การรู้จำคำศัพท์ขนาดใหญ่จากเสียงพูดโดยใช้ HTK

Large Vocabulary Speech Recognition Using HTK

นายศราวุธ จันทร์สด 40010763

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

การรู้จำคำศัพท์ขนาดใหญ่จากเสียงพูดโดยใช้ HTK

Large Vocabulary Speech Recognition Using HTK

โดย

นายศราวุธ จันทร์สด 40010763

อาจารย์ที่ปรึกษา

รศ.ดร. บุญธีร์ เครือตราชู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การรู้จำคำศัพท์ขนาดใหญ่จากเสียงพูดโดยใช้ HTK

Large Vocabulary Speech Recognition Using HTK

ผู้จัดทำ

1. นายศราวุธ จันทร์สด 40010763

อาจารย์ที่ปรึกษา

(รศ.ดร. บุญธีร์ เครือตราชู)

การรู้จำคำศัพท์ขนาดใหญ่จากเสียงพูดโดยใช้ HTK

นายศราวุธ จันทร์สด 40010763
รศ.ดร.บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์นี้ได้นำเสนอวิธีการในการออกแบบโมเดล เพื่อทำการรู้จะเสียงพูดที่เป็นประโยคสำหรับภาษาไทย โดยขั้นตอนแรกเสียงพูดจะถูกบันทึกมาเก็บไว้เป็นไฟล์เสียงและมีการสร้างทรานสคริปชันไฟล์ (Transcription files) ซึ่งทั้งสองอย่างนี้จะต้องถูกเตรียมทุกครั้งที่ทำการสร้างโมเดลในการรู้จำ ซึ่งในการสร้างโมเดลนั้นจะเริ่มจากการสร้างโมเดลของหน่วยเสียง (Phoneme) ในภาษาไทย จากนั้นจะนำหน่วยเสียงเหล่านั้นมาทำการปรับปรุงและสร้างโมเดลของเสียงหยุด (Shot pause) และโมเดลของเสียงเงียบ (Silence) ซึ่งเป็นส่วนที่สำคัญ จากนั้นก็จะนำโมเดลที่ได้สร้างขึ้นมาทั้งหมดมาทำการสร้างเป็นคำและเป็นประโยค ซึ่งการสร้างคำจากหน่วยเสียงนั้นจำเป็นต้องใช้ดิกชันนารี (Dictionary) ที่เป็นตัวบ่งบอกว่าคำแต่ละคำที่จะทำการรู้จำนั้นประกอบด้วยหน่วยเสียงอะไรบ้าง

ในส่วนต่อมา จะเป็นการนำเสนอถึงวิธีการปรับปรุงโมเดลซึ่งจะทำการเปลี่ยนจากโมเดลของหน่วยเสียงให้กลายเป็นการเชื่อมกันของ 3 หน่วยเสียงซึ่งจะทำให้การเชื่อมต่อของเสียงนั้นถูกสร้างเป็นโมเดลขึ้นมา จากนั้นจะเป็นการจัดกลุ่มข้อมูล (Clustering) เพื่อทำให้หน่วยเสียงเดียงกันของคำหลายๆคำมีการใช้ข้อมูลร่วมกันเพื่อให้ข้อมูลมีมากขึ้นซึ่งส่งผลให้โมเดลมีความถูกต้องมากขึ้น

ในส่วนสุดท้ายของปริญญานิพนธ์ฉบับนี้เป็นการนำเสนอการสร้างโมเดลไวยกรของภาษา (Grammar Language Model) ซึ่งจะพูดถึงการสร้างโมเดล 3 แบบด้วยกันคือการวนซ้ำของคำ (Word loop) , Extendded Backus-Naur Form (EBNF) , และ โมเดลทางสถิติ Bigram Grammar Language Model ซึ่งจะเป็นส่วนที่จะทำให้การถอดรหัส (Decode) สัญญาณเสียงที่รับเข้ามาให้เป็นคำมีความถูกต้องมากยิ่งขึ้นซึ่งจะทำการเปรียบเทียบให้เห็นความแตกต่างระหว่างไม่มีไวยกรกับที่มีไวยกรไว้ในส่วนท้ายของปริญญานิพนธ์ฉบับนี้

Large Vocabulary Speech Recognition Using HTK

Sarawoot Junsod 40010763

Asst.Dr.Boontee Kruatrachu Advisor

Acadamic year 2000

ABSTRACT

This thesis contains the details on how to design model for Thai language speech recognizing. The First stage is data preparation, which consists of recording voices in wave files format and creating the transcription files. Both of these processes must be prepared every time before producing speech recognizing model. Next stage, the model creation, starts by creating Thai language phoneme model. Then, the phoneme is taken into the fixing silence model in order to develop the stop pause model and silence model. After developing all models in the previous stages, a dictionary must be used to identify particular word and its label.

Also, the thesis proposes how to make the Triphone Model from the Monophone Model, and how to cluster phonemes from various words which share common descriptions. These steps help to increase correction percentage for the models.

Finally, the thesis describes how to create the Grammar Language Model, combining of Word loop, Extendded Backus-Naur Form (EBNF), and Bigram Grammar Language Model. (The BGLM is an essential part in decoding input.) The comparison for the model with and without BGLM is included.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้เนื่องด้วยได้รับความช่วยเหลือจากหลายๆฝ่าย ซึ่งคณะผู้จัดทำใคร่ขอขอบคุณทุกๆท่านที่มีส่วนช่วยสนับสนุน แนะนำชี้แนะแนวทางในทุกๆด้าน

ขอขอบพระคุณ รศ.ดร. บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ได้กรุณาเสียสละเวลาอันมีค่ามาให้คำปรึกษา คอยแนะนำโดยตลอดการทำงาน จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ขอกราบขอบพระคุณบิดา มารดา ผู้ที่คอยเป็นกำลังใจให้คอยให้ความช่วยเหลือด้านต่างๆตลอดมา

ขอขอบคุณพี่ๆทุกคนที่ทำงานด้วยกันในห้อง Information Science Lab ที่คอยแนะนำวิธีการ แนวทางในการทำงานและคอยบอกแหล่งข้อมูลในการค้นคว้าให้ตลอดการทำการวิจัย

ขอขอบคุณเพื่อนๆทุกคนที่ช่วยในการเก็บข้อมูลเสียงพูดที่จำเป็นที่สุดในการวิจัยครั้งนี้

สุดท้ายขอขอบคุณธุรการภาคที่คอยให้ความสะดวกในการติดต่องานในเรื่องต่างๆ

คุณค่าและประโยชน์ที่พึงมีในปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

ศราวุธ จันทร์สด

ผู้ทำการวิจัย

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV-VII
สารบัญภาพ	VIII
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของของงานวิจัย	2
1.3 ขอบเขตของโครงงาน	3
1.4 ขั้นตอนดำเนินงาน	3
บทที่ 2 หลักการในการทำการรู้จำเบื้องต้น	5
2.1 หลักการทั่วไป	5
2.2 Hiiden Markov Models (HMMs)	5
2.3 การรู้จำคำโดยใช้แบบจำลอง HMM	6
2.4 การทำการรู้จำคำแบบต่อเนื่องหรือเป็นประโยค	7
2.5 แนวคำการทำการรู้จำคำศัพท์จำนวนมาก	8
2.6 Triphone Subword	9
2.7 Speech Dependence and Independence	9
บทที่ 3 ระบบเสียงในภาษาไทย	10
3.1 ลักษณะร่วมของเสียง	10
3.1.1 เสียงก้องหรือเสียงไม่ก้อง	10
3.1.2 ความยาวของเสียง	10
3.1.3 ความสูง-ต่ำของเสียง	10
3.1.4 ความดัง	11
3.1.5 การลดน้ำหนักของเสียง	11
3.1.6 ช่วงต่อของเสียง	11
3.2 หน่วยเสียงสำคัญในภาษาไทย	11

สารบัญ (ต่อ)

หน้า

3.3 หน่วยเสียงสระ 12

3.3.1 ลักษณะของเสียงสระ 12

3.3.2 หน้าที่ของหน่วยเสียงสระในภาษาไทย 13

3.4 หน่วยเสียงพยัญชนะ 14

3.4.1 ลักษณะของเสียงพยัญชนะ 14

3.4.2 หน้าที่ของหน่วยเสียงพยัญชนะในภาษาไทย 15

บทที่ 4 การใช้งาน HTK 16

4.1 รู้จักกับ HTK 16

4.2 โครงสร้างของโปรแกรม HTK 16

4.3 เครื่องมือต่างๆ ใน HTK 17

4.3.1 เครื่องมือในการเตรียมข้อมูล 18

4.3.2 เครื่องมือในการ Train โมเดล 18

4.3.3 เครื่องมือในการทดสอบการรู้จำ 19

4.3.4 เครื่องมือในการวิเคราะห์ความถูกต้อง 20

บทที่ 5 การเตรียมข้อมูล 21

5.1 การสร้างดิกชันนารี 21

5.2 ไฟล์เสียง (Wave file) 21

5.3 การสร้างทรานสคริปชันไฟล์ 25

บทที่ 6 ลักษณะทั่วไปของ HMMs 32

6.1 HMM Definition files 32

6.2 พารามิเตอร์ของ HMMs 32

6.3 Basic HMM Definition 33

6.4 Marcro Definition 36

6.5 HMMs Set 38

6.6 HMM Definition Langauge 39

บทที่ 7 การสร้างโมเดล 41

7.1 การสร้างโมเดลของ Monophones 41

7.1.1 Creating Flat start monophones 41

7.1.2 HCOMPV 43

7.1.3 การทำ Re-estimating 44

สารบัญ (ต่อ)

	หน้า
7.1.4 Fixing silence model	45
7.1.5 เครื่องมือ HHEd	47
7.2 การสร้างโมเดลของ Triphones	47
บทที่ 8 การปรับปรุงโมเดล	50
8.1 Discrete and Tied-Mixture Model	50
8.1.2 Making Tied-State Triphones	50
8.1.2 Tied-Mixture System	50
8.2 HMM System Refinement	51
8.2.1 Construct Context-Dependence Model	52
8.2.2 Parameter Tying and Item List	52
8.2.3 Data-Driven Clustering	53
8.2.4 Tree-Based Clustering	55
8.2.5 Mixture Increment	56
8.3 Adapting the HMMs	56
บทที่ 9 การสร้างโมเดลไวยากรณ์ของภาษา	58
9.1 การสร้างไวยากรณ์ของภาษา	58
9.1.1 Word Network	58
9.1.2 การสร้าง Word Network ด้วยเครื่องมือ HPArse	58
9.2 โมเดลทางสถิติ	61
9.2.1 N-gram Model	61
9.2.2 ความสามารถของ N-gram Model	62
9.2.3 วิธีการสร้าง N-gram Model	62
9.2.4 การสร้าง Bigram Language Model โดยใช้เครื่องมือใน HTK	63
9.2.5 การสร้าง Word Network ของ Bigram โดยใช้ Hbuild	64
บทที่ 10 การทดสอบโมเดล	67
10.1 วิธีการทดสอบโมเดล	67
10.2 เครื่องมือในการ Recognize HVite	67
10.3 เครื่องมือในการวิเคราะห์ความถูกต้อง HREsults	69
10.4 การทดสอบโมเดล	71
10.5 สรุป	72

สารบัญ (ต่อ)

	หน้า
บทที่ 11 สรุปและวิจารณ์	73
11.1 สรุป	73
11.2 ข้อสังเกต ปัญหาในการทดลอง และข้อเสนอแนะ	74
ภาคผนวก	75
ภาคผนวก ก ตัวอย่างการตัดไฟล์เสียง	76
ภาคผนวก ข โปรแกรมที่เขียนขึ้นเพื่อช่วยในโครงงาน	79
ภาคผนวก ค Master Label File (MLF)	88
ภาคผนวก ง Backed-off Bi-gram Model และ SLF file	93
ภาคผนวก จ Hmmdefs และ Macro ไฟล์	94
บรรณานุกรม	100

สารบัญภาพ

ภาพที่	หน้า
2.1 การทำการรู้จำ	5
2.2 Viterbi Algorithm	6
2.3 ลักษณะของแบบจำลอง HMM	7
2.4 วิธีการเข้ารหัส (Encoding) และการถอดรหัส (Decoding)	8
4.1 โครงสร้างของโปรแกรม HTK	17
4.2 ประเภทการจัดการของเครื่องมือใน HTK	17
4.3 การ Train เสียงย่อยของโมเดล HMM	19
5.1 Speech Encoding Process	25
5.2 Example Transcription	26
6.1 Simple Lelt-Right HMM	32
6.2 Definition for Sample L-R HMM	33
6.3 Simple Mixture Guassian HMM	34
6.4 HMM ที่มีข้อมูล 2 Stream	35
6.5 ~v macro file	36
6.6 HMM Using ~v macro	36
6.7 Share State & Transition Matrix Macros	37
6.8 Simple Tied State System	38
7.1 ขั้นตอนการทำงานของเครื่องมือ HERest	45
7.2 Silence Model	45
8.1 Tied – Mixture System	50
8.2 การปรับปรุงโมเดลโดยเครื่องมือ HHEd	51
8.3 ผลจากการทำ Data Clustering	54
8.4 การ Tie โดยใช้ decision tree-based	56
8.5 Adapting of HMM by HEAdapt	57
9.1 Network ของระบบรู้จำตัวเลขแบบต่างๆ	59
9.2 Decimal Syntax	65
9.3 Backed-off Bigram Word-loop Network	66
10.1 Flow Chart แสดงขั้นตอนของ Recognizer Evaluation	67

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันเทคโนโลยีทางด้านต่างๆมีความเจริญก้าวหน้าเป็นอย่างมาก มีการผลิตเครื่องไม้เครื่องมือเพื่อให้มนุษย์สะดวกสะบาย จึงทำให้แต่ละวันมนุษย์ผูกพันธ์กับอุปกรณ์เหล่านั้นอย่างใกล้ชิด มีการคุยกันผ่านทางอุปกรณ์ป้อนข้อมูล(Input)ซึ่งปัจจุบันนี้มีหลายประเภทมากไม่ว่าจะเป็นแป้นพิมพ์ (Keyboard) เครื่องอ่านบาร์โค๊ด (Bar code) เครื่องสแกนเนอร์ (Scanner) เป็นต้น เพื่อให้มนุษย์สามารถทำงานได้ง่ายขึ้น รวดเร็ว และถูกต้องแม่นยำ อุปกรณ์อินพุทจึงถูกคิดค้นขึ้นมาหลายรูปแบบเพื่อให้เหมาะกับงานและจุดประสงค์ของผู้ใช้

การรู้จำเสียงพูดก็เป็นอีกวิธีการหนึ่งที่จะช่วยให้ผู้ใช้สามารถใช้เสียงพูดสั่งงานอุปกรณ์ที่จะใช้งานหรือกล่าวอีกนัยหนึ่งก็คือว่าให้อุปกรณ์จำคำพูดของมนุษย์และเมื่อมนุษย์ป้อนคำพูดเข้าไปก็จะรู้ว่าพูดอะไรบ้าง ซึ่งความคิดนี้ไม่ได้เป็นเรื่องใหม่เลยแต่ได้มีการคิดค้นและพัฒนารูปแบบการรู้จำมานานแล้ว มีวิธีการทำการรู้จำหลายๆรูปแบบซึ่งก็ทำให้การพัฒนากลายไปเป็นอุปกรณ์สั่งงานด้วยเสียงได้ดีในระดับหนึ่งอย่างเช่น โทรศัพท์มือถือ เป็นต้น

อุปกรณ์เล็กๆอย่างเช่นโทรศัพท์มือถือนั้นเป็นการนำเอาวิธีการแบบแพทเทิลแม็ทชิ่ง (Pattern matching) มาใช้ซึ่งก็จะทำการเปรียบเทียบระหว่างคำที่พูดกับคำที่ได้บันทึกเอาไว้ก่อนหน้านี้ว่าคล้ายกันหรือไม่ ถ้าใกล้เคียงกันก็จะเลือกมา ซึ่งจะก่อให้เกิดปัญหาตามมาก็คือคำพูดที่ใช้ในการรู้จำมีมากขึ้นตามจำนวนคำที่ต้องการทำการรู้จำ ทรัพยากรของระบบไม่เพียงพอ และทำให้การทำงานเพื่อเลือกเอาคำที่มีลักษณะเหมือนกันจากคำหลายๆคำเป็นไปอย่างล่าช้าเกินเวลาที่มนุษย์จะรับได้ อีกเหตุผลหนึ่งก็คือเมื่อมีการบันทึกเสียงพูดทีละคำเพื่อทำการรู้จำจะไม่สามารถทำให้อุกรณ์เลือกผลลัพธ์ของการพูดเป็นประโยคต่อเนื่องยาว ๆถูกต้องได้ เพราะการพูดต่อเนื่องกันยาวๆระหว่างคำจะมีช่วงต่อของเสียง (Juncture) ซึ่งทำให้คำที่ได้จากการพูดเป็นประโยคกับคำที่ได้จากการพูดทีละคำมีลักษณะที่แตกต่างกัน

การรู้จำเสียงพูดจำนวนมากๆจำเป็นต้องหาวิธีการอย่างอื่น วิธีหนึ่งที่นิยมกันก็คือใช้วิธีการแบบใช้หน่วยเสียง (Phoneme base) เป็นการแบ่งคำแต่ละคำออกเป็นหน่วยเสียงย่อยๆ เช่น คำว่า “การ” จะประกอบไปด้วยเสียง /ก/ /า/ และเสียง /น/ ซึ่งข้อดีของการทำลักษณะเช่นนี้ก็คือหน่วยเสียงจะมีจำนวนคงที่และเมื่อมีคำศัพท์เพิ่มเข้ามาก็เพียงแต่นำเอาหน่วยเสียงที่มีอยู่มาประกอบกันเป็นคำ ซึ่งทำให้สามารถเพิ่มคำศัพท์ให้มากตามที่ต้องการได้โดยใช้ทรัพยากรของระบบไม่มากนัก

การที่ลักษณะการพูดของคนเราที่แตกต่างกันออกไป ทุ้ม-แหลม สั้น-ยาว ทำให้ได้สัญญาณเสียงพูดที่มีลักษณะแตกต่างกันมาก จนทำให้คำเดียวกันออกเสียงได้หลายแบบ วิธีการเก็บเสียงพูดที่จะให้เครื่องทำการรู้จำที่นิยมมากก็คือใช้ Hidden Markov Models (HMMs) ซึ่งมีลักษณะคล้ายกับ Finite State Machine ทั่วไปแตกต่างกันตรงที่ HMMs จะใช้ค่าความน่าจะเป็นในการย้ายจาก state หนึ่งไปยังอีก state

หนึ่ง และมีการวนซ้ำที่ state เดิม ซึ่งจะสร้างเป็นโมเดลของแต่ละหน่วยเสียงจากนั้นเมื่อต้องการเพิ่มคำก็นำโมเดลที่ใช้มาเชื่อมต่อกันเป็นคำ การทำในลักษณะนี้ทำให้เวลาการพูดสั้นกับยาวมีลักษณะที่ไม่แตกต่างกัน เนื่องจากว่าเมื่อพูดคำเดียวกันยาวๆก็จำเกิดการวนซ้ำที่stateเดิมเรื่อยๆจนกว่าเสียงอื่นจะเข้ามาก็จำย้ายไปยัง state อื่น ทำให้การใช้งานในการรู้จำมีความยืดหยุ่นมากขึ้น

เมื่อคำศัพท์มีจำนวนมากขึ้นเป็นเหตุให้ความถูกต้องของการรู้จำลดลง เป็นเพราะว่ามีคำที่ออกเสียงคลายกันอย่างเช่นคำว่า วาง กับ กวาง ดังนั้นการใช้ไวยากรณ์ของภาษาจึงเป็นทางออกที่ดีในการเพิ่มความถูกต้องให้กับการรู้จำ N-gram Model เป็นโมเดลที่จะนำมาใช้สร้างไวยกรให้กับการรู้จำในครั้งนี้ เนื่องจากว่า N-gram Model เป็นการสร้างโมเดลจากประโยคที่ได้ทำการรู้จำแล้ว ซึ่งโมเดลนี้จะเก็บลักษณะของการต่อเชื่อมของคำว่าคำนั้นตามหลังคำว่าอะไรและมีคำอะไรนำหน้า ทำให้การสร้างไวยกรให้กับคำศัพท์จำนวนมาก ๆ เป็นไปได้สะดวกมากยิ่งขึ้น

HTK หรือว่า Hidden Markov Model Toolkits เป็นเครื่องมือที่จะใช้สร้างโมเดลสร้างไวยากรณ์สำหรับการทำการรู้จำซึ่งได้พัฒนาขึ้นมาหลายๆรุ่นสำหรับทำการรู้จำภาษาอังกฤษซึ่งก็สามารถดัดแปลงให้มาใช้กับภาษาไทยได้ แต่ว่าภาษาไทยเป็นภาษาที่แตกต่างออกไป มีความซับซ้อนในรูปแบบของคำ ความซับซ้อนในรูปแบบของไวยากรณ์ ซึ่งปัญหาหลายๆอย่างนี้เป็นผลทำให้การพัฒนาการรู้จำคำศัพท์ภาษาไทยจำนวนมาก ๆ เป็นไปอย่างยากลำบาก

ด้วยเหตุนี้ หวังว่า HTK เวอร์ชันใหม่ในอนาคตจะสามารถนำมาใช้กับภาษาไทยได้ โดยสามารถรู้จำเสียงวรรณยุกต์ในลักษณะที่เป็นประโยคได้ ซึ่งขณะนี้ทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เองก็ได้มีนักศึกษาและอาจารย์ทำการวิจัยและพัฒนาทางด้านนี้ เพื่อให้การทำการรู้จำมีความถูกต้องจนเป็นที่ยอมรับและสามารถนำไปพัฒนาเป็นโปรแกรมที่ใช้ประโยชน์ได้สำหรับคนไทยทุกคน

วัตถุประสงค์ของงานวิจัย

โครงงานฉบับนี้เป็นการทำการรู้จำคำศัพท์ภาษาไทยจากประโยคที่พูดเข้าไป (Continuous Speech) โดยใช้ Hidden Marcov Model Toolkits (HTK) โดยมีจุดประสงค์ดังนี้

1. เพื่อทำการศึกษาทฤษฎีของ Hidden Macov Model รูปแบบโมเดลที่เหมาะสมที่จะใช้งาน รวมทั้งอัลกอริทึมการทำงานของโมเดล
2. เพื่อทำการศึกษาวิธีทำการรู้จำคำศัพท์จำนวนมาก ๆ
3. เพื่อทำการศึกษาไวยกรที่จะนำมาใช้เพิ่มความถูกต้องให้กับการรู้จำ
4. เพื่อทำการศึกษา HTK ซึ่งเป็นเครื่องมือที่จะสามารถช่วยสร้างโมเดลตามทฤษฎี รวมทั้งสร้างไวยกรให้กับการรู้จำ และเพื่อวัดความถูกต้องของโมเดลออกมาเป็นเปอร์เซ็นต์
5. เพื่อทำการศึกษาประเมินความถูกต้องของโมเดลเมื่อใช้เสียงบันทึกของคนๆเดียวกันกับเสียงจากหลาย ๆคนในการทำการรู้จำ
6. เพื่อทำการศึกษาประเมินความถูกต้องของโมเดลเมื่อทำการเพิ่มคำศัพท์เข้าไปในการทำการรู้ว่าความถูกต้องเปลี่ยนแปลงอย่างไร

7. เพื่อทำการประเมินความถูกต้องเมื่อนำไวยากรณ์ของภาษาเข้ามาใช้กับการรู้จำและเปรียบเทียบกับการไม่ใช้ไวยากรณ์ว่าความถูกต้องมีมากขึ้นหรือไม่
8. เพื่อทำการศึกษาปัญหาของการรู้จำเสียงพูดภาษาไทยในลักษณะที่เป็นประโยคต่อเนื่องเพื่อเป็นแนวทางในการพัฒนาต่อในอนาคต

ขอบเขตของโครงงาน

โครงงานนี้เป็นการศึกษาทำการรู้จำคำศัพท์ภาษาไทยที่เป็นประโยคแบบต่อเนื่อง ซึ่งโครงงานนี้มีขอบเขตการทำดังนี้คือ

1. ทำการศึกษาการใช้งาน HTK เพื่อใช้ในการรู้จำโดยสามารถใช้งานในส่วนของการรู้จำแบบเป็นประโยค (Continuous Speech) ว่าต้องใช้โปรแกรมอะไรบ้างในการพัฒนารวมทั้งการปรับปรุงโมเดลให้มีประสิทธิภาพมากขึ้น
2. สร้าง Model ของเสียงย่อยของคำในภาษาไทยโดยการสร้างจากการอัดเสียงจากคนๆเดียวและ หลาย ๆคนเพื่อดูความแตกต่าง
3. นำโมเดลเดิมมาทำการรู้จำซ้ำพร้อมกับการเพิ่มคำศัพท์ใหม่เข้ามาเพื่อวิเคราะห์ค่าความถูกต้องของโมเดล
4. ทำการศึกษาและออกแบบไวยากรณ์ (Grammar Language) เพื่อเลือกไวยากรณ์ที่เหมาะสมมาใช้ในการรู้จำซึ่งจะเพิ่มความถูกต้องให้กับโมเดล
5. ทำการเปรียบเทียบความถูกต้องของการรู้จำเมื่อมีจำนวนเสียงพูดและจำนวนคนเพิ่มขึ้นในการทำการรู้จำนั้น
6. ทำการเปรียบเทียบความถูกต้องเมื่อมีการเพิ่มคำศัพท์มากขึ้น
7. ทำการเปรียบเทียบความถูกต้องเมื่อมีการใช้ไวยกรและไม่ใช้ไวยกรในการทำการรู้จำ
8. สามารถทำการสรุปข้อด้อย ปัญหาของการใช้งานเครื่องมือ HTK สำหรับใช้ในภาษาไทยรวมทั้งการเลือกไวยากรณ์ของภาษามาให้กับการรู้จำ และข้อเสนอแนะกับคนที่จะพัฒนาต่อว่าควรจะไปในแนวทางใดจึงจะเหมาะสม

ขั้นตอนการดำเนินงาน

ในการดำเนินงานทำการรู้จำนั้นจะต้องเป็นไปตามขั้นตอน เพราะขั้นตอนแต่ละขั้นตอนจะต้องรอขั้นตอนก่อนหน้า ซึ่งสามารถเรียงลำดับขั้นตอนต่างๆ ได้ดังนี้

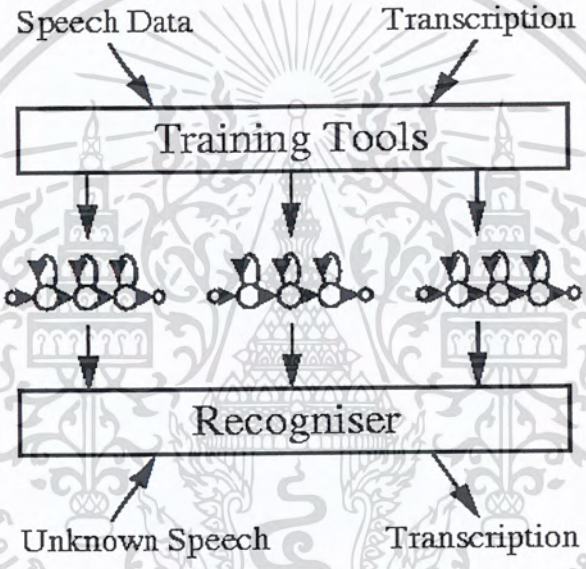
1. ทำการศึกษาขั้นตอนการออกแบบและสร้างโมเดล
2. ทำการอัดเสียงของคำศัพท์ภาษาไทยโดยแบบ Continuous Speech เก็บเป็นแฟ้มข้อมูลจากหลาย ๆ คนพูด
3. เริ่มต้นสร้างเสียงย่อยของคำภาษาไทยทุกหน่วยเสียงจากเสียงที่อัดโดยใช้เครื่องมือ HTK
4. ทำการสร้างคำจากเสียงย่อยเหล่านั้น
5. ทำการ Train โมเดลและเชื่อมโมเดลเป็นคำและเป็นประโยค

6. ทำการหาค่าความถูกต้องของโมเดลทุกครั้งเมื่อทำการ Train เสร็จ
7. ทำการศึกษาไวยากรณ์เลือกไวยากรณ์ที่เหมาะสม และ สร้างไวยากรณ์ให้กับโมเดลเพื่อทำการหาค่าความถูกต้องเปรียบเทียบกับไมเดลที่ไม่ใช้ไวยากรณ์
8. ทำการอัดเสียงคำใหม่เพิ่มทุกครั้งที่มีการเพิ่มคำรวมทั้งปรับค่าพารามิเตอร์ของเครื่องมือให้เหมาะสมกับการ Train ครั้งใหม่
9. ทำการบันทึกข้อผิดพลาดที่เกิดขึ้นทุกครั้งที่ทำการ Train รวมทั้งทำเอกสารรายละเอียดของการ Train ทุกครั้ง

บทที่ 2
หลักการในการทำการรู้จำเบื้องต้น

2.1 หลักการทั่วไป

ในการทำการรู้จำคำศัพท์นั้นมีวิธีการหลายภาพแบบ วิธีหนึ่งที่ได้รับความนิยมมากที่สุดก็คือใช้หลักการของ Hidden Macov Models (HMMs) มาช่วยในการสร้างโมเดลในการรู้จำ ดังที่แสดงในภาพที่ 2.1 โดยจะนำข้อมูลเสียงพูดและไฟล์ Transcription มาสร้างเป็นโมเดล (Training) ในการรู้จำได้ขึ้นอยู่กับในลักษณะต่างๆ แล้วแต่การประยุกต์ HMMs มีความสำคัญมากและควรที่จะทำความเข้าใจก่อนเป็นอันดับแรกว่ามีลักษณะอย่างไรและดีอย่างไร



ภาพที่ 2.1 การทำการรู้จำ

2.2 Hidden Markov Models (HMMs)

HMMs คือแบบจำลองทางสถิติที่ Markov ได้คิดขึ้นซึ่งพัฒนามาเพื่อแบ่งกลุ่มของอนุกรมเวลาหรือสัญญาณที่ไม่คงที่ นั่นคือใช้สำหรับจัดกลุ่มของสัญญาณที่ไม่รู้จัก (Unknown Signal) ให้ไปอยู่ในกลุ่มใดกลุ่มหนึ่งของสัญญาณ นำมาคำนวณหาค่าเฉพาะและสร้างเป็นโมเดลขึ้นมา โดยลักษณะของโมเดลจะมีลักษณะคล้ายกับ Finite State Machine ทั่วไปแต่แตกต่างกันที่ HMMs จะใช้ลักษณะของความน่าจะเป็นในการย้ายข้าม State ซึ่งความน่าจะเป็นนี้สามารถทำการคำนวณได้จากข้อมูลเสียงพูดและ Transcription ที่นำมาทำการสร้างโมเดล

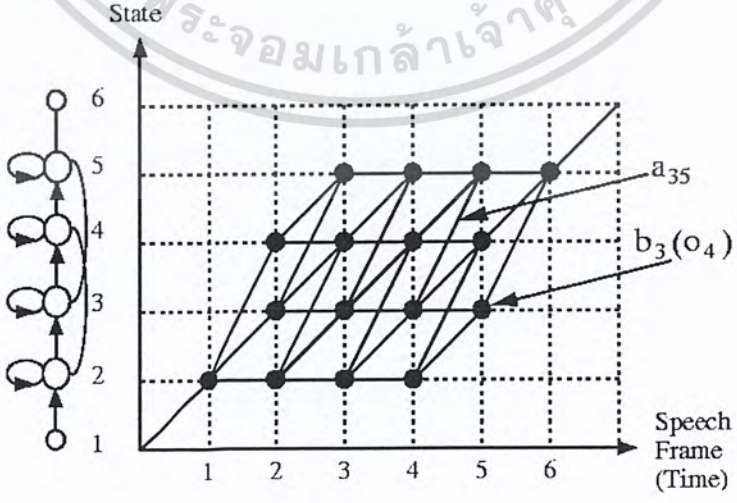
แบบจำลอง HMMs แบ่งออกเป็น 2 ประเภทตามลักษณะการใช้งาน คือ แบบต่อเนื่อง (Continuous) และแบบไม่ต่อเนื่อง (Discrete-time) ซึ่งมีข้อดีต่างกันไปแล้วแต่การประยุกต์ใช้งาน อย่างเช่น

แบบไม่ต่อเนื่องสามารถทำการรู้จำเสียงวรรณยุกต์ได้ แบบต่อเนื่องขณะนี้ยังไม่สามารถทำได้แต่สามารถทำการรู้จะคำศัพท์จำนวนมากได้ซึ่งโครงงานนี้ได้เลือกแบบต่อเนื่องมาทำการรู้จำ

2.3 การรู้จำคำโดยใช้แบบจำลอง HMM

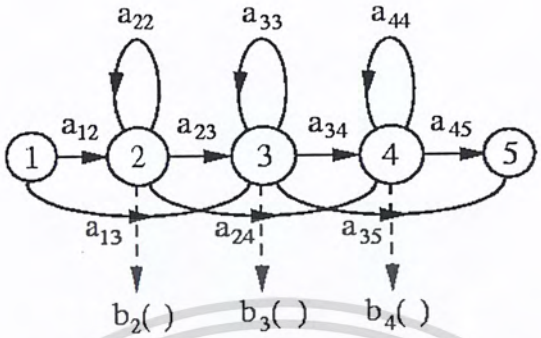
ในการทำการรู้จำโดยใช้แบบจำลอง HMM นั้น หลังจากที่เราได้เลือกและออกแบบโมเดลที่เหมาะสมแล้ว ซึ่งในโครงงานนี้เลือกใช้โมเดลแบบ 5 State Left-Right HMM ซึ่งมีลักษณะดังภาพที่ 2.3 ในการสร้างโมเดลนั้นต้องเริ่มจากการสร้างโมเดลเริ่มต้นซึ่งยังไม่มีข้อมูลของหน่วยเสียงขึ้นมาก่อน (Initialization) จากนั้นโมเดลจะถูก Train จากข้อมูลเสียงพูดที่ได้ทำการบันทึกโดยจะแปลงจากไฟล์ wave มาเป็น Vector ของสัญญาณเสียง จากนั้นข้อมูลทั้งหมดรวมทั้งไฟล์ Transcription จะถูกนำมาคำนวณเป็นโมเดลของแต่ละหน่วยเสียง ซึ่งประกอบไปด้วย State ทั้งหมด 5 state แต่ว่า State ที่ 1 กับ State ที่ 5 ไม่ได้ถูกสร้างขึ้นจากข้อมูลเสียงพูด เพียงแต่ถูกสร้างขึ้นมาไว้เชื่อมกับ State อื่นๆ เพื่อที่จะให้โมเดลของหน่วยเสียงย่อยเหล่านั้นประกอบกับหน่วยเสียงย่อยอื่นๆกลายเป็นคำหรือเป็นประโยคขึ้นมา ใน HTK แบบจำลองหนึ่งๆจะประกอบด้วยหลายๆหน่วยเสียง ซึ่งจะถูกทำการ Re-estimate ซ้ำหลายๆครั้งเพื่อให้โมเดลถูกต้องมากขึ้น ลักษณะของแบบจำลองใน HTK จะสามารถแสดงในภาพของ Text แก้ไขได้โดยใช้ Text Editor ทั้งไปได้ซึ่งจะใช้ภาษาในการอธิบายแบบจำลองทีเรียกว่า HMMs Definitions ซึ่งในโครงงานนี้จะแสดงตัวอย่างให้ดูในภาคผนวก ชื่อไฟล์ hmmdefs และ marcro

ในโมเดลของ HMM นั้นจะลักษณะที่แตกต่างจาก Finite State Machine เพราะว่า HMM จะใช้คำความน่าจะเป็นในการย้ายข้าม State ซึ่งอธิบายตาม Viterbi Algorithm แสดงดังภาพที่ 2.2 ซึ่งจะมี State เริ่มต้นที่ 1 จากนั้นจะเข้าสู่ State 2 และสามารถที่จะทำการย้าย State ไปยัง State 3 หรือ State 4 หรือวนอยู่ที่เดิม (State 2) ก็ได้ขึ้นอยู่กับความน่าจะเป็นในการย้ายข้าม State นั้นๆแกนนอนจะแสดงลำดับของเฟรมของเสียงพูดที่รับเข้ามาส่วนแกนตั้งจะแสดง State ในขณะนั้นซึ่ง State อื่นก็เป็นไปในลักษณะเดียวกัน ลักษณะของการวนซ้ำที่แตกต่างกันในหลายๆเฟรมของเสียงพูดทำให้ลักษณะของคำหรือหน่วยเสียงที่ทำการรู้จำมีลักษณะที่ต่างกัน



ภาพที่ 2.2 Viterbi Algorithm

จากภาพที่ 3.4 แสดงถึงลักษณะทั่วไปของโมเดลที่ใช้ในการวิจัยครั้งนี้ซึ่งค่า a_{ij} นั้นเป็นค่าความน่าจะเป็นในการย้ายจาก State i ไปยัง State j ซึ่งความน่าจะเป็นในการย้ายข้าม State นี้จะถูกเก็บอยู่ในภาพของ Trainsition Matrix ซึ่งผลรวมของความน่าจะเป็นในการย้ายออกจาก State หรือวนที่ State เดิมรวมกันจะได้เท่ากับหนึ่ง



ภาพที่ 2.3 ลักษณะของแบบจำลอง HMM

ค่า $b_j(o_t)$ ซึ่งเป็นค่า Observation probability distribution หรือว่าความน่าจะเป็นในการอยู่ใน State นั้น ณ เวลา t ซึ่งสามารถคำนวณได้จากสมการข้างล่างนี้

$$b_j(o_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} \mathcal{N}(o_{st}; \mu_{jsm}, \Sigma_{jsm}) \right]^{\gamma_s}$$

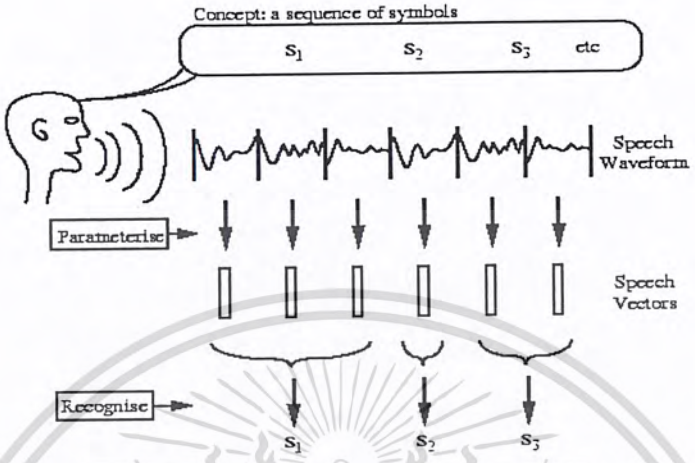
ซึ่งค่า M_{js} จะเป็นจำนวนของ Mixture component ของ State j สำหรับเสียงหนึ่งๆ ซึ่งจะรวมจากหลายๆโมเดลที่มีเสียงเดียวกันเพื่อให้ได้ข้อมูลเพิ่มขึ้น และค่า m เป็นค่าน้ำหนักของแต่ละ Component ค่า $N(o;\mu,\sum)$ เป็นค่าความผันผวน (Multivariate Gaussian) ซึ่งจะเปลี่ยนแปลงตามค่า Vector ของเสียงพูด μ และ ค่า Covariance matrix Σ โดยที่

$$\mathcal{N}(o;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)'\Sigma^{-1}(o-\mu)}$$

2.4 การทำการรู้จำคำแบบต่อเนื่องหรือเป็นประโยค (Continuous Speech Recognition)

ในการทำการรู้จำประโยคนั้น เสียงพูดทั้งประโยคจะถูปแปลงเป็น Vector ลักษณะเฉพาะของเสียงอย่างต่อเนื่องทั้งประโยคซึ่งไม่อาจจะรู้ได้ว่าคำที่อยู่ในประโยคนั้นสิ้นสุดที่ตรงไหนหรือเริ่มต้นในประโยคที่ตรงไหนดังนั้น Shot Pause (sp) และ Silence (sil) คือช่วงที่หยุดพูดซึ่งมีการทิ้งช่วงสั้นยาวต่างกัน หรือเป็นช่วงต่อของเสียงของคำหรือหรือหน่วยเสียง ซึ่งจะเป็นตัวบ่งบอกได้ว่าคำนั้นสิ้นสุดลงหรือยังซึ่งไม่แน่นอนเสมอไป เพราะว่าบางคำในประโยคอาจจะไม่มีการเว้นวรรคระหว่างคำก็เป็นไปได้ แต่ว่าการ

ใส่ Sil กับ Sp ลงไปในช่วงที่มีการหยุดพูดของเสียงนั้นก็เป็นการช่วยให้ความถูกต้องเพิ่มมากขึ้นได้ก็หมายความว่าช่วยให้การตัดคำเป็นไปอย่างถูกต้องด ซึ่งการที่จะรู้ว่าควรจะใส่ Sil และ Sp ตรงไหนของคำหรือประโยคนั้นสามารถดูได้จากไฟล์เสียงพูดเหล่านั้นซึ่งเปิดได้จากโปรแกรมที่จักการเกี่ยวกับเสียงเช่น Cool Edit 2000 เป็นต้น



ภาพที่ 2.4 วิธีการเข้ารหัส (Encoding)และ ถอดรหัส (Decoding) ประโยค

การทำการรู้จำแบบเป็นประโยคนั้นต่างจากการทำการรู้จำแบบเป็นคำเดี่ยวเพราะว่าการทำการรู้จำเป็นประโยคจะมีส่วนของ Juncture ระหว่างคำทำให้การออกเสียงในส่วนท้ายของคำผิดเพี้ยนไปซึ่งต้องทำการ Train เสียงของคำนั้นๆในลักษณะเดียวกันจากการพูดหลายๆครั้งเพื่อให้ช่วงต่อมีความถูกต้องมากยิ่งขึ้น

2.5 แนวคิดการทำการรู้จำคำศัพท์จำนวนมาก (Large vocabulary speech recognition)

แนวคิดในการรู้จำคำศัพท์มากๆนั้นมีข้อสังเกต ตรงที่ว่าทำอย่างไรให้การรู้จำคำศัพท์จำนวนมากๆเป็นไปได้อย่างถูกต้อง สามารถทำการถอดรหัสเสียงออกมาเป็นคำได้อย่างรวดเร็ว และสิ้นเปลืองทรัพยากรน้อยที่สุด ดังนั้นโมเดลหนึ่งโมเดลจะใช้ทำการรู้หน่วยของเสียงในภาษาไทย (Phoneme base Model) เหตุผลเพราะว่า ถ้าทำการรู้จำในลักษณะของหนึ่งคำต่อหนึ่งโมเดลจะทำให้การรู้จำคำศัพท์จำนวนมากต้องใช้ทรัพยากรของระบบจำนวนมาก และการทำงานก็จะช้าตาม ในการทำการรู้จำแบบใช้หน่วยเสียงนั้น เพียงแต่ทำการรู้จำหน่วยเสียงในภาษาไทยซึ่งมีอยู่จำกัด (จะกล่าวถึงในบทต่อไป) และนำโมเดลของแต่ละหน่วยเสียงนั้นมาทำการต่อเชื่อมกันจนเป็นคำและเป็นประโยคได้ซึ่งทำให้ประหยัดกว่า

ถึงแม้ว่าเรามีแบบจำลองที่ดีมีการสร้างโมเดลที่ดีเพียงใด ในการทำการรู้จำก็ยังมีปัญหาอยู่ดี เพราะเมื่อคำศัพท์เพิ่มจำนวนมากขึ้นความคล้ายกันของคำการแบ่งคำที่ไม่ดีพอ ซึ่งทำให้ผลลัพธ์ที่ได้จากการทำการรู้จำออกมาไม่ตรงตามความต้องการ

ดังนั้นในการทำการรู้จำคำศัพท์จำนวนมากๆไวยากรณ์ของภาษา (Grammar Language) ของภาษาจึงมีความสำคัญไม่แพ้กันเพื่อให้ได้โมเดลของไวยากรณ์ที่เหมาะสม และที่สำคัญต้องเข้าใจในหลัก

ไวยากรณ์ของภาษาด้วย โมเดลที่ช่วยทำในส่วนนี้มีหลายประเภทแต่ในโครงงานครั้งนี้ได้เลือกวิธีการ Bigram Language Model ซึ่งเป็นโมเดลที่ใช้กันแพร่หลาย ง่ายและช่วยให้การทำการรู้จำของโครงงานในครั้งนี้มีความถูกต้องเพิ่มขึ้น

2.6 Triphones subword

จากที่ได้กล่าวมาข้างต้นในการทำการรู้จำโดยใช้ลักษณะของหน่วยเสียงย่อยและนำมาประกอบกันเป็นประโยคนั้นยังมีการสร้างโมเดลในลักษณะของ Triphones ซึ่งโมเดลนี้จะทำการรู้จำหน่วยเสียงของคำหรือประโยคที่อยู่ติดกัน 3 หน่วยเสียง วิธีการนี้มีความจำเป็นมากในการช่วยเพิ่มความถูกต้องของโมเดลให้มากขึ้นได้ เพราะว่าในภาษาไทยเราการออกเสียงของหน่วยเสียงย่อยตัวเดียวกันอาจจะแตกต่างกันถ้าหน่วยเสียงตัวสะกดต่างกันที่เรียกว่า (Juncture) และการทำในลักษณะนี้จะสามารถทำการ Cluster ข้อมูลเพื่อทำให้โมเดลมีความถูกต้องมากขึ้นซึ่งจะได้กล่าวในบทต่อๆไป

2.7 Speech Dependence and Independence

เนื่องจากเสียงของแต่ละคนมีความแตกต่างกันไม่ว่าจะเป็นความถี่ซึ่งทำให้เสียงออกมาทุ้มแหลม ลักษณะการพูดสั้น-ยาว จึงทำให้ในการทำการรู้จำนั้นสามารถจดจำเสียงเฉพาะเสียงที่ใช้สร้างแบบจำลองเท่านั้นหรือเสียงที่คล้ายกัน เรียกว่า Speech Dependence ซึ่งข้อดีก็คือสามารถใช้ระบบความปลอดภัยซึ่งจะทำการรู้จำเฉพาะเสียงของบุคคลที่ได้รับอนุญาติเท่านั้น แต่ว่า Speech Dependence นั้นจะไม่สามารถนำมาใช้กับโปรแกรมประยุกต์ที่เป็นส่วนรวมได้ จึงต้องมีการสร้างโมเดลที่จะให้บุคคลใดก็ได้สามารถที่จะพูดและให้ระบบทำการคำนวณผลลัพธ์ออกมาได้อย่างถูกต้องได้ที่เรียกว่า Speech Independence ซึ่งต้องให้คนหลายๆคนหลายๆโทนเสียงความถี่มาทำการอัดเสียงและสร้างเป็นโมเดลขึ้นมา ซึ่งในโครงงานครั้งนี้ตั้งใจจะสร้างโมเดลที่อยู่ในลักษณะ Speech Independence ซึ่งได้สร้างโมเดลจากเสียงของสมาชิกภายในกลุ่ม

บทที่ 3
ระบบเสียงในภาษาไทย

3.1 ลักษณะร่วมของเสียงพูด

เสียงที่ใช้ในภาษาพูดนั้นจะมีลักษณะที่สำคัญบางประการร่วมกัน ซึ่งเรียกได้ว่าเป็นลักษณะร่วมของเสียงพูด ลักษณะที่กล่าวถึงนี้มีอยู่หลายประการ คือ

3.1.1 เสียงก้อง หรือ ไม่ก้องของเสียง

เสียงก้อง หรือ เสียงโฆษะ (Voice)

คือเสียงที่เกิดจากการที่เส้นเสียงเกิดการตึงตัวหรือเรียกว่าเส้นเสียงปิด เมื่อมีแรงดันให้อากาศไหลผ่านกล่องเสียงในขณะที่เส้นเสียงปิดจะเกิดการสั่นสะบัดของเส้นเสียง เป็นผลให้สัญญาณเสียงที่ได้ (Speech waveform) มีลักษณะเป็นคาบ (Quasi-periodic) ซึ่งสามารถเรียกความถี่ในการปิด-เปิดของเส้นเสียงนี้ว่า “ความถี่มูลฐาน” (Fundamental Frequency) ตัวอย่างของเสียงก้องได้แก่ เสียงสระต่างๆ และเสียงพยัญชนะเช่น บ ด ที่เกิดจากการเปล่งเสียงออกทางปาก หรือเสียงพยัญชนะ ม น ง ที่เกิดจากการเปล่งเสียงออกทางจมูก

เสียงไม่ก้อง หรือ เสียงอโฆษะ (Unvoice หรือ Voiceless)

คือเสียงที่เกิดในขณะที่เส้นเสียงคลายจากการตึงหรือเรียกว่าเส้นเสียงเปิด เมื่อมีแรงดันให้อากาศไหลผ่านกล่องเสียงในขณะที่เส้นเสียงเปิด อากาศที่ไหลผ่านอย่างรวดเร็วจะเกิดการไหลวน และปั่นป่วน ทำให้เกิดเสียงที่มีลักษณะเป็นเสียงของสัญญาณรบกวน (Noise) ซึ่งไม่เป็นคาบ ตัวอย่างของเสียงก้องได้แก่ เสียงพยัญชนะ ฟ ซ ส เป็นต้น หรือเกิดจากการสร้างแรงดันอากาศหลังตำแหน่งปิดกั้นของช่องทางเดินเสียง และเมื่อการปิดกั้นนี้ถูกเปิดออก อากาศจะถูกปล่อยออกมาอย่างทันทีทันใดเกิดเป็นเสียงที่เรียกว่าเสียงระเบิด (Plosive Sound) เช่น การเปล่งเสียงเริ่มแรกของพยัญชนะต้นของคำต่างๆ

3.1.2 ความยาวของเสียง (Length)

หมายถึง การที่เสียงใดเสียงหนึ่งเปล่งออกมาได้นานเท่าใด เสียงพูดบางเสียงอาจจะเปล่งออกมาติดต่อกันได้นาน เช่น เสียงสระ เสียงพยัญชนะนาสิก หรือ เสียงพยัญชนะเสียดแทรก

ในภาษาไทย เสียงพูดที่มีความยาวสั้น ก็มีเพียงเสียงสระเท่านั้น เช่น อะ อิ อุ เป็นเสียงสั้น อา อี อู เป็นเสียงยาว เป็นต้น

3.1.3 ระดับเสียงสูง-ต่ำ (Pitch)

เสียงพูดจะมีระดับ สูง หรือ ต่ำ อยู่ที่ความถี่ของเสียง (Fundamental frequency) ถ้าความถี่ต่ำเสียงก็จะต่ำ อวัยวะที่ทำให้เสียงมีระดับ สูง-ต่ำ คือ เส้นเสียง ดังนั้นระดับเสียงสูง-ต่ำก็คืออัตราการสั่นสะบัดของเส้นเสียงนั่นเอง

ในการพูดเสียงที่มีระดับสูง-ต่ำได้คือเสียงก้องเท่านั้นเพราะมีการสั่นสะเทือนของเส้นเสียงที่ทำให้เกิดมีความถี่ระดับต่างๆได้ ในภาษาไทยระดับเสียง สูง-ต่ำ ของคำเราเรียกว่า "วรรณยุกต์"

3.1.4 ความดัง (Loudness)

ความดังขึ้นอยู่กับปริมาณของลม ที่ผู้พูดแปล่งเสียงออกมาในช่วงเวลาหนึ่งๆ

3.1.5 การลดน้ำหนัก (Stress)

หมายถึง การออกเสียงพยางค์ใดพยางค์หนึ่งให้ดังเน้นมากหรือน้อยกว่าพยางค์อื่นที่อยู่ข้างเคียง (เพื่อต้องการเรียกร้องความสนใจเป็นพิเศษ หรือแสดงอารมณ์อย่างใดอย่างหนึ่ง)

3.1.6 ช่วงต่อของเสียง (Juncture)

หมายถึง ช่วงระยะเวลาที่ผู้พูดเปล่งเสียงหนึ่งแล้วต่อไปเปล่งอีกเสียงหนึ่งซึ่งเรียงกันมาเป็นลำดับ เสียงที่ประกอบกันเข้าเป็นพยางค์จะมีช่วงต่อของเสียงแนบสนิทจนไม่เห็นร่องรอย (Close juncture) แต่ถ้าเสียงปรากฏอยู่คนละพยางค์หรือคนละคำ จะมีช่วงต่อ "ห่าง" จนสังเกตได้ชัดเจน (Open juncture) ดังนั้น ช่วงต่อของเสียง โดยเฉพาะช่วงต่อหางจะมีความสำคัญมากในการแบ่งคำในภาษา

3.2 หน่วยเสียงสำคัญในภาษาไทย

"หน่วยเสียง" (Phoneme) เป็นหน่วยเล็กที่สุดของภาษา หน่วยดังกล่าวได้แก่เสียงสำคัญๆในภาษาใดภาษาหนึ่ง ซึ่งทำหน้าที่ให้ความหมายของคำที่ใช้ในภาษานั้น และทำให้ความหมายของคำนั้นๆ มีความหมายแตกต่างจากคำอื่น ๆ หน่วยเสียงที่สำคัญในภาษาไทยมี 3 ประเภทใหญ่ ๆ คือ เสียงพยัญชนะ เสียงสระ และเสียงวรรณยุกต์ ซึ่งหน่วยเสียงทั้ง 3 นี้เองที่ประกอบกันเข้าเป็นคำที่ใช้ในภาษาไทย แต่เครื่องมือใน HTK ไม่สามารถที่จะแยกแยะเสียงวรรณยุกต์ได้ ในที่นี้จะขอกล่าวถึงหน่วยเสียงที่สำคัญได้แก่ หน่วยเสียงสระ เสียงพยัญชนะ เสียงคำควบกล้ำ และเสียงตัวสะกด เท่านั้น

เสียงพูดของมนุษย์ซึ่งมีความแตกต่างกันมากมายนั้น ถ้าเราพิจารณาอย่างกว้าง ๆ จะพบว่าสามารถแบ่งออกเป็น 2 ประเภทใหญ่ คือ

1. **เสียงเรียง** (Segmental sound)

เป็นหน่วยเสียงที่สามารถแยกออกจากเสียงอื่นได้โดยเด็ดขาด เพราะมีลักษณะเด่นเฉพาะตัว ในภาษาไทยได้แก่เสียงสระ และเสียงพยัญชนะ

2. **เสียงซ้อน** (Supa-segmental)

เป็นเสียงที่ทำหน้าที่เป็นส่วนประกอบของเสียงอื่นเพราะไม่สามารถแยกเปล่งเสียงได้ตามลำพัง ในภาษาไทยได้แก่เสียงวรรณยุกต์และทำนองเสียง เป็นต้น

3.3 หน่วยเสียงสระ

3.3.1 ลักษณะของเสียงสระ

ลักษณะสำคัญของเสียงสระก็คือ "เป็นเสียงก้องที่เปล่งออกมาโดยให้ลมออกทางช่องปากโดยไม่ถูกลิ้นกักหรือขัดขวาง" ดังนั้นเวลาเราออกเสียงสระจะออกเสียงได้สะดวก และออกเสียงได้นาน ทั้งนี้เพราะคุณสมบัติของเสียงสระมีความดังเด่นกว่าเสียงอื่น ๆ ที่เรียงอยู่ข้างเสมอ อวัยวะที่เกี่ยวข้องกับการออกเสียงสระ ได้แก่ ลิ้น กับ ริมฝีปาก ถ้าลิ้นส่วนใดทำหน้าที่เพียงส่วนเดียว เสียงที่เกิดขึ้นก็จะมีเพียงเสียงเดียว เสียงเช่นนี้เรียกว่า "สระเดี่ยว" แต่ถ้าลิ้นส่วนอื่นทำหน้าที่ร่วมด้วยเสียงสระนั้นเรียกว่า "สระประสม"

สำหรับภาษาไทยมีหน่วยเสียงสระทั้งหมด 24 หน่วยเสียง แยกออกเป็นสระเดี่ยว 18 หน่วยเสียง และสระประสม 6 หน่วยเสียง

สระเดี่ยว

เสียงสระเดี่ยว 18 หน่วยเสียง พิจารณาจากการเกิดเสียงได้เป็น 2 กรณีใหญ่ ๆ คือ

1. พิจารณาการเกิดจากส่วนต่าง ๆ ของลิ้น หมายถึง ลมผ่านส่วนหน้า ส่วนกลาง หรือส่วนหลังของลิ้น
2. พิจารณาการเกิดจากลมผ่านลิ้นในขณะที่ลิ้นอยู่ในระดับ สูง กลาง ต่ำ

สระ	หน้า	กลาง	หลัง
สูง	อิ อี	อึ อือ	อุ อู
กลาง	เอะ เอ	เออะ เออ	โอะ โอ
ต่ำ	แอะ แอ	อะ อา	เออะ ออ

นอกจากนี้ หน่วยเสียงสระเดี่ยว 18 หน่วย สามารถแบ่งตามความสั้น-ยาวของการออกเสียงได้เป็น

- สระเดี่ยวเสียงสั้น 9 หน่วย ได้แก่ อะ อิ อึ อุ เอะ แอะ โอะ เอาะ เออะ
- สระเดี่ยวเสียงยาว 9 หน่วย ได้แก่ อา อี อือ อู เอ แอ โอ ออ เออ

สระประสม

เสียงสระประสม 6 หน่วยเสียง เกิดจากลมผ่านกระทบลิ้น 2 ส่วน คือ ส่วนบนและส่วนล่างซึ่งในขณะออกเสียงลิ้นจะอยู่ในระดับสูงแล้วลดลงต่ำ โดยเสียงหลังเป็นเสียงสระ อะ เสมอ

เสียงสระประสม 6 หน่วยเสียงได้แก่ เอียะ (อิ + อะ) เอีย (อี + อะ) เอือะ (อึ + อะ) เอือ (อือ + อะ) อัวะ (อุ + อะ) อัว (อู + อะ)

3.3.2 หน้าที่ของหน่วยเสียงสระในภาษาไทย

หน่วยเสียงสระในภาษาไทยทั้ง 24 หน่วยเสียงนี้ ทำหน้าที่เป็นแกนกลางของพยางค์หรือคำ กล่าวคือ คำ ทุกคำในภาษาไทยจะต้องมีเสียงสระอยู่ด้วย และเสียงสระในภาษาไทยจะสามารถเกิดกับเสียงพยัญชนะต้นได้ทุกเสียง และสามารถเกิดกับเกิดกับหน่วยเสียงวรรณยุกต์ได้ทุกหน่วย แต่ไม่สามารถเกิดกับหน่วยเสียงพยัญชนะสะกดได้ทุกหน่วย หน่วยเสียงสระที่ทำให้เกิดคำหรือพยางค์ ใช้ได้มากที่สุดในภาษามักเป็นหน่วยเสียงสระยาว

ตารางที่ 3.1 หน่วยเสียงในภาษาไทยเมื่อเทียบกับแบบสากล

ลำดับที่	เสียงสระ	สัญลักษณ์แทนหน่วยเสียง
1	อิ	i
2	อี	ih
3	อึ	v
4	อือ	vh
5	อุ	u
6	อู	uh
7	เอะ	e
8	เอ	eh
9	เออะ	er
10	เออ	erh
11	โอะ	o
12	โอ	oh
13	แอะ	x
14	แอ	xh
15	อะ	a
16	อา	ah
17	เอาะ	or
18	ออ	orh

3.4 หน่วยเสียงพยัญชนะ

เสียงพยัญชนะในภาษาไทยมีทั้งหมด 21 หน่วยเสียง (44 รูป) ดังแสดงในตารางที่ 3.3 หน่วยเสียงพยัญชนะออกเสียงได้ไม่สะดวกเท่าเสียงสระ เพราะเวลาออกเสียงลมหายใจที่พุ่งออกมาจากหลอดลมจะถูกขัดขวางจากส่วนต่างๆของปาก เสียงพยัญชนะจึงออกเสียงได้ยาวนานอย่างเสียงสระไม่ได้ และเสียงพยัญชนะก็ไม่ใช่เสียงก้องเสมอไป

3.4.1 ลักษณะของเสียงพยัญชนะ

หน่วยเสียงพยัญชนะ 21 หน่วยเสียงนี้จำแนกเป็น เสียงก้อง เสียงไม่ก้อง เสียงหนัก เสียงเบา และลักษณะการเกิดเสียง ดังนี้

เสียงก้อง (โฆษะ) มี 9 หน่วยเสียง คือ /ง/ /ย/ /บ/ /ด/ /ม/ /น/ /ร/ /ล/ /ว/

เสียงไม่ก้อง (อโฆษะ) มี 12 หน่วยเสียง คือ /ก/ /ค/ /จ/ /ช/ /ซ/ /ท/ /ต/ /ป/ /พ/ /ฟ/ /อ/ /ฮ/

เสียงหนัก (ธนิต) มี 4 หน่วยเสียง คือ /ค/ /ช/ /ท/ /พ/

เสียงเบา (สถิล) มี 4 หน่วยเสียง คือ /ก/ /จ/ /ต/ /ป/

ตารางที่ 3.2 เสียงพยัญชนะในภาษาไทย

ลำดับที่	อักษรไทยใช้แทนหน่วยเสียง	สัญลักษณ์แทนหน่วยเสียง	
		แบบสากล	แบบไทย
1	ก	g	ก
2	ข ฃ ค ฅ ฆ	kh	ค
3	ง	ng	ง
4	จ	j	จ
5	ฉ ช ฌ	ch	ช
6	ญ ย	y	ย
7	ซ ศ ษ ส	s	ซ
8	ฐ ฑ ฒ ถ ท ธ	th	ท
9	บ	b	บ
10	ฎ ด	d	ด
11	ฏ ต	dt	ต
12	ป	p	ป
13	ผ พ ภ	ph	พ
14	ฝ ฟ	f	ฟ
15	ม	m	ม
16	น ณ	n	น

ลำดับที่	อักษรไทยใช้แทนหน่วยเสียง	สัญลักษณ์แทนหน่วยเสียง	
		แบบสากล	แบบไทย
17	ร	r	ร
18	ล ฬ	l	ล
19	ว	w	ว
20	อ	-	อ
21	ฮ ห	h	ฮ
อักษร 44 รูป		21 หน่วยเสียง	

3.4.2 หน้าที่ของหน่วยเสียงพยัญชนะในภาษาไทย

เสียงพยัญชนะในภาษาไทย 21 หน่วยเสียงนี้สามารถทำหน้าที่ได้ดังนี้

1. เป็นพยัญชนะต้นของพยางค์ คือสามารถนำหน้าเสียงสระในพยางค์หนึ่งๆได้ ในตำแหน่งนี้เสียงพยัญชนะสามารถเกิดได้หน่วยเดียว หรือ สองหน่วยดังนี้
   - เกิดได้หน่วยเดียว คือ ทำหน้าที่เป็นพยัญชนะต้นเดี่ยว หน่วยเสียงทั้ง 21 หน่วยเสียงนี้สามารถทำหน้าที่เป็นพยัญชนะต้นเดี่ยวได้ทั้งสิ้น
   - เกิดได้สองหน่วย คือ ทำหน้าที่เป็นพยัญชนะต้นควบ โดยหน่วยเสียงแรกเป็น /ก/ /ค/ /ต/ /ป/ และ /พ/ กับหน่วยเสียงที่สองเป็น /ร/ /ล/ /ว/
2. เป็นพยัญชนะสะกดของพยางค์ ในตำแหน่งนี้เสียงพยัญชนะในภาษาไทยสามารถเกิดได้ 9 หน่วยเสียง คือ /บ/ (แม่กบ) /ด/ (แม่กด) /ก/ (แม่กก) /ม/ (แม่กม) /ง/ (แม่กง) /น/ (แม่กน) /ย/ (แม่เกย) /ว/ (แม่เกอว) /และ ไม่มีเสียงพยัญชนะสะกด (แม่กา)

ตารางที่ 3.3 เสียงพยัญชนะตัวสะกด

ลำดับที่	เสียงพยัญชนะตัวสะกด	สัญลักษณ์แทนหน่วยเสียง
1	กบ	b
2	กด	d
3	กก	g
4	กม	m
5	กง	ng
6	กน	n
7	เกย	y
8	เกอว	w
9	กา	-

บทที่ 4
การใช้งาน HTK

4.1 รู้จักกับ HTK

HTK หรือว่า Hidden Markov Model Toolkits เป็นเครื่องมือสำหรับสร้าง Hidden Markov model (HMMs) ซึ่งสามารถที่จะใช้งาน HMMs ในด้านต่างๆ โดยเฉพาะการทำการรู้จำ ซึ่ง HTK เป็นโปรแกรมอัฐประโยชน์ คือสามารถที่จะสร้างโมเดลได้อย่างยืดหยุ่น สามารถทำการรู้จำได้ทั้งแบบคำเดี่ยว หรือ แบบเป็นประโยคได้ซึ่งการที่จะสร้างโมเดลสำหรับการรู้จำนั้นจำเป็นต้องอาศัยเครื่องมือหลายๆอย่างใน HTK เพื่อทำการสร้าง ปรับปรุงโมเดล เพื่อให้ได้โมเดลที่มีประสิทธิภาพที่สุด

ซึ่ง HTK นั้นประกอบไปด้วยโปรแกรมจำนวนมากและมีหน้าที่ต่างกัน ซึ่งเป็นโปรแกรมที่ทำงานบน Command Line โดยในโครงงานนี้ใช้ HTK เวอร์ชัน 2.2 และทำงานบน Windows NT โดยการติดตั้งก็เพียงแต่ทำการ Copy ตัวโปรแกรมจากแผ่นมาเก็บไว้ในไดเรกทอรี่หลังจากนั้นก็ทำการสร้างและปรับปรุงไฟล์ .BAT และ ตั้ง PATH เพื่อชี้ไปยังไดเรกทอรีของโปรแกรม HTK ให้สามารถใช้งานได้จากทุก ๆ ไดเรกทอรี

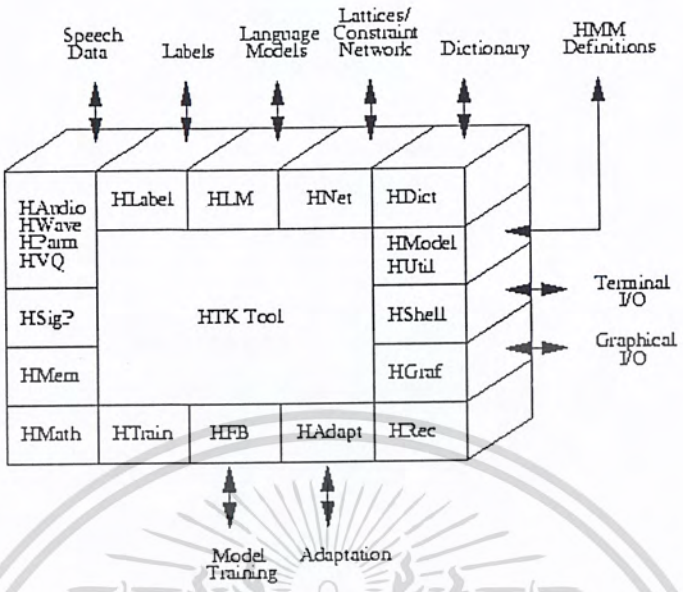
เนื่องจาก HTK เป็นโปรแกรมที่ทำงานบน Command Line ของ DOS และคำสั่งจะประกอบไปด้วยค่าพารามิเตอร์มากคำสั่งจึงยาวมาก ดังนั้นเพื่อเป็นการสะดวกในการสร้าง ปรับปรุงโมเดล หรือ การทดสอบโมเดล ซึ่งเมื่อตั้งค่าพารามิเตอร์ให้เหมาะสมกับการทำงานแล้วก็จะใช้ค่าเดิมตลอด เพื่อความสะดวกในการทำงาน ควรจะมีการเขียนแบทไฟล์การทำงานของแต่ละขั้นตอนเก็บไว้

4.2 โครงสร้างของโปรแกรม HTK

ฟังก์ชันหลักๆของ HTK จะถูกเก็บรวบรวมไว้รวมกันในไลบรารีโมดูล (Library Module) ซึ่งนั่นก็หมายความว่า เครื่องมือทุกๆอย่างที่มีใน HTK ที่ต้องการจะติดต่อกับโลกภายนอกเช่น อินพุท (Input) เอาท์พุท (Output) จะต้องติดต่อผ่านทางโมดูลนี้เท่านั้น เพื่อที่จะให้การทำงานเป็นไปในทางเดียวกัน ดังภาพที่ 4.1 จะแสดงถึงโครงสร้างโดยรวมของซอฟแวร์ HTK และแสดงถึงการติดต่อกับโลกภายนอก

เมื่อผู้ใช้ต้องการที่จะป้อนค่าต่างๆเข้าสู่สภาวะแวดล้อมของ HTK หรือต้องการให้ HTK แสดงผลออกมาและการกระทำการใดๆกับระบบจะถูกควบคุมโดยโมดูลที่ชื่อ HSHell การจัดการกับหน่วยความจำทั้งหมดจะถูกจัดการโดยโมดูลชื่อ HMem การจักการเกี่ยวกับคณิตศาสตร์ การคำนวณต่างๆ จะถูกจัดการโดยโมดูลชื่อ HMAth และการจัดการกับสัญญาณ (Signal Processing) ซึ่งจำเป็นในการวิเคราะห์สัญญาณเสียงจะถูกดำเนินการโดยโมดูลที่ชื่อ HSign ในการติดต่อกับไฟล์แต่ละแบบของ HTK จะถูกจัดการผ่านโมดูลหลายโมดูลดังนี้ HLabel จะใช้จัดการกับ Label ไฟล์ HLM สำหรับจัดการกับไฟล์โมเดลไวยกรของภาษา HNet ติดต่อกับเน็ตเวิก HDict สำหรับจัดการกับดิกชันนารี HModel สำหรับจักการกับโมเดล HMMs เป็นต้น

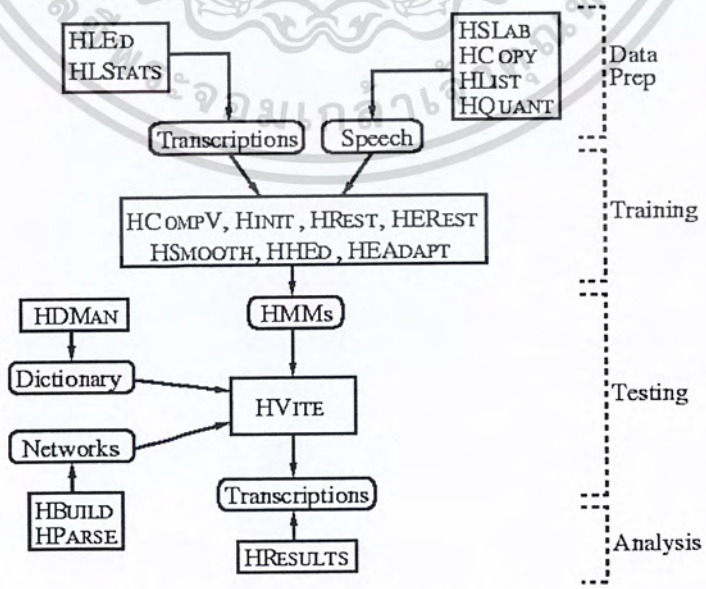
ในการควบคุมฟังก์ชันในไลบรารีโมดูลเหล่านี้ จะควบคุมผ่านทางการกำหนดค่าตัวแปรให้กับฟังก์ชัน หรือ ค่าพารามิเตอร์ ซึ่งจะพบในบทต่อๆไป



ภาพที่ 4.1 โครงสร้างของโปรแกรม HTK

4.3 เครื่องมือต่างๆใน HTK

ในการทำการรู้จำโดยใช้เครื่องมือใน HTK นั้น จะต้องใช้เครื่องมือหลายๆอย่างเพื่อสร้าง ปรับปรุง และทดสอบประเมินผล ของโมเดล ซึ่งต้องศึกษาการใช้งานเครื่องมือแต่ละอย่างว่ามีรายละเอียดปลีกย่อยอย่างไร ใช้ทำอะไรบ้าง เพื่อความสะดวกในการทำความเข้าใจ จะแบ่งเครื่องมือออกเป็น 4 ประเภท ตามประเภทการใช้งานได้แก่ เครื่องมือในการเตรียมข้อมูล เครื่องมือในการ Train โมเดล เครื่องมือในการทดสอบการรู้จำ และเครื่องมือในการวิเคราะห์ความถูกต้อง



ภาพที่ 4.2 ประเภทการจัดการของเครื่องมือใน HTK

4.3.1 เครื่องมือในการเตรียมข้อมูล (Data Preparation Tools)

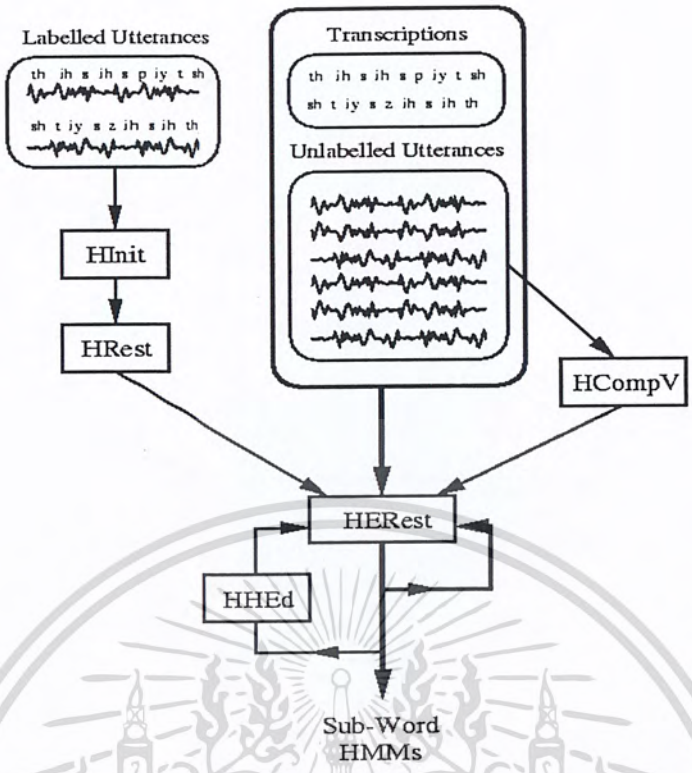
ในลำดับของการสร้างเซ็ตของ HMMs เซ็ตของไฟล์ข้อมูลเสียงพูดและเซ็ตของทราน สคริปชัน มีความสำคัญมากซึ่งต้องมีการจัดเตรียมก่อนจะทำการ Train บ่อยครั้งที่ข้อมูลเสียงพูดถูกนำมาจากฐานข้อมูลภายนอกอย่างเช่น CD-ROM หรือว่าจะทำการบันทึกเสียงเอง ก่อนที่จะนำข้อมูลเหล่านั้นมาทำการ Train จำเป็นต้องทำการแปลงไฟล์ข้อมูลเสียงไปเป็นไฟล์ในภาพแบบเฉพาะก่อน และไฟล์ Transcript ก็จะเป็นตัวบ่งบอกว่าข้อมูลไฟล์เสียงนั้นประกอบไปด้วยคำอะไรบ้าง ก็จะต้องถูกนำมาแปลงให้อยู่ในภาพแบบที่ถูกต้องเช่นกัน

ถึงแม้ว่าเครื่องมือแต่ละตัวใน HTK จะสามารถทำงานกับเวฟไฟล์ (Wave files) ได้แต่การทำงานกับ Wave files โดยตรงจะไม่ก่อให้เกิดผลดี ข้อมูลเสียงพูดใน Wave files จำเป็นต้องการขยาย หรือ รวมในบางส่วนของข้อมูลเสียงก่อนเพื่อให้ได้ข้อมูลเสียงที่รายละเอียดมากที่สุด จะทำให้การรู้จำเป็นไปได้ดีขึ้นมาก ในการแปลง Wave files จะใช้เครื่องมือใน HTK ที่ชื่อว่า HCopy ในการแปลงไปเป็นไฟล์ภาพแบบเฉพาะ ส่วนการแปลงไปในภาพแบบไหนจะขึ้นอยู่กับลักษณะการใช้งานและถูกกำหนดโดยค่าพารามิเตอร์

Transcription ไฟล์ข้อมูลเสียงมีความจำเป็นมากเช่นกัน เพราะถ้าไม่มี HTK ก็จะไม่รู้ว่าเสียงที่นำมาทำการ Train นั้นมีคำว่าอะไรบ้าง ในการเตรียม Transcription ไฟล์นั้นมีเครื่องมือใน HTK ที่ชื่อว่า HLEd สามารถที่จะแปลงจาก Transcription ไฟล์ไปเป็น Label ไฟล์ได้ โดยผลลัพธ์ที่ได้จะอยู่ในภาพแบบของ Master Label File (MLF) เพียงหนึ่งไฟล์ ซึ่งเหตุผลของการแปลงนั้นเพียงเพื่อให้การ Train เป็นไปอย่างสะดวกและง่ายขึ้นเพราะจะทำการแปลงจากคำให้อยู่ในภาพของหน่วยเสียงย่อย

4.3.2 เครื่องมือในการเทรนโมเดล (Training Tools)

ขั้นตอนที่สองของการสร้างระบบในการรู้จำคือการสร้างเซ็ตของ HMMs เพื่อทำการรู้จำคำศัพท์จากไฟล์เสียงพูดที่ได้ทำการอัดเข้าไป ซึ่งประกอบไปด้วยการทำ Prototype เพื่อทำการจัดการในเรื่องของการทำการ Initialize ค่าตัวแปรแบบจำลองก่อนโดยวิธีการแบบ Flat Start ซึ่งสามารถทำได้โดยการใช้เครื่องมือใน HTK ชื่อ HCompV ซึ่งข้อดีของ Flat Start ก็คือจะไม่มีการกำหนดของเขตของเสียงเองว่าเสียงอะไรอยู่ในช่วงไหนซึ่งจะทำให้การทำการรู้จำคำศัพท์ที่เป็นประโยคทำได้อย่างรวดเร็วขึ้น หลังจากที่ได้ทำการสร้างเซ็ตของ HMMs ของหน่วยเสียงขึ้นมาแล้วก็จะทำการเชื่อมหน่วยเสียงเหล่านั้นเข้าเป็นคำและเชื่อมคำให้เป็นประโยคต่อไปโดยเครื่องมือชื่อว่า HHEd อีกทั้งยังรวมไปถึงวิธีการในการปรับปรุงโมเดลอย่างเช่นการทำ Cluster ของแต่ละเสียงย่อยเพื่อนำข้อมูลมาใช้ร่วมกันและขณะเดียวกันก็ทำให้โมเดลมีความกระชับมากขึ้นซึ่งเป็นการทำโมเดลให้มีประสิทธิภาพดีขึ้นมากกว่าเดิมอีกด้วย



ภาพที่ 4.3 การ Train เสียงย่อยของโมเดล HMM

4.3.3 เครื่องมือในการทดสอบการรู้จำ (Recognition Tools)

HTK จะมีเครื่องมือที่ใช้ในการทดสอบโมเดลเรียกว่า HVITE ใช้วิธีการของ TokenPassing อัลกอริทึม ซึ่งจะทำในลักษณะของ Viterbi อัลกอริทึม ดังที่ได้อธิบายมาแล้วในบทต้นๆในการทดสอบการรู้จำ HVITE จะต้องมีการป้อนข้อมูลเข้าเป็นไฟล์ของ ไวยกร ที่มีการจัดลำดับของคำและดิกชันนารีที่เป็นตัวบ่งบอกว่าแต่ละคำออกเสียงอย่างไร และ เซ็ตของ HMMs โดยจะทำการแปลงในลักษณะของ word network ให้อยู่ในภาพของ phone network จากนั้นก็จะนำมาเปรียบเทียบกันกัน HMM definition (ไฟล์โมเดล) ของแต่ละเสียงย่อย

การทดสอบการรู้จำสามารถที่จะทำได้ทั้งไฟล์เสียงพูดที่อัดเก็บไว้ทั้งหมด หรือว่าจะทำการทดสอบจากเสียงที่พูดผ่านไมโครโฟนโดยตรงก็ได้

word network มีความจำเป็นต่อการใช้งาน HVITE ซึ่งโดยทั่งไปอาจจะเป็นแค่การวนคำ(word loop) แบบง่ายๆ ที่แต่ละคำสามารถที่จะตามด้วยคำใดๆก็ได้ หรือว่าอาจจะอยู่ในภาพแบบของ Task Grammar ก็ได้ซึ่งการลำดับของคำจะมีค่าความน่าจะเป็นซึ่งมีเครื่องมือ 2 ตัวในการช่วยสร้าง word network เครื่องมืออันแรกคือ HBUILD ที่สามารถสร้างโครงข่ายย่อยและสามารถนำไปใช้ในโครงข่ายระดับสูงได้ โดยสามรถที่จะอ่านไฟล์ในภาพแบบของ Backed-off bigram language model และทำการปรับปรุงการค่าความน่าจะเป็นในการเคลื่อนย้ายจากคำหนึ่งสู่อีกคำหนึ่ง ซึ่ง Backed-off bigram language model นี้ สามารถสร้างได้จากเครื่องมือที่ชื่อ HLStats

มีทางอื่นที่จะสามารถกำหนดภาพแบบของ word network ได้คือการใช้ไวยากรณ์ในระดับที่สูงขึ้นไปอย่างเช่น ใช้พื้นฐานของ Extended Backus Naur Form (EBNF) ที่ใช้กันในคอมไพเลอร์นั่นเอง ซึ่งสามารถนำมาใช้ร่วมกับภาษาในการเขียนไวยากรณ์ ใน HTK รุ่นใหม่ๆ จะมีเครื่องมือที่ชื่อ HPArse ใช้ในการแปลงจาก EBNF ให้อยู่ในภาพของ word network ที่ใช้งานได้

วิธีนี้จะต้องมีการทำความเข้าใจเกี่ยวกับตัวภาษาที่จะเขียนไวยากรณ์ ซึ่งสามารถที่จะใช้เครื่องมือช่วยได้ HGen จะทำการสุ่มและสับเปลี่ยนตำแหน่งคำเพื่อให้ได้การเชื่อมต่อของคำหลายๆ แบบจากนั้นก็ทำการเลือกเอาแบบที่ถูกต้องมาทำการอัดเสียงหรือทำการเขียนไวยกรได้สะดวกขึ้น

4.3.4 เครื่องมือในการวิเคราะห์ความถูกต้อง (Analysis Tools)

หลังจากที่โมเดลได้ผ่านขั้นตอนการ Train และ ถูกทำการทดสอบกับเสียงพูดจริงแล้ว ก็มีความจำเป็นที่จะต้องทำการประเมินค่าประสิทธิภาพของโมเดลซึ่งโดยทั่วไปจะใช้ไฟล์ข้อมูลที่ได้จากการทดสอบ (Recognition) ซึ่งจะเป็นไฟล์ที่เก็บค่าทดสอบว่าคำที่วิเคราะห์จากเสียงที่พูดเข้าไปตรงกันกับสคริปหรือไม่ ซึ่งจะนำมาเปรียบเทียบกันไปตามลำดับ และคิดค่าออกมาเป็นเปอร์เซ็นต์ความถูกต้อง โดยเครื่องมือใน HTK ที่ใช้วัดค่าประสิทธิภาพมีชื่อว่า HREsults ซึ่งใช้หลักการเปิดไฟล์ 2 ไฟล์โดยไฟล์แรกจะเป็นประโยคที่ถูกต้องและอีกไฟล์จะเป็นประโยคที่ได้จากการวิเคราะห์จากเสียงพูด เมื่อเปรียบเทียบเสร็จแล้ว จากนั้นก็จำทำการสับเปลี่ยนที่คำ (substitution) ทำการลบคำออก (Deletion) การเพิ่มคำ (Insertion) เพื่อประเมินค่าความผิดพลาดแบบต่างๆ ค่าพารามิเตอร์ต่างๆจะถูกใส่เข้าไปเพื่อกำหนดอัลกอริทึมและภาพแบบผลลัพธ์ที่จะให้ทำงานซึ่งทุกอย่างจะมีการกำหนดเป็นมาตรฐาน

บทที่ 5
การเตรียมข้อข้อมูล

ก่อนที่จะทำการสร้างโมเดลในการรู้จำนั้น เราจำเป็นต้องมีการเตรียมข้อมูลที่จำเป็นในการสร้างโมเดลเสียก่อน ซึ่งการเตรียมข้อมูลที่เหมาสมและถูกต้อง ก็เป็นสิ่งที่จะทำให้การสร้างโมเดลง่ายและไม่ผิดพลาด ซึ่งข้อมูลต่างๆที่จะเป็นต้องเตรียมและวิธีการเตรียมนั้นจะได้กล่าวถึงทีละอย่างภายในบทนี้

5.1 การสร้างดิกชันนารี (Creating Dictionary)

Dictionary เป็นเท็กซ์ไฟล์ที่เก็บรายการของคำที่เรียงกันแล้วทั้งหมดที่ใช้ในการรู้จำ และเสียงย่อย (phoneme) ของแต่ละคำ ซึ่งสร้างได้โดยใช้ Text Editor ทั่วไป ภาพแบบของ Dictionary จะมีภาพแบบดังนี้

WORD [output] phoneme1 phoneme2 phoneme3 …

ตัวอย่าง

```
[] sil
silence [] sil
sil [] sil
SENT-START []
SENT-END []
กรุง g r u ng sp
การ g ah n sp
…
```

จากข้างบนหลังคำว่า SENT-START และ SENT-END จะมี Silence Model (sil) เป็นการสะกดการออกเสียงและมี output เป็น null หรือ ไม่มีอะไรเลย

ซึ่งเสียงย่อยในภาษาไทยนั้นก็มีอยู่ไม่มากทำให้การสร้างคำเป็นไปได้อย่างง่ายและไม่สิ้นเปลือง ซึ่งสัญลักษณ์แทนเสียงย่อยในภาษาไทยได้พูดไปแล้วในบทที่ผ่านมา

5.2 ไฟล์เสียง (Wave files)

ในการทำการรู้จำและการทดสอบประสิทธิภาพของการรู้จำเสียงนั้น จำเป็นอย่างยิ่งที่จะต้องมีการใช้ไฟล์เสียงเข้ามาเป็นส่วนประกอบ ซึ่งคำศัพท์แต่ละคำจะต้องใช้ข้อมูลจากไฟล์เสียงเป็นตัวบ่งชี้ว่าคำนั้นออกเสียงอย่างไร เนื่องจากการออกเสียงของคำ จะแตกต่างออกไปในการออกเสียงแต่ละครั้ง แต่ละประโยค ดังนั้นจึงจำเป็นจะต้องใช้ไฟล์เสียงจำนวนมากต่อหนึ่งคำศัพท์ เพื่อการรู้จำที่ดี ทั้งนี้เราสามารถใช้โปรแกรมสำเร็จภาพทั่วไปในการบันทึกเสียงได้

ภาพแบบของไฟล์เสียงที่ใช้นั้น เป็นไฟล์ .WAV 16 บิท Mono และมี sampling rate เท่ากับ 44 Khz ซึ่งเป็นค่าที่อาจจะเกินความจำเป็นเพราะว่าปกติแล้วเสียงมนุษย์อยู่ในช่วงความถี่ 20-20,000 Hz ซึ่งเราสามารถกำหนด Sampling rate เพื่อให้ได้ข้อมูลมาอย่างครบถ้วนนั้นก็จะกำหนดแค่ครึ่งหนึ่งของความถี่ของ Source ซึ่งอัตรา Sampling ที่น่าจะเพียงพอก็ควรจะอยู่ที่ประมาณ 10,000 Hz แต่ด้วยเหตุผลที่ว่าในการที่จะทำการแปลงค่าเสียงให้เป็นคำนั้น (Recognize) เราทำการ Train ด้วยข้อมูลเสียงในการสร้างโมเดลด้วยข้อมูลชนิดไหนก็ควรจะใช้โมเดลนั้นในการ Recognize เสียงในลักษณะเดียวกันเพื่อให้เกิดความถูกต้องมากที่สุดและเนื่องจากปัจจุบันนิยมใช้เสียง 16 บิทและ sampling rate เท่ากับ 44 KHz ซึ่งให้คุณภาพเสียงที่ธรรมชาติจึงเป็นเหตุผลในการเลือกใช้

ภาพแบบของไฟล์ .WAV นั้น เป็นภาพแบบหนึ่งในภาพแบบไฟล์ชนิด RIFF ของบริษัทไมโครซอฟท์ใช้เพื่อการเก็บข้อมูลทางด้านมัลติมีเดีย ไฟล์ชนิด RIFF นั้นจะเริ่มต้นด้วยไฟล์เฮดเดอร์ ตามด้วยกลุ่มของข้อมูลที่จัดเรียงตามลำดับ ไฟล์ .WAV โดยมากจะมีกลุ่มของข้อมูลเสียงเพียงกลุ่มเดียว โดยกลุ่มข้อมูลนี้ จะถูกแบ่งย่อยออกไปเป็นสองส่วน ส่วนแรกจะเป็นส่วนที่ใช้บ่งบอกว่าภาพแบบข้อมูลเป็นอย่างไร ส่วนที่สองเป็นส่วนของข้อมูลจริง

5.2.1 ไฟล์ลักษณะเฉพาะ (MFCCs)

ในส่วนของการเตรียมข้อมูลนั้น ต่อมาจะเป็นการทำการแปลงสัญญาณเสียงจากภาพแบบสัญญาณคลื่นเสียงที่ได้ทำการบันทึกมาก่อนหน้านี้ ให้ออกมาเป็นเสียงในภาพแบบของเวคเตอร์สำหรับ HTK โดยนำไปผ่านกระบวนการแยกแยะคุณลักษณะเฉพาะออกมาโดยใช้เครื่องมือของ HTK ชื่อ HCopy ซึ่งจะทำการแปลงสัญญาณเสียงให้อยู่ในภาพแบบของ Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs เป็นภาพแบบการถอดรหัสสัญญาณเสียงวิธีหนึ่งซึ่งได้จากการแปลงจากไฟล์เสียงมาตรฐานเพื่อทำการเก็บลักษณะสำคัญของเสียงให้มากที่สุด ซึ่งได้รับความนิยมมากเพราะในตอนนี้ MFCCs เป็นภาพแบบที่ให้ความถูกต้องมากที่สุด

5.2.2 เครื่องมือ HCopy

โปรแกรมจะทำการคัดลอกไฟล์ข้อมูลต่างๆให้อยู่ในภาพแบบของเอาท์พุทไฟล์ที่ได้ทำการกำหนดไว้หรือกล่าวอีกนัยหนึ่งคือ การแปลงข้อมูลต่างๆ ให้อยู่ในภาพแบบที่สามารถวัดค่าได้ สำหรับ Source ไฟล์ (source file) ต่างๆนั้นจะอยู่ในภาพแบบใดก็ได้ที่โปรแกรมนี้สามารถรองรับได้ แต่เอาท์พุทไฟล์ที่ออกมานั้น จะต้องอยู่ในภาพที่กำหนดไว้โดย HTK เสมอ โดยปกติแล้ว ในการคัดลอก เราจะทำการคัดลอก Source ไฟล์ทั้งหมด คัดลอกลงสู่ตำแหน่งเป้าหมายที่ต้องการ แต่สำหรับโปรแกรมตัวนี้ จะคัดลอกเฉพาะในส่วนของไฟล์ทีมีลักษณะพิเศษมาเท่านั้น ในที่นี้ โปรแกรม HCopy ถูกใช้ในการแปลงไฟล์ข้อมูลที่อยู่ในภาพแบบต่างๆให้มาอยู่ในภาพที่กำหนดไว้ของ HTK จัดการแบ่งไฟล์ข้อมูลให้เป็นสัดส่วน หรือใช้ในการวัดค่าผลที่ออกมา เมื่อมีการสั่งทำงานโปรแกรมนี้ขึ้นมา จะมีการกระทำการคัดลอกไฟล์ข้อมูลมาเป็นส่วนๆ และส่วนต่างๆของไฟล์ข้อมูลทั้งหมดจะถูกคัดลอกสู่ไฟล์เป้าหมาย

HCopy ยังมีความสามารถใช้แปลงข้อมูลออกเป็นหลายๆภาพแบบ เช่น สามารถแปลงคลื่นเสียงเป็น MFCC หรือ LPC ได้โดยทั้งนี้ขึ้นอยู่กับการตั้งค่าคอนฟิกไว้อย่างไร และขึ้นอยู่กับภาพแบบของไฟล์ข้อมูลว่าสามารถแปลงเป็นภาพแบบใดได้บ้าง

ภาพแบบคำสั่งการใช้งาน HCopy เป็นดังนี้

HCopy src tgt

src คือ Source ไฟล์ ส่วน tgt คือ ไฟล์เป้าหมาย

HCopy ยังสามารถที่จะจัดข้อมูลให้เป็นสัดส่วน และจัดเก็บไว้ในไฟล์เป้าหมายไฟล์เดียวได้อีกด้วย โดยใช้คำสั่ง

HCopy src1 + src2 + src3 tgt

นอกจากจะสามารถจัดเก็บข้อมูลไว้ในไฟล์เดียว HCopy ยังสามารถจัดเก็บข้อมูลแยกไฟล์เป้าหมายตาม Source ไฟล์ได้อีกด้วย โดยเพิ่มคำสั่ง –s และ –e ต่อท้ายคำสั่ง HCopy เข้าไป ดังนี้

HCopy -s 100 -e -100 src tgt

โปรแกรมจะทำการแปลงข้อมูลที่ 100 จนถึงข้อมูลที่ N-100 ของ Source ไฟล์ ลงสู่ไฟล์เป้าหมาย N คือค่ารวมของจำนวน Source ไฟล์ทั้งหมด ทั้งนี้อาจจะเป็นหมายเลขของไฟล์ก็ได้ ขนาดของข้อมูลที่จะถูกคัดลอกนั้นขึ้นอยู่กับขนาดของ Source ไฟล์นั้น

หรือในอีกภาพแบบ เราสามารถแปลงคำสั่ง –s –e ให้อยู่ในภาพแบบของสคริปท์ไฟล์ (Script File) ซึ่งจะช่วยให้การทำงานกับไฟล์เสียงจำนวนมากๆ เป็นไปได้อย่างรวดเร็วไม่ต้องมาทำทีละไฟล์ ส่วนของภาพแบบ Script File แสดงอยู่ในส่วนของภาคผนวก

อย่างไรก็ตามในการใช้งาน HCopy นั้น ขึ้นอยู่กับว่าเราใช้โปรแกรมเพื่อประโยชน์ใด โปรแกรมจะรับข้อมูลภาพแบบใดเข้ามา และต้องการข้อมูลในไฟล์เอาท์พุทออกมาในภาพแบบใด HCopy จึงต้องมีการใช้ไฟล์คอนฟิกเข้ามา เพื่อกำหนดค่าการทำงานของโปรแกรม

ในขั้นตอนการเตรียมข้อมูลของไฟล์เสียง จะมีการแปลงข้อมูลจากไฟล์เสียง ให้อยู่ในภาพแบบ MFCCs เราจึงมีการกำหนดไฟล์คอนฟิก และกำหนดค่าต่างๆที่จำเป็นไว้ดังนี้

```
#Coding Parameters
SOURCEKIND=WAVEFORM
SOURCEFORMAT=NOHEAD
SOURCERATE=625
TARGETKIND=MFCC_D_A_0
TARGETRATE=100000.0
SAVECOMPRESSED=F
SAVEWITHCRC=F
USEHAMMING=T
ENORMALISE=T
PREEMCOEF=0.97
```

NUMCHANS=24

NUMCEPS=12

WINDOWSIZE=250000.0

ค่าที่ไม่ถูกกำหนดจะถือว่าใช้ค่าปกติของโปรแกรม สำหรับค่าที่ถูกกำหนดไว้ในไฟล์คอนฟิกนี้ได้แก่ ภาพแบบของข้อมูลที่เข้ามาจะอยู่ในภาพของสัญญาณคลื่น ภาพแบบของข้อมูลที่ออกมาจะต้องอยู่ในภาพของ MFCCs การจัดเก็บข้อมูลโดยให้บีบอัดหรือไม่

ในการใช้งานไฟล์คอนฟิกทำได้โดยใช้คำสั่ง

HCopy -c configfile –s scriptfile

คำสั่งของโปรแกรม HCopy ที่สำคัญ มีดังนี้

-e f หยุดการคัดลอก Source ไฟล์ ณ เวลาที่ f โดยค่าปกติจะถูกตั้งไว้ที่ตำแหน่งสุดท้ายของไฟล์ ถ้า f ถ้า f มีค่าเป็นลบหรือศูนย์ การทำงานจะหยุดลงที่จุดสิ้นสุดของไฟล์

-i mlf โปรแกรมจะทำการคัดลอกทับลงใส่ไฟล์ต้นแบบ

-l s จะทำการคัดลอกไฟล์ลงสู่ไดเรกทอรี s ค่าปกติจะตั้งไว้ที่ไดเรกทอรีเดิม

-s f เริ่มทำการคัดลอก Source ไฟล์ ณ เวลาที่ f ค่าปกติจะตั้งไว้ที่เวลา 0.0 วินาที หรือจุดเริ่มต้นของไฟล์

-F fmt ตั้งค่าภาพแบบของข้อมูลใน Source ไฟล์เป็นแบบ fmt

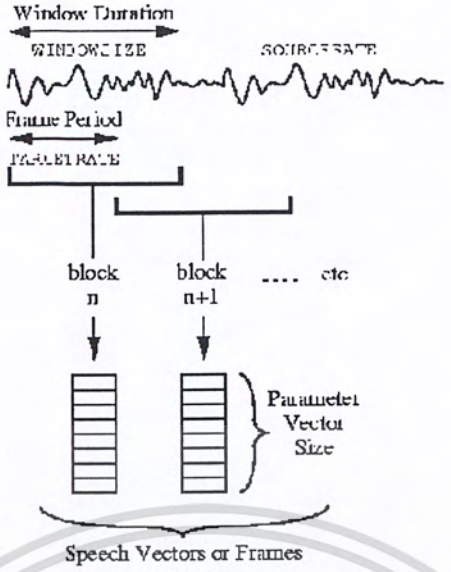
-L dir ค้นหา Source ไฟล์จากไดเรกทอรี dir

-O fmt ตั้งค่าภาพแบบข้อมูลที่จะได้ออกมาเป็นแบบ fmt

5.2.3 กระบวนการแปลงสัญญาณเสียงพูด (Speech Signal Processing)

กระบวนการแปลงข้อมูลสัญญาณคลื่นเสียงให้อยู่ในภาพของเวคเตอร์ดังภาพที่ 5.1 โดยปกติ HTK จะทำงานทั้งในส่วนไฟล์สัญญาณและพารามิเตอร์ไฟล์ (Parameter file) ไปพร้อมๆกัน โดยทำการทดสอบไฟล์สัญญาณแยกเป็นส่วนๆไป สิ่งเดียวที่แตกต่างจากการคัดลอกไฟล์ปกติคือ โดยปกติการคัดลอกข้อมูลที่คัดลอกมาจะอยู่ในภาพของตัวแปรแบบจำนวนเต็ม (Integer) ขนาด 2 byte แต่ในกระบวนการนี้ ข้อมูลที่คัดลอกมาจะอยู่ในภาพของชุดของเวคเตอร์ อัตราความเร็วในการทดสอบสัญญาณโดยปกติจะถูกกำหนดจากตัวข้อมูลที่ต้องการคัดลอกเอง

อย่างไรก็ตามอัตราความเร็วในการทดสอบสัญญาณนั้น เราสามารถกำหนดไว้ได้ในคอนฟิกไฟล์ โดยกำหนดในค่าตัวแปร SOURCERATE ช่วงเวลาระหว่างการเริ่มต้นแปลงข้อมูลของแต่ละชุดเวคเตอร์กำหนดในตัวแปร TARGETRATE ขนาดของส่วนของสัญญาณที่เรานำมาแปลงเป็น เวคเตอร์หนึ่งๆ กำหนดจากขนาดของวินโดว์ โดยเราสามารถกำหนดขนาดของวินโดว์ได้ โดยกำหนดในค่าตัวแปร WINDOWSIZE ข้อควรจำก็คือ ขนาดของวินโดว์และช่วงเวลาระหว่างการเริ่มต้นแปลงข้อมูลของแต่ละชุดเวคเตอร์เป็นคนละตัวกัน โดยปกติแล้วขนาดของวินโดว์จะมีขนาดใหญ่กว่าช่วงเวลาระหว่างการเริ่มต้นแปลงข้อมูลของแต่ละชุดเวคเตอร์ เพื่อจะทำให้เกิดการโอเวอร์แล็ปกันระหว่างชุดเวคเตอร์



ภาพที่ 5.1 กระบวนการถอดรหัสเสียงพูด

ตัวอย่างของการตั้งค่าเป็นดังนี้ ถ้าสัญญาณที่เราจะนำมาผ่านกระบวนการมีความถี่ 16 KHz และต้องการให้แปลงข้อมูลออกมาอยู่ในภาพของเวคเตอร์ ที่มีช่วงเวลาระหว่างการเริ่มต้นแปลงข้อมูลของแต่ละชุดเวคเตอร์ 100 ชุดเวคเตอร์ต่อวินาที และมีขนาดวินโดว์เท่ากับ 25 msec เราจะต้องทำการตั้งค่าไว้ในคอนฟิกไฟล์ดังนี้

SOURCERATE=625

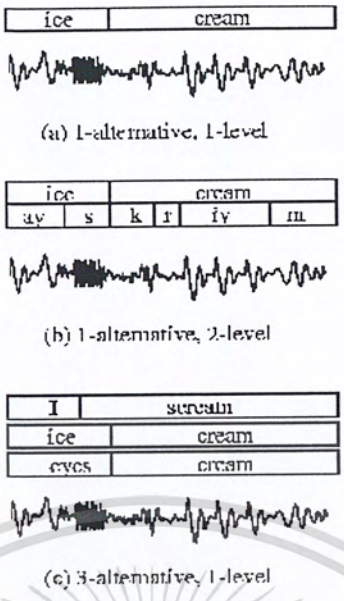
TARGETRATE=10000

WINDOWSIZE=25000

5.3 การสร้างทรานสกริปชันไฟล์ (Creating word level transcription file)

ในกระบวนการรู้จำที่กระทำโดย HTK มีความจำเป็นที่จะต้องนำข้อมูลของไฟล์เสียงแต่ละไฟล์เสียงมาเกี่ยวข้อง เพื่อนำมาใช้บ่งชี้ว่าในเสียงที่นำมาทำการรู้จำหนึ่งเสียงนั้น เสียงถูกแบ่งออกเป็นคำศัพท์อะไรบ้างในแต่ละส่วนของเสียงนั้น เราเรียกมันว่า Label

Label file ส่วนใหญ่ที่ใช้กันอยู่นั้น จะเป็นแบบ single-alternative และ single-level ภาพ 5.2(a) กล่าวคือ เสียงที่ออกเสียงเหมือนกันจะถูกมองออกมาเป็นคำศัพท์คำเดียวกัน และในการแยกแยะเสียง มีการแยกแยะเพียงระดับเดียว เราสามารถเพิ่มระดับความสามารถในการแยกแยะเสียง โดยทำการเพิ่ม level ให้กับ label file ดังที่แสดงในภาพ 5.2(b) แต่การเพิ่มระดับ level ให้กับ label file นั้น เรายังจำเป็นที่จะต้องใช้ lower level ในการแยกแยะศัพท์อยู่ดี เราสามารถแยกแยะเสียงเดียวออกมาเป็นคำศัพท์หลายๆคำได้เช่นกัน โดยทำการเพิ่ม alternative เข้าไป ดังที่แสดงในภาพ 5.2(c)



ภาพที่ 5.2 ตัวอย่าง Transcription file

ภาพแบบของ Label File โดยส่วนใหญ่จะทำเป็น text โดยในหนึ่ง label file สามารถทำเป็น multiple-alternative และ multiple-level ได้ ในแต่ละบรรทัดของไฟล์ จะมีลักษณะเป็นดังนี้

[start [end]] name

start คือเวลาเริ่มต้น มีหน่วยเป็น 100 ns end คือ เวลาสิ้นสุดของส่วนของเสียงนั้น name คือ ชื่อที่เราใช้เรียกส่วนของเสียงนั้น

ตัวอย่างของ label file ภาพแบบพื้นฐาน ในภาพ 5.2(a) เป็นดังนี้

0000000 3600000 ice

3600000 8200000 cream

การเพิ่มระดับของการแยกแยะเสียง ทำได้โดยการเพิ่มชื่อย่อยเข้าไปในชื่อแบบพื้นฐาน โดยที่เราจะต้องทำการทำ label file ให้อยู่ในภาพแบบพื้นฐานแล้วจึงมาแยกย่อยเสียง โดยแบ่งตามช่วงเวลาอีกที ตัวอย่างของ label file ตามภาพ 5.2(b) เป็นดังนี้

0000000 2200000 ay ice

2200000 3600000 s

3600000 4300000 k cream

4300000 5000000 r

5000000 7400000 iy

7400000 8200000 m

สุดท้ายการเพิ่ม alternative หรือการแยกแยะคำศัพท์ที่ออกเสียงอย่างเดียวกันออกมาเป็นคำศัพท์หลายๆคำ ทำได้โดยการใช้เครื่องหมาย /// คั่นกลางระหว่างแต่ละคำศัพท์ ตัวอย่าง label file ตามภาพ 5.2(c) เป็นดังนี้

0000000 2200000 I

2200000 8200000 screams

///

0000000 3600000 Ice

3600000 8200000 cream

///

0000000 3600000 eyes

3600000 8200000 creams

เนื่องจากการกำหนดลักษณะของ label file ตามภาพแบบ ดังที่กล่าวไปนี้ แม้จะเป็นภาพแบบที่ง่ายแต่ละไฟล์เสียงก็จะมี label file เป็นของตัวเองหนึ่งไฟล์แต่ในบางกรณี การใช้ label file ในภาพแบบนี้ ก็ทำให้เกิดปัญหาได้ยกตัวอย่างเช่น ถ้าไฟล์เสียงเป็นศัพท์คำเดียวกันและมีการออกเสียงเหมือนกันทั้งหมด การกำหนด label file เช่นนี้ ทำให้เราต้องสิ้นเปลืองพื้นที่ในการเก็บ label file เนื่องจาก label file ทุกไฟล์จะมีข้อมูลแบบเดียวกันหมด นอกจากนี้ยังเกิดปัญหาเรื่องความสะดวกในการใช้ เนื่องจาก label file ทุกตัวจะต้องเก็บในไดเรกทอรีเดียวกับไฟล์เสียง

ปัญหาทั้งหมดถูกแก้ไขโดยการใช้ label file แบบ Master Label File (MLF) โดยในเครื่องมือของ HTK ทุกชนิด มีการอนุญาตให้ใช้ label file ในลักษณะนี้ได้ โดยเพิ่มคำสั่ง –I เข้าไป ใน label ปกติ เมื่อมีการสั่งงาน โปรแกรมจะทำการเปิดไฟล์ทั้งหมดเมื่อไม่สามารถเปิดไฟล์ใดได้ ก็จะเกิด error ขึ้น แต่ในกรณีเมื่อมีการโหลด MLF เข้ามา โปรแกรมจะทำการค้นหาไฟล์ที่ต้องการใช้ทั้งหมดก่อน เมื่อพบทั้งหมด จึงจะทำการเปิดไฟล์ขึ้นมา

MLF มีความสามารถสองอย่างคือ อย่างแรกเก็บ label ได้ไม่จำกัดจำนวน ดังนั้นเราจึงสามารถรวม label จำนวนมากหรือทั้งหมดไว้ในไฟล์เดียวกัน ความสามารถอย่างที่สองของ MLF คือ มันสามารถเก็บรายการของไดเรกทอรีย่อย ดังนั้นมันสามารถค้นหา label files ในไดเรกทอรีย่อยต่างๆได้ ความสามารถทั้งสองอย่างสามารถใช้ใน MLF ไฟล์เดียวกันได้

แม้ MLF จะมีความซับซ้อนในการใช้งานและยากในการทำความเข้าใจ อย่างไรก็ตาม มันมีความสามารถในการตัดสินใจและมีความยืดหยุ่นกว่า นอกจากนี้ HTK ยังมีการเพิ่มความสามารถให้มัน โดยสามารถให้มันใช้งานร่วมกับคำสั่ง –S และ –L ซึ่งทำให้มันยังสามารถใช้งานกับทุกชนิดโครงสร้างข้อมูลและยังสามารถนำ label files อื่นๆมาใช้งานร่วมได้อีกด้วย

Syntax and Semantics

ใน MLF นั้นจะประกอบไปด้วยหนึ่งหรือหลายๆคำศัพท์ในหนึ่งไฟล์ ไม่อนุญาตให้มีการเว้นบรรทัดเอาไว้ ไฟล์จะต้องเริ่มบรรทัดแรกด้วย #!MLF!# เสมอ เพื่อระบุว่าไฟล์นี้เป็นไฟล์ MLF เนื่องจากไฟล์ไม่จำเป็นที่จะต้องใช้ร่วมกับคำสั่ง –I เพียงคำสั่งเดียว จึงต้องมีการระบุไว้ในบรรทัดแรกว่าไฟล์นี้เป็นไฟล์ MLF สำหรับ syntax ต่อมาสามารถอธิบายได้โดยใช้สัญลักษณ์ทาง BNF อธิบาย ความหมายของ

สัญลักษณ์ต่างๆใน BNF มีดังนี้ | ใช้แบ่งระหว่างทางเลือก, ( ) ใช้แยกตัวแปรออกจากกัน, [ ] ใช้แยกออปชั่น, { } ใช้แทนการซ้ำกัน

MLF = "#!MLF!#"

MLFDef {MLFDef}

ในแต่ละคำศัพท์เราสามารถนำเอา transcript มาจากไดเรกทอรีเดิมได้เลย หรือให้ MLF ค้นหาจากไดเรกทอรีย่อย

MLFDef = ImmediateTranscription | SubDirDef

ในกรณีที่ MLF สามารถโหลด transcript จากไฟล์เดิมได้เลย (Immediate Transcription) ภาพแบบในแต่ละบรรทัดจะประกอบด้วย pattern ในบรรทัดแรก ตามด้วย transcription ในบรรทัดต่อมา ซึ่ง MLF จะมองมันในภาพของ text เมื่อจบหนึ่ง transcript ก็จะใช้เครื่องหมายจุดแสดงแทน

ImmediateTranscription =

Pattern

Transcription

"."

ในกรณีที่เราให้ MLF ค้นหาจากไดเรกทอรีย่อย ก็จะต้องมีการระบุไดเรกทอรีย่อยที่ต้องการให้ค้นหาเสียก่อน ถ้า MLF สามารถค้นหา label file ที่ต้องการเจอก็จะทำการโหลดไฟล์นั้นขึ้นมา

SubDirDef = Pattern SearchMode String

SearchMode = "->" | "=>"

ข้อแตกต่างระหว่าง Search mode สองแบบ ก็คือในแบบ -> ถ้าเราสั่งให้เข้าไปค้นหาข้อมูลในไดเรกทอรี ...d3/d2/d1/ มันจะเข้าไปค้นหาที่ไดเรกทอรีนั้นเลย ถ้าค้นหาไม่เจอก็จะมีการรายงานความผิดพลาด ส่วนในการค้นหาแบบ => นั้น การค้นหาจะทำในไดเรกทอรีเดิม, ../d1/, ../d2/d1/, ../d3/d2/d1/ เจอไฟล์ดังกล่าวที่ใด ก็จะทำการโหลดมาเก็บไว้ทันที โดยไม่ต้องเข้าไปถึงไดเรกทอรีย่อยสุด

MLF Example

1. เมื่อมีชุดข้อมูลสองชุดต้องการนำมาเทรนดังนี้ :

ข้อมูลชุด a

0000000 0590000 sil

0600000 2090000 a

2100000 4500000 sil

ข้อมูลชุด b

0000000 0990000 sil

1000000 3090000 b

9100000 4200000 sil

เราสามารถรวมข้อมูลทั้งสองชุดใน MLF เดียวได้ ดังนี้

#!MLF!#

"*/a.lab"

0000000 0590000 sil

0600000 2090000 a

2100000 4500000 sil

.

"*/b.lab"

0000000 0990000 sil

1000000 3090000 b

9100000 4200000 sil

2. ในกรณีที่ไฟล์เสียง one.1.wav, one.2.wav, one.3.wav,..... มีเสียงที่ใช้ label file ร่วมกันได้ ในกรณีที่เราไม่ได้ใช้ MLF เราจำเป็นจะต้องมี label file แทนแต่ละไฟล์เสียง แต่ในกรณีที่เราใช้ MLF เราสามารถช่วยให้ไฟล์เสียงทั้งหมดใช้ label เดียวกันได้เลย โดยทำได้ดังนี้

#!MLF!#

"*/one.*.lab"

one

.

"*/two.*.lab"

two

.

3. ในกรณีที่ฐานข้อมูลที่ใช้ประกอบด้วยหลายไดเรกทอรีย่อย ในแต่ละไดเรกทอรีมี label file ที่ใช้สำหรับไดเรกทอรีนั้น MLF สามารถช่วยให้ Label ทั้งหมดใช้งานพร้อมกันได้โดยใช้คำสั่ง

#!MLF!#

"*" -> "db/d1/labs"

"*" -> "db/d2/labs"

"*" -> "db/d3/labs"

....

"*" -> "db/d8/labs"

Specify of MLF

ในการ Train เซ็ตของ HMM ทุกๆไฟล์ของข้อมูลที่จะนำมา Train นั้น จำเป็นจะต้องมี Transcript ในภาพแบบของ phone level โดยในกระบวนการนี้ จำเป็นต้องใช้ชุดของ phone level transcript 2 ชุด ในแต่ละชุดจะต้องไม่มี short-pause (sp) หลังจากที่ phone models ถูกสร้างขึ้นแล้วจะมีการใส่ sp ลงไประหว่างคำเพื่อจัดการช่องว่างของเสียงระหว่างคำที่เกิดขึ้นในการอัดเสียง

จุดเริ่มต้นของ phone level transcript ทั้งสองตัวนั้น จะอยู่ในภาพแบบของ HTK Label โดยสามารถใช้ Text editor ในการสร้างได้โดยง่าย ตัวอย่างของ transcription file เป็นดังนี้

#!MLF!#
"*/s0001.lab"
one
validated
acts
of
school
.
"*/s0002.lab"
two
other
cases
also
were
under
advertisement
.
(etc.)

ในแต่ละ label จะต้องมีการแปลงออกมาเป็นส่วนต่างๆ โดยแบ่งตามคำศัพท์ แต่ละคำศัพท์จะเขียนบนหนึ่งบรรทัด คำศัพท์แต่ละชุดจะถูกแบ่งออกจากกันโดยเครื่องหมายจุด บรรทัดแรกของไฟล์จะต้องระบุว่าไฟล์อยู่ในภาพแบบ MLF ในหนึ่งไฟล์นี้จะบรรจุ Transcript ไว้ครบตามไฟล์เสียงที่เราต้องการจะใช้ในการ Train นอกจากนี้ HTK ยังอนุญาตให้มี Transcript แยกเป็นของตัวเองในแต่ละไฟล์ โดยไม่ต้องเก็บไว้ใน MLF ทั้งหมดก็ได้

ในแต่ละ Transcription file นั้น จำเป็นจะต้องตั้งชื่อให้ตรงกับชื่อไฟล์ .wav ที่ทำการอัด เพราะ HTK จะทำการค้นหา label file ที่มีชื่อตรงกับไฟล์เสียงนั้นเอง ยกตัวอย่างเช่น ถ้าไฟล์เสียง /one.wav กำลัง

ทำการแปลงค่าอยู่ HTK จะค้นหาไฟล์ /one.lab มาใช้โดยอัตโนมัติ เมื่อมีการนำ MLF มาใช้ HTK จะทำการค้นหา pattern ที่ตั้งไว้ตรงกับชื่อของไฟล์ .wav และนำเอา Transcript ที่พบมาใช้

หลังจากที่ MLF ในภาพแบบของ word level ได้ถูกสร้างขึ้น เราสามารถใช้โปรแกรม HLed ในการแปลง word level ให้มาอยู่ในภาพของ phone level ยกตัวอย่างเช่น ในกรณีที่ Transcript ข้างต้นใช้ชื่อว่า words.mlf เราสามารถทำการแปลง Transcript นี้ให้เป็น phone level โดยใช้คำสั่ง

HLEd –I * -d dict –i phones0.mlf mkphone0.led words.mlf

-I <path> คือการบอกให้เก็บ output ไว้ใน path ถ้าไม่มีก็สร้างขึ้นใหม่

-d <dictionary> บอกว่าให้ใช้ไฟล์ dictionary ที่ชื่อ dict ช่วยในการสร้าง transcription file

-I output เป็นการบอกชื่อของไฟล์ output ที่ได้

Mkphone0.led เป็น Script ไฟล์ที่สามารถกำหนดได้ดังนี้ สร้างได้โดยใช้ Text editor

EX

IS sil sil

DE sp

คำสั่ง EX เป็นการสั่งให้แทนที่แต่ละคำใน words.mlf ด้วยการสะกดของแต่ละคำที่ได้จากไฟล์ของ dict คำสั่ง IS เป็นการสั่งให้มีการแทรก silence model (sp) ลงไปทุกครั้งที่มีการเริ่มต้นและจบลงของแต่ละไฟล์เสียง คำสั่ง DE sp เป็นคำสั่งที่ให้มีการลบทุกๆ short-pause (sp) ที่ไม่เป็นที่ต้องการในตอนนี้ออกให้หมด

การเตรียมข้อมูลนั้นเป็นขั้นตอนที่มีความจำเป็นต้องทำด้วยความระมัดระวัง เพราะว่าในการทำการ Train คำนั้นเมื่อเพิ่มคำให้มากขึ้นก็จำเป็นต้องใช้ข้อมูลเสียงและ Transcription file มากขึ้นด้วยซึ่ง Transcription file ต้องตรงตามเสียงพูดที่ทำการบันทึกมาด้วยเพราะไม่เช่นนั้นจะทำให้โมเดลที่ได้ไม่มีความถูกต้องได้

บทที่ 6
ลักษณะทั่วไปของ HMMs

เมื่อทำการเตรียมข้อมูลเสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปก็จะเป็นการนำข้อมูลเหล่านั้นมาทำการสร้างเป็นโมเดลที่จะใช้จดจำคำศัพท์โดยใช้ HTK ซึ่งการสร้างโมเดลนั้นจะอยู่ในภาพแบบของ HMM หรือ Hidden Markov Models ซึ่งรายละเอียดต่างๆรวมถึงขั้นตอนในการสร้างโมเดลจะได้อธิบายไปเป็นลำดับขั้นตอนเพื่อให้สามารถเข้าใจได้ง่าย

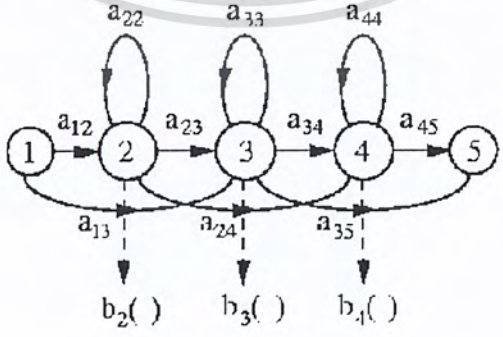
6.1 คำจำกัดความของไฟล์โมเดล (HMMs Definition Files)

ฟังก์ชันหนึ่งใน HTK ก็คือการถ่ายโอนข้อมูลให้อยู่ในภาพของ Hidden Markov Models (HMMs) คำศัพท์ที่ถูกบรรจุอยู่ใน HMMs จะเป็นตัวที่ทำให้โมเดลต่างๆมีลักษณะต่างกันออกไป ทั้งในส่วนของค่าที่ใช้ในการทำทรานสิชัน และค่าที่ใช้บอกถึง output ในแต่ละเวคเตอร์ของ HMM สามารถแบ่งออกเป็นข้อมูลย่อยๆได้หลายทิศทาง แต่ละทิศทางก็จะมีน้ำหนักในตัวของมันเอง HMM ยังสามารถรองรับได้ทั้งข้อมูลที่มีความหนาแน่นของข้อมูลแตกต่างกันไป และข้อมูลที่มีความหนาแน่นสม่ำเสมอ

เนื่องจากการที่จะต้องมีการรวบรวม HMMs ชนิดต่างๆเข้ามาอยู่ในเฟรมเวิร์กเดียว HTK จึงใช้ Formal language ในการสร้าง HMMs ขึ้นมา การทำงานทั่วไปของกระบวนการนี้ใน HTK ถูกจัดการโดย Library module HModel ซึ่งจะเป็นตัวจัดการการคำนวณค่าทั่วไปในการสร้าง HMMs รวมทั้งส่วนที่ใช้แปลงข้อมูลให้ออกมาอยู่ในภาพที่จะใช้ในการสร้าง HMMs อีกส่วนที่ใช้ในการจัดการเรื่องนี้คือ HUtil จะประกอบไปด้วยส่วนที่ใช้ในการสร้าง HMMs ขึ้นมา

6.2 HMM Parameters

HMM ประกอบไปด้วย state ต่างๆจำนวนหนึ่ง ในแต่ละ state j จะมีค่าความน่าจะเป็นของ observation ที่เกี่ยวข้อง $b_j(o_t)$ ซึ่งจะเป็นตัวบ่งบอกว่า ณ เวลา t ความน่าจะเป็นของ output ในขณะนั้นเป็นอะไร แต่ละคู่ของ state i และ j จะมีค่าความน่าจะเป็นในการเปลี่ยนแปลง state อยู่ (a_{ij}) ยกเว้นใน state แรกและ state สุดท้าย จะไม่มีตัวบ่งบอกความน่าจะเป็นของคำในขณะนั้น



ภาพที่ 6.1 Simple Left-Right HMM

ดังในภาพ 6.1 แสดงให้เห็น HMM แบบง่าย ซึ่งประกอบด้วย 5 state จะมี 3 state ที่มีความน่าจะเป็นของ output สำหรับ transition matrix ของภาพนี้ จะมี 5 แถว 5 หลัก โดยในแถวสุดท้ายจะมีค่าเป็น 0 เสมอ เพราะไม่มีการเปลี่ยนแปลงเกิดขึ้นใน state เพราะเป็น state สุดท้าย

ข้อมูลที่จำเป็นจะในการระบุลักษณะของ HMM หนึ่งๆ ประกอบด้วย

- จำนวนและความกว้างของแต่ละทิศทางการไหลของข้อมูล
- Optional model duration parameter vector
- จำนวน state
- Transition matrix
- สำหรับแต่ละ state และแต่ละทิศทางการไหลของข้อมูล
    - ค่าความน่าจะเป็นของ output
    - ค่าความน่าจะเป็นในการเปลี่ยนแปลง state

6.3 Basic HMM Definition

ในเครื่องมือของ HTK บางตัว จำเป็นจะต้องมี HMM ในการใช้งาน ยกตัวอย่างเช่น ในเครื่องมือ HRest จะต้องมีการเกี่ยวข้องกับ HMM ดังตัวอย่าง

HRest hmmdef s1 s2 s3.....

ซึ่งเป็นการสั่งการทำงานโดยมีการใช้งานไฟล์ hmmdef ซึ่งเป็นไฟล์ HMM

ไฟล์ HMM ประกอบไปด้วยกลุ่มของสัญลักษณ์ที่แสดงแทนตัวอักษรต่างๆในภาษาปกติ สัญลักษณ์เหล่านี้เป็นกุญแจสำคัญในการสื่อความหมาย โดยสามารถอยู่ได้ทั้งในภาพตัวอักษรหรือตัวเลข

```
~h "hmm1"
<BeginHMM>
    <VecSize> 4 <MFCC>
    <NumStates> 5
    <State> 2
        <Mean> 4
            0.2 0.1 0.1 0.9
        <Variance> 4
            1.0 1.0 1.0 1.0
    <State> 3
        <Mean> 4
            0.4 0.9 0.2 0.1
        <Variance> 4
            1.0 2.0 2.0 0.5
    <State> 4
        <Mean> 4
            1.2 3.1 0.5 0.9
        <Variance> 4
            5.0 5.0 5.0 5.0
    <TransP> 5
        0.0 0.5 0.5 0.0 0.0
        0.0 0.4 0.4 0.2 0.0
        0.0 0.0 0.6 0.4 0.0
        0.0 0.0 0.0 0.7 0.3
        0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

ภาพที่ 6.2 Definition for Sample Left-Right HMM

ในภาพที่ 6.2 แสดงให้เห็นไฟล์ HMM ที่ใช้แทน Simple left-right HMM ในภาพ 6.1 ในไฟล์ประกอบไปด้วย 5 state ซึ่งมี 3 state ที่มีการเคลื่อนย้ายระหว่าง state ได้ สัญลักษณ์ ~h ที่แสดงในบรรทัดแรกบ่งบอกว่าไฟล์นี้มีข้อมูลที่เป็น macro ชนิด h หรือการบ่งบอกว่าไฟล์นี้เป็นไฟล์ HMM โดยในตัวอย่าง HMM นี้ จะถูกเรียกว่า "hmm1" ข้อควรจำ ชื่อของ HMM ห้ามกำหนดเป็นหมายเลข สำหรับตัวของ HMM เองนั้นจะเริ่มไฟล์ด้วยสัญลักษณ์ <BeginHMM> และลงท้ายไฟล์ว่า <EndHMM>

ในบรรทัดแรกในส่วนของข้อมูลต่างๆ จะเป็นส่วนที่ใช้แสดงว่า HMMs มีลักษณะสำคัญอย่างไร เช่นในตัวอย่าง บ่งบอกว่าไฟล์นี้ Observation vectors ประกอบไปด้วย 4 component (<Vecsize> 4) และมีลักษณะเป็น MFCCs

บรรทัดต่อมาแสดงจำนวนของ state ทั้งหมดใน HMMs ซึ่งจะต่อด้วยส่วนที่ใช้บ่งบอกคำศัพท์ โดยเริ่มต้นจากการแสดงว่า state นี้เป็น state ที่เท่าไหร่ (<State>) ตามด้วย mean vector ใน state นั้น (<Mean>) และตามด้วยส่วนที่ใช้แสดง diagnol vectors (<Variance>) เมื่อครบทุก state แล้วจะปิดท้ายไฟล์ด้วย transition matrix (<TransP>)โดยค่า TransP ในแต่ละแถวจะต้องรวมกันได้เท่ากับหนึ่ง ยกเว้นในแถวสุดท้ายที่จะต้องเท่ากับ 0

```
~h "hmm2"
<BeginHMM>
    <VecSize> 4 <MFCC>
    <NumStates> 4
    <State> 2 <NumMixes> 2
        <Mixture> 1 0.4
            <Mean> 4
                0.3 0.2 0.2 1.0
            <Variance> 4
                1.0 1.0 1.0 1.0
        <Mixture> 2 0.6
            <Mean> 4
                0.1 0.0 0.0 0.8
            <Variance> 4
                1.0 1.0 1.0 1.0
    <State> 3 <NumMixes> 2
        <Mixture> 1 0.7
            <Mean> 4
                0.1 0.2 0.6 1.4
            <Variance> 4
                1.0 1.0 1.0 1.0
        <Mixture> 2 0.3
            <Mean> 4
                2.1 0.0 1.0 1.8
            <Variance> 4
                1.0 1.0 1.0 1.0
    <TransP> 4
        0.0 1.0 0.0 0.0
        0.0 0.5 0.5 0.0
        0.0 0.0 0.6 0.4
        0.0 0.0 0.0 0.0
<EndHMM>
```

ภาพที่ 6.3 Simple Mixture Guassian HMM

ภาพ 6.3 แสดงให้เห็นถึง HMM ที่ใน state ที่แต่ละ state ถูกแยกเป็นคู่ลำดับ จำนวนของคู่ลำดับบ่งบอกโดยสัญลักษณ์ <NumMixes> โดยคู่ลำดับแต่ละตัวจะมี prefix แสดงโดยสัญลักษณ์ <Mixture> ตามด้วยหมายเลขของคู่ลำดับ และ component weight ข้อควรจำคือ ไม่จำเป็นที่จำนวนของคู่ลำดับจะต้องเท่ากันในทุก state

```
~o <VecSize> 4 <MFCC>
    <StreamInfo> 2 3 1
~h "hmm4"
<BeginHMM>
    <NumStates> 4
    <State> 2
        <SWeights> 2 0.9 1.1
        <Stream> 1
            <Mean> 3
                0.2 0.1 0.1
            <Variance> 3
                1.0 1.0 1.0
        <Stream> 2
            <Mean> 1 0.0
            <Variance> 1 4.0
    <State> 3
        <Stream> 1
            <Mean> 3
                0.3 0.2 0.0
            <Variance> 3
                1.0 1.0 1.0
        <Stream> 2
            <Mean> 1 0.5
            <Variance> 1 3.0
    <TransP> 4
        0.0 1.0 0.0 0.0
        0.0 0.6 0.4 0.0
        0.0 0.0 0.4 0.6
        0.0 0.0 0.0 0.0
<EndHMM>
```

ภาพที่ 6.4 HMM ที่มีข้อมูล 2 Stream

ในภาพ 6.4 แสดงให้เห็นถึง HMM ที่มีการไหลของข้อมูลเป็น 2 ทิศทาง ขั้นแรกเราจะต้องใช้สัญลักษณ์ ~o เพื่อกำหนดลักษณะสำคัญทั้งหมดของ HMM เป็นอย่างไร ในตัวอย่าง บ่งบอกว่าไฟล์นี้ Observation vectors ประกอบไปด้วย 4 component (<Vecsize> 4) และมีลักษณะเป็น MFCCs หลังจากนั้นตามด้วยสัญลักษณ์ <StreamInfo> ซึ่งจะตามด้วยจำนวนของทิศทางการไหลของข้อมูลและตามด้วยจำนวน component ในแต่ละทิศทางการไหล ดังตัวอย่างหมายถึง HMM นี้มีทิศทางการไหล 2 ทาง ทางหนึ่งมี 3 component และอีกทางมี 1 component

ในแต่ละ state การไหลของข้อมูลจะถูกแบ่งเป็น 2 ทิศทาง แต่ละทิศทางจะใช้สัญลักษณ์ <Stream> ในการแบ่ง ใน state ที่สองมีการระบุ <Sweight> ไว้ด้วย นั่นคือในทิศทางแรกมี weight เท่ากับ

1.1 ขณะที่ทิศทางที่สองมี weight เท่ากับ 0.9 ถ้าไม่มีการกำหนด weight ไว้เหมือนใน state ที่ 3 ก็ตีความได้ว่าแต่ละทิศทางมี weight = 1.0

```
~o <VecSize> 4 <MFCC>

~v "var"
    <Variance> 4
        1.0 1.0 1.0 1.0
```

ภาพที่ 6.5 ~v macro ไฟล์

```
~h "hmm5"
<BeginHMM>
    <NumStates> 4
    <State> 2 <NumMixes> 2
        <Mixture> 1 0.4
            <Mean> 4
                0.3 0.2 0.2 1.0
            ~v "var"
        <Mixture> 2 0.6
            <Mean> 4
                0.1 0.0 0.0 0.8
            ~v "var"
    <State> 3 <NumMixes> 2
        <Mixture> 1 0.7
            <Mean> 4
                0.1 0.2 0.6 1.4
            ~v "var"
        <Mixture> 2 0.3
            <Mean> 4
                2.1 0.0 1.0 1.8
            ~v "var"
    <TransP> 4
        0.0 1.0 0.0 0.0
        0.0 0.5 0.5 0.0
        0.0 0.0 0.6 0.4
        0.0 0.0 0.0 0.0
<EndHMM>
```

ภาพที่ 6.6 HMM โดยใช้ ~v macro

6.4 Macro Definition

ใน HMM นอกจากที่เราสามารถใช้ macro ~h และ ~o แล้ว ยังมี macro ที่สามารถนำมาใช้งานได้อีก ยกตัวอย่างเช่น ~v ใช้ประโยชน์คล้ายกับการประกาศตัวแปรที่ใช้แทน diagonal variance vector รวมกัน ดังภาพที่ 6.5 ~v "var" ใช้แทนสัญลักษณ์ใน 2 บรรทัดต่อมา เมื่อเรานำไฟล์ในภาพ 6.4 มาและนำ macro ~v มาใช้งาน จะได้ผลลัพธ์ดังภาพ 6.6 นอกจากนี้ยังมี macro ชนิดอื่นๆ ความหมายของ macro แต่ละตัวมี ดังนี้

~s เป็นการใช้งาน State ร่วมกัน

~m เป็นการใช้งาน Component แบบร่วมกัน

~u เป็นการใช้งาน Mean vector ร่วมกัน

~v เป็นการ Diagonal variance vector ร่วมกัน

~t เป็นการใช้งาน Transition matrix ร่วมกัน

~w เป็นการใช้งาน Stream weight vector ร่วมกัน

นอกจากนี้ยังมี macro อีกสี่ชนิดที่ใช้งานต่างออกไปคือ

~l ระบุว่าไฟล์นี้มีลักษณะทางตรรกะแบบ HMM

~h ระบุว่าไฟล์นี้มีลักษณะทางกายภาพเป็น HMM

~o ใช้ประกาศลักษณะสำคัญของ HMM

~p Tied mixture (ใช้ในโปรแกรม HHEd ในการสร้าง mixture system)

~r Regression class tree

```
~o      <VecSize> 4 <MFCC>
~s "stateA"
    <Mean> 4
        0.2 0.1 0.1 0.9
    <Variance> 4
        1.0 1.0 1.0 1.0
~s "stateB"
    <Mean> 4
        0.4 0.9 0.2 0.1
    <Variance> 4
        1.0 2.0 2.0 0.5
~s "stateC"
    <Mean> 4
        1.2 3.1 0.5 0.9
    <Variance> 4
        5.0 5.0 5.0 5.0
~t "tran"
    <TransP> 5
        0.0 0.5 0.5 0.0 0.0
        0.0 0.4 0.4 0.2 0.0
        0.0 0.0 0.6 0.4 0.0
        0.0 0.0 0.0 0.7 0.3
        0.0 0.0 0.0 0.0 0.0
```

ภาพที่ 6.7 การใช้ State ร่วมกัน และ Transition Matrix Macros

6.5 HMM Sets

ในเครื่องมือของ HTK ต้องการกลุ่มของ HMM ที่สมบูรณ์มากกว่า HMM เดี่ยวๆ เมื่อเป็นเช่นนี้ ทำให้เราต้องทำการเชื่อมความสัมพันธ์ระหว่าง HMM ของหน่วยเสียงต่างๆเข้าไว้เป็นไฟล์เดียวกันซึ่งทำได้โดยใช้โปรแกรม HERest โดยพิมพ์คำสั่งตาม Syntax ดังนี้

HERest .... –H mf1 –H mf2 ... hlist

ด้วยคำสั่งนี้ คำสั่งหลัง –H จะเป็นชื่อของ macro file ชื่อของ HMM ต่างๆจะถูกบรรจุในไฟล์ที่ตามหลัง ~h อยู่ โดยในไฟล์นั้นๆชื่อจะถูกบันทึกไว้หนึ่งชื่อต่อหนึ่งบรรทัด ดังตัวอย่าง

ha

hb

hc

```
~h "ha"
<BeginHMM>
    <NumStates> 5
    <State> 2
        ~s "stateA"
    <State> 3
        ~s "stateB"
    <State> 4
        ~s "stateB"
    ~t "tran"
<EndHMM>

~h "hb"
<BeginHMM>
    <NumStates> 5
    <State> 2
        ~s "stateB"
    <State> 3
        ~s "stateA"
    <State> 4
        ~s "stateC"
    ~t "tran"
<EndHMM>

~h "hc"
<BeginHMM>
    <NumStates> 5
    <State> 2
        ~s "stateC"
    <State> 3
        ~s "stateC"
    <State> 4
        ~s "stateB"
    ~t "tran"
<EndHMM>
```

ภาพที่ 6.8 Simple Tied State System

ในภาพ 6.7 และ 6.8 เป็นตัวอย่างโดยที่ไฟล์แรกที่ใช้เก็บค่าที่ทุก HMM ใช้ร่วมกันได้ไว้ ไฟล์ที่สองจะเก็บลักษณะของ HMM ไว้ เซ็ตของ HMM ลักษณะนี้เรียกอีกอย่างว่า Tied-state system

วิธีในการสร้าง hlist ทำได้โดยการรวมไฟล์ HMM ไว้ในไดเรกทอรีใดๆ หลังจากนั้นใช้คำสั่ง

HERest –d hdir …. hlist ….

โปรแกรมจำทำการสร้าง hlist จากไดเรกทอรี hdir ให้โดยอัตโนมัติ

6.6 HMM Definition Language

อธิบายโดยใช้สัญลักษณ์แบบ BNF | ใช้แบ่งระหว่างทางเลือก, ( ) ใช้แยกตัวแปรออกจากกัน, [ ] ใช้แยกออปชั่น, { } ใช้แทนการซ้ำกัน

```
Hmmdef =        [~h macro]
                <BeginHMM>
                    [globleOpt]
                    <Numstates>short
                    state { state }
                    [regTree]
                    tranP
                    [duration]
                <EndHMM>
oprmacro =      ~o globalOpts
globleOpts =    option {option}
option = <HmmSetId> string |
                <StreamInfo> short {short} |
                <VecSize> short |
                covkind |
                durkind |
                parmkind
covkind =       <DiagC> | <InvDiagC> | <FullC> | <LLTC> | <XformC>
durkind =       <nullD> | <poissonD> | <gammaD> | <genD>
parmkind =      <basekind{_D|_A|_E|_N|_Z|_O|_V|_C|_K}>
basekind =      <discrete> | <Ipc> | <Ipcepstra> | <mfcc> | <fbank> |
                <melspec> | <Iprefc> | <Ipdelcep> | <user>
state =         <State : Exp> short stateinfo
stateinfo =     ~s macro |
                [mixes] [weight] stream {stream} [duration]
macro = string
```

```
mixes =          <NumMixes> short {short}
weights =        ~w macro | <Sweights> short vector
vector = float {float}
stream =         [<Stream>short]
                 (mixture {mixture} | tmixpdf | discdf)
mixture =        [<Stream> short float ] mixpdf
tmixdf =         <Tmix> macro weightList
weightList =     repShort {repShort}
repShort =       short [ * char ]
discpdf =        <Dprob> weightList
mixpdf =         ~m macro | [rclass] mean cov [ <Gconst> float ]
rclass =         <Rclass>
mean =           ~u macro | <Mean> short vector
cov =            var | inv | xform
var =            ~v macro | <Varience> short vector
inv =            ~I macro | (<InvCovar> | <LLTCovar> ) short tmatrix
xform = ~x macro | <Xform> short short matrix
matrix =         float {float}
tmatrix =        matrix
duration =       ~d macro | <Duration> short vector
regTree =        ~r macro tree
tree =           <RegTree> short nodes
nodes = (<Node> short short short | <Tnode> short int) [nodes]
tranP =          ~t macro | <TransP> short matrix
                 ---------------------------------
```

ใน HTK นั้นสามารถจัดการกับไฟล์ทั้งในลักษณะของ Text file และ Binary file ซึ่งส่วนใหญ่แล้วการทำงานในลักษณะของ Text file จะมีความสะดวกกว่าและสามารถที่จะมองความเป็นไปของโมเดลได้ซึ่งสามารถเปิดดูได้ด้วยโปรแกรม Text editor ทั่วไป แต่การใช้ Binary นั้นจะเป็นการประหยัดพื้นที่ในการเก็บและการอ่านเขียนจะทำได้เร็วกว่า Text file ซึ่งถ้าขั้นตอนไหนไม่จำเป็นต้องเปิดไฟล์ขึ้นมาดูก็สามารถที่จะใช้ภาพแบบ Binary file จะช่วยประหยัดเวลาได้

บทที่ 7
การสร้างโมเดล HMM

ในการทำการรู้จำนั้นขั้นตอนการสร้างโมเดลเป็นขั้นตอนที่สำคัญที่สุด ในการสร้างโมเดลที่ดีก็เท่ากับว่าเราได้โมเดลที่จะทำให้การรู้จำของเราเป็นไปอย่างถูกต้องมากที่สุด ซึ่งขั้นตอนการสร้างโมเดลแบบพื้นฐานนั้นเริ่มจากการสร้างโมเดลของ Monophones หรือโมเดลของหน่วยเสียงย่อยในภาษาไทย จากนั้นจะเป็นการสร้าง Triphones ซึ่งเกิดจากการนำ Monophones มาเชื่อมต่อกันซึ่งจะได้อธิบายรายละเอียดทีละส่วนไป

7.1 การสร้างโมเดลของโมโนโฟน (Monophones Training)

Monophones คือหน่วยเสียงเดี่ยวๆของเสียงที่เราต้องการจะสร้างซึ่งในปริญญานิพนธ์เล่มนี้จะหมายถึงหน่วยเสียงย่อยของเสียงในภาษาไทย ซึ่งได้อธิบายมาแล้วในบทที่ผ่านมาซึ่ง Monophones จะเป็นโมเดลเริ่มต้นในการสร้างโมเดลของคำต่างๆซึ่งได้จากการประกอบกันของ Monophones นั่นเอง

7.1.1 Creating flat start monophones

ขั้นตอนแรกในการทำ HMM Training เป็นการสร้าง prototype ขึ้นมาค่าใน prototype นี้จะไม่มีความสำคัญ มันมีเพื่อใช้เป็นภาพแบบเบื้องต้นให้กับ model โครงสร้างของ model ที่เหมาะสมกับ phoneme based system จะมีลักษณะเป็น 3 state right-left ที่ไม่มีการโดดข้าม state โดยมีตัวอย่างดังนี้

```
~o <VecSize> 39 <MFCC_D_A_0><StreamInfo>1 39
~h "proto"
<BeginHMM>
<NumStates>5
<State>2<NumMixes>1
<Stream>1
<Mixture>1 1.0000
        <Mean>39
                0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.00.0 0.0 0.0 0.0
        <Variance>39
                1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State>3 <NumMixes>1
```

<Stream>1

<Mixture>1 1.0000

<Mean>39

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.00.0 0.0 0.0 0.0

<Variance>39

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<State>4 <NumMixes>1

<Stream>1

<Mixture>1 1.0000

<Mean>39

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.00.0 0.0 0.0 0.0

<Variance>39

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<TransP>5

0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

0.000000e+000 7.000000e-001 4.000000e-001 0.000000e+000 0.000000e+000

0.000000e+000 0.000000e+000 7.000000e-001 4.000000e-001 0.000000e+000

0.000000e+000 0.000000e+000 0.000000e+000 7.000000e-001 4.000000e-001

0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

<EndHMM>

ในแต่ละ vector จะมีความกว้างเท่ากับ 39 ค่าที่จะนำมาใช้คำนวณได้จากนำ length of parameterised static vector + delta coefficient + acceleration coefficients

เครื่องมือใน HTK ชื่อ HCompV จะช่วยในการค้าหาเซ็ตของไฟล์ข้อมูล Feature vector ของเสียงพูด และนำข้อมูลทั้งหมดมาคำนวณหาค่า mean และ variance ซึ่งผลลัพธ์ที่ได้จากการคำนวณจะถูกนำไปใช้เป็นค่าเริ่มต้นให้กับแต่ละ HMM สมมติว่าชื่อไฟล์ Feature vector ของเสียงพูดถูก list ไว้ในไฟล์ชื่อ Train.scp ใช้คำสั่งดังนี้

HCompV -C config -f 0.01 -m -S Train.scp -M hmm0 proto

โดยที่ :

#configuration ไฟล์ (config) จะมีข้อมูลดังนี้

SOURCEKIND=MFCC_D_A_0

SOURCEFORMAT=HTK

SOURCERATE=100000.0

-f 0.01 ให้มีการสร้างไฟล์ที่ชื่อว่า Variance Floor Macro (vFloor)

-m ให้มีการคำนวณค่า mean

-M hmm0 ให้ Output ที่ได้เก็บใน Directory ชื่อ hmm0 ที่เราได้สร้างไว้แล้วก่อนหน้านี้

proto เป็นชื่อ ไฟล์ prototype ที่เราได้สร้างขึ้นก่อนหน้านี้

คำสั่ง HCompV นี้จะทำให้เราได้ prototype ตัวใหม่ขึ้นมาในไดเรกทอรี hmm0 โดยที่เครื่องหมายเลข 0 นั้นหมายถึงการที่ค่า variance ต่างๆ ถูกแทนที่โดย global means และ global variance โดย prototype ตัวใหม่นี้จะมีชื่อว่า proto และ vFloor เก็บไว้ในไดเรกทอรี hmm0 โดยไฟล์ที่ได้จะลักษณะเป็น Master Macro File (MMF) ไฟล์หนึ่งจะเก็บ monophone ที่จำเป็นไว้ โดยที่ HMMs จะถูกสร้างจากการนำเอา monophone ในส่วนนี้มาประกอบเข้าด้วยกัน อีกไฟล์จะเก็บค่า variance macro เอาไว้ โดย vFloor จะมีจำนวนขึ้นอยู่กับจำนวนของทิศทางการไหลของข้อมูล

จากนั้นจะทำการนำเอาค่า mean และ variance ที่ได้จาก ไฟล์ ทั้งสองนี้มาสร้างเป็น phoneme ของเสียงในภาษาไทยซึ่งประกอบไปด้วยเสียงพยัญชนะ 20 เสียง สระ 18 เสียง เสียงลงท้ายอีก 8 เสียง ซึ่งไม่มีเครื่องมือใน HTK ที่ช่วยสร้างให้ ต้องสร้างเองหรือไม่ก็ใช้ program ชื่อ mkphones.class ซึ่งเป็นโปรแกรมที่เขียนขึ้นเองด้วยภาษา JAVA ภาพแบบของคำสั่งมีดังนี้

c:\project\java mkphones ↵

ผลลัพธ์ที่ได้จะได้ไฟล์ชื่อว่า hmmdefs และ macros เก็บไว้ใน directory ที่ชื่อ hmm1 ซึ่งต้องสร้างขึ้นก่อนหน้านี้ (ควรจะสร้าง Directory hmm0 – hmm16 ไว้เลยเพราะ program ทุกตัวในการ Train จะไม่สร้างให้)

7.1.2 เครื่องมือ HCompV

โปรแกรมจะทำการคำนวณค่า global mean และ covariance ของเซ็ตของข้อมูลที่จะนำมาทำการ Train โดยปกติมักจะเป็นค่าที่เกี่ยวข้องกับค่าต่างๆใน HMM ซึ่ง component ต่างๆสามารถใช้ค่า mean และ covariance ร่วมกันได้ ซึ่งเป็นกระบวนการขั้นแรกในการทำ flat start Training ซึ่ง model ทั้งหมดจะถูกแปลงมาให้ใช้ค่าต่างๆร่วมกัน โดยที่ค่า covariance จะถูกใช้เป็นค่าพื้นฐานของ Fixed Variance และ Grand Variance

เมื่อมีการ Train เซ็ตของ model ที่มีขนาดใหญ่มาก แต่มีข้อมูลอยู่จำกัด มีความจำเป็นอย่างมากที่จะต้องรวบรวมค่า variance ที่มาจากข้อมูลที่มีอยู่จำนวนน้อยนั้นเข้าด้วยกัน โดยการสร้าง variance macro ขึ้นมาเรียกว่า varFloorN โดยค่า N คือหมายเลขของทิศทางการไหล ซึ่ง HCompV มีความสามารถที่จะสร้าง variances macro ขึ้นมาได้

คำสั่งพื้นฐานของ HCompV เป็นดังนี้

HCompV [options] hmm TrainFiles ...

โดยที่ option ต่างๆมีความหมายดังนี้

-f m เป็นการสั่งให้โปรแกรมสร้างไฟล์ Variance Floor Macro (vFloor) ขึ้นมา โดยที่ค่า global variance มีค่าเท่ากับ m

-l s เป็นการสั่งให้โปรแกรมนำเฉพาะไฟล์ข้อมูลที่มีเฉพาะไฟล์ที่มี string s นำหน้ามาทำการ Train

-m เป็นการสั่งให้โปรแกรมทำการคำนวณ mean สำหรับทุก HMM component โดยค่า mean นั้นได้มาจากการคำนวณจากข้อมูลที่นำมา Train

-o s คือการสั่งให้ output เก็บในชื่อ s ในไดเรกทอรีเดียวกับข้อมูลที่นำมา Train

-v f เป็นการเลือกค่า variance ให้มีค่าต่ำสุดไม่เกิน f (ค่าปกติ=0.0)

-B ทำ output ให้อยู่ในภาพไฟล์ HMM แบบเลขฐานสอง (Binary file)

-F fmt บอกโปรแกรมว่าข้อมูลต้นมีภาพแบบเป็น fmt

-G fmt บอกโปรแกรมว่า label file มีภาพแบบเป็น fmt

-H mmf เป็นการนำ HMM macro model ที่คำนวณไว้แล้วมาคำนวณ ใช้เมื่อมีการคำนวณ MFF หลายตัว

-I mlf เป็นคำสั่งโหลด mlf มาใช้ ใช้เมื่อต้องการคำนวณ MLF หลายตัว

-L dir เป็นการหาข้อมูลต้นจากไดเรกทอรี L

-M dir ให้ Output ที่ได้เก็บในไดเรกทอรี ชื่อ dir ที่เราได้สร้างไว้แล้ว

7.1.3 Re-estimating model

การทำ Re-estimate แสดงดังภาพ 7.1 กระบวนการจะเป็นการนำเอา HMM ที่นำมาสร้างเป็น phoneme แล้ว มาเข้ากระบวนการ Baum-Welch re-estimation โดยเป็นการหาความเป็นไปได้ในแต่ละ state ในแต่ละช่วงเวลาโดยใช้หลักการ forward/backward algorithm ความเป็นไปได้ที่ได้มาจะนำมาใช้คำนวณหาค่า weight เฉลี่ยของค่าต่างๆใน HMM

การทำ Re-estimate จะใช้ข้อมูลจาก feature vector ของเสียงพูดในการทำ re-estimate ทุกครั้ง ซึ่งจะถูกเก็บเป็นร เยการในไฟล์ชื่อ Train.scp ก่อนจะทำการ Re-estimate ต้องสร้าง ไฟล์ ชื่อ monophones0 ขึ้นมาก่อนโดย ไฟล์นี้จะเก็บรายการของ phoneme ของเสียงในภาษาไทยทั้งหมดไว้ และทำการ re-estimate โดยใช้เครื่องมือของ HTK ชื่อ HERest ดังนี้

HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 –S Train.scp –H hmm1\macros -H hmm1\hmmdefs –M hmm2 monophones0

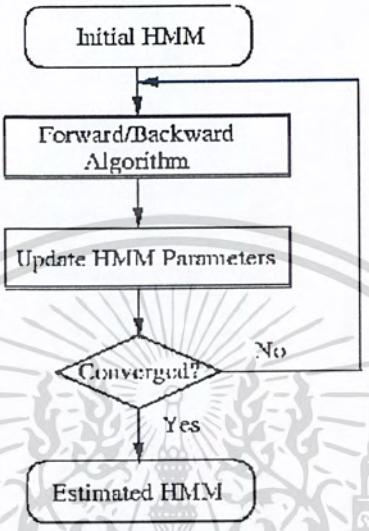
ไฟล์ที่ชื่อ phones0.mlf เป็นไฟล์ที่ได้จากการเตรียมข้อมูล

-H hmm1\macros –H hmm1\hmmdefs หมายถึงให้ใช้ model ชื่อ macro และ hmmdefs ที่อยู่ใน directory hmm1 มาทำการ re-estimate

-M hmm2 เป็น directory ที่ใช้เก็บไฟล์ Output ที่ชื่อเดียวกับไฟล์ Input คือ macros, hmmdefs

-t 250.0 150.0 1000.0 เป็นค่าที่ตั้งไว้เพื่อป้องกันการทำ re-estimate ผิดพลาด

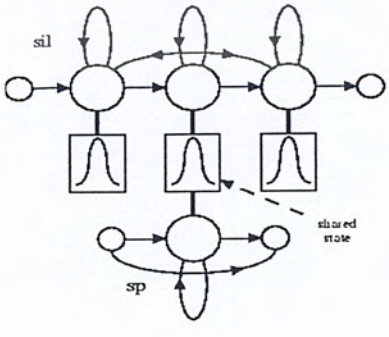
ทำการ Re-estimate อีกครั้งหลังจากเสร็จสิ้นขั้นตอนนี้จะได้ hmm3 ออกมา



ภาพที่ 7.1 ขั้นตอนในการทำงานของเครื่องมือ HERest

7.1.4 Fixing silence model

ในขั้นตอนที่ผ่านมาทั้งหมดจะเป็นการสร้าง 3 state left-to-right HMM สำหรับแต่ละ phoneme ซึ่งเมื่อทำการใช้งาน model จริง ๆ จะนำแต่ละ phoneme มาต่อกันให้เป็นคำซึ่งในบางครั้งในการพูดก็มีการหยุดหรือเว้นช่วงคำในการพูดไม่เหมือนกันแม้ว่าจะเป็นคำคำเดียวกัน ดังนั้นจึงต้องมีการปรับปรุง model ใหม่โดยจะใส่ shot-pause(sp) เข้าไปใน model รวมไปถึงการเพิ่ม transition เข้าไปใน state เพื่อจุดประสงค์ในการให้เกิดการวน loop ก่อนที่จะเกิดการย้าย state เร็วเกินไป และช่วยในการลดทอนสัญญาณรบกวนต่างๆ ได้อีกด้วย



ภาพที่ 7.2 Silence Model

Shot-pause model นั้นจะมีลักษณะคล้ายๆกับอักษร T กล่าวคือ จะมีเพียง 1 state ที่อยู่ระหว่าง state แรกและ state สุดท้าย และมีทิศทางการไหลเพียงทิศทางเดียว ตัวอย่างของ shot-pause model แสดงอยู่ในภาพที่ 7.2

วิธีการปรับปรุง model ของ Monophones นั้นแบ่งได้เป็น 2 ขั้นตอนดังนี้

1. สร้าง shot-pause model : สร้างจากโปรแกรมที่เขียนขึ้นเองชื่อว่า Mksp.class ซึ่งจะรับ Input เป็น model จาก directory hmm3 และจะให้ output ที่มี shot-pause model รวมอยู่ใน model ใหม่ใน directory hmm4 ภาพแบบของคำสั่งดังนี้

   c:\project\java Mksp

2. สร้าง transition เพิ่มใน model โดยใช้เครื่องมือของ HTK ชื่อ HHEd ภาพแบบของคำสั่งดังนี้

   HHEd -H hmm4\macros -H hmm4\hmmdefs -M hmm5 sil.hed monophones1

โดยที่ sil.hed คือ text ไฟล์ ที่บรรจุคำสั่งในการสร้าง transision ดังนี้

```
AT 2 4 0.2 { sil.transP }
AT 4 2 0.3 { sil.transP }
AT 1 3 0.3 { sp.transP }
TI silst { sil.state[3],sp.state[2] }
```

คำสั่ง AT เป็นคำสั่งในการเพิ่มการย้าย State ให้กับ transition matrix ตัวอย่าง บรรทัดแรกของคำสั่งข้างต้นเป็นการให้มีการเพิ่มการย้าย state จาก state 2 ไป state 4 โดยให้ความน่าจะเป็นเท่ากับ 0.2 ลงไปใน Transition matrix ของ model ชื่อ sil และคำสั่ง TI จะเป็นการสร้าง tied-state ที่เรียกว่า silst โดยจะนำเอา state ที่3 ของ sil และ state ที่ 2 ของ sp มาสร้าง

Monophones1 คือ monophones0 ที่ได้ทำการเพิ่ม sp ต่อท้าย

หลังจากนั้นก็ทำการ Re-estimate 2 ครั้ง โดยจะใช้ phone transcription ที่มี sp model ระหว่างคำ ซึ่งเป็น transcription ไฟล์ แบบที่ 2 ที่ได้กล่าวไปแล้วข้างต้นซึ่งจะสร้างได้ด้วย HLEd เช่นเดียวกับ ขั้นตอนการแปลง transcript ให้เป็น phone level แต่ต้องแก้ mkphone0.led โดยตัดบรรทัดสุดท้ายออก (DE sp) จากนั้น save เป็น mkphone1.led ทำการสร้าง phones1.mlf โดยใช้คำสั่ง

HLEd –l * -d dictPlus –I phones1.mlf mkphone1.led words.mlf

เมื่อได้ไฟล์ phone1.mlf นำไฟล์ phones1.mlf มาใช้ในการทำ Re-estimate อีก 2 ครั้งจะได้ hmm7 ออกมา

HERest -C config -I phones1.mlf –t 250.0 150.0 1000.0 –S Train.scp –H hmm5\macros -H hmm5\hmmdefs –M hmm7 monophones1

7.1.5 เครื่องมือ HHEd

มีลักษณะเป็น script ที่ใช้งานเป็น editor สำหรับ HMM โดยจะทำการโหลดข้อมูลในเซ็ตของ HMM มา และได้ output ออกมาในภาพของ transformation set โดยส่วนมากแล้ว HHEd ใช้กับงานแปลง คัดลอกค่าใน HMM ให้ออกมาในภาพแบบที่ต้องการ นอกจากนี้ยังสามารถใช้ในการโคลน HMMs และเปลี่ยนแปลงข้อมูลใน HMMs ได้อีกด้วย

คำสั่งพื้นฐานในการเรียกใช้ HHEd เป็นดังนี้

HHEd [option] edCmdFile hmmList

ในส่วนของการสั่งการ จะมีไฟล์คอยควบคุมการทำงานของ HHed ไว้ (edCmdFile) ภาพแบบของคำสั่งที่สำคัญใน edComdFile มีดังนี้

AT i j prob itemlist(t)

- จะเป็นการสั่งให้ทำการ transition จาก state i ไปยัง j โดยมีค่าความน่าจะเป็นเท่ากับ prob ลงไปในทุก Transition matrix ที่ระบุไว้ใน itemlist

CL hmmList

- เป็นการสั่งให้ทำการโคลน HMM ที่อยู่ใน hmmList โดยใน hmmList จะต้องมี HMM ที่มีชื่อตรงกับ HMM ที่โหลดอยู่ในขณะนั้น

CO newList

- เป็นคำสั่งให้สร้างรายชื่อของ HMM ที่ทำการโหลดอยู่ในขณะนั้นใหม่

SW s n

- เปลี่ยนค่า stream s ในทุก HMM ที่โหลดอยู่ในขณะนั้น ไปเป็นค่า n

TI macro itemlist

- เป็นการสร้าง macro โดยนำเอาข้อมูลจาก itemlist โดย output ที่ได้ออกมา ขึ้นอยู่กับข้อมูลที่ใส่เข้าไป

ในส่วนของ option มี option ต่างๆที่สำคัญดังต่อไปนี้

- d dir สั่งให้ HHEd ค้นหาในไดเรกทอรี dir เพื่อหา HMM ที่ต้องใช้
- w mmf สั่งจัดเก็บข้อมูล macro และ model ทั้งหมด ที่โหลดไว้ใน MMF ตัว เดียว
- H mmf สั่งให้โหลด HMM macro model ชื่อ mmf มาใช้
- M mmf เก็บ output ที่ได้ในไดเรกทอรี dir ถ้าไม่มีการตั้งตัวนี้ไว้ โปรแกรมจะเขียนทับลงไปใน model เดิม

7.2 การสร้างโมเดลของไตรโฟน (Triphones Training)

Triphones เป็นโมเดลที่ได้จากการนำเอา Monophones มาเชื่อมต่อกันซึ่งการเชื่อมต่อกันของ Monophones นั้นจะขึ้นอยู่กับคำและประโยคว่าประกอบไปด้วยโมเดลของ Monophones อะไรบ้างซึ่งจะสร้างไว้ทุกๆ Triphones ที่พบในประโยคที่นำมาทำการ Train ซึ่งจะได้โมเดลใหม่ขึ้นมาและจะถูกเรียกใช้เมื่อมีคำที่ประกอบไปด้วยหน่วยเสียงที่ตรงกับ Triphones เหล่านั้น

7.2.1 การสร้าง Triphones จาก Monophones

หลังจากเราได้เซ็ตของ monophone HMMs แล้ว ในขั้นตอนสุดท้ายของการสร้างแบบจำลองก็คือการสร้าง Context-dependent triphone HMMs ขั้นตอนแรกในการสร้าง Context-dependent นั้นสามารถสร้างได้จากการ Cloning จาก monophone หลังจากนั้นก็จะทำการ Re-estimate โดยใช้ Triphone transcription เริ่มต้นโดยการใช้ HLEd

7.2.2 สร้าง Triphones Transcription

สร้างได้จากเครื่องมือชื่อ HHEd โดยพิมพ์คำสั่งดังนี้

```
HLEd -n triphones1 -l * -I wintri.mlf mktri.led phones1.mlf
```

โดยที่ไฟล์ชื่อ triphones1 และ wintri.mlf จะเป็น transcription และ triphone สำหรับการ Train ครั้งต่อไป wintri.mlf จะแทน phones1.mlf และ triphones1 จะแทน monophones1

mktri.led เป็น script คำสั่งที่ช่วยในการสร้าง triphone ประกอบด้วยคำสั่งดังนี้

```
WB sp
WB sil
TC
```

ผลลัพธ์ที่ได้จากการทำคำสั่งข้างต้นก็คือจะได้ ไฟล์ชื่อ triphones1 และ wintri.mlf

7.2.3 Tied Transition Matrix

เป็นขั้นตอนที่ทำให้ HMM มากกว่าหนึ่งตัวใช้ transition matrix ร่วมกันเพื่อประหยัดเนื้อที่ในการเก็บ model โดยใช้คำสั่งดังนี้

```
HHEd -B -H hmm7\macros -H hmm7\hmmdefs -M hmm8 mktri.hed monophones1
```

โดยที่ : -B บอกว่าไฟล์ Output ที่ได้ให้อยู่ในภาพ Binary ไฟล์ เพื่อการประหยัดเนื้อที่

mktri.hed เป็น script คำสั่งที่ใช้ในการ Tied transition matrix เข้าด้วยกัน ตัวอย่างคร่าวๆ ดังนี้

```
CL triphones1
TI T_ah { (*-ah+*, ah+*,*-ah ).transP }
TI T_ax { (*-ax+*, ax+*,*-ax ).transP }
TI T_ah { (*-ey+*, ey+*,*-ey ).transP }
....
```

เนื่องจากการทำ Tied นั้นจะมีผลต่อประสิทธิภาพ หากทำโดยไม่มีหลักการ ดังนั้นสิ่งที่สำคัญคือการ Tied เฉพาะ parameter ที่มีผลกระทบน้อยที่สุดต่อการแยกแยะความแตกต่างของเสียง เช่น transition matrix บาง Triphones นั้นมีโอกาสเกิดน้อยมาก ทำให้การ Re-estimate จะให้ค่าที่ไม่ดีนักหากไม่ทำการ Tied ก่อน

การสร้าง Context-depenent model ขึ้นมาจากการลอกแบบเซ็ตของ Triphones จะต้องนำไปทำ Re-estimate อีกครั้งโดยใช้ HERest ดังนี้

HERest -C config -I wintri.mlf -t 250.0 150.0 1000.0 –s stats –S Train.scp –H hmm8\macros -H hmm8\hmmdefs –M hmm9 triphones1

ทำการ Re-estimate ซ้ำอีกครั้งจะได้ hmm10 ซึ่งจะได้โมเดลของ Triphones ออกมา

การสร้าง Triphones มีความจำเป็นมากเพราะว่าโมเดล Monophones อย่างเดียวนั้นไม่ได้เก็บเสียงเชื่อมระหว่างหน่วยเสียง ซึ่งเวลาพูดจริงๆตามธรรมชาติแต่ละคำแต่ละหน่วยเสียงจะมีเสียงช่วงต่อกันของหน่วยเสียงซึ่งต้องทำเป็น Triphones เพื่อเก็บโมเดลของเสียงเหล่านั้น จึงทำให้ความถูกต้องของการสร้างโมเดลมีความถูกต้องเพิ่มมากขึ้น ตลอดจนการทำการ Tied Transition Matrix ซึ่งก็เป็นสิ่งที่สำคัญเพราะถ้าคำเพิ่มมากขึ้นแล้วขนาดของโมเดลจะเพิ่มมากขึ้นซึ่งทำให้โมเดลมีขนาดใหญ่ซึ่งมีผลต่อความเร็วในการนำโมเดลนี้ไปใช้งาน

บทที่ 8
การปรับปรุงโมเดล

หลังจากที่ได้ทำการสร้างโมเดล Triphones แล้วก็สามารถทำให้ความถูกต้องของโมเดลเพิ่มขึ้นได้ในระดับหนึ่ง แต่ความถูกต้องสามารถเพิ่มขึ้นได้อีกโดยการปรับปรุงโมเดลให้ดีขึ้นโดยอาศัยเทคนิคต่างๆเข้ามาช่วย ซึ่งที่จะพูดถึงต่อไปในบทนี้นั้นเป็นวิธีการปรับปรุงโมเดลในรูปแบบต่างๆซึ่งสามารถที่จะเลือกใช้วิธีที่เหมาะสมมาใช้ในการทำการรู้จำ

8.1 Discrete and Tied – Mixture Model

เป็นการนำข้อมูลที่ไม่ต่อเนื่อง เช่น component ต่างๆมา share เพื่อใช้ประโยชน์ร่วมกัน โดยจะมี weight ของข้อมูล discrete ต่างๆเป็นตัวบ่งชี้ ซึ่งการแบ่ง discrete system ออกมานี้มีข้อได้เปรียบที่ความรวดเร็วและความยืดหยุ่นในการคำนวน หลังจากที่นำ Mixture model ต่างๆ มา tied เข้าด้วยกัน ด้วยเครื่องมือ HHEd โดยใช้คำสั่ง TI

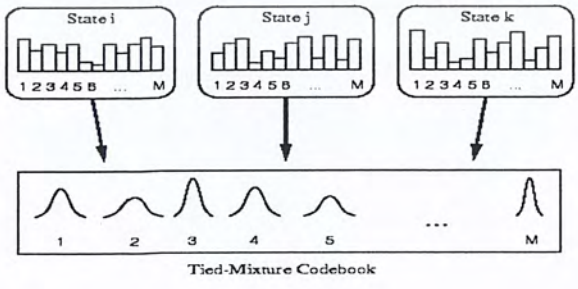
8.1.1 Making Tied-state Triphones

หลังจากที่ได้ Triphones มาเรียบร้อยแล้ว จะนำแต่ละ states ของ triphones มาทำการ tied เพื่อเป็นการใช้ข้อมูลของ states นั้นๆร่วมกัน และเพื่อความรวดเร็วในการคำนวณ ด้วยเครื่องมือ HHEd โดยใช้คำสั่ง TI ที่เป็นคำสั่งสำหรับการ tied ทุกสมาชิกใน transition matrix เข้าด้วยกัน และเราจะต้องทำการเลือกว่าจะ tied ที่ states ใดเพื่อประสิทธิภาพสูงสุด

เครื่องมือ HHEd นี้มีรูปแบบในการ tied states ที่มีประสิทธิภาพอยู่สองแบบ คือ แบบ data-driven และแบบ decision tree ซึ่งรายละเอียดจะกล่าวถึงในหัวข้อต่อไป

8.1.2 Tied-Mixture system

เป็นการนำ Gaussian components มารวบรวมเอาไว้เพื่อประโยชน์ในการใช้ข้อมูลร่วมกัน ซึ่งในแต่ละ states จะมีรายละเอียดเกี่ยวกับ components ต่างๆ ที่มี weight เป็นตัวบ่งชี้ และเมื่อมีการรวบรวม components ต่างๆ บรรจุไว้ใน code book และจะดูว่า มี component ใดบ้างที่เหมือนกัน ดังนั้นใน pool จึงไม่มี component ที่ซ้ำกัน และแต่ละ component จะมีข้อมูลของ feature vector อยู่



ภาพที่ 8.1 Tied-mixture system

ขั้นตอนการ tied mixture system

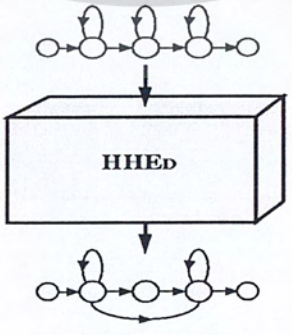
- เลือกขนาดของ codebook ในแต่ละ data stream และเลือก Gaussian component จาก initial set ของ monophones เพื่อที่จะบรรจุลง codebook อย่างเหมาะสม
- ทำการ Train initial set ของmonophones
- ใช้ HHEd เพื่อแยก data stream ออกมา เท่าจำนวนที่ต้องการและทำการ tied แต่ละ stream เข้าด้วยกัน แล้วจึงทำการ convert โดย TIEDHS ดังตัวอย่างสคริปต์ที่ทำกับ 4stream ดังนี้

```
SS 4
JO 258 2.0
TI st1 {*.state[2-4].stream[1].mix}
JO 128 2.0
TI st2 {*.state[2-4].stream[2].mix}
JO 128 2.0
TI st3 {*.state[2-4].stream[3].mix}
JO 84 2.0
TI st4 {*.state[2-4].stream[4].mix}
HK TIEDHS
```

- ทำการ Re-estimate โดยใช้เครื่องมือ HERest ตามปกติ

8.2 HMM System Refinement

ในการใช้งานจริงนั้นต้องการการรู้จำที่มีประสิทธิภาพ และความแม่นยำสูง จึงต้องการข้อมูลเสียงเป็นจำนวนมาก ซึ่งมีปัญหาตามมาอีกคือปัญหาของข้อจำกัดในการ Train ข้อมูลที่มากมายมหาศาล ดังนั้นจึงต้องมีวิธีการปรับปรุงโมเดลที่มีอยู่ให้มีประสิทธิภาพให้มากที่สุด



ภาพที่ 8.2 การปรับปรุงโมเดลโดยเครื่องมือ HHEd

อีกวิธีหนึ่งที่จะเพิ่มประสิทธิภาพของระบบคือการปรับปรุง Context independent monophones ซึ่งระบบสามารถเจาะจงหน่วยเสียงนี้ออกเป็น states ต่างๆได้ โดยที่เราจะใช้เครื่องมือ HHEd มาปรับปรุงโมเดลและทำการ Re-estimate โดย HERest ซึ่งการใช้คำสั่งดังกล่าวจะทำการ tied ชุดข้อมูลต่างๆ ให้มีการใช้ข้อมูลร่วมกัน และหลังจากใช้เครื่องมือ HHEd แล้ว จะสร้าง macro ใหม่ๆขึ้นมาเสมอ ซึ่งการทำงานของ HHEd กับโมเดลนั้นมีหลายประเภทตามขั้นตอนต่างๆ ดังต่อไปนี้

8.2.1 Construct Context-Dependent Model

ในขั้นตอนนี้เราต้องสร้าง Context dependent model ขึ้นมาก่อน จาก independent model เพื่อที่จะสร้างโมเดลให้อยู่ในรูปแบบ l+p-r ซึ่ง p เป็นหน่วยเสียงที่เราจะต้องใช้ ส่วน l และ r เป็นหน่วยเสียงที่อยู่ทาง ซ้าย และขวา ของหน่วยเสียง p ซึ่งเราจะต้องสร้างรายชื่อของ HMM เรียกว่า cdlist ที่บรรจุ Context dependent model ที่เราต้องการ และใช้ HHEd โดยใช้คำสั่งดังนี้

CL cdlist

ผลที่ได้คือ แต่ละโมเดลในรายชื่อ จะทำการคัดลอกหน่วยเสียง p ออกมาตามที่เราต้องการ และจะต้องทำการ re-estimate ด้วยเครื่องมือ HERest

ก่อนการสร้าง Context-dependent model เราต้องทราบว่าเราใช้ cross-word triphones หรือไม่ ถ้าใช้ เราไม่ต้องสนใจกับ word boundaries ก็สามารถเปลี่ยน monophone labels เป็น triphones ได้เลย อย่างไรก็ตาม หากมีการใช้บางส่วนของ phones ใน triphones จะมีการปรับแต่ง word boundaries ใน transcription จะต้องใช้ HLED ด้วยคำสั่ง WB เพื่อให้ คำสั่ง TC สามารถใช้ biphones หรือ triphones ใน word boundaries ได้

การใช้ HTK นี้สามารถเลือกว่าเราจะจัดการกับข้อมูลในรูปแบบ Binary หรือแบบ Text ได้ซึ่งมีข้อแตกต่างกันไม่มาก ตรงที่ข้อมูลในรูปแบบ binary นั้นมีขนาดเล็กและทำการอ่าน และบันทึกได้รวดเร็วกว่า text แต่ข้อมูลในรูปแบบ text นั้นได้เปรียบที่ เราสามารถอ่านและทำความเข้าใจได้ โดยเครื่องมือ HHEd นั้นมีพารามิเตอร์ –B สำหรับความต้องการ output ที่เป็นแบบ Binary ซึ่งมีความจำเป็นมาก หากข้อมูลอยู่ในรูปแบบ text ที่มีขนาดใหญ่ จะทำการบันทึก และอ่านได้ช้ากว่า

8.2.2 Parameter tying and Item List

เป็นการ tied โดยใช้ Item list ซึ่งจะเป็น list ของ โมเดลที่ต้องการ tied เข้าด้วยกันเพื่อประโยชน์ในการ Estimate ที่ไม่ซ้ำซ้อน ไม่สิ้นเปลืองเวลา ซึ่งการ tied สามารถทำได้โดยใช้เครื่องมือ HHEd โดยใช้คำสั่ง TI ซึ่งมีรูปแบบดังนี้

TI macroname itemlist

หลังจากรันคำสั่งนี้จะเป็นผลให้ ทุก model ที่อยู่ใน item list จะถูก tied เข้าด้วยกัน และได้ output เป็น macro ชื่อ macroname ซึ่ง item list เราสามารถสร้างเองได้จาก editors ทั่วไป

8.2.3 Data-Driven Clustering

การทำ clustering นั้นเหมือนเป็นการ share ข้อมูลของหน่วยเสียงร่วมกันเพื่อเพิ่มความถูกต้องของโมเดล โดยที่โมเดลใดใช้ข้อมูลเสียงเหมือนกันในส่วนหนึ่งๆ ก็จะทำการสุ่มหน่วยเสียงที่มีอยู่มาประกอบกันขึ้นมาเป็นโมเดล จะได้โมเดลเสียงใหม่ ซึ่งสามารถเพิ่มความถูกต้องในการรู้จำได้

HHEd ได้มีกลไกที่ถูกสร้างขึ้นมาเพื่อให้สามารถทำให้ states รวมกันเป็นกลุ่ม แล้วนำกลุ่มนั้นๆมาเชื่อมกัน กลไกนี้เรียกว่า data-driven

จากขั้นตอนที่ผ่านมาเป็นการสร้าง triphones โดยทำการลอกแบบจากทุก monophones แล้วทำการ re-estimation ที่ใช้ข้อมูล transcription ต่างๆเป็น triphones แทนที่จะเป็น monophones ซึ่งผลที่ได้คือเซ็ทของแบบจำลองที่มีขนาดใหญ่ และมีข้อมูลที่ใช้ Train น้อยเมื่อเทียบกันแต่ละแบบจำลองที่เกิดขึ้น เพราะ triphones ก็คือการเรียงกันในลักษณะเฉพาะของ monophones โดยการใช้เหตุผลที่ว่า states ตรงกลางใน triphones models นั้นจะไม่ได้รับผลกระทบมากนักจากสิ่งที่อยู่ข้างหน้า และสิ่งที่ตามมาข้างหลัง ทางหนึ่งที่สามารถลดขนาดของ parameters โดยไม่เป็นการลดความสามารถของแบบจำลองเท่าใด คือการเชื่อมทุกๆ states ตรงกลางที่เหมือนกัน โดยเชื่อมกันทุกๆแบบจำลองที่ได้มาจาก monophones เดียวกัน ซึ่งในการเชื่อมต่อกันนั้นสามารถทำได้โดยใช้คำสั่ง TI โดยเขียนลงบน script ได้ดังต่อไปนี้

```
TI "iys3" {*-iy+*.state[3]}
TI "ihs3" {*-iy+*.state[3]}
TI "ehs3" {*-iy+*.state[3]}
```

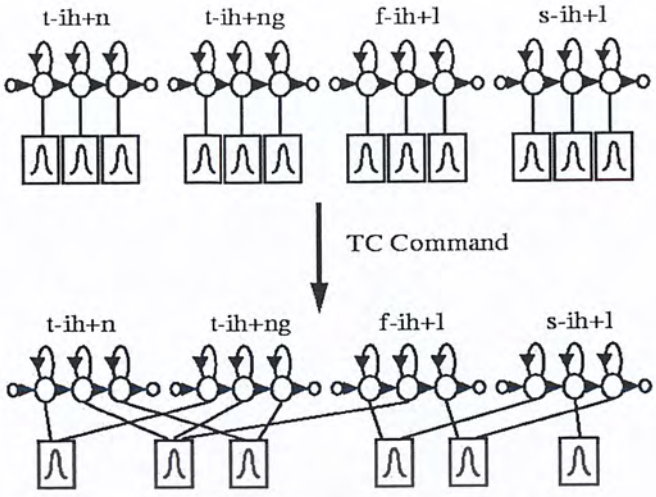
คำสั่ง TI แต่ละคำสั่งนั้นจะทำการเชื่อมต่อทุกๆ states ตัวกลางของ triphones ในแต่ละกลุ่มของ phone และถ้าหากว่าในที่นี้มีจำนวน triphones เฉลี่ย 100 triphones ต่อกลุ่มของ phones หนึ่งกลุ่ม จะทำให้ขนาดของ states ต่อกลุ่มจะถูกลดขนาดลงจาก 300 เป็น 201

การเชื่อมต่อกันเช่นนี้ ถึงแม้ว่าจะได้ผลที่ดี แต่โดยรวมทั้งหมดก็ยังไม่เป็นที่น่าพอใจอยู่ดี เพราะว่าปัญหาของการ Train ของ states ซ้ายและขวาก็ยังมีอยู่ ซึ่งทางที่ดีกว่านี้ก็คือใช้การรวมกลุ่มเป็นตัวตัดสินว่า states ใดบ้างที่จะต้องมีการเชื่อมต่อ

data-driven clustering สามารถทำได้โดยใช้คำสั่ง TC ซึ่งคำสั่งจะอ้างอิงถึง top-down hierachical procedure เหมือนกัน เริ่มต้นทุก states จะถูกวางไว้ในกลุ่มแบบใดแบบหนึ่ง และหลังจากนั้นจะมีการจับคู่กันในกลุ่มซึ่งเมื่อถูกนำมารวมกันจะถูกสร้างเป็นกลุ่มใหม่ขึ้น ซึ่งกระบวนการนี้จะทำซ้ำไปเรื่อยๆจนกระทั่งขนาดของกลุ่มที่ใหญ่กว่านั้นถึงค่า threshold ที่ตั้งไว้โดยคำสั่ง TC

ในตัวอย่างต่อไปนี้เป็น script ของ HHEd จะทำการ clusters และ tied สำหรับ states ที่เกี่ยวข้องกันในกลุ่มของ triphones สำหรับ phones ชื่อ ih

```
TC 100.0 "ihS2" {*-ih+*.state[2]}
TC 100.0 "ihS2" {*-ih+*.state[3]}
TC 100.0 "ihS2" {*-ih+*.state[4]}
```



รูปที่ 8.3 ผลจากการทำ Data Clustering

ในตัวอย่างนี้ แต่ละคำสั่ง TC นั้นจะทำการตัดกลุ่มเซ็ทของ states ที่ทำการกำหนดไว้ ซึ่งแต่ละการตัดกลุ่มนั้นจะเป็นการเชื่อม และให้ output ตาม macro ที่ให้ไว้ และชื่อ macro นั้นจะเป็นอะไร จะขึ้นอยู่กับชื่อที่ระบุไว้ในคำสั่ง ซึ่งผลที่ได้จากการทำคำสั่งนี้นั้นแสดงให้เห็นตามรูป ซึ่งถ้าหากว่ารู้ว่า word-internal triphone system จะถูกสร้างขึ้นก็ควรที่จะทำการใส่ biphones ลงไปด้วยซึ่งก็เหมือนกับที่ทำกับ triphones

ปัญหาของการทำคำสั่ง TC ก็คือ การที่จะได้ผลลัพธ์ที่เป็นผลลัพธ์เดี่ยวๆที่ไม่เกิดประโยชน์จากการเชื่อม ซึ่งทำให้เกิดการไม่มีข้อมูลที่เพียงพอสำหรับการทำการ Train ทางแก้อย่างหนึ่งสำหรับปัญหานี้ก็คือการใช้คำสั่ง RO เพื่อที่จะทำการกำจัดผลลัพธ์ที่ไม่ต้องการนี้

RO Tresh "statsfile"

ซึ่ง statsfile ในที่นี้ก็คือชื่อของ statistic file output ซึ่งได้จากการใช้ –s option ของการทำคำสั่ง HERest ซึ่งเป็นตัวที่เก็บ occupation counts ของทุกๆ states เซ็ทของ HMM ที่จะทำการ Train โดย occupation count นั้นจะเป็นจำนวนของเฟรมที่ถูกจองโดยแต่ละ states และสามารถใช้เป็นตัววัดได้ว่าแต่ละ state มีข้อมูลที่จะใช้ทำการ Train เท่าไร ที่สามารถใช้ได้สำหรับการทำการ Estimate ของ state นั้น โดย RO จะต้องถูกประมวลผลก่อนคำสั่ง TC โดยผลก็คือจะนำข้อมูล statistic จากการอ่านจากไฟล์เพื่อไปทำการตั้งค่า flag ให้แก่คำสั่ง TC เพื่อทำการกำจัดสิ่งที่ไม่ต้องการให้เกิด ที่ยังหลงเหลือจากการทำการตัดและเชื่อมคำตามปกติในขั้นตอนก่อนหน้านี้ ซึ่งจะสามารถทำได้โดยทำการหาการตัดเสียงที่มีค่า occupation counts ที่น้อยที่สุดแล้วทำการรวมกับตัวที่อยู่ใกล้ๆ โดยจะทำซ้ำไปเรื่อยๆจนกระทั่งค่าผลรวมของ occupation counts ทั้งหมดนั้นเกินว่า thresh ซึ่งเป็นการทำให้แน่ใจได้ว่าทุกๆ cluster ของ state จะสามารถนำไป Train ได้อย่างเหมาะสมโดยใช้ HERest ต่อไป

* ค่า Thresh นั้นเป็นการบอกว่าจะต้องเกิด Occupation Count เท่าใดจึงจะไม่ถูกกำจัด

จากการทำการ clustering และ tying ตามข้างต้นนั้น มีแบบจำลองมากมายที่ได้ผลที่ถือได้ว่าเหมือนกันเพราะว่า triphones ที่มีความเหมือนเป็นอย่างเดียวกันทุกๆ emitting state ของ triphones นั้นๆ

จะมีการใช้ร่วมกันก็จะมีการจัดให้อยู่ในประเภทของการทำการ cluster ที่เป็นกลุ่มเดียวกัน ซึ่งเรียกว่า generallised triphones และ ประโยชน์ของการทำการ Tying นั้นยังเป็นการลดการให้ความหมายแบบจำลอง หรือเป็นการลดจำนวนแบบจำลองอีกด้วย นอกจากนั้น HHEd ได้มีคำสั่งที่จะทำให้การเรียกข้อมูลขึ้น คือคำสั่ง CO

Co newlist

จะเป็นการทำให้เซ็ทของ HMM ที่ loaded มามีความกะทัดรัด โดยการบ่งชี้แบบจำลองที่ถือได้ว่าเหมือนกันและทำการ tying แบบจำลองเหล่านั้นเป็นแบบจำลองใหม่ ซึ่งจะให้ output เป็นไฟล์ที่ชื่อ newlist อย่างไรก็ตาม HMMs ที่จะ tied ได้จะต้องมีความเหมือนกันในทุกส่วนซึ่งเป็นอีกเหตุผลหนึ่งที่ทำให้ต้องมีการใช้ Transition Matrice ร่วมกันระหว่าง triphones เพราะ ไม่เช่นนั้นแล้วจะเกิดความแตกต่างกันเล็กน้อยระหว่าง transition matrices

โดยในการทำ Data Clustering นั้นจะมีการใช้คำสั่งดังนี้

HHEd –t 1 –B –H hmm10\macros –H hmm10\hmmdefs –M hmm11 dclust.hed triphones1

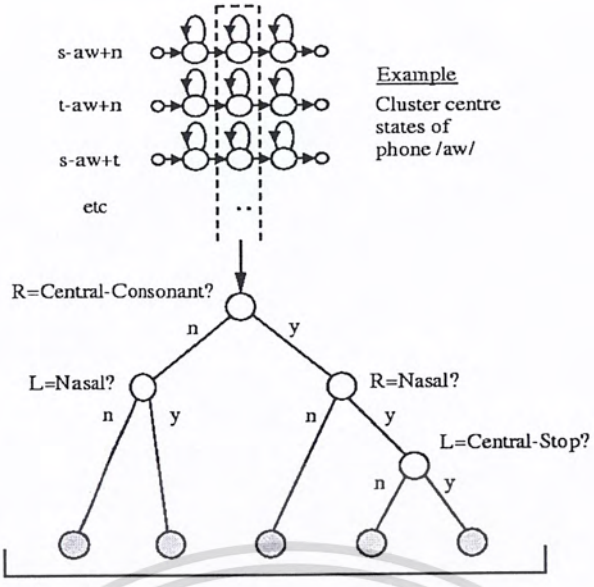
โดยไฟล์ที่ชื่อ dclust.hed นั้นจะเป็นไฟล์ที่ประกอบไปด้วย

```
RO 2 stats
TC 100.0 "aS2" {(*-a,a+*,*-a+*).State[2]}
TC 100.0 "aS2" {(*-a,a+*,*-a+*).State[3]}
TC 100.0 "aS2" {(*-a,a+*,*-a+*).State[4]}
...
CO newlist
```

เมื่อทำการ cluster เสร็จแล้วจำเป็นต้องทำการ re-estimation อีกสองครั้งเพื่อให้แบบจำลองมีประสิทธิภาพมากขึ้นโดยการทำการ re-estimation นั้นจะเปลี่ยนจากรายการที่เคยใช้ที่เป็น triphones1 จะเปลี่ยนเป็นการใช้ list ที่ได้จากคำสั่ง CO จาก HHEd จากการอธิบายข้างต้นนั้นได้ใช้ชื่อที่เป็น newlist

8.2.4 Tree- Based Clustering

จากขั้นตอนที่ผ่านมานั้นไม่ได้แก้ปัญหาเกี่ยวกับ triphones ซึ่งสามารถหลีกเลี่ยงได้โดยการวางแผนการ Train ที่ดี แต่เมื่อจำนวนคำศัพท์เพิ่มมากขึ้น ปัญหาเรื่อง triphones ที่ตกค้าง (ไม่ได้รับการ tied กับ triphones อื่นๆ) จึงเป็นสิ่งที่หลีกเลี่ยงไม่ได้ ซึ่ง HTK ได้เห็นปัญหาดังกล่าว และมีวิธีแก้ไขโดยเครื่องมือ HHEd โดยใช้คำสั่ง TB ซึ่งคำสั่งนี้จะทำการสร้าง Binary tree ซึ่งในแต่ละโนดจะมีเงื่อนไขให้แต่ละ model เป็น yes หรือ no ดังภาพที่ 8.4 เมื่อแต่ละ phones state ใน list เริ่มต้นจะอยู่ที่ node root และเมื่อผ่าน decision ในแต่ละ node จะจบลงที่ terminal node (แรเงา) และ phone ที่จบลงที่ node เดียวกันจะถูก tied เข้าด้วยกัน ดังนั้นจึงขจัดปัญหาดังกล่าวได้



ภาพที่ 8.4 การ tied โดยใช้ decision tree-based

8.2.5 Mixture Incrementing

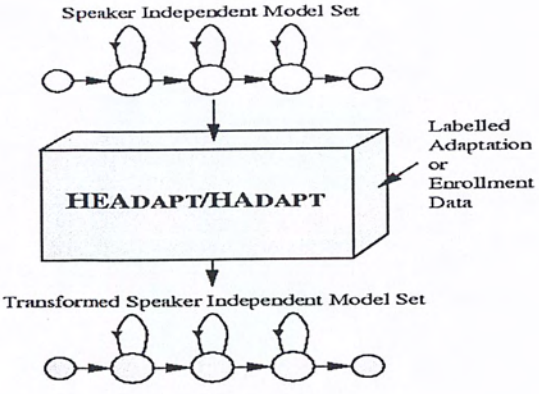
เมื่อเราสร้างระบบการรู้จำแบบ sub-word นั้น ในขั้นสุดท้ายจะมี context-dependent HMM อยู่มากมาย ซึ่งเป็น mixture component แต่ขั้นตอนก่อนหน้านี้สามารถสร้างได้เพียง single gaussian model เท่านั้น หากเราต้องการแปลง model แบบ single gaussian ไปเป็น multiple mixture component จำเป็นต้องใช้เครื่องมือ HHEd โดยใช้คำสั่ง MU ซึ่งจะเพิ่มจำนวน mixture component โดยจะแบ่ง single model ที่มีอยู่เป็นส่วนย่อยๆ ตามที่เราต้องการ จะเป็นผลให้ระบบมีความยืดหยุ่นและสามารถเพิ่มประสิทธิภาพได้ ซึ่งคำสั่ง MU มีรูปแบบดังนี้

MU n itemList

ซึ่ง n หมายถึงจำนวนของ mixture component ที่ต้องการ และ itemlist คือ mixture component ที่เราต้องการปรับแต่ง

8.3 Adapting the HMMs

การที่จะทำการรู้จำเสียงให้ได้ประสิทธิภาพ สามารถใช้งานกับคนหลายๆคนได้นั้นขึ้นอยู่กับเสียงที่นำมา Train นั้นต้องมาจากบุคคลหลายๆคน หากข้อมูลมาจากบุคคลเดียงคนเดียว การทำการรู้จำก็จะยังไม่สมบูรณ์ เพราะไม่สามารถตีความหมายเสียงพูดจากคนอื่นได้ จึงมีการนำเทคนิค Adaptation เข้ามาเกี่ยวข้อง เพื่อปรับปรุงโมเดลที่มีอยู่ให้มีประสิทธิภาพมากขึ้น โดยใช้ข้อมูลเสียงของคนคนเดียวเป็นหลัก และใช้ข้อมูลเสียงจากผู้อื่นเข้ามาเสริมเพียงเล็กน้อยเท่านั้น



ภาพที่ 8.5 Adaptation of HMM by HEAdapt

ในส่วนของ HTK นั้นสนับสนุนเทคนิคการ Adaptation สองแบบ คือแบบที่ใช้ HEAdapt และแบบที่ใช้ HVite โดยที่หากตัว transcription ของข้อมูลที่เราต้องการปรับปรุงนั้นเป็นที่รู้จักของระบบแล้ว เรียกว่าการ Adaptation แบบ supervised เราจะใช้เครื่องมือ HEAdapt และ หากข้อมูลเสียงยังเป็นแบบ unlabelled หรือ ยังไม่มี transcription เราจะเรียกว่า unsupervised adaptation จะใช้เครื่องมือ Hvite เพื่อที่จะสร้าง transcription และปรับปรุงโมเดลไปด้วยในตัว

การปรับปรุงโมเดลโดยวิธีต่างใน HTK ได้กำหนดวิธีการที่จะแก้ไขปัญหาต่างๆในการ Train ได้เป็นอย่างดีซึ่งจุดมุ่งหมายอยู่ที่การทำให้โมเดลมีความถูกต้อง มีความกระทัดรัด และโมเดลมีความยืดหยุ่นมากขึ้น ซึ่งมีวิธีที่หลากหลายในการปรับปรุงโมเดลโดยสามารถที่นำวิธีการที่ได้กล่าวมาทั้งหมดนี้มาช่วยให้การปรับปรุงโมเดลของเราให้มีประสิทธิภาพมากขึ้นได้เป็นอย่างดี

บทที่ 9
การสร้างโมเดลไวยากรณ์ของภาษา

หลังจากที่ได้ทำการสร้างและปรับปรุงโมเดลแล้ว ต่อไปในบทนี้จะได้อธิบายถึงวิธีการหรือเทคนิคอีกชนิดหนึ่งซึ่งนิยมใช้กันอย่างกว้างขวาง ซึ่งสามารถเพิ่มความถูกต้องให้กับการนำโมเดลได้ ตลอดทั้งบทนี้จะได้อธิบายถึงวิธีการสร้างไวยากรณ์ในภาพแบบต่างๆที่มีอยู่ใน HTK ซึ่งสามารถนำมาใช้งานจริงกับโมเดลการรู้จำได้เป็นอย่างดี

9.1 การสร้างไวยากรณ์ของภาษา

9.1.1 Word Network

ในส่วนนี้จะอธิบายถึงรายละเอียดของ word network ซึ่งเป็นการจัดลำดับของคำว่าควรเป็นไปอย่างไร ที่สามารถทำการรู้จำและนำมาสร้างเป็นโมเดลได้ word network นั้นจะเป็นการนำเสนอในภาพแบบของ Task Grammar ที่มีการกำหนดคำในตำแหน่งที่ถูกต้องตามลำดับของคำ หรือว่าจะเป็นส่วนของ word loop ซึ่งจะเป็นการวางคำแบบง่ายๆทุกๆคำให้วนซ้ำใน loop เดียวกันซึ่งจะสามารถให้คำทุกๆคำต่อท้ายคำอื่นๆได้ network มีความจำเป็นต้องสร้างขึ้นเพื่อทำการกำหนดและบังคำความเป็นไปของคำในการวิเคราะห์เสียงพูด

ใน HTK นั้นมีวิธีการสร้าง network ซึ่งจะอยู่ในภาพของ Standard Lattices Format (SLF) ซึ่งจะอธิบายในภาคผนวกซึ่งภาพแบบทั่วไปจะอยู่ในภาพของไฟล์ Text ที่ถูกใช้ในการนำเสนอข้อสมมติหลายๆอย่างในเอาท์พุทของ recognizer ซึ่งข้อดีของการที่ SLF อยู่ในภาพของ Text file นั้นคือสามารถทำการแก้ไขหรือเปิดดูได้โดยใช้ Text editor ทั่วไปได้ อย่างไรก็ตามมันก็น่าเบื่อและชักช้า ซึ่ง HTK มีเครื่องมือที่จะช่วยสร้าง SLF จากข้อมูลที่ได้ออกแบบโดยมีโครงสร้างง่ายๆ เครื่องมืออันแรกคือ HParse ซึ่งจะสามารถสร้าง network จาก text ไฟล์ที่ถูกสร้างขึ้นในภาพแบบไวยากรณ์ของ BNF ภาพแบบนี้ถูกนำมาใช้ใน HTK ตั้งแต่เวอร์ชัน 2.2 ขึ้นไป และ HPAarse ยังสามารถที่จะกำหนดภาพแบบของเวอร์ชันหลังๆได้

9.1.2 การสร้าง Word Network ด้วยเครื่องมือ HPArse

ถึงแม้ว่าการสร้าง Word Network ในภาพแบบของ SLF Network ขึ้นมาเองด้วยมือจะไม่ยาก แต่ก็ค่อนข้างจะน่าเบื่อไม่น้อย ในเวอร์ชันปัจจุบันของ HTK นั้นจะมีการใช้วิธีการเขียนไวยากรณ์ในระดับสูงขึ้นมาเพื่อที่จะทำให้การเขียนไวยากรณ์เป็นไปได้อย่างสะดวกและง่ายไม่น่าเบื่อ ซึ่งจะอยู่บนโครงสร้างไวยากรณ์ของ Extended Backus-Naur Form (EBNF) ที่ได้ถูกใช้ในการกำหนดไวยากรณ์ระดับสูงให้กับการรู้จำ

ภาพแบบไวยากรณ์นี้จะเป็นภาพแบบที่เครื่องมือใน HTK ที่ชื่อ HPArse อ่านเข้ามาใช้ได้โดยตรง ซึ่งจะถูกแปลงให้ไปอยู่ในภาพของ Finite State เพื่อนำไปใช้ในการทำการรู้จำตอน Run-time ไวยากรณ์ของ HPArse นั้นมีภาพแบบที่ประกอบไปด้วยส่วนเพิ่มเติมเข้ามาจากไวยากรณ์ของ Regular Expression สามารถสร้างขึ้นจากลำดับของคำและอักขระพิเศษดังนี้

	denote Alternatives
[ ]	enclose options
{ }	denote zero or more repetition
< >	denote one or more repetition
<< >>	denote context-sensitive loop

ตัวอย่างต่อไปจะเป็นการแสดงให้เห็นถึงการใช้ทุกๆอักขระพิเศษยกเว้นตัวสุดท้ายที่เป็นอักขระเพื่อใช้ในจุดประสงค์พิเศษ ซึ่งปกติจะพบในการสร้าง Context-dependence phone loop และ ไวยากรณ์แบบ Word-Pair

ตัวอย่างแรกเป็นการสร้างไวยากรณ์แบบคำเดียวสำหรับการรู้จำตัวเลข 0-9 จะมี Syntax ดังนี้

(one | two | three | four | five | six | seven | eight | nine | zero)

ซึ่งในตัวอย่างนี้มองให้อยู่ในภาพ Network ได้ดังภาพ 9.1 (a) ถ้า Definition นี้ถูกเก็บไว้ในไวยากรณ์ชื่อ digitsyn และจะทำการสร้าง Word Network ซึ่งเป็นเอาท์พุทชื่อ digitnet จะพิมพ์คำสั่งดังนี้

HPArse digitsyn digitnet

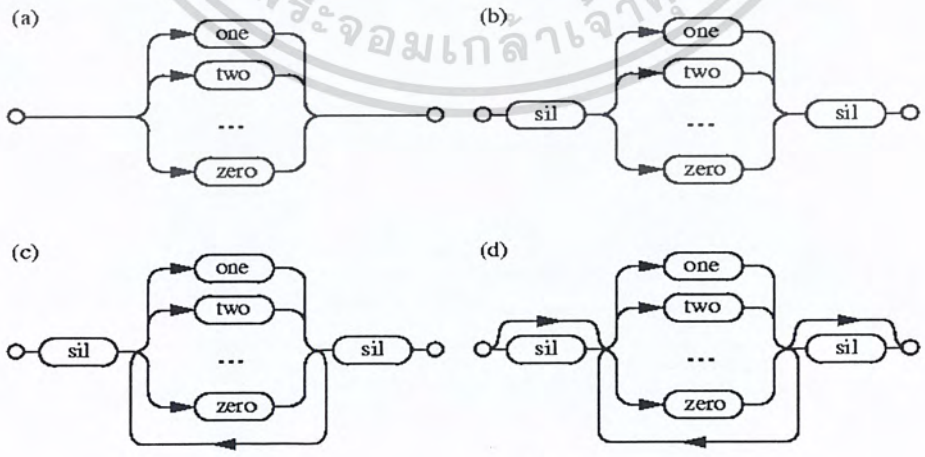
Syntax ของไวยากรณ์ข้างบน ถ้าเกิดว่าอินพุทของตัวเลขมี end-pointed ด้วย อย่างเช่น ช่วงของการหยุดไม่ได้ออกเสียงจะเรียกว่า Silence ก็ต้องมีการใส่ Silence โมเดลก่อนและหลังลำดับของตัวเลขเหล่านั้น ดังนี้

(sil (one | two | three | four | five | six | seven | eight | nine | zero) sil)

ที่ได้แสดงดังภาพ 9.1 (b) นั้นเป็นลำดับของการวนของโมเดล ที่ประกอบไปส่วนของ Silence และตามด้วยลำดับของตัวเลขและตามด้วย Silence และถ้าลำดับของตัวเลขต้องการที่จะทำการเลือกอย่างน้อยหนึ่งครั้งในตัวเลขทั้งหมดก็จะใส่ angle backet “< >” ซึ่งหมายถึงการวนซ้ำหนึ่งครั้งหรือมากกว่า จะทำให้ไวยากรณ์ของ HPArse มีลักษณะดังนี้

(sil < one | two | three | four | five |six | seven | eight | nine | zero > sil)

ซึ่งจะได้ Network ออกมาตามภาพที่ 9.1 (c) ซึ่งอธิบายได้ตามไวยากรณ์ข้างบน



ภาพที่ 9.1 Network ของระบบรู้จำตัวเลขแบบต่างๆ

ไวยากรณ์ของ HPArse สามารถที่จะทำการกำหนดตัวแปรที่จะนำเสนอในลักษณะของ Expression ย่อย โดยที่ชื่อของตัวแปรจะขึ้นต้นด้วยสัญลักษณ์ดอลล่า "$" และให้ค่ากับตัวแปรโดยใช้ Definition ที่อยู่ในภาพแบบดังนี้

$Var = Expression ;

จาก Syntax ข้างบนสามารถที่จะเขียนไวยากรณ์ของ HPArse ให้อยู่ในภาพแบบของตัวแปรได้ดังนี้

```
$digit = one | two | three | four | five | six | seven | eight | nine | zero;
( sil < $digit > sil )
```

ซึ่ง $digit เป็นค่าของตัวแปรซึ่งมีค่าของตัวแปรอยู่ทางด้านขวาของ Expression เมื่อใดก็ตามที่มีชื่อของตัวแปรปรากฏอยู่บน Expression ค่าของ Expression ทั้งทางด้านซ้ายและขวาจะสามารถใช้แทนกันได้ อย่างไรก็ตามตัวแปรจะต้องมีการกำหนดก่อนใช้และห้ามมีการ Recursion

ในการปรับปรุงครั้งสุดท้ายของไวยากรณ์ตัวเลข Silence ที่อยู่เริ่มต้นและสุดท้ายของไวยากรณ์จะถูกคร่อมด้วย Square backet "[ ]"

```
$digit = one | two | three | four | five | six | seven | eight | nine | zero;
( [sil] < $digit > [sil] )
```

ภาพที่ 9.1 (d) จะแสดงถึง Network ที่เป็นผลลัพธ์ของไวยากรณ์ที่ปรับปรุงครั้งสุดท้ายนี้

ไวยากรณ์ของ HPArse เป็นวิธีที่สะดวกสบายในการสร้าง Task Grammar สำหรับใช้งานกับเสียงพูด ตัวอย่างสุดท้ายของการสร้างไวยากรณ์นี้ จะพูดถึงการสร้างไวยากรณ์ของระบบโทรออกด้วยเสียงของโทรศัพท์แบบง่ายๆ ดังนี้

```
$digit = one | two | three | four | five | six | seven | eight | nine | zero;
$number = $digit { [pause] $digit};
$scode = shortcode $digit $digit;
$telnum = $scode | $number;
$cmd = dial $telnum | enter $scode for $number | redial | cancel;
$noise = lipsmack | breath | background;
( < $cmd | $noise > )
```

ในดิกชันนารีจะต้องมีการนิยามสำหรับการหยุดเสียง (Shot Pause) เสียงกระทบของริมฝีปาก (Lipsmack) เสียงหายใจ (Breath) และเสียงพื้นหลัง (Background) ที่จะถูกอ้างอิงไปยัง HMMs ที่ได้สร้างโมเดลของเสียงเหล่านี้แล้ว เพื่อที่จะนำมาอ้างอิงได้ในไวยากรณ์ ซึ่งเสียงทั้งหมดข้างต้นที่กล่าวมาจะเรียกว่า Noise และในดิกชันนารีของเสียงเหล่านี้จะให้เอาท์พุทเป็นสัญลักษณ์ว่างเปล่า หรือว่า Null

ในส่วนสุดท้ายในหัวข้อ Word Network เป็นส่วนที่ต้องพึงระวังสำหรับ Network ใดๆที่ทำการใส่ loop ที่ไม่มีจบสิ้น ของ tee-model จะทำให้เกิดข้อผิดพลาดขึ้น ตัวอย่างเช่น สมมติว่า sp ทั่วไปจะเป็น tee-model ที่มี State เพียง State เดียวใช้ในการแทนเสียง Shot Pause และถูกใส่เข้าไปใน Network ดังข้างล่างนี้ ซึ่งจะทำให้เกิดข้อผิดพลาดขึ้นได้

(sil < sp | $ digit > sil)

จากไวยากรณ์ข้างบนจะถูก Recognize ตามลำดับของ digit ที่อาจจะมีส่วนของ Shot Pauses (sp) อย่างไรก็ตาม Syntax ข้างต้นอาจจะมีการวนซ้ำในส่วนของ Sp หลายๆ ครั้งซึ่งเราไม่ต้องการเช่นนี้ ในการวนของคำในลำดับโดยไม่วนผ่านคำซ้ำๆโดยที่ค่าอินพุทที่เข้ามาไม่เหมือนเดิม วิธีแก้ปัญหานี้ก็คือทำการจัดวางตำแหน่งของคำใน Network ใหม่ อย่างเช่นตัวอย่างข้างบนจะเขียนใหม่ได้เป็น

(sil < $digit sp > sil)

9.2 โมเดลทางสถิติ (Statistical Model)

โมเดลทางสถิติถูกนำมาใช้อย่างกว้างขวางในการบัญชาการระบบให้เป็นไปตามภาพแบบที่ต้องการ ซึ่งใช้วิธีการทางสถิติและความน่าจะเป็นเป็นหลักการเบื้องต้น และได้ถูกนำมาใช้เป็นภาพแบบของโมเดลทางภาษา (Language Model)

หลักการนี้สามารถที่จะอธิบายถึงความยืดหยุ่นและโครงสร้างที่ไม่แน่นอนขององค์ประกอบของภาษาพูดของมนุษย์ ซึ่งมันจะทำการค้นหาค่าที่เหมาะสมและสามารถที่จะลดความซ้ำซ้อนความสับสนและทำให้การสร้างโมเดลทางภาษาง่ายขึ้นจากเดิมมากและมีประสิทธิภาพมากขึ้นอีกเช่นกัน

ส่วนสำคัญของโมเดลนี้เจาะจงไปยังการทำนายลำดับของคำซึ่งบางครั้งเรียกว่า Stochastic Language Models ซึ่งหลักการของโมเดลทางสถิติที่ถูกคิดค้นขึ้นมาเพื่อจุดประสงค์ทางการค้ามีอยู่ 2 โมเดลด้วยกันคือ N-gram Model และ N-class Model ซึ่งในปริญญานิพนธ์ฉบับนี้จะกล่าวถึงเพียงส่วนของ N-gram Model เพราะว่าเป็นโมเดลที่ถูกใช้ในการทำโครงงานครั้งนี้

9.2.1 N-gram Model

ในการสร้าง N-gram model นั้นเรียกได้ว่าเป็นการสร้างโมเดลที่เป็นที่นิยมและถูกใช้บ่อยๆในระบบการรู้จำ โดยเฉพาะการรู้จำคำศัพท์จากเสียงพูด

> *N-gram model ถูกค้นคว้าและนำมาใช้ครั้งแรกในการทำการรู้จำในปี 1970 โดย Frederick Jelinek จากบริษัท IBM และหลังจากนั้นก็ได้ทำการพัฒนาไปเป็น IBM's Tangora ซึ่งเป็นระบบรู้จำ จนกระทั่งปี 1980 Dragon System ได้รวมขึ้นเป็น Bigram model โดยให้ชื่อว่า Dragon Dictate ตั้งแต่นั้นมาก็กลายมาเป็นส่วนสำคัญในการสร้างไวยากรณ์ของระบบคำศัพท์ขนาดใหญ่และถูกใช้บ่อยๆในภาพแบบของ Bigram mdel และ Tri-gram model*

9.2.2 ความสามารถของ N-gram Model

ใน N-gram model นั้นจะเป็นการพิสูจน์คำในขณะนั้นซึ่งไม่รู้จักว่าเป็นคำอะไร โดยการสมมติโดยที่การพิสูจน์คำนั้นจะขึ้นอยู่กับคำก่อนหน้านั้น (N-1 word) และข้อมูลของเสียงที่ไม่รู้จัก

ตัวอย่างเช่น ใน Bigram model นั้นจะมีค่า N=2 ก็จะใช้คำในประโยคก่อนหน้านั้นหนึ่งคำและต่อด้วยคำที่ไม่รู้จักสมมติพูดคำว่า

คอมพิวเตอร์ของ [Unknown word]

คำที่ไม่รู้จักจะถูกทำการพิสูจน์ (Identified) ซึ่งจะใช้คำข้างหน้าหนึ่งคำคือคำว่า "ของ" เป็นตัวช่วยบ่งบอก และจำทำเป็นรายการการเรียงกันของคำที่มีอยู่ดูว่าความน่าจะเป็นของคำที่ไม่รู้จักน่าจะเป็นคำว่าอะไร ซึ่งการพิสูจน์จะใช้ข้อมูลของเสียงพูดที่ไม่รู้จักเข้ามาทำการพิสูจน์ด้วย ซึ่งจะช่วยให้การพิสูจน์ใกล้เคียงความจริงมากขึ้น

การสร้าง N-gram model เพื่อทำการพิสูจน์และวิเคราะห์คำที่ไม่รู้จักนั้นจะสร้างโดยการรวมค่าความน่าจะเป็นเข้ากับค่าของข้อมูลของเสียงพูดที่ไม่รู้จักคำนั้น ซึ่งได้จากค่าอินพุทของคำที่ไม่รู้นั้น และจะไม่มีการใช้ข้อมูลอื่น ระหว่างการวิเคราะห์คำนั้น N-gram model จะมีหน้าต่างที่เคลื่อนที่ได้เท่ากับ N-1 คำที่จะทำการสร้างรายการของการเรียงกันของคำที่จะสร้าง วิธีการนี้เป็นการจำกัดขอบเขตในการวิเคราะห์และช่วยทำให้ไม่สับสนและทำให้การวิเคราะห์เร็วขึ้นด้วย

N-gram model นั้นจะทำให้คำศัพท์ที่มีอยู่มีถูกกระตุ้นมีความเป็นไปได้ที่จะถูกเลือกและจะทำให้คำศัพท์ทั้งหมดถูกใช้อย่างครบถ้วนคำไหนจะถูกใช้มากหรือน้อยขึ้นอยู่กับประโยคทั้งหมดที่ทำการสร้างโมเดลนั้นใช้คำนั้นมากหรือไม่นั่นก็คือจะทำให้ความน่าจะเป็นซึ่งเกิดจากการรวมกันของข้อมูลของโมเดลและการเรียงกันของคำที่ใช้ในโมเดล ซึ่งเหตุผลข้างต้นนี้ทำให้ลดข้อจำกัดในการหาคำผิดการหาคำไม่ตรงหรือไม่สามารถใช้คำนั้นได้ ที่เกิดขึ้นใน Finite State Grammar

ข้อควรจำ โมเดลทางสถิติของภาษามีพื้นฐานอยู่บน N-gram model ถูกนำมาใช้ในการเพิ่มความถูกต้องในการรู้จำเสียงพูด ซึ่งเป็นภาพแบบที่ถูกนำมาใช้กันมากเพราะมันง่ายในการสร้าง แต่ว่าความต้องการของข้อมูลประโยคในลักษณะของ Text เพื่อมาทำการ Train รู้จำคำศัพท์จำนวนมากเพียงพอ ซึ่งจะใช้ในการคำนวณ ประเมิน และสร้างเป็นพารามิเตอร์ของโมเดล [ ]

แนวคิดของ N-gram model ทำให้เป็นข้อดีของการบังคับคำศัพท์จำนวนมากๆ โดยที่มันจะทำการค้นหาวิธีการที่จะทำให้เกิดความยืดหยุ่นมากที่สุด เร็วที่สุด ค้นหาอย่างถี่ถ้วน รวมไปถึงการคำนวณที่ไม่มากไม่ทำให้ระบบช้าลง ซึ่งเป็นเหตุผลให้โมเดลนี้ประสบความสำเร็จเป็นอย่างมาก

9.2.3 วิธีการสร้าง N-gram model

N-gram model สร้างจากการคลี่คำต่างๆจากลำดับของคำในแพทเทินโดยตรงจากข้อมูลทางภาษาขนาดใหญ่ เมื่อพบครั้งหนึ่งแล้วมันจะถูกทำการเรียงลำดับในภาพแบบของความน่าจะเป็น ตามค่า N ว่าจะให้เรียงกี่คำจะมีคำตามหลังเป็นจำนวน N-1 คำ เช่น Bigram model จะมีคำตามหลัง 2-1 = 1 คำ ซึ่งวิธีการข้างต้นนั้นเรียกว่า Training ระบบ

เมื่อทำการ Train แล้วจะได้ N-gram Language Model ออกมาซึ่งปกติแล้วโมเดลจะอยู่ในภาพแบบของโครงข่าย (Lattice) หรือ Link list ซึ่งแต่ละ Link เหล่านี้จะมีความน่าจะเป็นอยู่ด้วยซึ่งหาได้ในระหว่างการ Train

ในระหว่างการใช้งานโมเดลความน่าจะเป็นจะถูกใช้เป็นตัวเลือก ประเมิน และเรียงคำที่มีอยู่ใน List หลักสำคัญในการสร้างโมเดลที่ดีได้นั้นจะต้องมีข้อมูลที่มากเพียงพอ ซึ่งจะมากขึ้นตามขนาดของคำที่เพิ่มขึ้นในดิกชันนารีที่ใช้งาน และ ค่าของ N ถ้าค่า N มีขนาดน้อยๆอย่างเช่น N = 3 จะต้องการข้อมูลทางภาษามาใช้ในการ Train อย่างมากมาย Jelinek (1989) ได้ทำการประเมินมาแล้วว่า เพื่อที่จำทำการสร้างโมเดลที่มีความเป็นไปได้ในการใช้คำศัพท์ทุกๆคำของ Tri-gram model ของคำศัพท์จำนวน 7000 คำจะต้องทำรายการคำหรือ Corpus ที่บรรจุคำขนาด 129 ล้านล้านคำ

Bigram (N=2) และ Unigram (N=1) ความน่าจะเป็นของทั้งสองโมเดลนี้จะถูกใช้ประกอบขึ้นเป็น Tri-gram model สำหรับแต่ละลำดับใน Tri-gram model ใน Bigram model เช่นกัน ก็จะประกอบด้วยลำดับของ Unigram model ต่อๆกัน

9.2.4 การสร้าง Bigram Language Model โดยใช้เครื่องมือใน HTK

ก่อนที่จะทำการอธิบายในส่วนของการสร้าง Network โดยใช้เครื่องมือชื่อ HBUild การสร้างและการใช้งาน Bigram language model เป็นส่วนที่ต้องทำความเข้าใจก่อนเป็นอันดับแรก เพื่อที่จะทำให้ HTK รองรับแบบจำลองทางสถิติจึงต้องมีการกำหนดไลบรารีโมดูลที่ชื่อ HLM ขึ้นมาถึงแม้ว่าในการติดต่อกับ HLM สามารถที่จะรองรับภาพแบบทั่วไปของ N-gram model แต่ว่าความสะดวกในการสร้างและการใช้งาน N-gram model ถูกจำกัดอยู่ที่ Bigram model

Bigram Language Model สามารถสร้างได้โดยใช้เครื่องมือชื่อ HLStats ซึ่งจะนำข้อมูลจาก Master Label File ชื่อ words.mlf มาทำการ Train เพื่อเพื่อให้ได้ bigram model ออกมาชื่อ bigfn

HLStats -b bigfn -o wordlist words.mlf

คำทุกๆคำที่ถูกใช้ใน Label files จะต้องมีอยู่ในไฟล์ชื่อ wordlist คำสั่งข้างบนจะทำการอ่านทุก ๆ Transcription จากไฟล์ words.mlf ซึ่งจะสร้างตารางในการนับของ Bigram model ในหน่วยความจำและจะให้เอาท์พุทเป็นไฟล์ในภาพแบบของ backed-off bigram ชื่อว่า bigfn ซึ่งตัวอย่างไฟล์นี้จะอยู่ในภาคผนวก สูตรสำหรับการทำในส่วนนี้จะอยู่ในส่วนอ้างอิงของ HLStats แต่อย่างไรก็ตามพื้นฐานของแนวคิดจะอยู่ในสูตรดังนี้

$$p(i,j) = \begin{cases} (N(i,j) - D)/N(i) & \text{if } N(i,j) > t \\ b(i)p(j) & \text{otherwise} \end{cases}$$

โดยที่ค่าของ N(i,j) เป็นตัวเลขเวลาของคำที่ j ตามหลังคำที่ I และ N(I) คือตัวเลขของเวลาเมื่อคำที่ I ปรากฏขึ้น ที่สำคัญในส่วนเล็กๆของความน่าจะเป็นที่ถูกใช้เป็นน้ำหนักจะเป็นตัวลดจากค่าจำนวนนับ

ขอบ Bigram ที่ถูกนับมากกว่า และในส่วนของที่ bigram ที่กระจายและนานๆเกิดขึ้นที วิธีการนี้เรียกว่า *Discounting* ค่าคงที่ของการ Discount (D) คือ 0.9 แต่ว่าสามารถทำการเปลี่ยนเป็นค่าใหม่โดยการใช้ตัวแปรในการหนดที่ชื่อ DISCOUNT เมื่อการนับของ Bigram ต่ำลงมากกว่าค่า Threshold t bigram จะทำการคำนวณหาค่าความน่าจะเป็นสำหรับ Unigram และจะเก็บค่าของแต่ละ Unigram ไว้เพื่อจะได้นำมารวมกันเพื่อสร้าง Bigram ขึ้นมา

Backed-off bigram จะอยู่ในภาพของ Text file โดยใช้มาตรฐานของ ARPA MIT-LL format ที่ใช้ใน HTK มีภาพแบบดังนี้

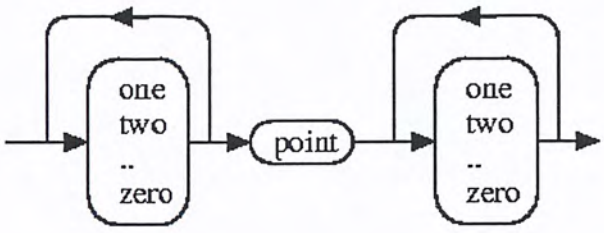
```
\data\
 ngram 1=<num 1-grams>
ngram 2=<num 2-ngrams>
\1-grams:
 P(!ENTER) !ENTER B(!ENTER)
 P(W1) W1 B(W1)
 P(W2) W2 B(W2)
...
P(!EXIT) !EXIT B(!EXIT)
 \2-grams:
P(W1 | !ENTER) !ENTER W1
 P(W2 | !ENTER) !ENTER W2
 P(W1 | W1) W1 W1
P(W2 | W1) W1 W2
 P(W1 | W2) W2 W1
 ....
P(!EXIT | W1) W1 !EXIT
P(!EXIT | W2) W2 !EXIT
\end\
```

ซึ่งค่าความน่าจะเป็นจะอยู่ในภาพแบบของ log ฐานสิบและค่าปกติของการเริ่มต้นประโยคและสิ้นสุดประโยคใน Bigram นี้คือ !ENTER และ !EXIT สามารถที่จะเปลี่ยนได้โดยใช้ HLStats –s แล้วตามด้วยค่าใหม่

9.2.5 การสร้าง Word Network โดยใช้ HBUild

หน้าที่หลักๆของ HBUild คือสามารถจะสร้าง Network ในระดับคำ (Word Level) จากโครงข่ายของคำ (Lattice) และกลุ่มของ Lattice ย่อย แต่ละ Lattice จะมีการบรรจุ Definition ของ node ที่สามารถ

อ้างอิงไปยัง Lattice อื่นๆได้และจะอนุญาติให้การทำการรู้จำในระดับของ Network ถูกแยกออกเป็นส่วนๆ ไปเป็นจำนวนของ Network ย่อยๆที่สามารถนำมาใช้ใหม่ได้ภายในจุดต่างๆของ Network นั้น



ภาพที่ 9.2 Decimal Syntax

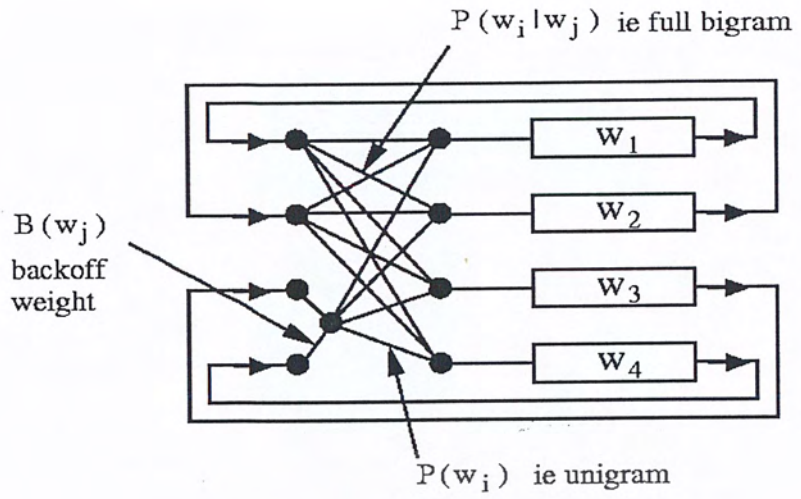
สมมติว่าต้องการค่าอินพุทเป็นจำนวนเต็มสิบ Network ที่ต้องการสร้างจะต้องเป็นไปดังภาพ 9.2 ซึ่งสามารถที่จะเขียนภาพนี้ได้โดยตรงจากไฟล์ SLF ซึ่งมีการใส่ Loop ของตัวเลขเข้า 2 ครั้ง จะทำให้ต้องสร้าง Network ที่ซ้ำซ้อน โดยสามารถหลีกเลี่ยง Loop ของตัวเลขนั้นได้โดยการกำหนด Loop ของตัวเลขเหล่านั้นในภาพแบบของ Network ย่อยและเมื่อต้องการใช้งานก็เพียงแค่อ้างอิงถึง Network ย่อยเหล่านั้นได้

Network ย่อยจะถูกตรวจสอบโดย ส่วนหัวของ Field ที่ชื่อ SUBLAT และจะสิ้นสุดการทำงานของมันเองในส่วน Body ของ Network ย่อยสามารถที่จะเขียนขึ้นได้ตามปกติ เมื่อมีการกำหนดค่าแล้ว Network ย่อยสามารถที่จะนำเอามาแทนที่เป็น Network ในระดับที่สูงกว่าโดยใช้ Field ที่ชื่อ L ใน Definition ของ node อย่างเช่นใน node 1 และ node 3 ของ Network ของตัวเลขข้างบน

แน่นอนว่าวิธีการนี้สามารถที่จะทำได้อย่างต่อเนื่องและ Network ในระดับที่สูงกว่าสามารถที่จะอ้างอิงกับ Network ที่กล่าวมาแล้วได้เพียงรู้ว่า Network นั้นชื่ออะไรเท่านั้นเอง

หนึ่งใน Network ที่ใช้ในการรู้จำที่ธรรมดาที่สุดก็คือ Word loop ซึ่งจะมีคำทุกๆคำเชื่อมกันในภาพแบบขนานที่สามารถวนไปคำใดๆก็ได้ซึ่งถูกใช้เป็นพื้นฐานในการบังคับทิศทางของคำหรือ ถูกใช้ใน Transcription

HBUild สามารถที่จะทำการสร้าง Loop จากรายการของคำได้ซึ่งมันสามารถที่จะอ่าน File ในภาพแบบของ Bigram ทั้ง ARPA MIT-LL format และ ภาพแบบของ HTK Format และจะนำเอาค่าความน่าจะเป็นใน Bigram model ไปใช้ในการ Transition อย่างไรก็ตาม ในการใช้ Bigram model แบบเต็มภาพแบบก็หมายความว่าคู่ของคำที่แตกต่างกันจะต้องมีการวนกลับ (Loop back) ในแต่ละคู่ซึ่งจะเป็นการเพิ่มปริมาณและขนาดของ Network ให้ถูกพิจารณามากขึ้นและในการใช้งานความเร็วจะลดลงมากเมื่อ Backed-off Bigram ถูกใช้



ภาพที่ 9.3 Backed-off Bigram Word-loop Network

อย่างไรก็ตาม Transition ของ Backed-off bigram สามารถที่จะทำการแชร์ Loop back ทั่วไปได้ ซึ่งแสดงดังภาพ 12.6 และเมื่อใช้งานเครื่องมือ HBUild นี้จริงๆ ไฟล์ของ backed-off bigram ที่ถูปอินพุทในภาพแบบของ ARPA MIT-LL HBUild จะทำการมองหาว่ามีส่วนไหนที่จะสามารถกระทำการแชร์ได้บ้าง

ตัวอย่างคำสั่งในการแปลงให้ Backed-off bigram อยู่ในภาพแบบที่ได้กล่าวมาข้างต้นโดยใช้ HBUild โดยพิมพ์ดังนี้

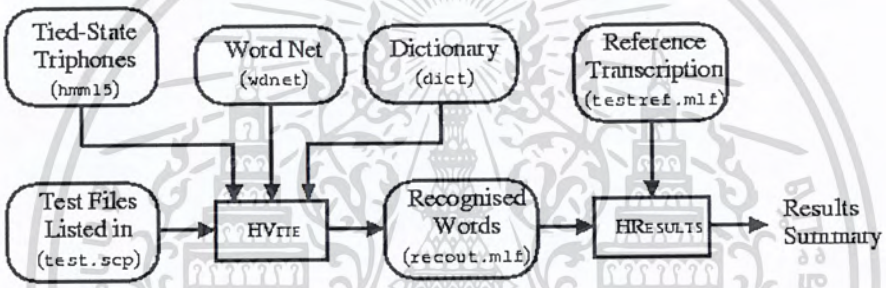
```
HBuild -n bigfn wordlist Ngram
```

ใน HTK นั้นได้มีเครื่องมือที่ช่วยในเรื่องไวยากรณ์ ทำให้สามารถสร้างได้สะดวกมาก แต่ถ้าหากจำเป็นต้องใช้ไวยากรณ์ในรูปแบบอื่นก็สามารถทำได้ แต่ไฟล์ต้องอยู่ในรูปแบบมาตรฐานที่ HTK รู้จัก เหตุผลที่ HTK มีเครื่องมือในการสร้าง Language Model แบบ Bigram Model เพียงอย่างเดียวก็เพราะว่า Bigram Model เป็น Model ที่ได้รับความนิยมสูงและสร้างได้ง่ายกว่าหลายๆ โมเดล

บทที่ 10
การทดสอบโมเดล

10.1 วิธีการทดสอบการรู้จำ (Recognizer Evaluation)

เป็นขั้นตอนของการประเมินความถูกต้องของการรู้จำคำของโมเดลที่ได้สร้างขึ้น ซึ่งจะต้องมี dictionary และ recognition network (Language Model) และไฟล์เสียงพูดที่จะใช้ทดสอบความถูกต้อง ซึ่งกระบวนการทดสอบความถูกต้องของโมเดลนั้นจะมีเครื่องมือชื่อ Hvite ซึ่งจะใช้ Recognize เสียงพูดที่นำมาทดสอบซึ่งจะให้ผลลัพธ์เป็นคำในประโยคออกมา ซึ่งจะนำมาเปรียบเทียบกับคำที่ถูกต้องโดยเครื่องมือที่ชื่อ HREsults ซึ่งจะวัดค่าผลลัพธ์ออกมาเป็นเปอร์เซ็นต์ความถูกต้องของการทดสอบออกมา ขั้นตอนการทดสอบแสดงดังรูป 10.1



รูปที่ 10.1 Flow-Chart ของขั้นตอน Recognizer Evaluation

10.2 เครื่องมือในการ Recognize ชื่อ HVite

เราจะต้องใช้เครื่องมือนี้ในการเตรียมข้อมูลไฟล์ Transcription ซึ่งเป็นไฟล์ที่ได้จากการ Recognition ที่ชื่อว่า recount.mlf ซึ่งเป็น output ของเครื่องมือนี้ ซึ่งเราจะต้องเตรียมรายชื่อไฟล์ .mfc ที่จะทำการ test ใส่ไว้ในไฟล์ที่ชื่อว่า test.scp และทำการรันคำสั่งดังต่อไปนี้

```
HVite -H hmm15/macros -H hmm15/hmmdefs -S test.scp -l * -i recout.mlf -w wdnet
-p 0.0 -s 5.0 dict tiedlist
```

ซึ่ง option –p และ –s เป็นการตั้งค่า word insertion penalty และ grammar scale factor ซึ่ง word insertion penalty เป็นการกำหนดค่าเพิ่มลงไปให้กับ token แต่ละตัวเมื่อมีการย้ายจากการสิ้นสุดของคำหนึ่งไปยังจุดเริ่มต้นคำอีกคำหนึ่ง สำหรับ grammar scale factor นั้นจะเป็นการทำการปรับค่า language model probability ก่อนที่จะใส่ลงไปใน token ขณะที่กำลังย้ายที่จากจุดสิ้นสุดของคำไปยังจุดเริ่มต้นของคำถัดไป ซึ่งค่า parameters ตัวนี้มีผลเป็นอย่างมากกับประสิทธิภาพของการทำการ recognition ซึ่งถือได้ว่าการปรับค่ากับข้อมูลที่ทำการแปลงแล้วนั้นเป็นการสมควร

ใน dictionary นั้นประกอบไปด้วย monophones transcriptions ซึ่งในขณะที่ HMM list นั้นเป็นแบบ triphones ในการนี้ Hvite จะทำการแปลงที่เหมาะสมกับ word network ที่ชื่อ wdnet แต่อย่างไรก็ตามถ้าหากว่า HMM list นั้นประกอบด้วยทั้ง monophones และ context-dependent phones แล้ว Hvite จะไม่สามารถเข้าใจได้

นอกจากนั้น Hvite ยังใช้ในการประเมินค่าในการปรับปรุงประสิทธิภาพการจดจำแบบ unsupervised adaptation ซึ่งรายละเอียดของการใช้งานและพารามิเตอร์ของเครื่องมือ HVite ต่างๆจะอยู่ในรูปแบบดังนี้

HVite [options] dictFile hmmList testFiles ...

ในส่วนของ option มีพารามิเตอร์ให้ใช้งานดังนี้

-a จะสร้าง network file ในรูปแบบ alignment ในแต่ละไฟล์ที่ทำการ test

-b s ใช้พารามิเตอร์ s เพิ่มเติมเพื่อแบ่งประโยคในโมเดล ระหว่างการทำ alignment

-d dir เป็นการเจาะจงไดเรกทอรีเพื่อค้นหาไฟล์ HMM definition ให้ตรงกับ label network ที่ใช้ในการทำการรู้จำ

-e เมื่อไฟล์อินพุท รับเข้ามาเป็น direct audio ไฟล์ transcription ที่ออกมาจะยังไม่ได้ถูกบันทึก เมื่อใช้ option นี้ จะสามารถบันทึกข้อมูล transcription เป็นไฟล์ได้โดยตั้งชื่อไฟล์ตามลำดับไฟล์ transcription ที่ถูกบันทึก

-f ในระหว่างการเชื่อม states ต่างๆ option นี้จะทำให้ไฟล์ที่ได้มีข้อมูลเป็น หมายเลขของ states และชื่อของคำที่ใช้ในการรันคำสั่งนี้

-g สามารถทำการ replay ไฟล์ audio ที่ใช้ในการทำการรู้จำได้ หากไฟล์นั้นเป็นอินพุทแบบ direct audio

-i s ให้ผล transcription ที่ออกมาเป็นแบบ MLF

-j I จะทำการปรับปรุงโมเดลโดยใช้หลักการ adaptation ในทุกๆคำ (I)

-l dir เป็นการเจาะจงไดเร็กทอรีในการบันทึกข้อมูล label ไฟล์ ถ้าไม่มีพารามิเตอร์ในส่วนนี้ เครื่องมือจะทำการบันทึกไฟล์ดังกล่าวไว้ที่ไดเร็กทอรีเดียวกันกับข้อมูล แต่เมื่อใช้ option ในรูปแบบ –l '*' จะทำให้ label file ที่ชื่อ xxx มีเครื่องหมาย * นำหน้าและจะอยู่ในรูปแบบ "*/xxx" ในไฟล์ MLF ที่ออกมา ซึ่งจะมีประโยชน์มากในการสร้างไฟล์ MLFs เมื่อไฟล์ข้อมูลอยู่ต่างไดเร็คทอรีกัน

-m ในระหว่างการกำหนดตำแหน่งของโมเดลต่างๆ option นี้จะทำให้ไฟล์ที่ได้มีข้อมูลเป็น หมายเลขของ states และ ชื่อของคำที่ใช้ในการรันคำสั่งนี้ ซึ่งไม่ใช่รูปแบบของ symbols

-p f ใช้เมื่อเซตค่าของความน่าจะเป็นของการแทรกคำเป็น f ซึ่งปรกติจะถูกเซตเป็น 0.0

-r f ใช้เมื่อต้องการเซตค่าความน่าจะเป็นของ dictionary pronunciation เป็น f ซึ่งแบบปรกติจะมีค่าเป็น 1.0

-s f เซตค่า Factor ของ grammar scale เป็น f โดยปรกติจะมีค่าเป็น 1.0

-t f การค้นหาโมเดลที่มีค่าความน่าจะเป็นสูงกว่าค่า f ที่ได้กำหนดไว้ หากไม่ต้องการ option การค้นหานี้ควรเซตค่า f เป็น 0.0 ซึ่งเท่ากับค่าปรกติที่เครื่องมือได้เซตไว้แล้ว

-u i เซตค่าของจำนวน active model สูงสุดไว้ที่ค่า i และเซตค่าไว้ที่ 0 เมื่อไม่ต้องการกำหนดค่า

-w [s] ทำการรู้จำโดยใช้ word level network ถ้ามี s จะเป็นการใช้ word network ในการรู้จำสำหรับทุกๆ ไฟล์

-L dir เป็นการเจาะจงไดเร็กทอรีเพื่อค้นหา input label เมื่อมี –a หรือ network files เมื่อมี -w

-X s เซ็ตข้อมูลใน input label หรือ network files ให้เป็น s

-F fmt เป็นการเซ็ตรูปแบบของข้อมูล ให้อยู่ในรูป fmt

-G fmt เซ็ตรูปแบบของ label file ให้เป็น fmt

-H mmf เป็นการเรียกข้อมูล HMM macro model file ที่อยู่ในรูปแบบ mmf ซึ่งสามารถพิมพ์คำสั่งต่อเนื่องกันได้เรื่อยๆ ถ้าต้องการเรียกข้อมูล mmf หลายๆไฟล์

-I mlf จะเรียก Master Label File ที่อยู่ในรูปแบบ mlf และสามารถพิมพ์ต่อกันไปได้ถ้าต้องการไฟล์ mlf หลายๆ ไฟล์

10.3 เครื่องมือในการวิเคราะห์ความถูกต้องชื่อ HResults

ถ้าไฟล์ testref.mlf นั้นประกอบไปด้วย word level transcription สำหรับไฟล์ที่นำมาทดสอบประสิทธิภาพที่แท้จริงนั้น สามารถทำการวัดได้โดยใช้คำสั่ง Hresults ตามตัวอย่างต่อไปนี้

HResults -I testref.mlf tiedlist recout.mlf

ซึ่งจะได้ผลลัพธ์ดังนี้

```
====================== HTK Results Analysis =======================
□
  Date: Mon Nov 13 14:15:44 2000
  Ref : words.mlf
  Rec : recout2.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=29.05 [H=52, S=127, N=179]
WORD: %Corr=88.29, Acc=45.99 [H=407, D=0, S=54, I=195, N=461]
```

เป็นการเปรียบเทียบไฟล์เพื่อหาความถูกต้อง ซึ่งในตารางจะมีค่า N เป็นจำนวนของคำทั้งหมดที่เป็นเสียงทดสอบ H เป็นจำนวนความถูกต้องที่ได้จากการรู้จำ ในการเปรียบเทียบจะเอาอักษรของคำมาทำการเปรียบเทียบ D เป็นจำนวนความผิดพลาดที่เกิดจากการลบตัวอักษร (ถ้าคำ 2 คำเมื่อเทียบกันแล้วไม่ถูกต้องจะทำการทดลองลบตัวอักษรออกแล้วนำมาเทียบกันอีกครั้ง) S เป็นจำนวนความผิดพลาดที่เกิดจากการแทนที่ I เป็นความผิดพลาดที่เกิดจากการแทรกคำ

ผลที่แสดงสามารถคำนวณได้ดังนี้

$$\text{Percent Correct} = \frac{N - D - S}{N} \times 100\%$$

$$\text{Percent Accuracy} = \frac{N - D - S - 1}{N} \times 100\%$$

ซึ่งโดยรวมแล้ว Percent Accuracy จะเป็นการวัดประสิทธิภาพที่ถูกต้องกว่า ซึ่งการใช้เครื่องมือ HRresults นี้คำสั่งจะอยู่ในรูปแบบ

HResults [options] hmmList recFiles ...

และต่อไปจะกล่าวถึง optionของ Hresult โดยละเอียด

-a s เปลี่ยน label SENT ของเอาท์พุท ให้เป็นแบบ s

-d N จะทำการค้นหาไฟล์ N ไฟล์แรกของแต่ละ test label file เพื่อให้แน่ใจว่าไฟล์ดังกล่าวเหมาะสมกับ label ที่อ้างอิงอยู่ให้มากที่สุด

-f เครื่องมือนี้จะแสดงสถิติของ input files ทั้งหมด และจะแสดงผลออกมาเมื่อทำงานเสร็จสิ้นแล้ว โดย Hresult จะทำการเลือกข้อมูลสถิติของแต่ละ input file ให้ตรงกัน

-m N ให้จบการทำงานหลังจากรวบรวมข้อมูลสถิติเมื่อถึงไฟล์ N

-p จะทำให้เอาท์พุทเป็น matrix ของหน่วยเสียงที่ประกอบกัน

-s จะทำให้หน่วยของเสียงที่อยู่ในรูปแบบ A+B-C เปลี่ยนเป็นรูปแบบ หน่วยเสียง B เพื่อการวิเคราห์ข้อมูลเสียงที่ใช้โมเดลแบบ context dependent model

-I mlf จะทำการเรียกข้อมูล master label file ที่อยู่ในรูปแบบ mlf ซึ่งจะพิมพ์ซ้ำๆกันได้หากต้องการเรียกไฟล์ mlf หลายๆไฟล์

-L dir ทำการค้นหา label file ในไดเร็กทอรีที่กำหนด

10.4 การทดสอบโมเดล

10.4.1 เมื่อผู้ทดสอบอยู่ในกลุ่ม Train โดยไม่ใช้ Grammar

วัตถุประสงค์

เพื่อเป็นการศึกษาทดลองใช้ HTK เพื่อทำการหาประสิทธิภาพของการรู้จำเสียงภาษาไทยโดยใช้ HTK สำหรับการทดสอบของคนที่ทำการ train และไม่ใช้ Grammar ในการ Recognize ว่ามีความถูกต้องเพียงใด

ขั้นตอนการทดลอง

1. นำแบบจำลองที่ได้มาใช้เพื่อหาผลการทดลอง
2. จากนั้นทำการสร้างไฟล์ word loop ขึ้นมา
3. แปลงไฟล์ word loop ที่ได้สร้างขึ้นให้อยู่ในรูปของ SLF โดยใช้คำสั่ง

HPArse wordloop wdnet

4. จากนั้นพิมพ์คำสั่งดังนี้

HVite -H hmm16\macros -H hmm16\hmmdefs -S test -l * -i recout.mlf –w wdnet -p 0.0 -s 5.0 dictPlus newlist

หลังจากทำการทดสอบแล้ว สามารถทำการดูแลผลการทดสอบได้โดยเรียกใช้คำสั่งนี้

HResults -I words.mlf newlist recout.mlf

โดยผลการทดสอบในการเขียน Grammar ในลักษณะ word loop หรือ ไม่ใช้ Grammar ได้ผลการทดสอบดังนี้

```
====================== HTK Results Analysis =======================
  Date: Mon Nov 13 14:20:44 2000
  Ref : words.mlf
  Rec : recout.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=20.41 [H=36, S=127, N=179]
WORD: %Corr=78.45, Acc=18.06 [H=361, D=0, S=25, I=140, N=461]
================================ ===========================
```

จากผลการทดลองที่ได้ มีเปอร์เซ็นต์ความถูกต้องของประโยค 20.41% ความถูกต้องของคำ 78.45% และความถูกต้องของคำจริง 18.06%

10.4.2 เมื่อผู้ทดสอบอยู่ในกลุ่ม Train โดยใช้ Bigram Language Model

วัตถุประสงค์

เพื่อเป็นการศึกษาทดลองใช้ HTK เพื่อทำการหาประสิทธิภาพของการรู้จำเสียงภาษาไทยโดยใช้ HTK สำหรับการทดสอบของคนที่ทำการ train และใช้ Bigram Language Model ในการ Recognize ว่ามี

ความถูกต้องเพิ่มขึ้นจากการทดสอบที่ไม่ใช้ Language Model เพียงใด

ขั้นตอนการทดลอง

1. นำแบบจำลองที่ได้มาใช้เพื่อหาผลการทดลอง
2. เมื่อทำการเรียกใช้คำสั่ง

   HLStats -b bigfn -o wordlist words.mlf

   ซึ่ง wordlist เป็นรายการของคำทั้งหมดที่ทำการ Train เสร็จแล้ว และ words.mlf เป็น Master Label File ที่ได้สร้างขึ้นในขั้นตอนการเตรียมข้อมูล และ bigfn เป็น Output file
3. HBuild -n bigfn wordlist Ngram

   เปลี่ยนรูปแบบไฟล์ของ bigfn ให้มาอยู่ในรู้ที่ HTK รู้จักได้ไฟล์ Ngram เป็น Output
4. จากนั้นพิมพ์คำสั่งดังนี้

HVite -H hmm16\macros -H hmm16\hmmdefs -S test -l * -i recout2.mlf –w Ngram -p 0.0 -s 5.0 dictPlus newlist

หลังจากทำการทดสอบแล้ว สามารถทำการดูแลผลการทดสอบได้โดยเรียกใช้คำสั่งนี้

HResults -I words.mlf newlist recout2.mlf

ผลการทดสอบในการเขียนไวยากรณ์ในลักษณะแบบ Language Model ได้ผลการทดสอบดังนี้

```
====================== HTK Results Analysis =======================
  Date: Mon Nov 13 14:15:44 2000
  Ref : words.mlf
  Rec : recout2.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=29.05 [H=52, S=127, N=179]
WORD: %Corr=88.29, Acc=45.99 [H=407, D=0, S=54, I=195, N=461]
===================================== ============================
```

จากผลการทดลองที่ได้ มีเปอร์เซ็นต์ความถูกต้องของประโยค 29.05% ความถูกต้องของคำ 88.29% และความถูกต้องของคำจริง 45.99%

10.5 สรุป

จากการทดสอบโดยใช้โมเดลเดียวกันข้างต้นจะได้ข้อสรุปว่า Language Model สามารถทำให้ความถูกต้องของการทำการรู้จำเพิ่มขึ้น ซึ่งเห็นได้ชัดจากเปอร์เซ็นต์ Acc ซึ่งเป็นความถูกต้องที่แท้จริงซึ่งเพิ่มขึ้นจากเดิมมาก แต่จากการทดสอบหลายๆครั้งจะพบว่าถ้าโมเดลอย่างเดียวมีความถูกต้องไม่มากเพียงพอก็จะเป็นผลทำให้ความผิดพลาดเพิ่มขึ้นได้เช่นกัน เพราะ Language Model ที่ใช้อยู่เป็นแบบ Bigram Model ซึ่งถ้าคำแรกผิดก็ส่งผลให้คำที่สองผิดพลาดได้เช่นกัน

บทที่ 11
สรุปบทวิจารณ์

11.1 สรุป

ในการทำการรู้จำเสียงนั้นจะเป็นต้องมีการออกแบบแบบจำลองที่ดีตั้งแต่ตอนแรกและมีการเตรียมข้อมูลที่ถูกต้องเพราะการสร้างโมเดลและทำการ Train Model เพื่อให้ความถูกต้องของโมเดลเพิ่มขึ้นจะต้องทำเพิ่มเข้าไปเรื่อยๆซึ่งถ้าการออกแบบหรือกำหนดค่าไม่ดีตั้งแต่ตอนแรกแล้วโมเดลที่ได้ออกมาท้ายสุดจะไม่สามารถแก้ไขส่วนใดๆได้ซึ่งต้องทำการ Train ใหม่เท่านั้น

เมื่อได้ทำการสร้างโมเดลที่ถูกต้องแล้วขั้นตอนในการ Train นั้นจะเป็นต้องใช้ข้อมูลเสียงจำนวนมากจากคนหลายๆคน เพราะว่าข้อมูลเสียงขอแต่ละคนนั้นสามารถนำมาใช้เป็นตัวแทนเสียงของเสียงหลายๆความถี่รวมถึงเป็นการเพิ่มข้อมูลในการ Train ให้มากขึ้นซึ่งจำทำให้โมเดลมีความถูกต้องมากเพิ่มขึ้นด้วยและในการใช้งานโมเดลนั้นโมเดลที่ได้จากการ Train จากเสียงของคนหลายๆคนจะสามารถนำไปใช้กับคนหลายๆคนได้เช่นกัน เพราะว่าโมเดลจำมีข้อมูลเสียงหลายๆความถี่อยู่แล้ว

ในการ Train โมเดลโดยใช้เสียงจากคนๆเดียวมา Train หรือใช้ข้อมูลเดิมมาทำการ Train นั้นจะสามารถเพิ่มความถูกต้องของโมเดลได้แต่ความถูกต้องนั้นจะน้อยมากถ้านำโมเดลนั้นไปใช้กับคนอื่นที่มีลักษณะของความถี่เสียงแตกต่างกันมากๆ

ถึงแม้ว่าในการ Train โมเดลด้วยข้อมูลมากๆแล้วก็ตามความถูกต้องของคำก็ยังไม่ได้เพิ่มขึ้นมากนักซึ่งเป็นเพราะว่าเกิดความสับสนในคำศัพท์ซึ่งบางคำมีการออกเสียงที่คล้ายกันมากๆ ซึ่งการ Train โมเดลที่ดีอย่างเดียวใช้ว่าจะทำให้โมเดลถูกต้องร้อยเปอร์เซ็นต์

ดังนั้นไวยกรของภาษาจึงถูกนำมาใช้ในการควบคุมการเป็นไปของคำว่าคำแต่ละคำถูกใช้ในลักษณะใดใช้มากใช้น้อยมีผลอย่างไร ซึ่ง Language Model เป็นโมเดลทางภาษาที่เหมาะสมและจำเป็นที่ต้องสร้างขึ้นเพื่อเพิ่มความถูกต้องให้กับโมเดล Bigram Language Model เป็นโมเดลที่ถูกสร้างขึ้นได้จากเครื่องมือของ HTK และเป็นโมเดลทางสถิติที่ใช้ความน่าจะเป็นในการควบคุมความเป็นไปของคำ ซึ่งนั่นก็เป็นการบ่งบอกว่าโมเดลนี้ได้รับความนิยมเป็นเพราะมีการสร้างที่ง่ายและมีประสิทธิภาพมากนั่นเอง

ซึ่งในการนำโมเดลไปใช้งานจริงนั้นเป็นเรื่องที่ทำได้ไม่ยากเพียงแต่ว่าเรามีโมเดลและไวยกรของภาษาที่ทำให้การรู้จำคำศัพท์มีความถูกต้องจนเป็นที่ยอมรับขั้นหนึ่งหรือยัง ซึ่งในการที่จะรู้ว่าโมเดลนั้นมีความถูกต้องมากแค่ไหนนั้น HTK มีเครื่องมือที่ใช้ในการวัดและวิเคราะห์และให้ข้อสรุปออกมาเป็นเปอร์เซ็นต์ความถูกต้องของระบบรู้จำนั้นๆ

HTK ในรุ่น 2.2 นั้นไม่สามารถทำการวิเคราะห์เสียงวรรณยุกต์ในภาษาไทยได้ซึ่งในการทำการรู้จำแบบคำเดี่ยวนั้นสามารถที่จะเพิ่มเติมในส่วนของการวิเคราะห์วรรณยุกต์ได้ เพราะว่าคำแต่ละคำแยกกันอย่าอิสระ แต่ทว่าในการทำการรู้จำในลักษระเป็นประโยคนั้นเราไม่สามารถรู้ได้เลยว่าประโยคที่พูดเข้ามานั้นคำในประโยคเริ่มต้นและสิ้นสุดตรงไหน ซึ่งก็เป็นเหตุทำให้ไม่สามารถทำการวิเคราะห์เสียงวรรณยุกต์ได้ซึ่งก็ได้มีการค้นคว้าวิธีการเพิ่มเติมในอนาคตและหวังว่าอีกไม่นานก็จะทำได้สำเร็จ

11.2 ข้อสังเกต ปัญหาในการทดลอง และข้อเสนอแนะ

ใน HTK เวอร์ชันปัจจุบันนั้นถูกทำขึ้นมาเพื่อทำการรู้จำคำของเสียงภาษาอังกฤษ ซึ่งในภาษาอังกฤษนั้นไม่มีในส่วนของวรรณยุกต์ เมื่อนำ HTK มาใช้ในการทำการรู้จำเสียงภาษาไทยจึงทำให้เกิดปัญหาเรื่องเสียงวรรณยุกต์ที่ไม่สามารถแยกแยะได้ ทำให้คำบางคำที่มีเสียงคล้ายกันอย่างเช่นคำว่า วางกับว่าง ซึ่งมีความหมายต่างกันแต่การออกเสียงนั้นคล้ายกันมากต่างกันแค่เสียงของวรรณยุกต์เท่านั้น เวลาทำการ Recognize แล้วไม่สามารถแยกแยะเสียงระหว่างคำทั้งสองได้จึงทำให้เกิดความผิดพลาดขึ้น ปัญหานี้ไม่สามารถแก้ไขได้ในตัวโมเดลที่ถูกสร้างมาแล้ว จะทำการสร้างโมเดลให้สามารถทำการวิเคราะห์เสียงวรรณยุกต์ได้นั้นต้องทำในส่วนของ Tools ทั้งหมดที่สำคัญคือในส่วนของการดึงเอา Vector ของเสียงออกมาซึ่งต้องพิจารณาถึงค่าระดับเสียงสูงต่ำ (Pitch) ที่เปลี่ยนแปลงในประโยค ซึ่งนั่นก็เป็นการบ่งบอกว่าคำในประโยคคำนั้นมีเสียงวรรณยุกต์อะไร

ถึงแม้ว่า HTK จะมีเครื่องมือที่ใช้ในการสร้างโมเดลที่หลากหลายแต่ว่าการทำงานอยู่ในลักษณะของ Command Line ซึ่งคำสั่งที่ใช้ในการสร้างระบบการรู้จำของ HTK นั้นยาวมากๆแต่ในการทำงานจริงๆก็ใช้แค่เพียงคำสั่งเดิมๆในการสร้างและทำการ Train วิธีหนึ่งที่จะสามารถช่วยลดความน่าเบื่อของการพิมพ์คำสั่งยาวๆคือการใช้ Batch file ซึ่งจะช่วยให้การทำงานสะดวกขึ้น ตลอดจนโปรแกรมที่ต้องนำมาใช้ในการสร้างระบบรู้จำในครั้งนี้ ที่ต้องทำการเขียนขึ้นมาเองเพื่อที่จะช่วยอำนวยความสะดวกให้กับเรา ซึ่งในการกระทำการกับ File ที่เกี่ยวกับโมเดลนั้นจะเป็น Text file ที่มีขนาดใหญ่มาก จะทำการแก้ไขหรือทำการสร้างด้วยมือนั้นสิ้นเปลืองเวลาเป็นอย่างมาก ซึ่งโปรแกรมที่เขียนขึ้นมาในปริญญานิพนธ์ครั้งนี้ก็ประกอบด้วยหลายโปรแกรมด้วนกันซึ่งจะช่วยอำนวยความสะดวกได้มากและมีอีกบางโปรแกรมที่ไม่ได้ทำการเขียนขึ้นเพราะจะทำให้ Scale ของปริญญานิพนธ์นี้มากเกินไป รายละเอียดของแต่ละโปรแกรมรวมทั้ง Code ของบางโปรแกรมได้แสดงไว้ในภาคผนวก

ในการบันทึกเสียงนั้นควรจะทำในสถานที่ที่เงียบเพียงพอเพราะต้องการคุณภาพของเสียงที่จะใช้เป็นข้อมูลในการ Train โมเดล ถ้าข้อมูลเสียงนั้นมีเสียงรบกวนหรือว่าพูดไม่ชัดหรือออกเสียงไม่ชัด ก็จะถูกจดจำในโมเดลส่งผลให้โมเดลของเราไม่มีคุณภาพ ตลอกจนการตัดข้อมูลประโยคของเสียงพูดที่จะนำมาทำการ Train โมเดลว่าตัดได้ถูกต้องหรือไม่เพราะถ้าไม่แล้วเสียงที่นำมาทำการ Train อาจจะมีเสียงของคำอื่นที่ไม่ได้อยู่ในประโยคนั้นติดมาด้วยก็ได้

ในการนำ Language Model มาช่วยในการเพิ่มความถูกต้องของระบบรู้จำของเรานั้น แม้ว่าจะช่วยเพิ่มความถูกต้องขึ้นมากก็ตาม แต่ระบบต้องการทรัพยากรที่มากมายเพราะไม่เช่นนั้นแล้วในการทำการ Recognize เสียงหรือทำการทดสอบโมเดลแต่ละครั้งจะใช้เวลาหลายนาทีซึ่งปกติแล้วน่าจะตอบสนองต่อมนุษย์ให้ทันท่วงที ซึ่งในการทดลองทำการรู้จำเฉพาะโมเดลทางภาษาอย่างเดียวที่ใช้คำในดิกชันนารี 2000 คำ จะต้องใช้เครื่องที่มีความเร็วสูงมาก และ RAM ถึง 256 MB จึงจะเพียงพอในการใช้งานเพราะไม่เช่นนั้นแล้วการทำระบบการรู้จำเสียงพูดโดยใช้ไวยกรด้วยจะสิ้นเปลืองเวลาเป็นอย่างมาก

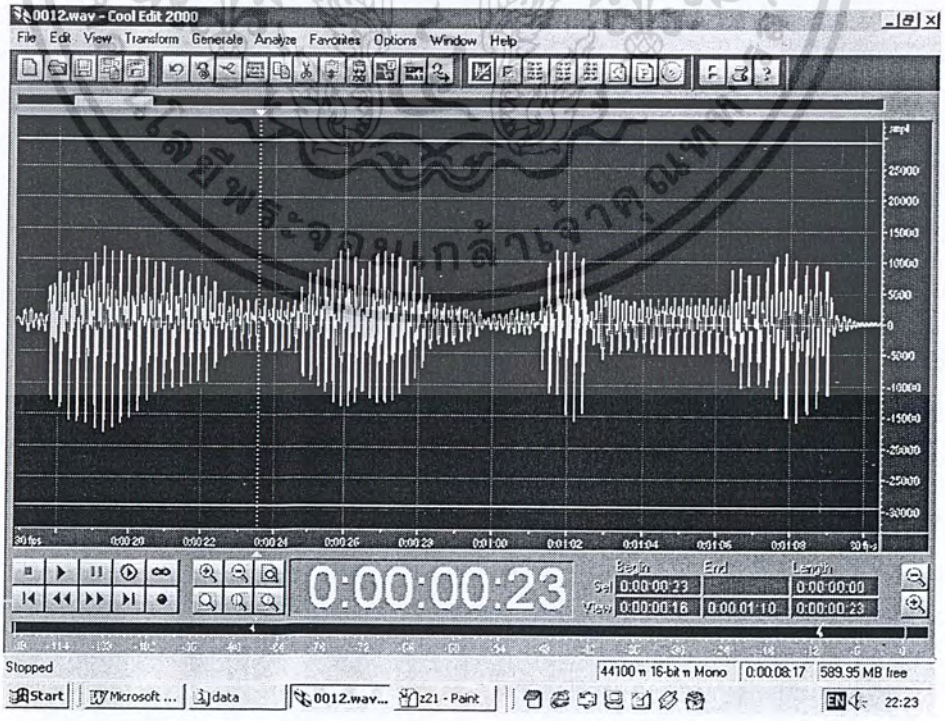
ภาคผนวก

ภาคผนวก ก
ตัวอย่างการตัดและแก้ไขไฟล์เสียงของประโยค

แสดงตัวอย่างการตัดประโยคคำว่า "โครงการหนองงูเห่า" จะได้ดังรูปข้างล่าง

แสดงการตัดเสียงประโยค “โดยเลื่อนกำหนด”

เมื่อทำการตัดแล้วจะได้ลักษณะของ Wave file ดังข้างล่าง

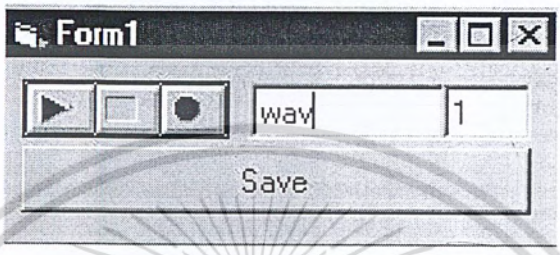


แสดงการตัดเสียงประโยค “ปีสองพันห้าร้อนสี่สิบเจ็ด”

เมื่อทำการตัดแล้วจะได้ลักษณะของ Wave file ดังข้างล่าง

ภาคผนวก ข
ตัวอย่างโปรแกรมและวิธีใช้ที่เขียนขึ้นเพื่อช่วยในโครงงานครั้งนี้

1. โปรแกรมอัดเสียง recorder.exe



ส่วนประกอบของโปรแกรม :

1. ปุ่ม play
2. ปุ่ม stop
3. ปุ่ม record
4. ส่วนแสดงและตั้งชื่อ label wav
5. ส่วนแสดงและตั้งลำดับของไฟล์ 1
6. ปุ่ม save Save

วิธีการใช้งาน :

1. จัดโปรแกรมให้อยู่ในไดเรกทอรีที่จะจัดเก็บไฟล์เสียง โปรแกรมจะต้องมี start.wav ซึ่งเป็นไฟล์เสียงต้นแบบ โดยไฟล์เสียง start.wav จะต้องอยู่ในรูปแบบไฟล์เสียงที่ต้องการอัด ในที่นี้จะใช้ไฟล์ start.wav ที่มีขนาด16 บิท โมโน และมี sampling rate เท่ากับ 44 Mhz
2. เปิดตัวควบคุมระดับความดังของเสียงซึ่งอยู่บน Task Bar หรือ control panel ขึ้นมา เลือก option => properties => Recording และกด ok กดเลือก select ของตัวควบคุมเสียงไมโครโฟน
3. ที่โปรแกรมตั้ง label ของไฟล์เสียงที่ต้องการจะอัด และตั้งหมายเลขของไฟล์เสียงที่ต้องการ

Note: ในกรณีที่มีไฟล์เสียงชื่อซ้ำกับไฟล์เสียงที่กำหนดไว้ในโปรแกรม recorder.exe โปรแกรมจะทำการอัดทับโดยอัตโนมัติ

4. เมื่อต้องการจะอัดเสียง กดปุ่ม record และพูดเสียงผ่านไมโครโฟน เมื่อจบประโยคที่ต้องการ กด stop เพื่อหยุดการบันทึก

5. ถ้าต้องการฟังเสียงที่อัดไปกด play ถ้าต้องการอัดเสียงใหม่กดปุ่ม record อีกครั้ง ถ้าต้องการบันทึกกดปุ่ม save โปรแกรมจะทำการบันทึกไฟล์เสียงตามชื่อที่ได้ตั้งไว้ และหมายเลขของไฟล์เสียงจะเพิ่มขึ้นโดยอัตโนมัติ

2. โปรแกรมเพิ่มคำที่ไม่ซ้ำลงไปใน Dictionary

ชื่อโปรแกรม : DictWatcher (Dw.exe)

หน้าที่หลัก : ใช้ในการใส่คำที่เป็นคำที่ไม่ซ้ำจากรายการคำลงไปใน Dictionary รวมทั้งนับจำนวนคำที่ใส่เข้าไปใหม่ทั้ง

หมดคำทั้งหมดใน Dictionary และก็จำเก็บคำที่ไม่ซ้ำไว้ในไฟล์รวมทั้งคำที่ซ้ำกันทั้งหมดด้วย

Input : ชื่อของ Dictionary และ ชื่อของรายการคำศัพท์

Output : รายการคำใหม่ที่รวมในไฟล์ Dictionary รายการคำที่ไม่ซ้ำในไฟล์ รายการคำที่ไม่ซ้ำในไฟล์และสรุปจำนวน

สถานะของคำศัพท์ทั้งหมดทางจอภาพ

การใช้งาน : พิมพ์ Dw ที่ Command prompt จากนั้นป้อน Input ทั้งหมดที่โปรแกรมต้องการซึ่งโปรแกรมจะถามหา

โดยป้อนไปเรื่อยๆจนครบโปรแกรมก็จะเริ่มทำงานและให้ผลลัพธ์ออกมาโดยอัตโนมัติ

```
program DictWatcher;
var F,F1,F2,F3 : Text;
   s1,s2,dictName,dict  : string;
   flag1,flag2,count,fg : integer;
   ch  : char;
   sameCount,newCount,wordCount,newword :integer;

function cutWord(var Fn : Text;var Flag : integer): string;
var  chr : char;
    str : string;
begin
```

```
    str := '';
    read(Fn,chr);
    while chr = ' ' do read(Fn,chr);
    while (chr <> ' ') and (chr <> #13) and not eof(Fn) do
    begin
        str := str + chr;
        read(Fn,chr);
    end;
    while chr <> #13 do
    begin
        if chr = #26 then
        begin
            flag := 1;
            exit;
        end;
        read(Fn,chr);
    end;
    read(Fn,chr); { read for char #10}
    cutWord := str;
    if eof(Fn) then Flag := 1;
end;

begin
  flag1 := 0;
  writeln('Dict Watcher for HTK (c)Copyright 2000 by HTK group .');
  writeln('This program case sensitive,please carefull !!! ');
  write('Enter dict file name : ');
  readln(dict);
  write('Enter new word file : ');
  readln(dictName);
  Assign(F, dict);
  Assign(F2,dictName);
  Assign(F3,'dupwd.txt');
```

```
Assign(F1,'newwd.txt');
rewrite(F1);
rewrite(F3);
reset(F2);
while flag1 <> 1 do
begin
   s1 := cutWord(F2,flag1);
   if s1 <> '' then wordCount := wordCount+1;
   reset(F);
   while flag2 <> 1 do
   begin
      s2 := cutWord(F,flag2);
      if (fg = 0) and (s2 <> '') then newCount := newCount +1;
      if s2 = s1 then
      begin
         count := count + 1;
         writeln(F3,s1);
      end;
   end;
   close(F);
   fg:= 1;
   if count = 0 then
   begin
      append(F);
      writeln(F,s1);
      writeln(F1,s1);
      newword := newword+1;
      close(F);
   end
   else sameCount:= sameCount+1;
   count := 0;
   flag2 := 0;
end;
```

```
  close(F3);
  close(F2);
  close(F1);
  newcount := newcount - 5 ; { for sp ,sil or not thai word}
  writeln;
  writeln('========================== Result
============================');
  writeln('              Input  : dictPlus',' , ',dictname          );
  writeln('              Output : dictPlus + new word(s)          ');
  writeln('         Dupplicate word list in : dupwd.txt           ');
  writeln('         New word(s) list in     : newwd.txt           ');
  writeln('-------------------------------------------------------');

  writeln('         Previous word in dict  : ',newcount                );
  writeln('         Current word in dict   : ' ,newcount+newword      );
  writeln('         Word in file ',dictname ,'  : ',wordCount          );
  writeln('         Dupplicate word total  : ',samecount           );
  writeln('         New word(s) add total  : ',newword );
  writeln('-------------------------------------------------------');

end.
```

3. โปรแกรมสร้าง Script ให้กับ HCopy

ชื่อโปรแกรม : Codetr.exe

หน้าที่หลัก : ใช้สร้าง Script ให้กับ HCopy ที่มีรายการของไฟล์เสียงที่เป็น Input และ ไฟล์ Output Mfcc

Input : ค่าไดเรกทอรี่ของไฟล์เสียง ค่าเริ่มต้นของลำดับไฟล์ที่เป็นของไฟล์เสียงและไฟล์ MFcc รวมทั้งค่าสุดท้ายของไฟล์

ทั้งสอง

Output : ไฟล์ชื่อ Codetr.scp ซึ่งจะเป็นรายการของไฟล์เสียงและไฟล์ MFcc ที่เป็นตัวเลข

วิธีใช้ : พิมพ์ Codetr ที่ Command prompt จากนั้นป้อน Input ทั้งหมดตามที่โปรแกรมต้องการซึ่งโปรแกรมจะบอกทีละขั้นตอนเป็นลำดับไป

```
program Codetr ;
var s1 : TEXT;
   i,b,e,x,k : integer;
   path : String ;
begin
    writeln('Generate script for Hcopy .....');
    write('Enter wave file path ending with \  : ');
    readln(path);
    write('First wave file value : ');
    readln(b);
    write('End wave file value : ');
    readln(e);
    write('First mfcc file value : ');
    readln(x);
    assign(s1,'codetr.scp');
    rewrite(s1);
    k := x;
    for i := b to e do
    begin
      if (i >= 1) and (i <=9)  then
        writeln(s1,path,'00',i,'.wav  c:\project\Mfccfiles\00',k,'.mfc')
      else if (i >= 10) and (i <= 99) then
        writeln(s1,path,'0',i,'.wav  c:\project\Mfccfiles\0',k,'.mfc');
      else
        writeln(s1, path , i , '.wav  c:\project\Mfccfiles\',k,'.mfc');
         k++;
   end;
   close(s1);
   writeln('Create  codetr.scp  completed .....');
end.
```

4. โปรแกรมแปลงไฟล์ Output ที่ได้จาก HTK ที่อยู่ในเลขฐาน 8 ให้เป็นอักขระ

ชื่อโปรแกรม : Focus.exe

หน้าที่หลัก : ใช้ในการแปลงไฟล์ Output ที่ได้จากการ Test โมเดลและ ไฟล์ที่ได้จากการสร้าง Bigram Language

Model จาก เครื่องมือใน HTK ให้เปลี่ยนจากการแสดงผลในเลขฐานแปดของอักขระต่อกับในไฟล์ให้กลายมา

เป็นอักขระภาษาไทยเป็นคำที่อ่านได้ง่าย

Input : ชื่อของไฟล์ที่แสดงผลในระบบเลขฐาน 8 , และชื่อไฟล์ Output ที่ต้องการให้โปรแกรมเก็บไฟล์ที่แปลงแล้ว

Output : ไฟล์ที่แปลงแล้วสามารถอ่านเข้าใจได้

วิธีใช้งาน : พิมพ์ Focus ที่ Command prompt จากนั้นใส่ Input ที่โปรแกรมต้องการ

```
program Focus;
var s1,s2 : TEXT;
   ch : char;
   sum,count,i : integer;
   st1,st2    : string;
function pow(i : integer) : integer;
var k,powt : integer;
begin
    powt := 1;
    for k := 1 to i do
    begin
        powt := 8 * powt;
    end;
    pow := powt;
end;
begin
    write('Source : ');
    readln(st1);
    write('Destination : ');
```

```
    readln(st2);
    assign(s1,st1);
    assign(s2,st2);
    reset(s1);
    rewrite(s2);
  while(not eof(s1))  do
  begin
    read(s1,ch);
    if (ch = '\') then
    begin
      for  i := 2 downto 0  do
      begin
        read(s1,ch);
        sum := sum + pow(i);
      end;
      write(s2,chr(sum));
      sum := 0;
    end
    else   write(s2,ch);
  end;
  close(s1);
  close(s2);
  writeln('Completed .....');
end.
```

5. โปรแกรมสร้างโมเดลเสียงย่อยภาษาไทยทุกหน่วยเสียง

ชื่อโปรแกรม : mkphones.class (java runtime)

หน้าที่หลัก : ใช้ในการสร้างโมเดลเสียงย่อยให้กับเสียงภาษาไทย

Input : เราไม่ต้องใส่ Input เอง โปรแกรมจะทำการหา File Vfloor และ proto ที่ได้จากการทำ Flat Start

Output : model Monophones ของเสียงภาษาไทยทั้งหมด hmmdefs และ macros

วิธีใช้ : พิมพ์ java mkphones ที่ Command prompt

หมายเหตุ : ไม่สามารถพิมพ์ Source ได้เพราะจะเปลืองเนื้อที่มาก Download ได้ที่ www.freedrive.com

User : Htk2000 Password : htk

6. โปรแกรมสร้างโมเดลของเสียง Shot Pause

ชื่อโปรแกรม : mksp.class

หน้าที่หลัก : ใช้ในการสร้างโมเดลของเสียง Shot pause

Input : hmmdefs

Output : hmmdefs with ShotPause model

วิธีใช้ : พิมพ์ java mksp ที่ Command prompt

หมายเหตุ : ไม่สามารถพิมพ์ Source ได้เพราะจะเปลืองเนื้อที่มาก Download ได้ที่ www.freedrive.com

User : Htk2000 Password : htk

ภาคผนวก ค

Master Level File

File Name : words.mlf (ตัวอย่าง)

File Description : Text File, เก็บ Transcript ของไฟล์เสียงในรูปแบบของ Phone Level

รายการ Transcript ของประโยค จะมีชื่อ " /*.lab" โดยที่ชื่อจะต้องตรงกับไฟล์เสียง

และจะต้องจบประโยคด้วยเครื่องหมายจุดเสมอ สร้างได้โดย Text Editor

```
#!MLF!#
"*/001.lab"
รัฐมนตรี
ว่า
การ
กลาโหม
มะกัน
จุ๊บ
ญวณ
แต่
ไม่
ขอโทษ
.
"*/002.lab"
การ
ค้า
ระหว่าง
มะกัน
กับ
ญวณ
ไม่
ประสบ
ความ
สำเร็จ
.
"*/003.lab"
รัฐมนตรี
การ
ค้า
ญวณ
ประสบ
ความ
สำเร็จ
ใน
การ
เลือก
ตั้ง
ที่
ผ่าน
มา
.
"*/004.lab"
รัฐมนตรี
ญวณ
เดิน
ทาง
ไป
ถึง
อเมริกา
แล้ว
เพื่อ
ตก
ลง
เรื่อง
การ
ค้า
ระหว่าง
ประเทศ
.
"*/005.lab"
รัฐมนตรี
การ
ค้า
มะกัน
ไม่
ลง
เลือก
ตั้ง
แต่
ไป
ญวณ
.
"*/006.lab"
รัฐมนตรี
ว่า
การ
กลาโหม
มะกัน
จุ๊บ
ญวณ
แต่
ไม่
ขอโทษ
.
"*/007.lab"
การ
ค้า
ระหว่าง
มะกัน
กับ
ญวณ
ไม่
ประสบ
ความ
สำเร็จ
.
"*/008.lab"
รัฐมนตรี
การ
ค้า
```

ญวณ
ประสบ
ความ
สำเร็จ
ใน
การ
เลือก
ตั้ง
ที่
ผ่าน
มา
.
"*/009.lab"
รัฐมนตรี
ญวณ
เดิน
ทาง
ไป
ถึง
อเมริกา
แล้ว
เพื่อ
ตก
ลง
เรื่อง
การ
ค้า
ระหว่าง
ประเทศ
.
"*/010.lab"
รัฐมนตรี
การ
ค้า
มะกัน
ไม่
ลง
เลือก
ตั้ง
แต่
ไป
ญวณ
.
"*/013.lab"
ระหว่าง
ประเทศ
.
"*/014.lab"
ไม่
ลง
เลือก
ตั้ง
.
"*/015.lab"
เดิน
ทาง
ไป
ถึง
อเมริกา
.
"*/016.lab"
ตก
ลง
เรื่อง
การ
ค้า
.
"*/017.lab"
การ
เลือก
ตั้ง
.
"*/018.lab"
ไป
ญวณ
.
"*/019.lab"
การ
เลือก
ตั้ง
ที่
ผ่าน
มา
.
"*/020.lab"
การ
ค้า
.
"*/021.lab"
ระหว่าง
.
"*/022.lab"
มะกัน
.
"*/023.lab"
ญวณ
.
"*/024.lab"
เดิน
ทาง
.
"*/025.lab"
ความ
สำเร็จ
เลือก
ตั้ง
รัฐมนตรี
.
"*/027.lab"
ญวณ
ตก
ลง
.
"*/028.lab"
ผ่าน
มา
.
"*/029.lab"
ประเทศ
.
"*/030.lab"
เลือก
ตั้ง
.
"*/031.lab"
ตก
ลง
.
"*/032.lab"
กับ
ญวณ
.
"*/033.lab"
อเมริกา
แล้ว
.
"*/034.lab"
เพื่อ
ตก
ลง

"*/035.lab"
เรื่อง
การ
ค้า
.
"*/036.lab"
ความ
สำเร็จ
.
"*/037.lab"
ไป
ถึง
.
"*/038.lab"
รัฐมนตรี
.
"*/039.lab"
การ
ค้า
ระหว่าง
มะกัน
กับ
ญวณ
ไม่
ประสบ
ความ
สำเร็จ
.
"*/040.lab"
รัฐมนตรี
การ
ค้า
ญวณ
ประสบ
ความ
สำเร็จ

ใน
การ
เลือก
ตั้ง
ที่
ผ่าน
มา
.
"*/041.lab"
รัฐมนตรี
ญวณ
เดิน
ทาง
ไป
ถึง
อเมริกา
แล้ว
เพื่อ
ตก
ลง
เรื่อง
การ
ค้า
ระหว่าง
ประเทศ
.
"*/042.lab"
รัฐมนตรี
การ
ค้า
มะกัน
ไม่
ลง
เลือก
ตั้ง
แต่

ไป
ญวณ
.
"*/043.lab"
ประสบ
ความ
สำเร็จ
.
"*/044.lab"
รัฐมนตรี
การ
ค้า
.
"*/045.lab"
ระหว่าง
ประเทศ
.
"*/046.lab"
ไม่
ลง
เลือก
ตั้ง
.
"*/047.lab"
เดิน
ทาง
ไป
ถึง
อเมริกา
.
"*/048.lab"
ตก
ลง
เรื่อง
การ
ค้า

.
"*/049.lab"
การ
เลือก
ตั้ง
.
"*/050.lab"
ไป
ญวณ
.
"*/051.lab"
การ
เลือก
ตั้ง
ที่
ผ่าน
มา
.
"*/052.lab"
การ
ค้า
.
"*/053.lab"
ระหว่าง
.
"*/054.lab"
มะกัน
.
"*/055.lab"
ญวณ
.
"*/056.lab"
เดิน
ทาง
.
"*/057.lab"

ความ ตก "*/077.lab" "*/089.lab"
สำเร็จ ลง ไม่ ที่
. . . .
"*/058.lab" "*/067.lab" "*/078.lab" "*/090.lab"
รัฐมนตรี เรื่อง ขอโทษ ผ่าน
. การ . .
"*/059.lab" ค้า "*/079.lab" "*/091.lab"
ญวณ . ขอโทษ มา
ตก "*/068.lab" . .
ลง ไป "*/080.lab" "*/092.lab"
. ถึง ค้า เดิน
"*/060.lab" . . .
ผ่าน "*/069.lab" "*/081.lab" "*/093.lab"
มา รัฐมนตรี ระหว่าง ทาง
. . . .
"*/061.lab" "*/070.lab" "*/082.lab" "*/094.lab"
ประเทศ ว่า กับ ไป
. . . .
"*/062.lab" "*/071.lab" "*/083.lab" "*/095.lab"
เลือก การ ประสบ ถึง
ตั้ง . . .
. "*/072.lab" "*/084.lab" "*/096.lab"
"*/063.lab" กลาโหม ความ อเมริกา
ตก . . .
ลง "*/073.lab" "*/085.lab" "*/097.lab"
. มะกัน สำเร็จ แล้ว
"*/064.lab" . . .
กับ "*/074.lab" "*/086.lab" "*/098.lab"
ญวณ จุ๊บ ใน เพื่อ
. . . .
"*/065.lab" "*/075.lab" "*/087.lab" "*/099.lab"
อเมริกา ญวณ เลือก ตก
แล้ว . . .
. "*/076.lab" "*/088.lab" "*/100.lab"
"*/066.lab" แต่ ตั้ง ลง
เพื่อ

"*/101.lab"
เรื่อง
.
"*/102.lab"
ประเทศ
.
"*/103.lab"
รัฐมนตรี
.
"*/104.lab"
ว่า
.
"*/105.lab"
การ
.
"*/106.lab"
กลาโหม
.
"*/107.lab"
มะกัน
.
"*/108.lab"
จุ๊บ
.
"*/109.lab"
ญวณ
.
"*/110.lab"
แต่
.
"*/111.lab"
ไม่
.
"*/112.lab"
ขอ

"*/113.lab"
โทษ
.
"*/114.lab"
ค้า
.
"*/115.lab"
ระหว่าง
.
"*/116.lab"
กับ
.
"*/117.lab"
ประสบ
.
"*/118.lab"
ความ
.
"*/119.lab"
สำเร็จ
.
"*/120.lab"
ใน
.
"*/121.lab"
เลือก
.
"*/122.lab"
ตั้ง
.
"*/123.lab"
ที่
.
"*/124.lab"
ผ่าน

"*/125.lab"
มา
.
"*/126.lab"
เดิน
.
"*/127.lab"
ทาง
.
"*/128.lab"
ไป
.
"*/129.lab"
ถึง
.
"*/130.lab"
อเมริกา
.
"*/131.lab"
แล้ว
.
"*/132.lab"
เพื่อ
.
"*/133.lab"
ตก
.
"*/134.lab"
ลง
.
"*/135.lab"
เรื่อง
.
"*/136.lab"
ประเทศ

"*/137.lab"
มือ
.
"*/138.lab"
ฉัน
.
"*/139.lab"
แกะ
.
"*/140.lab"
เงาะ
.
"*/141.lab"
ที่
.
"*/142.lab"
เปรอะ
.
"*/143.lab"
ใน

ภาคผนวก ง

Backed-off Bi-gram Model และ Standard Lattices Format

Backed-off Bi-gram Language Model

File name : bigfn

File description : เป็นไฟล์ที่เก็บ Bi-gram language model

\data\
ngram 1=908
ngram 2=4956

\1-grams:
-99.999 !ENTER -0.1760
-3.9855 []
-3.9855 sil
-3.9855 silence
-3.9855 กมล -0.2078
-3.9855 กรม -0.2078
-3.6845 กรอบ -0.2903
-3.9855 กระชั้น -0.2990
-3.9855 กระทบ -0.2988
-3.9855 กระทรวง -0.3001
-3.9855 กรุงเทพ -0.3004
-2.9855 กลับ -0.3492
-2.9855 กลาง -0.3574
-3.6845 กลาย -0.2028
-3.3835 กลาโหม -0.5082
-3.6845 กลืน -0.2073
-3.5084 กลุ่ม -0.2069
-3.3835 กล่อง -0.3319
-3.2074 กล่าว -0.2701
-3.6845 กล้า -0.2065
-3.6845 กล่ำ -0.2050
.....
.....
-2.6053 ไร -0.6480
-3.9855 ไร้ -0.2078
-2.6845 ไว้ -0.4894
-2.9855 ไหว้ -0.3880
-3.9855 ไห้ -0.2078
-3.6845 ไอ -0.2077
-0.7141 !EXIT

\2-grams:
-3.5724 !ENTER กระชั้น
-3.5724 !ENTER กระทรวง
-2.8734 !ENTER กลับ
-3.5724 !ENTER กลาง
-3.5724 !ENTER กลาย
-3.0953 !ENTER กลาโหม
-3.5724 !ENTER กลืน
.....
.....

-1.3802 ก่อน ก็
-1.3802 ก่อน จาก
-1.3802 ก่อน เกิด
-1.3802 ก่อน เคย
-0.2041 ก่อน !EXIT
-0.3010 ก้าวหน้า ใน
-0.3802 ขณะ ที่
-1.0792 ขณะ สำรวจ
-1.0792 ขณะ ส้ม
-1.0792 ขณะ !EXIT
-0.6021 ขนาด ของ
-0.6021 ขนาด นี้
-1.0792 ขนาด ใหญ่
-1.0792 ขนาด !EXIT
-1.5798 ขอ ข้อมูล
-1.5798 ขอ จูบ
-1.5798 ขอ ถาม
-1.5798 ขอ ทำ
-1.5798 ขอ มาก
-1.5798 ขอ มือ
-1.1027 ขอ อภัย
-1.5798 ขอ อยู่
-1.5798 ขอ เธอ
-1.5798 ขอ เพียง
-1.5798 ขอ แค่
-1.5798 ขอ แจ้ง
-1.1027 ขอ ให้
-1.5798 ขอ ได้
-0.8808 ขอ !EXIT
-2.2601 ของ กลาง
-1.5611 ของ การ
-2.2601 ของ คน
.....
.....

-2.3766 ไป ใช่
-2.3766 ไป ใช้
-1.8995 ไป ใน
-1.8995 ไป ได้
-1.2005 ไป ไม่
-0.4575 ไป !EXIT
-0.7782 ไฟ ฉาย
-0.3010 ไฟ !EXIT
-2.5185 ไม่ กล้า
-2.5185 ไม่ ขอ
-2.0414 ไม่ ขอโทษ
-2.5185 ไม่ คิด
-2.0414 ไม่ คุย
-2.5185 ไม่ ค่อย
-2.5185 ไม่ ง่าย
-2.5185 ไม่ ซื้ง
-2.0414 ไม่ ดี
-2.5185 ไม่ ตรง
-1.8195 ไม่ ตั้ง
-1.5643 ไม่ ต้อง
-2.0414 ไม่ ทัน
-1.8195 ไม่ นาน
-2.0414 ไม่ น่า
-2.5185 ไม่ น้อย
-2.0414 ไม่ บอก
-1.8195 ไม่ ประสบ
-2.5185 ไม่ ฝัน
-2.5185 ไม่ พอ
-2.5185 ไม่ มอง
-2.5185 ไม่ มั่นใจ
-2.5185 ไม่ มา
-1.8195 ไม่ มาก
-0.9744 ไม่ มี
-2.0414 ไม่ ยอม

-2.5185 ไม่ รัก
-2.5185 ไม่ รับ
-1.1963 ไม่ รู้
-2.0414 ไม่ ร้องไห้
-1.5643 ไม่ ลง
-2.5185 ไม่ ลา
-1.8195 ไม่ ลืม
-2.0414 ไม่ ว่า
-2.5185 ไม่ สน
-2.5185 ไม่ สนใจ
-1.8195 ไม่ สะดวก
-1.8195 ไม่ สามารถ
-2.0414 ไม่ หวั่น
-2.5185 ไม่ อภัย
-1.5643 ไม่ อยาก
-1.8195 ไม่ อาจ
-2.5185 ไม่ อ้วน
-2.5185 ไม่ เขียน
-2.5185 ไม่ เข้า
-1.1568 ไม่ เคย
-2.5185 ไม่ เคลื่อนไหว
-1.6734 ไม่ เจอ
-2.5185 ไม่ เต็ม
-2.0414 ไม่ เบา
-2.0414 ไม่ เปลี่ยน
-1.4046 ไม่ เป็น
-2.5185 ไม่ เสีย
-2.5185 ไม่ เหลือ
-1.8195 ไม่ เห็น
-2.5185 ไม่ แปลก
-1.8195 ไม่ ใช่
-2.5185 ไม่ ใช้
-2.5185 ไม่ ใส่
-2.5185 ไม่ ให้
-2.0414 ไม่ ไกล
-1.5643 ไม่ ได้
-1.8195 ไม่ ไหว
-1.6734 ไม่ !EXIT
-0.3010 ไม้ !EXIT
-1.6812 ไร ก็
-1.2041 ไร จะ
-1.6812 ไร ฉัน
-1.6812 ไร ตัว
-1.6812 ไร ยิ่ง
-1.6812 ไร แค่
-1.6812 ไร ไม่
-0.1899 ไร !EXIT
-0.3010 ไร้ !EXIT
-1.1249 ไว้ ก่อน
-1.6021 ไว้ จะ
-1.6021 ไว้ ชั่ว
-1.6021 ไว้ รอ
-1.6021 ไว้ อย่าง
-1.6021 ไว้ เพียง
-1.6021 ไว้ เพื่อ
-1.1249 ไว้ ใน
-1.6021 ไว้ ใจ
-0.3716 ไว้ !EXIT
-1.3802 ไหร่ เธอ
-0.1498 ไหร่ !EXIT
-1.1461 ไหว เรื่อย
-0.1047 ไหว !EXIT
-1.3010 ไหว้ ใหญ่
-0.3010 ไห้ !EXIT
-0.6021 ไอ อุ่น
-0.6021 ไอ !EXIT

\end\

Standard Lattices Format

File name : Ngram

File description : เป็นไฟล์ที่เก็บ Node และ Arcs เพื่อสร้าง Network ของคำ

```
VERSION=1.0
N=909 L=6770
I=0   W=!NULL
I=1   W=!ENTER
I=2   W=[]
I=3   W=sil
I=4   W=silence
I=5   W=\241\301\305
I=6   W=\241\303\301
I=7   W=\241\303\315\272
I=8
W=\241\303\320\252\321\351\271
I=9   W=\241\303\320\267\272
.....
.....
I=905 W=\344\313\307\351
I=906 W=\344\313\351
I=907 W=\344\315
I=908 W=!EXIT
J=0   S=0   E=2   I=-9.18
J=1   S=0   E=3   I=-9.18
J=2   S=0   E=4   I=-9.18
J=3   S=0   E=5   I=-9.18
J=4   S=0   E=6   I=-9.18
.....
.....
J=6766 S=906 E=908 I=-0.69
J=6767 S=907 E=0   I=-0.48
J=6768 S=907 E=670 I=-1.39
```

ภาคผนวก จ

HMMDEFS และ MACRO Files

File name : hmmdefs

File description : ได้จากการรันโปรแกรม mkphones.class

จะประกอบด้วย phoneme ในรูปแบบของ monophones ของเสียงในภาษาไทย

ประกอบด้วยเสียงพยัญชนะ 20 เสียง สระ 18 เสียง เสียงลงท้าย 8 เสียง

<STREAMINFO> 1 39

<VECSIZE> 39<NULLD><MFCC_D_A_0>

~s "silst"

<MEAN> 39

-5.850524e-001 3.132942e+000 -2.119225e+000 3.239129e+000 5.558130e+000 7.056183e+000

5.800900e+000 3.364599e+000 7.485495e+000 3.970573e+000 -2.229467e+000 1.764078e+000

5.316161e+001 -2.395430e-001 -1.401812e-001 -3.482345e-002 -1.343600e-001 -1.828237e-001 -

9.279607e-002 1.265374e-002 2.843624e-002 -7.911599e-003 1.121342e-001 1.939112e-001

9.258528e-002 -1.298513e-001 8.241982e-003 -6.914359e-002 -2.673463e-002 -3.228448e-002 -

2.945201e-002 1.361520e-003 -3.366423e-002 -1.536038e-002 -1.285580e-002 -2.660343e-002 -

2.927508e-002 -2.717321e-002 5.453653e-002

<VARIANCE> 39

4.184169e+001 3.810726e+001 3.515831e+001 5.246019e+001 4.237586e+001 4.702240e+001

5.832975e+001 5.511185e+001 3.977874e+001 3.320788e+001 4.285920e+001 3.148238e+001

2.732441e+001 1.148735e+000 1.441704e+000 1.440140e+000 1.540350e+000 1.910955e+000

2.028812e+000 2.381921e+000 2.300010e+000 2.064720e+000 1.748358e+000 1.917681e+000

1.443633e+000 6.703622e-001 1.788557e-001 2.654510e-001 2.599337e-001 2.650829e-001

3.326017e-001 3.681949e-001 4.221209e-001 4.194705e-001 3.898365e-001 3.456669e-001 3.620481e-

001 2.817650e-001 1.188951e-001

<GCONST> 1.102343e+002

~h "a"

<BEGINHMM>

<NUMSTATES> 5

<STATE> 2

<MEAN> 39

1.037935e+001 2.240227e+000 -7.927966e+000 -2.399965e+000 2.267147e+000 3.500989e+000 -1.303929e+000 6.677345e-001 8.377736e+000 2.944819e+000 -3.619583e+000 2.369235e-001 6.164901e+001 -1.542137e-001 2.763578e-002 6.773727e-002 7.696456e-002 2.564252e-002 7.495814e-003 1.097854e-001 1.623636e-001 1.845809e-001 4.561010e-002 -1.216577e-001 -9.119052e-002 -1.310395e-001 -2.238306e-001 2.025295e-002 1.235487e-001 6.105766e-002 1.939376e-002 1.697437e-001 3.014801e-001 1.632348e-001 7.050207e-003 3.810751e-002 7.176343e-002 5.115274e-002 -1.705629e-001

<VARIANCE> 39

2.226425e+001 1.885482e+001 3.195686e+001 4.139448e+001 3.192064e+001 4.207428e+001 4.178704e+001 4.399430e+001 4.954310e+001 3.204655e+001 4.074926e+001 3.125069e+001 2.189153e+001 1.370207e+000 1.242241e+000 1.631270e+000 1.599239e+000 1.452334e+000 1.952140e+000 3.183406e+000 2.325503e+000 1.885074e+000 1.371731e+000 1.467405e+000 1.180989e+000 1.036499e+000 1.815048e-001 1.995649e-001 2.470472e-001 2.458731e-001 2.475374e-001 3.253068e-001 4.714702e-001 3.869653e-001 3.453290e-001 2.800824e-001 2.711937e-001 2.191894e-001 1.024685e-001

<GCONST> 1.055234e+002

<STATE> 3

<MEAN> 39

6.607209e+000 3.894606e+000 -4.376304e+000 1.188909e+000 3.719585e+000 5.105258e+000 3.069686e+000 3.916185e+000 9.431617e+000 3.011808e+000 -4.748260e+000 -5.348125e-001 5.788862e+001 -1.741093e+000 3.425438e-001 7.853331e-001 1.732760e-001 1.966133e-001 1.094290e+000 2.035380e+000 1.156356e+000 -3.644966e-002 3.319446e-001 8.767928e-001 5.648524e-001 -1.286023e+000 -9.196945e-002 -7.830803e-002 -1.577250e-001 -2.304028e-001 1.086450e-001 2.448442e-001 2.235669e-002 -9.470357e-002 -1.314313e-001 2.473631e-002 2.666535e-001 1.644613e-001 -2.583943e-002

<VARIANCE> 39

2.904620e+001 2.239210e+001 3.049647e+001 3.997729e+001 2.781959e+001 3.944134e+001 4.755765e+001 5.406343e+001 4.723798e+001 3.231884e+001 3.706867e+001 3.600698e+001 1.902972e+001 2.113280e+000 1.812825e+000 1.978960e+000 1.721003e+000 1.879370e+000 2.436182e+000 3.525663e+000 3.137427e+000 2.505931e+000 2.623910e+000 2.790957e+000 1.929453e+000 1.229622e+000 1.495858e-001 2.314655e-001 2.914065e-001 2.891094e-001

2.789795e-001 4.351345e-001 5.203016e-001 4.389050e-001 4.153801e-001 2.978258e-001 3.601000e-001 2.781501e-001 8.623970e-002

<GCONST> 1.115897e+002

<STATE> 4

<MEAN> 39

-2.893839e+000 2.770155e+000 -4.181466e+000 -2.516830e+000 6.371720e-001 4.462351e+000 6.435575e+000 6.320452e+000 8.057813e+000 5.325186e+000 1.789517e+000 1.899758e+000 5.119106e+001 -4.227448e-001 -7.759210e-002 -7.281882e-002 -1.501482e-001 1.273993e-001 3.386531e-001 2.576013e-001 3.641408e-002 -1.173036e-001 3.762497e-002 2.009331e-001 8.140621e-002 -2.967998e-001 2.089434e-001 -8.010791e-002 -7.265135e-002 1.461074e-002 3.337961e-002 -2.516828e-002 -1.762344e-001 -1.242559e-001 -2.012978e-002 -7.772490e-002 -1.174848e-001 -7.237344e-002 1.752122e-001

<VARIANCE> 39

4.106383e+001 2.202322e+001 2.028051e+001 3.324873e+001 3.218457e+001 3.426801e+001 3.238008e+001 3.374033e+001 2.517778e+001 2.117605e+001 3.377107e+001 2.090559e+001 2.578328e+001 1.143954e+000 1.217287e+000 1.241732e+000 1.441506e+000 1.642888e+000 2.282342e+000 2.371921e+000 1.939424e+000 1.811139e+000 1.527789e+000 1.850960e+000 1.443980e+000 7.027659e-001 1.696848e-001 2.412865e-001 2.194243e-001 2.326174e-001 2.601725e-001 3.038631e-001 4.223074e-001 3.827592e-001 3.236710e-001 2.907711e-001 2.918437e-001 2.488344e-001 1.119044e-001

<GCONST> 1.027909e+002

<TRANSP> 5

0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

0.000000e+000 8.105720e-001 1.894280e-001 0.000000e+000 0.000000e+000

0.000000e+000 0.000000e+000 3.993999e-001 6.006001e-001 0.000000e+000

0.000000e+000 0.000000e+000 0.000000e+000 7.282218e-001 2.717782e-001

0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

<ENDHMM>

.....

.....

~h "ng"

<BEGINHMM>

<NUMSTATES> 5

<STATE> 2

<MEAN> 39

1.046598e+001 2.624215e+000 -7.672698e+000 -3.515716e+000 1.830447e+000 4.616509e+000 -7.584199e-002 2.022038e+000 9.837667e+000 2.944952e+000 -2.796543e+000 2.910759e-001 6.108890e+001 -2.941709e-001 2.674703e-001 8.642793e-002 1.128711e-001 2.507741e-001 1.701570e-001 2.599384e-001 7.724577e-002 -8.028215e-002 8.383998e-003 7.906927e-002 4.119686e-002 -3.134375e-001 -1.521335e-002 -1.604117e-002 2.469762e-002 2.005282e-002 1.050939e-002 1.934756e-002 3.712710e-003 -3.672757e-002 -1.883329e-002 1.477874e-002 -1.011924e-002 -1.309901e-002 2.513473e-003

<VARIANCE> 39

2.510422e+001 1.863590e+001 2.702540e+001 4.182809e+001 3.772622e+001 5.540953e+001 4.269812e+001 5.265923e+001 6.056771e+001 3.465495e+001 3.871650e+001 3.119410e+001 2.146280e+001 3.507399e-001 3.594266e-001 5.347580e-001 5.692442e-001 7.462461e-001 1.028307e+000 9.758495e-001 1.079610e+000 1.188091e+000 9.351677e-001 9.285943e-001 7.252914e-001 2.146256e-001 2.931771e-002 4.611383e-002 7.484816e-002 7.982009e-002 1.057475e-001 1.386184e-001 1.397849e-001 1.580244e-001 1.751530e-001 1.474026e-001 1.437150e-001 1.134899e-001 1.770772e-002

<GCONST> 8.116074e+001

<STATE> 3

<MEAN> 39

4.700715e+000 6.265777e+000 -5.133818e+000 2.383322e-001 9.739019e+000 1.157407e+001 4.445038e+000 6.286310e-001 4.814761e+000 2.181738e+000 -2.875889e+000 -2.372846e+000 5.566575e+001 -1.317897e-001 -3.252311e-003 3.516851e-002 3.594846e-002 3.568809e-002 7.298896e-002 1.256717e-001 1.082244e-001 3.824637e-002 1.078234e-001 8.670662e-002 8.999267e-003 -8.522402e-002 -7.356628e-004 -7.709725e-003 1.796869e-002 7.633194e-003 -3.757316e-002 -4.132889e-002 2.848506e-003 2.728403e-002 1.810267e-002 1.541770e-002 9.192820e-003 -3.665924e-003 3.118352e-003

<VARIANCE> 39

1.077926e+001 1.225045e+001 1.612329e+001 2.180707e+001 2.847672e+001 2.611806e+001 2.666841e+001 2.982854e+001 5.312647e+001 3.040405e+001 2.506372e+001 2.848956e+001 5.835558e+000 3.019639e-001 2.411572e-001 4.831284e-001 4.601494e-001 6.660292e-001 8.400316e-001 9.170418e-001 1.005496e+000 9.238077e-001 8.735898e-001 8.460469e-001 6.669077e-001 1.211667e-001 2.208840e-002 3.171481e-002 6.004279e-002 7.667141e-002 1.023746e-001 1.193182e-001 1.477882e-001 1.540747e-001 1.553711e-001 1.463637e-001 1.374105e-001 1.133667e-001 9.057004e-003

<GCONST> 7.027148e+001

<STATE> 4

<MEAN> 39

-1.507407e+000 6.036583e+000 -3.384722e+000 2.697529e+000 9.272805e+000 9.942971e+000 8.950790e+000 5.241942e+000 9.462034e+000 5.581700e+000 -6.411821e-001 1.746484e+000 5.147574e+001 -2.158448e-001 -3.763546e-002 5.633400e-002 1.738271e-002 -7.322808e-002 -9.155346e-002 9.327431e-002 1.147579e-001 2.705292e-002 8.005272e-002 9.932199e-002 1.657376e-002 -1.465786e-001 5.494158e-003 -1.696192e-002 2.003542e-003 -2.119522e-003 -2.314270e-002 -2.656355e-002 -3.053977e-002 -1.342934e-002 1.760942e-003 -8.034814e-003 -8.686093e-003 5.697279e-005 8.856761e-003

<VARIANCE> 39

1.290880e+001 9.203732e+000 1.664322e+001 1.982804e+001 2.944684e+001 2.376678e+001 2.584878e+001 2.545352e+001 2.992001e+001 2.356526e+001 2.435635e+001 2.093213e+001 6.769605e+000 2.440451e-001 3.071465e-001 6.071458e-001 6.679087e-001 9.950861e-001 1.120436e+000 1.404902e+000 1.172376e+000 1.196518e+000 9.990833e-001 1.041036e+000 8.016793e-001 1.100413e-001 2.593541e-002 4.788437e-002 9.159187e-002 1.127672e-001 1.679313e-001 1.824503e-001 2.401634e-001 2.003181e-001 2.129164e-001 1.813096e-001 1.937516e-001 1.554236e-001 1.220963e-002

<GCONST> 7.595734e+001

<TRANSP> 5

0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

0.000000e+000 7.991189e-001 2.008810e-001 0.000000e+000 0.000000e+000

0.000000e+000 0.000000e+000 9.131974e-001 8.680261e-002 0.000000e+000

0.000000e+000 0.000000e+000 0.000000e+000 9.401765e-001 5.982344e-002

0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

<ENDHMM>

File name : macros

File description : ได้จากการรันโปรแกรม mkphones

ไฟล์จะเก็บค่า variance macro ของไฟล์ hmmdefs ทีอยู่ในรูปของ monophones ไว้

~o

<STREAMINFO> 1 39

<VECSIZE> 39<NULLD><MFCC_D_A_0>

~v "varFloor1"

<VARIANCE> 39

5.903861e-001 4.974293e-001 4.034797e-001 5.183119e-001 5.772285e-001 6.794620e-001 5.797155e-001 5.959650e-001 4.728230e-001 3.473841e-001 4.770831e-001 3.646733e-001 8.556163e-001 1.160656e-002 1.212066e-002 1.069038e-002 1.158692e-002 1.326858e-002 1.586707e-002 1.895927e-002 1.736180e-002 1.471535e-002 1.303877e-002 1.364073e-002 1.179243e-002 2.140805e-002 1.273135e-003 1.870512e-003 1.705693e-003 1.863258e-003 2.151637e-003 2.445135e-003 2.918423e-003 2.783121e-003 2.520676e-003 2.288572e-003 2.278748e-003 2.005932e-003 2.872613e-003

บรรณานุกรม

1. Judith A. Markowitz . (1996) : "*Using Speech Recognition*", Prentice Hall PTR , Uper Saddle River ,New Jersey 07458 ISBN 0-13-186321-5

2. Lawrence Rabiner , Biing-Hwang Juang : "*Fundamental of Speech Recognition*" , Prentice Hall ISBN 0-13-015157-2

3. Steve Young , Dan Kershaw , Julian Odell , Dave Ollason , Valtcho Valtchev , Phil Woodland (1995) : *"The HTK book version 2.2 "* , The Entropic Coperation , Published January 1999

4. จิตรลดา จารุมิศร์ (Chitlada Charumit) , (1999) : "*การออกแบบแบบจำลองในการรู้จำเสียงวรรณยุกต์สำหรับเสียงภาษาไทย*" , วิทยานิพนธ์หลักสูตรปริญญาวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ISBN 974-622-417-1

5. ชาคริต อนันทราวัน : *หลักภาษาไทย* . กรุงเทพมหานคร : โอเดียนสโตร์ . 2524

6. รัฐวรนนท์ รอดศัตรู : "*การสร้างแบบจำลอง Hidden Macov Model สำหรับการรู้จำเสียงภาษาไทยโดยใช้โปรแกรม HTK*" (1999) , วิทยานิพนธ์หลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง