โครงการวิจัยย่อยลำดับที่ 3
เรื่อง ระบบรู้จำเสียงพูดภาษาไทย ปีที่ 5 : การพัฒนาการรู้จำเสียงพูดแบบเป็นประโยค เมื่อมีชุดคำศัพท์ขนาดใหญ่
ระบบรู้จำเสียงพูดต่อเนื่องภาษาไทย

1. ผู้รับผิดชอบโครงการ รองศาสตราจารย์ ดร. สมชาย จิตะพันธ์กุล

2. วัตถุประสงค์ของโครงการ

2.1 เพื่อพัฒนาระบบรู้จำเสียงพูดต่อเนื่องภาษาไทยให้สามารถใช้งานแบบชุดคำศัพท์ขนาดใหญ่และมีความสามารถรองรับการทำงานในรูปแบบที่กำหนดได้

2.2 เพื่อที่จะสามารถนำระบบรู้จำเสียงพูดต่อเนื่องภาษาไทยไปใช้งานในวิจัยขั้นสูงต่อไป

2.3 เพื่อที่จะสามารถพัฒนาระบบรู้จำเสียงพูดต่อเนื่องภาษาไทยให้ใช้ในเชิงพาณิชย์ได้

3. ขอบเขตหรือเป้าหมายของโครงการ

3.1 สร้างฐานข้อมูลเสียงพูดต่อเนื่องภาษาไทยเพื่อนำมาใช้วิเคราะห์ ทดสอบ และสร้างแบบจำลองในการรู้จำได้

3.2 สร้างระบบการรู้จำเสียงพูดต่อเนื่องภาษาไทยแบบชุดคำศัพท์ขนาดใหญ่

3.3 สร้างระบบการรู้จำวรรณยุกต์เสียงพูดต่อเนื่องภาษาไทย

3.4 สร้างระบบการรู้จำทำนองเสียงพูดสำหรับเสียงพูดภาษาไทย

4. ส่วนงานที่ได้ดำเนินการ

โครงการวิจัยระบบรู้จำเสียงพูดต่อเนื่องภาษาไทยแบบชุดคำศัพท์ขนาดใหญ่ที่ได้ดำเนินการวิจัย ตลอด 5 ปีที่ผ่านมาตั้งแต่ปี 2002 ถึง 2007 ได้ดำเนินการศึกษาและวิจัย รวมทั้งพัฒนาส่วนประกอบต่างๆ ที่จำเป็นต้องใช้ในการพัฒนาระบบรู้จำเสียงพูดลักษณะดังกล่าว โดยเริ่มต้นตั้งแต่การเก็บรวบรวมข้อมูลของเสียงพูดต่อเนื่องภาษาไทยที่ใช้ในการเป็นฐานข้อมูลสำหรับทำการวิจัย รวมทั้งพัฒนาโปรแกรมสำหรับวิเคราะห์เสียงพูดที่จำเป็นอย่างยิ่งในการวิเคราะห์คุณลักษณะของฐานข้อมูลเสียงพูดที่ได้เก็บรวบรวมมา จากนั้นจึงหาลักษณะของแบบจำลองเสียงพูดที่เหมาะสมกับระบบรู้จำเสียงพูด และได้ศึกษาและพัฒนาตัวรู้จำเสียงพูด ซึ่งใช้องค์ประกอบต่างๆ ที่ได้มาจากฐานข้อมูล รวมทั้งลักษณะของรูปแบบของประโยคตามที่ผู้ใช้กำหนด นอกจากนี้ในการวิจัยเรื่องระบบรู้จำเสียงพูดสำหรับภาษาไทยได้มีการศึกษาถึงเรื่องของการรู้จำวรรณยุกต์เสียงพูด ซึ่งเป็นลักษณะพิเศษ และเฉพาะของภาษาไทยด้วย รวมถึงทำนองของเสียงพูดของผู้พูด โดยรายละเอียดของส่วนต่างๆ ที่ได้กล่าวมา รวมทั้งการรู้จำเสียงพูดจะได้มีการกล่าวถึงต่อไป

4.1 การสร้างฐานข้อมูลสำหรับระบบรู้จำเสียงพูดต่อเนื่องภาษาไทย

4.1.1 ฐานข้อมูลและการวิเคราะห์

ในเบื้องต้นของการสร้างฐานข้อมูลสำหรับระบบรู้จำเสียงพูด เราจะต้องรู้ว่าลักษณะของเสียงพูดที่ได้บันทึกไว้นั้น แต่ละพยางค์ (syllable) หรือ คำ (word) ในประโยคประกอบไปด้วยข้อมูลของหน่วยย่อยเสียงพูด (phone) ใดบ้าง ซึ่งในโครงสร้างของพยางค์ในภาษาไทยจะประกอบไปด้วยส่วนต่างๆ ของเสียงดังต่อไปนี้

1. พยัญชนะ (consonant - C)
2. สระ (vowel - V)
3. วรรณยุกต์ (tonal - T)
4. ตัวสะกด (final consonant - F)

นั่นคือพยาง (S) จะเขียนอยู่ในรูปแบบดังนี้

S = C V (F) T

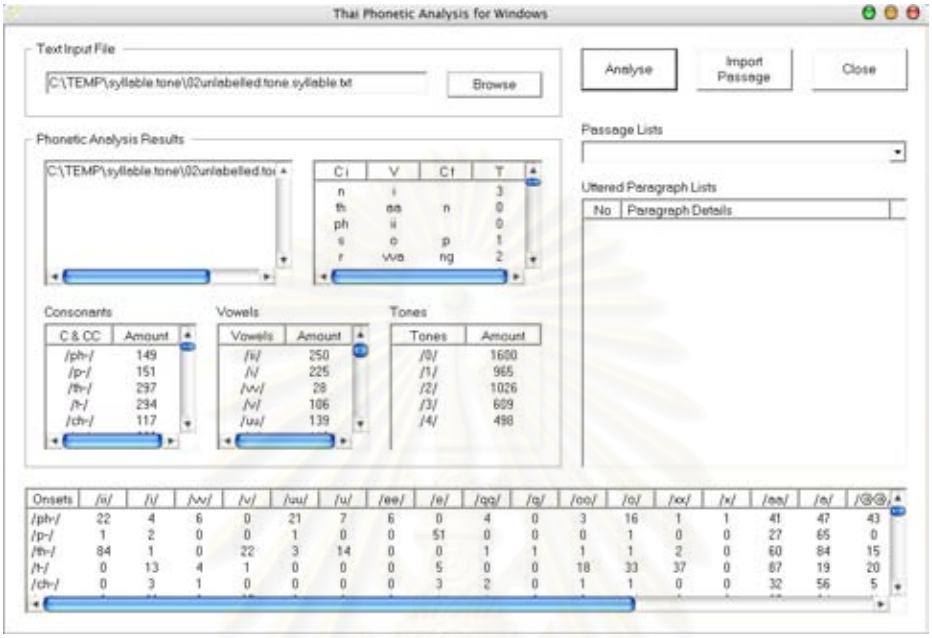
หมายเหตุ ตัวสะกดในแต่ละพยางค์อาจจะมีหรือไม่มีก็ได้ และวรรณยุกต์จะเป็นตัวกำหนดระดับสูงต่ำของพยางค์หรือคำนั้นๆ

เช่น ประโยค “ผมทานข้าวแล้วครับ” จะสามารถเขียนเป็นหน่วยย่อยเสียงพูดได้ดังนี้

“ph o m 4 th aa n 0 kh aa w 2 l xx w 3 khr a p 3”

การสร้างฐานข้อมูลเสียงพูดต่อเนื่องที่ดีจำเป็นต้องอาศัยการวิเคราะห์ และออกแบบรูปประโยคที่จะใช้ในการบันทึกเสียงให้เหมาะสม ในขั้นต้นได้เลือกข้อความจากหนังสือนิทานอีสปเนื่องจากใช้ภาษาที่ง่ายและมีคำภาษาต่างประเทศน้อยมากนำมาวิเคราะห์รูปประโยค และนอกจากนี้ยังได้ออกแบบข้อความเพิ่มเติมไว้สำหรับบันทึกเสียงอีก เพื่อให้เสียงที่บันทึกไว้ครอบคลุมหน่วยเสียงภาษาไทยที่เกิดขึ้นทั้งหมดและมีจำนวนเพียงพอที่จะใช้สร้างแบบจำลองเสียงพูดได้

ซึ่งในการวิจัยนี้ได้พัฒนาโปรแกรมสำหรับวิเคราะห์ข้อความภาษาไทย (Thai Text Analysis) เพื่อใช้ในการสร้างฐานข้อมูล โดยจะมีลักษณะของโปรแกรมที่พัฒนาดังรูปที่ 1



รูปที่ 1 ลักษณะของโปรแกรมวิเคราะห์ข้อความภาษาไทยที่ได้พัฒนาขึ้น

นอกจากนี้ยังได้พัฒนาและสร้างโปรแกรมสำหรับวิเคราะห์เสียงพูด (Thai Speech Analysis) เพื่อใช้กับเสียงพูดที่ได้บันทึกไว้เป็นฐานข้อมูล โดยสามารถพิจารณาหรือวิเคราะห์ข้อมูลได้ดังต่อไปนี้

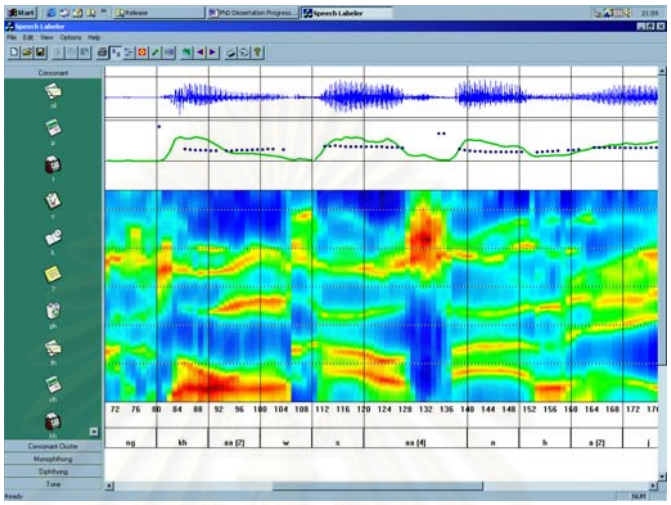
1. แสดงสัญญาณเสียงพูดที่ผ่านขั้นตอนวิธีการเน้นล่วงหน้า (Preemphasis) และการวางกรอบสัญญาณเสียงพูด (Frame Blocking)
2. แสดงพลังงานของเสียงพูด (Energy) ความถี่มูลฐาน (Fundamental Frequency) และ อัตราการตัดผ่านศูนย์ (Zero Crossing Rate)
3. แสดงสเปกโตรแกรม (Spectrogram) ของสัญญาณเสียงพูด
4. แสดงสเปกตรัมของสัญญาณเสียงพูด (Short Time Spectrum)
5. แสดงค่าจริง (Real) ของ Ceptrum

6. แสดงวงกลมหนึ่งหน่วย (Unit Circle) และขั้ว (Pole) ของพหุนามการประมาณพันธะเชิงเส้น (Linear Prediction)
7. แสดงสเปกตรัม 3 มิติของสัญญาณเสียงพูด
8. แสดงความถี่ฟอร์แมนท์ (Formant Frequency)
9. ปรับเปลี่ยนค่าพารามิเตอร์ของกรอบสัญญาณเสียงพูด (Frame Parameters) ได้แก่ การเลือกทำขั้นตอนวิธีการเน้นล่วงหน้า การกำหนดขนาดของวินโดว์ที่ใช้ในการวิเคราะห์ การกำหนดช่วงเวลาของตำแหน่งวินโดว์
10. ปรับเปลี่ยนค่าพารามิเตอร์ของการวิเคราะห์สเปกตรัม (Spectrum Parameters) ได้แก่ การกำหนดอันดับของค่าสัมประสิทธิ์ต่างๆ การกำหนดจำนวนจุดในการวิเคราะห์การแปลงฟูเรียร์อย่างเร็ว
11. ปรับเปลี่ยนค่าพารามิเตอร์ของความถี่ฟอร์แมนท์ (Formant Frequency Parameters) ได้แก่ วิธีการคำนวณความถี่ฟอร์แมนท์
12. ปรับเปลี่ยนค่าพารามิเตอร์ของความถี่มูลฐาน (Fundamental Frequency Parameters)

เนื่องจากการวิเคราะห์เสียงพูดที่ใช้ในการรู้จำ จะต้องทราบข้อมูลของช่วงเวลาที่เกิดขึ้นของคำหรือหน่วยเสียงที่พูดว่าเกิดขึ้นในช่วงเวลาใด ดังนั้นจึงได้จึงได้พัฒนาโปรแกรมเพื่อใช้ในการกำกับเสียงพูดภาษาไทย (Thai Speech Labeler) โดยโปรแกรมดังกล่าวมีความสามารถดังต่อไปนี้

1. แสดงสัญญาณเสียงพูดที่ผ่านขั้นตอนวิธีการเน้นล่วงหน้า (Preemphasis) และการวางกรอบสัญญาณเสียงพูด (Frame Blocking)
2. แสดงพลังงานของเสียงพูด (Energy) ความถี่มูลฐาน (Fundamental Frequency)
3. แสดงสเปกโตรแกรม (Spectrogram) ของสัญญาณเสียงพูด
4. ปรับเปลี่ยนค่าพารามิเตอร์ของกรอบสัญญาณเสียงพูด (Frame Parameters) ได้แก่ การเลือกทำขั้นตอนวิธีการเน้นล่วงหน้า การกำหนดขนาดของวินโดว์ที่ใช้ในการวิเคราะห์ การกำหนดช่วงเวลาของตำแหน่งวินโดว์
5. ปรับเปลี่ยนค่าพารามิเตอร์ของการวิเคราะห์สเปกตรัม (Spectrum Parameters) ได้แก่ การกำหนดอันดับของค่าสัมประสิทธิ์ต่างๆ การกำหนดจำนวนจุดในการวิเคราะห์การแปลงฟูเรียร์อย่างเร็ว

โดยจะมีลักษณะของโปรแกรมที่ได้พัฒนาขึ้นดังรูปที่ 2



รูปที่ 2 ลักษณะของโปรแกรมที่ใช้ในการวิเคราะห์เสียงเพื่อใช้ในการกำกับเสียงพูดที่ได้พัฒนาขึ้น

จากการวิเคราะห์คุณลักษณะของเสียงพูดเพศชายและเพศหญิงจากโปรแกรมวิเคราะห์เสียงพูดที่พัฒนาขึ้นข้างต้นพบว่ามีความแตกต่างกัน จึงมีความจำเป็นที่จะต้องเก็บรวบรวมเสียงพูดของทั้งเพศชายและเพศหญิงด้วย ในขณะนี้ได้เก็บรวบรวมข้อมูลเสียงพูดต่อเนื่องทั้งเพศชายและเพศหญิงจำนวน 30 คนแบ่งเป็นเพศชายจำนวน 15 คนและเพศหญิงจำนวน 15 คน หลังจากบันทึกเสียงพูดแล้วจำเป็นที่จะต้องมีการกำกับเสียงพูด (Labeling) เพื่อบอกว่าเสียงนี้อยู่ในช่วงเวลาใดของเสียงพูดทั้งหมด ข้อมูลที่ได้จากการกำกับเสียงพูดนี้จะใช้สำหรับการสร้างแบบจำลองเสียงพูดจากเสียงพูดต่อเนื่องต่อไป

โดยการสร้างฐานข้อมูลเสียงพูดของระบบรู้จำเสียงพูดต่อเนื่องภาษาไทยของการวิจัยในครั้งนี้นั้น คณะผู้วิจัยได้เล็งเห็นถึงการนำระบบดังกล่าวไปใช้งานจริงในสถานการณ์ต่างๆ ดังนั้นในการวิจัยนี้จึงได้นำตัวอย่างของเสียงพูดในหลายๆ ลักษณะนำมาสร้างเป็นแบบจำลองเสียงพูดนอกเหนือจากเสียงพูดปกติที่พูดผ่านไมโครโฟน เช่น การสร้างฐานข้อมูลของเสียงพูดที่ผ่านระบบโทรศัพท์บ้าน โดยในการบันทึกคณะผู้วิจัยได้นำการ์ดรับสัญญาณโทรศัพท์ไปติดตั้งไว้ที่

คอมพิวเตอร์เพื่อบันทึกเสียงพูดผ่านระบบโทรศัพท์บ้าน โดยที่เสียงพูดดังกล่าวจะถูกนำมาวิเคราะห์เพื่อปรับปรุงระบบบันทึกเสียงพูดผ่านระบบโทรศัพท์และสร้างแบบจำลองเสียงพูดผ่านระบบโทรศัพท์ ให้มีประสิทธิภาพการใช้งานให้ดีที่สุด และได้ทำการบันทึกเสียงพูดต่อเนื่องผ่านระบบโทรศัพท์ทั้งสิ้น 20 คน เป็นผู้พูดเพศชาย 10 คนเพศหญิง 10 คน โดยมีความยาวของเสียงพูดประมาณ 100 ชั่วโมง

นอกจากนี้ในการวิจัยยังได้การสร้างฐานข้อมูลเสียงพูดในระบบโทรศัพท์เคลื่อนที่แบบ CDMA โดยได้มีการบันทึกเสียงพูดจำนวน 300 คน จากผู้พูดเพศชาย 150 คน และเพศหญิงจำนวน 150 คนสามารถแบ่งตามสภาพแวดล้อมได้ดังตารางที่ 1

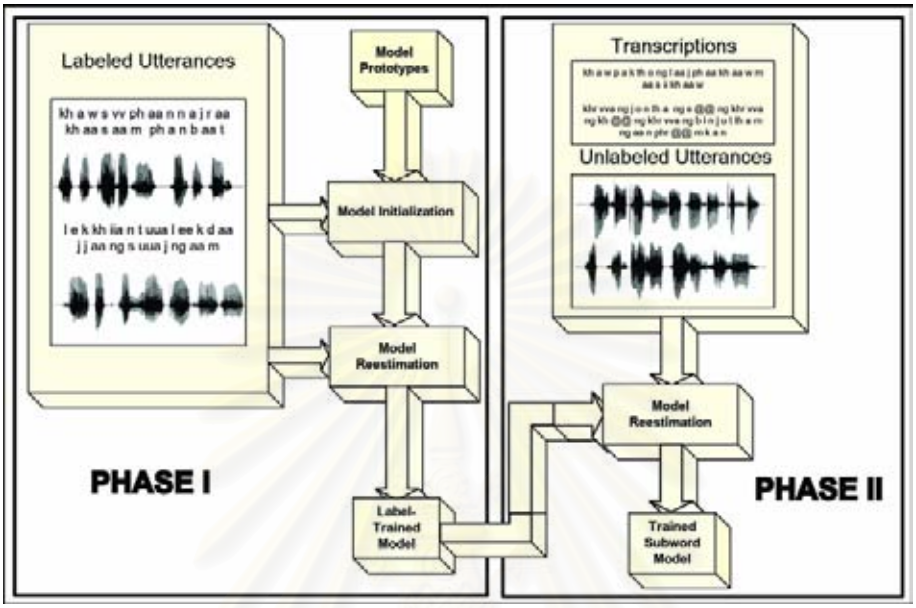
ตารางที่ 1 ลักษณะของฐานข้อมูลเสียงพูดจากระบบโทรศัพท์เคลื่อนที่แบบ CDMA

	สัญญาณรบกวนน้อย (120)	สัญญาณรบกวนปานกลาง (120)	สัญญาณรบกวนมาก (60)
หยุดนิ่ง	ในสำนักงาน , ในบ้าน (90)	ริมถนน , ในห้าง (90)	ริมถนน , ย่านอึกทึก (30)
เคลื่อนที่	รถส่วนตัว (30)	รถปรับอากาศ , รถไฟฟ้า (30)	รถเมล์ (30)

4.1.2 การสร้างแบบจำลองเสียงพูด

ในการสร้างระบบรู้จำเสียงพูดต่อเนื่องจำเป็นที่จะต้องมีการศึกษา และจำลองเชิงกลสัทศาตร์ (Acoustic Modeling) ของเสียงพูดภาษาไทย ในงานวิจัยนี้ได้ใช้ฐานข้อมูลเสียงพูดต่อเนื่องหลายรูปแบบจากที่ได้กล่าวมาข้างต้นมาสร้างแบบจำลองเสียงพูด โดยได้พัฒนาโปรแกรมที่ใช้สร้างแบบจำลองเสียงพูดขึ้น การสร้างแบบจำลองเสียงพูดจะใช้เสียงพูดต่อเนื่องและข้อความจากการกำกับเสียงพูด (Transcription) ที่ได้จากโปรแกรมกำกับหน่วยเสียงที่พัฒนาขึ้นมาเป็นข้อมูลในการสร้างแบบจำลองเชิงสัทศาสตร์ (Acoustic Model) ซึ่งในงานวิจัยนี้จะได้ใช้แบบจำลอง Hidden Markov (Hidden Markov Model : HMM) เป็นรูปแบบในการแสดงหน่วยเสียงเป็นหลัก ลักษณะของแบบจำลองนี้จะบันทึกค่าทางสถิติของหน่วยเสียงที่เกิดขึ้น และนำค่าดังกล่าวไปใช้ใน

การรู้จำ โดยจะมีขั้นตอนการสร้างแบบจำลองดังรูปที่ 3 โดยรายละเอียดของการนำค่าพารามิเตอร์ของแบบจำลอง HMM ไปใช้นั้นจะได้กล่าวถึงในหัวข้อของระบบรู้จำเสียงพูดแบบต่อเนื่องต่อไป



รูปที่ 3 **การพัฒนาระบบสร้างแบบจำลองเสียงพูด**

ลักษณะของ HMM นั้นเป็นเพียงรูปแบบการแสดงค่าทางสถิติของหน่วยเสียงเท่านั้น แต่ข้อมูลที่จะนำมาใช้ในการหาค่าเหล่านั้น จะขึ้นอยู่กับแบบจำลองทางสัทศาสตร์ที่เลือกใช้ ในการวิจัยนี้ได้ทดลองการเลือกใช้แบบจำลองทางสัทศาสตร์ในลักษณะต่างๆ โดยใช้หน่วยเสียงรูปแบบ phone, triphone และ onset-rhyme ซึ่งจะได้จำแนกลักษณะต่างๆ รวมทั้งรูปแบบที่เป็นไปได้ของประเภทหน่วยเสียงดังตารางที่ 2 ดังนี้

ตารางที่ 2 ประเภทของแบบจำลองทางสัทศาสตร์ที่ใช้ในงานวิจัยนี้

Speech Unit	Possible Units
monophone	58
intra-syllable triphone	7,769
inter-syllable triphone	64,475
CI Initial-Final	33I + 200F
CD Initial-Final	297I + 200F
Onset-Rhyme(CORM)	297O + 200R
Onset-Rhyme (PORM)	792O + 200R

จำนวนข้อมูลเสียงพูดที่ใช้สำหรับการสร้างแบบจำลองเสียงพูดและทดสอบในงานวิจัยนี้มี 29,669 ประโยค และ 3,000 ประโยคตามลำดับ โดยบันทึกเสียงจากผู้พูดเพศชายจำนวน 14 คน และเพศหญิงจำนวน 16 คน

โดยในการสร้างแบบจำลองให้กับเสียงพูดนั้น ในการวิจัยได้พัฒนาแบบจำลองรูปแบบที่มีการจำแนกตามเพศของผู้พูด และแบบจำลองที่ไม่ขึ้นกับเพศของผู้พูด โดยจะมีการนำแบบจำลองเชิงภาษา (Language Model) มาใช้ในการรู้จำด้วย ซึ่งผลการรู้จำที่ได้จากการทดสอบสำหรับจะเป็นดังตารางที่ 3-6 ดังนี้

ตารางที่ 3 ผลการรู้จำเสียงพูดของผู้พูดเพศชายแบบขึ้นกับผู้พูดโดยใช้ acoustic model เท่านั้น

Speech Unit	Correction	Accuracy
monophone	29.423	17.300
Intra-syllable triphone	51.878	32.440
Onset-Rhyme (CORMs)	51.941	43.679
Onset-Rhyme (PORMs)	53.823	45.709

ตารางที่ 4 ผลการรู้จำเสียงพูดของผู้พูดเพศหญิงแบบขึ้นกับผู้พูดโดยใช้ acoustic model เท่านั้น

Speech Unit	Correction	Accuracy
monophone	30.080	21.773
intra-syllable triphone	50.919	36.331
Onset-Rhyme (CORMs)	50.469	44.427
Onset-Rhyme (PORMs)	53.607	47.862

ตารางที่ 5 ผลการรู้จำเสียงพูดของผู้พูดเพศชายแบบขึ้นกับผู้พูดโดยใช้ acoustic model และ language model

Speech Unit	Correction	Accuracy
monophone	59.965	57.946
Intra-syllable triphone	69.601	66.678
Onset-Rhyme (CORMs)	75.873	73.305
Onset-Rhyme (PORMs)	77.793	75.486

ตารางที่ 6 ผลการรู้จำเสียงพูดของผู้พูดเพศหญิงแบบขึ้นกับผู้พูดโดยใช้ acoustic model และ language model

Speech Unit	Correction	Accuracy
monophone	60.716	58.666
intra-syllable triphone	70.491	68.205
Onset-Rhyme (CORMs)	76.957	74.837
Onset-Rhyme (PORMs)	79.334	77.338

จากตารางที่ 3-6 จะเห็นว่าการนำแบบจำลองเชิงภาษาเข้ามาในการรู้จำจะให้ผลการรู้จำที่ถูกต้องและน่าเชื่อถือมากกว่า แต่ตารางทุกแบบนั้นแบบจำลองจะแยกเพศของผู้พูด หมายความว่าแต่ละแบบจะต้องให้ผู้พูดเพศเดียวกันเท่านั้นที่พูดได้ โดยถ้าหากจะให้แบบจำลองสามารถใช้ร่วมกับผู้พูดทั้งเพศชายและเพศหญิงได้ จะต้องนำข้อมูลเสียงพูดของทั้งเพศชายและหญิงมา

เรียนรู้ ซึ่งผลการรู้จำเปรียบเทียบของแบบจำลองที่ขึ้นกับเพศของผู้พูดและไม่ขึ้นกับเพศของผู้พูด จะเป็นดังตารางที่ 7-9

ตารางที่ 7 อัตราการรู้จำของแบบจำลองที่ขึ้นกับเพศของผู้พูดและไม่ขึ้นกับเพศของผู้พูด

Gender	Accuracy	
	Gender-dependent	Gender-independent
male	4.5 %	2.7 %
female	5.4 %	3.2 %

การทดลองเพื่อเปรียบเทียบประสิทธิภาพของหน่วยเสียงประเภทต่างๆ โดยใช้การจำลองเชิงกลสัทศาสตร์เพียงอย่างเดียวมีผลการรู้จำแสดงในตารางที่ 8

ตารางที่ 8 อัตราการรู้จำของหน่วยเสียงประเภทต่างๆที่ใช้ acoustic model เพียงอย่างเดียว

Speech unit	Accuracy	
	Speaker-dependent	Speaker-independent
monophone	19.6 %	10.3 %
Intra-syllable triphone	34.8 %	19.6 %
Inter-syllable triphone	40.4 %	26.8 %
CI Initial-Final	38.4 %	26.1 %
CD Initial-Final	42.6 %	29.9 %
CORM	44.6 %	33.5 %
PORM	46.8 %	35.4 %

จากผลการทดลองแสดงให้เห็นว่าแบบจำลอง onset-rhyme (CORM และ PORM) ให้อัตราการรู้จำที่สูงกว่าแบบจำลองหน่วยเสียงอื่นๆ

ส่วนการนำแบบจำลองเชิงภาษามาร่วมในระบบรู้จำเสียงพูดจะทำให้ประสิทธิภาพของระบบดีขึ้นเหมือนที่ได้กล่าวมาแล้วข้างต้น ดังแสดงในตารางที่ 9

ตารางที่ 9 อัตราการรู้จำของหน่วยเสียงประเภทต่างๆที่ใช้ acoustic model และ language model

Speech unit	Accuracy	
	Speaker-dependent	Speaker-independent
monophone	42.8 %	37.0 %
Intra-syllable triphone	61.9 %	47.7 %
Inter-syllable triphone	69.5 %	54.3 %
CI Initial-Final	64.9 %	47.7 %
CD Initial-Final	71.2 %	57.1 %
CORM	73.6 %	61.8 %
PORM	75.4 %	63.8 %

จากผลการวิจัยที่ได้กล่าวมาทั้งหมดหากเราใช้เกณฑ์ในการประเมินผลประสิทธิภาพของหน่วยเสียง 3 ประการคือ accuracy, generalization และ trainability พบว่าแบบจำลอง onset-rhyme (CORM และ PORM) มีประสิทธิภาพที่ดีกว่าแบบจำลองหน่วยเสียงประเภทอื่นๆ

4.1.3 สรุปผลการวิจัยในการสร้างฐานข้อมูลสำหรับระบบรู้จำเสียงพูดต่อเนื่องภาษาไทย

โครงการวิจัยนี้ได้ทำการสร้างฐานข้อมูลเสียงพูดต่อเนื่องภาษาไทยในรูปแบบต่างๆ ทั้งเพศชายและหญิง เพื่อใช้ในการทดสอบและเรียนรู้ระบบรู้จำเสียงพูดภาษาไทยแบบต่อเนื่อง โดยได้ทำการพัฒนาโปรแกรมสำหรับวิเคราะห์ลักษณะข้อมูลทั้งรูปแบบภาษาและเสียงพูด ซึ่งจำเป็นอย่างยิ่งในการสร้างฐานข้อมูลเสียงพูด

โดยลักษณะของแบบจำลองเสียงพูดเชิงสัทศาสตร์ที่จะใช้ในการรู้จำจะใช้ข้อมูลในระดับหน่วยเสียง เนื่องจากมีความยืดหยุ่นกว่าในการใช้งาน และได้ทำการศึกษาถึงประสิทธิภาพในการรู้จำเมื่อนำแบบจำลองเสียงพูดเชิงสัทศาสตร์หลายรูปแบบมาใช้ ซึ่งใช้ HMM เป็นแบบจำลองทางสถิติในการประมวลผลการรู้จำ โดยลักษณะแบบจำลองเสียงพูดเชิงสัทศาสตร์แบบ onset-rhyme ให้ผลการรู้จำที่ดีเมื่อเทียบกับแบบจำลองรูปแบบอื่นๆ ซึ่งประสิทธิภาพในการรู้จำของการใช้แบบจำลองแต่ละแบบจะเพิ่มขึ้นหากเราแยกเพศของผู้พูด

4.2 การพัฒนาระบบรู้จำเสียงพูดแบบต่อเนื่อง

ระบบรู้จำเสียงพูดที่จะศึกษาและพัฒนาในโครงการวิจัยนี้จะใช้แบบจำลอง Hidden Markov ในตัวรู้จำเป็นหลัก แบบจำลองนี้จะจำลองลักษณะของเสียงพูด โดยค่าพารามิเตอร์ต่างๆ ที่ใช้ในแบบจำลองเสียงพูดจะได้มาจากการเรียนรู้คุณลักษณะของเสียงพูด (feature) จากฐานข้อมูลโดยอัลกอริทึม Baum-Welch ซึ่งในระบบการรู้จำเสียงพูดต่อเนื่อง เราจะใช้แบบจำลองของเสียงพูดในระดับของหน่วยเสียง เนื่องจากการเรียนรู้คำทั้งหมดในภาษาไทยหรือแต่ละภาษานั้นเป็นเรื่องที่เป็นไปได้ยากมาก เพราะมีจำนวนคำมหาศาลในแต่ละภาษา รวมถึงอาจจะมีการเกิดคำใหม่ขึ้นได้ตลอดเวลา ดังนั้นการเรียนรู้ในระดับหน่วยเสียง ซึ่งมีจำนวนไม่มากนัก แล้วนำหน่วยเสียงเหล่านี้มารวมเป็นคำจึงเป็นลักษณะของตัวรู้จำที่เหมาะสมกับการปฏิบัติจริงมากที่สุด

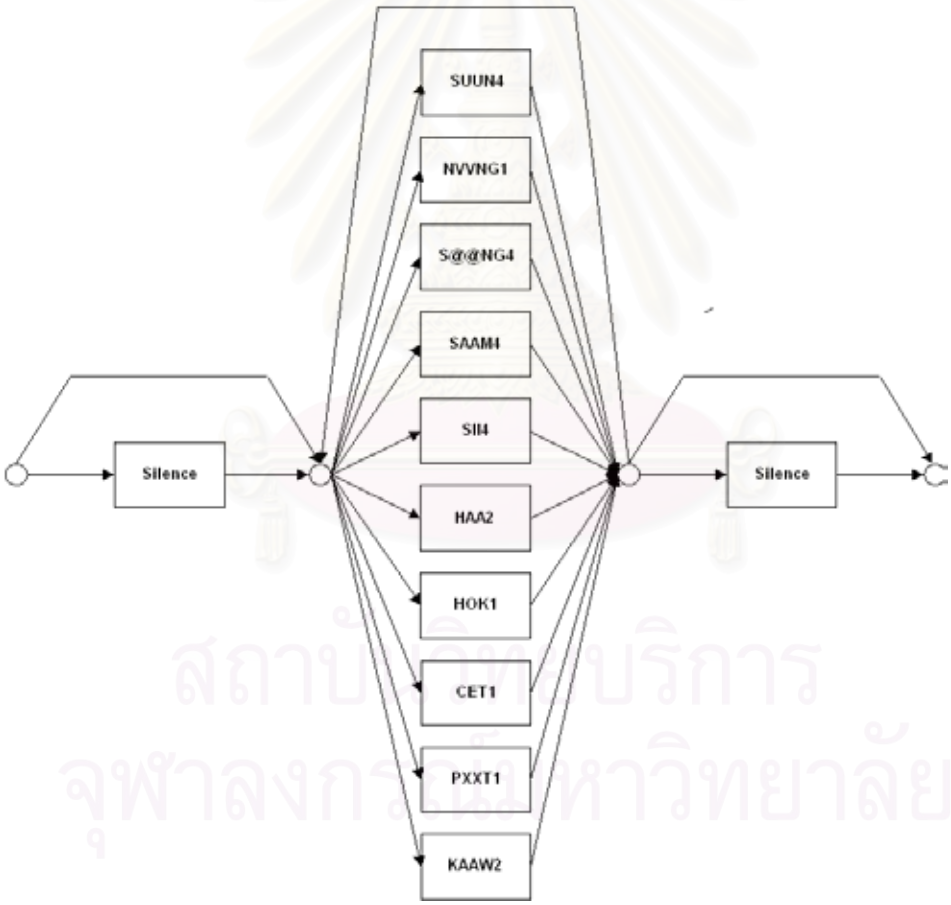
4.2.1 ไวยากรณ์ของรูปประโยค

การที่ประโยคของเสียงพูด คือ คำหลายๆ คำมาต่อกันตามลักษณะของไวยากรณ์ของแต่ละภาษา ดังนั้นเพื่อให้ได้ผลการรู้จำที่มีความถูกต้องและแม่นยำมากยิ่งขึ้น จึงต้องมีไวยากรณ์ของเสียงพูดในตัวรู้จำซึ่งผู้ใช้จะต้องกำหนดให้กับตัวรู้จำ ข้อมูลของไวยากรณ์เป็นส่วนที่สำคัญและมีบทบาทเป็นอย่างมากในตัวรู้จำ ซึ่งลักษณะไวยากรณ์ที่ใช้จะแบ่งออกเป็น 2 แบบ คือ

4.2.1.1 ไวยากรณ์แบบ Finite State

ลักษณะของไวยากรณ์แบบ Finite State (Finite State Grammar) จะเป็นการแสดงถึงการเชื่อมต่อทั้งหมดของกลุ่มคำในรูปประโยคที่กำหนด โดยเราจะมองว่าแต่ละกลุ่มคำที่เกิดขึ้นเป็น state หนึ่งๆ ในไวยากรณ์ และการพิจารณากลุ่มคำถัดไปจะเป็นการเปลี่ยน state จาก state ปัจจุบัน และจะเปลี่ยน state ไปเรื่อยๆ จาก state เริ่มต้น ถึง state สุดท้าย โดยโอกาสในการเปลี่ยน state จาก state ใดๆ ไปยัง state ถัดไปที่เป็นไปได้ทั้งหมดจะถือว่ามีโอกาสเท่ากัน โดยลักษณะของการเชื่อมต่อหรือไวยากรณ์นั้นผู้ใช้จะเป็นคนกำหนดเองตามความเหมาะสม

ยกตัวอย่างเช่น หากเราต้องการสร้างลักษณะประโยคของการระบุหมายเลขโทรศัพท์ อาจจะมีลักษณะของ Finite State ดังรูปที่ 4



รูปที่ 4 ไวยากรณ์แบบ Finite State สำหรับการระบุหมายเลขโทรศัพท์

โดยในที่นี้ จะมีกลุ่มคำอยู่สามชุด คือ 1. กลุ่มคำของตัวเลขตั้งแต่ 0 – 9 2. กลุ่มคำของเสียงเงียบต้นประโยคซึ่งมีตัวเดียว (Silence ตัวซ้ายสุด) และ 3. กลุ่มคำของเสียงเงียบท้ายประโยคซึ่งมีตัวเดียวเช่นกัน (Silence ตัวขวาสุด)

จากไวยากรณ์ดังกล่าวสามารถสร้างเป็นรูปแบบของหมายเลขโทรศัพท์ได้ เช่น

" SUUN4 PXXT1 KAAW2 HOK1 KAAW2 KAAW2 KAAW2
KAAW2 CET1 SII4 CET1 "

หรือในรูปแบบที่มีเสียงเงียบก่อนและหลังหมายเลขโทรศัพท์ได้ เช่น

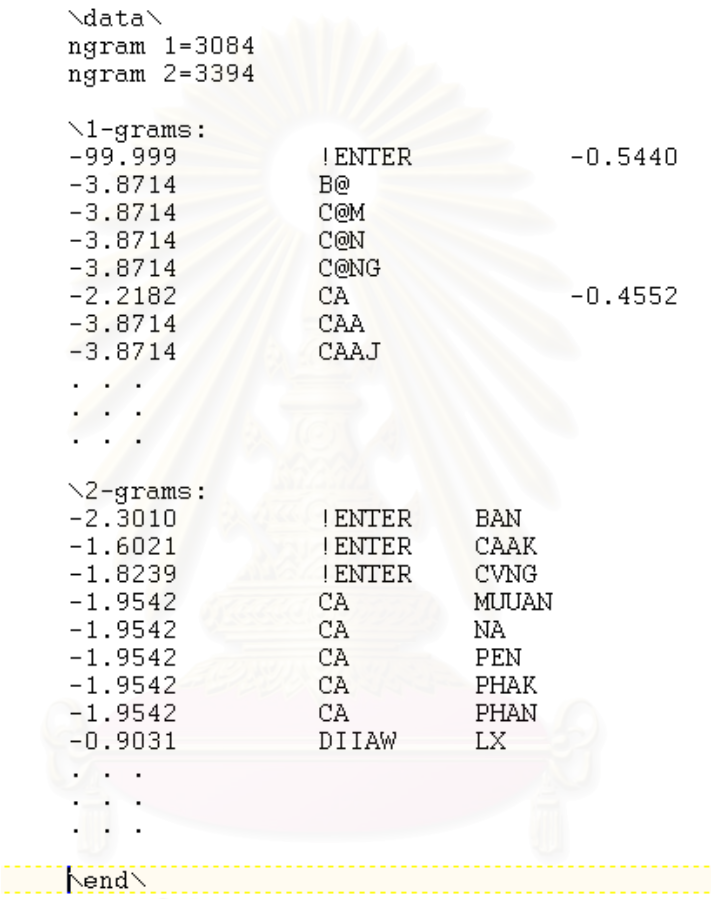
" Silence SUUN4 PXXT1 KAAW2 HOK1 KAAW2 KAAW2 KAAW2
KAAW2 CET1 SII4 CET1 Silence"

4.2.1.2 ไวยากรณ์แบบ N-gram

จากไวยากรณ์แบบ Finite State ที่ได้กล่าวมาข้างต้น จะเห็นว่าลักษณะการเชื่อมต่อของคำจะเป็นไปตามที่ผู้ใช้กำหนดเท่านั้น โดยที่ถ้าหากผู้พูดพูดประโยคหรือคำที่ไม่ได้อยู่ในไวยากรณ์หรือ state ที่เรากำหนด ก็จะไม่สามารถให้ผลการรู้จำที่ถูกต้องได้ การจะนำไวยากรณ์แบบ Finite State ไปใช้ในการรู้จำทั่วๆ ไปแบบไม่จำเพาะเจาะจงลักษณะของการพูดนั้นเป็นเรื่องที่ยาก เนื่องจากว่าเราไม่สามารถเขียนไวยากรณ์เพื่อคลอบคลุมลักษณะการพูดของภาษาทั้งหมดได้

ซึ่งไวยากรณ์แบบ N-gram (N-gram Grammar) จะเป็นการพิจารณาถึงการเชื่อมต่อระหว่างคำที่เกิดขึ้นจากโอกาสการเกิดของคำในภาษานั้นๆ โดยคำที่เกิดขึ้นในภาษาไทยหรือภาษาต่างๆ จะมีความสัมพันธ์กันอยู่ เช่น ลักษณะของรูปประโยคที่มักจะประกอบไปด้วย ประธาน + กริยา + กรรม หรือ คำบางคำมักเกิดคู่กัน เช่น ในปัจจุบันคำว่า "เศรษฐกิจ" มักจะคู่กับ คำว่า "พอ" และ "เพียง" มาก หรือ มักจะพบคำว่า "ความ" "น่า" "จะ" และ "เป็น" ต่อกันบ่อยในภาษาไทย ซึ่งทำให้โอกาสในการเกิดคำว่า "เป็น" จะสูง เมื่อพบคำว่า "ความ" "น่า" และ "จะ" ก่อนหน้า การนำโอกาสในการเกิดของคำในประโยค โดยขึ้นอยู่กับคำที่เกิดก่อนหน้า หรือติดกัน (context) มาใช้นั้น จะทำให้การรู้จำที่ได้มีความน่าเชื่อถือมากยิ่งขึ้น ซึ่งโอกาสการเกิดของแต่ละคำภายใต้ context แต่ละแบบนั้น สามารถทำได้โดยการนำประโยคหรือข้อความในภาษานั้นๆ มาบันทึกค่าทางสถิติหรือโอกาสในการเกิดไว้ ในที่นี้เราจะพิจารณาคำที่เกิดขึ้นต่อกัน N ตัว นั่นคือมี context N-1 ตัว(อันดับ N-1) ซึ่งจะเรียกว่า N-gram ข้อมูลของค่าทางสถิติที่ถูกบันทึกจะพิจารณาในอันดับตั้งแต่ 0 จนถึง N-1 เพื่อให้มีความเหมาะสมในทางปฏิบัติจริง

โดยรูปแบบที่จะแสดงผลการเชื่อมต่อของ N-gram และค่าทางสถิติ เราจะใช้ลักษณะของ N-gram arpa format ที่ออกแบบโดยมหาวิทยาลัย Carnegie Mellon และ มหาวิทยาลัย Cambridge ซึ่งจะมีลักษณะของการแสดงดังรูปที่ 5



```
\data\
ngram 1=3084
ngram 2=3394

\1-grams:
-99.999         !ENTER          -0.5440
-3.8714         B@
-3.8714         C@M
-3.8714         C@N
-3.8714         C@NG
-2.2182         CA              -0.4552
-3.8714         CAA
-3.8714         CAAJ
. . .
. . .
. . .

\2-grams:
-2.3010         !ENTER  BAN
-1.6021         !ENTER  CAAK
-1.8239         !ENTER  CVNG
-1.9542         CA      MUUAN
-1.9542         CA      NA
-1.9542         CA      PEN
-1.9542         CA      PHAK
-1.9542         CA      PHAN
-0.9031         DIIAW   LX
. . .
. . .
. . .

\end\
```

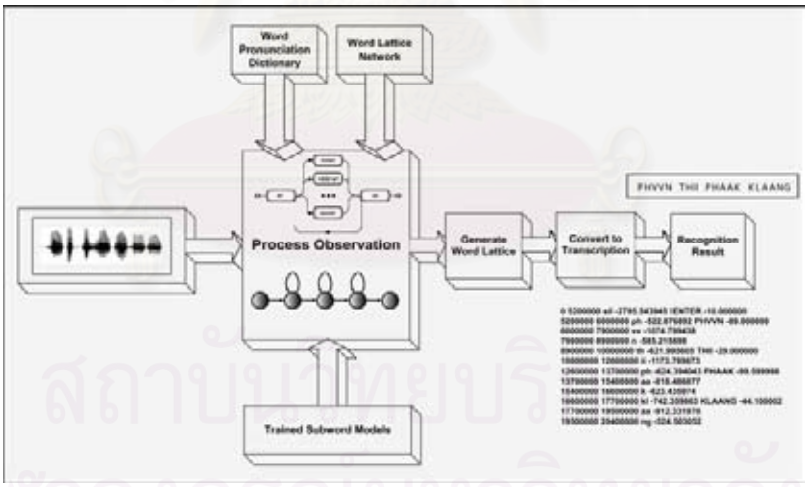
รูปที่ 5 ลักษณะของ N-gram arpa format แบบ 2-gram ที่ใช้ในการวิจัย

จากรูปจะเห็นว่าอันดับของ context ที่เราพิจารณาจะใช้อันดับมากที่สุด คือ 1 (2-gram) โดยข้อมูลของ 1-gram จะเห็นว่ามีทั้งหมด 3 หลัก โดยหลักแรกจะเป็นความน่าจะเป็นของคำ *w* ใดๆ ในรูปของลอการิทึม ส่วนหลักที่สองคือ คำ *w* และหลักที่สาม คือ ค่าถ่วงน้ำหนักแบบ backoff (backoff weight) ซึ่งเป็นค่าที่จะนำไปรวมกับความน่าจะเป็นของคำ *v* ใดๆ เมื่อไม่พบคำ *w* ปรากฏเป็น context อันดับ 1 ของคำ ส่วนคำไหนที่ไม่มีค่าถ่วงน้ำหนักแบบ backoff ก็แสดงว่าไม่ระบุค่าดังกล่าวนั่นเอง

ส่วนข้อมูลของ 2-gram ในที่นี้จะประกอบไปด้วย 3 หลักเช่นกัน โดยหลักแรกจะเป็นความน่าจะเป็นของ 2-gram ในรูปลอการิทึม ส่วน 2 หลักสุดท้ายก็จะแสดงถึงคู่คำที่เชื่อมต่อกันเป็น 2-gram นั่นเอง โดยในที่นี้จะไม่มีค่าถ่วงน้ำหนักแบบ backoff เนื่องจากว่าเราจะพิจารณาคำต่อกันมากที่สุด คือ 2 คำ แต่ถ้าหากเราพิจารณาคำต่อกันมากที่สุด คือ 3 คำ (3-gram) หลักสุดท้ายของ 2-gram ก็อาจจะมีข้อมูลของค่าถ่วงน้ำหนักแบบ backoff ส่วน 3-gram ก็จะมีเพียงแค่ความน่าจะเป็นในรูปของลอการิทึมเท่านั้น โดยในการวิจัยนี้จะใช้ข้อมูลในการพิจารณาไม่เกิน 3-gram ก็ให้ผลการรู้จำเป็นที่น่าพอใจ

4.2.2 ระบบรู้จำเสียงพูดแบบต่อเนื่องในลักษณะของ HTK

ในเบื้องต้นระบบรู้จำเสียงพูดแบบต่อเนื่องที่เลือกใช้ในการวิจัยจะใช้โปรแกรม HTK เป็นต้นแบบ ซึ่งเป็นโปรแกรมรู้จำเสียงที่เป็นที่นิยมในการวิจัยงานด้านนี้ โดยการทำงานของระบบรู้จำสำหรับเสียงพูดแบบต่อเนื่องแบบ HTK จะประกอบไปด้วยองค์ประกอบหลายส่วน ดังรูปที่ 6 ซึ่งในที่นี้จะกล่าวถึงองค์ประกอบแต่ละส่วนโดยสังเขป

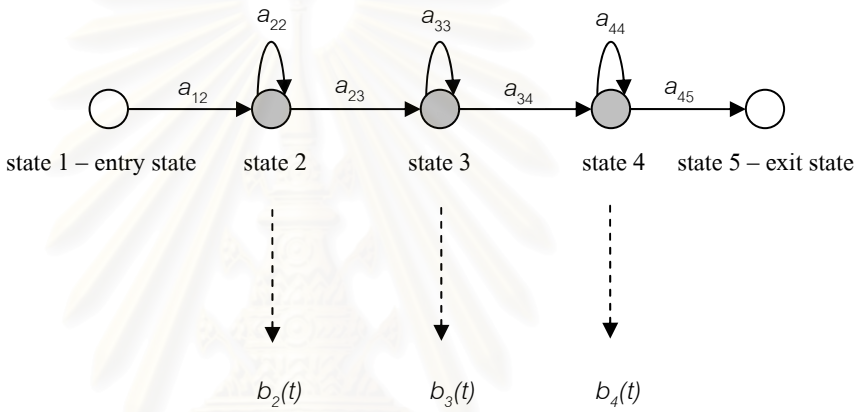


รูปที่ 6 ระบบรู้จำเสียงพูดแบบต่อเนื่องแบบ HTK

4.2.2.1 แบบจำลอง HMM ที่ผ่านการเรียนรู้แล้ว (Trained HMM models)

องค์ประกอบในส่วนนี้เป็นแบบจำลอง HMM ของหน่วยย่อยเสียงพูด ตามลักษณะของหน่วยเสียงที่เราเลือกใช้ (phone , onset-rhyme) ที่ผ่านการเรียนรู้โดยอัลกอริทึม Baum-Welch

ลักษณะของ HMM จะเป็นโมเดลที่ประกอบไปด้วย state ต่างๆ ทั้งหมด *n* state ดังรูปที่ 7 โดยจะมี state ทางเข้า (entry state) คือ state ที่ 1 และ state ทางออก (exit state) คือ state ที่ *n* แต่ละ state จะสามารถเชื่อมถึงกันได้ โดยมีอาร์คเชื่อมต่อกัน และสามารถเปลี่ยน state ได้ โดยมีโอกาสหรือความน่าจะเป็นในการเปลี่ยน state จาก state *i* ไปยัง state *j* ด้วยความน่าจะเป็น a_{ij} และแต่ละ state ตั้งแต่ state ที่ 2 จนถึง *n*-1 จะมีความน่าจะเป็นในการสร้างผลลัพธ์ของเฟรมเสียงที่แต่ละเวลา *t* คือ $b_j(t)$ เมื่อ *j* มีค่าตั้งแต่ 2 ถึง *n*-1 โดยหลังจากให้กำเนิดผลลัพธ์ของเสียงแล้วก็จะเปลี่ยน state ไปพิจารณาที่ state ถัดไปที่มีอาร์คเชื่อมต่อกัน



รูปที่ 7 ลักษณะของการพิจารณาความน่าจะเป็นของโนด HMM

ซึ่งแบบจำลองเหล่านี้ จะถูกนำค่าพารามิเตอร์ต่างๆ ไปใช้ในการคำนวณเพื่อหาผลการรู้จำ ซึ่งจะประกอบไปด้วย

1) พารามิเตอร์ของแต่ละ state

- จำนวนของแบบจำลองทางสถิติที่ใช้ในการประมาณค่าความน่าจะเป็นของการสร้างผลลัพธ์ซึ่งในการวิจัยนี้จะใช้จำนวนแบบจำลองทางสถิติแบบ Gaussian ทั้งหมด 16 ตัว โดยแต่ละแบบจำลอง Gaussian จะมีค่าถ่วงน้ำหนักในการเฉลี่ยเพื่อหาค่าความน่าจะเป็น
- μ_j เป็น เวกเตอร์ของค่าเฉลี่ยของคุณลักษณะของแบบจำลอง HMM ใน state ที่ *j*
- Σ_j เป็น เมตริกซ์ของความแปรปรวนของเวกเตอร์คุณลักษณะของแบบจำลองHMM ใน state ที่ *j*

โดยค่าความน่าจะเป็น $b_j(t)$ ที่แต่ละ state ถ้าหากใช้จำนวนแบบจำลองเพียง 1 ตัว สามารถหาได้จากความน่าจะเป็นของแบบจำลอง Gaussian ดังสมการที่ 1

$$b_j(t) = \frac{1}{\sqrt{(2\pi)^{n_M}\left|\Sigma_j\right|}} e^{-\frac{1}{2}(O_t-\mu_j)^T \Sigma_j^{-1}(O_t-\mu_j)} \qquad (1)$$

แต่ถ้าหากใช้จำนวนแบบจำลอง Gaussian มากกว่า 1 ตัว เช่น ในการวิจัยนี้ที่ใช้ 16 ตัว ค่า $b_j(t)$ จะเกิดจากการนำความสัมพันธ์ในสมการที่ 1 ของทุกๆ แบบจำลอง Gaussian มาเฉลี่ยกันโดยค่าถ่วงน้ำหนักของแต่ละแบบจำลอง

2) ความน่าจะเป็นในการเปลี่ยน state - a_{ij}

การแสดงค่าดังกล่าวจะอยู่ในรูปของเมตริกซ์ A ดังนี้

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ & \dots & & \dots & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

4.2.2.2 **พจนานุกรมเสียงพูดของคำ** (Word Pronouciation Dictionary)

เนื่องจากระบบการรู้จำเสียงพูดแบบต่อเนื่องที่ได้ศึกษาวิจัยจะใช้หน่วยย่อยของเสียงพูดในการแสดงคำแต่ละคำในประโยค ดังนั้นคำต่างๆ ที่มีโอกาสเกิดขึ้นในประโยคที่ผู้พูดจะพูดจึงต้องมีการระบุถึงหน่วยย่อยต่างๆ ที่มาประกอบเป็นคำนั้นๆ รวมทั้งสามารถแสดงคำที่จะเป็นผลลัพธ์ของการรู้จำได้ ซึ่งลักษณะของพจนานุกรมจะเป็นดังรูปที่ 8

CET1	c	e	t
HAA2	h	aa	
HOK1	h	o	k
KAAW2	k	aa	w
NVNG1	n	v	ng
PAUSE	sil		
PXXT1	p	xx	t
S@@NG4	s	@@	ng
SAAM4	s	aa	m
SII1	s	ii	
SUUN4	s	uu	n

รูปที่ 8 ลักษณะพจนานุกรมเสียงพูดของคำ

4.2.2.3 **แลตทิชของคำ** (Word Lattice)

จากที่ได้กล่าวมาในหัวข้อที่แล้วว่า สิ่งหนึ่งที่จำเป็นอย่างยิ่งในระบบรู้จำเสียงพูดแบบต่อเนื่อง คือ ไวยากรณ์ของรูปประโยคที่ผู้ใช้กำหนด ซึ่งไม่ว่าจะเป็นไวยากรณ์แบบ Finite State หรือ ไวยากรณ์แบบ N-gram ก็จะแสดงถึงการเชื่อมต่อของคำที่เกิดขึ้นนั่นเอง โดยที่การแสดงการเชื่อมต่อของคำที่จะถูกนำมาใช้เป็นไวยากรณ์สำหรับตัวรู้จำแบบ HTK นั้น จะอยู่ในรูปแบบของแลตทิชแบบมาตรฐาน (Standard Lattice Format : SLF) ที่จะประกอบไปด้วย โนด และ อาร์ค โดยโนดก็คือ คำที่อยู่ในแลตทิชนั่นเอง ส่วนอาร์ค คือการเชื่อมต่อระหว่างคำ โดยที่อาร์คที่เกิดจากการสร้างแลตทิชจากไวยากรณ์แบบ N-gram จะมีการระบุความน่าจะเป็นในการเชื่อมต่อด้วย ซึ่งลักษณะของแลตทิชที่ได้จากไวยากรณ์แบบ Finite Stateจะเป็นดังรูปที่ 9

```
VERSION=1.0
N=16    L=48
I=0     W=PAUSE
I=1     W=PAUSE
I=2     W=!NULL
I=3     W=SUUN4
I=4     W=KAAW2
.
.
.
I=14    W=!NULL
I=15    W=!NULL

J=0     S=0     E=14
J=1     S=1     E=2
J=2     S=2     E=0
J=3     S=2     E=14
J=4     S=2     E=12
J=5     S=2     E=11
J=6     S=2     E=10
.
.
.
J=46    S=15    E=3
J=47    S=15    E=1
```

รูปที่ 9 แลตทิชที่ได้จากไวยากรณ์แบบ Finite State

และลักษณะของแลตทิชที่ได้จากไวยากรณ์แบบ N-gram จะเป็นดังรูปที่ 10

```
VERSION=1.0
N=16    L=48
I=0     W=PAUSE
I=1     W=PAUSE
I=2     W=!NULL
I=3     W=SUUN4
I=4     W=KAAW2
.
.
.
I=14    W=!NULL
I=15    W=!NULL

J=0     S=0     E=14    l=-2.35
J=1     S=1     E=2     l=-2.96
J=2     S=2     E=0     l=-2.89
J=3     S=2     E=14    l=-0.85
J=4     S=2     E=12    l=-1.36
J=5     S=2     E=11    l=-2.05
J=6     S=2     E=10    l=-2.35
.
.
.
J=46    S=15    E=3     l=-3.08
J=47    S=15    E=1     l=-2.37
```

รูปที่ 10 แลตทิชที่ได้จากไวยากรณ์แบบ N-gram

จากแลตทิชดังกล่าวจะแบ่งด้วยกันออกเป็นสองส่วน คือ การแสดงโนดของแลตทิช หรือคำในไวยากรณ์จะมีดัชนีตามที่ระบุด้วย I = id โดยจะมีคำพิเศษในไวยากรณ์ คือ !NULL ซึ่งเป็นคำที่เป็นเสมือนตัวเชื่อมระหว่างคำ เพื่อลดจำนวนของอาร์คที่ต้องใช้ในการแสดงไวยากรณ์ ส่วนแต่ละอาร์คจะมีดัชนีระบุด้วย J = id โดย S = s และ E = e จะหมายถึงอาร์คดังกล่าวมีโนดที่เป็นจุดเริ่มต้น คือ โนดที่มีดัชนี s และสิ้นสุดที่โนดดัชนี e โดยกรณีแลตทิชที่สร้างจาก N-gram จะเพิ่มความน่าจะเป็นของอาร์คโดยการระบุ I = p ซึ่งค่านี้จะเป็นการระบุแบบลอการิทึมฐานธรรมชาติ เนื่องจากค่าความน่าจะเป็นที่ใช้ในการประมวลผลการรู้จำโดย HMM จะมีค่าน้อยมาก จึงใช้การระบุค่าที่ใช้ในการคำนวณความน่าจะเป็นทั้งหมดในรูปแบบของลอการิทึม

1. รูปแบบการแสดงไวยากรณ์สำหรับตัวรู้จำแบบ HTK

การจะทำให้ตัวรู้จำแบบ HTK สามารถรู้ลักษณะของไวยากรณ์ที่ผู้ใช้ระบุจำเป็นต้องสร้างลักษณะการแสดงของไวยากรณ์เพื่อใช้ในการสร้างแลตทิชสำหรับไวยากรณ์ทั้งสองแบบ สำหรับไวยากรณ์แบบ N-gram จะใช้การแสดงผลโดย arpa format ดังที่ได้กล่าวมาข้างต้น โดยการสร้างแลตทิชจากไวยากรณ์แบบ N-gram จะอ่านข้อมูลจาก arpa format โดยตรง สำหรับคู่คำไหนที่ปรากฏใน 2-gram เราก็จะให้อาร์คที่เชื่อมต่อระหว่างคำดังกล่าวมีค่าความน่าจะเป็น คือ ความน่าจะเป็นของ 2-gram เลย โดยเราจะต้องเผื่อกรณีที่คู่คำดังกล่าวไม่ปรากฏใน 2-gram โดยเราจะกำหนดโนดแลตทิชพิเศษขึ้นมา 1 โนด โดยอาร์คที่เชื่อมต่อมายังโนดดังกล่าว จะมาจากทุกโนดของคำ และจะมีความน่าจะเป็น คือ ค่าถ่วงน้ำหนัก backoff นั่นเอง จากนั้นจึงเชื่อมต่ออาร์คจากโนดพิเศษดังกล่าวไปยังคำทุกคำต่อไป โดยการนำข้อมูลจาก N-gram มาใช้สำหรับตัวรู้จำแบบ HTK จะใช้เพียงแค่ 2-gram เท่านั้น

ส่วนไวยากรณ์แบบ Finite State จะใช้การแสดงโดยการกำหนดตัวแปรให้กับไวยากรณ์ย่อย แล้วเชื่อมต่อตัวแปรเหล่านั้นในไวยากรณ์หลัก และเพื่อให้ลักษณะของไวยากรณ์ที่ผู้ใช้ระบุสามารถมีความซับซ้อนมากยิ่งขึ้นได้ จึงเพิ่มการระบุลักษณะประโยคโดยเพิ่มสัญลักษณ์พิเศษบางตัวไปในไวยากรณ์ ซึ่งสัญลักษณ์พิเศษดังกล่าว จะมีดังต่อไปนี้

- สัญลักษณ์ (.) หมายถึง การระบุว่า คำ หรือ ไวยากรณ์ย่อย ที่อยู่ในวงเล็บจะต้องถูกทำการตีความก่อนที่จะนำไปพิจารณากับคำหรือไวยากรณ์ย่อยอื่นๆ
- สัญลักษณ์ | หรือสัญลักษณ์ alternate หมายถึง การพิจารณาการตีความไวยากรณ์นั้นจะสามารถกระทำกับ คำ หรือ ไวยากรณ์ย่อย ที่มีสัญลักษณ์ดังกล่าวคั่นกลางตัวใดตัวหนึ่ง

- สัญลักษณ์ [.] หรือสัญลักษณ์ enclose หมายถึง คำ หรือ ไวยากรณ์ย่อย ที่อยู่ระหว่างเครื่องหมายดังกล่าวอาจจะเกิดขึ้นในไวยากรณ์ดังกล่าวหนึ่งครั้งหรือไม่ก็ได้
- สัญลักษณ์ < . > หมายถึง คำ หรือ ไวยากรณ์ย่อยที่อยู่ระหว่างเครื่องหมายดังกล่าวอาจจะเกิดขึ้นหนึ่งครั้ง หรือ วนซ้ำตั้งแต่หนึ่งครั้งเป็นต้นไป
- สัญลักษณ์ { . } หมายถึง คำ หรือ ประโยคย่อยที่อยู่ระหว่างเครื่องหมายดังกล่าวอาจจะไม่เกิดขึ้น หรือ เกิดขึ้นหนึ่งครั้ง หรือ วนซ้ำตั้งแต่หนึ่งครั้งเป็นต้นไป

ตัวอย่างเช่น ถ้าหากเราต้องการเขียนไวยากรณ์สำหรับการรู้จำหมายเลขโทรศัพท์ดังที่ได้กล่าวมารูปที่ 4 เราสามารถเขียนลักษณะไวยากรณ์เพื่อสร้างเป็นแลตทิซได้ดังรูปที่ 11ต่อไปนี้

```
$word = NVNG1 | S@@NG4 | SAAM4 | SII1 | HAA2 |
        HOK1 | CET1 | PXXT1 | KAAW2 | SUUN4 |
        PAUSE ;

( [PAUSE] < $word > [PAUSE] )
```

รูปที่ 11 การเขียนไวยากรณ์สำหรับการรู้จำหมายเลขโทรศัพท์

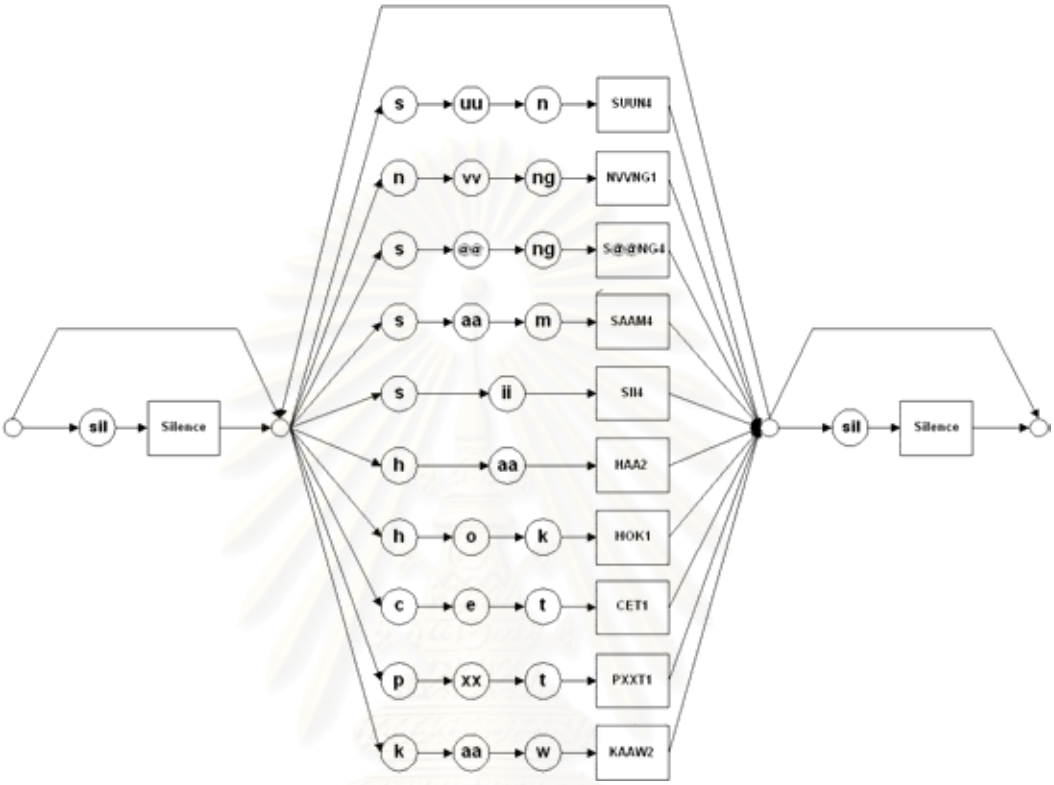
โดยไวยากรณ์หลักของไวยากรณ์แบบ Finite State คือไวยากรณ์บรรทัดสุดท้ายที่อยู่ในวงเล็บ ส่วนบรรทัดอื่นๆ ก่อนหน้าจะเป็นการกำหนดไวยากรณ์ย่อยให้กับตัวแปร ซึ่งจะมีสัญลักษณ์ $ อยู่ข้างหน้าชื่อตัวแปรเสมอ

ซึ่งในการวิจัยนี้ได้พัฒนาโปรแกรมการสร้างแลตทิซ จากไวยากรณ์ทั้งแบบ Finite State ที่ผู้ใช้กำหนด และไวยากรณ์แบบ N-gram เพื่อให้ตัวรู้จำในแบบ HTK สามารถนำแลตทิซดังกล่าวไปใช้ในการรู้จำได้

2. การสร้างโครงข่ายของคำจากแลตทิซ

ซึ่งทางปฏิบัติจริงของตัวรู้จำนั้น จะนำแลตทิซของไวยากรณ์ที่ได้มาขยายเป็นการเชื่อมต่อกันของ HMM เรียกว่าโครงข่ายของคำ (word network) โดยขยายแต่ละโนดของแลตทิซออกเป็นโนดของโครงข่าย คือ HMM ของหน่วยเสียงเชื่อมต่อกัน (HMM node) โดยจะนำข้อมูลของหน่วยเสียงของแต่ละคำมาจากพจนานุกรมเสียงพูดของคำเพื่อสร้างเป็นโครงข่ายของคำ โดยในแต่ละ HMM ก็จะมี state ต่างๆ เชื่อมถึงกันอยู่ และเพื่อให้มีความเหมาะสมในทางปฏิบัติมากยิ่งขึ้น จึงได้กำหนดลักษณะของโนดของโครงข่ายอีกประเภท คือ โนดสิ้นสุดของคำ (word end node) ระบุไว้

ที่ท้ายคำ โดยรายละเอียดต่างๆ ในการประมวลผลจะได้กล่าวถึงต่อไป ซึ่งลักษณะของโครงข่ายของคำที่ได้จากการขยายแลตทิชจากรูปที่ 9 จะเป็นดังรูปที่ 12



รูปที่ 12 โครงข่ายของคำที่ใช้ในการรู้จำหมายเลขโทรศัพท์

จากรูปที่ 12 โนด HMM คือโนดที่เป็นวงกลมแล้วมีชื่อของเสียงอยู่ข้างใน ส่วนโนดสิ้นสุดของคำ คือ โนดที่เป็นสี่เหลี่ยมผืนผ้านั่นเอง

4.2.2.3 การประมวลผลการรู้จำเสียงพูดผ่านโครงข่ายของคำ

การประมวลผลเสียงพูดที่ผ่านกระบวนการประมวลผลล่วงหน้า (preprocessing) และอยู่ในรูปที่ผ่านการนำคุณลักษณะเด่นออกมาแล้ว โดยอยู่ในรูปของ MFCC (Mel Frequency Cepstral Coefficient) ซึ่งในที่นี้จะเรียกแต่ละส่วนของคุณลักษณะเสียงซึ่งเป็นเวกเตอร์ ว่า เฟรม เราจะนำเสียงแต่ละเฟรมเข้าไปพิจารณาในโครงข่ายของคำ ซึ่งจะแบ่งขั้นตอนของการประมวลผล

ออกเป็นสองขั้นตอน คือ 1. การประมวลผลแบบไปข้างหน้า (forward search) และ 2. การประมวลผลแบบย้อนกลับ (reverse search) โดยการประมวลผลแบบไปข้างหน้า สิ่งที่ได้เมื่อประมวลผลเสร็จสิ้น คือ ผลการรู้จำที่ดีที่สุด ส่วนการประมวลผลย้อนกลับจะมีไว้เพื่อใช้ในการหาผลการรู้จำที่ดีที่สุดในอันดับรองลงมา ในกรณีที่ต้องการผลการรู้จำมากกว่า 1 แบบ (N-best)

1. การประมวลผลแบบไปข้างหน้า

การประมวลผลแบบไปข้างหน้าจะเป็นการนำเสียงเข้าไปประมวลผลผ่านโครงข่ายของคำ โดยเสียงแต่ละเฟรมจะถูกนำเข้าไปประมวลผล โดยเริ่มต้นที่เฟรมแรกประมวลผลที่โนด HMM แรกของโครงข่าย ซึ่งแต่ละ HMM state ที่ j ของแต่ละโนดที่มีจำนวน state เท่ากับ n เราจะคำนวณความน่าจะเป็นในการเกิดของผลลัพธ์ของเฟรมเสียงที่ t คือ $b_j(t)$ เมื่อ j มีค่าตั้งแต่ 2 ถึง n-1 โดยที่ j มีค่าเท่ากับ 1 คือ state เริ่มต้นของ HMM และ j มีค่าเท่ากับ n คือ state สิ้นสุดของ HMM ซึ่งจะไม่มีการเกิดผลลัพธ์ โดยเราจะเริ่มต้นจากเวลาที่เฟรมเท่ากับ 1 (t = 1) เป็นต้นไป

จากค่าพารามิเตอร์ต่างๆ ของ HMM เราจะใช้แบบจำลอง Guassian ในการประมาณค่าความน่าจะเป็นจากค่าเฉลี่ย และ ความแปรปรวนของแต่ละแบบจำลอง Guassian ที่ใช้ในการประมาณ (ในการวิจัยใช้จำนวนของแบบจำลอง Guassian ในการหาความน่าจะเป็นเท่ากับ 16 แบบจำลอง) ดังที่ได้กล่าวมาแล้ว จากนั้นเราจึงพิจารณาโอกาสในการเปลี่ยน state จาก state ที่ iไปยัง state ที่ j ของ HMM คือ a_{ij} ไปยังทุก state ที่เป็นไปได้

เมื่อเราพิจารณาการเปลี่ยน state จาก state ที่ i ไปยัง state ที่ j ถ้าหากค่าของ j ที่เป็นไปได้มีค่ามากกว่า 1 ค่า ก็จะเกิดเส้นทางในการพิจารณามากขึ้นทุกครั้งเสมอที่แต่ละเฟรม ยิ่งมี state มากเท่าไหร่ในโครงข่ายของคำ ก็จะยิ่งมีเส้นทางในการพิจารณามากยิ่งขึ้นเท่านั้น ซึ่งการจะพิจารณาเส้นทางทั้งหมดที่จะเกิดขึ้นในโครงข่ายของคำเป็นเรื่องที่ทำให้สิ้นเปลืองการคำนวณและหน่วยความจำในการประมวลผลมาก ดังนั้นในทางปฏิบัติจริงๆ เราจะใช้อัลกอริทึม Viterbi ในการลดเส้นทางการคำนวณลง โดยจะพิจารณาเฉพาะเส้นทางที่มีโอกาสสูงที่สุดที่แต่ละ state เท่านั้น (หรือแค่ m เส้นทางที่มีโอกาสที่ดีที่สุด) ดังนั้นถ้าหากเราพิจารณาที่ แต่ละ state โดยใช้เส้นทางที่มีโอกาสมากที่สุดเท่านั้น ที่แต่ละเฟรมใดๆ เราจะมีจำนวนของเส้นทางมากที่สุดที่เป็นไปได้ในการพิจารณา คือ จำนวนของ state ทั้งหมดในโครงข่ายของคำที่สามารถให้กำเนิดผลลัพธ์ได้เท่านั้น

การพิจารณาโอกาสของเส้นทางที่เกิดขึ้นว่าเส้นทางใดมีโอกาสมากหรือน้อยกว่านั้น เราสามารถหาได้จากการรวมค่า $b_i(t)*a_{ij}$ ตลอดเส้นทางที่พิจารณา ซึ่งก็คือค่าคะแนนของเส้นทางใดๆ ที่เฟรมที่ t นั่นเอง ซึ่งจะได้ความสัมพันธ์ดังสมการที่ 2

$$Score(t) = \prod_{\tau=1}^{t} a_{ij}(\tau) b_j(\tau) \qquad (2)$$

ในที่นี้การพิจารณาแต่ละเส้นทางและคะแนนนั้น เราจะใช้คำว่าการส่งโทเคน (token passing) โดยนิยามของโทเคนในที่นี้ก็คือ เส้นทางที่ผ่านมาทั้งหมด และ คะแนนของเส้นทางดังกล่าวนั่นเอง ซึ่งการเปลี่ยน state ทุกครั้งก็จะมีการส่งผ่านโทเคนไปยัง state ถัดไป แล้วจะนำคะแนนของโทเคนดังกล่าวไปเปรียบเทียบกับคะแนนของโทเคนที่ state ถัดไป โทเคนไหนที่มีคะแนนสูงกว่า ก็จะได้เป็นโทเคนของ state ถัดไปเพื่อนำไปประมวลผลในเฟรมของเสียงถัดไป

ที่แต่ละเฟรมเราก็จะใช้วิธีส่งผ่านทุกโทเคนที่เป็นไปได้ จาก state สู่ state ของ HMM เมื่อเราส่งโทเคนใดๆ ไปยัง state สิ้นสุดของ HMM เราก็จะส่งผ่านโทเคนดังกล่าวไปยัง state เริ่มต้นของ HMM ถัดไป โดยถ้าหากว่าเราได้ส่งโทเคนไปยัง state สุดท้ายของโนด HMM ในโครงข่ายของคำที่อยู่ก่อนโนดสิ้นสุดของคำ *W* ใดๆ จากนั้นจะต้องมีการส่งผ่านโทเคนไปยังโนดสิ้นสุดของคำ *W* ใดๆ ซึ่งการส่งผ่านโทเคนไปยังโนดดังกล่าวหมายถึงว่าขณะนี้เส้นทางที่ผ่านมาทั้งหมดในการประมวลผลนั้นได้เกิดคำ *W* ขึ้นมาเป็นคำล่าสุด ที่โนดสิ้นสุดของคำนี้จะไม่ได้คำนวณคะแนนใดๆ ของโทเคนเลย แต่จะทำหน้าที่ส่งผ่านโทเคนดังกล่าวไปยัง state เริ่มต้นของโนด HMM แรกของคำถัดไป เพื่อประมวลผลเสียงในเวลาถัดไปต่อไป

การส่งผ่านโทเคนในลักษณะที่กล่าวมานั้น จะกระทำทุกเฟรมของเสียงพูดและทุกโทเคนที่เป็นไปได้จนกระทั่งเมื่อประมวลผลครบทุกเฟรมของเสียง โดยโทเคนที่ดีที่สุดที่เป็นของโนดของโครงข่ายคำสุดท้ายในไวยากรณ์จะได้บันทึกเส้นทางที่ดีที่สุดที่เกิดขึ้น ซึ่งคำที่ปรากฏในเส้นทางดังกล่าวก็คือผลการรู้จำที่ดีที่สุดนั่นเอง

ในการนำวิธีดังที่ได้กล่าวข้างต้นมาใช้งานจริงจะต้องคำนึงถึงความเร็วการประมวลผลเป็นสำคัญ ถ้าหากแต่ละคำในโครงข่ายของคำมีเสียงที่มาประกอบเป็นคำซ้ำกัน ดังนั้นการคำนวณแต่ละเวลาหากพิจารณาทุกโนด HMM โดยต้องคำนวณความน่าจะเป็นในการสร้างผลลัพธ์ที่แต่ละ state ($b_j(t)$) ใหม่ทุกครั้งที่มีการพิจารณาคำที่มีเสียงประกอบซ้ำกัน จะทำให้เสียเวลาในการคำนวณมาก และไม่มีความจำเป็นต้องทำเช่นนั้น เนื่องจากที่เวลาเดียวกัน HMM แต่ละตัวจะมีความน่าจะเป็นในการสร้างผลลัพธ์เฟรมของเสียงเท่ากัน ดังนั้นวิธีที่เหมาะสมในทางปฏิบัติ คือ ทุกๆ เวลาแต่ละ HMM จะคำนวณความน่าจะเป็นในการสร้างผลลัพธ์ที่แต่ละ state เพียงแค่ครั้งแรกที่พิจารณาเท่านั้น แล้วจะเก็บค่าดังกล่าวไว้ในบัฟเฟอร์ ซึ่งในเวลานั้นๆ หากมีการพิจารณา HMM ตัวดังกล่าวอีก ก็จะใช้ค่าจากบัฟเฟอร์เลย ซึ่งเป็นวิธีที่เหมาะสมที่สุด

2. การประมวลผลแบบย้อนกลับ

จากการประมวลผลแบบไปข้างหน้าในหัวข้อที่ผ่านมา จะทำให้เราได้ผลการรู้จำที่ดีที่สุดเมื่อประมวลผลเสียงครบทุกเฟรม แต่ถ้าหากเราต้องการผลการรู้จำในอันดับที่รองๆ ลงมาการส่งผ่านโทเคนไปยัง state ต่างๆ ของ HMM เราไม่สามารถเลือกพิจารณาได้เฉพาะโทเคนที่ดีที่สุด

เท่านั้น แต่เราจะต้องพิจารณาและส่งผ่านโทเคนในลำดับรองลงมาด้วย เพื่อเป็นข้อมูลในการค้นหาผลการรู้จำลำดับอื่นๆ โดยการประมวลผลแบบย้อนกลับนั่นเอง

เมื่อประมวลผลเสียงครบทุกเฟรมแล้ว ถ้าหากต้องการคำตอบ N (N > 1) รูปแบบ จะต้องนำโทเคนที่ได้บันทึกไว้ทั้งหมด มาเปรียบเทียบค่าคะแนนของแต่ละเส้นทาง การค้นหาย้อนกลับจากโทเคนทั้งหมดที่ถูกบันทึกไว้ที่โนดของโครงข่ายคำสุดท้าย และค้นหาเส้นทางที่เป็นไปได้ทั้งหมด โดยจะนำเส้นทางที่ถูกบันทึกไว้ที่แต่ละโทเคนมาเปรียบเทียบกันโดยอัลกอริทึม A* (A-Star) ซึ่งเส้นทาง N ลำดับแรกที่มีค่าคะแนนสูงสุดจะเป็นคำตอบของการรู้จำทั้งหมด N แบบที่ผู้ใช้ต้องการ

3. พารามิเตอร์ที่ใช้ในการลดการประมวลผลการรู้จำ

จากที่ได้กล่าวมาในหัวข้อการประมวลผลแบบไปข้างหน้า จะเห็นว่าจำนวนของโทเคนที่เป็นไปได้มากที่สุดในโครงข่ายของคำ จะมีจำนวนเท่ากับจำนวน state ของ HMM ที่สามารถสร้างผลลัพธ์ได้ ซึ่งในกรณีที่ไวยากรณ์ที่ผู้ใช้กำหนด มีขนาดของจำนวนคำที่เกิดขึ้นมาก เช่น มีจำนวนคำหรือโนดของแลตทิซเท่ากับ 3,000 โนด แล้วแต่ละคำมีจำนวนหน่วยเสียงเฉลี่ยเท่ากับ 5 หน่วยเสียง แล้วแต่ละแบบจำลอง HMM ที่แทนหน่วยเสียงมีจำนวนของ state ที่สร้างผลลัพธ์ได้คือ 6 state ดังนั้นจำนวนโทเคนที่เป็นไปได้มากที่สุดในการพิจารณาเส้นทางที่ดีที่สุดแต่ละครั้ง จะมากถึง 3000 x 5 x 6 = 900,000 โทเคน ซึ่งมีจำนวนมาก ทำให้เปลืองหน่วยความจำ รวมทั้งทำให้มีเส้นทางในการพิจารณามากจนเกินไป

ดังนั้นการตัดบางเส้นทางที่ไม่จำเป็นออกไปในการพิจารณาจึงเป็นการเหมาะสม โดยเส้นทางหรือโทเคนไหนที่มีคะแนนต่ำๆ ก็ไม่ควรนำมาพิจารณา ซึ่งในโปรแกรมการรู้จำเสียงพูดแบบต่อเนื่องแบบ HTK ผู้ใช้สามารถกำหนดความกว้างบีม (beamwidth) ของช่วงคะแนนที่จะพิจารณาได้ โดยเราจะพิจารณาให้ความกว้างช่วงคะแนนของบีมนั้นพิจารณาจากโทเคนที่มีคะแนนที่ดีที่สุดนั่นเอง โดยโทเคนใดที่มีคะแนนต่ำกว่าช่วงนั้นลงไป เราจะไม่นำไปพิจารณาในในการประมวลผลเสียงในเวลาถัดไป ความกว้างของบีมจะแบ่งด้วยกันออกเป็น 2 ค่าพารามิเตอร์ คือ

- ความกว้างบีมของคะแนนที่ HMM state ซึ่ง โทเคนที่ HMM state ใดมีค่าคะแนนต่ำกว่าช่วงคะแนนความกว้างที่กำหนดจากโทเคนที่ดีที่สุดที่ HMM state เราก็จะไม่พิจารณาโทเคนนั้น

- ความกว้างบีมของคะแนนที่โนดสิ้นสุดของคำ ซึ่ง โทเคนที่จะถูกส่งจาก HMM state ไปยังโนดสิ้นสุดของคำใด มีค่าคะแนนต่ำกว่าช่วงคะแนนความกว้างที่กำหนดจากโทเคนที่ดีที่สุดที่ถูกส่งไปยังโนดสิ้นสุดของคำ เราก็จะไม่พิจารณาโทเคนนั้น

ซึ่งการกำหนดช่วงบีมที่กว้างเกินไปก็จะทำให้เราต้องพิจารณาโทเคนเยอะเกินจำเป็น แต่ถ้าเรากำหนดให้บีมแคบจนเกินไปก็จะทำให้ตัดโทเคนที่เหมาะสมบางตัวออกไปในการพิจารณา ดังนั้นเราจึงควรกำหนดช่วงของบีมให้มีค่าที่เหมาะสม เพื่อให้มีประสิทธิภาพมากที่สุดในการรู้จำ

4.2.3 ระบบรู้จำเสียงพูดแบบต่อเนื่องในลักษณะของ Julius

จากหัวข้อที่ผ่านมา โครงการวิจัยนี้ได้ศึกษาและพัฒนาตัวรู้จำในแบบของโปรแกรม HTK ซึ่งโปรแกรมดังกล่าวเป็นที่นิยมในงานวิจัยด้านการรู้จำเสียงพูด แต่ในกรณีที่จำนวนคำศัพท์ที่พิจารณามีจำนวนมากในไวยากรณ์ (มากกว่า 3,000 คำ) ก็อาจทำให้การประมวลผลช้าได้ แม้ว่าเราจะกำหนดค่าความกว้างบีมของคะแนนให้มีค่าไม่มากแล้วก็ตาม ทั้งนี้เนื่องจากลักษณะโครงสร้างข้อมูลของไวยากรณ์เวลาขยายเป็นโครงข่ายของคำยังไม่มีประสิทธิภาพเพียงพอ เพราะคำที่เกิดขึ้นบางคำอาจจะมีหน่วยเสียงหรือ HMM ร่วมกัน ถ้าหากเราสามารถพิจารณา HMM หน่วยเสียงของคำเหล่านั้นร่วมกัน ก็จะทำให้สามารถลดจำนวนการประมวลผลลงได้ โดยลักษณะโครงสร้างข้อมูลที่ใช้ลักษณะความสัมพันธ์ของหน่วยเสียงที่เกิดขึ้นร่วมกันของคำ คือ ลักษณะต้นไม้แบบ lexicon (lexicon tree) ซึ่งโปรแกรม Julius เป็นโปรแกรมรู้จำเสียงพูดที่ใช้ลักษณะโครงสร้างดังกล่าวในการรู้จำ โดยองค์ประกอบต่างๆ ของระบบรู้จำแบบในโปรแกรม Julius จะมีลักษณะเหมือนกับระบบรู้จำแบบ HTK เกือบทั้งสิ้น แต่จะมีแค่ลักษณะของโครงสร้างของคำที่พิจารณาเวลาประมวลผลแตกต่างออกไป เนื่องจากได้นำโครงสร้างต้นไม้แบบ lexicon มาประยุกต์ใช้นั่นเอง

4.2.3.1 โครงสร้างข้อมูลต้นไม้แบบ Lexicon

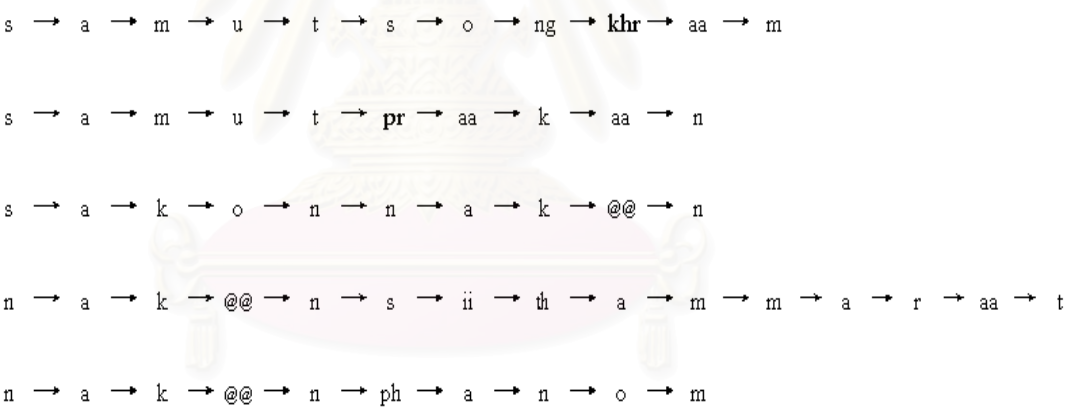
โครงสร้างข้อมูลต้นไม้แบบ lexicon เป็นลักษณะโครงสร้างข้อมูลของคำที่เตรียมไว้ในพจนานุกรม ซึ่งแต่ละคำจะถูกขยายออกเป็น HMM ที่แสดงหน่วยย่อยของเสียง ลักษณะของเสียงที่มาประกอบเป็นคำอาจจะมีบางคำที่มีเสียงเริ่มต้นของคำ (prefix phone) ซ้ำกับคำอื่น ดังนั้นเราจึงสามารถนำคำที่มีเสียงเริ่มต้นซ้ำกันมาสร้างรวมกันเป็นต้นไม้แบบ lexicon ได้ ซึ่งทำให้จำนวน state ของ HMM (จากนี้จะแทน state ของ HMM ว่า โนดของต้นไม้) ที่ต้องใช้ในการคำนวณที่แต่ละเวลาลดลง ยิ่งมีคำที่มีเสียงเริ่มต้นของคำซ้ำกันมากเท่าไหร่ในพจนานุกรม ก็จะยิ่งสามารถใช้

ประโยชน์ของลักษณะโครงสร้างต้นไม้แบบ lexicon ในการลดการคำนวณได้มากเท่านั้น ทำให้การรู้จำเสียงพูดเมื่อมีจำนวนคำในพจนานุกรมมากจึงเหมาะสมที่จะนำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้

ยกตัวอย่าง เช่น หากเราต้องการรู้จำชื่อจังหวัดในประเทศไทย ในพจนานุกรมก็จะมีคำเหล่านี้อยู่ คือ

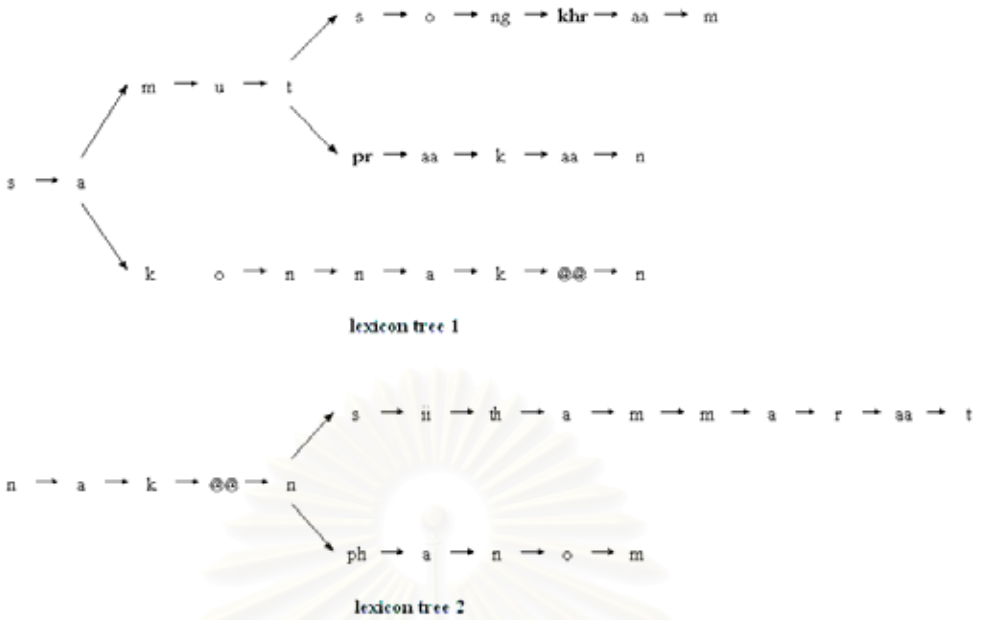
สมุทรสงคราม	s a m u t s o ng khr aa m
สมุทรปราการ	s a m u t pr aa k aa n
สกลนคร	s a k o n n a kh @@ n
นครพนม	n a kh @@ n ph a n o m
นครราชสีมา	n a kh @@ n r aa t ch a s ii m aa
นครศรีธรรมราช	n a kh @@ n s ii th a m r aa t

ถ้าหากไม่มีการนำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้ในการรู้จำ ชุดคำดังกล่าวจะมีลักษณะโครงสร้างเป็นดังนี้



รูปที่ 13 โครงสร้างของคำเมื่อไม่ได้นำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้ในการรู้จำ

จากรูปที่ 13 จะเห็นว่าถ้าหากไม่นำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้จะมีจำนวนของหน่วยย่อยเสียงพูด (HMM) ที่ต้องพิจารณามากถึง 56 ตัว โดยถ้าหาก HMM แต่ละตัวมีจำนวน โนดที่ต้องพิจารณาโดยเฉลี่ยคือ 3 โนด (ไม่คิด state เริ่มต้น และ state สิ้นสุด) จะมีจำนวนโนด ที่ต้องพิจารณามากถึง 3*56 = 168 โนด แต่ถ้าหากเรานำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้กับการรู้จำ ลักษณะโครงสร้างของชุดคำดังกล่าวจะเป็นดังรูปที่ 14



รูปที่ 14 โครงสร้างของคำเมื่อมีการนำต้นไม้แบบ lexicon มาใช้ในการรู้จำ

จากรูปที่ 14 เราจะได้โครงสร้างต้นไม้แบบ lexicon จำนวน 2 ต้น แต่ละต้นไม้ชื่อจังหวัด แต่ละตัวจะใช้ HMM ร่วมกัน โดยจำนวนของ HMM ที่ต้องพิจารณาที่แต่ละเวลาในการประมวลผลสำหรับโครงสร้างต้นไม้แบบ lexicon ในรูปที่ 13 จะลดจำนวน HMM จาก 56 ตัว (168 โนด) เหลือเพียง 44 ตัว (132 โนด) เท่านั้น หากในแต่ละกลุ่มคำมีคำที่สามารถใช้ HMM ร่วมกันได้มากเท่าไหร่ การนำโครงสร้างต้นไม้แบบ lexicon มาใช้ในการรู้จำก็จะยิ่งมีประสิทธิภาพในการลดการคำนวณลงมากเท่านั้น แต่สิ่งที่เกิดขึ้นตามมาเมื่อเราลดจำนวน HMM พิจารณาลงไป ก็คือ จำนวนเส้นทางที่เป็นไปได้ในต้นไม้นั้นๆ ก็จะลดลงไปจากเดิมด้วย โดยจากตัวอย่างนี้ในรูปที่ 13 เมื่อพิจารณาโทเคนหรือเส้นทางที่ดีที่สุดจะมีได้ถึง 168 เส้นทาง (เท่ากับจำนวนโนด) แต่เมื่อเราพิจารณาเป็นต้นไม้ lexicon ดังรูปที่ 14 แล้ว จะลดลงเหลือเพียงแค่ 132 เส้นทางเท่านั้น นั่นคือแต่ละคำในต้นไม้จะต้องมีการใช้เส้นทางบางส่วนร่วมกัน ซึ่งในการรวมคำเป็นต้นไม้ lexicon จะต้องคำนึงถึงผลดังกล่าวด้วย

ลักษณะของต้นไม้แบบ lexicon นั้นจะจำแนกด้วยกันออกเป็น 2 แบบ ตามลักษณะของคำที่รวมอยู่ในต้นไม้ lexicon ดังนี้

- ต้นไม้แบบหลายคำ (multiple word tree) คือ ต้นไม้ lexicon ที่ประกอบไปด้วยคำที่มีเสียงต้น (prefix phone) ซ้ำกันมากกว่า 1 คำ

- ต้นไม้แบบคำเดียว (single word tree) คือ ต้นไม้ lexicon ที่ประกอบไปด้วยคำเพียงคำเดียว โดยอาจจะเป็นคำที่ไม่มีเสียงต้นซ้ำกับคำอื่นๆ เลย หรือ อาจจะมีแต่จะถูกจัดแยกออกมา เพื่อให้ไม่รวมกับคำอื่นๆ ในต้นไม้เดียวกัน (จะได้กล่าวถึงรายละเอียดต่อไป)

การประมวลผลเพื่อหาผลการรู้จำจะใช้ข้อมูล HMM ของคำภายในแต่ละ ต้นไม้เพื่อการหาผลการรู้จำหรือเส้นทางของคำที่เหมาะสมที่สุด โดยการนำต้นไม้แบบ lexicon มาใช้กับไวยากรณ์ทั้งแบบ Finite Stateและ N-gram จะมีข้อแตกต่างกันในวิธีนำมาใช้เล็กน้อย ซึ่งในที่นี้จะขอกล่าวถึงการนำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้กับไวยากรณ์แบบ Finite State ก่อน

1.1 การนำโครงสร้างข้อมูลต้นไม้แบบ lexicon มาใช้กับไวยากรณ์แบบ Finite State

เนื่องจากไวยากรณ์แบบ Finite State เป็นการแสดงถึงการเชื่อมต่อกันของกลุ่มคำตามที่ผู้ใช้กำหนด ดังนั้นการสร้างต้นไม้แบบ lexicon สำหรับไวยากรณ์แบบนี้ จึงสร้างเฉพาะในแต่ละกลุ่มคำเท่านั้น โดยจะไม่นำคำที่มีหน่วยเสียงเริ่มต้นซ้ำกันแต่อยู่คนละกลุ่มคำไปรวมกับต้นไม้กลุ่มของคำอื่น และการเชื่อมต่อกันของกลุ่มคำที่เกิดจากการเปลี่ยน state ของ Finite State จะเป็นการเชื่อมต่อจาก HMM สุดท้ายของแต่ละกิ่งสาขาของแต่ละต้นไม้ ไปยังต้นไม้ lexicon แต่ละตัวของกลุ่มคำถัดไปในไวยากรณ์นั่นเอง

โดยขั้นตอนการสร้างต้นไม้แบบ lexicon สำหรับไวยากรณ์แบบ Finite State จะเป็นดังต่อไปนี้

1) พิจารณาที่แต่ละกลุ่มคำที่ n ใดๆ (word class n)

2) นำคำศัพท์ในกลุ่มคำดังกล่าวมาเรียงลำดับตามพจนานุกรม (lexical sorting) โดยเรียงลำดับตามชื่อของ HMM ที่มาประกอบเป็นคำนั้นๆ และให้คำที่มีจำนวนของ HMM มากกว่า มีดัชนีที่ใช้เรียงลำดับสูงกว่า ในกรณีคำที่มีจำนวนของ HMM น้อยกว่าเป็นเสียงต้นของคำที่มีจำนวน HMM มากกว่าแบบทั้งหมด (full embedded word) จากนั้นจึงเริ่มพิจารณาคำแรกในพจนานุกรมที่เรียงลำดับแล้ว และกำหนดให้ $i = 0$

3) พิจารณาต้นไม้ lexicon *i* ใดๆ

4) หากคำที่มีดัชนีสูงกว่ามีหน่วยเสียงเริ่มต้นบางส่วนร่วมกับคำที่มีดัชนีต่ำกว่า ให้คำที่มีดัชนีสูงกว่าใช้หน่วยย่อยของเสียงพูดบางส่วนที่ซ้ำกันร่วมกับคำที่มีดัชนีต่ำกว่าในต้นไม้ lexicon *i* ที่กำลังพิจารณา แต่ถ้าหากไม่มีหน่วยเสียงเริ่มต้นบางส่วนซ้ำกันให้ย้อนกลับไปพิจารณาข้อ 3. พร้อมทั้งเพิ่มค่า *i*

1.2 การนำโครงสร้างข้อมูลต้นไม้แบบ lexicon tree มาใช้กับไวยากรณ์แบบ N-gram

การนำต้นไม้ lexicon มาใช้กับไวยากรณ์แบบ N-gram มีความแตกต่างจากไวยากรณ์แบบ Finite State เล็กน้อย เนื่องจากไวยากรณ์แบบ N-gram จะพิจารณาการเชื่อมต่อของคำโดยตรงเลย ไม่ได้ถูกแบ่งเป็นกลุ่มคำ โดยการสร้างต้นไม้ lexicon จะนำคำใน 1-gram ที่ปรากฏใน N-gram arpa format ทั้งหมดมาใช้ในการสร้างแต่ละต้นไม้ขึ้น และเนื่องจากในการเชื่อมต่อของแต่ละคำจะระบุด้วยความน่าจะเป็นของคู่คำที่ต่อกัน รวมไปถึงแต่ละคำก็จะมีความน่าจะเป็นในการเกิดที่แตกต่างกัน การนำคำที่มีเสียงต้นซ้ำกันมารวมกันในต้นไม้ lexicon จึงอาจจะรวมเอาคำทางขวามือ (w_r) ที่คำทางซ้ายมือ (w_l) ไม่เคยเชื่อมต่อด้วย นั่นคือ ความน่าจะเป็น $P(w_r \mid w_l) =$ 0.00 นั่นเอง ทำให้การพิจารณาการรู้จำภายในต้นไม้ lexicon จะต้องคำนึงถึงโอกาสในการเกิดสถานการณ์ดังกล่าวด้วย โดยเพื่อความแม่นยำและถูกต้องของการหาผลของการรู้จำ จึงสามารถเลือกบางคำที่มีโอกาสในการเกิดสูงกว่าคำอื่นๆ ให้ไม่มารวมกับคำอื่นๆ ในต้นไม้แบบหลายคำ คำดังกล่าวจะแยกให้เป็นเพียงต้นไม้แบบคำเดียวเพื่อป้องกันโอกาสเกิดสถานการณ์ดังกล่าวให้น้อยที่สุด ส่วนคำอื่นๆ ที่มี HMM ซ้ำกัน ก็จะนำมารวมกันในต้นไม้แบบหลายคำเหมือนเช่นการประยุกต์ใช้ต้นไม้แบบ lexicon กับไวยากรณ์แบบ Finite State Grammar

ตัวอย่างของลักษณะของโครงสร้างข้อมูลต้นไม้แบบ lexicon เมื่อกระจายออกมาเป็นการเชื่อมต่อกันของโนดในต้นไม้ ซึ่งจะถูกนำมาใช้ในการรู้จำสำหรับคำที่มีเสียงประกอบดังต่อไปนี้

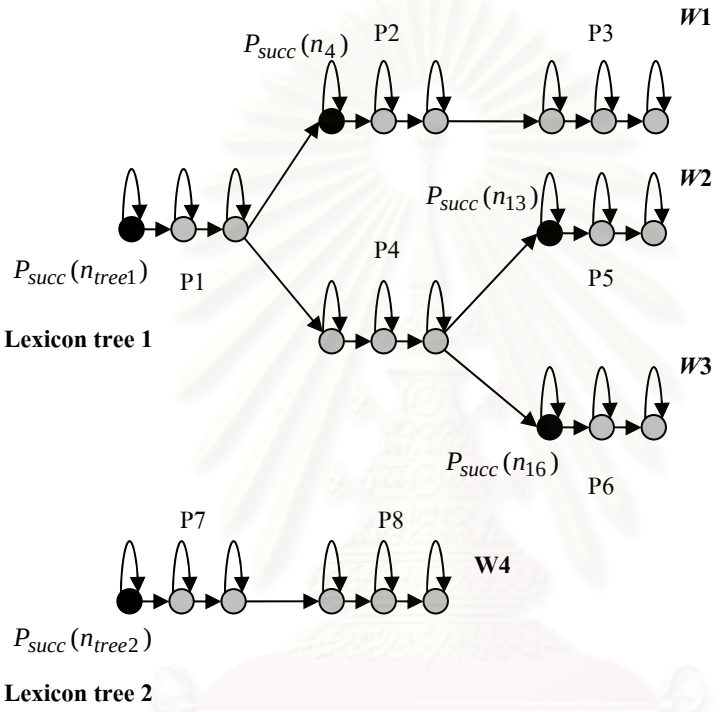
w_1 : p1 p2 p3

w_2 : p1 p4 p5

w_3 : p1 p4 p6

w_4 : p7 p8

ลักษณะของต้นไม้ lexicon ของคำเหล่านี้จะเป็นดังรูปที่ 6 โดยในที่นี้การแสดงโนดของ HMM ในต้นไม้ lexicon จะใช้เพียงโนด (state) ที่สามารถให้กำเนิดผลลัพธ์เท่านั้น แต่จะไม่รวม state ทางเข้า และ ทางออกของ HMM โดยความน่าจะเป็นในการเปลี่ยนโนดระหว่าง HMM จะใช้ความน่าจะเป็นในการเปลี่ยนโนดที่สามารถให้กำเนิดผลลัพธ์สุดท้ายของ HMM ก่อนหน้า ไปยัง state ทางออกของ HMM แสดง



รูปที่ 15 ลักษณะโนดของต้นไม้ lexicon เมื่อนำมาใช้กับไวยากรณ์แบบ N-gram

จากรูปที่ 15 จะเห็นว่ามีบางโนดถูกกำหนดค่าความน่าจะเป็นให้ นั่นคือ successive word probability ($P_{succ}(n)$) และเราเรียกโนดดังกล่าวว่าโนด successive โดยค่าความน่าจะเป็นดังกล่าวจะขึ้นอยู่กับตำแหน่งของโนดนั้นๆ ในต้นไม้ โดยที่โนดแรกของแต่ละต้นไม้ lexicon (n_{tree}) จะมีการกำหนดค่า $P_{succ}(n_{tree})$ ซึ่งก็คือ ความน่าจะเป็นที่มากที่สุดของคำในต้นไม้นั้นๆ ดังนั้นจากรูปโนด n_{tree1} ซึ่งเป็นโนดแรกของต้นไม้ lexicon ที่ 1 ที่เป็น ต้นไม้แบบหลายคำของ w_1, w_2 และ w_3 จึงมีค่า $P_{succ}(n_{tree1}) = \max(P(w_1), P(w_2), P(w_3))$ ส่วน โนดแรกที่เป็นโนดของคำที่ไม่มีการใช้ร่วมกับคำอื่นๆ เช่น n_4 , n_{13} และ n_{16} ก็จะมีค่า $P_{succ}(n)$ เท่ากับความน่าจะเป็นของคำนั้นๆ เลย นั่นคือ $P_{succ}(n_4) = P(w_1)$, $P_{succ}(n_{13}) = P(w_2)$ และ $P_{succ}(n_{16}) = P(w_3)$ นั่นเอง ต้นไม้

lexicon ที่ 2 นั้นเป็นต้นไม้แบบคำเดียว จึงกำหนดความน่าจะเป็น $P_{succ}(n_{tree2}) = P(w_4)$ เลย โดยค่าเหล่านี้จะถูกนำไปใช้เพื่อเปรียบเทียบคะแนนของเส้นทางในการรู้จำต่อไป

โดยขั้นตอนการสร้างต้นไม้แบบ lexicon สำหรับไวยากรณ์แบบ N-gram จะเป็นดังต่อไปนี้

1) พิจารณาทุกคำที่ปรากฏใน 1-gram

2) กำหนดจำนวนคำที่ไม่ต้องการให้รวมอยู่ในต้นไม้แบบหลายคำ (N_{sep})

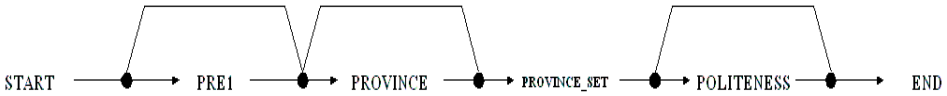
3) นำคำต่างๆใน 1-gram มาเรียงลำดับตามความน่าจะเป็นจากมากไปหาน้อย แล้วนำคำ N_{sep} ตัวแรกออกจากการพิจารณาในการหา prefix HMM ร่วมกับคำอื่นๆ ใน 1-gram โดยแต่ละคำดังกล่าวจะถูกสร้างเป็นต้นไม้แบบคำเดียวแทน

4) นำคำศัพท์ในกลุ่มคำที่เหลือมาเรียงลำดับตามพจนานุกรม (lexical sorting) โดยจะเรียงลำดับตามชื่อของ HMM ที่มาประกอบเป็นคำนั้นๆ โดยจะให้คำที่มีความยาวของ HMM มากกว่า มีดัชนีที่ใช้เรียงลำดับสูงกว่า ในกรณีที่คำที่มีความยาวของ HMM น้อยกว่าเป็น prefix ของคำที่มีความยาวมากกว่าแบบทั้งหมด

5) หากคำที่มีดัชนีสูงกว่ามี prefix HMM บางส่วนร่วมกับคำที่มีดัชนีต่ำกว่า ให้คำที่มีดัชนีสูงกว่าใช้หน่วยย่อยของเสียงพูดบางส่วนที่ซ้ำกันร่วมกับคำที่มีดัชนีต่ำกว่า

4.2.3.2 **รูปแบบการแสดงไวยากรณ์แบบ** Finite State **สำหรับตัวรู้จำแบบ** Julius

รูปแบบของการแสดงไวยากรณ์ของตัวรู้จำแบบ HTK จะใช้ลักษณะของแลตทิชแบบ slf ในการแสดงผลของไวยากรณ์ ซึ่งตัวรู้จำไวยากรณ์แบบ Julius จะใช้ลักษณะการแสดงไวยากรณ์ที่ต่างออกไป โดยจะใช้รูปแบบ DFA (Deterministic Finite Automachine) ในการแสดงไวยากรณ์แบบ Finite State แทน และแต่ละ state ในไวยากรณ์จะแสดงถึงกลุ่มคำ (word class) ต่างๆ โดยในที่นี้จะได้ยกตัวอย่างของไวยากรณ์แบบง่ายๆ ในการถามข้อมูลของจังหวัดต่างๆ ในประเทศไทย ดังรูปที่ 16 เพื่ออธิบายถึงลักษณะของ DFA ที่ใช้ในโปรแกรม Julius



รูปที่ 16 ไวยากรณ์ของรูปประโยคในการถามข้อมูลของจังหวัดต่างๆ ในประเทศไทย

ถ้าหากกำหนดให้แต่ละกลุ่มคำ PROVINCE_SET , PROVINCE , PRE1 , POLITENESS มีดัชนีของกลุ่มคำ (class_id) คือ { 0 , 1 , 2 , 3 } ตามลำดับ โดยแต่ละให้กลุ่มคำจะประกอบไปด้วยคำดังต่อไปนี้

1) PROVINCE_SET : กรุงเทพ , นนทบุรี , ปทุมธานี , สมุทรปราการ , เชียงใหม่ , . . . เป็นกลุ่มคำของจังหวัดทั้งหมดในประเทศไทย

2) PROVINCE : จังหวัด

3) PRE1 : เอา , ต้องการ , ขอ

4) POLITENESS : ครับ , ค่ะ

เราจะได้การแสดงไวยากรณ์ดังกล่าวในรูปแบบ DFA ได้ดังรูปที่ 17

State	Class_id	next
0	3	2
0	0	1
1	1	3
1	2	4
1	-1	-1
2	0	1
3	2	4
3	-1	-1
4	-1	-1

รูปที่ 17 ลักษณะการแสดงไวยากรณ์แบบ Finite State ในรูป DFA ที่ใช้ในตัวรู้จำแบบ Julius

การแสดงรูปแบบ DFA ดังกล่าวจะเป็นการแสดงการเชื่อมต่อแบบย้อนกลับจากกลุ่มคำสุดท้าย มาสู่กลุ่มคำแรก นั่นคือการเชื่อมต่อจาก END ในรูปที่ 16 มายังกลุ่มคำ POLITENESS และเชื่อมต่อมาเรื่อยๆ จนถึง START ซึ่งการเชื่อมต่อจาก END ไปยังคำต่างๆ ทางซ้ายจะถูกแสดงด้วย state 0 ซึ่งเป็น state เริ่มต้น จากรูปที่ 16 จะเห็นว่า END จะเชื่อมต่อกับ POLITENESS (class_id 3) และ PROVINCE_SET (class_id 0) และ PROVINCE_SET จะเชื่อมต่อไปยัง state ถัดไปคือ state 1 (next) และ POLITENESS จะเชื่อมต่อไปยัง PROVINCE_SET ถึงแม้ว่า PROVINCE_SET จะอยู่โดยจะกำหนดให้เป็น state 2 (next) ซึ่งลักษณะของการแสดงการเชื่อมต่อจะเป็นดังที่ได้กล่าวมา และจะทำเช่นนี้ไปเรื่อยๆ จนกระทั่งถึงการเชื่อมต่อไปยัง START หรือ กลุ่มคำที่เป็นสิ้นสุดของการเชื่อมต่อจากขวาไปซ้าย โดย state 1 ที่เป็นการเชื่อมต่อจากขวาไปซ้ายจาก PROVINCE_SET (class_id 0) จะมีการเชื่อมต่อไปยัง START ซึ่งจะแทนด้วย class_id พิเศษ คือ -1 เพื่อระบุว่าการเชื่อมต่อนั้นมาจนถึงต้นประโยคแล้ว การเชื่อมต่อดังกล่าวจะถูกระบุให้ไม่มี state ถัดไป นั่นก็คือให้ค่า next มีค่าเป็น -1 นั่นเอง ซึ่งก็จะกำหนด state ในลักษณะดังกล่าวให้ครบทุกการเชื่อมต่อ ก็จะได้ลักษณะการแสดงไวยากรณ์ Finite State แบบ DFA ของรูปประโยคที่ต้องการทุกรูปแบบดังรูปที่ 17

โดยลักษณะของไวยากรณ์แบบ Finite State เพื่อสร้างเป็น DFA สำหรับโปรแกรม Julius นั้น จะไม่ได้รองรับสัญลักษณ์พิเศษในการแสดงไวยากรณ์ เช่น [...] <...> หรือ {...} โดยจะแสดงได้ถึงการเชื่อมต่อแบบธรรมดาเท่านั้น และไม่สามารถเขียนไวยากรณ์ที่มีการวนซ้ำได้ในเบื้องต้น ทำให้ DFA ไม่สามารถแสดงรูปแบบของไวยากรณ์ในบางรูปแบบได้ ซึ่งทางคณะผู้วิจัยได้พัฒนาโปรแกรมในการสร้าง DFA เพื่อใช้ในตัวรู้จำแบบ Julius โดยสามารถเพิ่มสัญลักษณ์พิเศษดังกล่าวในลักษณะเดียวกับโปรแกรม HTK ไปในไวยากรณ์แบบ Finite State ได้ ทำให้สามารถเขียนรูปแบบของไวยากรณ์ได้หลากหลายมากยิ่งขึ้น

4.2.3.2 การประมวลผลเพื่อค้นหาผลการรู้จำเมื่อมีการนำโครงสร้างต้นไม้แบบ lexicon มาใช้

ขั้นตอนการประมวลผลเพื่อค้นหาผลการรู้จำในกรณีที่มีการนำโครงสร้างต้นไม้แบบ lexicon มาใช้ จะมีลักษณะที่คล้ายๆ กับขั้นตอนการหาผลการรู้จำในแบบโปรแกรม HTK โดยจะแบ่งลักษณะการประมวลผลออกเป็น 2 ส่วน คือ การค้นหาแบบไปข้างหน้า (forward search) และการค้นหาแบบย้อนกลับ (reverse search) เช่นกัน ซึ่งรายละเอียดของส่วนต่างๆ จะได้กล่าวถึงโดยสังเขปดังนี้

1. การค้นหาผลของการรู้จำแบบไปข้างหน้า

การประมวลผลเพื่อหาผลการรู้จำที่ดีที่สุดแบบไปข้างหน้าเมื่อมีการนำต้นไม้แบบ lexicon มาใช้ จะยังคงใช้วิธีการเลือกเส้นทางที่ดีที่สุดที่แต่ละโนดของ HMM ในต้นไม้ lexicon โดยจะพิจารณาคะแนนของเสียงพูดที่แต่ละเวลาจากเวลาเริ่มต้น คือ $t = 1$ จนถึงเวลาสุดท้าย คือ $t = T$ ด้วยอัลกอริทึม Viterbi ในการตัดสินใจเลือกเส้นทางที่ดีที่สุดที่แต่ละโนดของต้นไม้ โดยการส่งผ่านโทเคน เช่นเดียวกับการประมวลผลขั้นตอนเดียวกันในตัวรู้จำแบบ HTK

ซึ่งการส่งผ่านโทเคนภายในแต่ละต้นไม้จะพิจารณาทุกโนดในต้นไม้จนกระทั่งถึงการส่งผ่านโทเคนจากโนดที่เป็นกิ่งของแต่ละต้นไม้ (n_w) ซึ่งเป็นโนดสุดท้ายของคำ w ใดๆ เมื่อมีการส่งผ่านโทเคนออกจากโนด n_w เราจะได้คำใหม่เกิดขึ้นในเส้นทางของการรู้จำดังกล่าว โดยโทเคนที่ถูกส่งผ่านออกไปจะบันทึกคำ w ที่เกิดขึ้น แล้วจะนำไปเชื่อมต่อกับประโยคที่ได้จากเส้นทางดังกล่าวที่เวลา t-1

จากนั้นจึงเป็นการส่งผ่านโทเคนระหว่างต้นไม้ ซึ่งจะมีการพิจารณาที่แตกต่างกันไปสำหรับแต่ละไวยากรณ์ โดยจะนำโทเคนที่ถูกส่งผ่านระหว่างต้นไม้แล้วส่งไปยังต้นไม้เดียวกัน มาทำการเปรียบเทียบที่โนดแรกของต้นไม้ถัดไป โดยใช้อัลกอริทึม Viterbi การส่งผ่านจะทำเช่นนี้ไปเรื่อยๆ นั่นคือการส่งผ่านโทเคนภายในต้นไม้และระหว่างต้นไม้จนกระทั่งพิจารณาข้อมูลเสียงครบทุกเวลา จากนั้นเราจะนำเส้นทางของโทเคนที่ถูกส่งผ่านไปยังโนดสุดท้ายของคำที่เป็นคำสุดท้ายของแต่ละไวยากรณ์ เป็นเส้นทางเพื่อใช้ในการได้ผลลัพธ์การรู้จำแบบไปข้างหน้าต่อไป

เส้นทางของคำที่ดีที่สุดที่เกิดขึ้นเมื่อผ่านการประมวลผลที่เวลาสุดท้ายของเสียงพูดเสร็จจะเป็นคำตอบของการรู้จำที่ได้จากการค้นหาแบบไปข้างหน้า นอกจากนี้เส้นทางของคำที่เกิดขึ้นที่แต่ละเวลาก็จะถูกนำมาใช้ประกอบในการตัดสินใจเพื่อค้นหาผลการรู้จำในลำดับรองๆ ลงมาโดยการค้นหาแบบย้อนกลับในกรณีที่ผู้ใช้ต้องการผลการรู้จำมากกว่า 1 แบบต่อไป

ขั้นตอนการส่งผ่านโทเคนจากแต่ละโนดไปยังโนดถัดไปในต้นไม้ lexicon รวมทั้งระหว่างต้นไม้ lexicon จะมีลักษณะหลักๆ ดังที่ได้กล่าวมาแล้วข้างต้น โดยแต่ละไวยากรณ์ก็จะมีการพิจารณารายละเอียดปลีกย่อยที่แตกต่างกัน ซึ่งรายละเอียดต่างๆ จะได้กล่าวถึงต่อไป

1.1 การค้นหาผลการรู้จำแบบไปข้างหน้าสำหรับไวยากรณ์แบบ Finite State

ขั้นตอนการพิจารณาเพื่อหาผลการรู้จำแบบไปข้างหน้าสำหรับไวยากรณ์แบบ Finite Stateจะมีขั้นตอนหลักๆ ดังที่ได้กล่าวมา โดยสิ่งที่จะต้องพิจารณาข้อมูลจากไวยากรณ์ดังกล่าวก็

คือ การส่งผ่านโทเคนระหว่างต้นไม้ lexicon ซึ่งจะต้องพิจารณาว่าคำ w ที่เกิดขึ้นนั้นอยู่ในกลุ่มคำใด ($class_l$) แล้วเราจึงจะส่งผ่านโทเคนไปยังโนดแรกของต้นไม้ lexicon ทั้งหมดของกลุ่มคำที่เชื่อมต่อกับคำดังกล่าวทั้งหมด ($class_r$) โดยการเชื่อมต่อจาก $class_l$ ไปยัง $class_r$ ที่เป็นไปได้ทั้งหมดจะไม่คำนึงถึงลำดับของการเชื่อมต่อของกลุ่มคำ นั่นหมายความว่า ข้อมูลของ Finite State ที่จะนำมาใช้ในการค้นหาแบบไปข้างหน้าจะใช้เพียงการพิจารณาว่า $class_l$ มีการเชื่อมต่อกับ $class_r$ หรือไม่ใน Finite State เท่านั้น โดยที่อัลกอริทึม Viterbi จะยังคงสามารถเลือกเส้นทางที่ดีที่สุดได้ เพราะเราเลือกเพียงแค่เส้นทางเดียวที่ดีที่สุดนั่นเอง แต่ถ้าหากต้องการให้ลำดับการเชื่อมต่อของกลุ่มคำมีความหมาย เราก็จะต้องกำหนดให้กลุ่มคำเดียวกันที่เชื่อมต่อกันในไวยากรณ์และต่างลำดับกันเป็นกลุ่มคำที่ชื่อต่างกันแทน แต่ก็จะทำให้กระประมวลผลในการส่งผ่านโทเคนมากขึ้นด้วยเช่นกัน

1.2 การค้นหาผลการรู้จำแบบไปข้างหน้าสำหรับไวยากรณ์แบบ N-gram

การค้นหาผลการรู้จำแบบไปข้างหน้าเมื่อมีการนำไวยากรณ์แบบ N-gram มาประยุกต์ใช้จะมีลักษณะต่างๆ ที่แตกต่างจากไวยากรณ์แบบ Finite Stateเนื่องจากไวยากรณ์แบบ N-gram จะมีการนำความน่าจะเป็นของการเชื่อมต่อของคำ รวมทั้งความน่าจะเป็นของแต่ละคำมาใช้ ตามที่ระบุไว้ใน N-gram arpa format ซึ่งการส่งผ่านโทเคนระหว่างโนดจะต้องคำนึงถึงความน่าจะเป็นเหล่านี้ด้วย โดยจะแยกออกเป็น 2 กรณีด้วยกัน นั่นคือ

1.2.1 การส่งผ่านโทเคนภายในต้นไม้

จากที่ได้กล่าวมาแล้วข้างต้นว่าลักษณะของต้นไม้ lexicon ที่พิจารณาจะมี 2 แบบ คือ ต้นไม้แบบคำเดียว กับ ต้นไม้แบบหลายคำ ซึ่งการส่งผ่านโทเคนภายในต้นไม้กรณีต้นไม้แบบคำเดียวนั้นจะทำในลักษณะเดียวกันกับการส่งโทเคนภายในต้นไม้เมื่อนำไวยากรณ์แบบ Finite State มาใช้ นั่นคือใช้อัลกอริทึม Viterbi ในการเลือกเส้นทางที่ดีที่สุดนั่นเอง แต่สำหรับกรณีที่เป็นการส่งผ่านโทเคนในต้นไม้แบบหลายคำจะต้องคำนึงถึง $P_{succ}(n)$ ด้วย เมื่อมีการส่งผ่านโทเคนไปยังโนด successive คะแนนของโทเคนที่ถูกส่งผ่านไปยังโนด successive ที่เวลา t ใดๆ จึงเป็นดังสมการที่ 3

$$score_t(n) = score_t^T(n) \cdot P_{succ}(n) \cdot b_n(t) \qquad (3)$$

แต่ถ้าหากว่าในต้นไม้ดังกล่าวมีโนด successive มากกว่า 1 โนด แล้วโทเคนนั้นเคยถูกส่งผ่านโนด successive มาแล้ว เราจะต้องตัดความน่าจะเป็นของโนด successive ที่ผ่านมา (n_l) ออกก่อน แล้วค่อยรวมความน่าจะเป็นของโนด successive ล่าสุดเข้าไปดังสมการที่ 4

$$score_t(n) = \frac{score_t^T(n)}{P_{succ}(n_l)} \cdot P_{succ}(n) \cdot b_n(t) \qquad (4)$$

การนำความน่าจะเป็น $P_{succ}(n)$ มาใช้จากโนดของต้นไม้ lexicon เป็นการนำความน่าจะเป็นของ 1-gram จาก arpa format มาใช้ในการค้นหาผลการรู้จำแบบไปข้างหน้านั่นเอง

1.2.2 การส่งผ่านโทเคนระหว่างต้นไม้

การส่งผ่านโทเคนระหว่างต้นไม้เป็นการส่งผ่านโทเคนระหว่างคำ ซึ่งความน่าจะเป็นที่เกี่ยวข้องและจะถูกนำมาใช้ก็ คือ ค่าความน่าจะเป็นของ 2-gram แบบไปข้างหน้า โดยถ้าหากคู่คำที่พิจารณาไม่เคยเกิดก็จะลดลำดับของ context ลงมาใช้เพียงความน่าจะเป็นของแต่ละคำ แต่จะถูกถ่วงน้ำหนักด้วยค่าถ่วงน้ำหนัก backoff ดังที่ได้กล่าวมาแล้ว เนื่องจากการที่หลายๆ คำอาจจะถูกรวมกันอยู่ในต้นไม้ lexicon เดียวกันได้เมื่อมีการใช้โนดร่วมกัน ดังนั้นจึงจะมีการแยกพิจารณาการส่งโทเคนระหว่างต้นไม้ด้วยกัน 2 แบบ คือ

1) การส่งโทเคนจากต้นไม้ lexicon ใดๆ ไปยังต้นไม้แบบคำเดียว

การส่งโทเคนจากต้นไม้ใดๆ ไปยังต้นไม้แบบคำเดียวนั้นจะสามารถนำค่าความน่าจะเป็นของ 2-gram ที่เกิดขึ้น คือ $P(w_r \mid w_l)$ มารวมในค่าคะแนนของโทเคน $tok_t(n)$ คือ $score_t(n_w)$ ที่ส่งผ่านมาจากโนด n_w ซึ่งเป็นโนดที่กิ่งของต้นไม้ทางซ้าย ณ. เวลาที่ t-1 ได้โดยตรง เมื่อ w_l ในที่นี้คือ คำที่เกิดขึ้นจากการส่งผ่านโทเคน $tok_t(n)$ และ w_r คือ คำของต้นไม้แบบคำเดียวที่รับโทเคนดังกล่าว โดยมีโนดแรกของต้นไม้ คือ n_{tree} มาใช้ได้โดยตรงในการรวมกับคะแนนของเส้นทางดังกล่าวดังสมการที่ 5

$$score_t(n_{tree}) = score_{t-1}(n_w) \cdot P(right\_word \mid left\_word) \qquad (5)$$

2) การส่งโทเคนจากต้นไม้ lexicon ใดๆ ไปยังต้นไม้แบบหลายคำ

ส่วนการส่งโทเคนจากต้นไม้ใดๆ ไปยังต้นไม้แบบหลายคำจะแตกต่างออกไป เนื่องจากในต้นไม้แบบหลายคำจะมีคำในต้นไม้มากกว่าหนึ่งคำ ซึ่งคำล่าสุดที่เกิดขึ้นจากต้นไม้ทางซ้าย อาจจะไม่มีโอกาสไปเชื่อมต่อกับบางคำที่อยู่ในต้นไม้แบบหลายคำ เมื่อพิจารณาจากข้อมูลของ 2-gram

ดังนั้นเพื่อให้มีโอกาสเกิดข้อผิดพลาดจากกรณีดังกล่าวน้อยที่สุด การส่งผ่านโทเคนไปยังต้นไม้แบบหลายคำ จึงเลือกส่งผ่านเพียงโทเคนเดียวที่มีค่าคะแนนมากที่สุด โดยจะถูกส่งผ่านจาก n_w ใดๆ ที่เวลา t-1 ($tok_{t-1}^{\max}(n_w)$) ซึ่งในกรณีนี้จะได้นำค่า $P_{succ}(n_{mtree})$ เมื่อ n_{mtree} คือ โนดแรกของต้นไม้แบบหลายคำมารวมกับคะแนนของเส้นทางดังกล่าวในลักษณะเดียวกับการส่งผ่านโทเคน ภายในต้นไม้ดังสมการที่ 6 แทน

$$score_t(n_{mtree}) = score_{t-1}^{\max}(n_w) \cdot P_{succ}(n_{mtree}) \cdot b_{n_{mtree}}(t) \qquad (6)$$

เมื่อ $b_{n_{mtree}}(t)$ ในที่นี้เป็นความน่าจะเป็นในการเกิดเสียงพูดที่เวลา t ของโนด n_{mtree}

2. การค้นหาผลการรู้จำแบบย้อนกลับ

การหาผลการรู้จำแบบย้อนกลับจะมีลำดับขั้นตอนที่ตรงกันข้ามกับการค้นหาผลการรู้จำแบบไปข้างหน้า โดยจะทำการค้นหาจากเวลาที่ T ย้อนกลับไปจนถึงเวลาเริ่มต้น คือ 1 โดยการค้นหาแบบย้อนกลับนี้จะไม่ได้นำโครงสร้างข้อมูลต้นไม้ lexicon มาใช้โดยตรงเหมือนการค้นหาแบบไปข้างหน้า แต่จะนำข้อมูลที่ได้จากการค้นหาแบบไปข้างหน้า คือ เส้นทาง และ คำที่เกิดขึ้นทั้งหมดที่แต่ละเวลามาใช้ในการค้นหาเส้นทางของคำแบบย้อนกลับโดยวิธี stack decoding แทน การค้นหาแบบย้อนกลับนี้จะแบ่งขั้นตอนเป็นการค้นหาเส้นทางภายในคำที่เกิดในเส้นทางที่ถูกบันทึกจากการค้นหาแบบไปข้างหน้า และ การค้นหาเส้นทางระหว่างคำในลักษณะเดียวกับการค้นหาแบบไปข้างหน้า โดยผลการรู้จำที่ดีที่สุด N แบบ จะสามารถได้มาจากการค้นหาในลักษณะนี้นั่นเอง

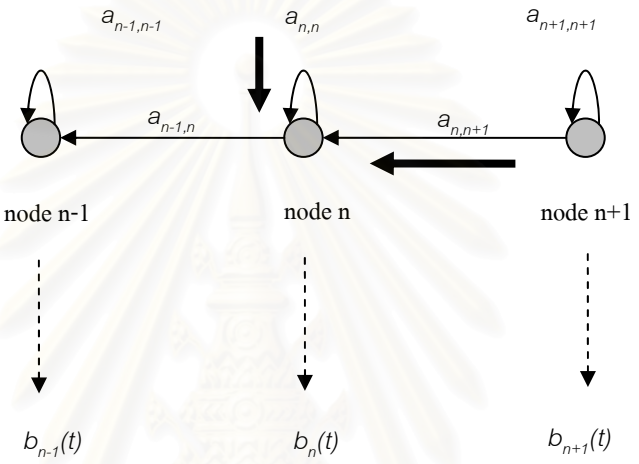
โดยคะแนนที่นำมาใช้ในการเปรียบเทียบผลการรู้จำอันดับรองลงมาในแต่ละลำดับของแต่ละเส้นทางแบบย้อนกลับ (*rev_path*) จะขึ้นอยู่กับความน่าจะเป็นการเกิดแบบย้อนกลับของเส้นทางจนถึงเวลา t ใดๆ คือ $G_t(rev\_path)$ ค่า $G_t(rev\_path)$ จะถูกนำไปรวมกับคะแนนก่อนหน้าของเส้นทางที่ได้จากการค้นหาแบบไปข้างหน้าของเส้นทางที่ทำให้เกิดคำ w คือ $score_t(n_w)$ และจะรวมความน่าจะเป็นการเกิดที่โนดสุดท้ายของแต่ละคำที่พิจารณา ($node_M$) ซึ่งความสัมพันธ์ของคะแนน $G_t(rev\_path)$ จะเป็นดังสมการที่ 7

$$G_t(rev\_path) = score_t(n_w) \cdot b_{n_M}(t) \cdot G_{t+1}(rev\_path) \qquad (7)$$

เริ่มต้นการค้นหาผลการรู้จำแบบย้อนกลับจะพิจารณาคำคำสุดท้ายของแต่ละไวยากรณ์ โดยจะพิจารณาที่เวลาสุดท้ายของเสียงพูด จากนั้นจะเริ่มทำการคำนวณหาความน่าจะเป็นในการเกิดเสียงพูดที่โนดสุดท้ายของคำดังกล่าว แล้วและจะบันทึกเป็นค่า $G_t(rev\_path)$ ซึ่งในที่นี้ คือ

$G_T(rev\_path)$ ของเส้นทางที่ทำการค้นหาย้อนกลับ ซึ่งจะเป็นค่าเริ่มต้นในการค้นหาผลการรู้จำแบบย้อนกลับ

เมื่อได้ $G_t(rev\_path)$ ที่คิดที่โนดสุดท้ายของคำที่พิจารณาแล้ว จะเข้าสู่การพิจารณาเส้นทางการค้นหาย้อนกลับภายในแต่ละคำนั้นๆ ซึ่งก็จะทำอัลกอริทึม Viterbi เปรียบเทียบ $G_t(rev\_path)$ ภายในคำโดยการเปลี่ยนโนดแบบย้อนกลับที่พิจารณาในแต่ละ HMM ของคำดังรูปที่ 18



รูปที่ 18 การพิจารณาเส้นทางสำหรับอัลกอริทึม Viterbi ที่แต่ละ node n ใดๆ แบบย้อนกลับภายใน word atom และที่เวลา *t* ใดๆ

ซึ่งการพิจารณาการค้นหาแบบย้อนกลับภายในคำ w ใดๆ เพื่อหา $G_t(rev\_path)$ นั้น จะทำการค้นหาที่เวลาตั้งแต่เวลาที่ $t = t_e$ จนถึงเวลา $t = t_s$ เมื่อ t_e คือเวลาสุดท้ายของคำ w และ t_s คือเวลาเริ่มต้นของคำ w โดยการทำอัลกอริทึม Viterbi เพื่อหาค่า $G_t(rev\_path)$ ที่แต่ละโนดภายในคำ w ซึ่ง $G_t(rev\_path)$ ที่ได้จากการทำอัลกอริทึม Viterbi ที่โนดแรกของ w ที่เวลา t_s จะถูกส่งผ่านในการหาเส้นทางของผลการรู้จำแบบย้อนกลับเพื่อใช้ในการหาเส้นทาง *rev_path* ต่อไปดังสมการที่ 7 โดยเส้นทางการค้นหาแบบย้อนกลับที่จะถูกนำมาใช้ในการค้นหาที่เวลาถัดไปก็จะเป็นเส้นทางการค้นหาที่มีคะแนน $G_t(rev\_path)$ สูงสุด โดยขั้นตอนการเลือกคำถัดไปในการพิจารณานั้นจะขึ้นอยู่กับชนิดของไวยากรณ์ที่พิจารณา ซึ่งรายละเอียดจะได้กล่าวถึงต่อไป

การประมวลผลเพื่อหาเส้นทางการรู้จำแบบย้อนกลับสำหรับตัวรู้จำแบบ Julius จะทำตามวิธีที่ได้กล่าวมาไปเรื่อยๆ จนกระทั่งเวลาเริ่มต้นของคำล่าสุดที่ได้เป็นเวลาแรกสุดของเสียงพูด ก็จะได้ผลการรู้จำที่ดีที่สุดจากการค้นหาแบบย้อนกลับมาก่อน เนื่องจากเราได้ทำการค้นหาจนกระทั่ง

เริ่มต้นพูดแล้ว โดยคำตอบทั้งหมดของประโยคที่ได้ ก็คือคำ w ที่พบทั้งหมดในเส้นทางการค้นหาแบบย้อนกลับนั่นเอง จากนั้นก็จะนำเส้นทางย้อนกลับที่เกิดขึ้นที่มี $G_t(rev\_path)$ ในอันดับรองๆ ลงมามาทำการค้นหาต่อจนกว่าเวลาเริ่มต้นของคำที่ได้จะเป็นเวลาแรกสุดของเสียงพูด แล้วจะทำเช่นนี้ไปเรื่อยๆ จนกว่าจะได้จำนวนผลการรู้จำครบ N แบบตามที่ต้องการ

หมายเหตุ การพิจารณาหาความน่าจะเป็นในการเกิดของเสียงที่เวลาต่างๆ ที่จะถูกนำมารวมในค่า $G_t(rev\_path)$ เพื่อหาค่าคะแนนของการค้นหาแบบย้อนกลับนั้น เราจะไม่ต้องทำการคำนวณใหม่ทั้งหมด โดยเราจะใช้ค่าที่เคยคำนวณที่แต่ละโนดของคำที่แต่ละเวลาจากการค้นหาแบบไปข้างหน้า ซึ่งจะถูกนำมาบันทึกไว้ในหน่วยความจำพิเศษเพื่อลดปริมาณการคำนวณลง

2.1 การเลือกคำที่จะพิจารณาถัดไปเมื่อใช้ไวยากรณ์แบบ Finite State

เนื่องจากไวยากรณ์แบบ Finite State จะคำนึงถึงการเชื่อมต่อกันระหว่างกลุ่มของคำเป็นหลัก โดยจะขึ้นอยู่กับ state ของไวยากรณ์ปัจจุบันที่กำลังพิจารณาในไวยากรณ์ขณะนั้น โดยเริ่มต้นจะพิจารณาที่ state แรกของการพิจารณาแบบย้อนกลับนั่นคือ state ที่ 0 ดังที่ได้กล่าวมาแล้วข้างต้นนั่นเอง ดังนั้นที่แต่ละคำ w_i ใดๆ ที่อยู่ในกลุ่มคำที่ i จะทำการเชื่อมต่อกับทุกๆ คำในกลุ่มคำที่ j (w_j) ใดๆ ตามที่ถูกระบุไว้ในไวยากรณ์ โดยเราจะต้องหาเส้นทางที่มีคะแนนสูงสุดที่ได้จากการค้นหาแบบไปข้างหน้าที่มีคำล่าสุดเป็น w_j ที่เวลา t ใดๆ ที่พิจารณา ซึ่งการพิจารณาก็จะเกิดเส้นทางใหม่ขึ้นมากที่สุดเท่ากับ N_j เสมอในทุกๆ ครั้งที่จะเชื่อมต่อกับ คำในกลุ่มคำที่ j เมื่อกำหนดให้ N_j คือ จำนวนคำทั้งหมดที่ปรากฏในกลุ่มคำที่ j

2.2 การเลือกคำที่จะพิจารณาถัดไปเมื่อใช้ไวยากรณ์แบบ N-gram

การค้นหาคำที่จะพิจารณาถัดไปจากคำ w_m เมื่อใช้ไวยากรณ์แบบ N-gram ในการรู้จำจะนำความน่าจะเป็นของคำที่เกิดขึ้นโดยมี context แบบย้อนกลับมาพิจารณารวมกับคะแนนในการค้นหาแบบย้อนกลับ เพื่อให้ผลการรู้จำที่ได้มีความน่าเชื่อถือตามค่าทางสถิติที่ได้ทำการรวบรวมมา

เริ่มต้นการพิจารณาคำถัดไปในไวยากรณ์แบบนี้ คือ จะทำการค้นหาว่าข้อมูลจาก N-gram แบบย้อนกลับนั้นมีการเชื่อมต่อแบบย้อนกลับจากคำ w_m ล่าสุดที่พิจารณา ไปยังคำ w_l ใดๆ ในเวลาที่พิจารณาหรือไม่ โดยการค้นหาแบบย้อนกลับนี้จะมีความพิเศษ โดยจะเพิ่ม context ในการพิจารณาได้มากกว่าการค้นหาแบบไปข้างหน้าที่จะใช้เพียงข้อมูลของ 2-gram แบบไป

ข้างหน้าเท่านั้น ซึ่งการค้นหาแบบย้อนกลับจะใช้ 3-gram แบบย้อนกลับในการพิจารณาโอกาสการเกิดคำเพื่อให้ผลการรู้จำมีความน่าเชื่อถือมากขึ้น โดยเราจะพิจารณา context อันดับสูงสุด คือ 3-gram ก่อน เริ่มต้นจะพิจารณาว่าโอกาสในการเกิดคำ 3 คำเรียงติดกัน คือ w_l, w_m, w_r หรือไม่ โดยจะต้องตรวจสอบก่อนว่า context ของ w_l คือ w_m และ w_r ปรากฏอยู่ใน 2-gram แบบย้อนกลับหรือไม่ ถ้าเคย ($P(w_m \mid w_r) \neq 0$) ก็จะพิจารณาว่าทั้ง 3 คำนี้ปรากฏใน 3-gram แบบย้อนกลับหรือไม่ ถ้าเคยก็จะใช้ $P(w_l \mid w_m, w_r)$ เข้าไปรวมใน $G_t(rev\_path)$ ที่ใช้ในการเปรียบเทียบคะแนนต่อไป แต่ถ้าหากทั้ง 3 คำไม่เคยปรากฏใน 3-gram แบบย้อนกลับ ก็จะนำค่าถ่วงน้ำหนัก backoff ของ w_m, w_r คือ $Wght(w_m, w_r)$ ไปรวมกับ $P(w_l \mid w_m)$ แทน นั่นคือ จะนำความน่าจะเป็น $P(w_l \mid w_m) \cdot Wght(w_m, w_r)$ ไปรวมกับ $G_t(rev\_path)$ แทนเพื่อเปรียบเทียบคะแนนต่อไป

แต่ถ้าหาก context ของ w_l คือ w_m และ w_r ไม่ปรากฏอยู่ใน 2-gram แบบย้อนกลับ ก็จะใช้เพียงแค่ความน่าจะเป็นของ 2-gram แบบย้อนกลับเท่านั้นในการหาผลของการรู้จำ โดยถ้า w_l และ w_m ปรากฏอยู่ใน 2-gram แบบย้อนกลับ ก็จะนำ $P(w_l \mid w_m)$ ไปรวมกับ $G_t(rev\_path)$ ในการเปรียบเทียบคะแนนต่อไป แต่ถ้า w_l และ w_m ไม่ปรากฏอยู่ใน 2-gram แบบย้อนกลับ ก็จะนำค่าถ่วงน้ำหนัก backoff ของ w_m ไปรวมกับความน่าจะเป็นของ w_l คือ $P(w_l) \cdot Wght(w_m)$ เพื่อนำไปรวมกับ $G_t(rev\_path)$ แทน

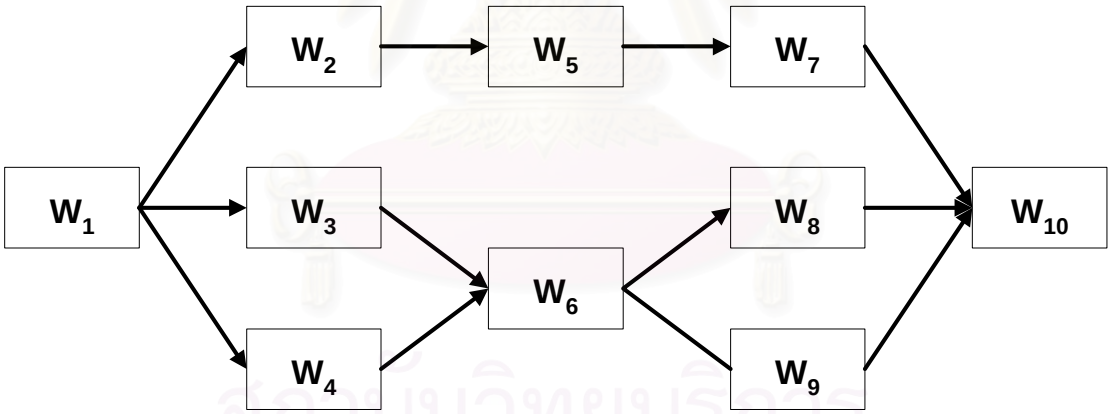
3. การตัดบางเส้นทางที่พิจารณาในการค้นหาผลการรู้จำแบบไปข้างหน้า

การตัดเส้นทางบางเส้นทางที่เกิดขึ้น (pruning) ในการค้นหาผลการรู้จำแบบไปข้างหน้าสำหรับการใช้โครงสร้างต้นไม้ lexicon ในการรู้จำ จะเป็นการลดจำนวนการประมวลผลของการรู้จำ ซึ่งจำนวนโทเคนที่เป็นไปได้มากที่สุดที่จะพิจารณาที่เวลาใดๆ จะมีได้มากที่สุดเท่ากับจำนวนโนดของต้นไม้ทั้งหมดในการพิจารณา สำหรับกรณีที่ใช้อัลกอริทึม Viterbi ในการตัดสินเส้นทางที่ดีที่สุดเพียงเส้นทางเดียวที่แต่ละโนดซึ่งบางเส้นทางที่มีคะแนนต่ำมากๆ นั้นอาจจะมีผลน้อยมากในการหาผลการรู้จำ ซึ่งเส้นทางเหล่านี้อาจจะละทิ้งได้ ยิ่งกรณีที่ถ้าหากมีคำที่จะต้องพิจารณามากเท่าไหร่ จำนวนโนด หรือ โทเคนที่เป็นไปได้ในการพิจารณาก็จะมีมากขึ้นทำให้ต้องใช้เวลาในการประมวลผลมาก โดยการจำกัดจำนวนของโทเคนสำหรับการใช้โครงสร้างต้นไม้ lexicon ในการรู้จำแบบโปรแกรม Julius ได้เลือกใช้วิธีจำกัดจำนวนของโทเคนที่ค่าคงที่ค่าหนึ่งแทน (N_{beam}) ซึ่งจะแตกต่างจากวิธีของโปรแกรม HTK ที่เลือกใช้ความกว้างของคะแนนแทน วิธีจำกัดจำนวนโทเคนนี้แม้อาจจะตัดจำนวนของโทเคนที่พิจารณาไปมากในกรณีที่ N_{beam} มีค่าน้อย ซึ่งอาจจะทำให้เกิดโอกาสที่จะสูญเสียความแม่นยำในการประมวลผลการรู้จำได้ แต่วิธีนี้จะสามารถจำกัดจำนวนการ

ประมวลผลได้อย่างแน่นอน เนื่องจากเราจะพิจารณาจำนวนโทเคนเพียงจำนวนจำกัดค่าหนึ่งเท่านั้น ซึ่งในการวิจัยนี้จะได้เลือกใช้ $N_{beam} = 400$ ก็ยังคงให้ผลการรู้จำที่น่าพึงพอใจ และใช้เวลาในการประมวลผลค่อนข้างเร็ว

4. การสร้างผลการรู้จำแบบ word graph

ข้อมูลการเชื่อมต่อของคำในประโยคที่เกิดขึ้นที่ได้จากการทำอัลกอริทึม Viterbi แบบย้อนกลับสำหรับตัวรู้จำแบบ Julius นอกจากจะทำให้ได้ผลการรู้จำในลำดับรองลงมาแล้ว ยังจะสามารถนำข้อมูลเหล่านี้มาสร้างเป็น word graph ซึ่งเป็นการแสดงการเชื่อมต่อของคำและข้อมูลต่างๆ ของคำที่รู้จำได้ โดย word graph ของผลการรู้จำจะมีประโยชน์ต่อผู้พัฒนามากในแง่การนำเสนอข้อมูลของผลการรู้จำ เช่น การเชื่อมต่อของคำ ข้อมูลของเวลา ค่าความเชื่อมั่นของแต่ละคำในโทเคนที่ได้จากการคำนวณ รวมไปถึงการใช้คำร่วมกันของประโยคผลลัพธ์ โดยลักษณะของ word graph ที่ได้จากการรู้จำจะมีลักษณะดังรูปที่ 19



รูปที่ 19 ลักษณะของ word graph ที่ได้จากการค้นหาแบบย้อนกลับ

จากรูปที่ 18 จะเป็น word graph ที่แสดงการเชื่อมต่อของผลการรู้จำ 5 รูปแบบ นั่นคือ 1) W_1 W_2 W_5 W_7 W_{10} 2) W_1 W_3 W_6 W_8 W_{10} 3) W_1 W_3 W_6 W_9 W_{10} 4) W_1 W_4 W_6 W_8 W_{10} 5) W_1 W_4 W_6 W_9 W_{10} ตามลำดับ โดยในที่นี้จะยังไม่ได้แสดงผลของเวลาของแต่ละคำรวมไปถึงค่าความเชื่อมั่น ซึ่งจะได้แสดงผลในส่วนของผลการทดสอบต่อไป และจากรูปที่ 1

จะเห็นว่าผลการรู้จำทั้ง 5 แบบนั้นจะมีการใช้คำ W_1 และ W_{10} ร่วมกัน และผลการรู้จำในแบบที่ 2 ถึง 4 จะใช้คำ W_6 ร่วมกัน โดยการใช้คำร่วมกันนั้นจะต้องเป็นคำที่มีช่วงของเวลาที่เกิดขึ้น และค่าความเชื่อมั่นค่าเดียวกันด้วย จึงจะถือว่าเป็นคำเดียวกัน แต่ถ้าหากว่ามีค่าใดค่าหนึ่งต่างกัน จะถือว่าไม่ได้ใช้คำร่วมกัน แม้ว่าคำที่เกิดขึ้นจะเป็นคำเดียวกันก็ตาม

โดยรายละเอียดในการสร้าง word graph ในขั้นตอนที่ทำการค้นหาย้อนกลับ ก็จะทำเช่นเดียวกับ การค้นหาผลการรู้จำแบบย้อนกลับในหัวข้อที่ 4.2.3.2 แต่เราจะเพิ่มการบันทึกข้อมูลของ wordgraph เช่น ข้อมูลของเวลา และ ค่าความเชื่อมั่นของแต่ละคำ เพิ่มลงไปทุกครั้งที่พิจารณาการค้นหาย้อนกลับของแต่ละคำเสร็จแล้วนั่นเอง

ตัวอย่างเช่น ผลการค้นหาประโยคที่ได้จากการประมวลผลแบบย้อนกลับที่ทำการรู้จำกับแฟ้มข้อมูลทดสอบที่บันทึกประโยคพูดคำว่า “พร้อมพงษ์” ที่ใช้ในการรู้จำสถานีปลายทางของการโดยสารรถไฟฟ้า BTS โดยใช้ไวยากรณ์แบบ FSG และให้จำนวนของผลการรู้จำที่ต้องการ คือ 5 ลำดับแรก (N = 5) ซึ่งข้อมูลทดสอบมีจำนวนของเฟรม(เวลา - *t*) ทั้งหมด คือ 258 เฟรม โดยผลคะแนนของประโยค (score) และ ค่าความเชื่อมั่นของแต่ละคำ (conf) จะเป็นดังนี้

1) ผลการรู้จำที่ดีที่สุดที่ได้จากการค้นหาแบบไปข้างหน้า (forward search) คือ

Sent : (sil PHROM_PHONG sil) มี score = -5389.309082

โดยที่ sil ตัวแรก มี conf = 0.9999

PHROM_PHONG มี conf = 0.759647

sil ตัวสุดท้าย มี conf = 1.0000

2) ผลการรู้จำที่ดีที่สุด 5 แบบ ที่ได้จากการค้นหาแบบย้อนกลับ (reverse search) คือ

2.1) Sent : (sil PHROM_PHONG sil) มี score = -5389.319336

โดยที่ sil ตัวแรก มี conf = 0.999789

PHROM_PHONG มี conf = 0.901396

sil ตัวสุดท้าย มี conf = 1.0000

2.2) Sent : (sil PHROM_PHONG sil) มี score = -5411.819336

โดยที่ sil ตัวแรก มี conf = 0.999789

PHROM_PHONG มี conf = 0.067588

sil ตัวสุดท้าย มี conf = 1.0000

2.3) Sent : (sil PHROM_PHONG KRUB sil)

มี score = -5427.428711

โดยที่ sil ตัวแรก มี conf = 0.999789

PHROM_PHONG มี conf = 0.930234

KRUB มี conf = 0.011205

sil ตัวสุดท้าย มี conf = 1.0000

2.4) Sent : (sil PHROM_PHONG KRUB sil)

มี score = -5427.817383

โดยที่ sil ตัวแรก มี conf = 0.999789

PHROM_PHONG มี conf = 0.930234

KRUB มี conf = 0.010714

sil ตัวสุดท้าย มี conf = 1.0000

2.5) Sent : (sil PHROM_PHONG KHA sil)

มี score = -5429.248047

โดยที่ sil ตัวแรก มี conf = 0.999789

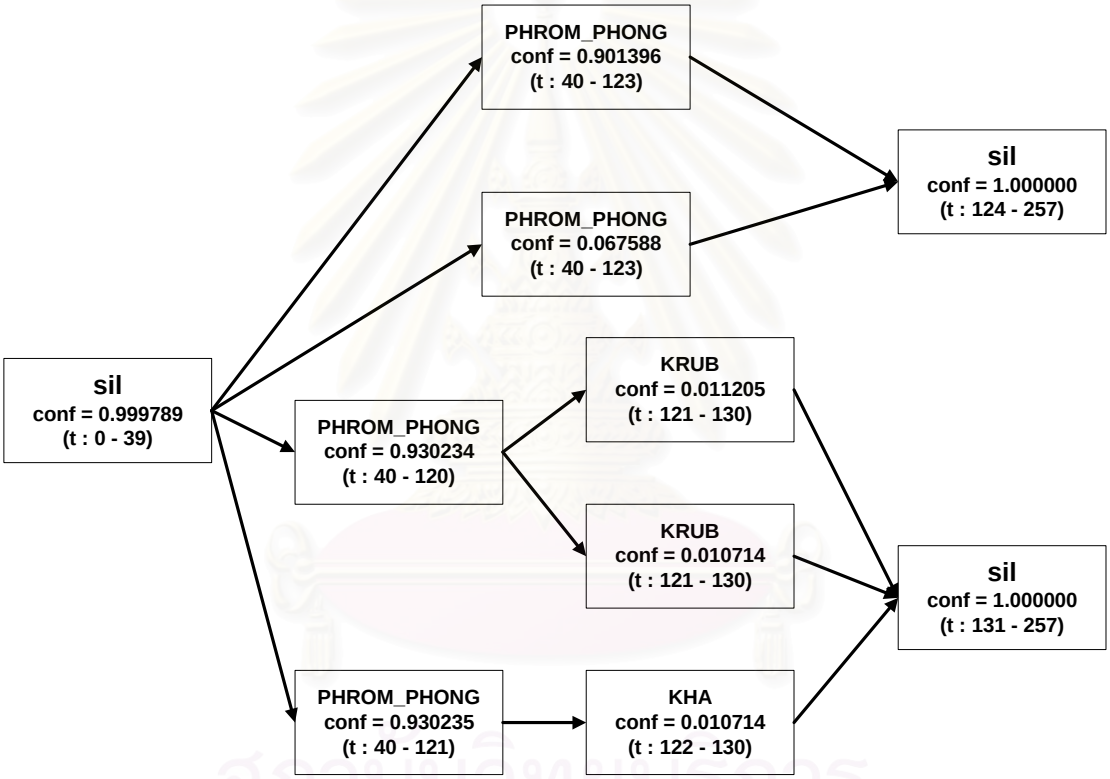
PHROM_PHONG มี conf = 0.930235

KHA มี conf = 0.09088

sil ตัวสุดท้าย มี conf = 1.0000

หมายเหตุ 1) sil คือ silence ที่แทนเสียงเงียบของการพูด 2) PHROM_PHONG คือ คำว่า "พร้อมพงษ์" 3) KRUB คือ คำว่า "ครับ" และ 4) KHA คือ คำว่า "ค่ะ"

ซึ่งผลการรู้จำที่ได้จากการค้นหาแบบย้อนกลับทั้ง 5 แบบนั้น สามารถนำมาเขียนเป็น wordgraph เพื่อให้ผู้ใช้สามารถดูลักษณะการเชื่อมต่อกันของคำในประโยคผลลัพธ์ทั้ง 5 แบบ จะเป็นดังรูปที่ 20



รูปที่ 20 การแสดงผล word graph ของผลการรู้จำที่ได้จากการค้นหาแบบย้อนกลับทั้ง 5 แบบ

จากผลที่ได้จะเห็นว่าผลการรู้จำทั้ง 5 แบบที่ได้จะมีการใช้คำบางส่วนร่วมกัน เช่น เสียงเงียบ (silence) เริ่มต้น หรือคำว่า PHROM_PHONG ที่เกิดขึ้นในเวลาตั้งแต่เฟรมที่ 40 ถึง 121 โดยจะเห็นว่าผลของ word graph ที่ได้จะมีเวลาของช่วงรอยต่อระหว่างคำที่ต่อเนื่องกันทั้งหมด

4.2.4 การเปรียบเทียบผลการรู้จำของตัวรู้จำแบบ HTK และ Julius

ในหัวข้อนี้จะเป็นการเปรียบเทียบผลการทดสอบประสิทธิภาพของการประมวลผลการรู้จำเสียงพูดแบบเป็นประโยค โดยจะเทียบวิธีที่ใช้ต้นไม้แบบ lexicon ในการรู้จำ คือ โปรแกรม Julius กับ โปรแกรม HTK ซึ่งเป็นโปรแกรมการรู้จำเสียงพูดที่ใช้กันอย่างแพร่หลาย โดยจะเปรียบเทียบเวลาของการประยุกต์ใช้ทั้งสองไวยากรณ์กับเสียงพูดประโยคต่างๆ แบบ offline โดยมีจำนวนคำในพจนานุกรมทั้งหมด 3,200 คำ ซึ่งผลที่ได้จากการรู้จำของทั้งสองวิธีนั้นจะได้ผลที่ถูกต้องในลักษณะเดียวกัน จึงขอไม่แสดงประโยคที่ได้จากการรู้จำในที่นี้ ซึ่งเวลาการประมวลผลต่างๆ จะเป็นดังตารางต่อไปนี้

ตารางที่ 10 ตารางการเปรียบเทียบเวลาการประมวลผลการรู้จำของโปรแกรม HTK และ โปรแกรม Julius สำหรับไวยากรณ์แบบ Finite State แบบ full search

Finite State Grammar(full search)			
แฟ้มข้อมูล	ความยาว (วินาที)	เวลาประมวลผล(วินาที)	
		HTK	Julius
testsentences02_001a_ekr.mfc	19.73	41	95
testsentences02_002a_ekr.mfc	25.29	53	125
testsentences02_003a_ekr.mfc	9.98	22	47
testsentences02_004a_ekr.mfc	25.11	53	124
testsentences02_005a_ekr.mfc	20.98	47	101
testsentences02_006a_ekr.mfc	20.04	43	96
testsentences02_007a_ekr.mfc	21.61	46	105
testsentences02_008a_ekr.mfc	21.61	48	104
testsentences02_009a_ekr.mfc	19.29	41	95
testsentences02_010a_ekr.mfc	19.98	46	96

จากตารางที่ 10 ซึ่งเป็นการเปรียบเทียบเวลาการประมวลผลการรู้จำของโปรแกรม HTK และโปรแกรม Julius ซึ่งใช้โครงสร้างต้นไม้แบบ lexicon ในการรู้จำสำหรับไวยากรณ์แบบ Finite State แบบไม่มีการตัดบางเส้นทางในการพิจารณาออก (full search) นั่นคือ จะทำอัลกอริทึม

Viterbi กับทุกเส้นทางที่เป็นไปได้ จะพบว่าระยะเวลาการประมวลผลของการประยุกต์ใช้ต้นไม้แบบ lexicon กับไวยากรณ์แบบ Finite State โดยโปรแกรม Julius จะใช้เวลาในการประมวลผลแฟ้มข้อมูลต่างๆ นานกว่าโปรแกรม HTK ถึงแม้ว่าจำนวน node(state) ของ HMM ที่ใช้ในการสร้างเป็นโครงข่ายของคำของโปรแกรม HTK คือ 28,833 node ซึ่งมากกว่าจำนวน node ที่ใช้ในโปรแกรม Julius คือ 19,911 node โดยสาเหตุที่ทำให้การประมวลผลของโปรแกรม Julius ใช้เวลามากกว่าโปรแกรม HTK เนื่องจากจะมีจำนวนครั้งการพิจารณาการส่งผ่านโทเคน ระหว่างคำ (link) มากกว่าโปรแกรม HTK ในกรณีที่ไวยากรณ์มีการวนซ้ำ โดยขณะนี้โปรแกรม Julius จะส่งผ่านโทเคนระหว่างคำ ไปยังทุกต้นไม้ lexicon ของกลุ่มคำที่เชื่อมต่อกันทั้งหมด ทำให้จำนวนการส่งผ่านจะมากกว่าโปรแกรม HTK ที่จะมีคำพิเศษมาคั่นระหว่างกลาง เพื่อทำให้จำนวน link ลดลง ซึ่งจะทำให้มีจำนวน link เหลือเพียง 12,804 link เท่านั้น โดยเวลาการประมวลผลของโปรแกรม Julius จะสามารถลดลงได้อย่างมาก เมื่อมีการจำกัดจำนวนของโทเคนที่พิจารณาในแต่ละเวลาดังตารางที่ 11 ซึ่งกำหนดให้พิจารณาโทเคนที่จะส่งผ่านที่แต่ละเวลาเพียง 250 โทเคน เท่านั้น ($N_{beam} = 250$) ซึ่งจะเห็นว่าจะใช้เวลาการประมวลผลลดลงมาประมาณ 10 เท่า แต่ในที่นี้จะไม่แสดงผลเปรียบเทียบกับโปรแกรม HTK เนื่องจากใช้วิธีการลดการคำนวณ ที่แตกต่างกัน นั่นคือโปรแกรม HTK จะใช้ความกว้างบีมของช่วงคะแนน

ตารางที่ 11 ตารางการเปรียบเทียบเวลาการประมวลผลการรู้จำของโปรแกรม Julius สำหรับไวยากรณ์แบบ Finite State แบบมีการ pruning โดยใช้ $N_{beam} = 250$

Finite State Grammar(pruning)		
แฟ้มข้อมูล	ความยาว	เวลาประมวลผล
testsentences02_001a_ekr.mfc	19.73	9
testsentences02_002a_ekr.mfc	25.29	12
testsentences02_003a_ekr.mfc	9.98	4
testsentences02_004a_ekr.mfc	25.11	11
testsentences02_005a_ekr.mfc	20.98	10
testsentences02_006a_ekr.mfc	20.04	9
testsentences02_007a_ekr.mfc	21.61	10
testsentences02_008a_ekr.mfc	21.61	10
testsentences02_009a_ekr.mfc	19.29	9
testsentences02_010a_ekr.mfc	19.98	9

ตารางที่ 12 ตารางการเปรียบเทียบเวลาการประมวลผลการรู้จำของโปรแกรม HTK และโปรแกรม Julius สำหรับไวยากรณ์แบบ N-gram แบบ full search

N-gram(full search)			
แฟ้มข้อมูล	ความยาว (วินาที)	เวลาประมวลผล(วินาที)	
		HTK	Lexicon tree
testsentences02_001a_ekr.mfc	19.73	69	57
testsentences02_002a_ekr.mfc	25.29	80	66
testsentences02_003a_ekr.mfc	9.98	35	27
testsentences02_004a_ekr.mfc	25.11	87	76
testsentences02_005a_ekr.mfc	20.98	70	59
testsentences02_006a_ekr.mfc	20.04	64	56
testsentences02_007a_ekr.mfc	21.61	71	61
testsentences02_008a_ekr.mfc	21.61	70	60
testsentences02_009a_ekr.mfc	19.29	67	54
testsentences02_010a_ekr.mfc	19.98	70	57

จากตารางที่ 12 ซึ่งเป็นการเปรียบเทียบเวลาการประมวลผลการรู้จำของโปรแกรม HTK และโปรแกรม Julius สำหรับไวยากรณ์แบบ N-gram เมื่อทำการค้นหาทุกเส้นทาง (full search) จะเห็นว่าเวลาการประมวลผลของการนำต้นไม้แบบ lexicon มาใช้ จะต่ำกว่าโปรแกรม HTK ในทุกๆ แฟ้มข้อมูล เนื่องจากว่าการพิจารณาไวยากรณ์แบบ N-gram จะพิจารณาคู่ของคำที่เกิดขึ้นในพจนานุกรมจาก 2-gram ทำให้จำนวน link ระหว่างคำ นั้นจะมีจำนวนที่ใกล้กัน โดยวิธีนี้จำนวนของ node ในต้นไม้แบบ lexicon ที่น้อยกว่ากว่า จำนวน node(state) ในโครงข่ายของคำ ดังที่ได้กล่าวมาข้างต้นทำให้วิธีที่นำต้นไม้แบบ lexicon มาใช้โดยโปรแกรม Julius จึงมีประสิทธิภาพในการประมวลผลเชิงเวลามากกว่าโปรแกรม HTK

4.2.5 รูปแบบของไวยากรณ์ตามมาตรฐานของ W3C

การจะนำระบบตัวรู้จำที่ได้ศึกษาพัฒนาไปใช้จริงในทางปฏิบัตินั้น จำเป็นจะต้องเรียนรู้และศึกษาการกำหนดลักษณะไวยากรณ์ที่เป็นมาตรฐาน โดยเฉพาะอย่างยิ่งไวยากรณ์แบบ Finite State จากที่ผ่านมาจะเห็นว่าลักษณะการแสดงของทั้งตัวรู้จำแบบ HTK และ Julius จะมีความแตกต่างกัน ทั้งในลักษณะของการเรียงลำดับตัวแปรที่เกิดขึ้นในไวยากรณ์ย่อย และสัญลักษณ์พิเศษดังที่ได้กล่าวข้างต้น ดังนั้นเพื่อให้ผู้ใช้แต่ละคนสามารถทำความเข้าใจกับลักษณะไวยากรณ์

ที่มีความเป็นมาตรฐานมากขึ้น ในการวิจัยนี้จึงได้ศึกษาลักษณะไวยากรณ์ที่มีลักษณะหรือมาตรฐานตามที่องค์กร W3C เป็นผู้กำหนด

ลักษณะของไวยากรณ์แบบ Finite State ที่ W3C เป็นผู้กำหนดนั้น จะมีลักษณะคล้ายคลึงกับตัวรู้จำแบบ HTK มาก แต่จะมีส่วนเพิ่มเติมและข้อแตกต่างอยู่บ้าง โดยส่วนที่เหมือนกัน คือ สัญลักษณ์พิเศษต่างๆ จะใช้ในลักษณะเดียวกันกับตัวรู้จำแบบ HTK เช่น (…) และ […] แต่การพิจารณาการวนซ้ำของไวยากรณ์จะใช้ลักษณะการบอกเป็นตัวเลขแทน เช่น หากต้องการให้มีการวนซ้ำจะใช้การระบุด้วยสัญลักษณ์ <…> แล้วระบุด้วยจำนวนครั้งแทน เช่น < n1-n2 > จะหมายถึงการวนจำนวนรอบตั้งแต่ n1 ถึง n2 รอบ ถ้าหากต้องการให้วนซ้ำแบบไม่จำกัดรอบก็สามารถทำได้โดยไม่ต้องระบุ n2 คือ < n1 - > ซึ่งจะหมายถึงการวนซ้ำตั้งแต่ n1 รอบเป็นต้นไป โดยถ้าระบุเป็น < n1 > จะหมายถึงการวนซ้ำด้วยจำนวน n1 รอบ ซึ่งจะเห็นว่ามีความสะดวกในการระบุจำนวนรอบกับผู้ใช้มากกว่าการระบุแบบ HTK

โดยสัญลักษณ์ {…} จะยังคงถูกนำมาใช้ในไวยากรณ์แบบ W3C แต่จะเป็นการบอกถึงรายละเอียดของไวยากรณ์นั้นๆ แทนซึ่งจะไม่มีผลกับการรู้จำแต่อย่างใด แต่จะมีผลกับการนำผลของการรู้จำไปใช้ เช่นการระบุว่าคำใดเป็นคำสำคัญ (keyword) ที่ต้องการนำมาใช้เป็นผลลัพธ์บ้าง ซึ่งทำให้เราสามารถตีความประโยคที่ได้จากการรู้จำได้ดียิ่งขึ้น เนื่องจากว่าคำที่ประโยครู้จำได้อาจจะมีบางคำที่ไม่ได้เป็นคำสำคัญที่ต้องการก็เป็นได้

ลักษณะของไวยากรณ์แบบ Finite State ที่เป็นไปตามมาตรฐานของ W3C สำหรับการโทรศัพท์สั่งพิซซ่าและเครื่องดื่มจะเป็นดังรูปที่ 21

```
#ABNF 1.0 UTF-8;
language en;
mode voice;
tag-format <semantics/1.0>;
root $order;
$order = I would like a $drink {out.drink = new Object();
         out.drink.liquid = rules.drink.type;
         out.drink.drinksize = rules.drink.drinksize;}
         and $pizza {out.pizza=rules.pizza;};
$kindofdrink = coke | pepsi | "coca cola"{out="coke";};

// "medium" is default if nothing said
$foodsize = {out="medium";}
            [small {out="small";} | medium |
            large {out="large";}| regular {out="medium";}];

// Construct Array of toppings, return Array
$tops = {out=new Array;} $top {out.push(rules.top);}
        (and $top {out.push(rules.top);})<1->;
$top = anchovies | pepperoni | mushroom{out="mushrooms";} | mushrooms;

// Two properties (drinksize, type) on left hand side Rule Variable
$drink = $foodsize $kindofdrink
         {out.drinksize=rules.foodsize; out.type=rules.kindofdrink; };

// Three properties on rules.pizza's Rule Variable
$pizza = $number $foodsize
         {out.pizzasize=rules.foodsize; out.number=rules.number;} pizzas
         with $tops {out.topping=rules.tops;};
$number = (a | one){out="1";} | two{out="2";} | three{out="3";};
```

รูปที่ 21 ไวยากรณ์แบบ Finite State ที่เป็นไปตามมาตรฐานของ W3C

จากลักษณะของไวยากรณ์ตามรูปที่ 21 จะเห็นว่าไวยากรณ์ตามแบบ W3C จะมีส่วนหนึ่งที่เพิ่มมา คือ ส่วนเฮดเดอร์ของไฟล์ ซึ่งจะมีส่วนสำคัญที่ระบุคือ root $order; โดยหมายความว่าไวยากรณ์หลัก คือ $order นั่นเอง ส่วนลำดับการเรียงของไวยากรณ์จะไม่เรียงลำดับจากด้านบนลงด้านล่างแบบ HTK หรือจากด้านล่างขึ้นไปด้านบน โดยจะเขียนอยู่ตรงส่วนไหนของแฟ้มข้อมูลก็ได้

จากรูปที่ 21 จะเห็นว่าไวยากรณ์ $order จะมีการอ้างอิงถึงไวยากรณ์ $drink และ $pizza โดยรายละเอียดของไวยากรณ์ $drink ใน {…} ที่จะถูกสร้างใน $order คือ บอกว่าจะสร้างผลลัพธ์ ชื่อ drink จาก out.drink = new Object(); และใน drink จะมีผลลัพธ์ย่อย คือ liquid จาก out.drink.liquid = rules.drink.type; ซึ่งจะได้รับมาจากผลลัพธ์ย่อย type จากไวยากรณ์ $drink นอกจากนี้ใน drink จะมีผลลัพธ์ย่อยอีกตัว คือ drinksize จาก out.drink.drinksize = rules.drink.drinksize; โดยจะได้รับมาจากผลลัพธ์ย่อย drinksize จากไวยากรณ์ $drink ซึ่งตัวไวยากรณ์ $drink จะประกอบไปด้วยไวยากรณ์ย่อย คือ $foodsize และ $kindofdrink โดยใน

รายละเอียด {…} บอกว่าจะมีผลลัพธ์ย่อย ชื่อ drinksize รับผลลัพธ์มาจาก $foodsize และ kindofdrink รับผลลัพธ์มาจาก $kinkofdrink จะเห็นว่าลักษณะของผลลัพธ์ดังกล่าวเป็นการอ้างถึงผลลัพธ์กันเป็นทอดๆ ในกรณีที่ไวยากรณ์ประกอบไปด้วยไวยากรณ์ย่อยหลายชั้น

ส่วนไวยากรณ์ย่อยอีกตัวที่ถูกอ้างถึงใน $order คือ $pizza โดยจะมีผลลัพธ์ชื่อ pizza ที่อ้างจากผลลัพธ์ของ $pizza โดยใน $pizza ก็จะมีการอ้างถึงไวยากรณ์ย่อย $number $foodsize และ $tops โดยการอ้างถึงแต่ละไวยากรณ์จะมีลักษณะเหมือนที่ได้กล่าวมาแล้วเกือบทั้งสิ้น ยกเว้นไวยากรณ์ $tops ที่จะมีการสร้างอาร์เรย์ให้กับคำตอบ คือ out = new Array; เนื่องจากไวยากรณ์ $top ที่อ้างถึง จะถูกอ้างถึงแบบวนซ้ำ จากสัญลักษณ์พิเศษ <1-> หมายถึงการวนซ้ำตั้งแต่ 1 ครั้งเป็นต้นไป ดังนั้นผลลัพธ์จากไวยากรณ์ $tops จะประกอบไปด้วยคำตอบจากไวยากรณ์ $top หลายๆ ตัวในอาร์เรย์จากการระบุ out.push(rules.top); นั่นเอง

โดยผลลัพธ์นั้นเราสามารถกำหนดคำที่ต้องการจะแสดงให้กับคำในไวยากรณ์ได้ด้วย out = “…” ซึ่งทำให้คำที่แสดงผลลัพธ์จะเป็นคำทั้งหมดที่อยู่ระหว่าง “ ” นั่นเอง และจากตัวอย่างนี้จะมีการระบุรายละเอียดโดยลักษณะดังกล่าวที่น่าสนใจ คือ

```
$foodsize = {out="medium";}
        [small {out="small";} | medium |
        large {out="large";}| regular {out="medium";}];
```

จะเห็นว่า $foodsize นั้นจะมีการกำหนดคำที่ต้องการแสดง คือ medium ที่หลังเครื่องหมายเท่ากับ นั่นหมายถึงว่า medium จะเป็นผลลัพธ์ของไวยากรณ์นี้เมื่อมีการอ้างถึงไวยากรณ์ดังกล่าวในการระบุคำสำคัญ (default output) แล้วไม่ปรากฏคำที่อยู่ในไวยากรณ์นี้ เช่น กรณีที่มีการระบุสัญลักษณ์พิเศษ […] ดังไวยากรณ์นี้นั่นเอง

โดยถ้าหากผลลัพธ์ที่ได้จากการรู้จำลักษณะประโยคข้างต้นเป็น

"I would like a coca cola and three large pizzas with pepperoni and mushrooms."

เราจะได้คำสำคัญจากประโยคนี้ คือ

```
{
  drink: {
    liquid:"coke",
    drinksize:"medium"},
  pizza: {
    number: "3",
    pizzasize: "large",
    topping: [ "pepperoni", "mushrooms" ]
  }
}
```

รายละเอียดของไวยากรณ์แบบ Finite State ตามที่ได้กล่าวมาข้างต้น จะเป็นลักษณะที่ระบุตามมาตรฐานขององค์กร W3C ที่ให้ความสะดวกกับผู้ใช้ในการนำสำคัญจากประโยคที่ได้จากการรู้จำมาเป็นผลลัพธ์ โดยสามารถหารายละเอียดเพิ่มเติมได้จาก http://www.w3.org

โดยในการวิจัยนี้ได้พัฒนาโปรแกรมในการสร้างแลตทิซจากไวยากรณ์ตามมาตรฐานขององค์กร W3C รวมทั้งโปรแกรมในการดึงคำสำคัญมาจากประโยคผลลัพธ์ของการรู้จำเป็นที่เรียบร้อยแล้ว

4.2.6 สรุปการวิจัยการพัฒนาระบบรู้จำเสียงพูดแบบต่อเนื่อง

ในการวิจัยนี้ได้ทำการศึกษาและพัฒนาระบบรู้จำเสียงพูดแบบต่อเนื่องเพื่อให้สามารถนำมาใช้งานได้จริง โดยใช้โปรแกรม HTK และโปรแกรม Julius ที่ใช้โครงสร้างต้นไม้แบบ lexicon เป็นต้นแบบในการพัฒนาการรู้จำ โดยโปรแกรมทั้งสองรูปแบบรองรับไวยากรณ์ทั้งในแบบ Finite State และ N-gram ซึ่งขึ้นอยู่กับลักษณะรูปแบบประโยคที่ผู้ใช้ต้องการ โดยการวิจัยได้ศึกษาขั้นตอนการประมวลผลของตัวรู้จำทั้งสองแบบ ทั้งการประมวลผลแบบไปข้างหน้า และย้อนกลับเพื่อหาผลการรู้จำ โดยโปรแกรม Julius สามารถประมวลผลได้รวดเร็วกว่าโปรแกรม HTK เนื่องจากใช้ลักษณะโครงสร้างต้นไม้แบบ lexicon ในการลดจำนวน HMM ที่ต้องพิจารณาลง ซึ่งโครงสร้างนี้จะเหมาะสมในการรู้จำเสียงพูดที่ใช้ชุดคำศัพท์จำนวนมาก แต่สำหรับกรณีไวยากรณ์แบบ Finite State จะใช้เวลาในการประมวลผลนานกว่า เนื่องจากโปรแกรม HTK มีการคำนึงถึงการลดจำนวน link ที่ต้องพิจารณา แต่โปรแกรม Julius ก็สามารถประมวลผลเร็วขึ้นได้ และสามารถจำกัดการคำนวณได้ดีกว่าโปรแกรม HTK (แต่อาจสูญเสียความสามารถในการรู้จำลง) เนื่องจากใช้การจำกัดจำนวนเส้นทางที่พิจารณา ในขณะที่โปรแกรม HTK ใช้ความกว้างบีมของคะแนนในการลดการคำนวณ ซึ่งไม่สามารถลดจำนวนเส้นทางได้อย่างแน่นอน

โดยการวิจัยนี้ได้พัฒนาการสร้างแลตทิซของคำจากทั้งไวยากรณ์แบบ Finite State และ N-gram สำหรับตัวรู้จำแบบ HTK โดยสามารถอ่านรูปแบบไวยากรณ์ที่มีความเป็นสากลมากขึ้นตามแบบขององค์กร W3C ได้ ซึ่งมีลักษณะพิเศษที่สามารถระบุคำสำคัญที่ต้องการให้เป็นผลของการรู้จำได้ ทำให้เหมาะสมในการนำไปใช้งานจริงมากขึ้น นอกจากนี้ยังได้พัฒนาการสร้างไวยากรณ์แบบ Finite State รูปแบบ DFA สำหรับโปรแกรม Julius ด้วยลักษณะการแสดงแบบโปรแกรม HTK ซึ่งมีความสามารถในการเขียนไวยากรณ์ได้หลากรูปแบบกว่า

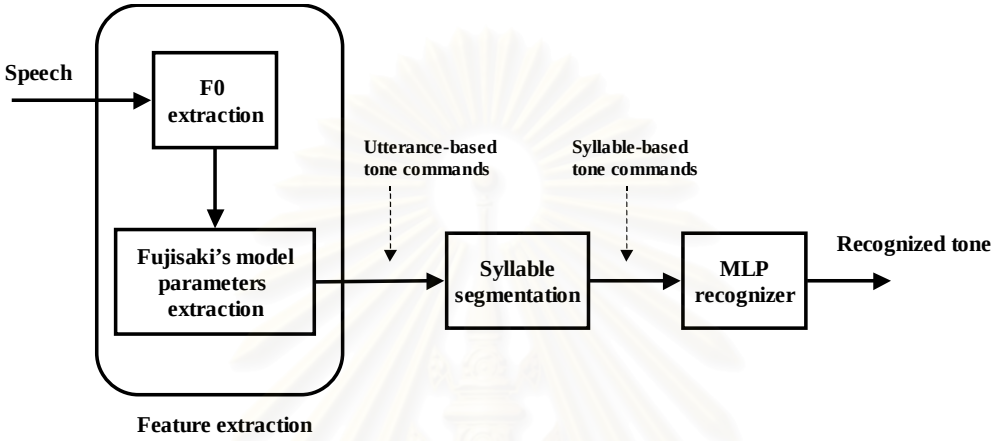
4.3 การรู้จำวรรณยุกต์เสียงพูดต่อเนื่องภาษาไทย

จากที่ได้กล่าวมาในหัวข้อที่แล้ว เรื่องการรู้จำเสียงพูดแบบต่อเนื่องสำหรับภาษาไทยนั้นจะเห็นว่า เราเน้นการรู้จำไปที่ส่วนประกอบของพยางค์ S ที่สามารถเขียนองค์ประกอบในรูปของหน่วยเสียงได้ดังนี้ คือ C V (F) แต่ยังไม่ได้เน้นไปที่ลักษณะเด่นอีกอย่างหนึ่งของคำหรือพยางค์ดังกล่าว นั่นคือ เสียงวรรณยุกต์ ของพยางค์ดังกล่าวนั่นเอง (T - Tone) ซึ่งองค์ประกอบดังกล่าวเป็นลักษณะพิเศษของภาษาไทย ที่ถึงแม้ว่าพยางค์หรือคำใดๆ มีองค์ประกอบของเสียงพยัญชนะ สระ และตัวสะกด เหมือนกัน แต่ถ้าหากมีเสียงวรรณยุกต์ที่แตกต่างกันแล้ว ก็จะถือว่าเป็นคนละคำกัน เช่น คำว่า การ กับ ก้าน นอกจากจะมีความแตกต่างกันในด้านความหมายแล้วยังมีความแตกต่างกันที่เสียงวรรณยุกต์ คือ การ จะมีเสียงวรรณยุกต์สามัญ ส่วน ก้าน จะมีเสียงวรรณยุกต์โท โดยมีรูปพยัญชนะ สระ และตัวสะกด เหมือนกัน ดังนั้นหากเราสามารถตรวจสอบได้ว่าคำหรือพยางค์ที่พูดนั้นมีเสียงวรรณยุกต์แบบใด ก็จะสามารถแยกแยะความแตกต่างของคำทั้งสองได้ดียิ่งขึ้น ซึ่งก็จะทำให้การรู้จำเสียงพูดมีความสมบูรณ์แบบมากยิ่งขึ้น

4.3.1 ระบบรู้จำวรรณยุกต์เสียงพูดต่อเนื่องภาษาไทย

จากการศึกษาเสียงพูด พบว่าส่วนของวรรณยุกต์ของเสียงต่าง ๆ เกิดขึ้นตามรูปแบบพลวัตของความถี่มูลฐานของเสียงพูด (Dynamic of fundamental frequency) เป็นอย่างมาก จึงได้มีการนำเส้นโค้งความถี่มูลฐาน (F0 contour) มาใช้ในการพิจารณาสร้างระบบรู้จำวรรณยุกต์เสียงพูดด้วยแบบจำลองต่างๆ กัน จากการศึกษาของปฐวี ชาญไววิทย์ ในการรู้จำวรรณยุกต์เสียงพูดพยางค์เดี่ยว โดยใช้ Polynomial Regression พบว่าสามารถทำงานได้เป็นอย่างดี อย่างไรก็ตาม วิธีเดียวกันนี้ เมื่อนำมาใช้กับการรู้จำวรรณยุกต์ของเสียงพูดต่อเนื่อง จะให้ผลการรู้จำที่ต่ำลงอย่างมาก อันเป็นผลเนื่องมาจาก Tonal assimilation ของพยางค์ข้างเคียง งานวิจัยนี้จึงทำการศึกษาวิธีการรู้จำวรรณยุกต์เสียงพูดต่อเนื่องภาษาไทยเพื่อใช้ร่วมกับการรู้จำเสียงพูดต่อเนื่องภาษาไทยระดับหน่วยย่อยของเสียงพูดที่ได้กล่าวมาแล้วในบทที่ผ่านมา

จากการศึกษาวิธีการและตัวแบบต่าง ๆ ทำให้มีการเลือกแบบจำลองฟูจิซากิ (Fujisaki's Model) เป็นแบบจำลองเพื่อทดสอบความเป็นไปได้ในการใช้แบบจำลองดังกล่าวในการรู้จำวรรณยุกต์ของเสียงพูด ซึ่งแสดงในรูปที่ 22



รูปที่ 22 การใช้ Fujisaki's Model ในการรู้จำวรรณยุกต์เสียงพูด

เหตุผลที่การวิจัยนี้เลือกที่จะนำพารามิเตอร์ของแบบจำลองฟูจิซากิ มาใช้ในการวิจัยเพื่อทำการรู้จำวรรณยุกต์ของเสียงพูดต่อเนื่องภาษาไทยเนื่องจาก

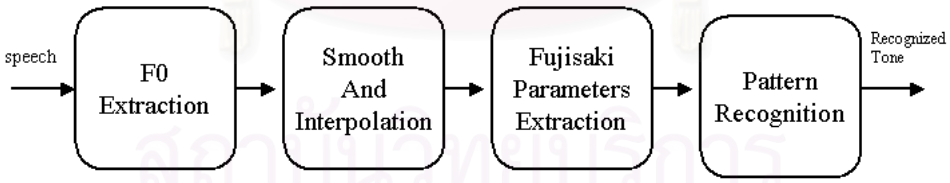
- แบบจำลอง Fujisaki สร้างขึ้นจากการเลียนแบบการทำงานของอวัยวะในการสร้างเสียงพูด (ส่วนที่เกี่ยวข้องกับโทนเสียง) จริง จึงน่าที่จะใช้เป็นตัวแทนของโทนเสียงได้เป็นอย่างดี
- เหมาะสมกับเสียงพูดต่อเนื่องที่มีการพูดลักษณะเป็นธรรมชาติ
- มีจำนวนพารามิเตอร์น้อย ทำให้การคำนวณในส่วนของ Classification เป็นไปอย่างรวดเร็ว
- มีการพิจารณาถึง Declination Effect และ Tonal Assimilation Effect ไว้แล้วในแบบจำลอง
- ค่าของพารามิเตอร์หลังการ Normalize จะไม่ขึ้นกับความยาวของพยางค์ และไม่ขึ้นกับลักษณะเสียงสระสั้น-ยาว

ในเบื้องต้นของการวิจัยได้ทดสอบอัตราการรู้จำเมื่อนำแบบจำลองฟูจิซากิมาใช้ในการรู้จำเปรียบเทียบกับการใช้วิธี Polynomial regression และได้ผลดังตารางที่ 13

ตารางที่ 13 อัตราการรู้จำเสียงวรรณยุกต์ของวิธี Polynomial regression และ Fujisaki's Model

	Speaker-dependent		Speaker-independent	
	Polynomial regression	Fujisaki's Model	Polynomial regression	Fujisaki's Model
Recognition Rate	79.2%	82.6%	31.3%	74.3%

จะเห็นว่าการนำแบบจำลองฟูจิซากิมาใช้ในการรู้จำวรรณยุกต์ของเสียงพูดแบบต่อเนื่องนั้นให้ผลที่ดีและเหมาะสมที่จะใช้แบบจำลองดังกล่าวในงานวิจัยนี้ ซึ่งจะใช้แบบจำลองฟูจิซากิเป็นพื้นฐานในการหาคุณลักษณะ และใช้โครงข่ายประสาทเทียม (Neural Network) ในการจำแนก (classified) ออกเป็นวรรณยุกต์ต่างๆ โดยโครงสร้างของระบบรู้จำวรรณยุกต์ของเสียงพูดจะเป็นดังรูปที่ 23



รูปที่ 23 โครงสร้างระบบรู้จำวรรณยุกต์ของเสียงพูด

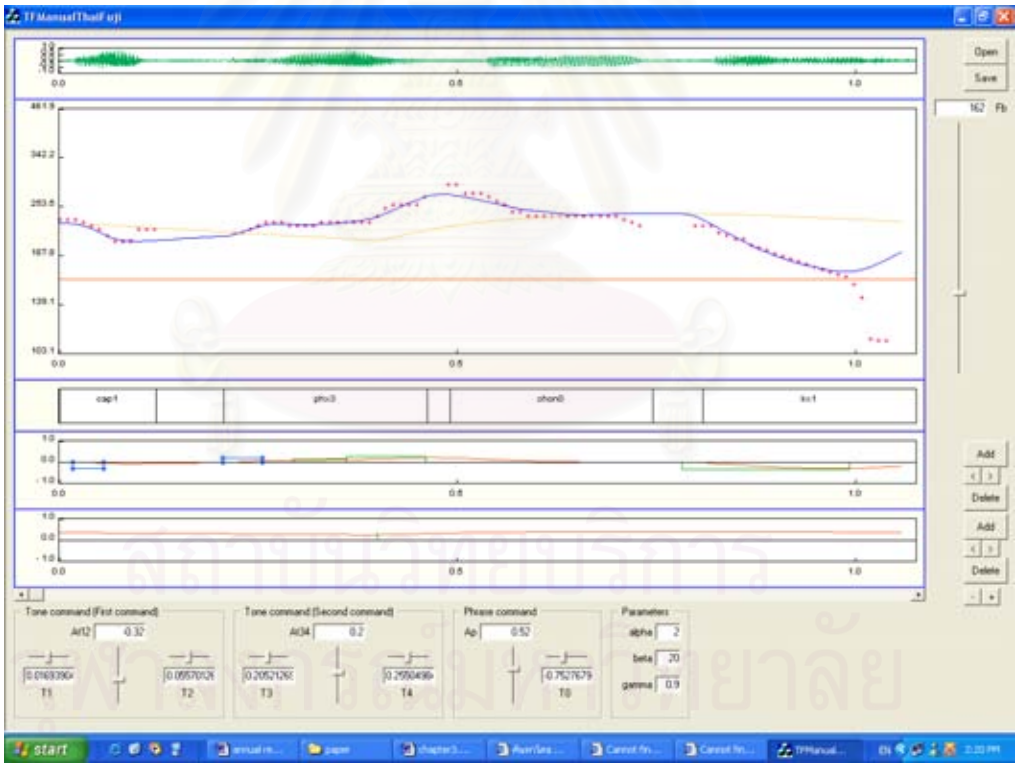
จากรูปที่ 23 เบื้องต้นเสียงพูดจะถูกนำไปคำนวณหาเส้นโค้งความถี่มูลฐาน จากนั้นจึงผ่านการ smoothing และ Interpolation เพื่อลดสัญญาณที่อาจเกิดจากสัญญาณรบกวน ซึ่งถือเป็นกระบวนการประมวลผลก่อนหน้า จากนั้นทำการแยกคุณลักษณะด้วยการแยกพารามิเตอร์ของแบบจำลองฟูจิซากิ และขั้นตอนสุดท้ายเป็นการทำการจำแนก (Classification) โดยใช้โครงข่ายประสาทเทียม

กระบวนการแยกพารามิเตอร์ที่ทำการทดสอบประกอบด้วยการแยกพารามิเตอร์แบบ manual และการแยกพารามิเตอร์แบบอัตโนมัติอีก 3 วิธี คือ การแยกพารามิเตอร์ตามวิธีของ Hansjorg Mixdorff, การแยกพารามิเตอร์โดยไม่ใช้ syllable boundary และการแยกพารามิเตอร์โดยใช้ syllable boundary

การแยกพารามิเตอร์แบบ manual นั้นเป็นการศึกษาความเป็นไปได้ในการนำพารามิเตอร์ของแบบจำลองฟูจิซากิมาใช้เป็นคุณลักษณะในการรู้จำวรรณยุกต์ของเสียงพูด

4.3.1.1การแยกพารามิเตอร์แบบ Manual

การแยกพารามิเตอร์แบบ Manual กระทำผ่านโปรแกรมที่พัฒนาขึ้นภายในห้องปฏิบัติการวิจัยกรรมวิธีสัญญาณดิจิทัลซึ่งแสดงหน้าจอการทำงานได้ดังรูปที่ 24

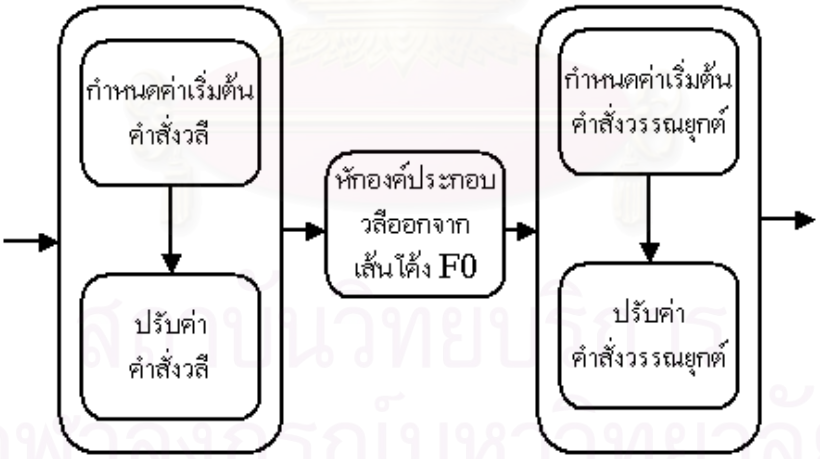


รูปที่ 24 หน้าจอการทำงานของโปรแกรมหาค่าพารามิเตอร์ที่ได้พัฒนา

การหาพารามิเตอร์แบบ Manual ประกอบด้วยขั้นตอนต่างๆ ดังนี้

- พิจารณาการเปลี่ยนแปลงโดยรวมของ F0 Contour ที่ได้จากเสียงพูด และกำหนดค่าเริ่มต้นสำหรับ Phrase command
- ปรับค่า Phrase command ทั้งในส่วนของเวลา และแมกนิจูดให้ Phrase component ที่สร้างขึ้นมีค่าใกล้เคียงกับการเปลี่ยนแปลงโดยรวมของ F0 Contour ที่ได้จากเสียงพูด
- นำ Phrase component ที่ได้จาก Phrase command ที่ปรับแล้วไปหักออกจาก F0 Contour
- กำหนดค่าเริ่มต้นสำหรับ Tone command โดยพิจารณาจากเส้นโค้งส่วนที่เหลือจากการหัก Phrase component ออกจาก F0 Contour
- ปรับค่า Tone command ทั้งเวลาเริ่มต้น เวลาสิ้นสุด และแมกนิจูดจนมี F0 Contour ที่สร้างขึ้นจากการรวมกันของ Phrase component และ Tone component ใกล้เคียงกับ F0 Contour ที่ได้จากเสียงพูดมากที่สุด

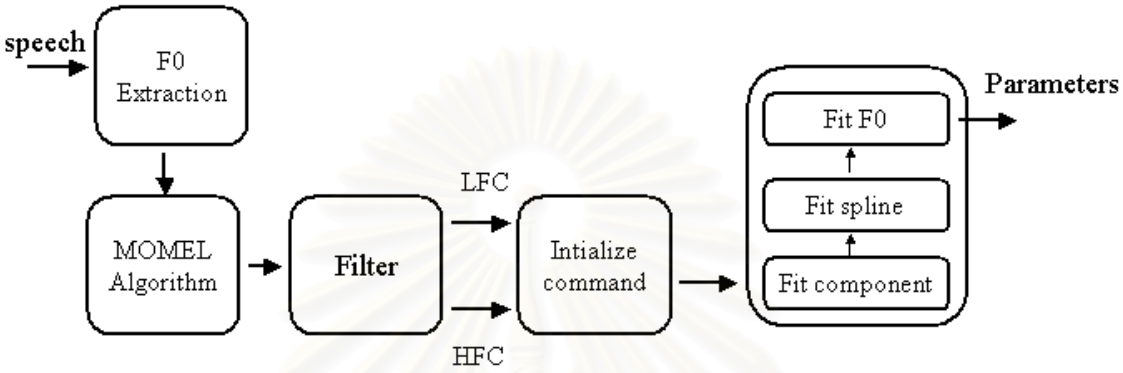
การหาพารามิเตอร์แบบ manual แสดงได้ดังรูปที่ 25



รูปที่ 25 ขั้นตอนการหาพารามิเตอร์แบบ manual

4.3.1.2 การแยกพารามิเตอร์ตามวิธีของ Mixdorff

การแยกพารามิเตอร์ตามวิธีของ Mixdorff เป็นวิธีการแยกพารามิเตอร์โดยอัตโนมัติ ซึ่งแสดงได้ดังรูปที่ 26

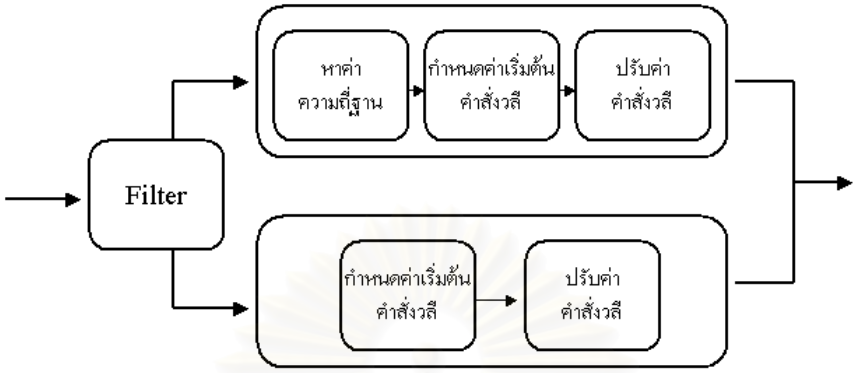


รูปที่ 26 แผนภาพแสดงการแยกพารามิเตอร์ตามกรรมวิธีของ Mixdorff

ตามวิธีของ Mixdorff การแยกพารามิเตอร์สำหรับแบบจำลองฟูจิซากิจาก F0 Contour ที่ได้ประกอบด้วย 4 ขั้นตอนหลักคือ

- Quadratic Spline Stylisation เป็นการทำ Smoothing และ Interpolation โดยใช้ MOMEL Algorithm
- Filtering and Component Separation เป็นการแยก F0 Contour ที่ได้หลังจากทำ MOMEL Algorithm ออกเป็น 2 ส่วนโดยใช้วงจรกรอง องค์ประกอบ 2 ส่วนที่แยกได้เรียกว่า LFC (Low Frequency Contour) และ HFC (High Frequency Contour)
- ให้ค่าเริ่มต้นของแบบจำลอง (Fujisaki Model Command Initialization) โดยค่าเริ่มต้นของ Phrase command พิจารณาจาก LFC และค่าเริ่มต้นของ Tone command พิจารณาจาก HFC
- ปรับค่าพารามิเตอร์ของแบบจำลอง โดยทำการปรับค่าทั้งสิ้น 3 ครั้งคือ
    - ปรับค่าเทียบกับ HFC และ LFC
    - ปรับค่าเทียบกับ F0 Contour ที่ได้หลังจาก MOMEL Algorithm
    - ปรับค่าเทียบกับ F0 Contour ตั้งต้น

4.3.1.3 การแยกพารามิเตอร์โดยไม่ใช้ Syllable boundary



รูปที่ 27 แผนภาพแสดงการแยกพารามิเตอร์โดยไม่ใช้ขอบเขตพยางค์

การหาพารามิเตอร์โดยไม่ใช้ขอบเขตพยางค์ เป็นการหาค่าพารามิเตอร์โดยอัตโนมัติ ซึ่งข้อมูลที่ใช้ในการหาพารามิเตอร์มีเพียง F0 Contour เพียงอย่างเดียว โดยเป็นการดัดแปลงวิธีจากการหาค่าพารามิเตอร์ตามวิธีของ Hansjorg Mixdorff ซึ่งข้อแตกต่างหลักคือการไม่ใช้อัลกอริทึม MOMEL เนื่องจากพบว่า การนำอัลกอริทึม MOMEL มาใช้กับเสียงในภาษาไทย จะทำให้การเปลี่ยนแปลงของ F0 Contour ถูกทำให้เรียบจนข้อมูลเกี่ยวกับวรรณยุกต์หายไปมาก

การหาพารามิเตอร์โดยไม่ใช้ขอบเขตพยางค์ประกอบด้วยขั้นตอนต่าง ๆ ดังนี้

- ทำ Neutralizetion และ Median Filtering เพื่อทำการลด Noise ต่าง ๆ ที่เกิดขึ้น
- ทำ Linear Interpolation ในส่วนที่ไม่ปรากฏค่า F0
- ทำการ Filter เพื่อแยก F0 Contour ที่ได้หลังจากการทำ Linear Interpolation ออกเป็น 2 ส่วนคือ HFC และ LFC
- กำหนดค่าตั้งต้นให้กับ Phrase command โดยพิจารณาจาก LFC
- กำหนดค่าตั้งต้นให้กับ Tone command โดยพิจารณาจาก HFC
- ปรับค่าพารามิเตอร์ของ Phrase command และ Tone command ให้ Phrase component ที่ได้จาก Phrase command มีค่าใกล้เคียงกับ LFC และ Tone component ที่ได้จาก Tone command มีค่าใกล้เคียงกับ HFC

4.3.1.4 การแยกพารามิเตอร์โดยใช้ Syllable boundary

การแยกพารามิเตอร์โดยใช้ Syllable boundary เป็นการแยกพารามิเตอร์โดยอัตโนมัติที่พัฒนาจากการแยกพารามิเตอร์โดยไม่ใช้ Syllable boundary โดยนำ Syllable boundary เข้าไปร่วมกำหนดค่าตั้งต้นของ Tone command ขั้นตอนต่าง ๆ จะเหมือนกับการแยกพารามิเตอร์โดยไม่ใช้ Syllable boundary ยกเว้นจะมีการกำหนดให้ใน 1 syllable มี 2 Tone command เสมอ และพิจารณาจาก HFC ในแต่ละช่วงพยางค์เพื่อกำหนดค่าเริ่มต้นให้เหมาะสม

4.3.1.4 ผลการทดสอบการรู้จำโดยใช้แบบจำลองฟูจิซากิ

ผลการทดสอบโดยใช้วิธี Five fold validation กับชุดข้อมูล Thai Proverb Corpus (TPC) เป็นดังนี้

วิธีที่ใช้	ผลที่ได้
Manual	96.35%
Hansjorg Mixdorff process	56.78%
Automatic Extract without Syllable Boundary Data	70.27%
Automatic Extract with Syllable Boundary Data	68.27%

4.3.2 สรุปผลการวิจัยการรู้จำวรรณยุกต์เสียงพูดต่อเนื่องภาษาไทย

จากผลการทดสอบข้างต้นแสดงให้เห็นว่าพารามิเตอร์ของแบบจำลองฟูจิซากิสามารถนำมาใช้เป็นคุณลักษณะในการรู้จำวรรณยุกต์เสียงต่อเนื่องภาษาไทยได้จริง แต่การนำพารามิเตอร์มาใช้นั้น ยังไม่มีกรรมวิธีในการหาพารามิเตอร์ที่แน่นอน และเป็นที่ยอมรับอย่างกว้างขวาง กรรมวิธีการหาพารามิเตอร์โดยอัตโนมัติที่ทำการวิจัยสามารถใช้ในการรู้จำวรรณยุกต์ได้ในระดับหนึ่ง แต่ยังได้ผลไม่น่าพอใจถึงขั้นที่สามารถนำไปใช้งานได้จริง อีกทั้งเวลาที่ใช้ในการคำนวณในขั้นตอนการรู้จำนั้น แม้จะใช้เวลาในการประมวลผลน้อย เนื่องจากจำนวนพารามิเตอร์ที่น้อย แต่การได้มาซึ่งพารามิเตอร์นั้นจะต้องอาศัยเวลาในการประมวลผลมาก

สิ่งที่น่าสนใจคือ การพัฒนากระบวนการแยกพารามิเตอร์ให้สามารถทำได้รวดเร็ว และใช้ในการรู้จำได้ดี และเป็นที่ยอมรับทั้งกับภาษาไทย และภาษาต่างประเทศ

4.4 การรู้จำทำนองเสียงพูดสำหรับเสียงพูดภาษาไทย

นอกเหนือจากการรู้จำเสียงพูดต่อเนื่องแบบเป็นประโยค และการรู้จำวรรณยุกต์ของคำที่ได้กล่าวมาข้างต้นแล้ว โครงการวิจัยนี้ยังได้ศึกษาวิจัยการรู้จำทำนองเสียงพูด เพื่อให้คอมพิวเตอร์สามารถรู้ถึงลักษณะอื่นๆ นอกเหนือจากความหมายของผู้พูด เช่น อารมณ์ หรือความรู้สึกของผู้พูดนั่นเอง

โดยงานวิจัยนี้จะได้ศึกษาถึงวิธีออกแบบคอนทัวร์ลักษณะ (feature contour) จากคอนทัวร์ความถี่มูลฐานของเสียงพูด (fundamental frequency: F_0) เพื่อนำมาใช้รู้จำทำนองเสียงพูดภาษาไทยโดยใช้โครงข่ายประสาทเทียม

4.4.1 ทำนองเสียงพูดภาษาไทย

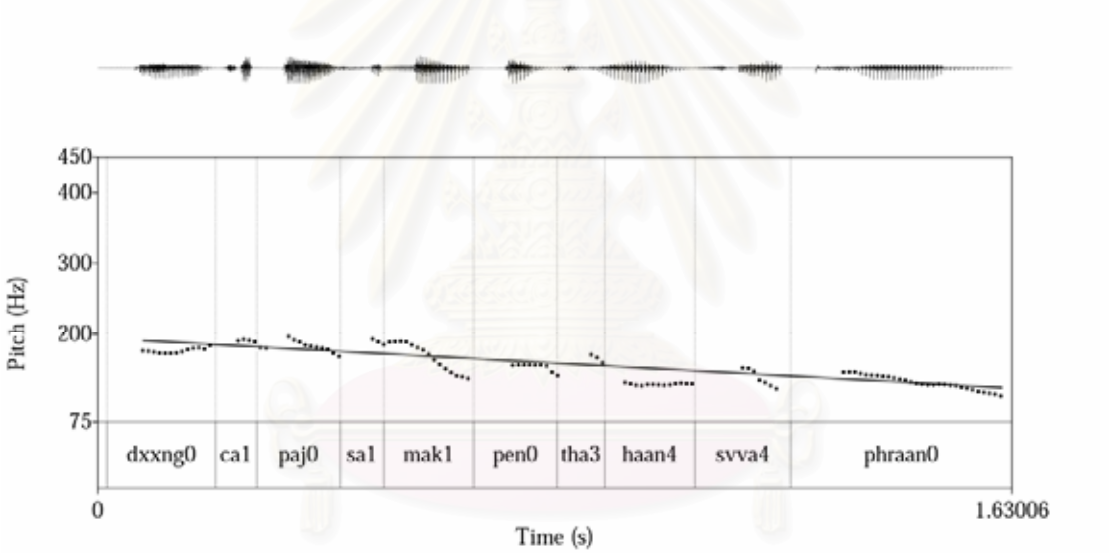
ทำนองเสียงพูดภาษาไทย แบ่งได้เป็น 3 ชนิด คือ ทำนองเสียงตก ทำนองเสียงขึ้น และทำนองเสียงผสม ลักษณะของประโยคเสียงพูดภาษาไทยแบบต่าง ๆ ซึ่งทำให้เกิดทำนองเสียงพูดทั้ง 3 ประเภท รวมทั้งผลของทำนองเสียงพูดต่อลักษณะของคอนทัวร์ F_0 มีดังนี้

1) ทำนองเสียงตก (the Fall Class) เกิดเมื่อผู้พูดพูดประโยคบอกเล่า (statement) พูดชมเชย หรือพูดเป็นคำ ๆ (citation form) พูดแบบไม่แสดงความคิดเห็น (attitudinally unmarked) พูดแบบยอมจำนน (submissive) พูดแบบซ่อนความโกรธ (concealed anger) พูดแบบเบื่อ (bored) และพูดแบบแสดงอำนาจ (authoritative) เมื่อพิจารณารูปร่างของคอนทัวร์ F_0 แบบแคบ (ในระดับพยางค์) จะพบว่ารูปร่างของคอนทัวร์ F_0 ของแต่ละพยางค์ ขึ้นกับชนิดของเสียงวรรณยุกต์ของพยางค์นั้น แต่ระดับของ F_0 ของพยางค์ที่ถัดจากพยางค์แรกจะค่อย ๆ ลดระดับลงไปเรื่อย ๆ (downdrift) จนจบประโยค ดังนั้นเมื่อพิจารณาแบบกว้าง (ในระดับประโยค) จึงเห็นได้ว่าคอนทัวร์ F_0 ของทำนองเสียงตก จะค่อย ๆ ลดระดับลง (decline) เมื่อพูดไปเรื่อย ๆ จนจบประโยค

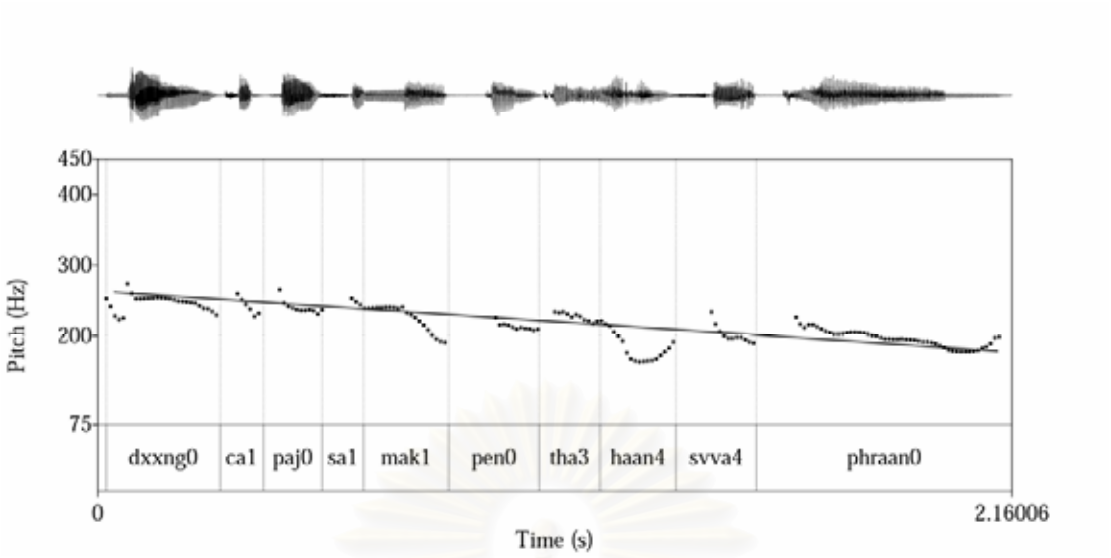
2) ทำนองเสียงขึ้น (the Rise Class) เกิดเมื่อผู้พูดพูดประโยคที่แสดงให้เห็นถึงการถาม (question) แสดงความไม่เห็นด้วย (disagreeable) แสดงความไม่เชื่อ (disbelieving) แสดงความไม่คาดฝัน (surprised) และพูดแบบแสดงให้เห็นว่ายังพูดไม่จบความ (unfinished) เมื่อพิจารณารูปร่างของคอนทัวร์ F_0 แบบแคบ จะพบว่ารูปร่างของคอนทัวร์ F_0 ของแต่ละพยางค์ขึ้นกับชนิดของเสียงวรรณยุกต์ เช่นเดียวกับทำนองเสียงตก แต่ช่วงค่า F_0 จะแคบกว่าเล็กน้อย และมีระดับของ F_0 ที่สูงกว่า จึงส่งผลให้รูปร่างของคอนทัวร์มีลักษณะสูงขึ้นกว่าทำนองเสียงตกเมื่อพิจารณาในระดับประโยค

3) ทำนองเสียงผสม (the Convolution Class) เกิดเมื่อผู้พูดพูดเพื่อต้องการเน้นหนัก (emphatic) พูดแสดงความโกรธ (anger) แสดงความเห็นด้วยอย่างมาก (very agreeable) แสดงความสนใจมาก (very interested) และแสดงความเชื่อถือมาก (very believing) ในกรณีของทำนองเสียงผสม รูปร่างของคอนทัวร์ F_0 ในแต่ละพยางค์จะขึ้นกับประเภทของเสียงวรรณยุกต์ เช่นเดียวกับทั้ง 2 ทำนองเสียง ดังที่ได้กล่าวไปแล้ว แต่ช่วงค่า F_0 ของเสียงวรรณยุกต์ทุกเสียงจะกว้างกว่า ทำนองเสียงตก และทำนองเสียงขึ้น นอกจากนี้ในกรณีของเสียงวรรณยุกต์ สามัญ เอก โท และตรี จะพบว่า F_0 มีระดับที่สูงกว่า ทำนองเสียงตก แต่เสียงวรรณยุกต์จัตวาจะมีระดับ F_0 ที่ต่ำ ซึ่งส่งผลให้รูปร่างของคอนทัวร์ F_0 ในระดับประโยค มีลักษณะสูงกว่าทำนองเสียงตก และช่วงการแกว่งกว้างกว่าทำนองเสียงตก และทำนองเสียงขึ้น

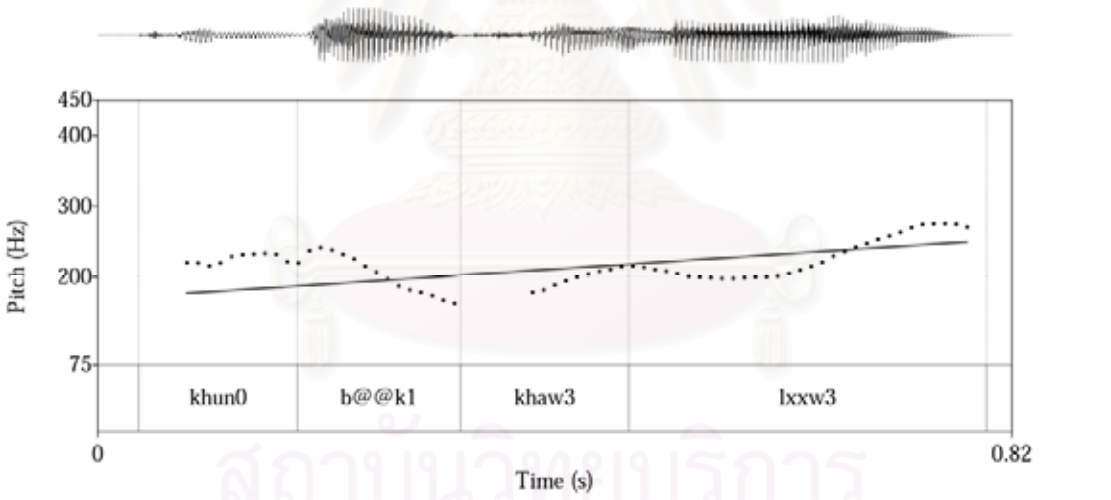
ตัวอย่างของคอนทัวร์ F_0 ของทำนองเสียงพูดทั้ง 3 แบบ ของเสียงผู้ชาย และเสียงผู้หญิง แสดงดังรูปที่ 28 ถึง 35



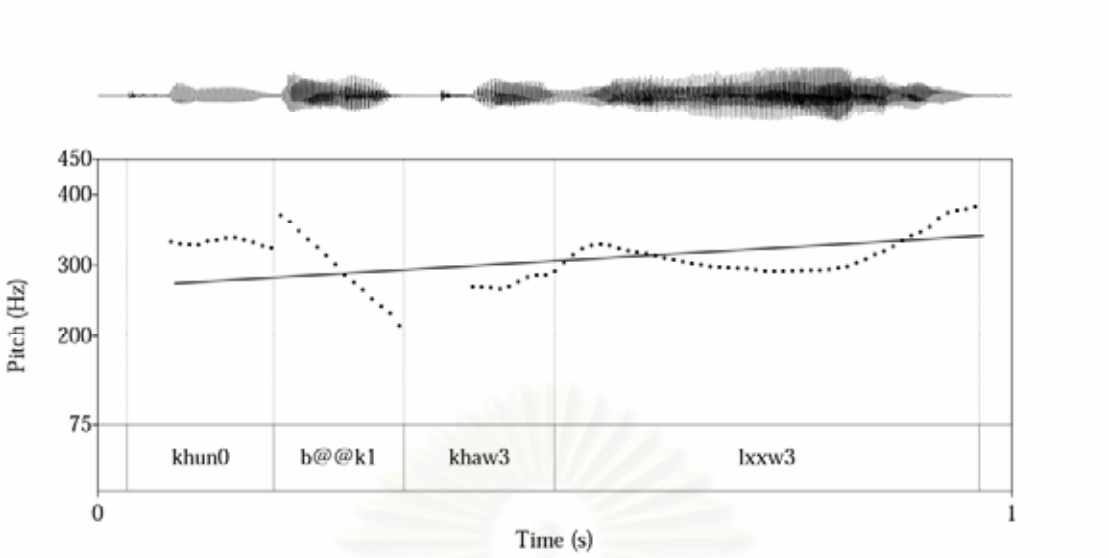
รูปที่ 28 ตัวอย่างของทำนองเสียงตก: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “แดงจะไปสมัครเป็นทหารเสือพราน” เมื่อผู้พูดเป็นผู้ชาย



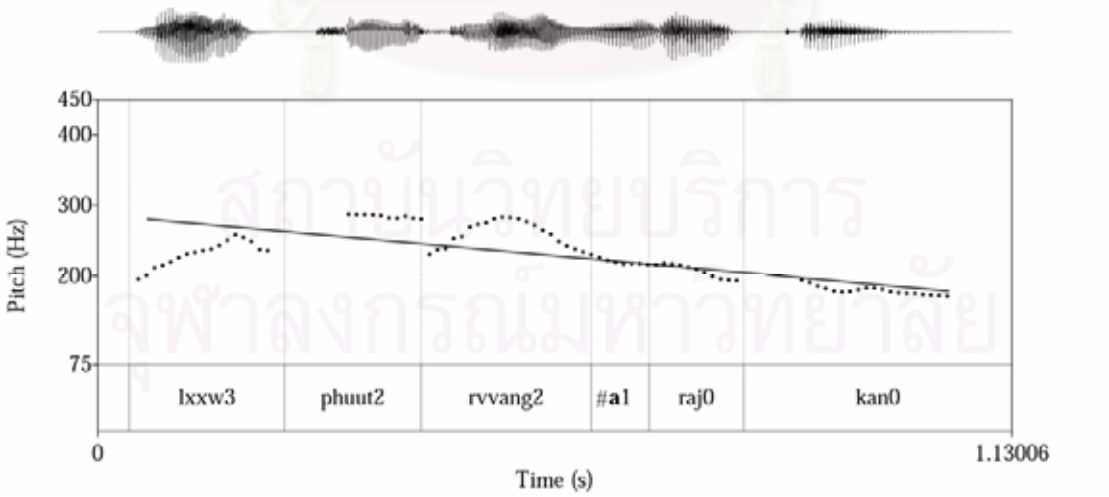
รูปที่ 29 ตัวอย่างของทำนองเสียงตก: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค
“แดงจะไปสมัครเป็นทหารเสือพราน” เมื่อผู้พูดเป็นผู้หญิง



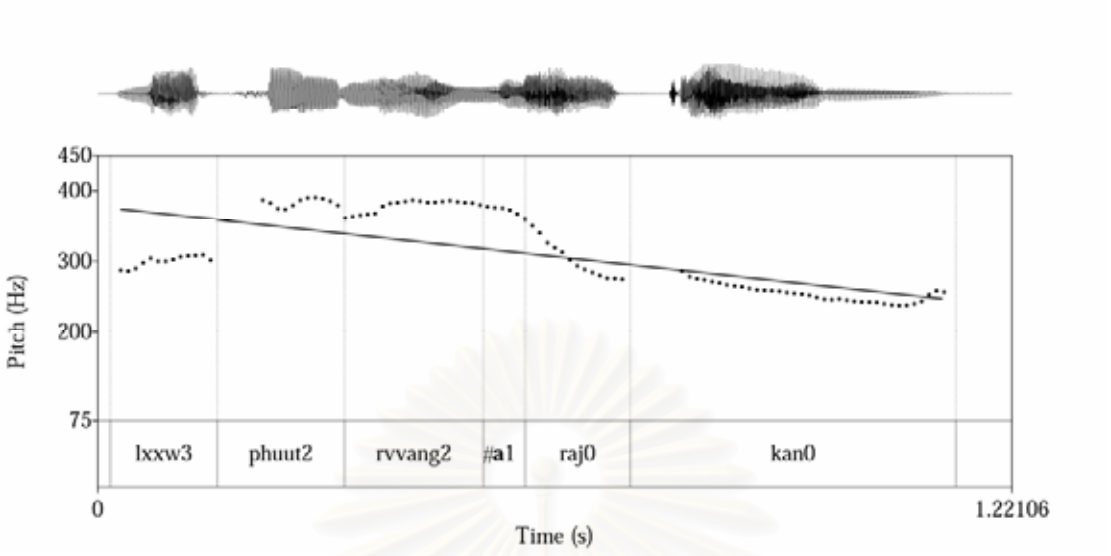
รูปที่ 30 ตัวอย่างของทำนองเสียงขึ้นแบบที่ 1: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “คุณบอกเค้าแล้ว?”
เมื่อผู้พูดเป็นผู้ชาย



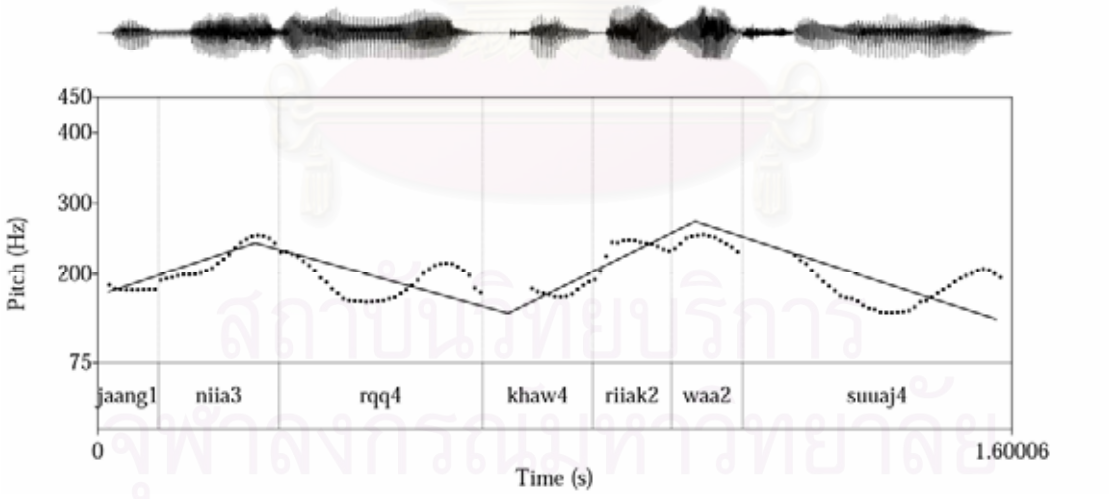
รูปที่ 31 ตัวอย่างของทำนองเสียงขึ้นแบบที่ 1: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “คุณบอกเค้าแล้ว?” เมื่อผู้พูดเป็นผู้หญิง



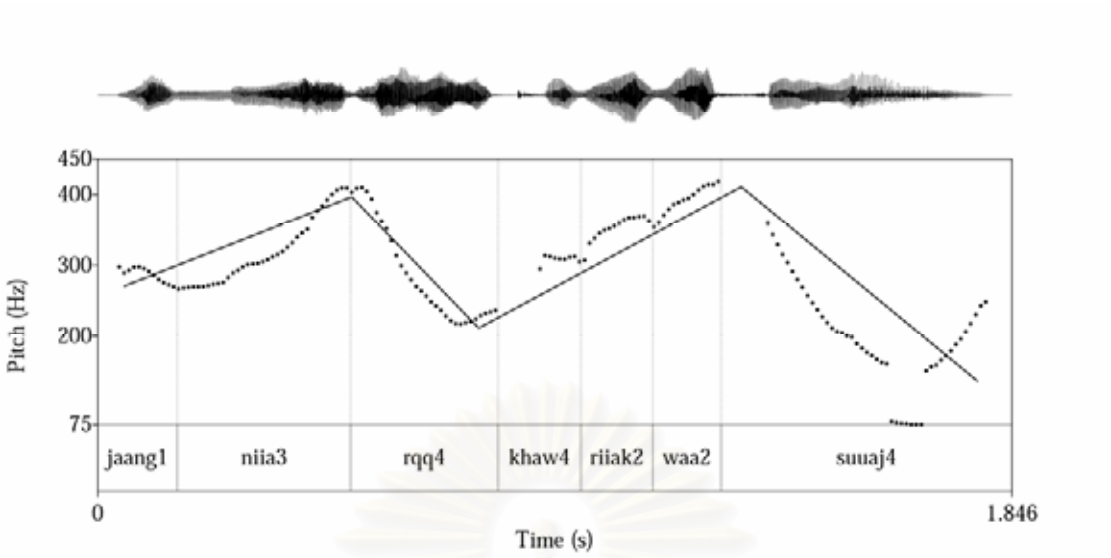
รูปที่ 32 ตัวอย่างของทำนองเสียงขึ้นแบบที่ 2: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “แล้วพูดเรื่องอะไรกัน?” เมื่อผู้พูดเป็นผู้ชาย



รูปที่ 33 ตัวอย่างของทำนองเสียงขึ้นแบบที่ 2: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “แล้วพูดเรื่องอะไรกัน?” เมื่อผู้พูดเป็นผู้หญิง



รูปที่ 34 ตัวอย่างของทำนองเสียงผสม: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “อย่างเนี้ยเหรอเขาเรียกว่าสวย!” เมื่อผู้พูดเป็นผู้ชาย



รูปที่ 35 ตัวอย่างของทำนองเสียงผสม: รูปคลื่นเสียง (รูปบน) คอนทัวร์ F_0 (เส้นประในรูปล่าง) และเส้นแสดงทำนองเสียง (เส้นทึบในรูปล่าง) ของประโยค “อย่างเนี้ยเหรอเขาเรียกว่าสวย!” เมื่อผู้พูดเป็นผู้หญิง

4.4.2 การออกแบบคอนทัวร์

คอนทัวร์ F_0 ประกอบด้วยสารสนเทศ ทางภาษาศาสตร์ (linguistic information) สารสนเทศกึ่งภาษาศาสตร์ (paralinguistic information) และสารสนเทศที่ไม่ใช่ภาษาศาสตร์ (nonlinguistic information) ในภาษาไทย สารสนเทศทางภาษาศาสตร์ เป็นสารสนเทศที่ให้ข้อมูลเกี่ยวกับชนิดของเสียงวรรณยุกต์ในภาษาไทย สารสนเทศกึ่งภาษาศาสตร์เป็นสารสนเทศที่ให้ข้อมูลเกี่ยวกับทำนองเสียงพูดภาษาไทย สารสนเทศที่ไม่ใช่ภาษาศาสตร์ เป็นสารสนเทศที่ให้ข้อมูลเกี่ยวกับลักษณะทางความสูงต่ำของเสียงอื่น ๆ เช่น เพศ และ อายุ ของผู้พูด

จะเห็นได้ว่าการจะสร้างระบบรู้จำทำนองเสียงพูด จากคอนทัวร์ F_0 นั้น มีความจำเป็นต้อง ลดผลของสารสนเทศอื่น ๆ ที่ไม่ใช่สารสนเทศกึ่งภาษาศาสตร์ ออกไปให้มากที่สุด จากการศึกษาพบว่า เมื่อพิจารณาว่าคอนทัวร์ F_0 เป็นสัญญาณดิสครีตทางเวลา (discrete time signal) จะพบว่า สารสนเทศทางภาษาศาสตร์คือ รูปร่างขององค์ประกอบความถี่สูงของสัญญาณ (high frequency component) ซึ่งเป็นองค์ประกอบที่เปลี่ยนแปลงอย่างรวดเร็ว โดยรูปร่างของการเปลี่ยนแปลงของคอนทัวร์ F_0 ในแต่ละพยางค์จะแสดงให้เห็นถึงเสียงวรรณยุกต์ประเภทต่าง ๆ ของภาษาไทย ส่วนสารสนเทศกึ่งภาษาศาสตร์คือ รูปร่างขององค์ประกอบความถี่ต่ำของสัญญาณ (low frequency component) รวมทั้งช่วงกว้างในการแกว่งขององค์ประกอบความถี่สูงของ

สัญญาณ ส่วนสารสนเทศที่ไม่ใช่ภาษาศาสตร์คือระดับความสูงของคอนทัวร์ F_0 รวมทั้งช่วงกว้างในการแกว่งของคอนทัวร์ F_0 ด้วย

งานวิจัยนี้จึงได้นำเสนอคอนทัวร์ลักษณะ (feature contour) 2 คอนทัวร์ คือ คอนทัวร์ LFC (low frequency contour) และคอนทัวร์ FVC (F_0 variation contour) ซึ่งได้มาจากการนำคอนทัวร์ F_0 ไปลดผลของสารสนเทศทางภาษาศาสตร์ออกไป เพื่อให้เหมาะสมต่อการนำมาใช้รู้จำทำนองเสียงพูด ขั้นตอนในการหาคอนทัวร์ลักษณะมีดังนี้

4.4.2.1 การหาคอนทัวร์ F_0 จากสัญญาณเสียงพูด

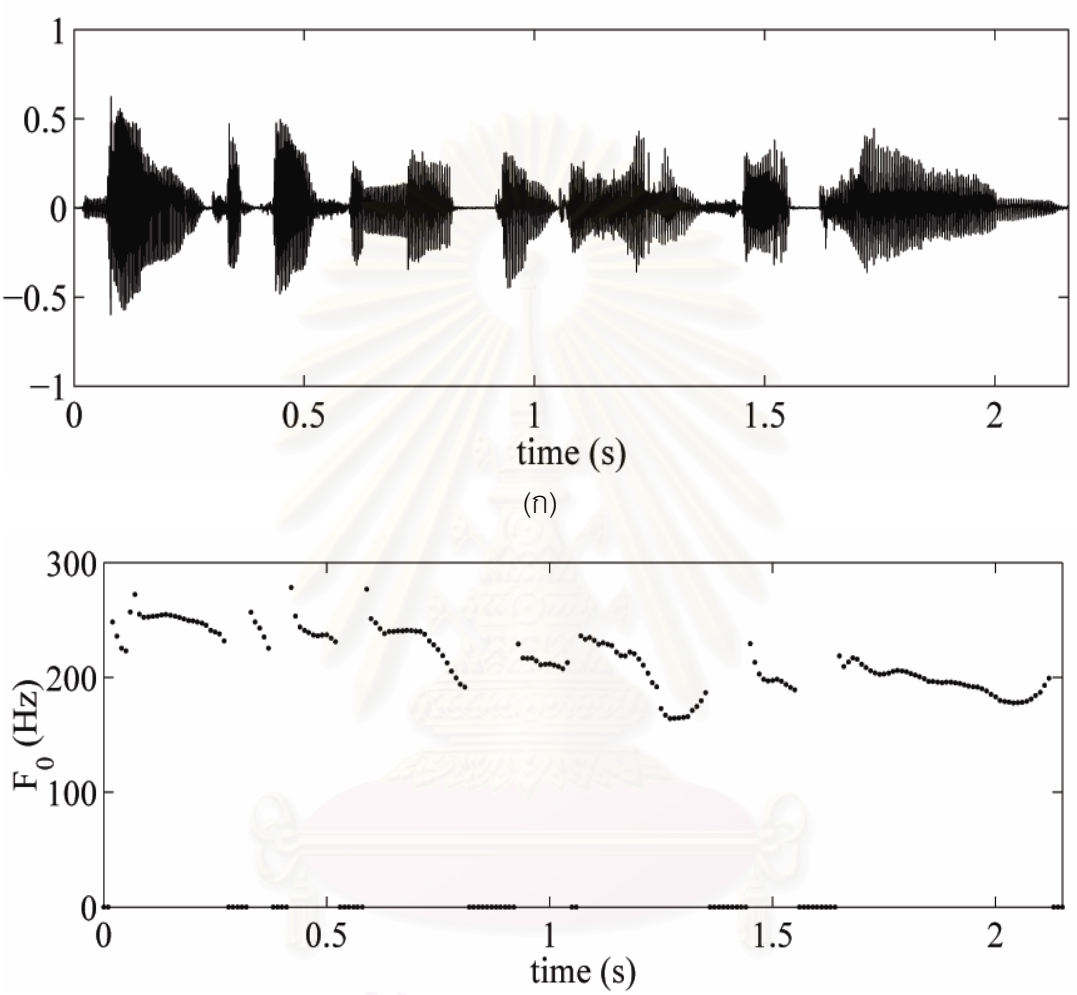
งานวิจัยนี้เลือกใช้โปรแกรม Praat (http://www.fon.hum.uva.nl/praat/) ในการหาคอนทัวร์ F_0 จากสัญญาณเสียงพูด ลักษณะของรูปคลื่นสัญญาณเสียง และคอนทัวร์ F_0 ที่หาได้โดยใช้โปรแกรม Praat แสดงดังรูปที่ 29

4.4.2.2 การทำให้คอนทัวร์ F_0 เรียบ

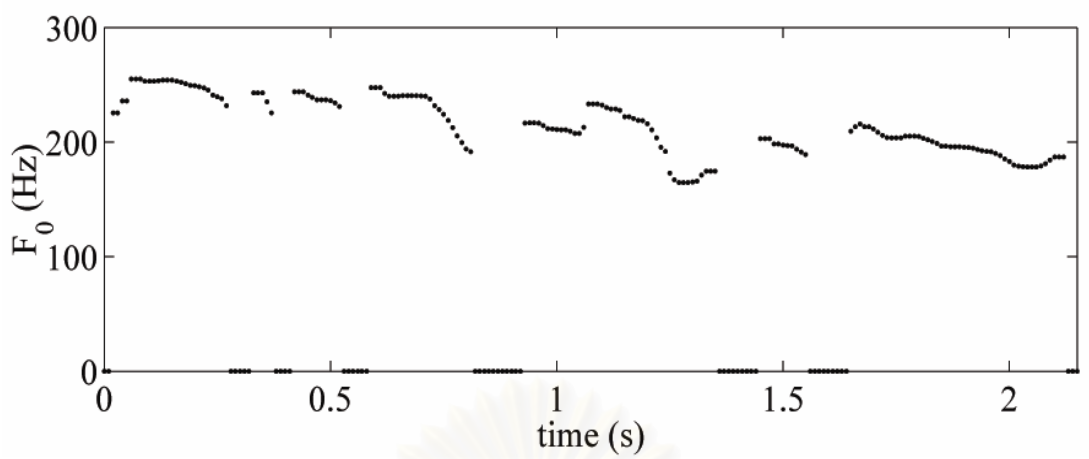
จากรูปที่ 36 จะเห็นได้ว่าคอนทัวร์ F_0 ที่หาได้ มีบางช่วงที่ไม่เรียบ ซึ่งเป็นผลมาจากกลไกในการให้กำเนิดเสียงพูด ทำให้สัญญาณพัลส์ที่ได้จากเส้นเสียงเกิดความไม่สม่ำเสมอที่เรียกว่า ปรากฏการณ์ไมโครโพรโซดิก (microprosodic effect)

วิธีที่นิยมใช้ลดความผิดพลาดดังกล่าวที่เป็นที่นิยมคือ การนำคอนทัวร์ F_0 ไปผ่านตัวกรองมัธยฐาน (median filter) จากการทดสอบกับคอนทัวร์ F_0 ที่ใช้ในงานวิจัยนี้ พบว่า การใช้ตัวกรองมัธยฐานขนาด 5 จุด (นำเอาค่า มัธยฐานของ F_0 ของF_0 ของเฟรมที่ต้องการหารวมทั้งเฟรมที่อยู่ข้างเคียง 4 เฟรม มาแทนที่ค่า F_0 ของเฟรมนั้น ๆ) สามารถกำจัดความไม่สม่ำเสมอของคอนทัวร์ F_0 ได้ดี ตัวอย่างของ คอนทัวร์ F_0 หลังจากผ่านตัวกรองมัธยฐานมีลักษณะดังรูปที่37 โดยจะเห็นได้ว่าคอนทัวร์ F_0 เรียบขึ้นเมื่อเทียบกับรูปที่ 36

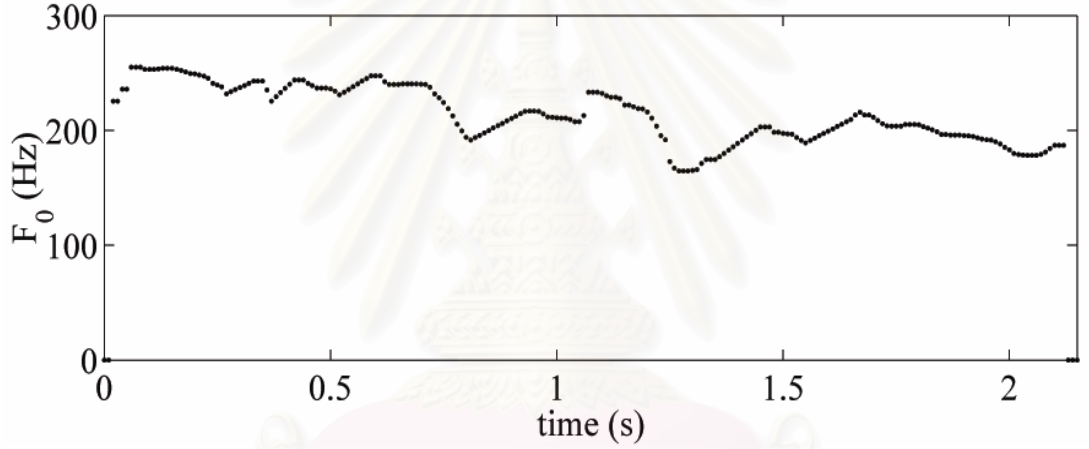
หลังจากนั้นจะประมาณค่า F_0 ในช่วงที่เป็นเสียงไม่ก้องด้วยเส้นตรง เพื่อให้คอนทัวร์ F_0 มีความต่อเนื่องกันทั้งประโยค และสามารถนำไปผ่านตัวกรองได้ โดยเรียกคอนทัวร์ที่ได้จากกระบวนการนี้ว่า CF_0 (connected F_0) ดังที่แสดงในรูปที่ 38



รูปที่ 36 สัญญาณเสียงพูดที่ต้องการนำมาหาคอนทัวร์ลักษณะ:
(ก) รูปคลื่นของสัญญาณเสียงพูด (ข) คอนทัวร์ F_0 ที่หาโดยใช้โปรแกรม Praat



รูปที่ 37 คอนทัวร์ F_0 หลังจากผ่านตัวกรองมัธยฐาน



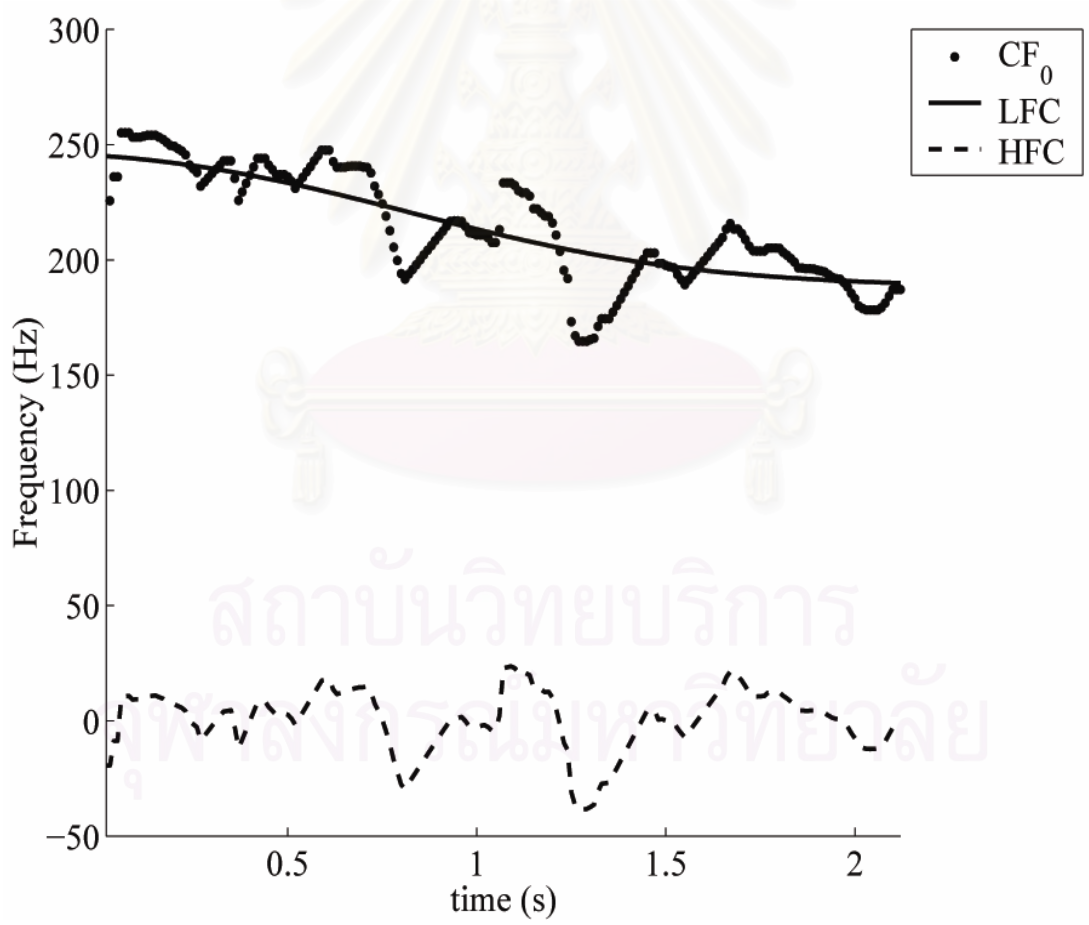
รูปที่ 38 คอนทัวร์ F_0 หลังจากผ่านการกำจัดช่วงที่เป็นเสียงไม่ก้อง (คอนทัวร์ CF_0)

4.4.2.3 การหาคอนทัวร์ LFC

สามารถหาคอนทัวร์ LFC ได้โดยการพิจารณาคอนทัวร์ CF_0 ว่าเป็นสัญญาณดิสครีตทางเวลา ในลักษณะเดียวกับ โดยพิจารณาว่าองค์ประกอบวรรณยุกต์เป็นองค์ประกอบที่มีการเปลี่ยนแปลงอย่างรวดเร็ว เมื่อเทียบกับองค์ประกอบวลี จึงสามารถกำจัดองค์ประกอบวรรณยุกต์ออกไปได้โดยการใช้ตัวกรองผ่านต่ำ (low-pass filter) ซึ่งจะยอมให้สัญญาณในส่วนที่มีการเปลี่ยนแปลงอย่างช้า ๆ เท่านั้นที่สามารถผ่านไปได้ งานวิจัยนี้ได้เลือกใช้ตัวกรองแบบ FIR (finite

impulse response) เนื่องจากจะทำให้สัญญาณขาออกมีการเลื่อนเฟสแบบเชิงเส้น จึงสามารถชดเชยผลของการเลื่อนเฟสของสัญญาณขาออกได้โดยง่าย โดยกำหนดให้สัญญาณขาเข้าที่ช่วงเวลาก่อนเริ่มคอนทัวร์ CF_0 และสัญญาณขาเข้าที่ช่วงเวลาหลังจากคอนทัวร์ CF_0 มีค่าเท่ากับค่าเฉลี่ยของคอนทัวร์ CF_0 เพื่อให้รูปร่างของสัญญาณขาออกไม่เพี้ยนที่ปลายทั้งสองข้าง โดยเรียกสัญญาณขาออก นี้ว่าคอนทัวร์ LFC ตามที่ได้กล่าวไปแล้วโดยกำหนดให้ Fc_{LFC} คือค่าความถี่ตัดของตัวกรองที่ใช้ในการหาคอนทัวร์ LFC ตัวอย่างของคอนทัวร์ LFC แสดงดังรูปที่ 39 (เส้นทึบ)

ในการทดลองหาค่า Fc_{LFC} ที่ให้อัตราการรู้จำทำนองเสียงพูดสูงที่สุด นอกจากจะใช้คอนทัวร์ LFC ที่ได้จากตัวกรองผ่านต่ำ งานวิจัยนี้ยังได้ใช้ LFC ที่เป็นเส้นตรงด้วย โดยเลือกใช้เส้นตรง 2 แบบ คือ เส้นตรงที่มีความชันเป็น 0 (เส้นแนวระดับ) และเส้นตรงที่มีความชัน โดยสามารถหาสมการเส้นตรงได้โดยใช้สมการถดถอย (regression equation) กับคอนทัวร์ CF_0



รูปที่ 39 คอนทัวร์ LFC และ HFC ที่หาได้จากคอนทัวร์ CF_0

4.4.2.4 การหาคอนทัวร์ FVC

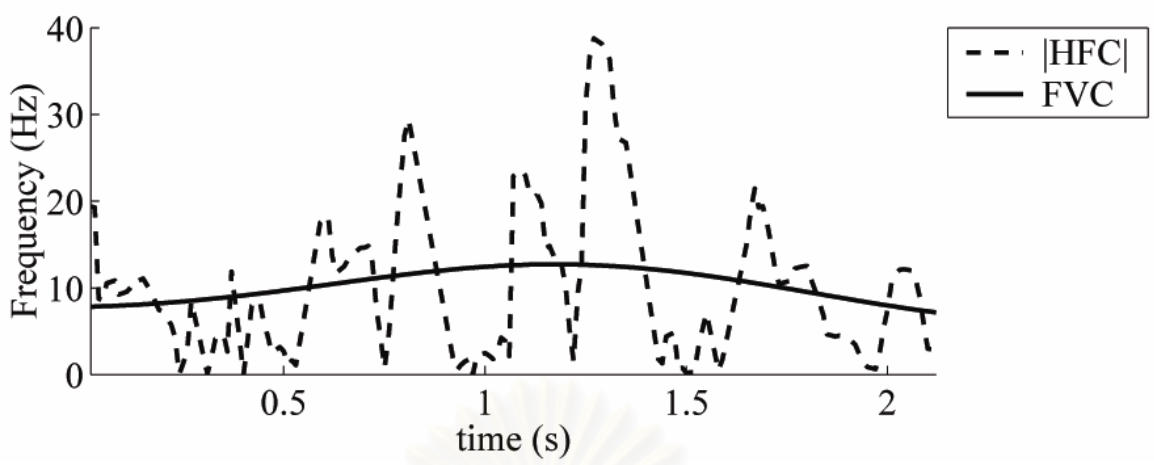
จากที่กล่าวไปแล้วข้างต้นว่า องค์ประกอบเสียงวรรณยุกต์มีลักษณะที่แสดงให้เห็นถึงประเภทของทำนองเสียงได้ นั่นก็คือช่วงกว้างของการเปลี่ยนแปลงค่าของคอนทัวร์ F_0 เราจึงไม่สามารถทิ้งองค์ประกอบเสียงวรรณยุกต์ไปได้

องค์ประกอบเสียงวรรณยุกต์เป็นองค์ประกอบที่มีการเปลี่ยนแปลงอย่างรวดเร็ว เราจึงสามารถสกัดเอาองค์ประกอบเสียงวรรณยุกต์จากคอนทัวร์ CF_0 ได้โดยใช้ตัวกรองผ่านสูง (high-pass filter) แต่เนื่องจากเรามีองค์ประกอบความถี่ต่ำ คือคอนทัวร์ LFC อยู่แล้ว เราจึงสามารถหาองค์ประกอบความถี่สูงได้โดยการนำคอนทัวร์ LFC ไปลบออกจากคอนทัวร์ CF_0 โดยเรียกคอนทัวร์ที่ได้ว่า คอนทัวร์ HFC (high frequency contour) แสดงดังรูปที่ 39 (เส้นประ)

สิ่งที่แสดงให้เห็นถึงลักษณะของทำนองเสียงในคอนทัวร์ HFC คือ ช่วงกว้างในการแกว่งของคอนทัวร์ HFC โดยไม่ต้องคำนึงถึงว่า เป็นการแกว่งขึ้น (HFC > 0) หรือการแกว่งลง (HFC < 0) จึงนำ HFC ไปใส่ค่าสัมบูรณ์ ลักษณะของ |HFC| แสดงดังรูปที่ 39

จากรูปที่ 39 จะเห็นว่า คอนทัวร์ |HFC| นั้นไม่เรียบ ซึ่งเป็นผลจากเสียงวรรณยุกต์ของแต่ละพยางค์ ที่ให้สารสนเทศภาษาศาสตร์ งานวิจัยนี้จึงได้นำ |HFC| ไปผ่านตัวกรองผ่านต่ำแบบ FIR แบบเดียวกับที่ใช้ในการหาคอนทัวร์ LFC เพื่อกำจัดสารสนเทศทางภาษาศาสตร์ โดยจะได้ว่าสัญญาณขาออกของตัวกรอง แสดงให้เห็นถึงความมากน้อยในการแกว่งของ HFC ซึ่งก็คือความมากน้อยในการกวัดแกว่ง หรือการเปลี่ยนแปลงค่าของคอนทัวร์ F_0 นั่นเอง จึงเรียกสัญญาณขาออกนี้ว่า คอนทัวร์ FVC (F_0 variation contour) ดังแสดงในรูปที่ 40 โดยกำหนดให้ความถี่ตัดของตัวกรองผ่านต่ำ ที่ใช้หาคอนทัวร์ FVC มีค่าเป็น Fc_{LFC}

คอนทัวร์ FVC เป็นคอนทัวร์ที่แสดงให้เห็นถึงความมากน้อยในการเปลี่ยนแปลงค่าของ F_0 ถ้าคอนทัวร์ FVC มีระดับที่สูง แสดงว่าคอนทัวร์ F_0 มีช่วงการแกว่งที่กว้าง ซึ่งจะพบในทำนองเสียงทำนองเสียงพูดแบบผสม แต่ถ้าคอนทัวร์ FVC มีระดับที่ต่ำ แสดงว่าคอนทัวร์ F_0 มีช่วงการแกว่งที่แคบ ซึ่งมักจะพบได้ในทำนองเสียงพูดแบบตก หรือทำนองเสียงพูดแบบขึ้น



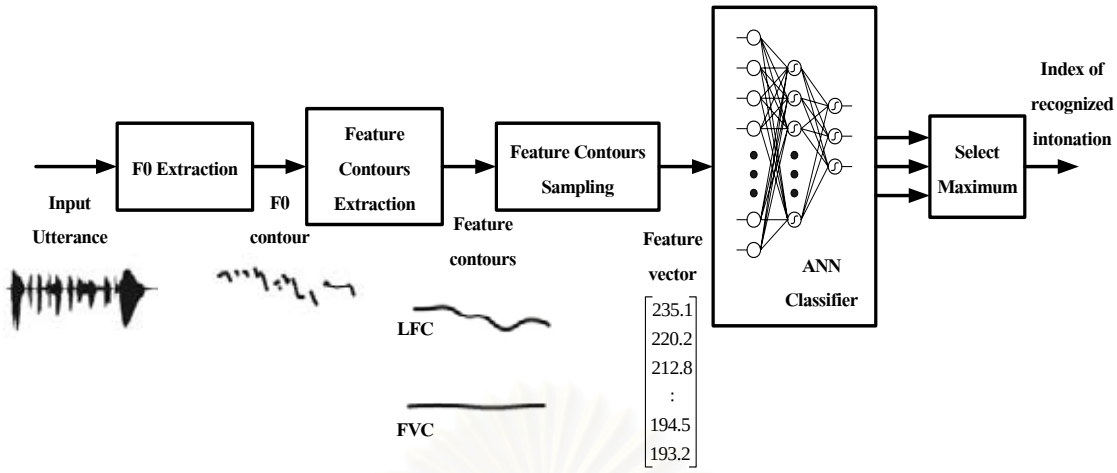
รูปที่ 40 คอนทัวร์ FVC ที่หาได้จากการนำ |HFC| ไปผ่านตัวกรองผ่านต่ำ

4.4.3 ระบบรู้จำทำนองเสียงพูด

ระบบรู้จำทำนองเสียงพูดภาษาไทยที่ออกแบบขึ้นเพื่อใช้ในการทดลองแสดงในรูปที่ 33 ระบบจะนำเสียงพูดที่ต้องการรู้จำทำนองเสียงไปหาค่า F_0 ในส่วน F_0 Extraction แล้วจึงนำคอนทัวร์ F_0 ที่ได้ไปหาคอนทัวร์ลักษณะ LFC และ FVC ในส่วน Feature Contours Extraction ตามวิธีที่แสดงในหัวข้อที่ผ่านมา

จากนั้นจึงสุ่มตัวอย่าง (sampling) คอนทัวร์ลักษณะทั้งสอง โดยกำหนดให้จำนวนจุดที่ใช้สุ่มตัวอย่างขึ้นกับค่าความถี่ตัดของตัวกรองที่ใช้ในการหาคอนทัวร์ LFC และ FVC ตามทฤษฎีการสุ่มตัวอย่างของไนควิสต์ (Nyquist's sampling theory) คือสุ่มด้วยอัตรา 2 เท่าของความกว้างของช่วงความถี่ (bandwidth)

ค่าที่นำมาใช้เป็นเวกเตอร์ลักษณะ ได้จากความยาวของประโยค ค่าของ LFC และ FVC รวมทั้งค่าผลต่างอันดับหนึ่ง (first difference) ของคอนทัวร์ทั้งสอง (ΔLFC และ ΔFVC) ที่จุดที่สุ่มตัวอย่าง



รูปที่ 41 แผนภาพของระบบรู้จำทำนองเสียงพูด

4.4.4 ข้อมูลเสียงพูดที่ใช้ในการวิจัย

ประโยคที่ใช้ทดสอบ มีทั้งสิ้น 61 ประโยค มาจากบทสนทนา 6 บท แต่ละบทมีผู้พูด 2 คนพูดคุยกัน โดยในแต่ละประโยคพูดได้เขียนกำกับไว้ว่าจะต้องพูดด้วยทำนองเสียงพูดแบบใด ความยาวของแต่ละประโยคจะอยู่ระหว่าง 1 – 11 พยางค์

ข้อมูลเสียงที่ใช้มาจากผู้พูด 12 คน ผู้ชาย 6 คน ผู้หญิง 6 คน ผู้พูดจะจับคู่กันและสนทนาตามบทพูด โดยผู้พูดทุกคนจะพูดแต่ละบทสนทนา 2 ครั้ง โดยสลับบทพูดกัน เพื่อให้ผู้พูดทุกคนได้พูดครบทั้ง 61 ประโยค จึงมีประโยคสำหรับทดสอบทั้งสิ้น $12 \times 61 = 732$ ประโยค

เนื่องจากผู้พูดแต่ละคนได้รับคำสั่งว่าให้พูดตามบทสนทนาให้เป็นธรรมชาติมากที่สุด จึงทำให้มีบางประโยค ที่ผู้พูดพูดด้วยทำนองเสียงที่ไม่ตรงกับทำนองเสียงที่ได้กำหนดไว้ให้ จึงได้กำหนดให้มีการนำข้อมูลเสียงทั้งหมดมาตรวจสอบประเภทของทำนองเสียงอีกครั้ง โดยการเขียนโปรแกรมเพื่อนำเสียงทั้ง 732 ประโยค มาสลับลำดับแบบสุ่ม แล้วเปิดให้ผู้ฟังทั้ง 2 คนฟังแล้วให้ผู้ฟังตัดสินใจว่าเสียงที่ได้ยิน เป็นทำนองเสียงพูดแบบใด โดยให้ผู้ฟังแต่ละคนตรวจสอบเสียงพูดทุกเสียงคนละ 2 รอบ ดังนั้นจะได้ว่ามีการทดสอบการฟัง 4 ครั้งสำหรับประโยคเสียงพูดแต่ละประโยค งานวิจัยนี้จะนำเฉพาะประโยคที่มีทำนองเสียงที่ชัดเจนเท่านั้น มาทำการทดลอง โดยกำหนดว่าประโยคที่มีทำนองเสียงที่ชัดเจน คือประโยคที่ผู้ฟังเลือกให้มีทำนองเสียงประเภทเดียวกัน 3 ครั้งขึ้นไป จากการฟังทั้งหมด 4 ครั้ง

4.4.5 การทดลองรู้จำทำนองเสียง

การทดลองในงานวิจัยนี้จะแบ่งเป็น 2 ส่วนใหญ่ ๆ คือ การทดลองในชุดแรก จะแบ่งทำนองเสียงออกเป็น 3 ประเภท คือ ทำนองเสียงตก ทำนองเสียงขึ้น และทำนองเสียงผสม ส่วนการทดลองในชุดที่สอง จะแบ่งทำนองเสียงออกเป็น 2 ประเภท คือ ทำนองเสียงตก และทำนองเสียงขึ้น โดยจะจัดกลุ่มทำนองเสียงผสม เข้าไปอยู่ในประเภทเดียวกับทำนองเสียงขึ้น

การทดลองในแต่ละชุด จะเริ่มจากการใช้ความยาวของประโยค และ LFC จากนั้นจึงเพิ่ม ΔLFC และ FVC เข้าไป เพื่อทดสอบผลของลักษณะต่าง ๆ ต่ออัตราการรู้จำ ในแต่ละการทดลองจะเปลี่ยนค่าความถี่ตัดของตัวกรองที่ใช้สร้างคอนทัวร์ LFC (Fc_{LFC}) และค่าความถี่ตัดของตัวกรองที่ใช้สร้างคอนทัวร์ FVC (Fc_{FVC}) โดยกำหนดให้ค่าทั้งสองมีค่าเป็น 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0 และ 4.5 Hz โดยจับคู่ Fc_{LFC} และ Fc_{FVC} ให้ครบทุกแบบเท่าที่เป็นไปได้ นอกจากนี้ยังได้ใช้ LFC และ FVC ที่เป็นเส้นตรง และเส้นตรงแนวระดับ (มีความชันเป็น 0) ซึ่งได้จากการหาสมการถดถอย (regression equation) ของคอนทัวร์ CF_0 และ คอนทัวร์ |HFC| อีกด้วย

การทดลองแต่ละประเภท จะเป็นลักษณะที่ขึ้นกับผู้พูด (speaker dependent) นั่นคือ ผู้พูดที่อยู่ในกลุ่มฝึกฝน และผู้พูดที่อยู่ในกลุ่มทดสอบเป็นกลุ่มเดียวกัน โดยในแต่ละการทดลองจะทดลองเสียงของผู้ชาย และเสียงของผู้หญิงแยกจากกัน เนื่องจากคอนทัวร์ F_0 ของเสียงผู้ชาย และคอนทัวร์ F_0 ของเสียงผู้หญิงมีความแตกต่างกันมาก

4.4.6 ผลการทดลอง

4.4.6.1 กรณีที่แบ่งทำนองเสียงออกเป็น 3 ประเภท

ในกรณีของเสียงผู้ชาย การทดลองที่ให้อัตราการรู้จำที่ดีที่สุด (อัตราการรู้จำเฉลี่ยมีค่าสูงที่สุด และอัตราการรู้จำของทำนองเสียงที่มีอัตราการรู้จำต่ำที่สุดมีค่าสูงที่สุด) เกิดขึ้นเมื่อใช้ความยาวของประโยค และจุดสุ่มตัวอย่างของ LFC และ FVC เป็นเวกเตอร์ลักษณะ ซึ่งให้อัตราการรู้จำเฉลี่ยที่ร้อยละ 61.6 ผลการรู้จำแสดงในตารางที่ 14

ตารางที่ 14 ผลการรู้จำทำนองเสียง (%) ในกรณีที่ให้อัตราการรู้จำสูงที่สุดของเสียงผู้ชาย เมื่อ Fc_{LFC} = 3.5 Hz และ Fc_{FVC} = 3.0 Hz

ประเภทของทำนองเสียงที่นำมารู้จำ	ประเภทของทำนองเสียงที่รู้จำได้			จำนวนประโยค
	ตก	ขึ้น	ผสม	
ตก	84.7	10.2	5.1	98
ขึ้น	29.7	47.3	23.1	91
ผสม	25.3	25.3	49.4	79
	จำนวนประโยคทั้งหมด			268

ในกรณีของเสียงผู้หญิง การทดลองที่ให้อัตราการรู้จำที่ดีที่สุด เกิดขึ้นเมื่อใช้ความยาวของประโยค และจุดสุ่มตัวอย่างของ LFC ΔLFC และ FVC เป็นเวกเตอร์ลักษณะ ซึ่งให้อัตราการรู้จำเฉลี่ยอยู่ที่ร้อยละ 73.7 ผลการรู้จำแสดงในตารางที่ 15

ตารางที่ 15 ผลการรู้จำทำนองเสียง (%) ในกรณีที่ให้อัตราการรู้จำสูงที่สุด ของเสียงผู้หญิง เมื่อ Fc_{LFC} = 3.0 Hz และ Fc_{FVC} = 1.5 Hz

ประเภทของทำนองเสียงที่นำมารู้จำ	ประเภทของทำนองเสียงที่รู้จำได้			จำนวนประโยค
	ตก	ขึ้น	ผสม	
ตก	90.6	0.9	8.5	106
ขึ้น	16.9	43.1	40.0	65
ผสม	11.5	13.1	75.4	122
	จำนวนประโยคทั้งหมด			293

4.4.6.2 กรณีที่แบ่งทำนองเสียงออกเป็น 2 ประเภท

ในกรณีของเสียงผู้ชาย การทดลองที่ให้อัตราการรู้จำที่ดีที่สุด เกิดขึ้นเมื่อใช้ความยาวของประโยค และจุดสุ่มตัวอย่างของ LFC และ FVC เป็นเวกเตอร์ลักษณะ ซึ่งให้อัตราการรู้จำเฉลี่ยที่ร้อยละ 81.7 ผลการรู้จำแสดงในตารางที่ 16

ตารางที่ 16 ผลการรู้จำทำนองเสียง (%) ในกรณีที่ให้อัตราการรู้จำสูงที่สุด ของเสียงผู้ชาย เมื่อ Fc_{LFC} =1.5 Hz และใช้ FVC เป็นเส้นตรงที่มีความชันเป็น 0

ประเภทของทำนองเสียงที่นำมารู้จำ	ประเภทของทำนองเสียงที่รู้จำได้		จำนวนประโยค
	ตก	ขึ้น	
ตก	77.6	22.4	98
ขึ้น	15.9	84.1	170
จำนวนประโยคทั้งหมด			268

ในกรณีของเสียงผู้หญิง การทดลองที่ให้อัตราการรู้จำที่ดีที่สุด เกิดขึ้นเมื่อใช้ความยาวของประโยค และจุดสุ่มตัวอย่างของ LFC ΔLFC และ FVC เป็นเวกเตอร์ลักษณะ ซึ่งให้อัตราการรู้จำเฉลี่ยที่ร้อยละ 90.8 ผลการรู้จำแสดงในตารางที่ 17

ตารางที่ 17 ผลการรู้จำทำนองเสียง (%) ในกรณีที่ให้อัตราการรู้จำสูงที่สุด ของเสียงผู้หญิง เมื่อ Fc_{LFC} = 3.0 Hz และใช้ Fc_{FVC} = 1.0 Hz

ประเภทของทำนองเสียงที่นำมารู้จำ	ประเภทของทำนองเสียงที่รู้จำได้		จำนวนประโยค
	ตก	ขึ้น	
ตก	90.6	9.4	106
ขึ้น	9.1	90.9	187
จำนวนประโยคทั้งหมด			293

4.4.7 วิเคราะห์ผลการทดลอง

จากผลการทดลอง จะเห็นได้ว่า ทั้ง คอนทัวร์ LFC และคอนทัวร์ FVC ต่างก็ช่วยให้อัตราการรู้จำทำนองเสียงสูงขึ้นกว่าการใช้เพียงคอนทัวร์ F_0 (งานวิจัยนี้ไม่ได้ใช้คอนทัวร์ F_0 โดยตรง แต่สามารถเปรียบเทียบได้ว่าการสุ่มตัวอย่างคอนทัวร์ LFC ที่มีค่า Fc_{LFC} สูง ๆ มีค่าใกล้เคียงกับการสุ่มตัวอย่างคอนทัวร์ F_0) ทั้งในการทดลองที่แบ่งทำนองเสียงเป็น 3 ประเภท และในการทดลองที่แบ่งทำนองเสียงเป็น 2 ประเภท

ในการทดลองที่แบ่งทำนองเสียงเป็น 3 ประเภท จะเห็นได้ว่าตัวรู้จำมีความสับสนระหว่างทำนองเสียงขึ้น และทำนองเสียงผสมอยู่มาก เป็นผลทำให้อัตราการรู้จำเฉลี่ยมีค่าเพียงร้อยละ 61.6 สำหรับเสียงผู้ชาย และร้อยละ 73.7 สำหรับเสียงผู้หญิง

ในการทดลองที่แบ่งทำนองเสียงออกเป็น 2 ประเภท โดยกำหนดให้ทำนองเสียงผสมเป็นประเภทเดียวกับทำนองเสียงขึ้น พบว่าอัตราการรู้จำเฉลี่ยเพิ่มสูงขึ้นมาก โดยมีค่าเป็นร้อยละ 81.7 สำหรับเสียงผู้ชาย และร้อยละ 90.8 สำหรับเสียงผู้หญิง

4.4.8 สรุปผลการรู้จำทำนองเสียงพูดสำหรับเสียงพูดภาษาไทย

งานวิจัยนี้ได้นำเสนอคอนทัวร์ LFC และคอนทัวร์ FVC ขึ้นเพื่อนำมาใช้เป็นคอนทัวร์ลักษณะสำหรับระบบรู้จำทำนองเสียงพูดภาษาไทย ซึ่งจากการทดลองสามารถสรุปได้ว่าคอนทัวร์ทั้งสองสามารถทำให้อัตราการรู้จำทำนองเสียงพูดสูงขึ้นกว่าการใช้คอนทัวร์ F_0

ในการทดลองที่แบ่งทำนองเสียงเป็น 3 ประเภท คือทำนองเสียงตก ทำนองเสียงขึ้น และทำนองเสียงผสม จะเห็นได้ว่าตัวรู้จำมีความสับสนระหว่างทำนองเสียงขึ้น และทำนองเสียงผสมอยู่มาก ซึ่งจากการวิเคราะห์คอนทัวร์ LFC และคอนทัวร์ FVC โดยเฉลี่ย พบว่าคอนทัวร์ทั้งสองของสองทำนองเสียงนี้มีความใกล้เคียงกันมาก และจำเป็นต้องมีการศึกษาโดยละเอียดเพื่อหาลักษณะอื่น ๆ ของเสียงพูดมาช่วยในการรู้จำ เช่นอัตราเร็วในการพูด หรือระดับพลังงานของเสียงพูด โดยสามารถนำระบบรู้จำทำนองเสียงที่นำเสนอในงานวิจัย ไปใช้จำแนกทำนองเสียงตกออกจากทำนองเสียงขึ้น และทำนองเสียงผสมก่อน แล้วจึงนำลักษณะอื่น ๆ ไปใช้จำแนกทำนองเสียง 2 ประเภทนี้ ซึ่งในการทดลองที่แบ่งทำนองเสียงออกเป็น 2 ประเภท (กำหนดให้ทำนองเสียงผสมเป็นทำนองเสียงประเภทเดียวกับทำนองเสียงขึ้น) ได้แสดงให้เห็นว่าระบบรู้จำสามารถจำแนกความแตกต่างระหว่างทำนองเสียงตกออกจากทำนองเสียงประเภทอื่น ๆ ได้ดี (ร้อยละ 81.7 สำหรับเสียงผู้ชาย และร้อยละ 90.8 สำหรับเสียงผู้หญิง)

5. ผลงานทางวิชาการที่ได้จากโครงการวิจัยนี้

5.1 วิทยานิพนธ์จำนวน 3 เรื่องได้แก่

5.1.1 ระดับปริญญาดุษฎีบัณฑิต

- นายเอกฤทธิ์ มณีน้อย, "การศึกษาหน่วยตามของพยางค์เชิงกลสัทศาสตร์ : พื้นฐานสำหรับระบบการรู้จำเสียงพูดต่อเนื่องภาษาไทย"

5.1.2 ระดับปริญญามหาบัณฑิต

- นายนัทธี งามเจตนารมย์, "การรู้จำวรรณยุกต์ในคำพูดต่อเนื่องภาษาไทยบนพื้นฐานแบบจำลอง ฟูจิซากิ"
- นายปฐวี ชาญไววิทย์, "ระบบรู้จำทำนองเสียงพูดสำหรับเสียงพูดภาษาไทยโดยใช้โครงข่ายประสาทเทียม"

5.2 บทความทางวิชาการ

1. Maneenoi E., Ahkuputra V, Luksaneeyanawin S., and Jitapunkul S., "Acoustic Modeling of Onset-Rhyme for Thai Continuous Speech Recognition", Proceedings of the 9th Australian International Conference on Speech Science & Technology, Melbourne, Australia, 2003.

2. Maneenoi E., Ahkuputra V., Luksaneeyanawin S., and Jitapunkul S., "A Study on Acoustic Modeling for Speech Recognition of Predominantly Monosyllabic Languages", IEICE Transactions on Information and System, Vol.E87-D, No. 5, May 2004.

3. Maneenoi E., Ahkuputra V, Luksaneeyanawin S., and Jitapunkul S., "Modeling of Onset-Rhyme for Speech Recognition of Thai Language" Proceedings of the 8th

European Conference on Speech Communication and Technology (Eurospeech 2003), Geneva, Switzerland, 2003.

4. Jitapunkul S., Maneenoi E., Ahkuputra V, and Luksaneeyanawin S. "Performance Evaluation of Phonotactic and Contextual Onset-Rhyme Models for Speech Recognition of Thai Language", Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech 2003), Geneva, Switzerland, pp. 1841-1844, 2003.

5. Charnvivit P., Thubthong N., Maneenoi E., Luksaneeyanawin S., and Jitapunkul S., "Recognition of Intonation Patterns in Thai Utterance", Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech 2003), Geneva, Switzerland, pp. 137-140, 2003.

6. Ngarmchatetanarom N., Maneenoi E., Asdornwised W., and Jitapunkul S., "Tone Recognition of Thai Continuous Speech Using Fujisaki's Model", Proceedings of the 17th IEEE Canadian Conference on Electrical and Computer Engineering, Niagara Falls, Canada, May, 2-5, 2004.

5.3 การร่วมมือกับภาคเอกชน

โครงการ "Thai Text to Speech and Thai Automatic Word Recognition for SUN IVR Systems" ร่วมกับบริษัท SUN Systems Corporation Limited โดยผ่านทางศูนย์วิจัยประมวลผลภาษาและวัจนะ คณะอักษรศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย