาณที่เรียกว่า ไดอัลโทน หลังจากที่ผู้ใช้ได้รับสัญญาณนี้ ก็หมายความว่าระบบพร้อมที่จะรับหมายเลขโทรศัพท์ปลายทางแล้ว และทุกครั้งที่กดหมายเลขปลายทาง วงจรสวิตช์จะรู้ว่าผู้ใช้กดหมายเลขใดบ้าง ก็จะเชื่อมต่อวงจรไปที่หมายเลขปลายทางตามต้องการทันที

มาตรฐานการเชื่อมต่อ RS-232-C มาตรฐาน RS-232-C แบ่งการเชื่อมต่อออกเป็นสองลักษณะ คือ การต่อกับเทอร์มินัล (DTE : Data Terminal Equipment) และการต่อกับอุปกรณ์สื่อสารข้อมูล (DCE : Data Communication Equipment) ซึ่งในกรณีปกตินั้น DCE จะต้องต่อเข้ากับ DTE เสมอ เช่น การต่อโมเด็มเข้ากับเครื่อง PC โดยเครื่อง PC จะเป็นอุปกรณ์ DTE และ โมเด็มเป็นอุปกรณ์ DCE การเชื่อมต่อตามมาตรฐานนี้โดยปกติจะใช้คอนเน็กเตอร์รูปตัว D ชนิด 25 ขา กำหนดให้ปลายสายสัญญาณด้านหนึ่งเป็นตัวผู้ใช้ต่อกับอุปกรณ์ DCE และปลายสายอีกด้านจะเป็นคอนเนคเตอร์ตัวเมียใช้ต่อกับอุปกรณ์ DTE แต่อาจมีมาตรฐานแบบใหม่เช่น คอนเนคเตอร์แบบ 9 ขา สัญญาณที่เกี่ยวข้องกับการสื่อสารอนุกรมมีการแยกประเภทดังนี้

1. สัญญาณข้อมูล

วงจรส่งผ่านข้อมูลมีสองวงจร คือ วงจรที่ทำหน้าที่ส่งข้อมูลจาก DTE ไป DCE และอีกวงจรก็คือวงจรรับข้อมูลจาก DCE เพื่อส่งไป DTE รูปแบบสัญญาณไฟฟ้าที่ใช้แทนข้อมูลเป็นรูปสัญญาณสี่เหลี่ยมที่ถูกสร้างจากสัญญาณไฟกระแสตรงประมาณ 3 ถึง 25 โวลต์ แทนข้อมูล "0" และ -3 ถึง -25 แทนข้อมูล "1" และช่วงแรงดัน -3 ถึง 3 โวลต์ นั้นจะเป็นช่วงระดับแรงดันที่ใช้ในการแบ่งแยกระดับสัญญาณระหว่างสถานะ "0" และ "1"

2. วงจรควบคุม

มีหน้าที่สร้างสัญญาณควบคุมต่างๆ ขึ้นมา เพื่อทำให้เครื่อง PC และ โมเด็ม ทราบสถานะในการทำงานของกันและกัน โดยที่สัญญาณควบคุมจะมีลักษณะทางกายภาพเช่นเดียวกับสัญญาณข้อมูล แต่โมเด็มส่วนใหญ่ไม่ได้ใช้วงจรควบคุมทุกวงจร ซึ่งสัญญาณควบคุมต่างๆมีดังนี้

- Request To Send (RTS) และ Clear To Send (CTS)

เป็นส่วนที่ใช้สำหรับควบคุมการส่งผ่านข้อมูลระหว่างเครื่อง PC และโมเด็ม โดยโมเด็มจะส่งสัญญาณ CTS สถานะ "ON" ให้แก่ PC เมื่อโมเด็มพร้อมที่จะรับข้อมูล และเครื่อง PC ก็จะให้สัญญาณ RTS สถานะ "ON" เมื่อ PC พร้อมที่จะรับข้อมูล

- Data Terminal Ready (DTR)และ Data Set Ready (DSR)

สัญญาณ DTR จะใช้เป็นสิ่งบอกโมเด็มให้ทราบว่าเครื่อง PC นั้นกำลังอยู่ในสถานะที่พร้อมจะติดต่อกับโมเด็มหรือไม่ และในกรณีเดียวกันสัญญาณ DSR นั้นจะใช้เป็นตัวบอกให้เครื่อง PC ทราบว่าโมเด็มพร้อมจะติดต่อหรือไม่ โดยที่สัญญาณ DSR จะออยู่ในสถานะ "ON" ก็ต่อเมื่อโมเด็มได้รับสัญญาณ DTR แล้ว

- Carrier Detect และ Ring Indicator

CD เป็นสัญญาณที่บอก PC ให้ทราบว่าโมเด็มกำลังเชื่อมต่อกับโมเด็มเครื่องอื่นๆ และกำลังได้รับสัญญาณพาหะจาโมเด็มปลายทาง ส่วนสัญญาณ RI เป็นการบอกเครื่อง PC ให้ทราบว่า มีสัญญาณกริ่งโทรศัพท์เรียกเข้ามาที่โมเด็ม ซึ่งโมเด็มส่วนใหญ่ก็มีวงจรตอบรับโทรศัพท์อัตโนมัติอยู่แล้ว จึงไม่จำเป็นต้องใช้สัญญาณ RI แต่ในบางโปรแกรม ก็อาจจะใช้สัญญาณ RI เป็นตัวกำหนดให้เริ่มทำการรันโปรแกรมอื่นๆ ได้

การเชื่อมต่อโมเด็มเข้ากับคอมพิวเตอร์ โมเด็มภายนอก (external modem) เป็นอุปกรณ์อนุกรมตัวหนึ่ง โมเด็มทั่วไปจะประกอบด้วยอุปกรณ์เพิ่มเติมในการใช้งานดังนี้

1. สายสัญญาณและคอนเนคเตอร์ RS-232-C
2. สายโทรศัพท์และคอนเนคเตอร์ RJ - 11
3. แหล่งจ่ายกำลังไฟฟ้าให้แก่โมเด็ม

โมเด็มจะอยู่ในภาวะใดภาวะหนึ่ง คือภาวะคำสั่ง (command mode) หรือ ภาวะออนไลน์ (on-line mode) ขณะที่อยู่ในภาวะคำสั่งสามารถสั่งงานโมเด็มได้จากคอมพิวเตอร์ เช่น สั่งให้โมเด็มทำการหมุนหมายเลขโทรศัพท์ เมื่อมีการเชื่อมต่อเกิดขึ้นกับโมเด็มระยะไกล โมเด็มท้องถิ่นจะเข้าสู่สภาวะออนไลน์และจะไม่แปลงความหมายข้อมูลที่ส่งให้กับมัน แต่จะส่งผ่านข้อมูลออกไปแทน ถ้าสัญญาณพาหะหายไป เช่น โมเด็มระยะไกลวางหู โมเด็มจะอยู่ในสภาวะคำสั่ง คำสั่งที่ถูกส่งโดยคอมพิวเตอร์ไปยังโมเด็มสามารถถูกส่งได้โดยซอฟแวร์สื่อสาร หรือพิมพ์เข้าไปจากแป้นพิมพ์ โดยเอาต์พุตของแป้นพิมพ์ถูกเปลี่ยนทิศทางไปยังพอร์ตอนุกรม คำสั่งที่ถูกส่งในสภาวะคำสั่งควรถูกส่งด้วยเจ็ดบิตข้อมูล หนึ่งพาริตี้ หรือแปดบิตข้อมูล ไม่มีพาริตี้

ขั้นตอนการโทรศัพท์ออก โดยเริ่มตั้งแต่ขั้นตอนการสั่งให้โมเด็มหมุนหมายเลขโทรศัพท์จนกระทั่งวางหูโทรศัพท์เป็นดังนี้

1) ผู้ใช้เลือกคำสั่ง " Dial " แล้วซอฟแวร์เปิดสัญญาณ DRT เพื่อส่งคำสั่งหมุนหมายเลขไปยังโมเด็มโดยใช้คำสั่ง ATDT3266281 ที่โมเด็มต้นทางจะยกหูโทรศัพท์ รอสัญญาณให้หมุนเลขหมาย

2) ซอฟแวร์รอ result code จากโมเด็ม และโมเด็มต้นทางจะรอการตอบรับจากปลายทาง ทั้งนี้ระยะเวลารอขึ้นกับการกำหนดค่ารีจิสเตอร์

3) ที่โมเด็มปลายทางมีเสียงโทรศัพท์ดังขึ้น

4) โมเด็มปลายทางตอบรับสัญญาณ

5) โมเด็มต้นทางรับสัญญาณตอบรับและส่งสัญญาณ originate carrier

6) โมเด็มรับทราบวิธีการมอดูเลชั่นและความเร็วของแต่ละฝ่าย

7) โมเด็มต้นทางส่ง result code " CONNECT " ไปยังPC ปิดลำโพงและเปิดสัญญาณ CD

8) ซอฟแวร์ รับรู้ result code และส่งสัญญาณ CD ให้ผู้ใช้ทราบว่าการติดต่อได้เกิดขึ้นแล้ว

9) ผู้ใช้เริ่มติดต่อกับโฮสคอมพิวเตอร์ ซอฟแวร์ดำเนินการสื่อสาร ในขณะที่โมเด็มทำหน้าที่ส่งข้อมูล

10) เมื่อการสื่อสารเสร็จ ผู้ใช้ก็เลือกคำสั่ง " DISCONNECT " แล้วซอฟแวร์จะปิดสัญญาณ DTR ตามด้วยคำสั่ง ATH

11) โมเด็มต้นทางวางสายโทรศัพท์ และโมเด็มปลายทางยกเลิกสัญญาณ และวางสายโทรศัพท์

ขั้นตอนการเชื่อมต่อระหว่างโมเด็มต้นทางและปลายทางนั้น โมเด็มทั้งคู่จำเป็นจะต้องมีพื้นฐานที่เหมือนกันบางประการ เช่น ความเร็ว รูปแบบของข้อมูล และโปรโตคอลในการรับส่งข้อมูล การเชื่อมต่อโมเด็มในแต่ละครั้งว่าจะโทรออกเบอร์ใด ควรจะตั้งเวลาในการรอเสียงตอบจากโมเด็มปลายทางนานเท่าไร ระบบโทรศัพท์เป็น Tone หรือ Pulse ซึ่งการดำเนินการทั้งหมดจะสามารถควบคุมโดยชุดคำสั่ง AT คำสั่งทั้งหมดของโมเด็มเริ่มต้นด้วย AT โมเด็มสามารถตรวจสอบอัตราบอด ความยาวเวิร์ด และพาริตี้จากอักษรสองตัวนี้ ตัวอย่างคำสั่งได้แก่

ATDT หมุนหมายเลขโทรศัพท์แบบ Touch -Tone ซึ่งจะทำให้โมเด็มยกหูโทรศัพท์ขึ้น (Off-hook) และหมุนหมายเลขโทรศัพท์ออก ซึ่งคำสั่งนี้จะต้องการพารามิเตอร์ต่างๆ เช่น ! (วางหูโทรศัพท์ชั่วคราว) ,(หยุดรอเป็นเวลา 2 วินาที) w (รอฟังเสียงสายว่าง)

ATH ยกและวางหูโทรศัพท์ มีรูปแบบดังนี้ ATH0 หรือ ATH ให้โมเด็มวางหูโทรศัพท์ ซึ่งมักจะใช้หลังจากส่ง (+++) ให้แก่โมเด็ม ส่วน ATH1 ให้โมเด็มยกหูโทรศัพท์

ATZ วางสายและรีเซตกลับสู่ค่าโดยปริยาย

รหัสผลลัพธ์จากโมเด็ม เมื่อโมเด็มได้รับคำสั่ง มันจะส่งรหัสผลลัพธ์ (result code) กลับมา รหัสนี้อาจอยู่ในรูปของข่าวสารข้อความหรือรหัสตัวเลข รหัสคำสั่งต่างๆ แสดงดังตารางที่ 2.1

ตารางที่ 2.1 รหัสผลลัพธ์จากโมเด็ม

รหัสตัวเลข	รหัสข้อมูล	ความหมาย
0	OK	คำสั่งถูกนำไปปฏิบัติ
1	CONNECT	เชื่อมต่อที่ 0-300 bps
2	RING	ตรวจพบสัญญาณกริ่ง
3	NO CARRIER	ไม่พบโมเด็มระยะไกล
4	ERROR	ความผิดลาดในคำสั่ง
5	CONNECT 1200	เชื่อมต่อที่ 1200 bps
6	NO DIALTONE	ไม่พบสัญญาณให้หมุน
7	BUSY	สายไม่ว่าง
8	NO ANSWER	ไม่มีการตอบสนองจากโมเด็มระยะไกล
9	CONNECT 2400	เชื่อมต่อที่ 2400 bps

บทที่ 3
การคำนวนและการสร้าง

โปรแกรมการทำงานของการต่อหมายเลขโทรศัพท์ด้วยเสียง ประกอบด้วย 2 ส่วนหลักได้แก่

- ส่วนวิเคราะห์เสียง (recognize)
- ส่วนควบคุมการสั่งงานโมเด็ม (dialer)

โปรแกรมทั้งสองส่วนนี้เขียนด้วยภาษาซี โดยรันโปรแกรมบน Borland C++

3.1 ส่วนวิเคราะห์เสียง

โดยส่วนการวิเคราะห์เสียงสามารถแบ่งได้เป็น 2 ขั้นตอน คือ ขั้นตอนการเรียนรู้ และขั้นตอนการรับรู้

1. ขั้นตอนการเรียนรู้

เป็นขั้นตอนที่ต้องการสร้างแบบจำลองของเสียงขึ้นมา เพื่อนำไปใช้ในขั้นตอนการรู้จำเสียงพูดต่อไป ประกอบด้วยส่วนย่อยๆ ดังนี้

1.1 การวิเคราะห์สัญญาณเสียงเบื้องต้น

มีการเลือกใช้ ค่าต่างๆ ในการคำนวนและออกแบบโปรแกรมดังนี้

1) การพรีเอมฟาซีส

ใช้วงจรอันดับหนึ่ง ซึ่งมีฟังก์ชั่นถ่ายโอน คือ

$$H(z) = 1 - \alpha z^{-1}$$

ค่า α ที่ใช้คือ 15/16 = 0.9375

2) การแบ่งช่วงสัญญาณ

ขนาดของช่วงสัญญาณมีเงื่อนไขในการเลือก คือ

- ค่า N ต้องสั้นพอที่คุณสมบัติของเสียงไม่ เปลี่ยนแปลง
- ค่า N ต้องยาวพอที่จำนวนของตัวอย่างมี เพียงพอสำหรับการหาสัมประสิทธิ์
- การเลื่อนในการวิเคราะห์ (ค่า M) ต้องไม่ข้ามข้อมูล

ดังนั้นค่า M จะน้อยกว่า N แต่ถ้าค่า M มีขนาดเล็กเกินไปจะทำให้การคำนวณช้าลง ดังนั้นจึงเลือกค่า M = 100 แซมเปิ้ล และค่า N = 300 แซมเปิ้ล

3) ความถี่ที่ใช้ในการสุ่มสัญญาณ

เนื่องจาก ความถี่ที่ใช้ในการแซมปลิ้ง มากกว่าหรือเท่ากับ สองเท่าของความถี่เสียง ($f_s \geq f_N$) ดังนั้น $f_s \geq$ 8 KHz แต่เนื่องจากใน wave studio มีความถี่ที่ใกล้เคียงที่สุด คือ 11.025 Khz

ดังนั้น ช่วงเวลาที่ใช้ในการวิเคราะห์แต่ละเฟรม คือ 300/11.025 = 27.21 ms และระยะที่ใช้ในการเลื่อนเฟรมคือ 100/11.025 = 9.07 ms

4) การเลือกวินโดว์ที่เหมาะสมสำหรับการวิเคราะห์เสียง

โดยพิจารณาลักษณะสเปคตรัม คือ

- ความถี่เรโซลูชั่นสูง (high frequency resolution) คือ มีโลบหลักแคบและแหลม

- การลดทอน (Attenuation) นอกช่วงความถี่ที่ผ่านได้ ต่ำ คือ ไซด์โลบมีค่าน้อยฟังก์ชันวินโดว์มีหลายชนิด แต่ชนิดที่เหมาะสมที่สุดที่นำมาใช้ได้แก่ แฮมมิ่งวินโดว์ ซึ่งมีค่าไดโลบต่ำ (- 40 dB) และเมนโลบแคบพอใช้

5) การหาคุณลักษณะของเสียง

ใช้การประมาณพันธะเชิงเส้นในการวิเคราะห์หาค่าสัมประสิทธิ์ LPC ซึ่งการประมาณประมาณเชิงเส้นที่เลือกใช้ คือ วิธีอัตสัมพัทธ์ (autocorrelation method) ซึ่งวิธีนี้มีการคำนวณที่ง่ายกว่าวิธีอื่นๆ และมีความแน่นอนด้านเสถียรภาพ อีกทั้งมีวิธีการเก็บข้อมูลที่น้อยกว่า

เนื่องจากเป็นการวิเคราะห์โดยวิธีอัตสัมพัทธ์ อันดับการประมาณเชิงเส้น (P) ที่มากจะทำให้การประมาณเสียงมีความใกล้เคียงมากยิ่งขึ้น แต่ถ้าอันดับ P มีค่ามากเกินไปจะทำให้การคำนวณมีความยุ่งยากและใช้เวลานาน ดังนั้น เพื่อความเหมาะสมค่าอันดับ P ที่ใช้ คือ 12

อัลกอริทึมของขั้นตอนการวิเคราะห์เสียงแสดง ดังรูปที่ 1 ในภาคผนวก

1.2 การสร้างโค้ดบุค

จากการทดสอบ (L.R. Rabiner, S.E. Levinson และ Sondhi ,1982) จะได้ว่าที่ขนาดโค้ดบุคเท่ากับ 64 จะมีค่าความคลาดเคลื่อนเฉลี่ยประมาณ 0.2 ซึ่งเป็นค่าที่น้อยมากสำหรับเวคเตอร์ควอนไตซ์เซชั่น

การวัดค่าความคลาดเคลื่อน ใช้วิธีการคำนวณแบบ square error distortion ในการหาระยะทางเนื่องจากเป็นวิธีที่ง่าย และรวดเร็ว

การสร้างโค้ดบุค โดยนำเทรนนิ่งเซตที่ได้จากการประมาณเชิงเส้นมาผ่านกระบวนการดังนี้

1) สุ่มค่าโค้ดบุคเริ่มต้นเริ่มต้นมา 64 ตัว ตัวละ 19 มิติ
2) หาระยะทางระหว่างโค้ดบุคกับเทรนนิ่งเซต แต่ละตัวโดยใช้ความคลาดเคลื่อนกำลังสอง
3) จัดกลุ่มของเวคเตอร์อินพุตโดยพิจารณาจาระยะทางที่น้อยที่สุด
4) หาจุดศูนย์กลางของกลุ่ม
5) ทำขั้นตอน 3 และ 4 ซ้ำจนกว่าความคลาดเคลื่อนรวมจะน้อยกว่า 0.001 ซึ่งจุดศูนย์กลางที่ได้ก็คือโค้ดบุคนั่นเอง

อัลกอริทึมของการสร้างโค้ดบุค ดังรูปที่ 2 ในภาคผนวก

1.3 การสร้างแบบจำลอง HMM ของเสียง มีขั้นตอนดังนี้

1) สุ่มค่าเริ่มต้น a, b และ กำหนดให้ π = [100000] ตามเงื่อนไขในการใช้แบบจำลองแบบ Left - Right

2) หาค่า α ,β จากค่า a,b เริ่มต้น และลำดับค่าปรากฏ $O = \{ O_1 O_2 O_3 \ldots O_T \}$ ซึ่งเรียก ว่าลำดับเทรนนิ่ง ตามวิธีของ Forward - Backward Procedure โดยใช้ลำดับของค่าปรากฏหลายๆ ลำดับเข้ามาเทรนเพื่อความถูกต้องมากขึ้น

3) ทำการสเกลลิ่ง α ,เพื่อให้ค่าอยู่ในย่านที่คอมพิวเตอร์สามารถคำนวณได้อย่างถูกต้อง

4) หาค่าพารามิเตอร์ a,b,π ที่ให้ค่าความน่าจะเป็นสูงสุด ที่จะเป็นแบบจำลอง λ ที่เหมาะสมของคำ

5) ตรวจสอบค่าพารามิเตอร์ของแบบจำลองที่ได้ ว่า ลู่เข้าหรือยัง โดยใช้วิธีการคำนวณค่า a,b ซ้ำประมาณ 50 รอบ เมื่อมีการเปลี่ยนแปลงน้อยมากจนเป็นที่พอใจตามระดับค่าที่ตั้งไว้ในที่นี้ใช้ค่าเท่ากับ 10^{-5} ก็จะหยุด และได้ค่าพารามิเตอร์ a,b และ π ของแบบจำลองที่ต้องการ

6) เก็บค่าพารามิเตอร์ a,b,π ที่ได้จากข้อ 5. เป็นพารามิเตอร์ของแบบจำลองไว้

อัลกอริทึมของการสร้างแบบจำลองแสดงดังรูปที่ 3 ในภาคผนวก

2. ขั้นตอนการรับรู้

2.1 การหาดัชนีโค้ดบุด

โดยการนำเวคเตอร์เสียงจากการประมาณเชิงเส้นมาทีละเสียง แล้วเปรียบเทียบกับโค้ดบุคที่ได้จากการสร้างในขั้นตอนการเรียนรู้ ทีละเฟรม โดยวิธีความคลาดเคลื่อนกำลังสอง เวคเตอร์เสียงห่างจากโค้ดบุคใดน้อยที่สุดจะได้ว่าเป็นดัชนีโค้ดบุคของเฟรมเสียงนั้น และเก็บดัชนีโค้ดบุคของแต่ละเฟรมในแต่ละเสียงไว้เป็นลำดับค่าปรากฏ (observation sequence) สำหรับการสร้างแบบจำลองต่อไป

อัลกอริทึมของการหาดัชนีโค้ดบุคแสดงดังรูปที่ 4 ในภาคผนวก

2.2 การรู้จำเสียง

หลังจากที่ได้แบบจำลอง HMM ของแต่ละคำศัพท์แล้ว เมื่อมีลำดับของค่าปรากฏ $O = \{ O_1\ O_2\ O_3 \ldots O_T\}$ ของเสียง unknown ซึ่งเป็นเสียงที่ต้องการทดสอบเข้ามา เราจะทำการคำนวณหาความน่าจะเป็น $P(O|\lambda)$ ทุกแบบจำลองของแต่ละคำศัพท์โดยใช้วิธี viterbi algorithm แล้วเลือกเอาคำศัพท์ที่มีความน่าจะเป็นสูงสุด ซึ่งก็คือ คำศัพท์ที่แบบจำลองจำได้นั่นเอง

อัลกอริทึมของส่วนนี้แสดงดังรูปที่ 5 ในภาคผนวก

3.2 ส่วนควบคุมการสั่งงานโมเด็ม

เมื่อส่วนของการรู้จำเสียงสามารถรู้จำได้ว่าเป็นคำศัพท์คำใด (recognized word) หรือเป็นโมเดลใด แล้วจะไปหาหมายเลขโทรศัพท์ที่ตรงกับโมเดลนั้นแล้วนำไปสั่งโมเด็ม โดยในครั้งแรกจะทำการรีเซตโมเด็มก่อน แล้วจึงค่อยทำการต่อโทรศัพท์ โดยเมื่อสั่งโมเด็มให้โทรศัพท์แล้วผู้ใช้จะต้องยกหูไว้ก่อน เมื่อโทรเสร็จแล้วจึงค่อยกดปุ่มตกลง เพื่อให้โปรแกรมหยุดการทำงานของโมเด็ม และส่งการทำงานไปให้โทรศัพท์ก็จะทำการสนทนาได้ อัลกอริทึมของส่วนนี้แสดงดังรูปที่ 6 ในภาคผนวก

บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลอง

ขั้นตอนการทดลอง แบ่งเป็น 2 ขั้นตอน คือ ขั้นตอนการเรียนรู้ และขั้นตอนการวิเคราะห์

ขั้นตอนการเรียนรู้

เป็นขั้นตอนที่สร้างฐานข้อมูลของเสียงขึ้นมา ซึ่งเป็นเสียงที่ต้องการให้ระบบรับรู้ได้ เช่น ต้องการให้ระบบรับรู้เสียงได้ 10 เสียงที่ต่างกัน จะมีขั้นตอนดังนี้

1. เก็บเสียงที่ต้องการไว้เพื่อสร้างโมเดลโดยแบ่งเป็น

- เก็บเสียงตัวเลข ' 0 - 9 ' 4 กรณี เพื่อใช้ เปรียบเทียบ ดังนี้

<u>กรณีที่ 1</u> ผู้หญิง 1 คน พูด 0 - 9 เสียงละ 5 ครั้ง รวมเป็น 50 เสียง

<u>กรณีที่ 2</u> ผู้ชาย 1 คน พูด 0 - 9 เสียงละ 5 ครั้ง รวมเป็น 50 เสียง

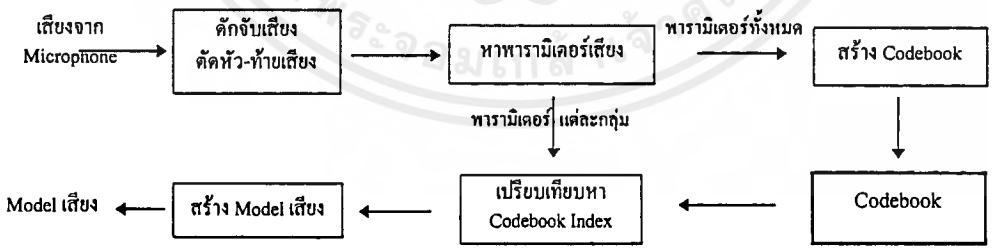
<u>กรณีที่ 3</u> ผู้หญิง 7 คน พูด 0 - 9 เสียงละ 1 ครั้ง รวมเป็น 70 เสียง

<u>กรณีที่ 4</u> ผู้ชาย 3 คน หญิง 7 คน เสียงละ 1 ครั้ง รวมเป็น 100 เสียง

- เก็บเสียงชื่อคนทั้งหมด 20 ชื่อ ดังนี้ ' ดาว , จุลชิต , กรุณา , กิ๊ก , มาลี , นาวิน , ปู , ศิริลักษณ์ , สมชาย , ตะวัน ,สุเมธ,ยิ้ม, อรดี อั้ม, วี, จิตรลดา, โสภิณ, อาร์ท, มัณฑนา และ ต่อศักดิ์ ' เพื่อใช้สำหรับการทดลองใช้งานจริง

2. นำข้อมูลเสียงทั้งหมดที่เก็บไว้ในแต่ละกรณี มาทำการหาค่าพารามิเตอร์ ที่เป็นตัวแทนของเสียงโดยวิธี Linear Predictive Coding (LPC)

3. นำพารามิเตอร์เสียง (file LPC) ทั้งหมดจากทุกกลุ่มเสียงมาทำการลดจำนวนข้อมูลโดยวิธีการ Vector Quantization (VQ) ซึ่งจะได้เวคเตอร์ตัวแทนของเสียงที่ระบบสามารถรับรู้ได้ทั้งหมดออกมา เรียกว่า **โค้ดบุ๊ค** (Code Book)



รูปที่ 4.1 แสดงขั้นตอนการเรียนรู้

4. ทำการหา Codebook Index ของแต่ละกลุ่มเสียง โดยการนำ file LPC ทุก file ของกลุ่มนั้นๆ มาผ่านขั้นตอนอีก ส่วนหนึ่งของ VQ จะได้ตัวเลขมากลุ่มหนึ่งที่บอกถึงความสัมพันธ์ของเสียงกลุ่มนั้นๆกับ Codebook

5. สร้างโมเดลของเสียง โดยการนำ Codebook Index ของแต่ละกลุ่มเสียงมาผ่านขั้นตอนการสร้างโมเดลด้วย วิธี Hidden Markov Model (HMM) เมื่อทำจนครบทุกกลุ่มเสียง ก็จะได้โมเดลครบทุกเสียง และพร้อมที่จะนำไปใช้งานในขั้นการวิเคราะห์ต่อไป

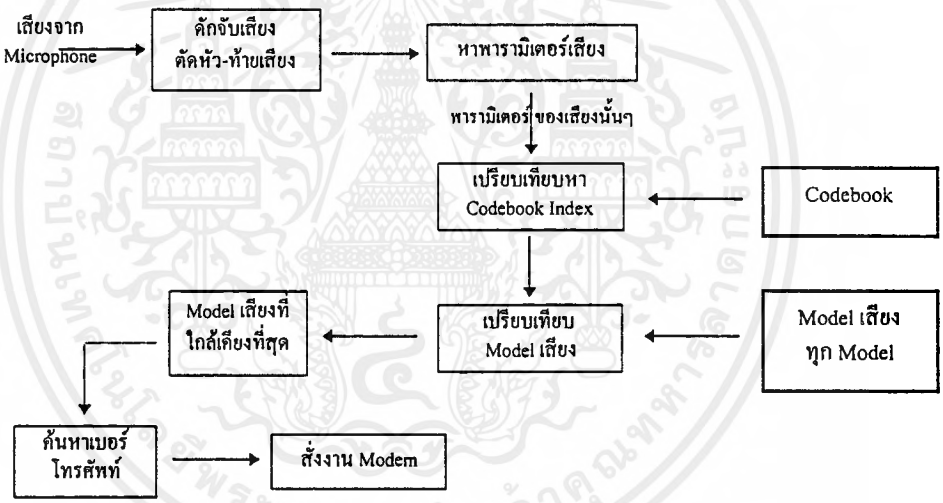
ขั้นการวิเคราะห์

เป็นขั้นตอนที่จะวิเคราะห์เสียงที่ไม่ทราบว่าเป็นคำใด ว่ามีความเหมือนกับโมเดลของเสียงใดที่เก็บไว้มากที่สุด ขั้นตอนในช่วงแรกจะคล้ายกับขั้นการเรียนรู้ ดังนี้

1. นำเสียงที่ต้องการทดสอบ (เพียง 1 เสียง) มาผ่านขั้นตอน LPC จะได้พารามิเตอร์ของเสียงนั้น และนำพารามิเตอร์มาเปรียบเทียบกับ Codebook จะได้ Codebook Index

2. นำ Codebook Index มาหาค่าความน่าจะเป็นกับทุกโมเดลที่เก็บไว้ทั้งหมด โดยวิธี Viterbi Algorithm ก็จะได้ค่าความน่าจะเป็นของแต่ละโมเดลออกมา ค่าความน่าจะเป็นเมื่อเปรียบเทียบกับโมเดลใดมีค่าสูงที่สุด ผลจะได้ว่าเสียงที่นำมาทดสอบนั้นตรงกับโมเดลนั้นนั่นเอง

3. นำเสียงพูดที่จำได้ มาเขียนโปรแกรมสั่งงานให้โมเด็มต่อโทรศัพท์ให้ตามหมายเลขที่ตรงกับโมเดลนั้น



รูปที่ 4.2 แสดงขั้นตอนการวิเคราะห์

ที่กล่าวขั้นต้นเป็นขั้นตอนการทำงานโดยรวมของโปรแกรม ต่อไปจะเป็นการอธิบายในรายละเอียดของส่วนต่างๆ ของโปรแกรมว่ามี Input, Output และหน้าที่การทำงานอย่างไร โดยจะแบ่งเป็น 2 ส่วนหลัก คือ

<u>ส่วนการรู้จำเสียง</u>

ส่วนนี้เขียนโปรแกรมด้วยภาษา C โดยใช้ Borland C++ เขียนในส่วน function ที่ใช้คำนวณ ชื่อ " lpc.c , vqln.c , vqcm.c " ซึ่งเป็นขั้นตอนการทำงานเพื่อให้สามารถวิเคราะห์เสียงได้ มีการทำงานเป็นลำดับขั้นดังนี้

ขั้นการเรียนรู้

1. จากเสียงที่อยู่ในรูป wave format (ให้เป็น . lpc)

2. โปรแกรม LPC.C รับไฟล์ .wav เพื่อทำการหาค่าพารามิเตอร์ของ file ทุก file

3. โปรแกรม VQLN.C รับไฟล์ .lpc เพื่อทำการสร้าง Codebook

4. โปรแกรม VQCM.C จะนำพารามิเตอร์ของเสียงที่เป็นส่วนประกอบในแต่ละโมเดลมาทำการเปรียบเทียบ กับ Codebook เพื่อให้ได้ Codebook Index

5. Codebook Index ที่ได้จากขั้นตอนที่ 4 มาเป็น Observation sequenceในการสร้างโมเดลของเสียงแต่ละเสียงโดยใช้โปรแกรม HMM.C

6. ทำซ้ำตั้งแต่ข้อ 4 - 5 จนครบทุก Model ก็จะสำเร็จขั้นตอนการเรียนรู้

ขั้นการวิเคราะห์

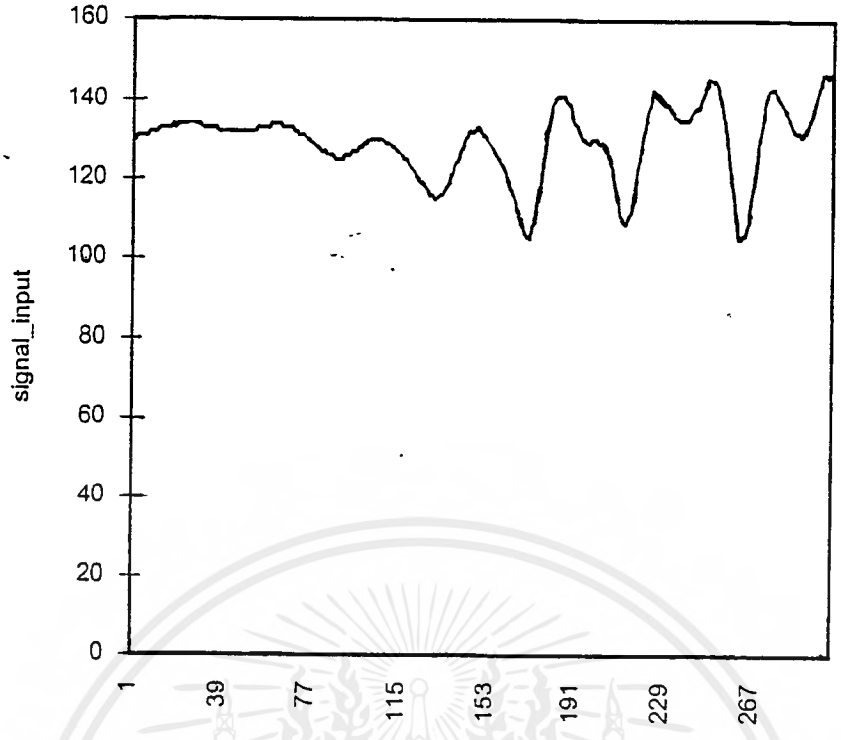
1. หลังจากที่ได้รับไฟล์เสียง จะนำมาหาค่าพารามิเตอร์
2. นำพารามิเตอร์ของเสียงมาเปรียบเทียบกับ Codebook ที่ได้จากขั้นตอนการเรียนรู้จะได้Codebook Index ของเสียงนั้น
3. นำCodebook Indexที่ได้มาหาค่าความน่าจะเป็นกับทุกโมเดลที่มีอยู่ จะได้ค่าความน่าจะเป็นออกมาจำนวนหนึ่ง ซึ่งจะมีจำนวนเท่ากับจำนวนของโมเดล
4. ทำการเลือกค่าความน่าจะเป็นที่มากที่สุดที่ได้จากขั้นตอนที่ 3 ว่าตรงกับโมเดลใด เป็นผลลัพธ์ของการวิเคราะห์ โดยโปรแกรม ASRD.C

<u>ส่วนการต่อโทรศัพท์</u>

หลังจากที่วิเคราะห์ได้ว่าเสียงที่นำมาทดสอบตรงกับโมเดลใดแล้ว จะไปหาหมายเลขโทรศัพท์ที่ตรงกับโมเดลนั้นแล้วนำไปสั่ง Modem โดยในครั้งแรกจะทำการ Reset Modem ก่อนแล้วจึงค่อยทำการโทรศัพท์ โดยเมื่อสั่ง Modem ให้โทรศัพท์แล้วผู้ใช้จะต้องยกหูโทรศัพท์ไว้ก่อน เมื่อโทรติดแล้วจึงค่อยกดปุ่มตกลงเพื่อให้โปรแกรมหยุดการทำงานของ Modem และส่งการทำงานไปให้โทรศัพท์ก็จะทำการสนทนาได้ โดยใช้โปรแกรม ASRD_APP.C

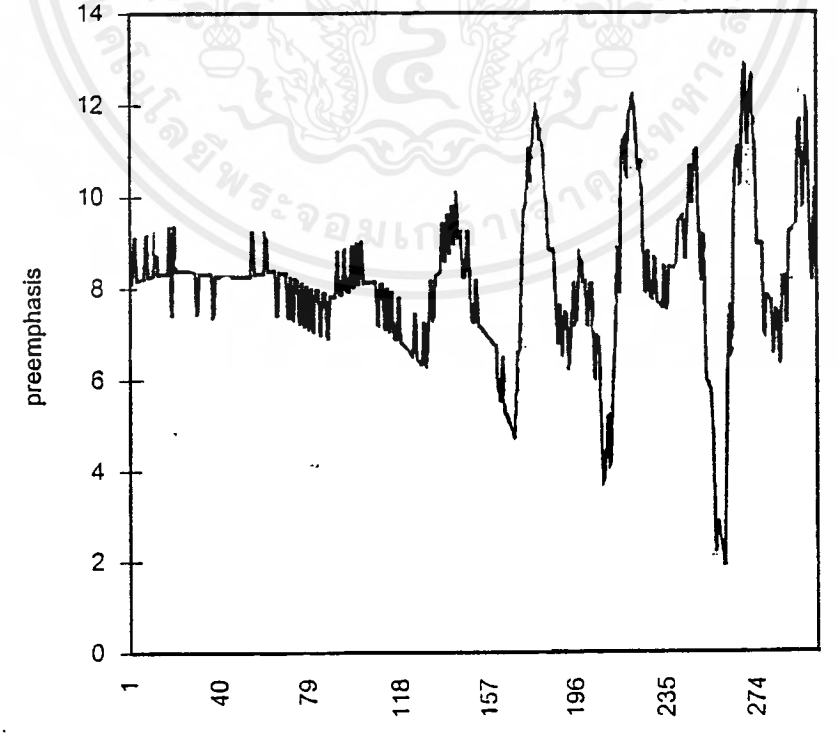
4.2 ผลการทดลอง

4.2.1 เมื่อทำการอัดเสียงโดยใช้โปรแกรมอัดเสียงเพื่อจะนำไปผ่านกระบวนการต่างๆ โดยเสียงที่อัดแล้วนี้จะเป็น input เริ่มแรกอยู่ในรูป ไฟล์.wav และทำการ samplingแล้ว แต่ได้นำมาพล็อตกราฟเพียง 1 เฟรมซึ่งมี 300 ตัวอย่างเสียง แสดงได้ดังรูปที่ 4.1



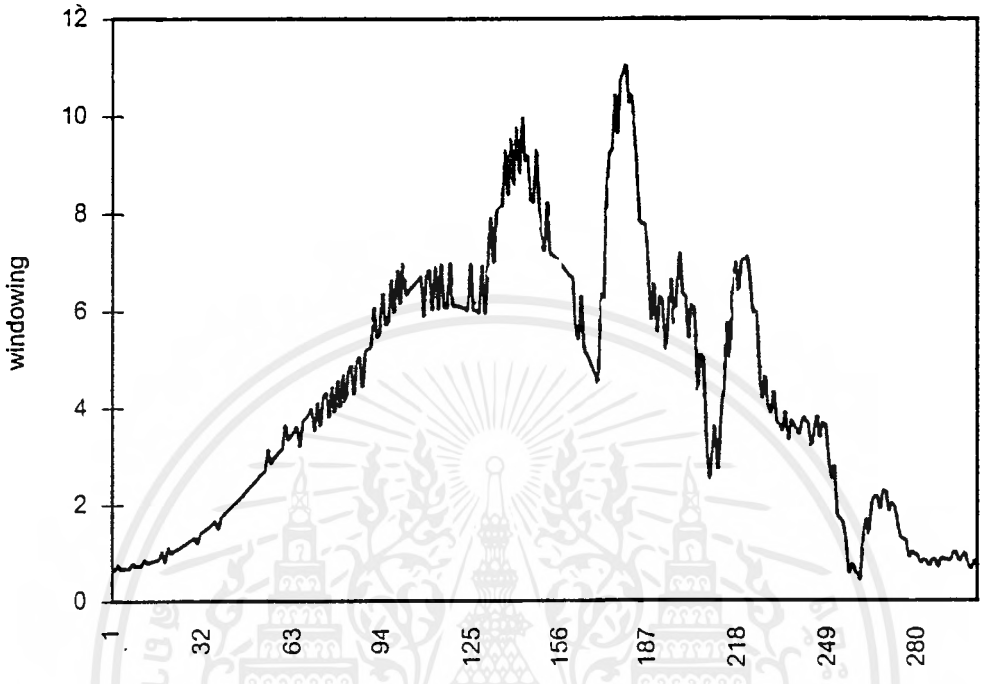
รูปที่ 4.3 แสดงสัญญาณเสียงอินพุท 1 เฟรมของเสียง 1

4.2.2 จากนั้นเมื่อนำเสียง input มาผ่านการพรีเอมฟาซีส ก็จะได้ output ออกมาเป็นค่าต่างๆตามการคำนวณจากสมการ แสดงกราฟค่าที่ผ่านการพรีเอมฟาซีส โดยเลือกมา 1 เฟรม ได้ดังรูปที่4.2



รูปที่ 4.4 แสดงสัญญาณที่ผ่านการพรีเอมฟาซีส 1 เฟรมของเสียง 1

4.2.3 ขั้นตอนต่อไปก็ผ่านการวินโดว์วิ่ง ได้ output ออกมาเป็นค่าตามการคำนวณจากสมการ แสดงกราฟค่าที่ผ่านการวินโดว์ โดยเลือกมา 1 เฟรม แสดงได้ดังรูป 4.3



รูปที่ 4.5 แสดงสัญญาณที่นำมาผ่านการวินโดว์ 1 เฟรมของเสียง 1

4.2.4 จากนั้นก็ทำการหาค่าสัมประสิทธ์การประมาณเชิงเส้น โดย 1 เฟรมจะมี 12 ค่า และผ่านลมการเซปสตรัม 19 ค่าแล้วแปลงเป็นค่าพารามิเตอร์ที่เวทแล้ว 19 ค่า แสดงค่าต่างได้ดังตาราง โดยเลือกแสดง 1 เฟรมของเสียง

ตารางที่ 4.1 แสดงค่าสัมประสิทธิ์ LPC , เซปสตรัม และค่าพารามิเตอร์ที่เวทแล้ว

ลำดับที่	ค่าสัมประสิทธิ์ LPC	ค่าที่ผ่านเซปสตรัม	ค่าที่ผ่านการเวท
1	0.556066	2.0423820018768	2.042382001876
2	0.516202	0.5560658574104	1.379794716835
3	0.318525	0.6708063483238	2.631100654602
4	- 0.098680	0.6628803610802	3.502346038818
5	- 0.258899	0.3951893448829	2.579762220380
6	- 0.176349	0.1967030018568	1.499174952507
7	0.150480	0.1103119403123	0.941457092761
8	- 0.009164	0.2558520734310	2.364574670791
9	- 0.167745	0.0942191705107	0.916243970394
10	- 0.118151	- 0.0683846250176	- 0.681744158267
11	0.170232	- 0.0969219654798	- 0.966240286827
12	0.114524	0.060305200517	0.586444079875
13		0.010675645430	0.09866388887
14		- 0.035597965120	- 0.30381077527
15		0.039001714438	0.29725214838
16		0.051078598900	0.33343672750
17		0.026084324300	0.13781721889
18		0.026122003793	0.10245820879
19		0.0696291103959	0.17277428507

4.2.5 เมื่อได้พารามิเตอร์ของเสียงทั้งหมดที่จะสร้างแบบจำลองในกรณีใดกรณีหนึ่งก็นำมาผ่านการจัดระดับเวกเตอร์

4.2.6 จากนั้นก็นำมาสร้างแบบจำลองเสียงด้วยกระบวนการของ Hidden Markov Model (HMM) จนได้เป็น 1 แบบจำลองของ 1 เสียง แล้วทำซ้ำไปเรื่อยๆจนครบทุกเสียง ในที่นี้ทำการสร้างแบบจำลองทั้งหมด 5 กรณี โดยเมื่อนำเสียงแต่ละกรณีมาสร้างแบบจำลองเรียบร้อยแล้วก็ทำการทดสอบว่าได้ผลเป็นอย่างไร

4.2.7โดยการทดสอบเสียงตามแบบจำลองที่สร้างไว้ทั้งหมด 5 แบบจำลอง โดยแต่ละกรณีจะทำการทดสอบ 2 แบบ คือ ใช้เสียงจากผู้พูดต้นแบบที่ใช้สร้างแบบจำลอง และใช้เสียงจากบุคคลอื่น หรือเรียกว่าเสียง unknown พูดเสียงตามเสียงต้นแบบ เช่น พูดเสียง 0-9 สำหรับกรณีที่ 1-4 และพูดเสียงชื่อคน 20 ชื่อ สำหรับกรณีที่5 เมื่อเสียงผ่านแบบจำลองแล้วจะทำการ recognize ออกมาเป็นเสียงพูดที่จำได้ แสดงดังตารางผลการทดลองในแต่ละกรณี ดังนี้

<u>กรณีที่1</u> แบบจำลองเสียงต้นแบบจากผู้หญิงคนเดียว พูด 0-9 เสียงละ 5 ครั้ง รวมเป็น 50 เสียง

ตารางที่ 4.2 ผลจากการทดสอบโดยให้ผู้พูดคนเดิมพูด 0-9 เสียงละ 5 ครั้ง

	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่1	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่2	0	1	2	3	4	5	9	7	8	9
พูดครั้งที่3	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่4	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่5	0	1	2	3	4	5	6	7	8	9
เปอร์เซ็นต์	100	100	100	100	100	100	80	100	100	100

ตารางที่ 4.3 ผลจากการทดสอบโดยใช้เสียงผู้อื่น(unknown) พูด 0-9 เสียงละ 1 ครั้ง

	0	1	2	3	4	5	6	7	8	9
คนที่1	0	1	2	3	4	9	0	7	8	9
คนที่2	2	1	2	3	1	8	0	1	8	9
คนที่3	0	1	2	3	4	9	0	7	9	9
คนที่4	0	1	2	3	4	5	6	7	8	9
คนที่5	0	1	2	9	4	8	0	7	8	9
เปอร์เซ็นต์	80	100	100	80	80	20	20	80	80	100

<u>กรณีที่ 2</u> แบบจำลองเสียงต้นแบบจากผู้ชายคนเดียว พูด 0-9 เสียงละ 5 ครั้ง รวมเป็น 50 เสียง

ตารางที่ 4.4 ผลจากการทดสอบโดยให้ผู้พูดคนเดิมพูด 0-9 เสียงละ 5 ครั้ง

	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่1	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่2	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่3	0	1	2	3	4	5	6	7	8	9
พูดครั้งที่4	0	1	2	3	4	5	6	7	5	9
พูดครั้งที่5	0	1	2	3	4	5	6	7	8	9
เปอร์เซ็นต์	100	100	100	100	100	100	100	100	80	100

ตารางที่ 4.5 ผลจากการทดสอบโดยใช้เสียงผู้อื่น(unknown) พูด 0-9 เสียงละ 1 ครั้ง

	0	1	2	3	4	5	6	7	8	9
คนที่1	0	1	2	3	0	3	6	7	7	9
คนที่2	0	1	2	5	4	5	6	6	2	9
คนที่3	0	1	2	3	0	5	6	7	5	9
คนที่4	0	1	2	5	4	5	6	2	2	9
คนที่5	0	0	2	3	4	8	6	7	5	9
เปอร์เซ็นต์	100	80	100	60	60	60	100	60	0	100

กรณีที่3 แบบจำลองเสียงต้นแบบจากผู้หญิง 7 คน พูด 0-9 เสียงละ 1 ครั้ง รวมเป็น 70 เสียง

ตารางที่ 4.6 ผลจากการทดสอบโดยให้ผู้หญิงคนเดิมทั้ง 7 คน พูด 0-9 เสียงละ 1 ครั้ง

	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่1	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่2	0	1	2	3	4	5	6	7	9	9
ผู้หญิงคนที่3	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่4	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่5	0	1	2	3	4	9	6	7	8	9
ผู้หญิงคนที่6	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่7	0	1	2	3	4	5	6	7	8	9
เปอร์เซ็นต์	100	100	100	100	100	85.72	100	100	85.72	100

ตารางที่ 4.7 ผลจากการทดสอบโดยใช้เสียงผู้อื่น(unknown) พูด 0-9 เสียงละ 1 ครั้ง

	0	1	2	3	4	5	6	7	8	9
ผู้ชายคนที่ 1	0	1	2	3	4	5	6	7	9	9
ผู้ชายคนที่ 2	0	1	2	2	4	9	6	9	8	9
ผู้ชายคนที่3	0	1	2	3	4	9	6	4	8	9
ผู้หญิงคนที่ 1	0	4	2	3	4	8	6	7	8	9
ผู้หญิงคนที่ 2	0	1	2	9	4	9	6	7	8	9
เปอร์เซนต์	100	80	100	60	100	20	100	60	80	100

กรณีที่4 แบบจำลองเสียงผู้ชาย 3 คน ผู้หญิง 7 คน พูด 0-9 เสียงละ 1 ครั้ง รวมเป็น 100 เสียง

ตารางที่ 4.8 ผลจากการทดสอบโดยใช้ผู้พูดคนเดิม พูด 0-9 เสียงละ 1 ครั้ง

	0	1	2	3	4	5	6	7	8	9
ผู้ชายคนที่1	0	1	2	3	4	3	6	7	8	9
ผู้ชายคนที่2	0	1	2	3	4	5	6	7	8	9
ผู้ชายคนที่3	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่1	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่2	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่3	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่4	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่5	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่6	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่7	0	1	2	3	4	5	6	4	8	9
เปอร์เซ็นต์	100	100	100	100	100	90	100	90	100	100

ตารางที่ 4.9 ผลจากการทดสอบโดยใช้เสียงคนอื่น(unknown) พูดเสียง 0-9 เสียงละ 1 ครั้ง

	0	1	2	3	4	5	6	7	8	9
ผู้ชายคนที่ 1	0	1	2	3	4	9	6	7	8	9
ผู้ชายคนที่ 2	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่ 1	0	1	2	3	4	5	6	7	8	9
ผู้หญิงคนที่ 2	0	1	2	3	4	5	6	7	1	9
เปอร์เซนต์	100	100	100	100	100	75	100	100	75	100

<u>กรณีที่ 5</u> แบบจำลองเสียงของผู้หญิงคนเดียวพูดชื่อคน 20 ชื่อ ชื่อละ 5 ครั้ง รวมเป็น 100 เสียง

ตารางที่ 4.10 ผลการทดสอบโดยใช้เสียงผู้พูดคนเดิมพูดชื่อคน 20 ชื่อ อย่างละ 5 ครั้ง

	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	นาวิน	ปู่	ศิริลักษณ์	สมชาย	ตะวัน
ครั้งที่ 1	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	นาวิน	ปู่	ศิริลักษณ์	สมชาย	ตะวัน
ครั้งที่ 2	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	สมชาย	ปู่	ศิริลักษณ์	สมชาย	ตะวัน
ครั้งที่ 3	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	มาลี	ปู่	ศิริลักษณ์	สมชาย	ตะวัน
ครั้งที่ 4	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	นาวิน	ปู่	ศิริลักษณ์	สมชาย	ตะวัน
ครั้งที่ 5	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	นาวิน	ปู่	ศิริลักษณ์	สมชาย	ตะวัน
เปอร์เซนต์	100	100	100	100	100	100	100	100	100	100

	สุเมธ	ยิ้ม	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
ครั้งที่ 1	สุเมธ	ยิ้ม	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
ครั้งที่ 2	สุเมธ	นาวิน	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
ครั้งที่ 3	สุเมธ	ยิ้ม	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
ครั้งที่ 4	สุเมธ	จิตรดา	อรดี	อาร์ท	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
ครั้งที่ 5	สุเมธ	ยิ้ม	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
เปอร์เซนต์	100	60	100	80	100	100	100	100	100	100

ตารางที่ 4.11 ผลจากการทดสอบโดยใช้เสียงคนอื่น(unknown) พูดเสียงชื่อคน ชื่อละ 1ครั้ง

	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	นาวิน	ปู	ศิริลักษณ์	สมชาย	ตะวัน
ผู้หญิงคนที่1	ดาว	สุเมธ	กรุณา	กิ๊ก	มาลี	นาวิน	ปู	จิตรลดา	สมชาย	ตะวัน
ผู้หญิงคนที่2	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	นาวิน	ปู	ศิริลักษณ์	สมชาย	ตะวัน
ผู้หญิงคนที่3	ดาว	จุลชิต	กรุณา	กิ๊ก	วี	นาวิน	ปู	ศิริลักษณ์	สมชาย	ตะวัน
ผู้หญิงคนที่4	ดาว	จุลชิต	กรุณา	กิ๊ก	มาลี	สมชาย	ปู	จิตรลดา	สมชาย	ตะวัน
เปอร์เซนต์	100	100	100	100	75	75	100	50	100	100

	สุเมธ	ยิ้ม	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	อาร์ท	มัณฑนา	ต่อศักดิ์
ผู้หญิงคนที่1	สุเมธ	วี	อรดี	อั้ม	วี	จิตรลดา	โสภิณ	ดาว	มัณฑนา	ต่อศักดิ์
ผู้หญิงคนที่2	สุเมธ	จิตรลดา	อรดี	อั้ม	มาลี	กรุณา	โสภิณ	ดาว	มัณฑนา	ต่อศักดิ์
ผู้หญิงคนที่3	สุเมธ	ยิ้ม	มาลี	อั้ม	วี	กรุณา	โสภิณ	ดาว	มัณฑนา	ต่อศักดิ์
ผู้หญิงคนที่4	สุเมธ	ยิ้ม	มาลี	ตะวัน	วี	จิตรลดา	โสภิณ	กรุณา	มัณฑนา	ต่อศักดิ์
เปอร์เซนต์	100	50	50	75	75	50	100	75	100	100

4.2.8 ผลการเปรียบเทียบความถูกต้องของการทดสอบการรู้จำเสียงพูด

ความถูกต้องของเสียงที่นำมาทดสอบ คิดเป็นเปอร์เซนต์แล้วได้ผลดังตาราง 4.12 ดังนี้

ผู้ทดสอบ	ความถูกต้อง
กรณีที่ 1 ผู้หญิงคนเดียว	
1.การทดสอบกับคนเดิม	98%
2.การทดสอบกับกลุ่มทดสอบ	
ผู้ชาย	65%
ผู้หญิง	80%
โดยรวม	74%
กรณีที่ 2 ผู้ชายคนเดียว	
1.การทดสอบกับคนเดิม	98%
2.การทดสอบกับกลุ่มทดสอบ	
ผู้ชาย	73.33%
ผู้หญิง	70%
โดยรวม	72%
กรณีที่ 3 ผู้หญิง 7 คน	
1.การทดสอบกับ 7 คนเดิม	97.14%
2.การทดสอบกับกลุ่มทดสอบ	
ผู้ชาย	80%
ผู้หญิง	80%
โดยรวม	80%
กรณีที่ 4 ผู้ชาย 3 คน,ผู้หญิง 7 คน	
1.การทดสอบกับ 10 คนเดิม	99%
2.การทดสอบกับกลุ่มคนทดสอบ	
ผู้ชาย	95%
ผู้หญิง	95%
โดยรวม	95%
กรณีที่ 5 ผู้หญิงคนเดียวพูดชื่อคน 20 ชื่อ	
1.การทดสอบกับคนเดิม	95%
2.การทดสอบกับกลุ่มคนทดสอบซึ่งเป็นผู้หญิง 4 คน	78.75%

จากผลการทดสอบการรู้จำเสียงพูด สามารถอธิบายผลการทดสอบได้ดังนี้

กรณีที่1 เนื่องจากเสียงต้นแบบเป็นเสียงผู้หญิงคนเดียว ประสิทธิภาพการรู้จำของเสียงผู้หญิงคนเดิมจึงมีสูงถึง 98 เปอร์เซนต์ และสามารถรู้จำเสียงของผู้หญิงคนอื่นได้ดีถึง 80 เปอร์เซนต์ และยังสามารถรู้จำเสียงของผู้ชายได้ถึง 65 เปอร์เซนต์ แม้ว่าเสียงต้นแบบจะไม่มีเสียงผู้ชายอยู่เลยก็ตาม เมื่อเปรียบเทียบโดยรวมแล้วนั้น กรณีที่ 1 นี้สามารถรู้จำเสียงของผู้พูดคนเดิมได้ 98 เปอร์เซนต์ และสามารถรู้จำเสียงของผู้พูดคนอื่นได้ 74 เปอร์เซนต์

กรณีที่ 2 เนื่องจากเสียงต้นแบบเป็นเสียงผู้ชายคนเดียว จึงมี ลักษณะคล้ายคลึงกับกรณีที่ 1 คือ ประสิทธิภาพการรู้จำของเสียงผู้ชายคนเดิมมีสูงเป็น 98 เปอร์เซนต์ ในขณะเดียวกันก็สามารถรู้จำเสียงของผู้พูดคนอื่นได้ 72 เปอร์เซนต์ โดยแยกเป็นชายหญิงได้ว่า สามารถรู้จำเสียงผู้ชายคนอื่นได้ 73.33 เปอร์เซนต์ และรู้จำเสียงผู้หญิงที่มาทดสอบได้สูงถึง 70 เปอร์เซนต์

กรณีที่ 3 กรณีนี้ใช้เสียงผู้หญิง 7 คน เป็นเสียงต้นแบบ ซึ่งเพิ่มจำนวนคนพูดเพิ่มขึ้นจากกรณีที่ 1 และ 2 คือ จาก 1 คน เป็น 7 คน ทำให้มีประสิทธิภาพการรู้จำของเสียงคนอื่นได้สูงขึ้น คือ 80 เปอร์เซนต์ เพราะเสียงต้นแบบมีความหลากหลายขึ้น แต่กระนั้นก็ยังสามารถจำเสียงผู้ชายได้ถึง 80 เปอร์เซนต์เช่นกัน กับรู้จำเสียงผู้หญิงคนอื่นซึ่งได้ 80 เปอร์เซนต์เช่นกัน ส่วนการรู้จำเสียงผ็หญิง 7 คนเดิมได้ 97.14 เปอร์เซนต์

กรณีที่ 4 กรณีนี้ใช้เสียงผู้ชาย 3 คน และผู้หญิง 7 คน เป็นเสียงต้นแบบ ซึ่งเป็นกรณีที่มีเสียงต้นแบบถึง 10 คน และมีทั้งเสียงผู้ชายและผู้หญิง ทำให้มีประสิทธิภาพการรู้จำเสียงได้ดีที่สุดกว่าทุกกรณี คือสามารถรู้จำเสียงผู้พูด 10 คนเดิมได้ 99 เปอร์เซนต์ และสามารถรู้จำเสียงของผู้พูดคนอื่นๆได้ถึง 95 เปอร์เซนต์

กรณีที่ 5 เป็นกรณีพิเศษจาก 4 กรณี ที่กล่าวมา คือ ผู้หญิง 1 คนพูดชื่อคน 20 ชื่อ แทนการพูดเสียง 0-9 ซึ่งมีประสิทธิภาพการรู้จำเสียงผู้หญิงคนเดิมได้ 95 เปอร์เซนต์ และสามารถรู้จำเสียงของกลุ่มทดสอบที่เป็นผู้หญิงทั้งหมดได้ 78.75 เปอร์เซนต์

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

5.1 สรุปผลการทดลอง

ในส่วนของการวิเคราะห์สัญญาณเสียงเพื่อเข้ารหัสนั้นได้ใช้วิธีการประมาณเชิงเส้น (Linear Predictive Coding : LPC) เนื่องจากเป็นวิธีที่วิเคราะห์พารามิเตอร์ได้แม่นยำและสามารถย่อข้อมูลได้อย่างมีประสิทธิภาพ และในการประมาณเชิงเส้น ซึ่งมีอยู่หลายวิธี และได้เลือกวิธีออโตคอร์รีเลชั่น เพราะเป็นวิธีที่มีการคำนวณที่ใช้สมการน้อยที่สุด

จากการทดลองในบทที่ 4 โดยใช้สัญญาณเสียงที่สุ่มด้วยความถี่ 11.025 กิโลเฮิร์ท ใช้การประมาณเชิงเส้นอันดับที่ 12 สรุปผลได้ดังนี้

1. ค่าสัมประสิทธิ์ของเสียงตัวเลขใดๆที่วิเคราะห์ได้ใน 1 เฟรม ประกอบไปด้วย

   เกน 1 ค่า

   สัมประสิทธิ์ LPC 12 ค่า

   สัมประสิทธิ์เซปสตรัม 19 ค่า

   โดยใน 1 เสียงจำนวนเฟรมที่วิเคราะห์จะขึ้นอยู่กับความยาวของสัญญาณเสียงที่พูด

2. ค่าสัมประสิทธิ์เซปสตรัมได้จากการปรับปรุงสัมประสิทธิ์ LPC โดยการใช้สมการเซปสตรัมเพื่อให้คงลักษณะของเสียงได้มากขึ้น เนื่องจากสัมประสิทธิ์เป็นพารามิเตอร์ที่มีลักษณะน่าเชื่อถือได้ดีกว่าสัมประสิทธิ์ LPC และมีความสัมพันธ์ใกล้ชิดกับการรับรู้เสียงตามความรู้สึกของมนุษย์โดยแท้จริง และก็จะได้จำนวนสัมประสิทธิ์จากการวิเคราะห์สัญญาณเสียง มากกว่าจำนวนสัมประสิทธิ์ LPC ซึ่งจะทำให้วิเคราะห์สัญญาณเสียงได้ละเอียดมากยิ่งขึ้น

3. การพูดเสียงต่างกันก็จะได้สัมประสิทธิ์แตกต่างกัน

4. การพูดเสียงเดียวกันหลายๆครั้งก็จะได้สัมประสิทธิ์ต่างกันทุกชุด

5. ใน 1 เสียงจะวิเคราะห์ทีละเฟรม ซึ่งในแต่ละเฟรมก็จะมีสัมประสิทธิ์เซปสตรัม 1 ชุดคือ 19 ตัว ดังนั้นใน 1 เสียงก็จะมีสัมประสิทธิ์หลายชุดที่แตกต่างกัน

6. ในการพูดนั้น ถ้าพูดด้วยเสียงที่เป็นธรรมชาติมากที่สุด ก็จะได้ค่าสัมประสิทธิ์ที่ถูกต้องมากยิ่งขึ้น

ในส่วนของการสร้างแบบจำลองเสียงนั้น ได้เลือกใช้วิธีการแบบ Hidden Markov Models(HMM) ที่เป็นแบบ Left-Right Model ชนิดไม่ขึ้นกับผู้พูด ซึ่งเป็นวิธีที่เหมาะสม ที่จะใช้หาต้นแบบเสียง และเหมาะสำหรับที่จะใช้กับต้นแบบในการรู้จำจำนวนมาก เพื่อให้ครอบคลุมความเป็นไปได้ของเสียง ซึ่งก็คือเมื่อเพิ่มจำนวนบุคคลที่ใช้เป็นต้นแบบให้มากขึ้นเรื่อยๆ เปอร์เซ็นต์ความถูกต้องในการทดสอบก็จะดีขึ้น ทั้งนี้ก็ขึ้นอยู่กับจำนวนโค้ดบุคที่ใช้ ในที่นี้ใช้จำนวนโค้ดบุคเท่ากับ 64 ก็เพียงพอแล้วสำหรับวิธีการแบบ HMM แต่จะมีขีดความสามารถการรู้จำลดลงเมื่อเพิ่มจำนวนเสียงต้นแบบจนถึงระดับหนึ่ง จึงจำเป็นต้องเพิ่มขนาดโค้ดบุคให้มากขึ้นกว่านี้ ซึ่งการเลือกค่าพารามิเตอร์ต่างๆนี้ แสดงดังภาคผนวก

5.2 วิจารณ์

1. สัมประสิทธ์ LPC ที่ได้จำเป็นจะต้องเปลี่ยนเป็นสัมประสิทธ์เซปสตรัม เนื่องจากสัมประสิทธ์ LPC จะมีความคลาดเคลื่อนมากกว่า

2. การออกเสียงแต่ละครั้งจะต้องให้เป็นธรรมชาติมากที่สุด มิฉะนั้นจะได้ค่าสัมประสิทธ์ที่คลาดเคลื่อนจากความเป็นจริง

3.การออกเสียงแต่ละครั้งของคำหนึ่งคำจะไม่เหมือนเดิม อาจจะสั้นหรือยาวกว่าเดิม ทำให้ความยาวของสัญญาณเสียงและจำนวนเฟรมไม่เท่าเดิม และ codebook index ที่ได้ก็จะไม่เหมือนเดิม เมื่อนำไปผ่านกระบวนการการทดสอบการรู้จำโดย Viterbi Algorithm จะทำให้การรู้จำผิดพลาดได้ นอกจากนี้ความผิดพลาดอาจเกิดขึ้นเนื่องจาก

- ในขั้นตอนการจัดเก็บเสียง ทำการจัดเก็บเสียงที่ผิดปกติเพื่อนำไปสร้างแบบจำลอง ทำให้ได้แบบจำลองที่ไม่สมบูรณ์

- ในกระบวนการตัดหัวท้ายของเสียงนั้นไม่สมบูรณ์เพียงพอ ยังมีการตัดคำเกินหรือขาดไป จึงควรมีอัลกอริทึมที่มีประสิทธิภาพในการตัดหัวท้ายของเสียง จะทำให้ได้หัวเสียงและท้ายเสียงของคำนั้นอย่างสมบูรณ์

- โค้ดบุคที่ได้ไม่มาตรฐานเพียงพอ ทำให้ได้ codebook index จากขั้นตอนการควอนไตซ์ซึ่งต้องอ้างอิงกับโค้ดบุคออกมาผิดพลาด ทำให้ความน่าจะเป็นเปลี่ยนไป จึงทำให้การตัดสินใจในขั้นตอนการทดสอบการรู้จำผิดพลาดได้

4. ในขั้นตอนการสร้างโค้ดบุคและการสร้างแบบจำลองเสียงต้องทำการสุ่มค่าเริ่มต้นหลายๆครั้งเพื่อให้ได้ค่าที่เหมาะสมที่สุดจึงจะได้แบบจำลองที่ดีที่สุด จากผลการทดลองนี้ทำการสุ่มค่าเริ่มต้นในการสร้างโค้ดบุคเพื่อหา codebook index แล้วนำไปสร้างแบบจำลองของแต่ละเสียงเพียง1-3 ครั้ง จึงไม่ใช่แบบจำลองเสียงที่ดีที่สุด

5. ในขั้นตอนการต่อโทรศัพท์นี้ เป็นเพียงแนวทางที่จะนำไปสู่ความสะดวกในการใช้งานจริง การเขียนโปรแกรมสั่งงานให้โมเด็มหมุนหมายเลขโทรศัพท์ที่ตรงกับคำที่จำได้จึงยังต้องใช้ระบบ manual อยู่บ้าง

5.3 แนวทางพัฒนา

1.จำนวนโค้ดบุคขนาด 64 โค้ดบุค ที่ใช้ในปริญญานิพนธ์นี้เพียงพอกับการรู้จำเสียง ถ้ามีการใช้ต้นแบบเสียงที่มากกว่านี้ ควรจะเพิ่มขนาดโค้ดบุคให้มากกว่านี้ตามความเหมาะสม

2.HMM เป็นวิธีการที่สามารถใช้กับการรู้จำคำต่อเนื่อง ในปริญญานิพนธ์นี้สามารถสั่งงานโมเด็มให้ต่อโทรศัพท์ได้ แต่ยังต้องอาศัยการเก็บหมายเลขโทรศัพท์เป็นฐานข้อมูลของคำศัพท์เดี่ยวอยู่ ดังนั้นเพื่อความสะดวกในการใช้จึงควรพัฒนาให้รู้จำคำต่อเนื่องที่เป็นคำสั่งสั้นๆ เพื่อให้สามารถเรียกหมายเลขได้เลยโดยไม่ต้องเก็บเป็นฐานข้อมูลของคำศัพท์แต่ละคำนั้นๆ

3.โปรแกรมการสั่งงานโมเด็มให้ต่อโทรศัพท์ยังต้องอาศัยการตัดหัวท้ายเสียงของเสียงทดสอบก่อน หลังจากนั้นจึงค่อยรันโปรแกรมสั่งงานโมเด็มให้ต่อโทรศัพท์อีกที ทำให้เกิดความล่าช้า จึงควรพัฒนาโปรแกรมให้มีอัลกอริทึมในการตัดหัวท้ายเสียงได้เลย จะทำให้สามารถสั่งงานให้ต่อโทรศัพท์ได้ทันทีที่พูดเสร็จ

4. พัฒนาโปรแกรมที่ได้สร้างเป็นวงจร (Hardware) เพื่อเป็นแนวทางในการพัฒนาเพื่อสร้างเป็นเครื่อง โทรศัพท์แบบต่อหมายเลขโทรศัพท์ได้ด้วยเสียง

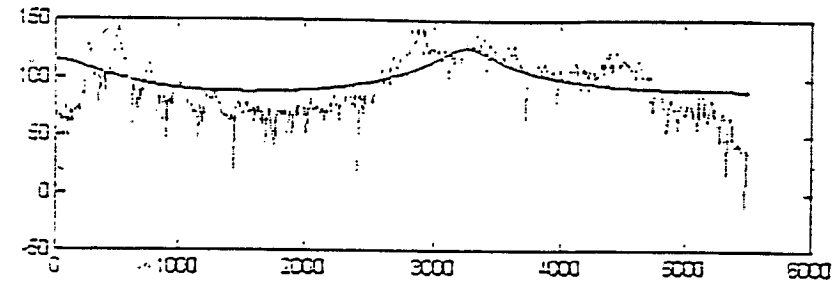
ภาคผนวก

ภาคผนวก ก

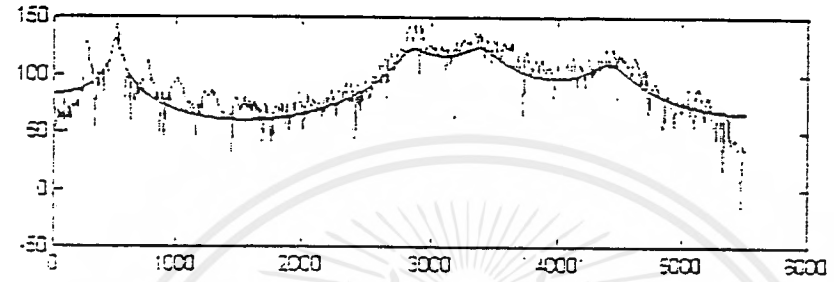
การใช้ค่าพารามิเตอร์ที่เหมาะสมเพื่อใช้ในการทดลอง

1. จำนวนสัมประสิทธิ์ (P)

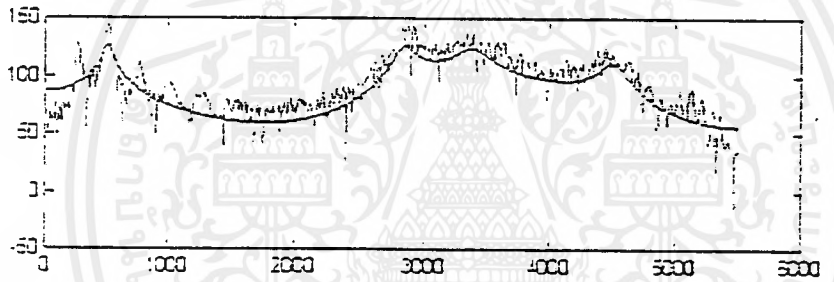
มีการทดลองการประมวลเสียงพูดโดยการประมาณเชิงเส้น(สุนทร อรอินทร์ และ อัฐ เครือพัก 2537) ซึ่งเป็นการทดลองที่มีการนำค่าสัมประสิทธิ์ LPC ที่หาได้โดยการใช้วิธีอัตสัมพันธ์ มาทำการสังเคราะห์เสียงขึ้นมาใหม่ จากการทดสอบดังกล่าว ให้ผลเป็นที่น่าพอใจ งานปริญญานิพนธ์ชิ้นนั้นมีการใช้ LPC ด้วยจำนวน P= 4 , 8 , 12 ,16 และ 20 ซึ่งใช้สำหรับเปรียบเทียบสเปคตรัม และสัญญาณความผิดพลาดระหว่างเสียงจริงและเสียงที่ได้จากการที่สังเคราะห์ขึ้นด้วยการประมาณเชิงเส้น ได้ผลสรุปว่า การประมาณสเปคตรัมจะมีความใกล้เคียงเสียงจริงมากยิ่งขึ้น เมื่อค่า P ยิ่งสูงขึ้น ดังรูปที่ ก.1 ส่วนรูปที่ ก.2 เป็นการแสดงสัญญาณความผิดพลาดที่เกิดจากการประมาณเชิงเส้น พบว่าสัญญาณความผิดพลาดจะมีขนาดลดลงเมื่อค่า P สูงขึ้น อย่างไรก็ดีถึงแม้การเพิ่มค่า P จะทำให้การประมาณเสียงมีความใกล้เคียงมากยิ่งขึ้น แต่ก็ทำให้การคำนวณมีความยุ่งยาก และใช้เวลานานขึ้น ดังนั้น เพื่อให้ได้ความเหมาะสม ปริญญานิพนธ์นี้จึงเลือกใช้ค่า P=12



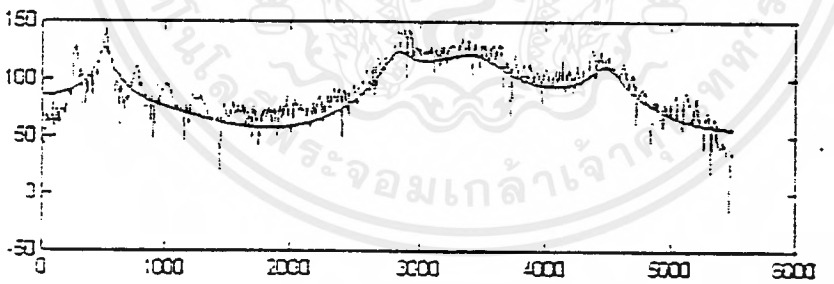
(ก)



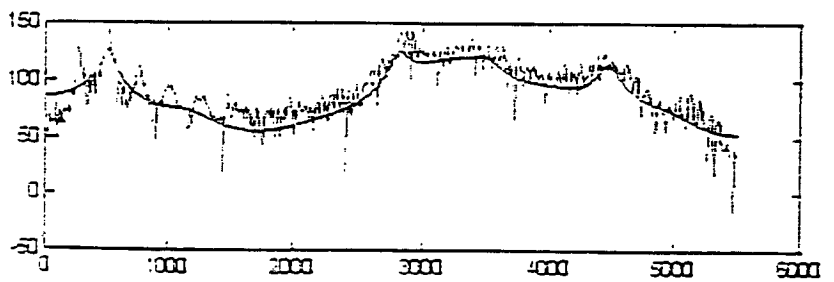
(ข)



(ค)



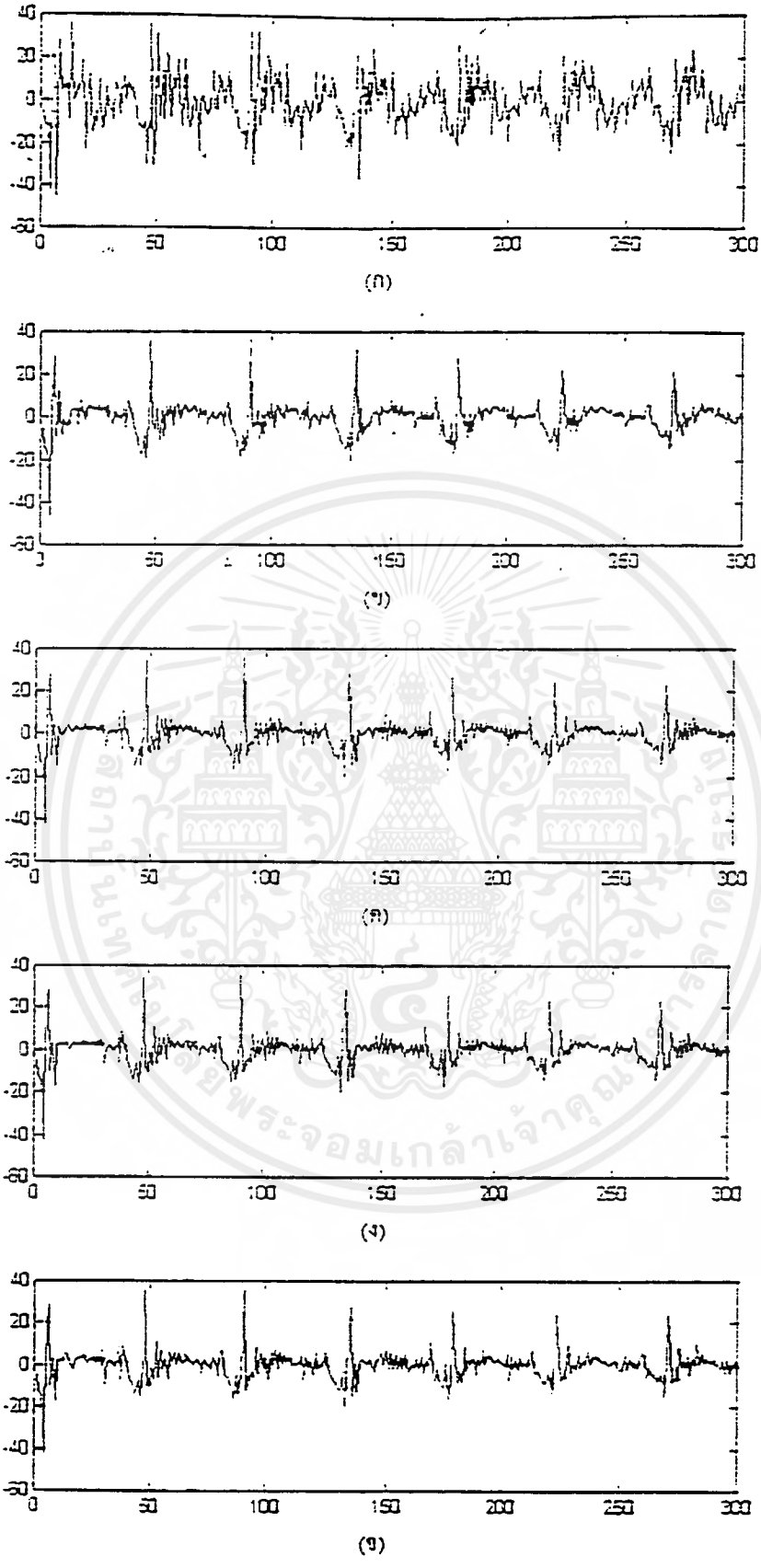
(ง)



(จ)

รูปที่ ก.1 สเปคตรัมของเสียงเปรียบเทียบกับสเปคตรัมที่ได้จากสัมประสิทธิ์ LPC

(ก) p=4 (ข) p=8 (ค) p=12 (ง) p=16 (จ) p=20



รูปที่ ก.2 สัญญาณความลลาดเคลื่อน

(ก) p=4 (ข) p=8 (ค) p=12 (ง) p=16 (จ) p=20

2. ขนาดของโค้ดบุค

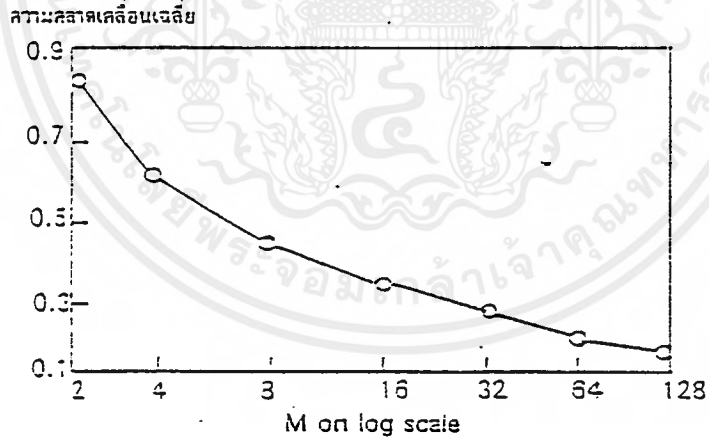
ในการทดสอบหาค่าขนาดของโค้ดบุคที่เหมาะสม (L.R. Rabiner , S.E. Levinson และ M.M. Sondhi,1982) เป็นการทดสอบโดยใช้เวคเตอร์ของ LPC จำนวน 39,708 เวคเตอร์ จากคน 100 คน (ชาย 50 คน หญิง 50 คน) พูดตัวเลขภาษาอังกฤษ 0-9 จากนั้นนำเวคเตอร์มาหาค่าควอนไตซ์ที่โค้ดบุคขนาดต่าง ๆ M = 2 ,4 , 8 , 16 , 32 , 64 , 128 ใช้การพิจารณาดังนี้

ค่าความคลาดเคลื่อนเฉลี่ย $\|D_m\|$

$$\|D_m\| = \min_{\hat{a}_m} \left\{ \frac{1}{I} \sum_{i=1}^{I} \min_{1 \le m \le M} \left[d(\hat{a}_m, a_i) \right] \right\}$$

โดย a_i เป็นเวคเตอร์ LPC I= 1,2,3,...,I $\hat{a}_m$ เป็นโค้ดบุค LPC เวคเตอร์ m = 1 , 2 , ... , M

จากผลการทดสอบจะได้ดังรูป ก.3 โดยรูป ก.3 เป็นการพลอตค่า $\|D_m\|$ กับค่า M (ขนาดโค้ดบุค โดยใช้ log scale) โดย M มีค่า 2 - 128 เราจะเห็นว่าที่ $M \geq 32$ ค่าความคลาดเคลื่อนเฉลี่ยจะต่ำกว่า 0.3 ถ้า M = 64 จะมีค่า $\|D_m\|$ ประมาณ 0.2 ซึ่งค่า $\|D_m\| < 0.3$ เป็นค่าความคลาดเคลื่อนที่น้อยมากสำหรับเวคเตอร์ ควอนไตซ์เซชัน (S.E. Levinson,L.R. Rabiner , A.E. Rosenberg และ J.G. Wilpon,1979) ดังนั้นสำหรับวิธีของ HMM ควรใช้ M = 64 เพราะจะมีค่าความคลาดเคลื่อนเฉลี่ยน้อย เมื่อ M = 64 - 128 (L.R. Rabiner , S.E. Levinson และ M.M. Sondhi,1982)



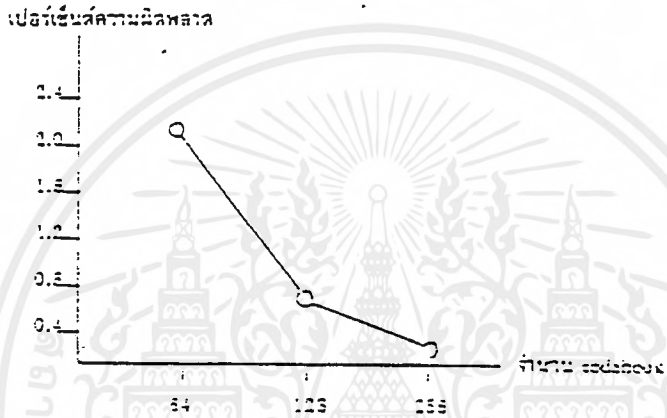
รูป ก.3 เปรียบเทียบค่าความคลาดเคลื่อนเฉลี่ยกับขนาดโค้ดบุค

จากผลการทดสอบโดยใช้โปรแกรมที่พัฒนาขึ้น (วิทยานิพนธ์ เสาวลักษณ์ อารีย์พงศา 1995) นำข้อมูลจากกลุ่มต้นแบบกลุ่มที่ 1 และกลุ่มที่ 2 จำนวน 20 คน มาหาแบบจำลองต้นแบบ และทำการทดสอบความถูกต้องในการรู้จำ

2.1) ใช้กลุ่มที่ 1 และกลุ่มที่ 2 ทดสอบกับแบบจำลองต้นแบบที่คำนวณได้ เปรียบเทียบกับขนาดโค้ดบุคดังตารางที่ ก.1 และกราฟรูปที่ ก.4

ตาราง ก.1 เปรียบเทียบความผิดพลาดเป็นร้อยละกับขนาดโค้ดบุคของกลุ่มต้นแบบ

จำนวนโค้ดบุค	ความผิดพลาด (%)
64	2.25
128	0.75
256	0.25

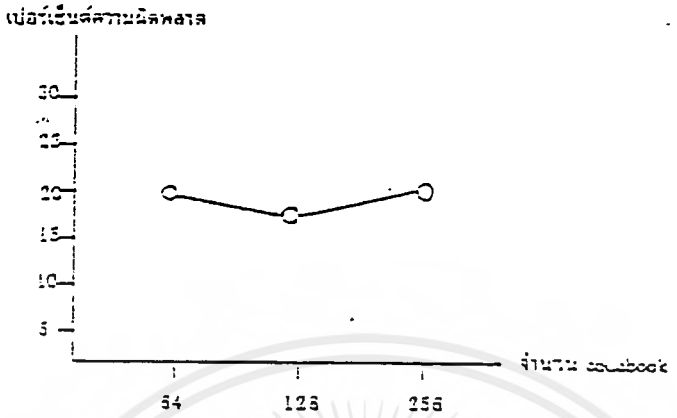


รูปที่ ก.4 เปรียบเทียบความผิดพลาดเป็นร้อยละกับขนาดโค้ดบุคของกลุ่มต้นแบบ

2.2) ใช้กลุ่มทดสอบมาทดสอบกับแบบจำลองต้นแบบที่คำนวณได้ ซึ่งเป็นบุคคลที่ไม่มีการฝึกฝน ดังแสดงผลความผิดพลาดในการรู้จำที่ขนาดโค้ดบุคต่าง ๆ ดังตารางที่ ก.2 และรูปที่ ก.5

ตารางที่ ก.2 เปรียบเทียบความผิดพลาดเป็นร้อยละกับขนาดโค้ดบุคของกลุ่มทดสอบ

จำนวนโค้ดบุค	ความผิดพลาด(%)
64	20
128	18
156	21

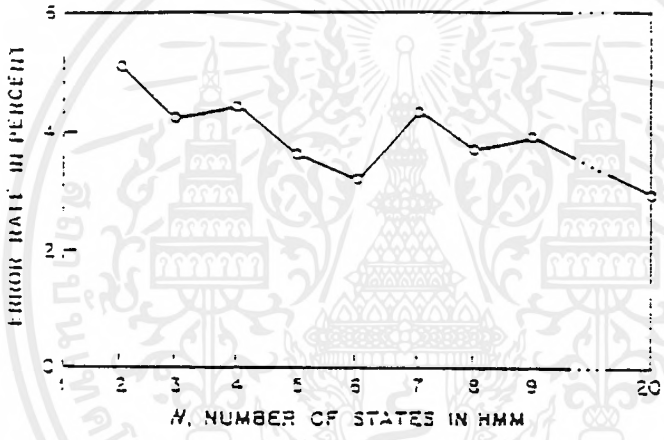


รูปที่ ก.5 เปรียบเทียบความผิดพลาดเป็นร้อยละกับขนาดโค้ดบุคของกลุ่มทดสอบ

3. การเลือกพารามิเตอร์ของแบบจำลอง (model)

ในปริญญานิพนธ์นี้ได้เลือกสร้างแบบจำลอง HMM เป็นแบบ left-right เพราะเป็นแบบจำลองที่เหมาะสำหรับการจดจำรูปแบบของคำ ประเภทคำศัพท์เดี่ยว (Isolated word) เนื่องจากจะสามารถนำเวลาเข้ามาเกี่ยวข้องกับสเตท (state) ของแบบจำลองได้โดยตรง และอาจตีความหมายทางกายภาพของสเตทเป็นเสียงที่แตกต่างกันของคำได้

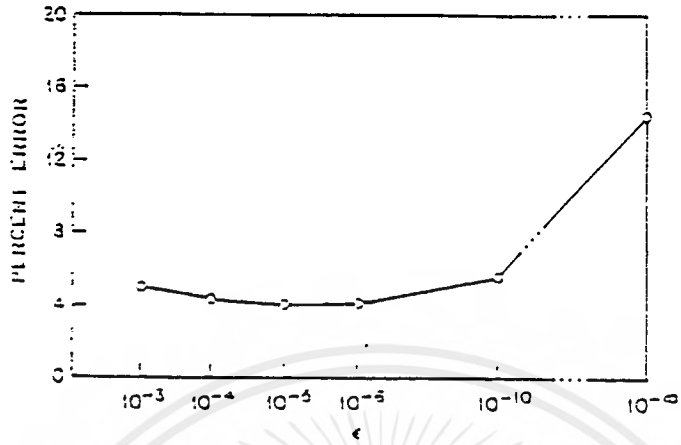
การเลือกจำนวนสเตท จะใช้จำนวนสเตทเท่า ๆ กันทุก ๆ คำ (0 - 9) จากรูปที่ ก.6 แสดงอัตราการผิดพลาด เมื่อใช้ จำนวนสเตท (N) ค่าต่าง ๆ สำหรับตัวเลขคำเดี่ยว ซึ่งจะเห็นว่ามีค่าต่ำสุดที่ N = 6 ในปริญญานิพนธ์นี้จึงเลือกใช้จำนวนสเตทเท่ากับ 6 สเตท



รูปที่ ก.6 แสดงอัตราการผิดพลาดเมื่อใช้จำนวนสเตท (N) ค่าต่าง ๆ

สำหรับเวคเตอร์เหตุการณ์ เนื่องจากในปริญญานิพนธ์นี้ทำแบบจำลองเป็นแบบไม่ต่อเนื่องตามเวลา เราได้ใช้วิธีการประมาณเชิงเส้นร่วมกับสัมประสิทธิ์เซปสตรัมและการเวทค่าพารามิเตอร์ ทั้งยังใช้โค้ดบุคในการสร้างสัญลักษณ์แบบไม่ต่อเนื่องขึ้น

ในการเทรน (traing) HMM นั้นต้องมีการจำกัดการคำนวณพารามิเตอร์ $b_j(k)$ เมื่อป้องกันไม่ให้คำนั้นน้อยเกินไปโดยการกำหนดค่าที่น้อยที่สุดที่จะเป็นไปได้ไว้ ($\in$) นั่นคือค่า $b_j(k)$ จะต้องมากกว่าหรือเท่ากับค่านี้ เนื่องจากอาจจะไม่เกิดเหตุการณ์ที่ k^{th} เกิดขึ้นเลยในสเตท j ในเหตุการณ์ที่เอาเข้ามาเทรน ซึ่งทำให้ได้ค่าพารามิเตอร์ที่ไม่ถูกต้อง และจำกัดค่าความน่าจะเป็นของการเกิดเหตุการณ์ลำดับของเหตุการณ์ที่เราไม่รู้ การกำหนดค่าพารามิเตอร์ $\in$ นี้จะอยู่ในช่วง $10^{-10} \leq \in \leq 10^{-3}$ ซึ่งรูปที่ ก.7 ได้แสดงค่าความผิดพลาดในกรณีที่ใช้ $\in$ ค่าต่าง ๆ ในปริญญานิพนธ์นี้เลือกใช้ค่า $\in = 10^{-5}$



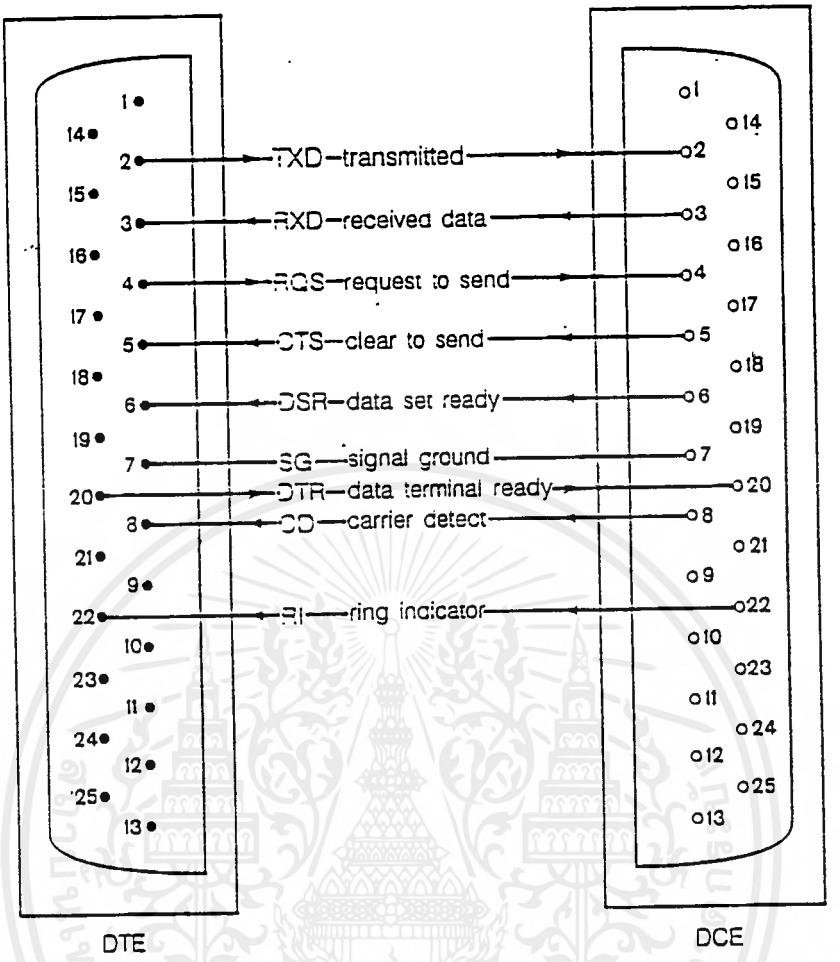
รูปที่ ก.7 แสดงค่าความผิดพลาดในกรณีที่ใช้ $\in$ ค่าต่าง ๆ

ภาคผนวก ข

การเชื่อมต่อ RS-232

วงจรสมบูรณ์ RS-232

PIN	CIRCUIT	ABBREVIATION	FULL NAME	DIRECTION
DATA:				
2	BA	TXD	Transmitted data	DTE to DCE
3	BB	RXD	Received data	DCE to DTE
Primary Handshaking Lines:				
6	CC	DSR	Data Set ready	DCE to DTE
20	CD	DTR	Data terminal ready	DTE to DCE
Secondary Handshaking Lines:				
4	CA	RTS	Request to send	DTE to DCE
5	CB	CTS	Clear to send	DCE to DTE
Modem Line:				
8	CF	CD	Carrier detect	DCE to DTE
22	CE	RI	Ring Indicator	DCE to DTE
Ground or Common:				
7	AB	SG	Signal ground	
Less commonly Used Circuit:				
1	AA		Protective ground	
12	SCF		Secondary received line signal detector	DCE to DTE
13	SCE		Secondary Clear to send	DTE to DCE
14	SBA		Secondary Transmitted data	DTE to DCE
15	DB		Trasmitter signal Element timing	DCE to DTE
16	SBB		Secondary Received data	DCE to DTE
17	DD		Received signal element timing	DCE to DTE
19	SCA		Secondary request to send	DTE to DCE
21	CG		Signal Quality detector	DCE to DTE
23	CH		Data signal rate selector	DTE to DCE
23	CI		Data signal rate selector	DCE to DTE
24	DA		Transmitter signal element timing	DTE to DCE



รูป ข.1 แสดงการเชื่อมต่อ DTE (ไอบีเอ็มพีซี) ไปยัง DCE(โมเด็ม) แบบมาตรฐาน

ภาคผนวก ค

โฟลว์ชาร์ตและโปรแกรมที่ใช้ในปริญญานิพนธ์นี้

-รูปที่ 1 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม LPC

-รูปที่ 2 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมโค้ดบุค

-รูปที่ 3 โฟลว์ชาร์ตแสดงการทำงานของการรับรู้เสียงพูด

-รูปที่ 4 โฟลว์ชาร์ตแสดงการสร้างแบบจำลอง HMM ทีละเสียง

-รูปที่ 5 โฟลว์ชาร์ตแสดงการจำเสียงได้ของแบบจำลอง HMM

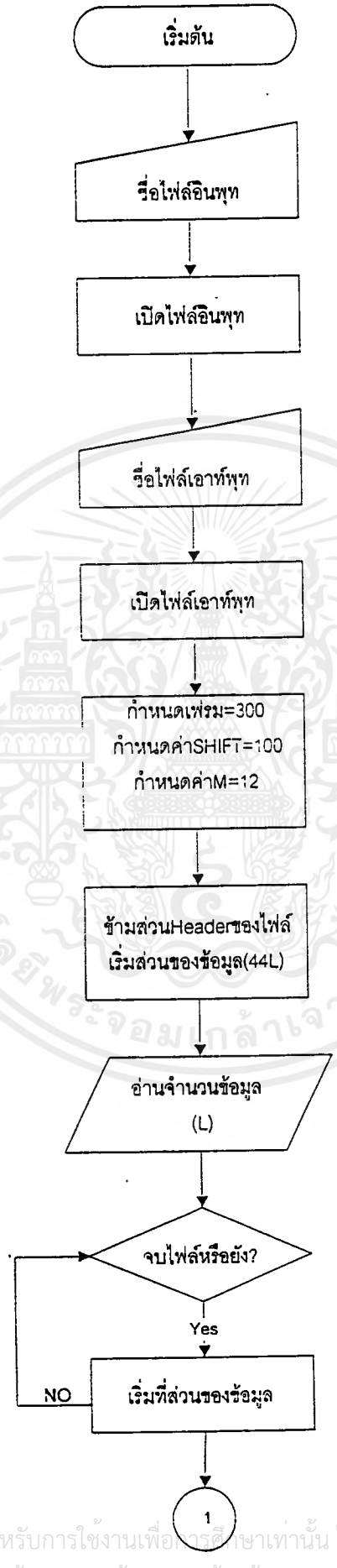
-รูปที่ 6 โฟลว์ชาร์ตแสดงการทำงานของการต่อโทรศัพท์โดยใช้เสียง

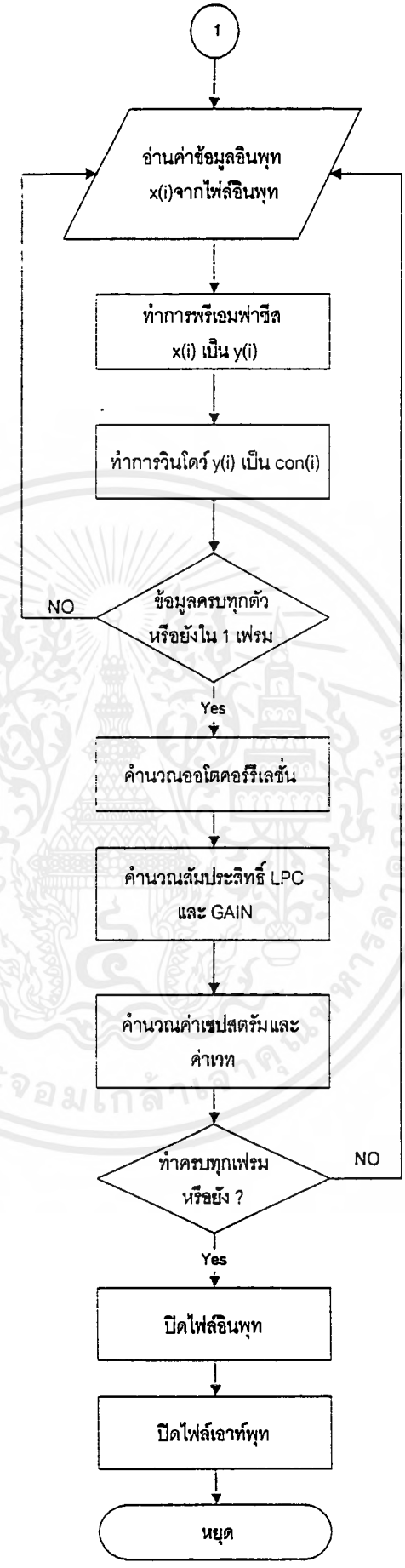
-โปรแกรมที่ใช้ในการสร้างแบบจำลองเสียง คือ HMM.C

-โปรแกรมที่ใช้ในการทดสอบเสียง คือ ASR.C

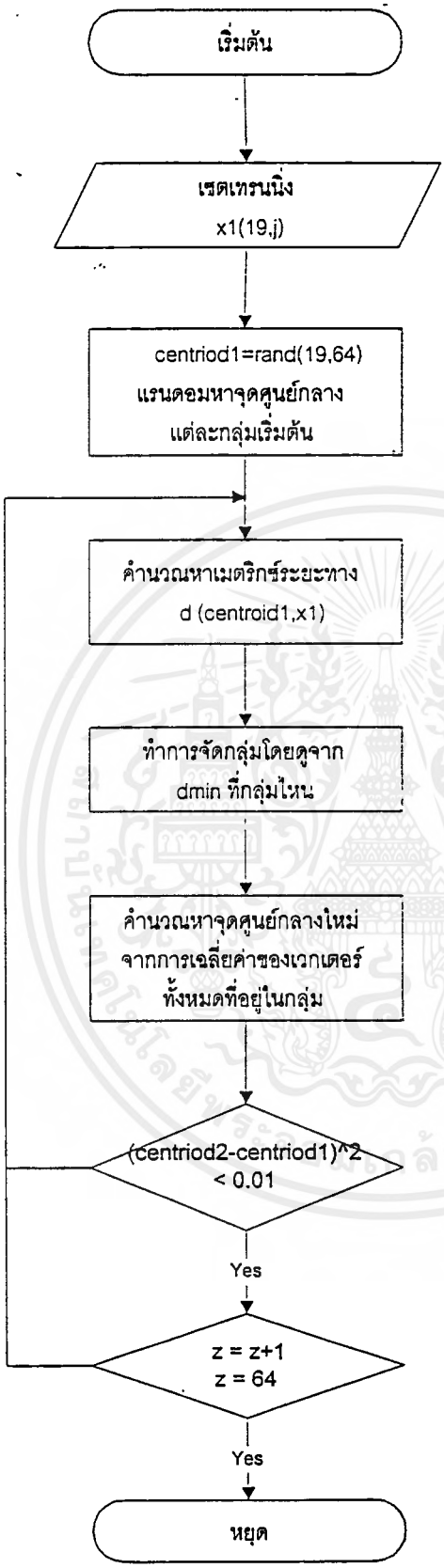
-โปรแกรมที่ใช้ในการสร้างโค้ดบุค คือ VQLN.C

-โปรแกรมที่ใช้ในการสั่งงานโมเด็มให้ต่อโทรศัพท์ได้ด้วยเสียง คือ ASRD_APP.C

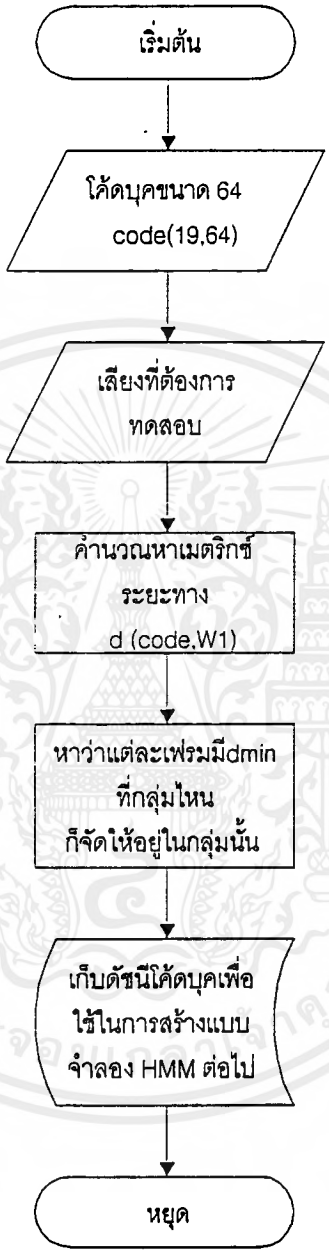




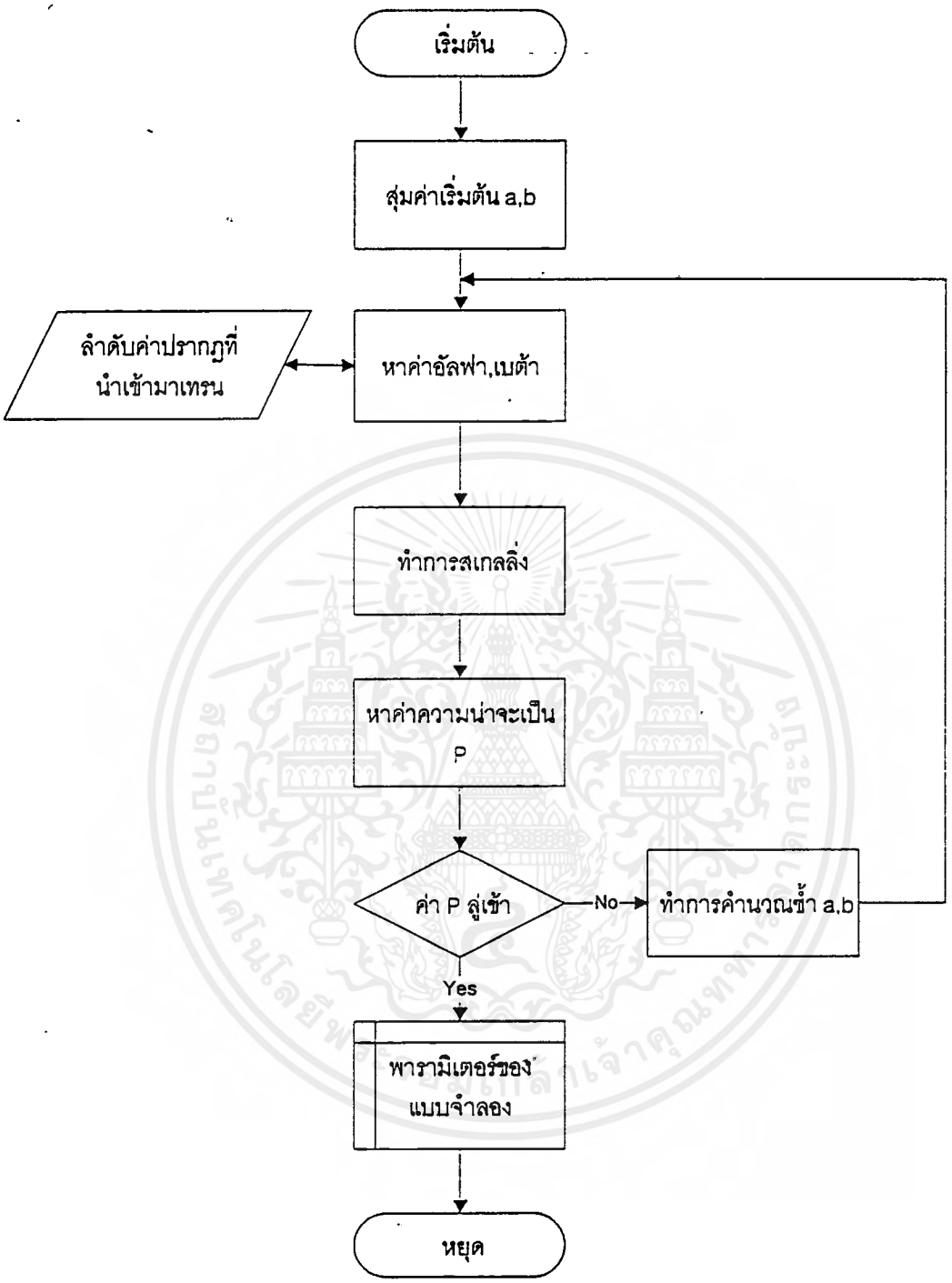
รูปที่ 1 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมLPC



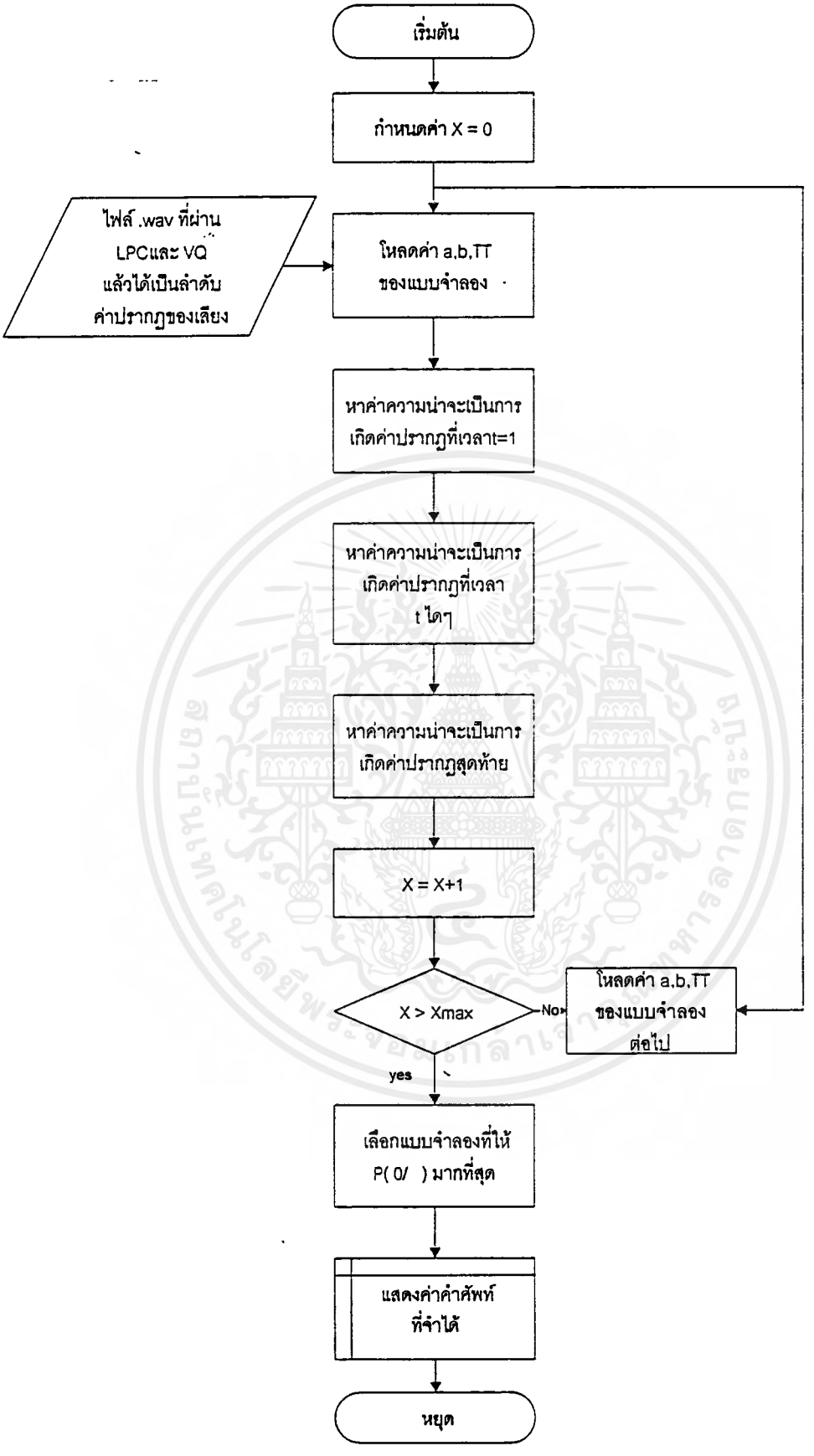
รูปที่ 2 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมโค้ดบุค



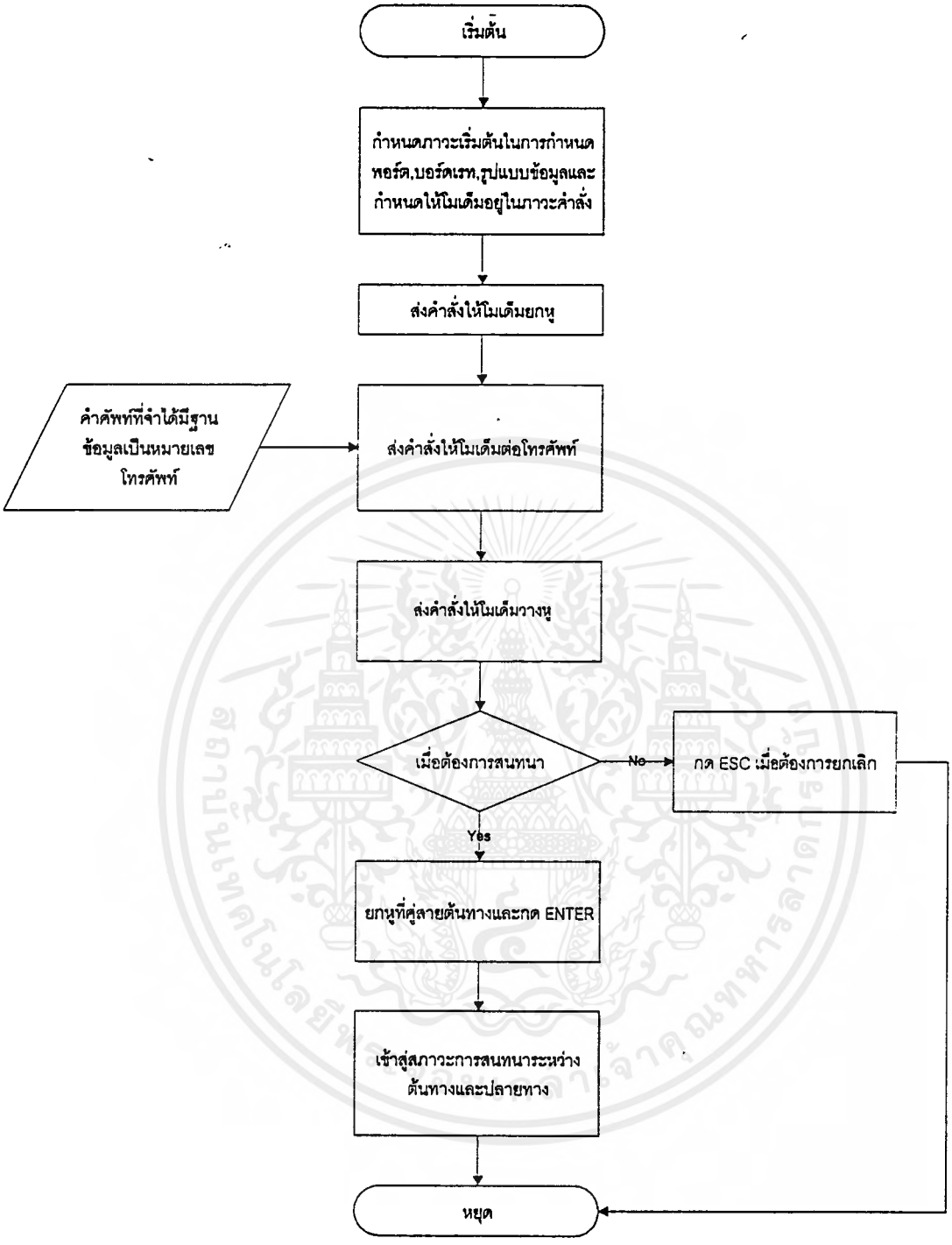
รูปที่ 3 โฟลว์ชาร์ตแสดงการทำงานของการรับรู้เสียงพูด



รูปที่4โฟลว์ชาร์ตแสดงการสร้างแบบจำลอง HMM ทีละเสียง



รูปที่ 5 โฟลว์ชาร์ตแสดงการจำเสียงได้ของแบบจำลอง HMM



รูปที่ 6 โฟลว์ชาร์ตแสดงการทำงานของการต่อโทรศัพท์โดยใช้เสียง

โปรแกรม HMM.C

```
/********HIDDEN MARKOV MODELS FOR SPEECH RECOGNITION ******/
/***************input  : Observation Sequence*************/
/***************output : Markov models of Voice************/
#include <stdlib.h>
#include <stdio.h>
#include  <conio.h>
#include   <time.h>
#include   <math.h>
#define N           6   /* number of state in each model  */
#define K          64   /* number of distinct observation */
#define ROUND      50   /* number of reestimate round     */
#define MIN_B  0.00001  /* minimum value of b[][]         */

FILE   *observation_sequence_file;
FILE   *model_file;
FILE   *model_text_file;
char   observation_sequence_file_name[50];
char   model_file_name[50];
char   model_text_file_name[50];
char   *T;      /* number of observation for each training */
char   *O;           /* training sequence */
int    number_of_observation;
int    i,j,k,w,v,t,max;
double Prob_log[ROUND];/* check distortion in each round  */
float  Pi[N] = {1,0,0,0,0,0};
float  a[N][N], aprime[N][N];
float  b[N][K], bprime[N][K];
double  x[N][N], y[N][K], z[N];
double Aprime[N][N], AMprime[N];
double Bprime[N][K], BMprime[N], l[N];
float imb[N];
double *alpha,          /* alpha coefficient*/
       *pres_alpha,  /* alpha coefficient prepare scale */
       *alphas,         /* alpha coefficient scaled */
       *beta,            /* beta coefficient*/
       *pre_beta,      /* beta coefficient prepare scale */
       *betas,          /* beta coefficient scaled */
       *sc,
       *c;                /* scaling coefficient */
float  *prob_log;       /* for check distorsion */
/*****************************************************/
void open_file(void)
{  char temp;
   clrscr();
  /* open observation file */
  printf("\n Please enter observation (input) index file
name : ");
  gets(observation_sequence_file_name);
  if ((observation_sequence_file =
fopen(observation_sequence_file_name,"rb")) == NULL)
    {      printf("\n\7 þ Cannot open input file !");
exit(1);    };
  /* open model output file */
  printf("\n Please enter Model (output) file name : ");
  gets(model_file_name);
  if ((model_file = fopen(model_file_name,"wb")) == NULL)
```

```
    {     printf("\n\7 þ Cannot open output file !");
exit(1);    };
 printf("\n Please enter Model TEXT (output) file name : ");
 gets(model_text_file_name);
 if ((model_text_filem = fopen(model_text_file_name,"wt"))==
NULL)
    {     printf("\n\7 þ Cannot open output text file !");
exit(1);    };
  /* count number of observation in observation file */
  number_of_observation = 0;
  for (;;)
   {     fread(&temp,1,1,observation_sequence_file);
     if (temp != 0 )  number_of_observation++; else break;
     if (feof(observation_sequence_file))
      {    printf("\n\7 þ Invalid input file !\n");
exit(1);      };
   }
  rewind(observation_sequence_file);
  /*********** read each observation index length *********/
  T = (char *) calloc(number_of_observation,sizeof(char));
  if (T == NULL)
    {     printf("\n\7 þ Cannot allocate memory for data
!\n");  exit(1);    };

fread(T,1,number_of_observation,observation_sequence_file);
  fread(&temp,1,1,observation_sequence_file);  /* for bypass
FLAG (-1) */
  /********* find maximum of number of observation ********/
  max = T[0];
  for (i=1;i<number_of_observation;i++)
    if (max < T[i]) max = T[i];
  O = (char *)
calloc(number_of_observation*max,sizeof(char));
  if (O == NULL)
    {     printf("\n\7 þ Cannot allocate memory for data
!\n");  exit(1);    };
  /*************** read observation data ****************/
  for (i=0;i<number_of_observation;i++)
   {     fread(&O[i*max],1,T[i],observation_sequence_file);
}
  /******************** allocate memory ******************/
  alpha      = calloc(max*N,sizeof(double));
  pres_alpha = calloc(max*N,sizeof(double));
  alphas     = calloc(max*N,sizeof(double));
  beta       = calloc(max*N,sizeof(double));
  pre_beta   = calloc(max,  sizeof(double));
  betas      = calloc(max*N,sizeof(double));
  sc         = calloc(max,  sizeof(double));
  c          = calloc(max,  sizeof(double));
  prob_log   = calloc(number_of_observation,sizeof(float));
  if
((alpha||pres_alpha||alphas||beta||pre_beta||betas||sc||c||
prob_log) == NULL)
    {    printf("\n\7 Cannot allocate memory for data !\n");
exit(1);    };
}
/*********************************************************/
void random_abvalue(void)
{float sum_a[N],sum_b[N];/* Left-Right Model conditions */
```

```
a[0][0]=(1.00/3.00); a[0][1]=(1.00/3.00);
a[0][2]=(1.00/3.00); a[0][3]=0;
a[0][4]=0;           a[0][5]=0;
a[1][0]=0;           a[1][1]=(1.00/3.00);
a[1][2]=(1.00/3.00); a[1][3]=(1.00/3.00);
a[1][4]=0;           a[1][5]=0;
a[2][0]=0;           a[2][1]=0;
a[2][2]=(1.00/3.00); a[2][3]=(1.00/3.00);
a[2][4]=(1.00/3.00); a[2][5]=0;
a[3][0]=0;           a[3][1]=0;
a[3][2]=0;           a[3][3]=(1.00/3.00);
a[3][4]=(1.00/3.00); a[3][5]=(1.00/3.00);
a[4][0]=0;           a[4][1]=0;
a[4][2]=0;           a[4][3]=0;
a[4][4]=(1.00/2.00); a[4][5]=(1.00/2.00);
a[5][0]=0;           a[5][1]=0;
a[5][2]=0;           a[5][3]=0;
a[5][4]=0;           a[5][5]=(1.00);
  /* random start b[][] value */
  randomize();
  for(i=0;i<N;i++)
    for(k=0;k<K;k++)
      {    b[i][k]=((float)(random(99)+1));      }

  /* find a_prime and b_prime */
  for(i=0;i<N;i++)
   {  sum_a[i] = 0;     sum_b[i]  = 0;
      for(j=0;j<N;j++) sum_a[i] += a[i][j];
      for(k=0;k<K;k++) sum_b[i] += b[i][k];
      printf("sum_a[%d] = %f\n",i,sum_a[i]);
      printf("sum_b[%d] = %f\n",i,sum_b[i]);
      getch();
   }
  for(i=0;i<N;i++)
    {      for(j=0;j<N;j++) aprime[i][j] = a[i][j]/sum_a[i];
           for(k=0;k<K;k++) bprime[i][k] = b[i][k]/sum_b[i];
    }
}
/*************************************************************/
void copy_abprime_to_ab(void)
{  for(i=0;i<N;i++)
    for(j=0;j<N;j++)
      a[i][j] = aprime[i][j];
 for(i=0;i<N;i++)
    for(k=0;k<K;k++)
      b[i][k] = bprime[i][k];
}
/*************************************************************/
void find_alpha_bt_value(void)
{  float sum_alpha;
 /* Alpha Initialization */
  for(i=0;i<N;i++)
    alpha[i] = Pi[i] * b[i][O[v*max+0]];
  /* Alpha Induction */
  for(t=0;t<T[v]-1;t++)
    for(j=0;j<N;j++)
     {    sum_alpha = 0;
      for(i=0;i<N;i++)
        sum_alpha += alpha[t*N+i] * a[i][j];
```

```
      alpha[(t+1)*N+j] = sum_alpha * b[j][O[v*max+(t+1)]];
      }
  /* Beta Initialization */
  for(i=0;i<N;i++)
    beta[(T[v]-1)*N+i] = 1;
  /* Beta Induction */
  for(t=T[v]-2;t>=0;t--)
    for(i=0;i<N;i++)
     {    beta[t*N+i] = 0;
     for(j=0;j<N;j++)
       beta[t*N+i] +=
a[i][j]*b[j][O[v*max+(t+1)]]*beta[(t+1)*N+j];
      }
}
/*************************************************************/
void scaling_alpha_bt(void)
{  for (i=0;i<(max*N);i++)  pres_alpha[i] = 0;
  for (i=0;i<max    ;i++)    pre_beta[i] = 0;
 /* find sum coefficient */
  for(i=0;i<T[v];i++)
   {
      sc[i] = 0;
      for(j=0;j<N;j++)  sc[i] += alpha[i*N+j];
   }
   /*-------------- find alpha scale --------------*/
   c[0] = 1/sc[0];  /* at  t = 0  */
   for(i=0;i<N;i++)
    {       pres_alpha[i] = alpha[i];
            alphas [i] = c [0] * alpha[i];    }
   /* at t=1 to T */
   for(t=1;t<T[v];t++)
    {       c[t] = 0;
            for(i=0;i<N;i++)
     { pres_alpha[t*N+i] = 0;
       for(j=0;j<N;j++)
       pres_alpha[t*N+i] += alphas[(t-
       1)*N+j]*a[j][i]*b[i][O[v*max+t]];
        c[t] += pres_alpha[t*N+i];
     }
       c[t] = 1/c[t];
       for(i=0;i<N;i++)
      alphas[t*N+i] = c[t] * pres_alpha[t*N+i];
    }
  /*----------------- find beta scale -----------------*/
  pre_beta[T[v]-1] = c[T[v]-1];
  for(t=T[v]-2;t>=0;t--)
    pre_beta[t] = pre_beta[t+1] * c[t];
  for(t=0;t<T[v];t++)
    for(i=0;i<N;i++)
      betas[t*N+i] = pre_beta[t] * beta[t*N+i];
}
/*************************************************************/
void find_logP(void)
{  prob_log[v] = 0;
  for(t=0;t<T[v];t++)
  prob_log[v] += log10(c[t]);
  prob_log[v]  = -prob_log[v];
  Prob_log[w] += prob_log[v];
}
```

```
/*********************************************************/
void find_A_AM_B_BM_prime(void)
{  double q,G;
  /*------------ find A_prime and AM_prime ------------*/
  for(i=0;i<N;i++)
    for(j=0;j<N;j++)
     {    x[i][j] = 0;
     for(t=0; t<T[v]-1;t++)
      {      x[i][j] += alphas[t*N+i] * a[i][j] *
b[j][O[v*max+(t+1)]] * betas[(t+1)*N+j];      }
     Aprime[i][j] += (double) x[i][j];
     }
  for(i=0;i<N;i++)
     {    z[i] = 0;
     for(t=0;t<T[v]-1;t++)
      {      q = 0;
           for(j=0;j<N;j++)
            q += alphas[t*N+i] * a[i][j] *
b[j][O[v*max+(t+1)]] * betas[(t+1)*N+j];
        z[i] += q;
      }
     AMprime[i] += (double) z[i];
     }

  /*-------------- find B_prime and BM_prime -------------*/
  for(j=0;j<N;j++)
    for(k=0;k<K;k++)
     {    y[j][k] = 0;
     l[j]     = 0;
     for(t=0;t<T[v]-1;t++)
      { G = 0;
         for(i=0;i<N;i++)
         G +=
alphas[t*N+j]*a[j][i]*b[i][O[v*max+(t+1)]]*betas[(t+1)*N+i];
         if(O[v*max+t] == k+1)    y[j][k] += G;
         l[j] += G;
      }
     /* final value */
     if(O[v*max+(T[v]-1)] == k+1)
          y[j][k] += alphas[(T[v]-1)*N+j];
          l[j] += alphas[(T[v]-1)*N+j];
     }
  for(i=0;i<N;i++)
    for(j=0;j<K;j++)  Bprime[i][j] += (double) y[i][j];
  for(i=0;i<N;i++)
    BMprime[i] += (double) l[i];
}
/*********************************************************/
void find_new_abvalue(void)
{  double bcomplex;
 /*---------------- find new_aprime -----------------*/
  for(i=0;i<N;i++)
    for(j=0;j<N;j++)
       aprime[i][j] = (double)( Aprime[i][j]/AMprime[i] );
  /*---------------- find new_bprime -----------------*/
  for(j=0;j<N;j++)
    for(k=0;k<K;k++)
     {    bprime[j][k] = Bprime[j][k]/BMprime[j];
```

```
      /* improve b_prime value */
      if(bprime[j][k] < MIN_B)
        {   bprime[j][k] = MIN_B;      }
      }
   for (i=0;i<N;i++)
    {      bcomplex = 0;
       for (j=0;j<K;j++) bcomplex += bprime[i][j];
    }
   for(i=0;i<N;i++)
    {     imb[i] = 0;
          for(k=0;k<K;k++)
          imb[i] += bprime[i][k];
    }
   for(i=0;i<N;i++)
     for(k=0;k<K;k++)
       bprime[i][k] /= imb[i];
   for (i=0;i<N;i++)
    {       bcomplex = 0;
            for (j=0;j<K;j++) bcomplex += bprime[i][j];
    }
}
/*************************************************************/
void display_a_b_pi_parameter(void)
{  double bcomplex;
   for (i=0;i<N;i++)
    {      for (j=0;j<N;j++)
           printf("A[%d][%d]=%12.10f   ",i,j,aprime[i][j]);
           printf("\n Press any key ...");
           getch();
           printf("\n");
    }
   for (i=0;i<N;i++)
    {      for (j=0;j<K;j++)
           printf("B[%d][%d]=%12.10f   ",i,j,bprime[i][j]);
           printf("\n Press any key ...");
           getch();
           printf("\n");
    }
   printf("Pi = ");
   for (i=0;i<N;i++) printf("%3.2f   ",Pi[i]);
   printf("\n");
   for (i=0;i<N;i++)
    {       bcomplex = 0;
            for (j=0;j<K;j++) bcomplex += bprime[i][j];
            printf("B complex [%d] = %12.10f\n",i+1,bcomplex);
    }
   getch();
}
/*************************************************************/
void save_model_file(void)
{  fwrite(&aprime,4,N*N,model_file);
   fwrite(&bprime,4,N*K,model_file);
   fwrite(&Pi,4,N,model_file);
   for (i=0;i<N;i++)
       for (j=0;j<N;j++)
       fprintf(model_text_file,"A [%3d][%3d] =
                %.40f\n",i,j,aprime[i][j]);
   fprintf(model_text_file,"\n");
   for (i=0;i<N;i++)
```

```
    for (j=0;j<K;j++)
      fprintf(model_text_file,"B [%3d][%3d] =
            %.40f\n",i,j,bprime[i][j]);
  fprintf(model_text_file,"\n");
  for (i=0;i<N;i++)
    fprintf(model_text_file,"Pi[%d]=%.2f   ",i+1,Pi[i]);
  fprintf(model_text_file,"\n");
  fclose(observation_sequence_file);
  fclose(model_file);
  fclose(model_text_file);
  free(O);
  free(T);
  free(alpha);
  free(pres_alpha);
  free(alphas);
  free(beta);
  free(pre_beta);
  free(betas);
  free(sc);
  free(c);
  free(prob_log);
}
/*********************************************************/
void main(void)
{  open_file();
  random_abvalue();
  for(i=0;i<N;i++)
     for(j=0;j<N;j++)
     {  Aprime[i][j] = 0; AMprime[i] = 0; }
  for(i=0;i<N;i++)
     for(k=0;k<K;k++)
     {  Bprime[i][k] = 0; BMprime[i] = 0; }
  for (w=0;w<ROUND;w++)
   {  printf("w = %d\n",w+1);
      copy_abprime_to_ab();
      for (v=0;v<number_of_observation;v++)
       { printf("v = %d\n",v);
       find_alpha_bt_value();
       scaling_alpha_bt();
       find_logP();
       find_A_AM_B_BM_prime();
       printf("prob_log[%d] = %13.10f\n",v,prob_log[v]);
       }
      find_new_abvalue();
      printf("Prob_log[%d] = %20.15f\n",w+1,Prob_log[w]);
      if (w > 0)
     if (Prob_log[w] > Prob_log[w-1])
       printf("P distortion = %22.20f\n",Prob_log[w]-
              Prob_log[w-1]);
     else  printf("P distortion = %22.20f\n",Prob_log[w-1]-
                Prob_log[w]);
     if ( Prob_log[w]-Prob_log[w-1] == 0 )
        printf("P distortion = %13.10f\n",Prob_log[w-1]-
               Prob_log[w]);
     break; }
  getch();
  display_a_b_pi_parameter();
  save_model_file();
}
```

โปรแกรม ASR.C

```
/*******  AUTOMATIC SPEECH RECOGNITION TEST PROGRAM *******/

#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <io.h>
#include <math.h>

#define  PI                    3.141592654
#define  FRAME_LENGTH          300
#define  SHIFT_LENGTH          100
#define  NP     19     /* number of parameter in each frame */
#define  P      12     /* order of LPC */
#define  ADAPTIVE              0.9375
#define  FIRST_CALL            1
#define  NEXT_CALL             2
#define  vector_dimension      19
#define  number_of_codebook 64
#define  N      6      /*  number of state in each model  */
#define  K      64     /*  number of distinct observation */
#define  STORED_MODEL  10  /*  number of stored model    */
FILE    *input_wav_file;
FILE    *parameter_file;
FILE    *input_vector_file; /* input file  */
FILE    *codebook_file;     /* codebook file */
FILE    *codebook_index_file; /* codebook index file */
FILE    *codebook_index_text_file;/*codebook index textfile*/
FILE    *model_file;
FILE    *unknown_word_file;
char    input_wav_file_name[]     ="c:\\recog\\test.wav";
char    parameter_file_name[]     ="para.tmp";
char    input_vector_file_name[] ="para.tmp";
char    codebook_file_name[]="c:\\mspeech\\remodel4\\c1.vq";
char    codebook_index_file_name[]="cidx.tmp";
char    codebook_index_text_file_name[]="cidx.txt";
char    unknown_word_file_name[] = "cidx.tmp";
char    *model_file_name[] =
     {"c:\\mspeech\\remodel4\\m0_1.hmm",
      "c:\\mspeech\\remodel4\\m1_1.hmm",
      "c:\\mspeech\\remodel4\\m2_1.hmm",
      "c:\\mspeech\\remodel4\\m3_1.hmm",
      "c:\\mspeech\\remodel4\\m4_1.hmm",
      "c:\\mspeech\\remodel4\\m5_1.hmm",
      "c:\\mspeech\\remodel4\\m6_1.hmm",
      "c:\\mspeech\\remodel4\\m7_1.hmm",
      "c:\\mspeech\\remodel4\\m8_1.hmm",
      "c:\\mspeech\\remodel4\\m9_1.hmm"};
int     length;
int     number_of_frame;
int     nf, fl, i, j, k, l, c, m, n, t;
int     signal[FRAME_LENGTH+1];
float   preemphasis[FRAME_LENGTH+1],windowed[FRAME_LENGTH+1];
float   R[P+1],A[P][P],X[P][P];
float   alpha[P+1],phi[P+1];
float   cepstrum[NP],weight[NP];
float   gain,q;
```

```
float  *input_vector; /* keep all input vector in memory */
float  *codebook;
char   *min_index;/*minimum of distortion vector of each
                    frame */
float  vector_distance[number_of_codebook]; /* vector and
                                        codebook distance */
float  distance,total_distance;
float  min_distance;
int    nc,vd;
int    file_number;
int    number_of_input_file;
int    number_of_frame,nf;
int    model;
char   T;              /* number of observation sequence*/
char   *O;             /* training sequence */
float  Pi[N] = {1,0,0,0,0,0};
float  a[N][N];
float  b[N][K];
double *dt;
char   *ar;
double d[N];
double prob[STORED_MODEL];
char   *path;
int    max_index;
double dmax;
/*********************************************************/
float Abs(float value)
{   if (value<0) return(-value); else return(value);   }
/*********************************************************/
void display(void)
{  clrscr();
  printf(" /// Automatic Speech Recognition /// ");
}
/*********************************************************/
void open_all_lpc_file(void)
{  if ((input_wav_file = fopen(input_wav_file_name,"rb")) ==
NULL)
   {
      printf("\7 Cannot open input wave '%s' file!\n",
              input_wav_file_name);
      exit(-1);
   }
 if ((parameter_file = fopen(parameter_file_name,"wb")) ==
NULL)
   {  printf("\7 Cannot open output parameter '%s' file!\n",
              parameter_file_name);
          exit(-1);
   }
}
/*********************************************************/
void prepare_data(void)
{   /* find length of data */
  length = (int)filelength(open(input_wav_file_name,0));
  length -= 44L;  /* bypass the header of wave file */
     /* ( 44L  means 44 items in long )  */
  printf(" File '%s' have %d
sampler(s).\n\n",input_wav_file_name,length);
  fseek(input_wav_file,44L,SEEK_CUR); /* set pointer to
first data */
```

```
   /* find number of frame ( is round of loop ) */
  number_of_frame = (length-
FRAME_LENGTH+SHIFT_LENGTH)/SHIFT_LENGTH;
  /* read first data for correct the formula */
  signal[0] = (int)fgetc(input_wav_file);
}
/*********************************************************/
void find_preemphasis_and_window(void)
{  static int called = FIRST_CALL;
  /* Preemphasis and windowed */
  if (called == FIRST_CALL)
   {      for ( fl=1;fl<=FRAME_LENGTH;fl++ )
        {    signal      [fl] = (int)fgetc(input_wav_file);
preemphasis[fl]=(signal[fl]-ADAPTIVE*(float)signal[fl-1]);
       windowed [fl] = preemphasis[fl]*(0.54-
0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1)));
        }
      called = NEXT_CALL;
   }
else/*Second and later called,Use some prevous data again */
   {       for (fl=1;fl<=(FRAME_LENGTH - SHIFT_LENGTH);fl++)
        {
        signal      [fl] = signal[fl+SHIFT_LENGTH];
        preemphasis[fl] = preemphasis[fl+SHIFT_LENGTH];
        windowed    [fl] = preemphasis[fl]*(0.54-
0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1)));
        }
       for (fl=(FRAME_LENGTH -
SHIFT_LENGTH+1);fl<=FRAME_LENGTH;fl++)
        {    signal      [fl] = (int)fgetc(input_wav_file);
     preemphasis[fl] = (signal[fl] - ADAPTIVE *
(float)signal[fl-1]);
        windowed    [fl] = preemphasis[fl]*(0.54-
0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1)));
        }
   }
}
/*********************************************************/
void find_autocorrelation(void)
{  int   m,n,i,j,k,l,maxrow,change=0;
  float alk,akk,maxvalue,temp,deta;
  float matrix_temp[P][P];
  /* Find R[0..P] */
  for (k=0;k<=P;k++)
   {   R[k] = 0;
      for (m=0;m<=FRAME_LENGTH-1-k;m++)
       {    R[k] += windowed[m+1] * windowed[m+k+1];       }
   }
  /* Define MATRIX */
  for (i=0;i<P;i++)
   {  m = 0;
      for (k=i;k<P;k++)
       {    A[i][k] = R[m++];
       A[k][i] = A[i][k];
       }
   }
  /* copy matrix */
  for (i=0;i<P;i++)
    for (j=0;j<P;j++)
```

```
      matrix_temp[i][j] = A[i][j];
 for (i=0;i<P;i++)
   for (j=0;j<P;j++)
     if (i==j) X[i][j] = 1; else X[i][j]=0;
  for (k=0;k<P;k++)
   {  maxrow = k;
      maxvalue = Abs(matrix_temp[k][k]);
      for (i=k+1;i<P;i++)
       {   if (maxvalue < Abs(matrix_temp[i][k]))
       {
          maxvalue = Abs(matrix_temp[i][k]);
          maxrow   = i;
       }
       }
      if (maxrow != k)
       {    change += 1;
       for (j=0;j<P;j++)
        { /* swap */
          temp                    =  matrix_temp[k][j];
          matrix_temp[k][j]       =  matrix_temp[maxrow][j];
          matrix_temp[maxrow][j]  =  temp;
        }
       for (j=0;j<P;j++)
        { /* swap */
          temp                    =  X[k][j];
          X[k][j]                 =  X[maxrow][j];
          X[maxrow][j]            =  temp;
        }
       }
     akk = matrix_temp[k][k];
     for (j=k;j<P;j++)
       matrix_temp[k][j] /= akk;
     for (j=0;j<P;j++)
       X[k][j] /= akk;
     akk = matrix_temp[k][k];
     for (l=0;l<P;l++)
      {    alk = matrix_temp[l][k];
      if (k != l)
       {     for (m=k;m<P;m++)
            matrix_temp[l][m]  -= matrix_temp[k][m]/akk*alk;
            for (m=0;m<P;m++)
            X[l][m] -= X[k][m]/akk*alk;
       }
      }
   }
  /* find phi[1..P] */
  for (i=1;i<=P;i++)
   {      phi[i] = R[i];   }
}
/*************************************************************/
void find_coefficient(void)
{  /* Find Coefficient of LPC */
  for (i=0;i<P;i++)
   {     alpha[i+1] = 0;
         for (k=0;k<P;k++)
       {    alpha[i+1] += X[i][k]*phi[k+1];        }
   }
}
/*************************************************************/
```

```
void find_gain(void)
{  /* Find GAIN */
  q = R[0];
  for (i=1;i<=P;i++)
   {      q -= ( alpha[i] * R[i] );    }
//  printf("\n Frame %d q is %.16f\n\n",j,q);
  gain = (float)sqrt(q);
}
/*************************************************************/
void find_cepstrum(void)
{  /* Find Cepstrum */
  cepstrum[0] = (float)log(gain);
  for (m=1;m<NP;m++)
   {      if (m<=P)  cepstrum[m] = alpha[m];
      else   cepstrum[m] = 0;

       for (k=1;k<=m-1;k++)
     cepstrum[m] += ((float)k/(float)m) * cepstrum[k] *
alpha[m-k];
   }
}
/*************************************************************/
void find_weight(void)
{  /* Find Weight */
    for (m=0;m<NP;m++)
    weight[m] = cepstrum[m] * (1+((NP-
1)/2)*sin(PI*(float)m/(NP-1)));
}
/*************************************************************/
void write_parameter_file(void)
{      for (i=0;i<NP;i++)
     fwrite(&weight[i],sizeof(float),1,parameter_file);
}
/*************************************************************/
void close_all_lpc_file(void)
{  fclose(input_wav_file);
  fclose(parameter_file);
}
/*************************************************************/
void lpc(void)
{  open_all_lpc_file();
  prepare_data();
  for (nf=0;nf<number_of_frame;nf++)
   {   find_preemphasis_and_window();
      find_autocorrelation();
      find_coefficient();
      find_gain();
      find_cepstrum();
      find_weight();
      write_parameter_file();
   }
  close_all_lpc_file();
}
/*************************************************************/
void open_codebook_and_output_file(void)
{  if ((codebook_file = fopen(codebook_file_name,"rb")) ==
NULL)
   {
     printf("\n\7 þ Cannot open codebook file !\n");
```

```
      exit(-1);
   };
  if ((codebook_index_file =
fopen(codebook_index_file_name,"wb")) == NULL)
   {
      printf("\n\7 þ Cannot open codebook index file !\n");
      exit(-1);
   };
  if ((codebook_index_text_file =
fopen(codebook_index_text_file_name,"wt")) == NULL)
   {   printf("\n\7 Cannot open codeidx index text file !");
      exit(-1);
   };
}
/*********************************************************/
void allocate_codebook_memory(void)
{  codebook = (float *)
calloc(number_of_codebook*vector_dimension,sizeof(float));
  if (codebook == NULL)
   { printf("\n\7 Cannot allocate memory for codebook !\n");
      exit(-1);
   };
}
/*********************************************************/
void read_codebook_file(void)
{
fread(codebook,sizeof(float),number_of_codebook*vector_dimen
sion,codebook_file);
}
/*********************************************************/
void
get_number_of_input_file_and_set_codebook_index_file_pointer
(void)
{ int  flag=-1;
  number_of_input_file = 1;
  fseek(codebook_index_file,number_of_input_file,SEEK_SET);
  fwrite(&flag,1,1,codebook_index_file);
}
/*********************************************************/
void open_input_vector_file(void)
{  if ((input_vector_file =
fopen(input_vector_file_name,"rb")) == NULL)
   { printf("\n\7 Cannot open '%s' file !\n",
             input_vector_file_name);
     exit(-1);
   };
}
/*********************************************************/
void find_number_of_frame(void)
{ int file_size;
  file_size =
(int)filelength(open(input_vector_file_name,0));
  file_size -= file_size % (sizeof(float)*vector_dimension);
  number_of_frame = (int)(file_size / (sizeof(float)
                    *vector_dimension));
  printf(" File '%s' have %d frame(s).\n\n",
          input_vector_file_name,number_of_frame);
}
/*********************************************************/
```

```
void allocate_input_vector_and_min_index_memory(void)
{   input_vector = (float *)
calloc(number_of_frame*vector_dimension,sizeof(float));
    if (input_vector == NULL)
 {printf("\n\7 Cannot allocate memory for input vector!\n");
  exit(-1);
 };
  min_index =(char*)calloc(number_of_frame, sizeof(char));
  if (min_index == NULL)
   {  printf("\n\7 þ Cannot allocate memory for data !\n");
      exit(-1);
   };
}
/*********************************************************/
void read_input_vector(void)
{  /* read input vector into memory */
  fread(input_vector,sizeof(float),
        number_of_frame*vector_dimension,input_vector_file);
}
/*********************************************************/
void find_codebook_index(void)
{  /**** Find codebook and input vector distance ****/
  printf(" Find distance ... ");
  for (nf=0;nf<number_of_frame;nf++)
   {    for (nc=0;nc<number_of_codebook;nc++)
       {    total_distance = 0;
       distance       = 0;
       for (vd=0;vd<vector_dimension;vd++)
        {distance=input_vector[nf*vector_dimension + vd]
                 -codebook[nc*vector_dimension + vd];
            distance *= distance;
            total_distance += distance;
        }
       vector_distance[nc] = total_distance;
       }
      /**** Find minimum distance ****/
      min_distance  = vector_distance[0];
      min_index[nf] = 0;
      for (nc=0;nc<number_of_codebook;nc++)
       {    if ( min_distance > vector_distance[nc] )
        {      min_distance  = vector_distance[nc];
             min_index[nf] = (char)nc;
        }
       }
   }  /* end all frame */
  printf("done.\n");
}
/*********************************************************/
void write_codebook_index_file(void)
{  int  index_file_pointer;
  int  codeword_index;
  index_file_pointer = (int)ftell(codebook_index_file);
  fseek(codebook_index_file,file_number,SEEK_SET);
  fwrite(&number_of_frame,1,1,codebook_index_file);
  fseek(codebook_index_file,index_file_pointer,SEEK_SET);
  for (nf=0;nf<number_of_frame;nf++)
     min_index[nf]+=1;/*adjust index value in range 1-64 */
  fwrite(min_index,1,number_of_frame,codebook_index_file);
}
```

```
/*********************************************************/
void free_input_vector_and_min_index_memory(void)
{  free(input_vector);
   free(min_index);
}
/*********************************************************/
void close_input_vector_file(void)
{  fclose(input_vector_file);  }
/*********************************************************/
void write_codebook_index_text_file(void)
{  char item;
  fclose(codebook_index_file);
  if ((codebook_index_file =
fopen(codebook_index_file_name,"rb")) == NULL)
   {    printf("\n\7 Cannot open codebook index file !\n");
        exit(-1);
   };
  while  (!feof(codebook_index_file))
   {      fread(&item,1,1,codebook_index_file);
         fprintf(codebook_index_text_file,"%2d\n",item);
   }
}
/*********************************************************/
void free_codebook_memory(void)
{  free(codebook); }
/*********************************************************/
void close_all_vq_file(void)
{  fclose(codebook_file);
   fclose(codebook_index_file);
   fclose(codebook_index_text_file);
}
/*********************************************************/
void vq_compair(void)
{open_codebook_and_output_file();
 allocate_codebook_memory();
 read_codebook_file();
 get_number_of_input_file_and
 _set_codebook_index_file_pointer();
 open_input_vector_file();
 find_number_of_frame();
 allocate_input_vector_and_min_index_memory();
 read_input_vector();
 find_codebook_index();
 write_codebook_index_file();
 free_input_vector_and_min_index_memory();
 close_input_vector_file();
 write_codebook_index_text_file();
 free_codebook_memory();
 close_all_vq_file();
}
/*********************************************************/
void load_unknown_word_file(void)
{  char temp;
 /* open unknown word file */
  if ((unknown_word_file =
fopen(unknown_word_file_name,"rb")) == NULL)
   {     printf("\n\7 þ Cannot open unknown word file !\n");
         exit(-1);
   };
```

```
  fread(&T,1,1,unknown_word_file);
  fread(&temp,1,1,unknown_word_file);/*for bypass FLAG(-1)*/
  if (temp != -1)
   {      printf("\n\7 þ Invalid input data file !\n");
          exit(-1);
   };
  O = (char *) calloc((int)T,sizeof(char));
  if (O == NULL)
   {   printf("\n\7 þ Cannot allocate memory for data !\n");
       exit(-1);
   };
  /* read observation sequence */
  fread(O,1,(int)T,unknown_word_file);
  printf(" Number of stored model = %d\n",STORED_MODEL);
}
/*********************************************************/
void allocate_memory(void)
{  dt     = (double *) calloc((int)T*N,sizeof(double));
   ar     = (char   *) calloc((int)T*N,sizeof(char));
   path   = (char   *) calloc((int)T  ,sizeof(char));
  if ( (dt||ar||path) == NULL)
   {    printf("\n\7  Cannot allocate memory for data !\n");
        exit(-1);
   };
}
/*********************************************************/
void load_model(int model_name)
{  /* open model file */
  if ((model_file = fopen(model_file_name[model_name],"rb"))
== NULL)
   { printf("\n\7 Cannot open model '%s' file !\n",
                model_file_name[model_name]);
     exit(-1);
   };
  fread(&a,4,N*N,model_file);
  fread(&b,4,N*K,model_file);
  fclose(model_file);
}
/*********************************************************/
void viterbi(void)
{  /*---------------- find initial values ----------------*/
for (i=0;i<N;i++)
   {     dt[0*N+i] = (double)Pi[i] * (double)b[i][O[0]-1];
         ar[0*N+i] = 0;
   }
  /*---------------- find recursion values ---------------*/
  for (t=1;t<(int)T;t++)
   {       for (j=0;j<N;j++)
     {for (i=0;i<N;i++) d[i]=dt[(t-1)*N+i] *(double)a[i][j];
       dmax      = 0;
       max_index = 0;
       for (k=0;k<N;k++)
        {
            if (dmax < d[k])
              {   dmax = d[k];
            max_index = k;
              }
        }
       dt[t*N+j] = dmax * (double)b[j][O[t]-1];
```

```
        ar[t*N+j] = max_index;
      }
    }
   /*--------------- find terminal values -----------------*/
   dmax      = 0;
   max_index = 0;
   for (k=0;k<N;k++)
    {       if (dmax < dt[(int)(T-1)*N+k])
           { dmax = dt[(int)(T-1)*N+k];
             max_index = k;
         }
    }
   prob[model] = dmax;
   path[(int)T-1]    = max_index;
   /*------------------ path backtracking ----------------*/
   for (t=(int)T-2;t>=0;t--)
    {       path[t] = ar[(t+1)*N+(path[t+1])];    }
   /*---------------------- Show Path ---------------------*/
   printf("ProbMAX with MODEL#[%2d]=%e and Path is
...\n",model,prob[model]);
//  for (t=0;t<(int)T;t++)
//    printf("%d",path[t]+1);
//  printf("\n");
//  getch();
}
/*********************************************************/
void display_recognized_word(void)
{  dmax      = 0;
   max_index = 0;
   for (k=0;k<STORED_MODEL;k++)
    {    if (dmax < prob[k])
        {    dmax = prob[k];
        max_index = k;
        }
    }
   printf(" |  Recognized word is \" %d \"  | \n",max_index);
   printf("            Press ENTER to continue . . .");
   while (getch()!=13);
}
/*********************************************************/
void recognize(void)
{  load_unknown_word_file();
   allocate_memory();
   for (model=0;model<STORED_MODEL;model++)
   {   load_model(model);
      viterbi();
   }
  display_recognized_word();
}
/*********************************************************/
void main(void)
{   display();
    lpc();
    vq_compair();
    recognize();
}
```

โปรแกรม VQLN.C

```
/******* Vector Quantization for Speech Recognition *******/
/*****=--------------=* LEARNING PART =*-------------=*****/
/*****  Use :- file seek       : for training set     *****/
/*****          memory         : for codebook         *****/
/*****   and :- array          : for distance         *****/
/*****=-----------------------------------------------=*****/
/***** INPUT :- output file from LPC (training set)   *****/
/*****       ( it have 'number_of_frame' frames,      *****/
/***** each frame have 19 vector dimension parameter, *****/
/*****   each vector is float (4 bytes) )             *****/
/* OUTPUT :- Codebook of ALL Training Set.(codebook.vq )  */
/*****         ( it have 64 codebook,                 *****/
/*** each codebook have 19 vector dimension parameter,  ***/
/*****     each vector is float (4 bytes) )           *****/
/*** - Codebook in text file format.  ( codebook.txt )  ***/
/*** - Distortion of each round of new centroid finding ***/
/***    in text file format.      ( distore.txt )       ***/
/*********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <io.h>
#include <math.h>
#include <time.h>

#define  vector_dimension        19
#define  number_of_codebook      64
#define  REJECT                   0
#define  ACCEPT                   1
#define  REJECT_VALUE         0.001

float training_set     [vector_dimension];  /* for keep all
training set in memory */
float  previous_centroid[number_of_codebook]
       [vector_dimension];
           /* previous centroid of codebook, random first */
float final_centroid[number_of_codebook][vector_dimension];
                    /*  new adjusted centroid of codebook  */
int   *min_index;  /* min distortion vector of each frame */

float vector_distance[number_of_codebook];
                    /*    vector and codebook distance     */
float distance,total_distance;
float min_distance;
int   nc,vd,round_number=1;
int   codebook_member;
int   number_of_frame,nf;

FILE  *training_set_file;  /* input file  */
FILE  *codebook_file;      /* output file */
FILE  *codebook_text_file; /* output codebook text file   */
FILE  *distore_text_file;  /* output distortion text file */
```

```
char  training_set_file_name [50] = "mix.lpc";
char  codebook_file_name     [50] = "codebook.vq";
char  codebook_text_file_name[50] = "codebook.txt";
char  distore_text_file_name [50] = "distore.txt";

/*********************************************************/
void display_vqln_title(void)
{
   clrscr();
   printf("#####   VQLearnfor Speech Recognition   ######");
printf(" Input  : Collection of LPC parameter file .lpc\n");
   printf("  Output : Codebook file codebook.vq,.txt\n   ");
   printf("  Distorsion file distore.txt\n               ");

 printf("ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ");
}

/*********************************************************/
void open_all_file(void)
{
   char temp_file_name[50];

   printf(" Please enter Mix LPC parameter file(mix.lpc):
");
   gets(temp_file_name);
   if (strlen(temp_file_name) > 0)
     strcpy(training_set_file_name,temp_file_name);
   if ((training_set_file =
fopen(training_set_file_name,"rb")) == NULL)
    {
       printf("\n\7 þ Cannot open input '%s' file !\n",
               training_set_file_name);
       fcloseall();
       exit(-1);
    }

   printf(" Please enter Codebook file name(codebook.vq):");
   gets(temp_file_name);
   if (strlen(temp_file_name) > 0)
     strcpy(codebook_file_name,temp_file_name);
        if ((codebook_file = fopen(codebook_file_name,"wb"))
     == NULL)
    {
       printf("\n\7 þ Cannot open output '%s' file!\n",
               codebook_file_name);
       fcloseall();
       exit(-1);
    }

   printf("Please enter Codebook Text file name
          (codebook.txt):");
   gets(temp_file_name);
   if (strlen(temp_file_name) > 0)
     strcpy(codebook_text_file_name,temp_file_name);
   if ((codebook_text_file =
fopen(codebook_text_file_name,"wt")) == NULL)
    {
       printf("\n\7 þ Cannot open output '%s' file !\n",
               codebook_text_file_name);
```

```
        fcloseall();
        exit(-1);
     }

      printf(" Please enter Distortion Text file name
             (distore.txt) : ");
    gets(temp_file_name);
    if (strlen(temp_file_name) > 0)
      strcpy(distore_text_file_name,temp_file_name);
    if ((distore_text_file =
fopen(distore_text_file_name,"wt")) == NULL)
     {
        printf("\n\7 þ Cannot open output '%s' file !\n",
               distore_text_file_name);
        fcloseall();
        exit(-1);
     }
    printf("ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ");
}

/*********************************************************/
void find_number_of_frame(void)
{
    long file_size=0;
    int  file_handle;

    file_handle = open(training_set_file_name,0);
    file_size   = filelength(file_handle);
    file_size  -= file_size %
(sizeof(float)*vector_dimension);
    close(file_handle);

    number_of_frame = (int)(file_size /
(sizeof(float)*vector_dimension));
    printf(" File '%s' have %d
frame(s).\n",training_set_file_name,number_of_frame);
}

/*********************************************************/
void allocate_min_index_memory(void)
{
    /* allocate memory for minimun distance data */
    min_index = (int *) calloc(number_of_frame, sizeof(int));
    if (min_index == NULL)
     {
        printf("\n\7 þ Cannot allocate memory for Min
distance data !\n");
        fcloseall();
        exit(-1);
     }
}

/*********************************************************/
void random_start_centroid_from_training_set(void)
{
    int random_frame_number;

    printf(" Initial start centroid . . . ");
    randomize();
```

```
   for(nc=0;nc<number_of_codebook;nc++)
    {
       random_frame_number = random(number_of_frame);

fseek(training_set_file,sizeof(float)*random_frame_number*
      vector_dimension,SEEK_SET);

fread(previous_centroid[nc],sizeof(float),vector_dimension,
      training_set_file);
    }
   printf("done.\n\n");
}

/*********************************************************/
int find_codebook(void)
{
 /*** Find training_set and previous_centroid distance ***/
   printf(" Find distance ...");
   for (nf=0;nf<number_of_frame;nf++)
    {
       while(kbhit()) if (getch()==27) return(0);

fseek(training_set_file,sizeof(float)*nf*vector_dimension,
      SEEK_SET);
       fread(training_set,sizeof(float),vector_dimension,
             training_set_file);
       for (nc=0;nc<number_of_codebook;nc++)
     {
        total_distance = 0;
        distance       = 0;
        for (vd=0;vd<vector_dimension;vd++)
         {
            distance = training_set[vd] -
previous_centroid[nc][vd];
            distance *= distance;
            total_distance += distance;
         }
        vector_distance[nc] = total_distance;
     }

       /**** Find min distance ****/
       min_distance  = vector_distance[0];
       min_index[nf] = 0;
       for (nc=0;nc<number_of_codebook;nc++)
     {
        if ( min_distance > vector_distance[nc] )
         {
            min_distance  = vector_distance[nc];
            min_index[nf] = nc;
         }
     }
    }   /*end all frame*/

   /**** Find new centroid ****/
   printf(" Find new centroid #%3d ...",round_number);
   fprintf(distore_text_file,"%3d   ",round_number);
   round_number++;
   for (nc=0;nc<number_of_codebook;nc++)
    {
```

```
        while(kbhit()) if (getch()==27) return(0);
        codebook_member = 0;
        for(vd=0;vd<vector_dimension;vd++)
       final_centroid[nc][vd] = 0;
        for (nf=0;nf<number_of_frame;nf++)
      {
         if ( nc == min_index[nf] )
          {
             codebook_member += 1;

fseek(training_set_file,sizeof(float)*nf*vector_dimension,
      SEEK_SET);

fread(training_set,sizeof(float),vector_dimension,
      training_set_file);
            for(vd=0;vd<vector_dimension;vd++)
           final_centroid[nc][vd] += training_set[vd];
          }
      }

        if (codebook_member != 0 )
      {
         for(vd=0;vd<vector_dimension;vd++)
           final_centroid[nc][vd] /= codebook_member;
      }
        else
      {
         for(vd=0;vd<vector_dimension;vd++)
           final_centroid[nc][vd] =
previous_centroid[nc][vd];
      }
     }
   return(1);
}

/*********************************************************/
int check_distance(void)
{
   /**** Check error < reject_value ****/
   for (nc=0;nc<number_of_codebook;nc++)
    {
       while(kbhit()) if (getch()==27) return(0);
       total_distance = 0;
       distance       = 0;
       for (vd=0;vd<vector_dimension;vd++)
      {
         distance = (final_centroid[nc][vd] -
previous_centroid[nc][vd]);
         distance *= distance;
         total_distance += distance;
      }
     }
   printf(" Total distortion = %13.10f\n",total_distance);
   fprintf(distore_text_file,"%13.10f\n",total_distance);

   if (total_distance > REJECT_VALUE) return(REJECT);
   else return(ACCEPT);
}
```

```
/*********************************************************/
void copy_final_centroid_to_previous_centroid(void)
{

memcpy(previous_centroid,final_centroid,sizeof(float)*number
 _of_codebook*vector_dimension);
}

/*********************************************************/
void write_codebook_file(void)
{
   printf("\n\7 þ Finish !\n");
   fwrite(final_centroid,sizeof(float),number_of_codebook*
          vector_dimension,codebook_file);

   for (nc=0;nc<number_of_codebook;nc++)
    {
       for (vd=0;vd<vector_dimension;vd++)
     {
        fprintf(codebook_text_file,"%13.10f\n",
                final_centroid[nc][vd]);
     }
       fprintf(codebook_text_file,"\n");
    }
}

/*********************************************************/
void close_all_file(void)
{
   fcloseall();
}

/*********************************************************/
void main(void)
{
   display_vqln_title();
   open_all_file();
   find_number_of_frame();
   allocate_min_index_memory();
   random_start_centroid_from_training_set();

   for(;;)
    {
       if (!find_codebook()) break;
       if ( check_distance() == REJECT )
          /* if REJECT --> find new centroid */
      copy_final_centroid_to_previous_centroid();
       else break;
       while (kbhit()) if (getch()==27) break;
    }

   write_codebook_file();
   close_all_file();
}
/*********************************************************/
```

โปรแกรม ASRD_APP.C

```
/*************************************************************/
/*******DIALING TELEPHONE NUMBER BY SPEECHRECOGNITION******/
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <io.h>
#include <math.h>
#include <time.h>
#include <string.h>

#define  PI                    3.141592654
#define  FRAME_LENGTH          300
#define  SHIFT_LENGTH          100
#define  NP   19          /*number of parameter in each frame */
#define  P                     12              /* order of LPC */
#define  ADAPTIVE              0.9375
#define  FIRST_CALL            1
#define  NEXT_CALL             2
#define  vector_dimension      19
#define  number_of_codebook 64
#define  N    6       /*   number of state in each model   */
#define  K    64  /* number of distinct observation        */
#define  STORED_MODEL   10        /*number of stored model */

/* Register Position in UART */
#define REG_RX       port_address
#define REG_TX       port_address
#define REG_LCONT    port_address + 3
#define REG_MCONT    port_address + 4
#define REG_LSTAT    port_address + 5
#define REG_MSTAT    port_address + 6

/* Line Status Register */
#define LSTAT_DATA_READY          1
#define LSTAT_THRE               32

/* Line Control Register */
#define LCONT_STOP                4
#define LCONT_PARITY_ENABLE       8
#define LCONT_PARITY_SELECT      16
#define LCONT_DLAB              128

/* Modem Control Register */
#define MCONT_DTR                 1
#define MCONT_RTS                 2

/* Parity */
#define PARITY_NONE               0
#define PARITY_ODD                1
#define PARITY_EVEN               2

/* Communication Port Address */
#define COM1          0x3F8
#define COM2          0x2F8
```

```
#define COM3            0x3E8
#define COM4            0x2E8

#define TRUE            1
#define FALSE           0
#define ESC             27
#define ENTER           13

int  port_address;
char *reset_string   = "ATZ";
char *dial_string    = "ATDT";
char *hang_up_string = "ATH";

FILE   *input_wav_file;
FILE   *parameter_file;
FILE   *input_vector_file;          /* input file  */
FILE   *codebook_file;              /* codebook file */
FILE   *codebook_index_file;        /* output file */
FILE   *codebook_index_text_file;   /*output text file */
FILE   *model_file;
FILE   *unknown_word_file;

char   input_wav_file_name[50]    = "c:\\recog\\test.wav";
char   parameter_file_name[]     ="para.tmp";
char   input_vector_file_name[] ="para.tmp";
char   codebook_file_name[]="c:\\mspeech\\amodel5\\c1.vq";
char   codebook_index_file_name[]="cidx.tmp";
char   codebook_index_text_file_name[]="cidx.txt";
char   unknown_word_file_name[] = "cidx.tmp";
char   *model_file_name[] =
{"c:\\mspeech\\amodel5\\mdaw_1.hmm ",
 "c:\\mspeech\\amodel5\\mjul_1.hmm ",
 "c:\\mspeech\\amodel5\\mkar_1.hmm ",
"c:\\mspeech\\amodel5\\mkig_1.hmm ",
 "c:\\mspeech\\amodel5\\mmal_1.hmm ",
 "c:\\mspeech\\amodel5\\mnaw_1.hmm ",
 "c:\\mspeech\\amodel5\\mpoo_1.hmm ",
 "c:\\mspeech\\amodel5\\msir_1.hmm ",
 "c:\\mspeech\\amodel5\\msom_1.hmm ",
 "c:\\mspeech\\amodel5\\mtaw_1.hmm "};

int    length;
int    number_of_frame;
int    nf, fl, i, j, k, l, c, m, n, t;
int    signal[FRAME_LENGTH+1];
float  preemphasis[FRAME_LENGTH+1],windowed[FRAME_LENGTH+1];
float  R[P+1],A[P][P],X[P][P];
float  alpha[P+1],phi[P+1];
float  cepstrum[NP],weight[NP];
float  gain,q;
float  *input_vector; /* keep all input vector in memory */
float  *codebook;
char   *min_index;  /*min distortion vector of each frame */

float  vector_distance[number_of_codebook];
float  distance,total_distance;
float  min_distance;
int    nc,vd;
int    file_number;
```

```
int     number_of_input_file;
int     number_of_frame,nf;

int     model;
char    T;              /* number of observation */
char    *O;             /* training sequence */
float   Pi[N] = {1,0,0,0,0,0};
float   a[N][N];
float   b[N][K];
double *dt;
char    *ar;
double d[N];
double p_st[STORED_MODEL];
char    *q_st;
int     max_index;
int     recog_index;
double dmax;

char *phone_number[] =
                { "11",
                  "12",
                  "13",
                  "14",
                  "15",
                  "16",
                  "17",
                  "18",
                  "16",
                  "554"};
/*******************************************************/
void select_comm_port(int port)
{
  port_address = port;

  (void)inportb(REG_LSTAT); /* clear line status  */
  (void)inportb(REG_MSTAT); /* clear modem status */
  (void)inportb(REG_RX);    /* clear character at input-
output address */
}
/*******************************************************/
void set_baud_rate(int baudrate)
{
  unsigned char divisor_lsb,divisor_msb;

  switch (baudrate)
   { case  9600 : divisor_lsb = 0x0C; divisor_msb = 0x00;
break;
     case 19200 : divisor_lsb = 0x06; divisor_msb = 0x00;
break;
     case 38400 : divisor_lsb = 0x03; divisor_msb = 0x00;
};
  outportb(REG_LCONT     , LCONT_DLAB);
  outportb(port_address  ,divisor_lsb);
  outportb(port_address+1,divisor_msb);
  outportb(REG_LCONT     ,          0);
}

/*******************************************************/
```

```
void set_data_stop_parity_bit(int databits,int stop,
                              int parity)
{
  unsigned char parmbyte;

  parmbyte = databits - 5;
  if (stop==2)                  parmbyte |= LCONT_STOP;
  if (parity != PARITY_NONE)    parmbyte |=
LCONT_PARITY_ENABLE;
  if (parity == PARITY_EVEN)    parmbyte |=
LCONT_PARITY_SELECT;
  outportb(REG_LCONT,parmbyte);
}
/*********************************************************/
void open_hand_checking_signal(void)
{
   outportb(REG_MCONT, MCONT_DTR|MCONT_RTS);
}
/*********************************************************/
void close_hand_checking_signal(void)
{
   outportb(REG_MCONT, 0);
}
/*********************************************************/
int send_char(int wait_time,char chr)
{
  time_t start_time,now_time;

  time(&start_time);
  /* check port ready for sent */
  while (!(inportb(REG_LSTAT) & LSTAT_THRE))
   {
      time(&now_time);
      if ((int)difftime(now_time,start_time) >= wait_time)
return(FALSE);
   }
  outportb(REG_TX,chr);  /* send a character to port */
  return(TRUE);
}
/*********************************************************/
int receive_char(int wait_time,char *chr)
{
  time_t start_time,now_time;
  char received_char;

  time(&start_time);
  /* check port for data ready */
  while (!(inportb(REG_LSTAT) & LSTAT_DATA_READY))
   {
      time(&now_time);
      if ((int)difftime(now_time,start_time) >= wait_time)
return(FALSE);
   }
  received_char = inportb(REG_RX); /* get a character from
port */
  memcpy(chr,&received_char,1);
  return(TRUE);
}
```

```
/*********************************************************/
int send_message(int wait_time,char *message)
{
  int  length;
  int  count;
  char compair_char;

  length = strlen(message); /* find length of message */
  for (count=0;count<length;count++)
   {
      if (send_char(wait_time,message[count])==FALSE)
return(FALSE);
      if (receive_char(wait_time,&compair_char)==FALSE)
return(FALSE);
      if (compair_char != message[count]) return(FALSE);
   }
  /* send ENTER to Modem for execute the message command */
  if (send_char(wait_time,ENTER)==FALSE)
return(FALSE);
  return(TRUE);
}
/*********************************************************/
void clear_receive_port(int wait_time)
{
  char dummy;

  while (receive_char(wait_time,&dummy));
}

/*********************************************************/
int reset_modem(void)
{
  /* reset modem */
  if (send_message(2,reset_string)==FALSE) return(FALSE);
  return(TRUE);
}
/*********************************************************/
void dial(int com_port, char *phone_number)
{
   char chr;
   char string_command[80];
   select_comm_port(com_port);
   set_baud_rate(9600);
   set_data_stop_parity_bit(8,1,PARITY_NONE);
   open_hand_checking_signal();

   printf("\nþ Resetting Modem . . . ");
   if (reset_modem() == FALSE)
    {  printf("\7þ Modem not found or not response !\n");
exit(-1);  }

   clear_receive_port(4);
   printf("OK.\n");
   /* dial phone number "AT?T???????" */
   strcpy(string_command,dial_string);
   strcat(string_command,phone_number);
   printf("þ Dialing %s . . .\n",string_command);
   printf("þ Pick-Up the receiver and Press ESC to TALK
         when connected . . .\n");
```

```
    if (send_message(2,string_command) == FALSE)
     {  printf("\7þ Modem not response !\n");  exit(-1);  }

    /* Check Input-Output character */
    for(;;)
     {
        /* if key pressed -> send that character to modem  */
        /* if ESC key     -> hang-up modem                 */
        if (kbhit())
      {
         chr=getch();
         while(kbhit()) getch();
         if (chr==ESC)
          {
             printf(" þ Hanging-Up Modem . . . ");
             send_message(2,hang_up_string);
             close_hand_checking_signal();
             printf("OK.\n");
             return;
          }
         else send_char(1,chr);
      }
        /* get response from modem and print it on screen*/
        if (receive_char(0,&chr)==TRUE) putchar(chr);
     }
}
/*********************************************************/
float Abs(float value)
{
   if (value<0) return(-value); else return(value);
}
/*********************************************************/
void display(void)
{
  clrscr();
  printf("/// Automatic Speech Recognition + Dialer /// ");
}
/*********************************************************/
void open_all_lpc_file(void)
{
//  printf("\n Please enter input wave filename
            (and directory ) : ");
//  gets(input_wav_file_name);
  if ((input_wav_file = fopen(input_wav_file_name,"rb")) ==
NULL)
    {
       printf("\7 Cannot open 'input_wav_file'!\n");
       exit(-1);
    }

if ((parameter_file = fopen(parameter_file_name,"wb")) ==
NULL)
    {
      printf("\7 Cannot open output parameter '%s'
      file!\n",parameter_file_name);
       exit(-1);
    }
}
/*********************************************************/
```

```
void prepare_data(void)
{
  /* find length of data */
  length = (int)filelength(open(input_wav_file_name,0));
  length -= 44L;  /* bypass the header of wave file */
     /* ( 44L  means 44 items in long )  */
  printf(" File '%s' have %d
sampling.\n\n",input_wav_file_name,length);
  fseek(input_wav_file,44L,SEEK_CUR); /* set pointer to
first data */

  /* find number of frame ( is round of loop ) */
  number_of_frame = (length-
FRAME_LENGTH+SHIFT_LENGTH)/SHIFT_LENGTH;

  /* read first data for correct the formula */
  signal[0] = (int)fgetc(input_wav_file);
}
/*********************************************************/
void find_preemphasis_and_window(void)
{
  static int called = FIRST_CALL;
  /* Preemphasis and windowed */
  if (called == FIRST_CALL)
    {
      for ( fl=1;fl<=FRAME_LENGTH;fl++ )
       {
       signal     [fl] = (int)fgetc(input_wav_file);
       preemphasis[fl] = (signal[fl]-ADAPTIVE *(float)
                          signal[fl-1]);
       windowed   [fl] = preemphasis[fl]*(0.54-
       0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1)));
       }
      called = NEXT_CALL;
    }
else/*Second and later called,Use some prevous data again */
    {
      for (fl=1;fl<=(FRAME_LENGTH - SHIFT_LENGTH);fl++)
       {
       signal     [fl] = signal[fl+SHIFT_LENGTH];
       preemphasis[fl] = preemphasis[fl+SHIFT_LENGTH];
       windowed   [fl] = preemphasis[fl]*(0.54-
0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1)));
       }
      for (fl=(FRAME_LENGTH -
SHIFT_LENGTH+1);fl<=FRAME_LENGTH;fl++)
       {
       signal     [fl] = (int)fgetc(input_wav_file);
       preemphasis[fl] = (signal[fl] - ADAPTIVE *
(float)signal[fl-1]);
       windowed   [fl] = preemphasis[fl]*(0.54-
0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1)));
       }
    }
}
/*********************************************************/
void find_autocorrelation(void)
{
  int   m,n,i,j,k,l,maxrow,change=0;
```

```
   float alk,akk,maxvalue,temp,deta;
   float matrix_temp[P][P];

   /* Find R[0..P] */
   for (k=0;k<=P;k++)
    {  R[k] = 0;
       for (m=0;m<=FRAME_LENGTH-1-k;m++)
        {
        R[k] += windowed[m+1] * windowed[m+k+1];
        }
    }
   /* Define MATRIX */
   for (i=0;i<P;i++)
    {  m=0;
       for (k=i;k<P;k++)
        {
        A[i][k] = R[m++];
        A[k][i] = A[i][k];
        }
    }
 /* copy matrix */
   for (i=0;i<P;i++)
     for (j=0;j<P;j++)
       matrix_temp[i][j] = A[i][j];

   for (i=0;i<P;i++)
    for (j=0;j<P;j++)
      if (i==j) X[i][j] = 1; else X[i][j]=0;

   for (k=0;k<P;k++)
    {
       maxrow = k;
       maxvalue = Abs(matrix_temp[k][k]);
       for (i=k+1;i<P;i++)
        {
       if (maxvalue < Abs(matrix_temp[i][k]))
        {
           maxvalue = Abs(matrix_temp[i][k]);
           maxrow   = i;
        }
        }
       if (maxrow != k)
        {
        change += 1;
        for (j=0;j<P;j++)
         { /* swap */
            temp                    =  matrix_temp[k][j];
            matrix_temp[k][j]       =  matrix_temp[maxrow][j];
            matrix_temp[maxrow][j]  =  temp;
         }
        for (j=0;j<P;j++)
         { /* swap */
            temp                    =  X[k][j];
            X[k][j]                 =  X[maxrow][j];
            X[maxrow][j]            =  temp;
         }
        }
      akk = matrix_temp[k][k];
```

```
      for (j=k;j<P;j++)
        matrix_temp[k][j] /= akk;
      for (j=0;j<P;j++)
        X[k][j] /= akk;
      akk = matrix_temp[k][k];
      for (l=0;l<P;l++)
       {
       alk = matrix_temp[l][k];
       if (k != l)
        {
         for (m=k;m<P;m++)
          matrix_temp[l][m]-= matrix_temp[k][m]/akk*alk;
         for (m=0;m<P;m++)
          X[l][m] -= X[k][m]/akk*alk;
        }
       }
    }
/* find phi[1..P] */
  for (i=1;i<=P;i++)
    {
       phi[i] = R[i];
    }
}
/*********************************************************/
void find_coefficient(void)
{
  /* Find Coefficient of LPC */
  for (i=0;i<P;i++)
    {
       alpha[i+1] = 0;
       for (k=0;k<P;k++)
        {
        alpha[i+1] += X[i][k]*phi[k+1];
        }
    }
}
/*********************************************************/
void find_gain(void)
{
  /* Find GAIN */
  q = R[0];
  for (i=1;i<=P;i++)
    {
       q -= ( alpha[i] * R[i] );
    }
//  printf("\n Frame %d q is %.16f\n\n",j,q);
  gain = (float)sqrt(q);
}
/*********************************************************/

void find_cepstrum(void)
{
  /* Find Cepstrum */
  cepstrum[0] = (float)log(gain);
  for (m=1;m<NP;m++)
    {
       if (m<=P)      cepstrum[m] = alpha[m];
       else   cepstrum[m] = 0;
```

```
      for (k=1;k<=m-1;k++)
     cepstrum[m] += ((float)k/(float)m) * cepstrum[k] *
alpha[m-k];
   }
}
/*********************************************************/
void find_weight(void)
{
  /* Find Weight */
  for (m=0;m<NP;m++)
weight[m] = cepstrum[m] * (1+((NP-1)/2)
            *sin(PI*(float)m/(NP-1)));
//   fwrite(&cepstrum[i],sizeof(cepstrum[i]),1,outputfile3);
//  for (i=0;i<=NP+1;i++)
//   {
//     printf("cepstrum[%d] = %.16f  W[%d] = %.16f\n",
           i,cepstrum[i],i,weight[i]);
//   }
}
/*********************************************************/
void write_parameter_file(void)
{
   for (i=0;i<NP;i++)
     fwrite(&weight[i],sizeof(float),1,parameter_file);
}
/*********************************************************/
void close_all_lpc_file(void)
{
  fclose(input_wav_file);
  fclose(parameter_file);
}
/*********************************************************/
void lpc(void)
{
  open_all_lpc_file();
  prepare_data();
  for (nf=0;nf<number_of_frame;nf++)
   {
      find_preemphasis_and_window();
      find_autocorrelation();
      find_coefficient();
      find_gain();
      find_cepstrum();
      find_weight();
      write_parameter_file();
   }
  close_all_lpc_file();
}
/*********************************************************/
void open_codebook_and_output_file(void)
{
  if ((codebook_file = fopen(codebook_file_name,"rb")) ==
NULL)
   {
      printf("\n\7 þ Cannot open 'case1.vq' file !\n");
      exit(-1);
   };
```

```
  if ((codebook_index_file =
fopen(codebook_index_file_name,"wb")) == NULL)
    {
       printf("\n\7 þ Cannot open 'codebook.idx' file !\n");
       exit(-1);
    };

  if ((codebook_index_text_file =
fopen(codebook_index_text_file_name,"wt")) == NULL)
    {
       printf("\n\7 þ Cannot open 'codeidx.txt' file !");
       exit(-1);
    };
}
/*********************************************************/
void allocate_codebook_memory(void)
{
  codebook = (float *)
calloc(number_of_codebook*vector_dimension,sizeof(float));
  if (codebook == NULL)
    {
     printf("\n\7 Cannot allocate memory for codebook !\n");
       exit(-1);
    };
}
/*********************************************************/
void read_codebook_file(void)
{

fread(codebook,sizeof(float),number_of_codebook*vector_dimen
sion,
     codebook_file);
}
/*********************************************************/
void get_number_of_input_file_and_set_
     codebook_index_file_pointer(void)
{
  int  flag=-1;

  number_of_input_file = 1;
  fseek(codebook_index_file,number_of_input_file,SEEK_SET);
  fwrite(&flag,1,1,codebook_index_file);
}
/*********************************************************/
void open_input_vector_file(void)
{
  if ((input_vector_file =
fopen(input_vector_file_name,"rb")) == NULL)
    {
       printf("\n\7 Cannot open '%s' file !\n",
              input_vector_file_name);
       exit(-1);
    };
}
/*********************************************************/
void find_number_of_frame(void)
{
  int file_size;
```

```
  file_size =
(int)filelength(open(input_vector_file_name,0));
  file_size -= file_size % (sizeof(float)*vector_dimension);
  number_of_frame = (int)(file_size /
(sizeof(float)*vector_dimension));
  printf(" File '%s' have %d frame(s).\n\n",
          input_vector_file_name,number_of_frame);
}
/*********************************************************/
void allocate_input_vector_and_min_index_memory(void)
{
  input_vector =
    (float *) calloc(number_of_frame*vector_dimension,
                     sizeof(float));
  if (input_vector == NULL)
   {
      printf("\n\7 þ Cannot allocate memory for input vector
!\n");
      exit(-1);
   };

  min_index = (char *) calloc(number_of_frame,
sizeof(char));
  if (min_index == NULL)
   {
      printf("\n\7 þ Cannot allocate memory for data !\n");
      exit(-1);
   };
}
/*********************************************************/
void read_input_vector(void)
{
  /* read input vector into memory */

fread(input_vector,sizeof(float),number_of_frame*vector_dime
nsion,
     input_vector_file);
}
/*********************************************************/

void find_codebook_index(void)
{
  /**** Find codebook and input vector distance ****/
  printf(" Find distance ... ");
  for (nf=0;nf<number_of_frame;nf++)
   {
      for (nc=0;nc<number_of_codebook;nc++)
       {
       total_distance = 0;
       distance       = 0;
       for (vd=0;vd<vector_dimension;vd++)
        {
           distance = input_vector[nf*vector_dimension + vd]
-codebook[nc*vector_dimension + vd];
           distance *= distance;
           total_distance += distance;
        }
       vector_distance[nc] = total_distance;
       }
```

```
        /**** Find min distance ****/
        min_distance  = vector_distance[0];
        min_index[nf] = 0;
        for (nc=0;nc<number_of_codebook;nc++)
         {
         if ( min_distance > vector_distance[nc] )
          {
             min_distance  = vector_distance[nc];
             min_index[nf] = (char)nc;
          }
         }
     }  /* end all frame */
   printf("done.\n");
}
/*********************************************************/
void write_codebook_index_file(void)
{
   int  index_file_pointer;
   int  codeword_index;

   index_file_pointer = (int)ftell(codebook_index_file);

   fseek(codebook_index_file,file_number,SEEK_SET);
   fwrite(&number_of_frame,1,1,codebook_index_file);

   fseek(codebook_index_file,index_file_pointer,SEEK_SET);
   for (nf=0;nf<number_of_frame;nf++)
      min_index[nf] += 1;          /* adjust index value in
range 1 - 64 */
   fwrite(min_index,1,number_of_frame,codebook_index_file);
}
/*********************************************************/
void free_input_vector_and_min_index_memory(void)
{
   free(input_vector);
   free(min_index);
}
/*********************************************************/
void close_input_vector_file(void)
{
   fclose(input_vector_file);
}
/*********************************************************/
void write_codebook_index_text_file(void)
{
   char item;

   fclose(codebook_index_file);
   if ((codebook_index_file =
fopen(codebook_index_file_name,"rb")) == NULL)
     {
        printf("\n\7 þ Cannot open 'codebook.idx' file !\n");
        exit(-1);
     };

   while  (!feof(codebook_index_file))
     {
        fread(&item,1,1,codebook_index_file);
        fprintf(codebook_index_text_file,"%2d\n",item);
```

```
    }
}
/*************************************************************/
void free_codebook_memory(void)
{
  free(codebook);
}
/*************************************************************/
void close_all_vq_file(void)
{
  fclose(codebook_file);
  fclose(codebook_index_file);
  fclose(codebook_index_text_file);
}
/*************************************************************/
void vq_compair(void)
{
  open_codebook_and_output_file();
  allocate_codebook_memory();
  read_codebook_file();
  get_number_of_input_file_and
  _set_codebook_index_file_pointer();
  open_input_vector_file();
  find_number_of_frame();
  allocate_input_vector_and_min_index_memory();
  read_input_vector();
  find_codebook_index();
  write_codebook_index_file();
  free_input_vector_and_min_index_memory();
  close_input_vector_file();

  write_codebook_index_text_file();
  free_codebook_memory();
  close_all_vq_file();
}
/*************************************************************/
void load_unknown_word_file(void)
{
  char temp;
  /* open unknown word file */
  if ((unknown_word_file =
fopen(unknown_word_file_name,"rb")) == NULL)
   {
      printf("\n\7 þ Cannot open input file !\n");
      exit(-1);
   };
  fread(&T,1,1,unknown_word_file);
  fread(&temp,1,1,unknown_word_file);/* bypass FLAG(-1) */
  if (temp != -1)
   {
      printf("\n\7 þ Invalid input data file !\n");
      exit(-1);
   };

  O = (char *) calloc((int)T,sizeof(char));
  if (O == NULL)
   {  printf("\n\7 þ Cannot allocate memory for data !\n");
      exit(-1);
   };
```

```
  /* read observation data */
  fread(O,1,(int)T,unknown_word_file);

  printf(" Number of stored model = %d\n",STORED_MODEL);
}
/*********************************************************/
void allocate_memory(void)
{
  dt   = (double *) calloc((int)T*N,sizeof(double));
  ar   = (char   *) calloc((int)T*N,sizeof(char));
  q_st = (char   *) calloc((int)T  ,sizeof(char));

  if ( (dt||ar||q_st) == NULL)
   {
      printf("\n\7 þ Cannot allocate memory for data !\n");
      exit(-1);
   };
}
/*********************************************************/
void load_model(int model_name)
{
  /* open model file */
  if ((model_file = fopen(model_file_name[model_name],"rb"))
== NULL)
   {
      printf("\n\7 þ Cannot open model '%s' file !\n",
             model_file_name[model_name]);
      exit(-1);
   };
  fread(&a,4,N*N,model_file);
  fread(&b,4,N*K,model_file);
  fclose(model_file);
}
/*********************************************************/
void viterbi(void)
{
  /*---------------- find initial values -----------------*/
for (i=0;i<N;i++)
   {
      dt[0*N+i] = (double)Pi[i] * (double)b[i][O[0]-1];
      ar[0*N+i] = 0;
   }
  /*------------------- find recursion values -------------*/
  for (t=1;t<(int)T;t++)
   {
      for (j=0;j<N;j++)
       {
       for (i=0;i<N;i++) d[i]=dt[(t-1)*N+i]*(double)a[i][j];
       dmax      = 0;
       max_index = 0;
       for (k=0;k<N;k++)
        {  if (dmax < d[k])
            {
            dmax = d[k];
            max_index = k;
            }
        }
      dt[t*N+j] = dmax * (double)b[j][O[t]-1];
```

```
        ar[t*N+j] = max_index;
        }
    }
  /*------------------ find terminal values ---------------*/
  dmax      = 0;
  max_index = 0;
  for (k=0;k<N;k++)
   {
      if (dmax < dt[(int)(T-1)*N+k])
       {
       dmax = dt[(int)(T-1)*N+k];
       max_index = k;
       }
    }
  p_st[model] = dmax;
  q_st[(int)T-1]    = max_index;
  /*------------------ path backtracking -----------------*/
  for (t=(int)T-2;t>=0;t--)
   {
      q_st[t] = ar[(t+1)*N+(q_st[t+1])];
   }
  /*----------------------- Show Path --------------------*/
// printf("ProbMAX with MODEL#[%2d] = %e and Path is
...\n",model,p_st[model]);
//  for (t=0;t<(int)T;t++)
//    printf("%d",q_st[t]);
//  printf("\n");
//  getch();
}
/*********************************************************/
void recog(void)
{
  load_unknown_word_file();
  allocate_memory();

  for (model=0;model<STORED_MODEL;model++)
   {
      load_model(model);
      viterbi();
   }
}
/*********************************************************/
void display_recognized_word(void)
{
  char    chr;
  dmax        = 0;
  recog_index = 0;
  for (k=0;k<STORED_MODEL;k++)
   {  if (dmax < p_st[k])
       {  dmax = p_st[k];
          recog_index = k;
       }
   }
if (recog_index == 0)
     {printf("    |  Recognized word is \" DAW \"    | \n");}
     else if (recog_index==1)
     {
     printf(" |  Recognized word is \" JULACHIT \"    | \n");
     }
```

```
      else if (recog_index==2)
      {
      printf(" |  Recognized word is \" KARUNA \"    | \n");
      }
      else  if (recog_index==3)
      {
      printf("    |  Recognized word is \" KIG \"    | \n");
      }
      else  if (recog_index==4)
      {
      printf("    |  Recognized word is \" MALEE \"    | \n");
      }
      else  if (recog_index==5)
      {
      printf("    |  Recognized word is \" NAWIN \"    | \n");
      }
      else  if (recog_index==6)
      {
      printf("    |  Recognized word is \" POO \"     | \n");
      }
      else  if (recog_index==7)
      {
      printf("    |  Recognized word is \" SIRILUK \"   | \n");
      }
      else  if (recog_index==8)
      {
      printf("    |  Recognized word is \" SOMCHAI \"   | \n");
      }
      else  if (recog_index==9)
      {
      printf("    |  Recognized word is \" TAWAN \"     | \n");
      }
printf("            Press ENTER = Dial   ESC = Cancel . . .");
  for(;;)
   {   if (kbhit())
       {
       chr = getch();
       if (chr == ESC)   exit(1);
       if (chr == ENTER)
        {
           printf("\n");
           dial(COM2,phone_number[recog_index]);
           exit(1);
        }
       }
   }
}
/*********************************************************/
void main(void)
{
   display();
   lpc();
   vq_compair();
   recog();
   display_recognized_word();
}
```

เอกสารอ้างอิง

[1] จิรศักดิ์ เหลืองอุไร, คัมภีร์ใช้งาน การสื่อสารอนุกรมบน PC, กรุงเทพ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, พ.ศ. 2538.

[2] ชวดี อิศรปรีดา และคณะ, การรู้จำเสียงพูด, ปริญญานิพนธ์ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2538.

[3] วัลลภ สุระกำพลธร, การประมวลผลสัญญาณเชิงเลข, กรุงเทพ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2533.

[4] วิทยา เรืองพรวิสุทธิ์, การเขียนโปรแกรมภาษา C สำหรับผู้เริ่มต้น, กรุงเทพ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, พ.ศ. 2537.

[5] สุนทร อรอินทร์ และอัฐ เครือพัก, การประมวลเสียงพูดโดยการประมาณเชิงเส้น, ปริญญานิพนธ์ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2537.

[6] สุธรรม ศรีเกษม และคณะ, MATLAB เพื่อการแก้ปัญหาทางวิศวกรรม,กรุงเทพ : มหาวิทยาลัยรังสิต, พ.ศ. 2538.

[7] เสาวลักษณ์ อารีย์พงศา, การรู้จำเสียงพูดตัวเลขเป็นภาษาไทยแบบไม่ขึ้นกับผู้พูด โดยวิธีฮิดเดนมาร์-คอฟโมเดล และเวคเตอร์ควอนไตซ์เซชัน, วิทยานิพนธ์ : จุฬาลงกรณ์มหาวิทยาลัย, พ.ศ. 2538.

[8] ธันวา ศรีประโมง,การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม, กรุงเทพ : มหาวิทยาลัยเทคโนโลยีมหานคร, พ.ศ. 2537.

[9] D.Fowley and M.Horton, The Student Edition of MATLAB, New Jersey : Prentice - Hall, 1995.

[10] L.R. Rabiner and B. - H. Jaung, Fundamentals of Speech Recognition, New Jersey : Pretice - Hall, 1993.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ สำเร็จลงได้ด้วยความช่วยเหลือจากหลายๆฝ่าย ที่ให้ความช่วยเหลือและให้คำแนะนำในทุกๆอย่าง

ขอขอบพระคุณ อาจารย์อภิชาติ ตั้งทางธรรม จากมหาวิทยาลัยธรรมศาสตร์ ที่ให้คำปรึกษาและข้อเสนอแนะเพื่อเป็นแนวทางในการแก้ปัญหา

ขอขอบคุณ พี่แอ็ค, พี่ซัช, พี่วินิจ, พี่แต้น, พี่ปลา, ตู่ และนันต์ ที่คอยให้ความรู้ และเป็นที่ปรึกษาพร้อมทั้งให้กำลังใจตลอดมา นอกจากนี้ยังต้องขอขอบคุณพี่ๆปริญญาโทห้อง A404 ที่เอื้อเฟื้ออาหารการกินและที่พักพิงในยามกลับหอไม่ได้ พร้อมทั้งโทรศัพท์เมื่อยามไม่มีเหรียญ และให้คำแนะนำที่ดีเสมอมา

ที่ขาดไม่ได้คือ ขอกราบขอบพระคุณ บิดา มารดา ที่คอยให้กำลังใจ และขอขอบคุณคนใกล้ชิดไม่ว่าจะเป็นพี่น้องหรือเพื่อนสนิทที่คอยถามไถ่อยู่ตลอดเวลา

ขอขอบพระคุณเป็นอย่างสูงอีกครั้ง