การรู้จำลายมือเขียนภาษาไทยแบบออนไลน์ โดยใช้ดอนเท็กฟรีแกรมม่าจากการเรียนรู้เพิ่มเติม

ON-LINE THAI HAND WRITTEN RECOGNITION USING INCREMENTAL LEARNING OF CONTEXT-FREE GRAMMARS

ปองเกษม พลสันติกุล

PONGKASEM POLSUNTIKUL

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2548

ISBN 974-15-1261-5

การรู้จำลายมือเขียนภาษาไทยแบบออนไลน์ โดยใช้คอนเท็กฟรีแกรมม่าจากการเรียนรู้เพิ่มเติม

ON-LINE THAI HAND WRITTEN RECOGNITION USING INCREMENTAL LEARNING OF CONTEXT-FREE GRAMMARS

ปองเกษม พลสันติกุล
PONGKASEM POLSUNTIKUL

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2548

ISBN 974-15-1361-5

ON-LINE THAI HAND WRITTEN RECOGNITION USING INCREMENTAL LEARNING OF CONTEXT-FREE GRAMMARS

PONGKASEM POLSUNTIKUL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTER ENGINEERING
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2005

ISBN 974-15-1361-5

หัวข้อวิทยานิพนธ์	การรู้จำลายมือเขียนภาษาไทยแบบออนไลน์ โดยใช้คอนเท็กฟรีแกรมม่าจากการเรียนรู้เพิ่มเติม
นักศึกษา	นางสาว ปองเกษม พลสันติกุล
รหัสนักศึกษา	45061031
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2548
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร. บุญธีร์ เครือตราชู

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอการสร้างไวยากรณ์ สำหรับใช้เป็นโมเดลของตัวอักษรภาษาไทยที่เป็นลายมือเขียน โดยมีเป้าหมายหลักของงานวิจัยคือสร้างไวยากรณ์แบบอัตโนมัติสำหรับตัวอักษรที่เป็นลายมือเขียนเพื่อให้ได้ไวยากรณ์ที่เป็นของตัวอักษรนั้นๆ การสร้างไวยากรณ์นี้จะใช้กลุ่มตัวอย่างรูปแบบของตัวอักษรซึ่งแสดงทิศทางการเขียนโดยใช้เชนโค้ดแบบ 8 ทิศทาง ไวยากรณ์ที่สร้างต้องครอบคลุมตัวอักษรอื่น นอกเหนือจากกลุ่มตัวอักษรที่ใช้สร้างไวยากรณ์นั้นด้วย ไวยากรณ์แรกเริ่มจะถูกสร้างมาจากกลุ่มตัวอย่างของเชนโค้ดของตัวอักษรนั้นๆ จากนั้นจะทำการปรับแต่งไวยากรณ์เริ่มต้นนี้ให้มีความเหมาะสม เมื่อได้ไวยากรณ์ที่เหมาะสมแล้วก็จะสามารถนำไปใช้ในการรู้จำตัวอักษรโดยใช้หลักการกระจายไวยากรณ์ (parse) ได้เหมือนกับการใช้ในงานวิจัยเกี่ยวกับภาษาธรรมชาติ สิ่งที่ต้องคำนึงถึงสำหรับแกรมม่าที่สร้างคือแกรมม่าจะต้องสามารถยอมรับตัวอักษรตัวนั้นที่เขียนในรูปแบบต่างๆได้ แต่ก็ต้องมีความเฉพาะสำหรับตัวอักษรตัวนั้นเท่านั้นด้วย วิธีการที่จะทำให้ไวยากรณ์มีลักษณะดังกล่าวนี้จะต้องใช้แม่แบบมาช่วยซึ่งได้แก่ แม่แบบชนิดลูป แม่แบบชนิดคู่ลูป และแม่แบบชนิดกับดักลูป การสร้างไวยากรณ์ที่ได้จะนำมาเปรียบเทียบผลการทดลองกับโมเดลแบบ HMM ผลการทดลองที่ได้จากการรู้จำโดยใช้ไวยากรณ์ที่สร้างตามหลักการในวิทยานิพนธ์นี้มีเปอร์เซนต์การรู้จำดีกว่า HMM

Thesis Title	On-line Thai Hand Written Recognition Using Incremental Learning of Context-Free Grammars
Student	Pongkasem Polsuntikul
Student ID.	45061031
Degree	Master of engineering
Program	Computer engineering
Year	2005
Thesis Advisor	Assoc.Prof. Boontee Kruaprachue

ABSTRACT

This thesis proposes a new approach for on-line handwriting recognition. In this study, string of direction information (chain code) of alphabets was used for generating recognition models. Each character model is represented by a context free grammar (CFG). The main problem, addressed in this study, is how to automatically generate CFG of chain code sequences for each training characters. This CFG has to be generalized enough to classify other similar characters but not too generalized to except other characters. CFG is automatic induction from a selected string of chain code for a character and incrementally refined to be more generalized to accept difference patterns of that character. The there strategies for this purpose are called Loop strategy, Loop-pair strategy and Trap-loop pair strategy. The induction CFG is shown for a sample training set strings. The recognition results of the CFG models are better than the HMM models.

กิตติกรรมประกาศ

คุณความดีอันใดที่บังเกิดจากวิทยานิพนธ์ฉบับนี้ ขอมอบแด่บิดาและมารดาของผู้วิจัย ผู้ที่คอยห่วงใย เอาใจใส่ เป็นแรงใจในการทำวิทยานิพนธ์ฉบับนี้และให้การสนับสนุนการศึกษามาโดยตลอด

วิทยานิพนธ์ฉบับนี้ประสบความสำเร็จลุล่วงได้เป็นอย่างดี โดยได้รับความกรุณาจาก รศ.ดร. บุญธีร์ เครือตราชู ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ที่ได้ให้ความเอาใจใส่แนะนำความรู้และทฤษฎีต่างๆที่ใช้ ชี้แนะแนวทางการแก้ปัญหา ให้คำปรึกษาและให้ความช่วยเหลือเสมอมา ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำในทุกๆเรื่อง ทั้งวิธีการแก้ปัญหาที่เกิดขึ้นในวิทยานิพนธ์และมุมมองในเชิงวิศวกรรมอื่นๆ ซึ่งช่วยให้ผู้วิจัยมีวิสัยทัศน์ที่กว้างไกลขึ้น

ขอขอบคุณอาจารย์ทุกๆท่านที่ประสิทธิ์ประสาทวิชาความรู้ คำแนะนำต่างๆในการศึกษาและการทำวิจัยจนสำเร็จได้ด้วยดี รวมทั้งคำสั่งสอนและอบรมให้ข้าพเจ้าเป็นคนดี ข้าพเจ้าขอกราบขอบพระคุณเป็นอย่างสูง

ข้าพเจ้าขอขอบคุณสำหรับกำลังใจ คำแนะนำ และประสบการณ์ที่ดีจากพี่ ๆ เพื่อน ๆ และน้อง ๆ นักศึกษาป.โททุกท่าน ที่ทำให้บรรยากาศในห้องวิจัยเป็นไปด้วยความสนุกสนาน

ขอขอบคุณ บัณฑิตวิทยาลัยและบัณฑิตศึกษา สถาบันเทคโนโลยีประจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้การสนับสนุนการทำวิทยานิพนธ์นี้

สุดท้ายนี้ หากวิทยานิพนธ์ฉบับนี้มีข้อผิดพลาดประการใดข้าพเจ้าขอน้อมรับไว้เพียงผู้เดียว

ปองเกษม พลสันติกุล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

คอมพิวเตอร์ได้เข้ามามีบทบาทสำคัญในชีวิตประจำวันมากยิ่งขึ้น การใช้ประโยชน์จากคอมพิวเตอร์มีอย่างกว้างขวางและแพร่หลาย กลไกการทำงานระหว่างมนุษย์และคอมพิวเตอร์จึงมีการติดต่อสื่อสารกันมากขึ้นซึ่งก็มีหลายวิธี อาทิเช่น การติดต่อผ่านแป้นพิมพ์หรือคีย์บอร์ด ผ่านการคลิ๊กปุ่มโดยใช้เมาท์ แต่สิ่งที่เราพึงปรารถนาก็คือการติดต่อกับคอมพิวเตอร์ที่ง่าย สะดวก และรวดเร็ว

การใช้วิธีการรู้จำตัวลายมือเขียนของมนุษย์ก็เป็นอีกวิธีหนึ่งที่จะช่วยให้การติดต่อระหว่างคอมพิวเตอร์และมนุษย์มีความสะดวก ง่าย และไม่ต้องใช้ทักษะพิเศษอะไรมาก เพียงแค่การเขียนคำสั่งลงบนอุปกรณ์ที่ใช้รับข้อความของคอมพิวเตอร์ เช่น เขียนผ่านจอมอนิเตอร์ของคอมพิวเตอร์โดยตรง แต่คอมพิวเตอร์จะต้องมีกลไกในการรับข้อมูล แปลงคำสั่ง รู้จำว่าตัวอักษรที่ผู้ใช้เขียนลงไปนั้นว่าคือตัวอักษรอะไร จากนั้นทำการแปลงเป็นคำสั่งที่ระบบสามารถเข้าใจและตอบสนองต่อคำสั่งนั้นๆได้ ซึ่งในงานวิจัยนี้ได้มุ่งประเด็นของการวิจัยไปที่การรู้จำตัวอักษรที่เป็นลายมือเขียนแบบอัตโนมัติ

งานวิจัยทางด้านการรู้จำลายมือเขียนเท่าที่ผ่านมามีการนำเสนอ และตีพิมพ์ในรูปแบบของการประยุกต์ใช้ปัญญาประดิษฐ์ (Artificial intelligence) มากมาย เช่น การใช้รหัสลูกโซ่แบบ GCC [1] ในการเข้ารหัสตัวอักษรเพื่อทำการรู้จำตัวอักษรลายมือเขียนภาษาอังกฤษแบบออนไลน์, งานวิจัยที่ [2] เป็นการนำเสนอการรู้จำลายมือเขียนตัวอักษรภาษาญี่ปุ่นแบบออนไลน์โดยใช้ทิศทางของ feature และการเปลี่ยนทิศทางของ feature ตามเข็มนาฬิกาและทวนเข็มนาฬิกามาเป็นลักษณะเด่นในการรู้จำ งานวิจัยที่ [3] เป็นการนำเสนอวิธีการใช้ Hidden Markov Model และลำดับของสโตร์ก (stroke) ในการรู้จำลายมือเขียนของตัวเลขแบบออนไลน์ งานวิจัยที่ [4] เป็นการนำเสนอการรู้จำตัวอักษรเกาหลีแบบออนไลน์และแบบออฟไลน์ในระบบเดียวกัน โดยใช้วิธีการแยกสโตร์กออกมา แล้วทำการจัดเรียงสโตร์กขึ้นมาใหม่จากนั้นทำการแบ่งเซ็กเมนต์จากชุดของสโตร์ก งานวิจัยที่ [5] เป็นการนำเสนอการรู้จำสมการลายมือเขียนแบบออนไลน์โดยใช้ Hidden Markov Model และ Context dependent graph grammars

วิทยานิพนธ์นี้เป็นงานวิจัยเกี่ยวกับการรู้จำลายมือเขียน ที่มุ่งเน้นไปที่การสร้างโมเดลที่สามารถแสดงตัวอักษรเขียนเหล่านั้นได้ โดยวิธีการที่ใช้จะทำการสร้างโมเดลเหล่านี้จากกลุ่มตัวอย่างของตัวอักษร ซึ่งเรียกว่าเป็นวิธีการเรียนแบบ Passive learning โดยก่อนหน้านี้มีงานวิจัยที่ใช้วิธีการเรียนโมเดลแบบ Passive learning ในงานวิจัยด้านต่างๆเช่น speech recognition การเรียนภาษาธรรมชาติและงานวิจัยอื่นๆอย่างแพร่หลาย

Passive learning คือการเรียนที่ผู้เรียนหรือในที่นี้ก็คือโมเดลที่จะสร้างนี้ ไม่ได้มีบทบาทใดๆในการรับข้อมูลที่เป็นแนวคิดของความรู้ที่จะเรียนนั้นๆเลย ส่วนการเรียนแบบ active learning คือการที่ผู้เรียนสามารถถามและรับรู้แนวคิดของข้อมูลที่จะเรียนได้

วิทยานิพนธ์นี้จึงเป็นการนำเสนอการสร้างโมเดลทางภาษาในการรู้จำลายมือเขียนแบบอัตโนมัติ โดยใช้การแสดงไวยากรณ์ของตัวอักษรหรือ “Grammar” ของอักษรนั้นๆ ที่มีการประยุกต์ใช้หลักการเรียนแบบ passive learning มาช่วยในการเรียนไวยากรณ์ โมเดลทางภาษาที่ใช้ในวิทยานิพนธ์นี้จะเป็นอีกแนวทางหนึ่ง ที่สามารถนำมาประยุกต์ใช้กับงานประเภทรู้จำได้นอกเหนือจากการประยุกต์ใช้วิธีการทางด้านปัญญาประดิษฐ์อย่างเดียว

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

- เพื่อศึกษาวิธีการรู้จำลายมือเขียนที่มีการนำเสนอไปแล้วและทำการวิเคราะห์หาข้อดี-ข้อเสีย
- เพื่อหาแนวทางและประยุกต์ใช้วิธีการใหม่ๆสำหรับการรู้จำลายมือเขียนตามหลักแนวทางการคิดเชิงการวิจัยและการพัฒนา
- เป็นแนวทางเพื่อก่อให้เกิดการวิจัยทางด้านการรู้จำลายมือเขียนต่อเนื่องไปในอนาคต

1.3 สมมุติฐานของการศึกษา

เราสามารถหาคอนเท็กฟรีแกรมม่า (Context Free Grammar) ของตัวอักษรลายมือเขียนจากกลุ่มตัวอย่างเซนโค้ดของลายมือเขียนนั้นๆ และสามารถปรับปรุงคอนเท็กฟรีแกรมม่าให้สามารถแสดงลักษณะทั่วไปหรือที่เรียกว่า “Generalized” ของตัวอักษรนั้นๆได้อย่างมีประสิทธิภาพ โดยแกรมม่าหรือไวยากรณ์ที่ได้ จะต้องมีคุณสมบัติการรู้จำเฉพาะตัวอักษรเขียนที่นำมาสร้างเท่านั้น เนื่องจากตัวอักษรเขียนแต่ละตัวถึงแม้ว่าจะมาจากการเขียนโดยลายมือของคนที่แตกต่างกัน แต่ลักษณะโดยทั่วไป (generalized) ของตัวอักษรนั้นยังคงสามารถบ่งบอกได้ว่าลายมือเขียนนั้นคือตัวอักษรอะไร

1.4 งานวิจัยที่เกี่ยวข้อง

การวิจัยที่เกี่ยวข้องกับการรู้จำตัวอักษรทั้งที่เป็นภาษาไทยและภาษาต่างประเทศอื่นๆเป็นงานวิจัยที่ได้ทำออกมาอย่างต่อเนื่อง โดยใช้วิธีการต่างๆในการวิเคราะห์ข้อมูลตัวอักษรที่จะใช้ในการรู้จำ มีการประยุกต์ใช้วิธีการทางด้านปัญญาประดิษฐ์หรือที่เรียกว่า Artificial Intelligence อย่างกว้างขวาง เช่น การประยุกต์ใช้ฟัซซี่ (Fuzzy) และราฟเซ็ต (Rough set), การใช้เจนเนติกอัลกอริทึม (Genetic Algorithm), การใช้นิวรอลเน็ตเวิร์ค (Neural Network) และการใช้ Hidden Markov Model เป็นต้น ซึ่งจากการศึกษางานวิจัยที่เกี่ยวกับการรู้จำตัวอักษร มีงานวิจัยที่เกี่ยวข้องดังนี้

- Printed Thai Character Recognition using Multifeature and Multilevel Classification [42] งานวิจัยนี้ได้นำการหาลักษณะเด่น ได้แก่ การหาค่าลักษณะขอบนอกของตัวอักษรและความหนาแน่นของจุดตัดภายในตัวอักษรมาจัดกลุ่มหยาบโดยใช้ K-means algorithm ข้อมูลตัวอักษรที่มีความใกล้เคียงกันจะอยู่ในกลุ่มเดียวกัน และเมื่อพิจารณาในแต่ละกลุ่มจะประกอบด้วยตัวอักษรหลายชนิด หลังจากนั้นจะนำข้อมูลเหล่านี้ไปทำการเรียนรู้โดยใช้นิวรอลเน็ตเวิร์คเพื่อจำแนกชนิดของตัวอักษรภายในกลุ่มหยาบนั้นอีกครั้งหนึ่ง
- Online Handwritten Feature with Propotional Invariant [43] งานวิจัยนี้ใช้ลักษณะเด่นทางโครงสร้างแบบยืดหยุ่นมาใช้ในการหาคุณลักษณะเด่นของตัวอักษร เพื่อนำไปเข้ารหัสลูกโซ่แบบ GCC (Generalize Chain Code) การเข้ารหัสนี้ไม่มีการทำ Re-sampling ใหม่ สำหรับลักษณะเด่นของตัวอักษรแต่ละตัวนั้นจะใช้การเซ็กเมนต์ (Segment) โดยใช้ทิศทางแบบตามเข็มนาฬิกาและทวนเข็มนาฬิกาในการแบ่งเซ็กเมนต์ เมื่อได้คุณลักษณะเด่นทั้งหมดมาแล้ว การรู้จำจะใช้วิธีการทดสอบโดยใช้โครงข่ายประสาทเทียม (Neural Network)
- A New Stroke-Base Directional Feature Extraction Approach for Handwritten Chinese Character Recognition [44] งานวิจัยนี้นำเสนอการรู้จำลายมือแบบออนไลน์โดยใช้ทิศทางของเส้นและการเปลี่ยนทิศทางมาเป็นลักษณะเด่นในการรู้จำตัวอักษร โดยทิศทางที่ใช้จะพิจารณาตามการเขียนแบบตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา
- A Handwritten Numeral Character Classification Using Tolerant Rough Set [45] งานวิจัยนี้เป็นการนำเอา Tolerant Rough Set มารู้จำตัวเลขที่เป็นลายมือเขียนโดยการนับจุดดำในส่วนต่างๆของภาพตัวเลข แล้วทำการแบ่งภาพตัวเลขออกเป็น 4 ส่วนตามแนวทแยงมุม จากนั้นนำเอาข้อมูลนี้มาทำการหา upper และ lower approximation objects

การทดสอบการรู้จำจะใช้วิธีการวัดค่าความเหมือน (Similarity measure) ของลักษณะเด่นที่ทำการดึงออกมานั้น.

- Recogniton of Printed Thai Characters using Fuzzy and Rough Sets Theory [46] งานวิจัยนี้ใช้วิธีการแบ่งช่วงของข้อมูลมาทำการหาลักษณะเด่นของตัวอักษร ซึ่งพบว่าการแบ่งช่วงนี้ทำให้สูญเสียค่าของข้อมูลไป แต่จะทำให้เห็นลักษณะของข้อมูลได้มากขึ้น งานวิจัยนี้จะนำเอาวิธีการทางฟัซซี่ มาช่วยในการแบ่งช่วงให้มีความเหมาะสมมากยิ่งขึ้น จากนั้นจึงนำไปเรียนรู้โดยใช้ราฟเซ็ตและสร้างกฎสำหรับการจำแนกกลุ่มหยาบ การทดสอบการจำแนกกลุ่มของตัวอักษรจะใช้ Min-Max Boundaries มาช่วยในการรู้จำ.
- Online Cursive Handwritten Character Recognition Using Hidden Markov Model [47] ในหลักการของ Hidden Markov Model (HMM) มาใช้ในการรู้จำ ซึ่ง HMM ใช้กระบวนการด้านค่าความน่าจะเป็นของกระบวนการที่ซ่อนอยู่ (underlying process) มากำหนดกระบวนการที่อยู่ในความสังเกตุ กระบวนการที่ซ่อนอยู่จะถูกกำหนดจากค่าการกระจายของความน่าจะเป็นสำหรับทรานซิชัน (Transition probability distribution) ในขณะที่สเตจปัจจุบันจะขึ้นกับจะขึ้นกับทรานซิชันของสเตจก่อนหน้ามัน

1.5 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

งานวิจัยทางด้านการรู้จำลายมือเขียนสามารถแบ่งเป็นกลุ่มใหญ่ๆได้ 2 ส่วน คือการรู้จำแบบ On-line และการรู้จำแบบ Off- line ส่วนวิธีการและเทคนิคที่ได้รับความนิยมคือการใช้เทคนิคทางด้านปัญญาประดิษฐ์ (Artificial intelligence) เช่น การเข้ารหัสแบบเชนโค้ด (Generalize Chain Code: GCC), การใช้ Hidden Markov Models (HMMs) และ Genetic Algorithm เป็นต้น ในส่วนของงานวิจัยที่ทำการศึกษาเป็นวิธีการรู้จำลายมือเขียนแบบ On-line ซึ่งจะประยุกต์ใช้วิธีแสดงโมเดลที่ใช้รู้จำด้วยคอนเท็กฟรีแกรมม่า (Context Free Grammar) และทำการปรับปรุงแกรมม่าโดยใช้เทคนิคแบบ ฮิวริสติก (Heuristic)

วัตถุประสงค์ของการนำโมเดลทางภาษาที่เรียกว่าคอนเท็กฟรีแกรมม่ามาสร้างเป็นโมเดลสำหรับการรู้จำนั้น ทั้งนี้เนื่องจากคอนเท็กฟรีแกรมม่าสามารถแปลงเป็น Deterministic Finite Automata (DFA) และ DFA สามารถแปลงเป็น Regular Expression (RE) [41] ได้ ซึ่งจากลักษณะของเชนโค้ดที่ใช้ในการทดลองนั้นหากเขียนให้อยู่ในรูปของ RE จะสามารถแสดงลำดับของเชนโค้ดได้ง่ายและชัดเจนกว่า

ในการเรียนรู้ลำดับเชนโค้ดของมนุษย์เมื่อมีลำดับสตริงแรกแล้วทำการเพิ่มลำดับสตริงแบบใหม่เข้าไปในลำดับสตริงเดิม มนุษย์จะพยายามเทียบสตริงแบบใหม่เข้ากับสตริงอันเดิมที่มี

อยู่โดยจะพยายามหาว่าสตริงใหม่นี้สามารถใช้ร่วมกับสตริงเดิมที่มีอยู่ในช่วงใดได้บ้าง หากมีช่วงใดช่วงหนึ่งในลำดับสตริงใหม่ที่สามารถเข้ากันได้กับสตริงเดิมก็จะสามารถใช้ร่วมกันได้ แต่หากมีช่วงใดช่วงหนึ่งในลำดับสตริงใหม่ที่แตกต่างจากของเดิมออกไปก็จะสร้างเป็นทางเลือกไว้ และเพื่อจำลองการเรียนรู้แบบมนุษย์ในการเรียนเซนโค้ดของวิทยานิพนธ์นี้ ในขั้นตอนการสอนเพื่อให้โมเดลรู้จักสตริงของตัวอักษรนั้นๆในรูปแบบอื่นๆด้วย จึงนำลำดับเซนโค้ดนั้นๆมาสร้างเป็น DFA โดยเส้นทางที่ไปยังสเตจถัดไปของ DFA จะเป็นไปตามลำดับของเซนโค้ด ทำให้โมเดลที่ได้ยังคงลำดับของเซนโค้ดเดิมอยู่ จากนั้นทำการสอนโมเดลที่สร้างขึ้นมานี้โดยพยายามหาเส้นทางที่ใช้สเตจเดิมที่มีอยู่ให้มากที่สุดและหากมีลำดับเซนโค้ดใหม่ที่ไม่เข้ากับเส้นทางในโมเดลเดิมก็จะทำการสร้างสเตจเพิ่มเพื่อเป็นทางเลือกใหม่ โมเดลที่ได้ก็จะสามารถยอมรับสตริงในแบบอื่นที่นำมาสอนได้ด้วย

จาก DFA ที่ทำการสอนแล้วเพื่อให้ง่ายต่อการทดสอบการรู้จำและการแสดงโมเดลที่สร้างได้ การแสดงโมเดลจะใช้รูปแบบของคอนเท็กฟรีแกรมม่า โดยการทดสอบการรู้จำจะใช้วิธีการพาร์สแกรมม่าตามลำดับของสตริงที่ต้องการทดสอบการรู้จำ หากสามารถใช้แกรมม่าพาร์สลำดับสตริงนั้นได้แสดงว่าลำดับสตริงนั้นเป็นลำดับเซนโค้ดของตัวอักษรตามแกรมม่า

1.6 ขอบเขตการวิจัย

เพื่อศึกษาและทดลองถึงประสิทธิภาพของวิธีการที่ได้นำเสนอเปรียบเทียบกับวิธีการอื่นที่มีผู้ทำการวิจัยไปแล้วในด้านการรู้จำตัวอักษรแบบออนไลน์โดยที่

1. วิทยานิพนธ์นี้เป็นการรู้จำตัวอักษรภาษาไทยที่เป็นลายมือเขียนแบบออนไลน์เฉพาะพยัญชนะในภาษาไทยเท่านั้น
2. ในการเขียนตัวอักษรจะต้องไม่มีการยกมือหรือปากกาในเวลาที่เขียน โดยตัวอักษรภาษาไทยที่ใช้ทดลองในงานวิจัยทั้งหมดมี 38 ตัว คือ ก ข ฃ ค ฅ ง จ ฉ ช ซ ฌ ฎ ฏ ฒ ณ ด ต ถ ท ฑ ธ น บ ป ผ ฝ พ ฟ ภ ม ย ร ล ว ห ฬ อ ฮ
3. วิทยานิพนธ์นี้ใช้ข้อมูลเข้าที่ทำการเข้ารหัสแบบเซนโค้ด (GCC) มาทำการสร้างโมเดลทางภาษาที่เรียกว่าคอนเท็กฟรีแกรมม่า (Context Free Grammar) สำหรับการรู้จำตัวอักษร
4. เซนโค้ดของตัวอักษรที่นำมาใช้ในการทดลองได้ทำการผ่านกระบวนการเข้ารหัสเซนโค้ด (GCC) แบบ 8 ทิศทางและตัด noise แล้วเท่านั้น
5. งานวิจัยนี้จะรวมถึงการวิเคราะห์ข้อดี-ข้อเสีย และจุดที่สามารถปรับปรุงได้ของวิธีการของ Stolcke Andreas [7,8]

1.7 ขั้นตอนของการศึกษา

- ศึกษาทฤษฎีและวิธีการรู้จำลายมือเขียนจากงานวิจัยที่เคยมีผู้นำเสนอ
- ศึกษาทฤษฎี วิธีสร้างและการเรียนรู้แกรมม่าจากกลุ่มตัวอย่าง
- ทดลองสร้างแกรมม่าตามวิธีการที่นำเสนอ และวัดผลการรู้จำของแกรมม่าพร้อมทั้งวิเคราะห์ผล
- จัดทำเอกสารประกอบวิทยานิพนธ์

1.8 รายละเอียดในแต่ละบท

เพื่อศึกษา ทดลองถึงประสิทธิภาพของวิธีการนำเสนอและเปรียบเทียบกับวิธีการที่มีอยู่ รวมถึงวิเคราะห์ข้อดีข้อเสียต่างๆ วิทยานิพนธ์เล่มนี้แบ่งออกเป็น 7 บท ซึ่งประกอบด้วยเนื้อหาต่างๆ ดังนี้

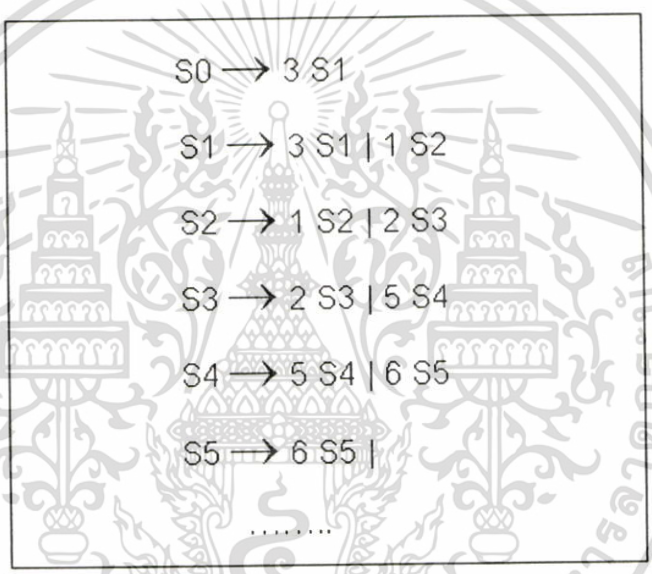
1. บทที่ 1 กล่าวถึงความสำคัญและความเป็นมาของปัญหา จุดประสงค์ของการศึกษาและของเขตของงานวิจัย
2. บทที่ 2 กล่าวถึงกล่าวถึงคอนเท็กฟรีแกรมม่า นิยามต่างๆ ภาษาของไวยากรณ์และการพาร์ส (Parse) ไวยากรณ์
3. บทที่ 3 กล่าวถึงการเรียนแบบ passive learning ของโมเดลทางภาษาแบบต่างๆ
4. บทที่ 4 กล่าวถึงวิธีการเรียนไวยากรณ์จากกลุ่มตัวอย่าง
5. บทที่ 5 การสร้างโมเดลการรู้จำด้วยคอนเท็กฟรีแกรมม่าและวิธีการปรับปรุงคอนเท็กฟรีแกรมม่าของงานวิจัยที่นำเสนอ
6. บทที่ 6 กล่าวถึงการทดลองและผลการทดลอง
7. บทที่ 7 กล่าวถึงการวิเคราะห์และสรุปผลการทดลองพร้อมทั้งเสนอแนะแนวทางที่จะสามารถปรับปรุงต่อไปในอนาคต
8. เอกสารอ้างอิง
9. ภาคผนวกและงานวิจัยที่ได้รับการตีพิมพ์

บทที่ 2

คอนเท็กฟรีแกรมม่า

2.1 บทนำ

เพื่อความเข้าใจในการอธิบายนิยาม และรูปแบบของคอนเท็กฟรีแกรมม่าที่ใช้ในงานวิจัย จึงจำเป็นต้องยกตัวอย่างของไวยากรณ์แบบไม่เป็นทางการก่อน โดยตัวอย่างที่แสดงเป็นส่วนหนึ่งของไวยากรณ์ที่ได้จากการทดลองการสร้างไวยากรณ์ของลายมือเขียน "ก"



รูปที่ 2.1 แสดงตัวอย่างไวยากรณ์

คอนเท็กฟรีแกรมม่าเป็นการนิยามแบบเป็นทางการสำหรับการแสดงนิยาม (notation) ที่เป็นแบบ recursive (recursive definition) ของภาษา ไวยากรณ์ประกอบด้วยตัวแปร (variables) หรือนอนเทอร์มินอล (Nonterminal) ตั้งแต่ 1 ตัวขึ้นไปที่แสดงคลาสของสตริงซึ่งก็คือ "ภาษา" นั่นเอง ในตัวอย่างข้างต้นประกอบด้วยตัวแปร คือ S0, S1, S2, S3, S4 และ S5 ซึ่งแสดงลักษณะเซตของเซนโค้ด "ก" นั่นคือสตริงที่จะสามารถนำมาสร้างรูปแบบ"ภาษา"ของ "ก" แทนโดยสัญลักษณ์ L_n วิธีการสร้างภาษาสามารถใช้สัญลักษณ์และตัวแปรซึ่งเป็นเซตที่กำหนดให้สำหรับไวยากรณ์นั้นๆ อย่างมีหลักการและกฎเกณฑ์ [6] โดยกฎเกณฑ์ในการสร้างจะได้อธิบายต่อไปในบทนี้

2.2 นิยามทั่วไปของคอนเท็กฟรีแกรมม่า (Context Free Grammar)

ส่วนประกอบที่ใช้อธิบายคอนเท็กฟรีแกรมม่าที่สำคัญมี 4 ส่วน คือ

1. เซตจำกัด (finite set) ของสัญลักษณ์ (symbols) ที่ใช้สร้างสตริงของภาษาซึ่งถูกนิยามโดยไวยากรณ์นั้น

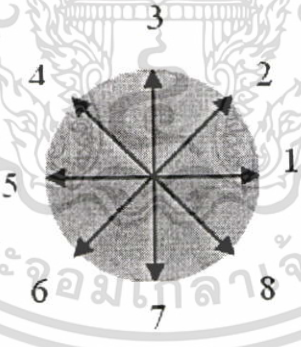
หมายเหตุ: ในงานวิจัยนี้สิ่งที่ใช้ในการสร้างไวยากรณ์คือตัวอย่างเชนโค้ดของตัวอักษร โดยเชนโค้ดนี้อยู่ในรูปของ 8- connected grid ดังรูป 2.2 ดังนั้นเซตจำกัดของสัญลักษณ์ในงานวิจัยนี้จึงหมายถึงหมายเลขเชนโค้ด {1, 2, 3, 4, 5, 6, 7, 8} เราเรียกสัญลักษณ์เหล่านี้ว่าเทอร์มินอล (terminals) หรือ terminal symbols

2. เซตจำกัดของตัวแปรหรือ Variables ซึ่งจะขอเรียกว่านอนเทอร์มินอล (nonterminals) หรือ nonterminal symbols โดยแต่ละ nonterminal แสดงถึงภาษานั่น ก็คือเซตของสตริง

หมายเหตุ: จากตัวอย่างรูปที่ 2.1 มี nonterminals คือ S0, S1, S2, S3, S4 และ S5 ในงานวิจัยนี้จะเริ่มจาก S0 , S1, S2, … , Si

3. ตัวแปรเริ่มต้นที่ใช้แสดงภาษาที่นิยาม ตัวแปรนี้เรียกว่า start symbol ส่วนตัวแปรอื่นๆ จะใช้แสดง auxiliary class ของสตริงซึ่งใช้ช่วยในการนิยามภาษาของ start symbol

หมายเหตุ: จากตัวอย่างในรูปที่ 2.1 มี start symbol คือ "S0"



รูปที่ 2.2 เชนโค้ดในรูป 8-connected grid.

4. เซตจำกัดของโปรดักชั่น (production) หรือกฎ (rules) ที่แสดงนิยามของภาษาแบบ recursive โดยแต่ละโปรดักชั่นประกอบด้วย
   - ตัวแปรที่ถูกนิยามโดยโปรดักชั่นนั้นๆ เรียกว่า head ของโปรดักชั่น (head of production)
   - สัญลักษณ์ของโปรดักชั่น (production symbol)

- สตริงตั้งแต่ศูนย์ตัวขึ้นไปของเทอร์มินอลและนอนเทอร์มินอล เรียกว่า บอดี้ (body) ของโปรดักชั่น ซึ่งเป็นการแสดงการกระจายจาก head แบบทางเดียว

จากองค์ประกอบทั้ง 4 ที่ใช้อธิบายไวยากรณ์ (Grammar) หรือ คอนเท็กฟรีแกรมม่า (context free grammar) หรือ CFG ได้ดังสมการ 2.1

$$G = (V, T, P, S) \qquad (2.1)$$

โดยที่

V คือ เซตของตัวแปร

T คือ เซตของเทอร์มินอล (terminals)

P คือ เซตของโปรดักชั่น (productions)

S คือ start symbol

2.3 การได้มาของไวยากรณ์ (Derivation) โดยใช้คอนเท็กฟรีแกรมม่า

เราจะใช้ไวยากรณ์ในการลงความเห็น (Inference) ว่าสตริงนั้นๆเป็นภาษาของไวยากรณ์หรือไม่ 2 วิธี ดังนี้

- recursive inference คือ การนำเอาสตริงมากระจายโดยใช้ไวยากรณ์เพื่อให้ได้ start symbol โดยมีวิธีการ คือ จากสตริงที่มี จะต้องพยายามแทนจากส่วน body ไปยัง head ของโปรดักชั่นไปเรื่อยๆจนกระทั่งได้ start symbol หากสตริงนั้นสามารถอ้างไปยัง start symbol ได้แสดงว่าสตริงนั้นเป็นภาษาของไวยากรณ์
- derivation คือ เราจะใช้กฎจาก head ไปยัง body โดยจะเริ่มจากการกระจาย start symbol ด้วยโปรดักชั่นใดโปรดักชั่นหนึ่งในไวยากรณ์ นั่นคือใช้โปรดักชั่นอันใดอันหนึ่งในแกรมม่าที่เริ่มด้วย start symbol จากนั้นขยายสติงที่เป็นผลลัพธ์โดยการแทนที่ตัวแปรตัวใดตัวหนึ่งด้วย body ของโปรดักชั่นในไวยากรณ์ไปเรื่อยๆ จนกระทั่งการดีไรว์ได้เทอร์มินอลทั้งหมด ภาษาของไวยากรณ์ทั้งหมดก็คือสตริงทั้งหมดของเทอร์มินอลที่ได้จากการดีไรว์

การทำดีไรว์มี 2 ทางคือ

- กระจายจากซ้ายสุดไปยังขวาสุด เรียกว่า Leftmost derivation
- กระจายจากขวาสุดไปซ้ายสุด เรียกว่า Rightmost derivation

2.3.1 Leftmost derivation

เพื่อให้เห็นการจำกัดจำนวนทางเลือกที่มีในการได้มาของแกรมม่าตามสตริง ที่เป็นอินพุต จึงทำการกำหนดวิธีการแทนตัวแปรแบบ Leftmost derivation ดังนี้คือ จะทำการแทนตัวแปรที่อยู่ทางซ้ายสุดก่อน โดยจะแทนด้วยส่วนของโปรดักชั่นที่เป็น body ของตัวแปรนั้นๆ โดยสัญลักษณ์ที่แสดงการได้มาซึ่งไวยากรณ์ leftmost คือ

$\underset{lm}{\Rightarrow}$ สำหรับการได้มาซึ่งไวยากรณ์ Left-most 1 ครั้ง

$\overset{*}{\underset{lm}{\Rightarrow}}$ สำหรับการได้มาซึ่งไวยากรณ์ Left-most ตั้งแต่ 1 ครั้งขึ้นไป

หรือเราสามารถแสดงการได้มาซึ่งไวยากรณ์ จากแกรมม่า G ได้โดยการวาง G ไว้ใต้ลูกศร $\underset{G}{\Rightarrow}$ ได้

2.3.2 Rightmost derivation

การได้มาซึ่งไวยากรณ์แบบ Rightmost นี้จะทำการแทนตัวแปรที่อยู่ทางขวาสุดก่อนโดยจะแทนส่วนโปรดักชั่นที่เป็น body ของตัวแปรนั้นๆ เช่นกัน สัญลักษณ์ที่ใช้คือ

$\underset{rm}{\Rightarrow}$ สำหรับการได้มาซึ่งไวยากรณ์ Rightmost 1 ครั้ง

$\overset{*}{\underset{rm}{\Rightarrow}}$ สำหรับการได้มาซึ่งไวยากรณ์ Rightmost ตั้งแต่ 1 ครั้งขึ้นไป

และเช่นเดียวกันกับการได้มาซึ่งไวยากรณ์แบบ Leftmost เราสามารถใส่ชื่อแกรมม่าที่ดีไรว์ใต้ลูกศรได้เช่นกัน

2.4 ภาษาของไวยากรณ์

2.4.1 **นิยามภาษาของคอนเท็กฟรี** (Context free language)

ภาษาของไวยากรณ์ [6] คือภาษาที่สามารถได้มาจากแกรมม่านั้นๆ นั่นคือ ถ้า G = (V, T, P, S) เป็น CFG ภาษาของ G สามารถแทนด้วยสัญลักษณ์ L(G) ซึ่งก็คือเซตของเทอร์มินอลสตริง (terminal string) ที่ได้จากการทำการดีไรว์จาก start symbol ของแกรมม่า G นั่นคือ

$$L(G) = \{w \quad in \quad T^* \mid s \overset{*}{\underset{G}{\Rightarrow}} \mathrm{w}\} \qquad (2.2)$$

โดยที่ w คือ ภาษาของแกรมม่า

w คือ คำใดๆที่อยู่ในภาษา W

s คือ Start symbol

ถ้าภาษา L เป็นภาษาของคอนเท็กฟรีแกรมม่าใดๆแล้วจะกล่าวได้ว่า L เป็น context free language หรือ CFL ดังตัวอย่างเช่น

1. $P \rightarrow \varepsilon$
2. $P \rightarrow 0$
3. $P \rightarrow 1$
4. $P \rightarrow 0P0$
5. $P \rightarrow 1P1$

รูปที่ 2.3 Context-free grammar สำหรับ palindromes

จากแกรมม่าในรูปที่ 2.3 ซึ่งเป็นคอนเท็กฟรีแกรมม่าสำหรับ Palindromes นั่นคือสตริงใดๆที่อยู่ในรูป $\{0,1\}^*$ เราสามารถทำการดีไรว์ตัวเลขที่เป็น palindromes ใดๆ ได้โดยการทำ derivation โดยใช้แกรมม่าข้างต้นนี้ได้ดังนี้

กำหนดให้ G_{pal} เป็นแกรมม่าของ Palindromes

$L(G_{pal})$ เป็นภาษาที่สามารถดีไรว์ได้จากแกรมม่า Palindromes

สตริง w = 11011

ทำการดีไรว์โดยใช้แกรมม่าข้างต้นดีไรว์สตริง w โดยเริ่มจาก start symbol P จะได้

$$P \Rightarrow 1P1 \Rightarrow 11P11 \Rightarrow 11011 = w$$

ดังนั้นจากตัวอย่างสามารถสรุปได้ว่า สตริง w เป็น $L(G_{pal})$

การทำ derivation จาก start symbol สามารถสร้างสตริงที่มีส่วนสำคัญในการสร้างภาษาของแกรมม่านั้น ซึ่งเรียกว่า "Sentential forms" นั่นคือ ถ้า G = (V, T, P, S) เป็น CFG แล้วสตริง α ใดๆใน (V U T) จะมี $S \overset{*}{\Rightarrow} \alpha$ เป็น sentential form.

ถ้า $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$ แล้ว α เป็น left-sentential form และ ถ้า $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$ แล้ว α เป็น right-sentential form ดังนั้น L(G) ก็คือ sentential form ใดๆ ที่อยู่ใน T^*

2.5 Regular Grammar

จาก Non-deterministic Finite Automata (NFA) [6] เราสามารถการแสดงโมเดลจาก NFA ให้อยู่ในรูปของแกรมม่าที่เรียกว่า "ริกูล่าแกรมม่า" (Regular Grammar) [41] ซึ่งสามารถทำได้โดยกำหนดให้แต่ละสเตจใน NFA เป็นนอนเทอร์มินอลสำหรับกฎในแกรมม่า แต่ละสัญลักษณ์ที่อยู่บนทรานซิชั่นเป็นเทอร์มินอล

กำหนดให้ S_i เป็นนอนเทอร์มินอลที่อยู่ในฐานะเดียวกับสเตจ i ของ NFA แล้ว

1. start symbol ของแกรมม่าคือ S_0 ซึ่งเป็นนอนเทอร์มินอลที่อยู่ในฐานะเดียวกันกับสเตจเริ่มต้นของ NFA
2. สำหรับแต่ละทรานซิชั่นจากสเตจ i ไปยังสเตจ j บนสัญลักษณ์ a ใดๆ จะสามารถสร้างเป็นกฎที่เป็นโปรดักชั่นของแกรมม่าได้คือ $S_i \rightarrow aS_j$
3. สำหรับเสตจ i ใดๆของ NFA ที่เป็นสเตจสุดท้ายหรือสเตจจบ จะสามารถสร้างเป็นกฎที่เป็นโปรดักชั่นของแกรมม่าได้คือ $S_i \rightarrow \varepsilon$

หมายเหตุ: $\varepsilon - transitions$ ระหว่างสเตจสามารถสร้างเป็นกฎของแกรมม่าได้คือ $S_i \rightarrow S_j$ ซึ่งอาจกล่าวได้ว่าที่สเตจ S_i ใดๆ เราสามารถแทนด้วย S_j นั่นคือ $\varepsilon - transitions$ สามารถบ่งบอกถึงความสมมาตร (equivalence) ของสองสเตจนั้นๆได้

กระบวนการแปลงออโตมาตอนให้เป็นแกรมม่าดังที่กล่าวมาแล้วนี้ สามารถแปลง Deterministic Finite automata (DFA) ให้เป็นริกูลาแกรมม่าได้เช่นกัน และยังสามารถแปลงจากแกรมม่านี้ให้เป็น regular expression ที่สัมพันธ์กับ NFA หรือ DFA นี้ได้อีกด้วย ดังความสัมพันธ์ในรูป 2.4

$$Regular\ Expression \longleftrightarrow NFA\ or\ DFA \longleftrightarrow Regular\ Grammar$$

รูปที่ 2.4 แสดงการความสัมพันธ์ของ Regular Expression, NFA/DFA และ Regular Grammar

2.6 พาร์สทรี (Parse Tree)

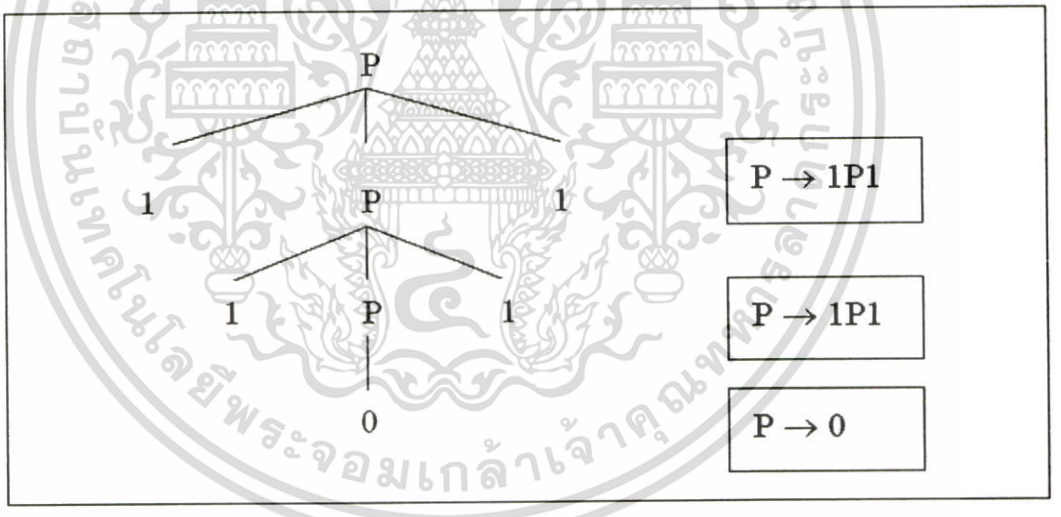
จากคอนเท็กฟรีแกรมม่า จะมีการแทนการดีไรว์ที่เรียกว่า " ทรี " (tree) ซึ่งจะทำให้การดีไรว์ง่าย ชัดเจน และสะดวกมากยิ่งขึ้น ทรีสามารถแสดงให้เห็นการจัดกลุ่ม symbol ของเทอร์มินอลสตริงในสับสตริง (substring)ได้อย่างชัดเจน สามารถทราบได้ว่า symbol นั้นๆเป็นของนอนเทอร์มินอลตัวใดในไวยากรณ์ อย่างไรก็ตามยังมีสิ่งที่สำคัญกว่านั้นก็คือพาร์สทรี (parse tree) ซึ่งเป็นโครงสร้างข้อมูลที่สำคัญในการแสดงทางเลือกที่จะแสดงแหล่งที่มาของสตริงเหล่านั้น

เพื่อให้เห็นโครงสร้างของการกระจายเซนโค้ดได้อย่างชัดเจน ในงานวิทยานิพนธ์นี้เราจะทำการทดสอบการรู้จำของโมเดลโดยการสร้างพาร์สทรีซึ่งจะทำการอ้างถึงแบบ bottom-up นั่นคือการพาร์สไวยากรณ์โดยเริ่มจากเทอร์มินอลไปยัง start symbol

2.6.1 การสร้างพาร์สทรี

กำหนดให้ Grammar G = (V, T, P, S) แล้วพาร์สทรี (Parse tree) ของ G จะสามารถสร้างได้ดังนี้

1. แต่ละโหนดที่มีโหนดภายในสามารถถูกแทนด้วยตัวแปรที่อยู่ใน V
2. แต่ละ child สามารถถูกแทนด้วยตัวแปรหรือเทอร์มินอลอย่างใดอย่างหนึ่งหรือ empty string (ε) อย่างไรก็ตามหากแทนด้วย ε แล้ว จะต้องเป็นแค่ leaf node เท่านั้น ไม่สามารถดีไวร์หรือแทนตัวแปรอย่างอื่นได้อีก
3. ถ้า A มี child คือ $X_1, X_2, \ldots, X_n$ ตามแบบ recursive จากซ้ายไปขวา จะได้ว่าโปรดักชั่นของมันในแกรมม่าคือ $A \rightarrow X_1, X_2, \ldots, X_n$



รูปที่ 2.5 พาร์สทรี (Parse tree) แสดงการดีไวร์ $P \overset{*}{\Rightarrow} 11011$

จากรูป 2.4 แสดงการสร้างพาร์สทรี (Parse tree) ของภาษา palindromes โดยทำการดีไวร์สตริง w = 11011 ซึ่งได้ยกตัวอย่างไปแล้วข้างต้นโดยใช้แกรมม่าจากรูปที่ 2.3

2.6.2 ผลลัพธ์ของพาร์สทรี

จากพาร์สทรีในรูปที่ 2.4 หากพิจารณาที่ leaf node ของทรี จะได้สตริงซึ่งเรียกว่าผลลัพธ์ของพาร์สทรี โดยผลลัพธ์เหล่านี้ได้จากการดีไรว์จาก root หรือ start symbol ของไวยากรณ์ โดยมีสิ่งสำคัญที่ต้องกล่าวถึงสำหรับผลลัพธ์ของทรี คือ

1. ผลลัพธ์ของพาร์สทรี คือ สตริงที่เป็นเทอร์มินอล (terminal string) นั่นคือ เทอร์มินอลหรือ empty string (ε)
2. root ของพาร์สทรี คือ start symbol

บทที่ 3

Passively Learning Finite Automata

3.1 บทนำ

ในงานวิจัยของ Augluin and Smith [10] ได้นิยาม "Inductive inference" ว่าคือกระบวนการของการตั้งสมมติฐาน อ้างอิงและนำไปสู่กฎทั่วๆไป (a general rule) จากกลุ่มตัวอย่าง ซึ่งเป็นเรื่องที่น่าสนใจว่าเราจะมีวิธีการอ้างไปยังกฎต่างๆเหล่านี้ได้อย่างไร สิ่งที่เป็นประเด็นหลักก็คือเราจะสร้างกฎทั่วๆไปหรือ "general rule" สำหรับ Deterministic หรือ Probabilistic Finite Automata (DFA หรือ PFA) ที่มาจากกลุ่มตัวอย่าง ซึ่งกลุ่มตัวอย่างที่เราพูดถึงในบทนี้มาจาก regular language

ตารางที่ 3.1 แสดงตัวย่อของแมชชีนและคำอธิบายโดยสังเขปของแมชชีนในที่จะอ้างถึงในบทนี้

3.1.1 โปรแกรมประยุกต์ต่างๆที่ใช้การอ้างออโตเมตอน

การอ้างออโตเมตอนมีการนำมาใช้ในโปรแกรมประยุกต์จริงบ้าง เช่น ในงานด้านวิศวกรรมอิเล็กทรอนิกส์ทางด้าน Synthesizing Finite Machines (FSMs คือ DFAs ที่มีเอาท์พุตด้วย) ดังตัวอย่างงานวิจัยของ A. Oliveria and S. Edwards [11] DFAs ยังได้ถูกนำไปใช้ในการสร้างเกมส์คอมพิวเตอร์โดยอยู่ในทฤษฎีเกมส์ (a game-theoretic sense) ซึ่งเป็นการเรียนรู้พฤติกรรมของการเลือกทางเลือกในการเล่นเกมส์ เราสามารถสร้างทางเลือกว่าหากผู้เล่นเลือกทางเลือกนี้จากเกมส์จะสามารถมีทางเลือกต่อจากนี้ไปได้ทางใดอีกบ้าง หรือในงานด้านวิศวกรรมหุ่นยนต์ซึ่งนำไปใช้สำหรับการเรียนรู้สภาพแวดล้อมของหุ่น [12][13] เป็นต้น

PFAs ที่อยู่ในรูปของ Hidden Markov Models (HMMs) ได้ถูกประยุกต์ใช้ในโปรแกรมประยุกต์ต่างๆ เช่น ในงานด้านการรู้จำทั้ง speech recognition และการรู้จำลายมือเขียนอย่างแพร่หลายดังเช่นในงานวิจัยของ D. Ron, Y. Singer and N. [14], งานทางด้านวิศวกรรมชีวภาพและการวิเคราะห์ DNA และโปรตีนของ [15][16][17] เป็นต้น.

ตาราง 3.1 แสดงชื่อย่อของแมชชีนและคำอธิบาย

ตัวอักษรย่อ	คำอธิบาย
DFA	Deterministic Finite Automata : เมื่อมีอินพุตสำหรับออโตเมตอนเข้ามาแล้วจะสามารถไปยังสเตจถัดไปได้เพียงหนึ่งเส้นทางเท่านั้น
DPFA	Deterministic Probabilistic Finite Automata : เป็น DFA ที่แต่ละสเตจมีค่าความน่าจะเป็นของการเปลี่ยนไปเป็นสเตจ (state) นั้นๆ
APFA	Acyclic deterministic Probabilistic Finite Automata : เป็น DPFA ที่มีโครงสร้างภายในเป็นแบบที่ไม่มีการวนซ้ำกลับภายในแมชชีน (machine) หรือเรียกว่า "acyclic"
FSM	Finite State Machine : เป็น DFA ที่เปลี่ยนสตริงที่เป็นอินพุตให้กลายเป็นสตริงของเอาท์พุต หรือที่เรียกอีกอย่างหนึ่งว่า "transducer" ซึ่งมี 2 ชนิด คือ Moore machine และ Mearly machine
PFA	Probabilistic Finite Automata : ออโตเมตอนที่มีค่าความน่าจะเป็นสำหรับการไปยังสเตจถัดไป
NFA	Non-deterministic Finite Automata : เป็นแมชชีนเหมือน DFA แต่ NFA จะมีทางเลือกที่เป็นไปได้ของการเปลี่ยนสเตจ ณ สเตจปัจจุบันที่ symbol เดียวกันได้หลายทางเลือก
NPFA	Non-deterministic Probabilistic Finite Automata : เป็น NFA ที่มีค่าความน่าจะเป็นของการเปลี่ยนสเตจไปยังแต่ละสเตจของทางเลือกที่เป็นไปได้ของการเปลี่ยนสเตจ ณ สเตจปัจจุบันที่ symbol เดียวกัน
PSA	Probabilistic Suffix Automata : เป็นกรณีพิเศษของ DPFA โดยที่เลเบลของสเตจไม่ได้เป็นแค่ symbol แต่สามารถเป็นสตริงที่มีความยาว (length) L ซึ่งสามารถใช้กับ Markov Chain ในบางสเตจที่มีเลเบลอยู่ในรูป order L และสเตจอื่นๆที่อยู่ใน order ที่ต่ำกว่ามัน
HMM	Hidden Markov Model : เป็นโมเดลที่มีลักษณะพิเศษของ NPFA นั่นคือการเปลี่ยนค่าความน่าจะเป็นของการเปลี่ยนสเตจจะขึ้นกับ symbol ที่จะทำให้มันกระจาย (emit) ไปยังสเตจนั้นๆ ดังนั้น HMM คือ Markov Chain ซึ่งการเปลี่ยนสเตจขึ้นกับค่าความน่าจะเป็นของ symbol ที่จะทำให้มันเปลี่ยนจากสเตจเดิมไปเป็นสเตจใหม่นั้น

3.1.2 การเปลี่ยนอินพุตเป็นเอาท์พุตจากอัลกอริทึม

อินพุตที่กล่าวถึงในบทนี้เป็นเซต (set) หรือมัลติเซต (multi-set) ของสตริงที่มีความยาวจำกัด โดยบางอัลกอริทึมอาจจะยอมรับหรือ accept แค่สตริง 1 ตัว (single string) ที่มีความยาว m บางอัลกอริทึมอาจยอมรับมัลติเซต (multi-set) ของ m-l+1 สตริง การเรียนแบบแพซซีพที่กล่าวในบทนี้เป็นการเรียนที่มีอินพุตทั้งที่เป็นสตริงที่โมเดลนั้นยอมรับเรียกว่า positive samples ส่วนอินพุตที่โมเดลนั้นไม่ยอมรับแต่นำมาช่วยในการสอน (train) โมเดลเรียกว่า negative samples

3.1.3 อัลกอริทึมแบบ batch และ online

อัลกอริทึมแบบ online จะรับสตริงของอินพุต X_t (สตริงตามเลเบลของมัน) ทุกๆครั้ง ณ ขั้นตอน t แล้วจะรีเทิร์นการเดาทางเลือกที่ดีที่สุด H_t ที่จะเป็นเป้าหมายของออโตเมตอน โดยที่ H_t จะหามาจาก H_{t-1} และ X_t เท่านั้น โดยทั่วไปแล้วมักจะกำหนดข้อบังคับเพิ่มเกี่ยวกับขนาดของ H_t คือจะต้องเป็นฟังก์ชั่นแบบ sublinear ของ t นั่นคือจะทำการเพิ่มอินพุตให้กับโมเดลขณะนั้นแล้วทำการสร้างโมเดลจากอินพุตนั้นไปเรื่อยๆ

อัลกอริทึมแบบ batch จะตรงข้ามกับอัลกอริทึมแบบ online คือ เป็นอัลกอริทึมที่ต้องรอรับเซตของอินพุตสตริง i ให้ครบทั้งหมดก่อนจึงจะเริ่มกระบวนการเดาทางเลือก ดังนั้นอาจกล่าวได้ว่าอัลกอริทึมแบบ batch นี้จะต้องจำทุกอินพุตสตริงในการสร้างออโตเมตอน แต่อย่างไรก็ตามอัลกอริทึมแบบ batch หลายอัลกอริทึมสามารถแปลงให้เป็นอัลกอริทึมแบบ online ได้ในขั้นตอนการ "growing" และ "shrinking" ของโมเดล ซึ่งก็คือขั้นตอนการขยายโมเดล (growth stage) สตริงตัวใหม่จะถูกเพิ่มเข้าไปให้กับโมเดลขณะนั้นและเมื่อใดก็ตามที่โมเดลมีขนาดใหญ่เกินไปก็จะทำการรวมสเตจ [14][18].

3.1.4 อัลกอริทึมแบบ bottom-up และ top-down

อัลกอริทึมแบบ bottom-up [7][8][9] คือวิธีการสร้างโมเดลของออโตเมตอนจากเซตของอินพุตแต่ละตัวก่อน จากนั้นจึงทำการรวมแต่ละสเตจที่เหมือนกันหรือสมมาตร (equivalent) กันในออโตเมตอนเข้าด้วยกันเพื่อให้โมเดลมีขนาดเล็กลง

อัลกอริทึมแบบ top-down [11] คือ วิธีการที่เริ่มต้นด้วยโมเดลที่มี 1 สเตจจากนั้นพยายามทำให้สตริงตัวใหม่สามารถเข้ากับโมเดลให้ได้โดยอาจทำการแยกสเตจออกจากสเตจเดิมแล้วสร้างเส้นทาง (transition) หรือ "link" ไปยังสเตจที่แยกออกมา

ในความเป็นจริงแล้วเราไม่สามารถบอกได้อย่างเป็นเอกฉันท์ว่าโมเดลแบบใดเป็นโมเดลที่ดีที่สุด ดังนั้นในกรณีการเรียน FSMs จากตัวอย่างที่ไม่สมบูรณ์ (incomplete sample) เรามักจะใช้อัลกอริทึมแบบ top-down ซึ่งใช้เวลาในออร์เดอร์ของ n โดย n คือจำนวนสเตจที่น้อยที่สุดในโมเดลสุดท้าย

ปัญหาหลักของวิธีการทั้งแบบ top-down และ bottom-up คือปัญหาในเรื่องการค้นหาภายในโมเดลเพื่อทำการแยก (splitting) และ/หรือการรวม (merging) สเตจในโมเดล

3.2 นิยามของ Finite Automata

คลาสของโมเดลที่เราจะพิจารณาคือ deterministic finite automata (DFAs) และ probabilistic finite automata (PFAs) โดย PFAs จะมีความสัมพันธ์กับ Markov Chains และ Hidden Markov Models (HMMs) ซึ่ง สามารถศึกษา DFAs อย่างละเอียดได้จาก [16][19] และ PFAs จาก [20] สำหรับ Hidden Markov Models สามารถอ่านจาก [21].

3.2.1 Deterministic Finite Automata

นิยาม 3.1 : Deterministic Finite Automaton (DFA) คือ tuple ของ $M = (\Sigma, Q, q_0, F, \delta)$ โดยที่ Σ คือตัวอักษร ที่เป็นอินพุต, Q คือเซตของสเตจ, q_0 คือสเตจเริ่มต้น, $F \subseteq Q$ คือเซตของสเตจที่เป็น final state และ $\delta : \Sigma \times Q \to Q$ คือ transition function ส่วน Non-deterministic Finite Automaton (NFA) จะเหมือน DFA ทุกประการ ต่างกันที่ transition function $\delta \subseteq \Sigma \times Q \times Q$ นั่นคือในแต่ละสเตจ q ที่มีอินพุตเป็น a จะสามารถมีเส้นทาง (transition) ไปยังสเตจถัดไปได้หลายสเตจ นั่นคือสเตจถัดไปไม่สามารถเจาะจงได้แน่นอนว่าจะเป็นสเตจไหน

จากนิยามข้างต้นจะเห็นว่า DFA/NFA เป็นเสมือน "acceptor" ของสตริง ซึ่งสตริง $x = x_1 x_2 ... x_m$ จะถูกยอมรับโดย DFA/NFA ถ้ามีอย่างน้อย 1 เส้นทางตามเลเบลของแต่ละสตริง x จากสเตจเริ่มต้นไปยังสเตจสุดท้าย เซตของสตริงที่ยอมรับโดย DFA/NFA M เรียกว่า ภาษาที่ยอมรับโดย M สัญลักษณ์ที่ใช้คือ L(M) ภาษาใดๆที่ยอมรับโดย DFA/NFA จะเรียกว่า regular ในทำนองเดียวกันเราสามารถใช้ DFA/NFA สร้างภาษาได้โดยการเดินไปตามทุกเส้นทางที่เป็นไปได้จากสเตจเริ่มต้นไปยังสเตจสุดท้าย นั่นคือเรานิยาม L(M) ตามเซตของสตริงที่สร้างโดย M

การไปตามเส้นทางจนถึงสเตจสุดท้ายบ่งบอกถึงการยอมรับ (accept) ภาษานั้นว่าเป็นภาษาของแมชชีน หากการไปตามเส้นทางไม่สามารถไปถึงสเตจสุดท้ายได้ นั่นหมายความว่า DFA/NFA ไม่ยอมรับ (reject) สตริงนั้น หรือก็คือสตริงนั้นไม่ใช่ภาษาของ DFA/NFA นั้น โดยทั่วไปแล้วจากสเตจเดิม เราสามารถไปยังสเตจถัดไปตามเส้นทางที่ถูกเลเบลด้วยสัญลักษณ์ใดๆก็ได้ เรียกว่า Moore machine เมื่อเราเดินไปตามเส้นทางใดๆในออโตเมตอนตามอินพุต x ใดๆแสดงว่าเรากระจายตามชุดสตริงของอินพุตซึ่งอยู่ ณ แต่ละสเตจ ผลลัพธ์ที่ได้คือสตริงของเอาต์พุต y อาจกล่าวได้ว่าผลลัพธ์ของแมชชีนคือ "transducer" ที่เปลี่ยนสตริงอินพุตไปเป็นสตริงเอาต์พุต อีก

วิธีการหนึ่งหากพิจารณาถึงความสัมพันธ์ของเอาต์พุตกับเส้นทาง (arc) แทนที่จะเป็นสเตจเรา เรียกว่า Mearly machine ซึ่ง transducer จะเรียกว่า Finite State Machines (FSMs)

3.2.2 Probabilistic Finite Automata

PFA คือ finite automaton ที่มีความน่าจะเป็นเข้ามาเกี่ยวข้อง ซึ่งโดยทั่วไปออโตเมตอนที่เกี่ยวข้องกับความน่าจะเป็นมีด้วยกัน 2 แบบคือ

1. ค่าความน่าจะเป็นจะถูกกระจายไปตาม emission function ของแต่ละสเตจ ดังนั้น $\gamma(q,a)$ คือความน่าจะเป็นที่สเตจ q กระจายออก (emit) ตามสัญลักษณ์ a ซึ่งเราเรียกว่า Deterministic Probabilistic Finite Automaton (DPFA) เนื่องจากเมื่อสัญลักษณ์ 1 ตัวที่กระจายออกจากสเตจปัจจุบันไปยังสเตจถัดไป ได้ถูกบ่งชี้อย่างแน่นอนแล้วว่าต้องไปยังสเตจใดด้วยความน่าจะเป็นเท่าใด
2. การกำหนดค่าความน่าจะเป็นให้กับแต่ละเส้นทางของ nondeterministic finite automata ซึ่งเรียกว่า Non-deterministic Probabilistic Finite Automaton (NPFA) เนื่องจากสัญลักษณ์เดียวกันอาจมีเส้นทางที่สามารถไปยังสเตจถัดไปได้มากกว่า 1 เส้นทาง และแต่ละเส้นทางก็มีค่าความน่าจะเป็นที่ไม่เท่ากันด้วย ซึ่งความสัมพันธ์ของทรานซิชัน (transition) δ จะเป็นชุดของ $|\Sigma|$ transition matrices $T(\bullet)$ ขนาด $n \times n$ โดย n คือ จำนวนสเตจของ NPFA ฟังก์ชันของทรานซิชันแสดงโดย $T(a)[i,j]$ หมายถึงเส้นทางจากสเตจ i ไปยังสเตจ j ด้วย สัญลักษณ์ a (a เรียกว่า input-NPFA) NPFA นี้มีลักษณะคล้าย Hidden Markov Model ซึ่งจะได้กล่าวถึงในรายละเอียดต่อไป

3.2.2.1 Input-NPFAs และ Output-NPFAs

เรานิยาม input-NPFA คือ $T(a)[i,j] = \Pr(q_j \mid q_i, a)$ เมตริกของการ transition T(a) เป็นเมตริกแบบสต๊อกเชสติก (stochastic matrices) หมายถึง ผลรวมของความน่าจะเป็นในแต่ละแถวมีค่าเป็น 1 นั่นคือ $\sum_j T(a)[i,j] = 1$ สำหรับทุก i ดังนั้นจำนวน degree of freedom คือ $(n-1) \times n \times |\Sigma|$ ส่วนใน output-NPFA ค่าความน่าจะเป็นคือค่าความน่าจะเป็นร่วม (joint probabilities) ทั้งหมดบนทรานส์ซิชัน(transition) และการแพร่กระจายหรืออิมิสชัน (emission) นั่นคือ $T(a)[i,j] = \Pr(a, q_j \mid q_i)$ และ $\Sigma_i \Sigma_a T(a)[i,j] = 1$ สำหรับทุก i จำนวน degree of freedom คือ $n(n|\Sigma| - 1)$ จะเห็นว่า output-NPFA มี degree of freedom มากกว่า input-NPFA ตัวอย่างเช่น สำหรับตัวอักขระใดๆที่ output-NPFA มี degree of freedom $2n^2 - n$ ส่วน input-NPFA จะมี degree of freedom คือ $2n^2 - 2n$ อาจกล่าวได้ว่า output-NPFA มีความเจเนอรัล (general) มากกว่า input-NPFA.

เราสามารถเปลี่ยน input-NPFA เป็น output-NPFA ได้โดยทำการกำหนดการกระจาย (distribution) $\Pr(a \mid q)$ เช่น กำหนดค่าความน่าจะเป็นของการสร้างสตริง เป็นต้น ซึ่งจะกลายเป็นค่า degree of freedom พิเศษเพิ่มขึ้นมาอีก $n(|\Sigma|-1)$ ค่าการกระจายนี้แฝงอยู่ใน output-NPFA เนื่องจาก $\Pr(a \mid q) = \sum_{q'} \Pr(a, q' \mid q))$ จากที่กล่าวมานี้จะเห็นได้ชัดว่า output-NPFA มีความเป็นเจนเนอรัลมากกว่า input-NPFA

การยอมรับสตริงของ PFA จะเกิดขึ้นเมื่อค่าความน่าจะเป็นของมันมากกว่าค่า threshold $\theta \in [0,1]$ ที่กำหนดไว้ เนื่องจากความจริงที่ว่าจำนวน θ ที่เป็นไปได้มีนับไม่ถ้วน จึงสามารถแสดงได้ว่าคลาสของภาษาที่ยอมรับโดย PFAs มีมากกว่าคลาสของภาษาที่ยอมรับโดย DFAs แต่ค่าของ θ ต้องเป็นค่าที่มีความสมเหตุสมผลด้วย PFAs จึงจะมีประสิทธิภาพด้วย PFA ที่มีจุด isolated cut-point สามารถแปลงเป็น DFA ได้ ซึ่ง cut-point θ จะถูกเรียกว่า "isolated" ที่จำกัด PFA ถ้ามี $\delta > 0$ โดยที่ $|\Pr(x) - \theta \geq \delta|$ สำหรับทุกๆ $x \in \Sigma^*$

3.2.2.2 Generators และ Evaluators

Evaluators ของความน่าจะเป็นของการกระจาย D บนสตริงที่มีความยาวจำกัดใดๆ Σ'' จะรับอินพุตเป็นสตริง x แล้วคืนค่ากลับเป็นค่าความน่าจะเป็นภายใต้ D โดย D คือข้อมูลใดๆที่เป็นอินพุตของออโตเมตอน

Generators จะรับอินพุตเป็นสตริง ค่าเอาต์พุตที่ได้จะเป็นสตริงที่สร้างจากอินพุตนั้นๆซึ่งกระจายตัวสอดคล้องกับค่าความน่าจะเป็น D โดย D คือข้อมูลใดๆที่เป็นอินพุตของออโตเมตอน

DPFA และ output-NPFA สามารถสร้างสตริงที่มีความยาว m ในออร์เดอร์ของเวลาที่ O(m) สำหรับ DPFA หากสเตจปัจจุบันอยู่ ณ สเตจ q ที่มี emission symbol คือ a ตามทรานซิชัน $\gamma(q,a)$ จากสเตจเดิมจะไปยังสเตจถัดไปตาม $\delta(q,a)$ ในการเรียน finite automata เราจะใช้โมเดลเสมือนเป็น evaluator ที่จะพิจารณาเส้นทางจากสเตจเริ่มต้นตามเลเบล $x = x_1 x_2 \ldots x_m$ ของ edge และอินพุตไปยังสเตจสุดท้าย หากโมเดลเป็น deterministic ก็จะมีเพียงเส้นทางเดียว แต่หากเป็น non-deterministic ก็จะมีหลายเส้นทางซึ่งเราอาจต้องใช้เมตริกในการหาเส้นทางและค่าความเป็นไปได้ของแต่ละเส้นทางจาก i ไปยัง j ที่เส้นทางซึ่งถูกเลเบลด้วย x แสดงโดย $T(x)[i,j]$ โดย $T(x) = T(x_1)T(x_2)...T(x_m)$ เวลาที่ใช้ในการหาคือ $O(mn^3)$ ค่าความน่าจะเป็นสามารถหาได้จาก

$$\Pr(x) = \sum_{q0,q1,q2,\ldots qm} \pi(0) \Pr(q_1 \mid q_0, x_1) \Pr(q_2 \mid q_1, x_2) \ldots \Pr(q_m \mid q_{m-1}, x_m) \qquad (3.1)$$

เวลาที่ใช้คำนวณค่าความน่าจะเป็นคือ $O(mn^m)$ เนื่องจากมีเส้นทางที่เป็นไปได้ n^m ทาง ของสตริงที่มีความยาว m การหาเส้นทางนี้สามารถใช้อัลกอริทึมแบบไดนามิกที่เรียกว่า forward-backward algorithm (ดูรายละเอียดที่ [21]) ซึ่งสามารถวิเคราะห์ความน่าจะเป็นที่ output-NPFA จะสร้างสตริงโดยวิเคราะห์จากสตริงที่ส่งให้กับ input-NPF เพื่อชักนำ (induce) ไปสู่เอาต์พุต

3.2.3 Hidden Markov Model and Markov Chains

นิยามของ Hidden Markov Model (HMMs) มี 2 นิยาม นิยามแรกซึ่งเป็นนิยามพื้นฐาน กล่าวว่า แต่ละสเตจมีค่า probability distribution Pr(a | q) บน symbol ที่ออกไป (emitted) จากสเตจเดิม (ดังนั้นสเตจก็คือ "hidden") และอีกค่าการกระจายหนึ่งซึ่งไม่ขึ้นกับค่าแรกคือ Pr(q' | q) บน transition (ดังนั้น topology ที่ซ่อนอยู่ก็คือ Markov chain) จากนิยามดังกล่าวจะเห็นได้ว่า HMM มีลักษณะเหมือน NPFA ที่มีค่าความน่าจะเป็นของ transition และ emission ไม่ขึ้นต่อกัน เนื่องจาก Pr(a, q' | q) = Pr(a | q) .Pr(q' | q).

อีกนิยามหนึ่งของ HMM จะเป็นนิยามที่มีความน่าจะเป็นเกี่ยวข้องกับเส้นทาง (arc) คือ มันจะมีค่าความน่าจะเป็นที่เฉพาะสำหรับ joint distribution Pr(a, q' | q) ซึ่งนิยามนี้จะเหมือนกับ NPFA ข้างต้น ประโยชน์ของนิยามนี้คือสามารถมีเส้นทางที่เรียกว่า silent arc นั่นคือ เส้นทางที่กระจายด้วย empty symbol ซึ่งใช้ในการข้ามบางสเตจหรือบางส่วนของโมเดลหรือใช้ในกรณีของ "looping back" ดังตัวอย่างใน [21] ของการทำ speech recognition.

HMM ไม่สามารถแปลงเป็น DPFA ได้เนื่องจาก transition probabilities ของแต่ละ symbol ไม่ขึ้นต่อกัน ดังนั้นสเตจถัดไปจึงไม่สามารถบ่งชี้ได้อย่างแน่นอนจากสเตจและอินพุต ณ ขณะนั้นได้ หมายเหตุ: HMM มีลักษณะของ non-determinism.

พิจารณา subclass ของ DPFAs ที่เรียกว่า Probabilistic Suffix Automata (PSAs) ซึ่งใน PSA นั้น ทุกๆสเตจที่ถูกเลเบลโดยสตริงที่มีความยาวจำกัดใน Σ^* ถ้าทุกๆสตริงมีความยาวมากที่สุดได้ที่ L เราจะเรียกว่า L-PSA เลเบลจะสามารถบ่งบอกถึงว่า "history" ที่เราสนใจของสตริงนั้นๆว่ามีมากเท่าใด เช่นถ้ามี arc $p \xrightarrow{a} q$ แล้วเลเบลของ q จะต้องเป็น subfix ของ s.a โดยที่ s คือเลเบลของ p และ s.a แสดงการต่อกันของสตริง (string concatenation) ถ้าเซตของสเตจคือทุกๆสตริงใน Σ^L แล้วจะได้ว่า ลำดับของ L เป็น markov chain เนื่องจาก ค่าความน่าจะเป็นของสเตจถัดไปจะขึ้นกับสัญลักษณ์ L ตัวสุดท้าย.

3.3 การเรียน DFAs

3.3.1 Definition of success

เป้าหมายที่สำคัญที่พิจารณาคือการพยายามหาออโตเมตอนที่เล็กที่สุด และเหมาะสมกับอินพุตที่มีลักษณะเป็นอินพุตที่มีความยาวจำกัด ซึ่งเป็นสิ่งที่ยากมากที่จะหาได้ ดังนั้นเราจึงพยายามหาออโตเมตอนที่พอจะยอมรับได้แทน เราจะทำการเรียนโมเดลที่พอจะเป็นโมเดลที่คล้ายคลึงกับโมเดลคอนเซ็ปต์หรือโมเดลเป้าหมายของเรา การเรียนนี้เรียกว่า PAC-learning (Probably Approximately Correct) โดยสามารถหารายละเอียดเพิ่มได้จาก [22][23]

นิยาม 3.2 : ให้ X เป็นขอบเขตของปัญหา C เป็นคลาสที่เป็นคอนเซ็ปต์ของ X (นั่นคือเซตของสับเซตของ X) และ H เป็นคลาสของสมมติฐานของ X (นั่นคือเซตของตัวแทนสำหรับทุกๆ ฟังก์ชันใน C เช่น X เป็น Σ^* ดังนั้น C คือเซตของ regular language และ H คือเซตของ DFAs) อาจกล่าวว่า C คือ PAC ที่สามารถเรียนได้โดยใช้ H หากมีอัลกอริทึม L ที่มีคุณสมบัติดังนี้ สำหรับทุกๆคอนเซ็ปต์ที่เป็นเป้าหมาย $c \in C$ สำหรับการกระจาย D บน X และสำหรับทุกๆ พารามิเตอร์ของค่าความผิดพลาด $0 < \varepsilon < 1/2$ และพารามิเตอร์ของค่าความเชื่อมั่น $0 < \delta < 1/2$ ถ้า L ได้รับตัวอย่างที่สุ่มมาของ c ซึ่งเกิดขึ้นมาโดยสัมพันธ์กับ D พร้มกับความน่าจะเป็นอย่างน้อย $1-\delta$ แล้ว L จะให้ผลลัพธ์เป็นคอนเซ็ปต์ของสมมติฐาน $h \in H$ ซึ่งมี $\varepsilon \geq error(h) = pr_{x \in D}[c(x) \neq h(x)]$ และถ้า L รันโดยใช้เวลาที่เป็นโพลิโนเมียลใน $1/\varepsilon, 1/\delta, n$ (มิติของขอบเขตตัวอย่าง) และ size(c) (ขนาดของคอนเซ็ปต์ภายใต้กลุ่มตัวอย่างที่คงที่) แล้วจะเรียก C ว่าเป็น PAC ที่สามารถเรียนได้อย่างมีประสิทธิภาพและยอมรับได้

3.3.2 Complexity results

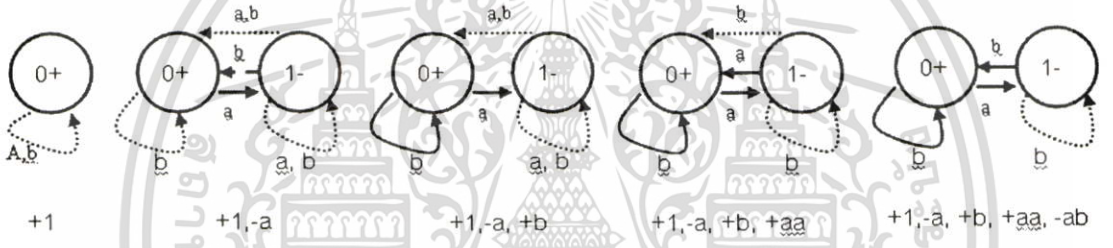
Uniform complete sample คือ เซตของทุกสตริงที่เป็นเลเบลที่มีความยาวมากที่สุด m

การหาและการแสดงภาษา L ให้ครบถ้วนนั้นทำได้ยาก [6] ได้แสดงให้เห็นว่ามันเป็นไปได้ยากที่จะบ่งชี้ L ได้อย่างถูกต้องสมบูรณ์ ถึงแม้ว่าจะมีข้อมูลเป็นเซตจำกัดก็ตาม การเรียน DFA จะต้องอาศัยกลุ่มตัวอย่างซึ่งมีผลต่อการบ่งชี้ L การหากลุ่มตัวอย่างต้องครอบคลุมทุกกรณีของภาษา หากตัวอย่างไม่ใช่ตัวอย่างแบบ uniform complete และมีความผิดพลาดที่ไม่ครอบคลุมเพียงในสัดส่วนของ $0 < \varepsilon < 1$ โดย ε คือ พารามิเตอร์ของค่าความผิดพลาด (นั่นคืออินพุตมีสตริงเพียง $(|\Sigma''|)^{\epsilon}$ ตัว) แล้ว Angluin[24] แสดงให้เห็นว่าปัญหานี้จะเป็นปัญหาแบบ NP-hard (ปัญหาที่ยากเกินกว่า non-deterministic turing machine จะสามารถแก้ปัญหาได้)

3.3.3 อัลกอริทึมแบบ Top-down

3.3.3.1 Exactly learning DFAs using an ordered complete presentation

Porat และ Feldman [25] นำเสนออัลกอริทึมแบบออนไลน์สำหรับการเรียน DFAs นั่นคือ แต่ละขั้นตอนจะต้องได้ผลลัพธ์ที่เป็น DFA ซึ่งเกิดจากการเดาที่ดีที่สุดออกมา (ในที่นี้คือโมเดลที่เล็กที่สุดที่มีข้อมูลที่เห็นอยู่ ณ ขณะนั้น) การเดานี้จะนำการเดาก่อนหน้านี้ ร่วมกับข้อมูลของอินพุต ขณะนั้น เวลาที่ใช้จะเป็นโพลิโนเมียล n โดย n คือจำนวนสเตจน้อยที่สุดในออโตเมตอน (หมายเหตุ: อัลกอริทึมแบบ batch จะเก็บอินพุตทั้งหมดไว้ก่อนแล้วจึงค่อยประมวลผล) อัลกอริทึมจะทำการหาสเตจใหม่แล้วสร้างเส้นทางที่เป็นไปได้ทั้งหมดไปยังสเตจใหม่และเส้นทางจากสเตจใหม่ไปยังสเตจอื่นด้วย ดังตัวอย่างในรูป 3.1



รูปที่ 3.1 แสดงตัวอย่างการเรียน DFA แบบออนไลน์

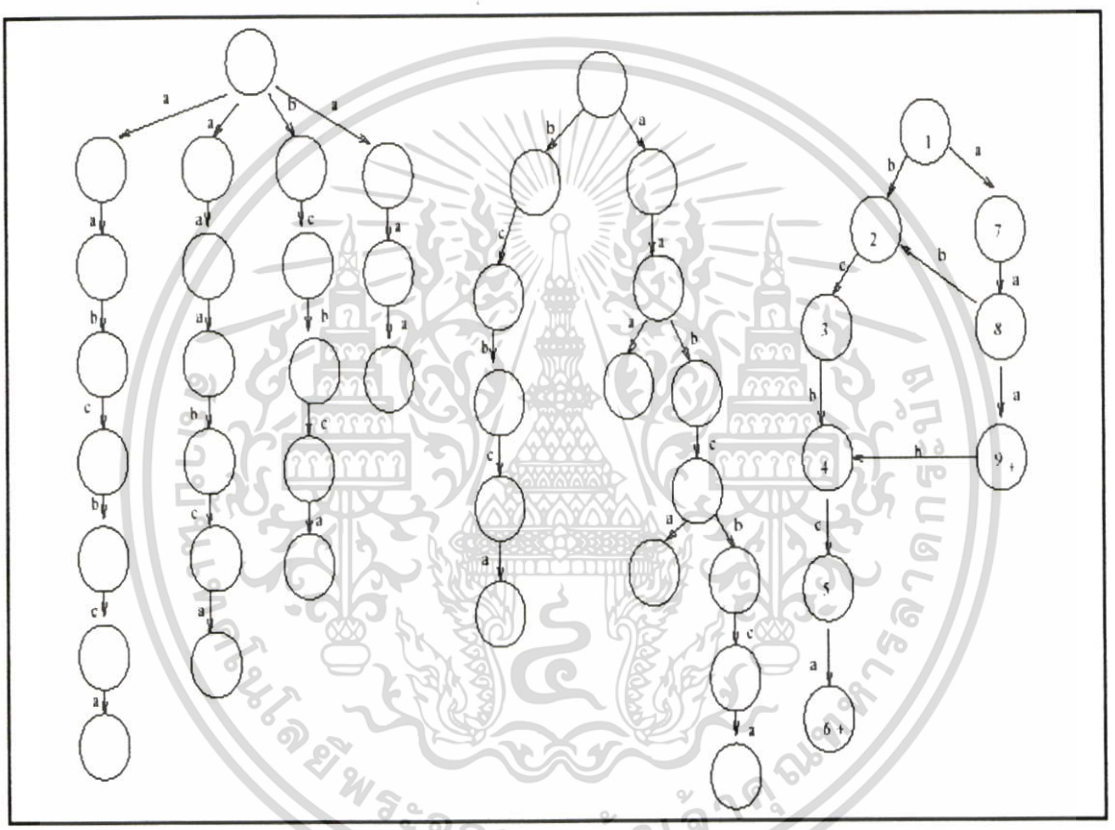
รูปที่3.1 เป็นการเรียน DFA แบบออนไลน์โดยเซตของอินพุตคือ $+\lambda,-s,+b,+ss,-sb,...$ โดยที่ $\Sigma = \{a,b\}$ นั่นคือมี a,b เป็นสมาชิกของภาษานี้ จากการเรียน DFA ที่อยู่ขวาสุดคือออโตเมตอนที่เล็กที่สุด เส้นประแสดงการเพิ่มเส้นทางเมื่อได้รับอินพุตส่วนเส้นปกติคือเส้นทางเดิมที่มีอยู่แล้ว I แสดง λ หรือ empty string 0 คือสเตจที่เป็น accept state และ 1 คือ reject state เวลาที่ใช้คือ $O(n^2)$

3.3.3.2 Exactly learning FSMs from an incomplete specification

Oliveira และ Wdwards [11] ได้เสนอวิธีการที่ใช้กับการเรียน DFA ทั่วไปที่มีเซตของคู่อินพุตและเอาท์พุตที่ตายตัว กรณีนี้เกิดขึ้นเมื่อทำการสร้าง FSMs จากรายละเอียดบางส่วนที่ได้ถูกระบุซึ่งเป็น behavioral specification ไว้ เช่น อาจเป็นเลเบลที่เป็นพรีฟิกซ์ทรี (prefix tree) ดังตัวอย่างรูปที่ 3.2 อัลกอริทึมแบบนี้เป็นแบบ batch วิธีการนี้นำ bottom-up มาใช้ในการพยายามที่จะรวมสเตจในทรีโดยเวลาที่ใช้เป็นเอ็กโปรแนลเชียลของ t (จำนวนโหนดในทรี) และ top-down

มาใช้ในการเดินไปภายในส่วนต่างๆของทรีและใช้ในการขยายทรี เวลาที่ใช้เป็นเอ็กโปรแนลเชียล ของ n (จำนวนสเตจใน minimal automaton สุดท้าย)

วิธีการของอัลกอริทึมนี้คือพยายามสร้างแมชชีนแบบ deterministic Moore หรือ Mearly ที่มีสเตจไม่เกิน n ที่สอดคล้องกับอินพุตที่เห็น การรวมสเตจต้องทำการค้นหาไปในทรีเพื่อให้ได้การรวมสเตจที่เหมาะสมและได้จำนวนสเตจที่น้อยที่สุดที่เหมาะกับอินพุต แต่ดังที่กล่าวไว้ข้างต้นว่าการหาแมชชีนที่เหมาะสมจากกลุ่มตัวอย่างเป็นปัญหาแบบ NP-complete ดังนั้นเวลาที่ใช้อาจเป็นแอ็กโปแนลเชียล



รูปที่ 3.2 แสดง canonical automata สำหรับเซตกลุ่มตัวอย่างที่เป็นบวก (positive) {bcbca,aaabca,aabcbca,aaa}

ด้านซ้าย คือ maximum canonical automaton กลางคือ prefix tree canonical automaton ด้านขวาคือ DAG (Directed Acyclic Graph) canonical automaton

จากรูป 3.2 กลุ่มตัวอย่างที่เป็นบวกคือ {bcbca,aaabca,aabcbca,aaa} ทุกสเตจเป็น reject state ยกเว้นสเตจที่เป็น leave ที่เป็น accept state โดยสเตจที่เป็นอินเทอร์นอล (internal)

จะเลเบลด้วย + หากเราต้องการแสดง FSM แต่ละเส้นทางสามารถเลเบลด้วยคู่อินพุตและเอาต์พุต ซึ่งสิ่งนี้เรียกว่า behavioral specification

สมมติว่ามีแมชชีน M และสเตจปัจจุบันคือ q วิธีการคือจะพยายามหาทรานซิชันของเลเบล $q \xrightarrow{x/y'} q'$ ที่เหมาะสมภายในพรีฟิกซ์ทรีใน M ($q \xrightarrow{x/y'} q'$ แสดงเส้นทางจาก q ไปยัง q' โดยเปลี่ยนสตริงที่เป็นอินพุต x เป็นสตริงเอาต์พุต y) ซึ่งมีความเป็นไปได้ 2 ทางคือ

- ไม่มีทรานซิชันในแมชชีน M ปัจจุบันเลย อัลกอริทึมจะพิจารณาทุกๆเส้นทางที่เป็นไปได้ที่จะเพิ่มทรานซิชันเข้าไปใน M นั่นคือ ทดลองทุกสเตจเพื่อจะเป็นสเตจปลายทางให้กับทรานซิชันซึ่งจะทำการลองแบบนี้ไปเรื่อยๆไปจนครบทุกทรานซิชัน หากมันสามารถหาสเตจถัดไปที่เหมาะสมได้อัลกอริทึมจะเทอร์มิเนต (terminate) แบบสมบูรณ์ แต่หากไม่สำเร็จมันจะแบ็คแทรก (backtracks) แล้วลองหาปลายทางใหม่ที่ยังไม่ได้ลอง และมันต้องพิจารณาปลายทางที่เป็นสเตจใหม่ที่อาจต้องเพิ่มเข้าไปด้วย
- มีทรานซิชั่นที่เหมาะสมกับ $q \xrightarrow{x/y'} q'$ หากเอาต์พุตที่ได้สอดคล้องกับ $y = y'$ ก็สามารถใช้ทรานซิชั่นนั้นได้เลย หากไม่ก็ต้องพิจารณาเป็นกรณีแรกไป

เพื่อลดงานในการค้นหาสเตจปลายทางของทรานซิชัน หากมี 2 เส้นทาง $q \xrightarrow{x/y} q'$ $q \xrightarrow{x/z} q'$ และออโตเมตอนนี้เป็น deterministic ดังนั้นจะสามารถทราบได้เป็นนัยว่า y และ z จะต้องเท่ากัน ดังนั้นจึงสามารถลดการค้นหาได้ การหาเอาต์พุตที่เท่ากันนี้สามารถใช้ไดอะแกรมที่เรียกว่า Multi-value Decision Diagram (MDD) [6] ช่วยได้

3.3.3.3 Enumerating all compatible automata of a given size

Gaines[26] แสดงวิธีการในการระบุว่า จากทุกแมชชีน n-state ของ Moore แมชชีนใดเหมาะสมกับอินพุตมากที่สุด หากแมชชีนแบบ n-state นี้เล็กเกินไปแสดงว่าแมชชีนของ Moore นี้อาจจะเป็น non-deterministic แต่ละอินพุตจะถูกส่งไปในแมชชีน เวลาที่ใช้ขึ้นกับจำนวนเส้นทางที่เกิดทรานซิชั่น ในกรณีนี้เราสามารถหาความน่าจะเป็นของการกระจายตามทรานซิชั่นเพื่อแก้ไขข้อผิดพลาดของการเรียนที่เกิดขึ้น

อัลกอริทึมนี้จะให้ผลลัพธ์ที่เรียกว่า admissible subspace ของการแก้ปัญหานี้ โดยประกอบด้วยชื่อของสเตจนั้น กับค่าความเหมาะสมของมันว่ามันเหมาะสมกับอินพุตมากน้อยเพียงใด เมื่อทราบค่าความเหมาะสมนี้มาแล้วก็จะเป็นหน้าที่ของผู้ใช้ที่จะพิจารณาว่าควรตัดสเตจนั้นออกหรือว่าให้มันคงไว้ในแมชชีน

3.3.4 อัลกอริทึมแบบ Bottom-Up: การทำ minimizing canonical automata

เราสามารถสร้าง canonical automaton M ที่เข้ากันได้กับเซตจำกัดของอินพุต I ดังแสดงในรูปที่ 3.2 (ออโตเมตอน M จะเรียกว่า canonical ถ้า L(M) = I) สิ่งที่เราต้องการคือหาออโตเม

ตอนที่เล็กที่สุดที่เหมาะสมกับเซตของอินพุต ดังนั้นจาก canonical ที่ได้ ต้องทำการย่อขนาดออโตเมตอนโดยการรวมสเตจที่สมมาตรกันให้เป็นสเตจเดียวกัน

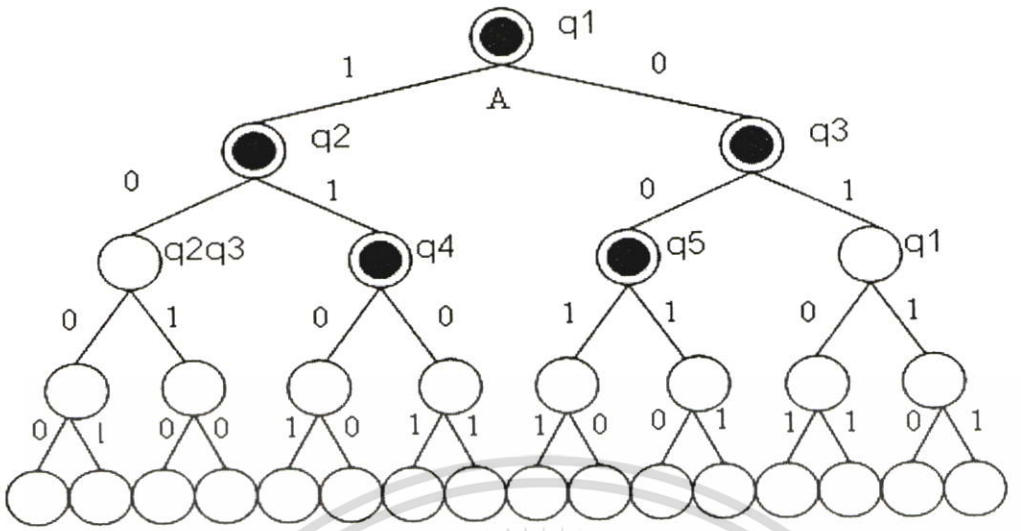
วิธีการที่ง่ายที่สุดในการย่อส่วน DFA ที่มีจำนวนสเตจ n สเตจจะเป็น $O(n^2)$ อีกอัลกอริทึมหนึ่งที่มีความซับซ้อนขึ้นมาจะใช้เวลา $O(n \log n)$ [19] วิธีการนี้จะใช้ความสัมพันธ์ของความสมมาตรมาจัดสเตจเหล่านี้เป็นคลาส โดยความสัมพันธ์ของการสมมาตรของสเตจ $q \equiv q'$ ก็ต่อเมื่อ $T(q) \equiv T(q')$ โดยที่ $T(q) = \{x : \delta(q,x) \in F\}$ เป็นส่วนปลาย (tail) ของ q รายละเอียดเพิ่มเติมสามารถดูได้จาก [6]

หากเซตของอินพุตประกอบด้วยตัวอย่างที่เป็นบวก (positive sample) นั่นคือ ตัวอย่างที่เป็นภาษาของออโตเมตอน แล้วออโตเมตอน M' ใดๆที่สืบมาจาก canonical automaton M ด้วยวิธีการรวม (merge) สเตจจะเข้ากับเซตของอินพุตนั้นๆได้หาก $L(M') \supseteq L(M) = I$ ตัวอย่างเช่น หากต้องการรวมทุกสเตจเข้าด้วยกันเพื่อให้กลายเป็นออโตเมตอนแบบ 1 สเตจ ดังนั้นภาษาของออโตเมตอนนี้คือ $L(M') = \Sigma^*$ วิธีการรวมสเตจเหล่านี้มีหลายอัลกอริทึมที่ถูกนำเสนอแต่การเลือกใช้แต่ละอัลกอริทึมนั้นต้องชั่งน้ำหนักกันระหว่างขนาดของโมเดลและความ generalize ของโมเดล (วัดได้จาก $|L(M') - L(M)|$) ในการเรียน PDA (Push Down Automata) หากสนใจเรื่องความน่าจะเป็นของการกระจายจะใช้การเรียนจากตัวอย่างที่เป็นบวกเท่านั้น

3.3.4.1 การเรียน FSMs จากตัวอย่างที่เป็น Uniform complete: อัลกอริทึมแบบ Russian

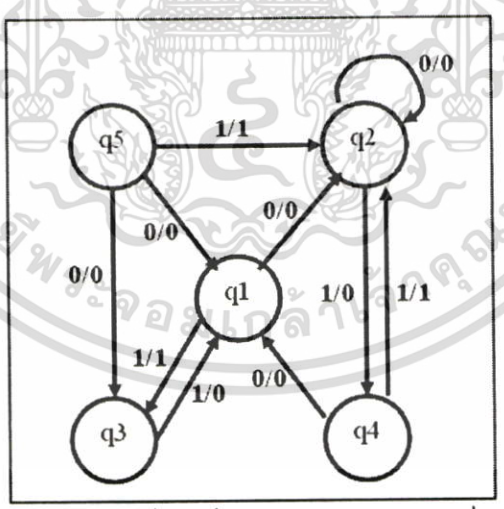
อัลกอริทึมแบบ Russian ถูกนำเสนอโดย [27] อินพุตของอัลกอริทึมนี้เป็นทรีของลาเบลแบบ complete ที่มีความสูง $2n-1$ วิธีการนี้มี 2 ช่วงด้วยกัน ช่วงแรกเป็นการท่องเข้าไปในทรีแบบ breadth-first (lexicographic) เพื่อทำการบ่งชี้สเตจ Q นั่นคือการคำนวณการแม็พ (mapping) จากโหนดในทรีไปสู่สเตจในแมชชีน ช่วงที่สองเป็นการหาโหนดที่สอดคล้องกับแต่ละสเตจแล้วเพิ่มทรานซิชั่นออกจากสเตจนั้น สเตจที่อยู่ในคลาสที่แสดงความสมมาตรกันจะแสดงโดย $q_1 \equiv q_2$, ให้ $[n_i]$ แสดงคลาสที่สมมาตรกับโหนด $[n_i]$ สำหรับแต่ละสเตจที่สมมาตร $[n_i]$ จะต้องมีตัวแทนที่ใช้แสดงโหนดนั้นๆ ซึ่งอาจจะใช้โหนดใดโหนดหนึ่งในคลาสนั้นก็ได้แต่ต้องใช้อย่างชัดเจนไม่เปลี่ยนแปลงตลอดการทำงาน ในที่นี่จะขอใช้โหนดแรกของคลาสนั้นๆแสดงด้วยสัญลักษณ์ $[n_i]$

อินพุต : เลเบลของพรีฟิกซ์ทรี T ที่ complete ซึ่งมีความสูง $2n-1$



รูปที่ 3.3 แสดงทรีของลาเบลแบบ Uniform complete

จากรูปที่ 3.3 เป็นรูปทรีของลาเบลซึ่งอาจกล่าวได้ว่าทรีนี้เป็นสิ่งที่ใช้บ่งชี้พฤติกรรมของอินพุต/เอาท์พุตของ FSM ตัวอย่างเช่นอินพุค 111 จะถูกเปลี่ยนรูปเป็น 0101 (ตามการเดินเส้นทางด้านขวาของทรีลงมา) โหนดที่มีสีดำข้างในคือสเตจที่อยู่ในแมชชีนที่เล็กที่สุด (final minimal machine) ส่วนโหนดอื่นๆเป็นโนหดที่มีสับทรีเหมือนโหนดอื่น



รูปที่ 3.4 แสดง FSM ที่เล็กที่สุด (minimal FSM) ที่ได้จากทรีในรูปที่ 3.3

เอาต์พุต : FSM M ที่เล็กที่สุดที่ประกอบด้วย T

อัลกอริทึมแบบ Russian แสดงดังรูป 3.5

Algorithm:

Q := ϕ

For each node $[n_i]$ in T in breadth-first order

If $\exists [n_i] \in Q$ s.t. $n_j \equiv n_i$

Then add n_i to $[n_i]$

Else $Q := Q \cup [n_i]$ /*make a new state*/

For each $[n_i] \in Q$

For each edge $n_i \xrightarrow{a} n_j$ in T

Add the transition $[n_i] \xrightarrow{a} [n_j]$ to M

รูปที่ 3.5 แสดงอัลกอริทึมแบบ Russian

จากอัลกอริทึมแบบ Russianโหนด 2 โหนด คือ n_j, n_i มีความสมมาตรกัน (แสดงโดย $n_j \equiv n_i$) ถ้าส่วนปลายหรือเทล (tail) ของทั้ง 2 โหนดมีความสมมาตรกันด้วย ซึ่งส่วนปลายในที่นี้คือสับทรีภายใต้โหนดนั้นๆ โหนดบ่งชี้พฤติกรรมของแมชชีนโดยเริ่มที่สเตจ q สำหรับสตริงที่ปรากฏในส่วนเทล ตัวอย่างเช่นโหนด A ระบุว่า 1111 จะต้องถูกเปลี่ยนเป็น 0101 แสดงโดยสัญลักษณ์ $A(1111) = 0101$ โหนด A และโหนด B สมมาตรกันเพราะมันระบุพฤติกรรมที่เหมือนกันสำหรับการเปลี่ยนสตริงเช่น $A(11) = 01$ และ $B(11) = 01$ ซึ่งในกรณีนี้จากตัวอย่างทรีรูปที่ 3 จะเห็นได้อย่างชัดเจนว่ามี 5 สเตจที่สมมาตรและผลการแปลงเป็น FSM ที่เล็กที่สุด (minimal FSM) จะได้ดังแสดงในรูปที่ 3.4

จากอัลกอริทึมข้างต้น อาจมีกรณีที่เป็นไปได้ที่ในช่วงที่ 2 นั้น n_j ซึ่งเป็นโหนดลูกสามารถอยู่ในคลาสที่สมมาตรได้หลายคลาส ซึ่งเป็นสาเหตุที่ทำให้บางโหนดในรูปที่ 3.3 มีได้หลายเลเบล โดยกรณีที่โหนดอยู่ในคลาสที่สมมาตรได้หลายคลาสนี้ทำให้สามารถสร้างแฟมมิลี (Family) ของ deterministic FSMs ที่สมมาตรกันได้

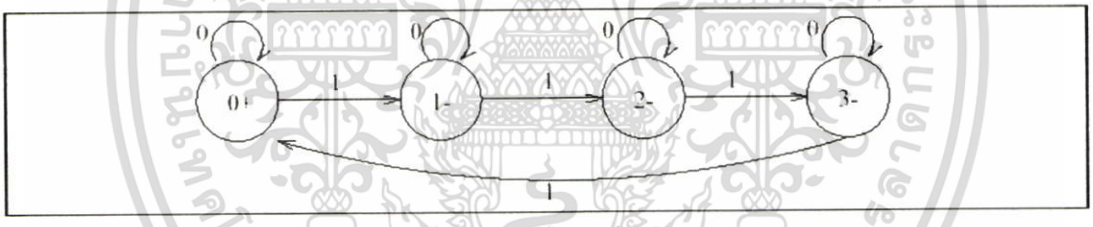
3.3.4.2 ความซับซ้อนของอัลกอริทึมแบบ Russian

เมื่อเปรียบเทียบโหนด 2 โหนดเพื่อดูความสมมารตรกันนั้นไม่จำเป็นต้องทำการเปรียบเทียบทุกตัวในสับทรี เราจะกล่าวได้ว่าสองสเตจ q, q' เป็น k-distinguishable หากมีคำ (word) x ที่มีความยาว k ซึ่งมีเส้นทางที่เลเบลโดย x เริ่มต้นที่สเตจ q แล้วสิ้นสุดที่ accept state และอีกเส้นทางหนึ่ง เริ่มต้นที่สเตจ q แล้วสิ้นสุดที่สเตจอื่นที่ไม่ใช่ accept state แต่หากมีเส้นทางทั้งจาก q หรือ q' ไปสิ้นสุดที่ accept state จะเรียกว่า 2 สเตจนี้เป็น k-indistinguishable หรือ k-

equivalent แสดงโดย $q \equiv kq'$ สองสเตจเป็น indistinguishable หรือสมมาตรกันถ้ามี k-indistinguishable สำหรับทุก k ในรูปแบบของทรี 2 โหนดจะเป็น k-equivalent ถ้าสับทรีที่ความสูงต่ำกว่า k มีความเหมือนกัน โดยสับทรีที่ใต้ความสูง k ของโหนดนั้นเรียกว่า k-tail หรือ k-signature, คำว่า indistinguishable ก็คือความสัมพันธ์แบบสมมาตรกันหรือ equivalent กันนั่นเอง

ดีกรีของการ distinguishable ของ (minimal) FSM (สัญลักษณ์คือ $d(M)$) คือค่าที่เล็กที่สุด k ที่สเตจ 2 สเตจใดๆเป็น k-distinguishable ดีกรีของความสามารถในการเข้าถึง (accessibility) ของ FSM (สัญลักษณ์คือ $a(M)$) คือ ค่าที่เล็กที่สุด k ที่สเตจใดๆสามารถเข้าถึงได้จาก q_0 โดยคำที่มีความยาวทั้งหมด k

จากรูป 3.6 เครื่องหมาย + แสดงถึง accept state ส่วน – แสดงถึง reject state แมชชีนนี้ยอมรับสตริง $x \in \{0,1\}^+$ ซึ่งจำนวนของ 1 ใน x จะคูณด้วย 4 เท่า 0 คือสเตจเริ่มต้น จากรูปจะเห็นชัดเจนว่าเราต้องการสตริงที่มีความยาวเป็น 3 จึงจะไปยังสเตจที่ 3 ได้ และหากต้องการแยก (distinguish) สเตจ 0 จากสเตจ 1 เราต้องใช้สตริงที่มีความยาวเป็น 3 นั่นคือ สเตจ 0 ไปเป็น 111 เพื่อไปยัง reject state และสเตจ 1 จาก 111 ไปยัง accept state



รูปที่ 3.6 แสดงตัวอย่างของ FSM ที่มีค่าดีกรีที่มากที่สุดในการเข้าถึงและเป็นแบบ Distinguish ability

Theorem 1: พิจารณา minimal FSM ที่มี n สเตจ อักขระที่เป็นอินพุตคือ Σ และเอาต์พุตคือ Ω แล้วดีกรีของ distinguishability (d) จะอยู่ในช่วง [6]

$$\log_{|\Sigma|} \log_{|\Omega|} n - 1 \le d \le n - 1 \qquad (3.2)$$

และดีกรีของ accessibility (a) จะอยู่ในช่วง

$$\log_{|\Sigma|} n - 1 \le a \le n - 1 \qquad (3.3)$$

กำหนด $N(\Sigma, h)$ เป็นจำนวนโหนดในทรีแบบคอมพลีท $|\Sigma|$ –*ary* ที่มีความสูง h จะได้

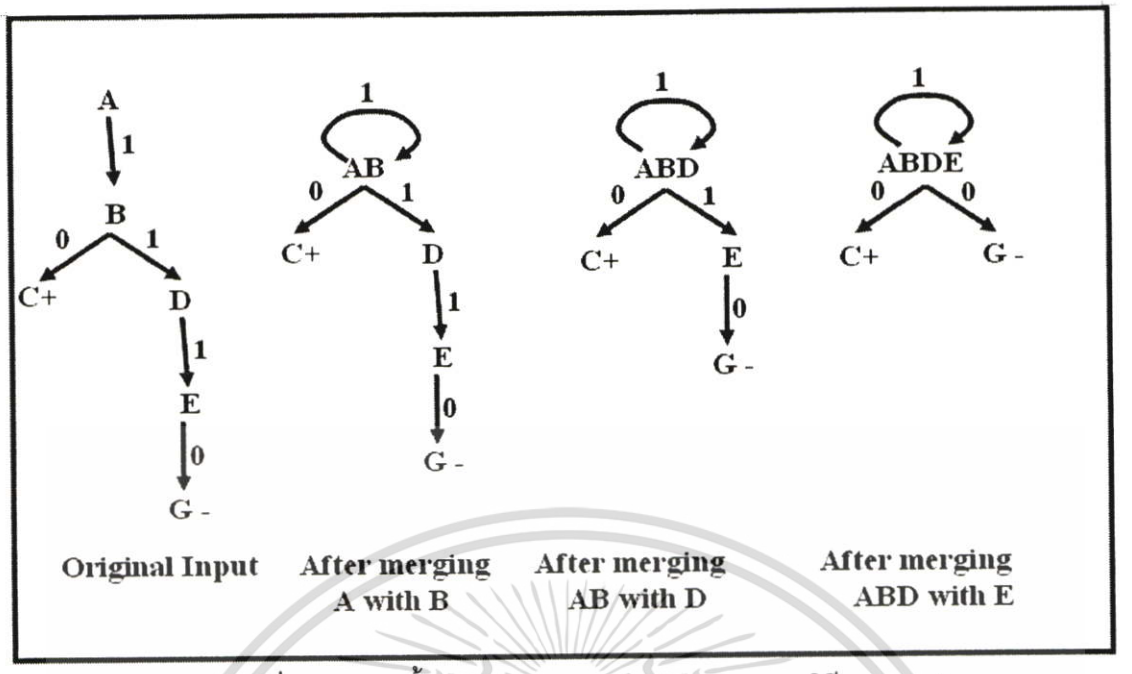
$$N(\Sigma,h) = |\Sigma|^0 + \ldots + |\Sigma|^h = \frac{|\Sigma|^{h+1} - 1}{|\Sigma| - 1} \qquad (3.4)$$

จากตัวอย่างที่มีความซับซ้อน กรณีที่แย่ที่สุดคือ $N(\Sigma, 2n-1) \approx 2^{2n}$ เมื่อ $|\Sigma| = 2$ ช่วงแรกจะใช้เวลานานที่สุดโดยอาจจะใช้เวลาถึง $1 + \ldots + (t-1) = O(t^2)$ ซึ่งเป็นเวลาที่ใช้ในการเปรียบเทียบโดย $t = N(\Sigma, 2n-1)$ เป็นจำนวนของโหนดในทรีและในการตรวจสอบอาจจะใช้ทำถึง $N(\Sigma, n-1) \approx t^{0.5}$ โหนด ดังนั้นเวลารวมทั้งหมดคือ $O(t^{2.5})$ โดยประมาณแล้วคือ 2^{5n} สำหรับตัวอักษรแบบไบนารี่

3.3.4.3 การเรียน DFAs แบบสุ่มจากตัวอย่างแบบ Sparse: อัลกอริทึมแบบ Greedy Russian

Lang [28] นำเสนออัลกอริทึมแบบ Russian ที่แตกต่างออกไปโดยทำการรวม 2 ขั้นตอนจากอัลกอริทึมเดิมในการมองหา (looking) สเตจแล้วหาทรานซิชั่นไว้ในการเดินเพียงครั้งเดียวในทรี เมื่อพบโหนด n_i ที่สมมาตรกับโหนด n_j ซึ่งเป็นโหนดก่อนหน้านี้ หมายความว่าทั้ง 2 โหนดมีสับทรีที่เหมือนกัน ดังนั้นเราจึงสามารถกำหนดให้โหนดแม่ของ n_i ชี้ไปที่ n_j แทนแล้วตัด n_i และสับทรีของมันทิ้งได้เพราะกลุ่มทรีภายใต้ n_i ไม่สามารถเข้าถึงได้อีกต่อไป รูปร่างของออโตเมตอนก็จะไม่เป็นทรีอีกต่อไป เวลาการคำนวณจะเป็น $O(tn^2)$ โดย t คือจำนวนโหนดในทรีและ n คือจำนวนสเตจในแมซชีนสุดท้ายที่มีขนาดเล็กสุด ทั้งนี้เนื่องมาจากการเปรียบเทียบต้องทำกับทุกๆ โหนดทีมีจำนวน t โหนดในออโตเมตอน

Lang [28] นำเสนอวิธีการข้างต้นเพื่อจัดการกับกรณีของทรีที่เป็นอินพุตเป็นทรีที่ไม่สมบูรณ์ (incomplete tree) ในกรณีนี้เลเบลที่หายไปอาจจะขัดแย้งกันระหว่างสับทรีที่มีรูท (root) เป็น n_i และ n_j ซึ่งจะทำให้แมซชีนเกิดข้อผิดพลาดและไม่ใช่แมซชีนที่เล็กที่สุดที่ต้องการ อย่างไรก็ตามเราสามารถใช้วิธีที่เรียกว่า "greedy Russian" แล้วรวมสเตจได้ วิธีนี้จะต้องรู้การเรียงโหนดภายใต้ n_i เพื่อจะเปลี่ยนสับโหนดนี้ไปสู่โหนดที่สมมาตรกับมันซึ่งอยู่ภายใต้ n_j มีข้อควรระวังอีกประเด็นหนึ่งคือสับทรีอาจมีไซเคิล (cycle) หรือโครงสร้างที่ไม่ใช่โครงสร้างแบบทรีซึ่งมันจะไม่สามารถตรวจสอบได้หากใช้การท่องทรีที่พิจารณาเพียง 2 โหนด Lang [28] ได้แก้ปัญหานี้โดยหาทางรวมเสตจที่คิดว่าดีที่สุด พร้อมทั้งทำทำแบ็คอัพอีกชุดสำหรับแต่ละการเปลี่ยนแปลงที่เกิดขึ้นซึ่งหากไม่ใช่การรวมที่ถูกต้องหรือเกิดการขัดแย้งกันอันเนื่องมาจากการรวม ก็สามารถยกเลิกการกระทำนั้นได้ทันที



รูปที่ 3.7 แสดงขั้นตอนของ Greedy Russian ตามวิธีการของ Lang

จากรูป 3.7 พรีฟิกซ์ทรีสำหรับการสอนเซต $10 \rightarrow accept$, $1110 \rightarrow reject$ โดย + แสดง accept state และ – แสดง reject state ขั้นตอนแรกของ Lang [28] คือพยายามที่จะรวมสเตจ A และ B โดยการรวมนี้ไม่ได้ทำให้ไปถึง C และ G พร้อมกันดังนั้นไม่เกิดความขัดแย้งเมื่อรวม A และ B อย่างไรก็ตามเมื่อรวม AB แล้ว D และ E ถูกรวม เข้าด้วยกันแล้วจะทำให้เกิดความขัดแย้งกัน เพราะ C เป็น accept แต่ G เป็น reject ดังแสดงในรูป 3.7 ขวามือสุด ฉะนั้นเพื่อคงผลลัพธ์ของแมชชีนเดิมคือ $10 \rightarrow accept$, $1110 \rightarrow reject$ การรวมสเตจจะทำได้แค่การรวม ABD เข้าด้วยกัน

3.3.4.4 การเรียน DFAs แบบ typical จากการเดินแบบสุ่ม

ออโตเมตอนแบบ "typical" คือออโตเมตอนที่สเตจถูกสุ่มเพื่อเลเบลให้เป็น accept หรือ reject แต่ว่าออโตเมตอนชนิดนี้โทโปโลจีอาจไม่สมารถควบคุมได้เพราะขึ้นกับการสุ่ม Freund [29] จึงขยายนิยามและทฤษฎีข้างต้นใหม่

นิยาม 3.3 : จะกล่าวว่า uniformly almost automata มีคุณสมบัติ $P_{n,\delta}$ ถ้ามีสิ่งเหล่านี้คือ สำหรับทุก $\delta > 0$ (δ คือพารามิเตอร์สำหรับความเชื่อมั่น : confidence parameter) สำหรับทุก $n > 0$ (n คือจำนวนสเตจ) และสำหรับออโตเมตอนแบบกราฟ G_M ใดๆที่มี n โหนด หากเราสุ่มเลือก $\{+,-\}$ เลเบลสำหรับสเตจที่ความน่าจะเป็นอย่างน้อย $1-\delta$ แล้วจะมี $P_{n,\delta}$ สำหรับออโตเมตอน M ที่เรียกว่า "typical" หากโทโปโลจีของออโตเมตอนนั้นๆถูกสุ่มด้วยเซ่นกันเราจะไม่ใช้คำว่า "Uniform" นำหน้า (เนื่องจากคุณสมบัติ P อาจจะไม่ใช่การกระจายที่มีรูปแบบหรือ "spread uniformly" สำหรับออโตเมตอน n สเตจนั้น)

Theorem 2 : สำหรับทุก uniformly almost automata (ที่มีอักขระอินพุตและเอาต์พุตเป็นไบนารี่) ดีกรีของ distinguishability จะมีค่ามากที่สุดเป็น $2\log(n^2/\delta)$ โดยที่ δ คือพารามิเตอร์ของค่าความเชื่อมั่น ดังนั้นสำหรับ $d \geq 2\log(n^2/\delta)$ ของ uniformly almost automata จะมีคุณสมบัติคือ d-signature จะไม่ซ้ำกัน (unique) โดย d-signature ของสเตจคือความลึก d ของสับทรีของมัน [6]

เนื่องจาก signature มีขนาดเล็ก เราสามารถสร้าง signature เพื่อแสดงการเดินภายในกราฟได้ Freund [29] ได้นำเสนอ "robot" ที่รับบิทของอินพุตซึ่งสุ่มให้แล้วทำนายเอาต์พุตของสเตจ i ขณะนั้นว่าเป็น + (accept) หรือ – (reject) เรียกว่า "default mistake" ซึ่งมันจะต้องสังเกตุเอาต์พุตที่ถูกต้องแล้วกลับไปยังสเตจเริ่มต้น และเนื่องจาก signature มีลักษณะที่ไม่ซ้ำกันและไม่เหมือนกัน (unique) ดังนั้น "robot" สามารถรู้จำได้เมื่อมันกลับไปยังสเตจนั้นและแล้วเติมข้อมูลการทรานซิชั่นได้ วิธีการของ [29] มีประสิทธิภาพดีใน uniformly almost n-stage automata และยอมรับจำนวนของ default mistake ที่มากที่สุดที่ $O((n^5/\delta^2)\log(n/\delta))$ อัลกอริทึมนี้สามารถขยายเพื่อนำไปใช้ในการเรียน DPFA โดยการนำจำนวนครั้งที่เกิดจากแต่ละทรานซิชั่น

3.3.4.5 การเรียน DFAs แบบการประมาณโดยใช้อัลกอริทึมแบบ K-tail

Recall ของ q, q' คือค่า k-equivalent หากมันไม่ distinguishable โดยสตริง x ใดๆที่ $|x| \leq K$ ในทำนองเดียวกันหมายความว่ามันมี k-tail ที่เหมือนกัน โดย

$$q \equiv_{k+1} q' \Leftrightarrow \begin{cases} q \equiv_k q' \; and \\ \forall a \in \Sigma : \delta(q,a) \equiv_k \delta(q',a) \end{cases} \tag{3.5}$$

อีกทั้งยังสามารถกล่าวได้ว่าส่วนของสเตจใดๆที่ชักนำมาจากความสมมาตร ($\equiv$) สามารถนำไปสู่ DFA ที่มีขนาดเล็กสุดได้ ซึ่งจะได้กลยุทธ์แบบ iterative สำหรับการลดขนาด DFA ให้เล็กที่สุดที่เรียกว่า Moore's algorithm [19] ดังรูปที่ 3.8

k:=0

compute $\equiv_0$

while $k \le n-2$ and $\equiv_k \neq \equiv_{k-1}$ do

$k := k+1$

compute $\equiv_k$ in term of $\equiv_{k-1}$

merge the k-equivalent states

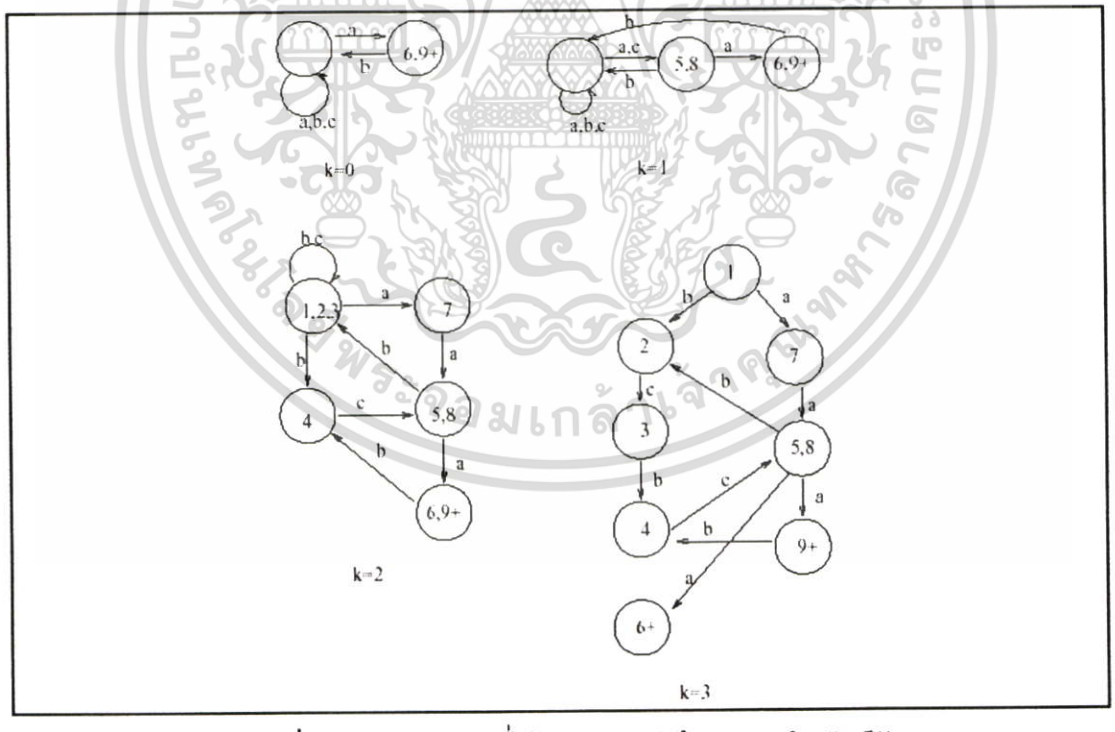
รูปที่ 3.8 แสดง Moore's algorithm

อัลกอริทึมนี้สามารถรันในเวลา $O(|\Sigma|n^2)$ ดังแสดงใน [6] การสร้างอาร์เรย์แบบ $n \times n$ และทำเครื่องหมายที่เซล (i, j) (และ (j,i)) เมื่อพบ q_i และ q_j ที่เป็น distinguishable ซึ่งเริ่มจากการทำเครื่องหมาย (i, j) สำหรับทุก $q_i \in F, q_j \notin F$ แล้วแต่ละคู่ของเสตจที่เป็นคนละสเตจ p,q ถ้า สำหรับบาง a ที่ลูกของสเตจ $r = \delta(p, a)$ และ $s = \delta(p, a)$ เป็น distinguishable โดยบางสตริง x แล้ว p และ q จะเป็น distinguishable โดย ax ดังนั้นจะทำเครื่องหมายที่เซล (p,q) และเป็น dependent มิฉะนั้นแล้ว (p,q) จะถูกใส่ไว้ในรายการของ dependents ที่สัมพันธ์กับ (r,s) entry และหาก (r,s) ได้รับการทำเครื่องหมายในเวลาต่อมาแล้ว (p,q) จะถูกทำเครื่องหมายด้วย ทั้งนี้เพราะแต่ละคู่ (p,q) จะถูกพิจารณามากที่สุด $|\Sigma|$ ครั้ง

อาจกล่าวได้ว่าอัลกอริทึมแบบ k-tail ของ Biermann และ Feldman [30] เป็นเสมือนการรันอัลกอริทึมของ Moore เพียงแต่จะออกจากโปรแกรมที่ส่วนของค่า k ใดๆ หากหยุด ณ ค่าที่ต่ำกว่า k แสดงว่าผลลัพธ์ของแมชชีนจะมีขนาดเล็กแต่ว่ามีความเป็น non-deterministic มาก (ดังนั้นจึงสมมาตรกับ DFA เป้าหมาย) หากหยุดที่ค่า k ที่มากกว่าหรือเท่ากับ n-1 แล้วเราจะได้แมชชีนที่เล็กที่สุดที่ยอมรับเซตของอินพุตนั้นได้ เรียกว่า DAG canonical automata ดังแสดงตัวอย่างในตารางที่ 3.2 เป็น k-tail ของแต่ละสเตจที่มาจาก DAG DFA จากรูปที่ 3.2 ในฟังก์ชั่นของ k โดย λ แสดง empty string และรูปที่ 3.8

ตารางที่ 3.2 k-tail ของแต่ละสเตจใน DAG DFA ในรูปที่ 3.2 ซึ่งเป็นฟังก์ชันของ k

q	k=0	k=1	k=2	k=3	k=4
1	ϕ	ϕ	ϕ	aaa	aaa
2	ϕ	ϕ	ϕ	ϕ	$cbca$
3	ϕ	ϕ	ϕ	bca	bca
4	ϕ	ϕ	ca	ca	ca
5	ϕ	a	a	a	a
6	λ	λ	λ	λ	λ
7	ϕ	ϕ	aa	aa	aa
8	ϕ	a	a	a	$a, abca$
9	λ	λ	λ	λ, bca	λ, bca



รูปที่ 3.9 แสดง DFA ที่ได้จากอัลกอริทึม K-tail ในฟังก์ชันของ K

จากรูปที่ 3.9 ซึ่งเป็น DFA ที่ได้จากการใช้อัลกอริทึม K-tail ในฟังก์ชันของ K เซตของอินพุตที่ใช้คือ $\{bcbca, aaabca, aabcbca, aaa\}$ สำหรับ k=4 อัลกอริทึมนี้คืนกลับ DAG canonical automaton ในรูปที่ 3.2 ที่ใช้ข้อมูลจากตารางที่ 3.2 ในการคำนวณ

3.3.4.6 อัลกอริทึมการจัดกลุ่มแบบ K-tail (K-tail clustering)

ในอัลกอริทึมแบบ K-tail เรานิยามสเตจ 2 สเตจที่จะเป็น k-equivalent ถ้า k-tail ของทั้ง 2 สเตจเท่ากัน เราสามารถพิจารณาค่าความคล้ายกัน (similarity) ระหว่าง k-tail จากนั้นก็ใช้อัลกอริทึมของการจัดกลุ่ม (clustering algorithm) เพื่อจัดกลุ่ม k-tail (โดยพิจารณาในสเปซของมิติ Σ^k) เนื่องจากเราสามารถสร้าง DFA ได้ใหม่จาก k-tail (สำหรับ $k \geq n-2$) นั่นคือแต่ละสเตจมีความแตกต่างกัน (unique) ในการระบุ k-tail ดังนั้นหาก 2 k-tail เหมือนกันแล้ว สเตจทั้ง 2 ก็จะสามารถรวมกันได้ Miclet [31] เรียกวิธีนี้ว่า K-tail clustering algorithm หมายเหตุของการจัดกลุ่มคือ เป็นการรวมสิ่งที่คิดว่าคล้ายกันหรือเหมือนกันเข้าด้วยกัน ซึ่งไม่ได้หมายความว่ามันจะสมมาตรกันด้วย

เราต้องกำหนดว่าการจัดกลุ่มจะเปลี่ยนแปลงไปอย่างไรเมื่อเพิ่มจุดใหม่เข้าไป เพื่อลดเวลา Miclet [31] ได้กำหนด $k(q \cup q') = k(q) \cup k(q')$ โดย $k(q)$ คือ k-tail ของ q ด้วยเหตุที่ความสัมพันธ์ที่ถูกต้องจริงๆคือ $k(q \cup q') \supseteq k(q) \cup k(q')$ (ทั้งนี้เพราะเมื่อเราทำการรวมสเตจจำนวนสตริงที่ยอมรับได้ก็จะมีมากขึ้นด้วย)

กำหนดระยะห่างในการวัดระหว่าง 2 กลุ่ม (clusters) โดยใช้การวัดที่เรียกว่า ad hoc วัดระยะห่างระหว่าง k-tails

$$d_1(k(q),k(q')) \underset{=}{\overset{def}{}} \min\{|k(q)| - |k(q) \cap k(q')|, |k(q')| - |k(q) \cap k(q')|\} \quad (3.6)$$

วัดจำนวนของสตริงที่ไม่มีอยู่ทั่วไปแต่ว่าอยู่ใน "local" ที่อยู่บนพื้นฐานของระยะทาง $d_s(x,x')$ ระหว่าง 2 สตริง

$$d_2(k(q),k(q')) \underset{=}{\overset{def}{}} \max_{x \in k(q)} \sum_{x' \in k(q')} d_s(x,x') \quad (3.7)$$

วัดกรณีที่เลวร้ายที่สุดของจำนวนการเปลี่ยนแปลงที่เกิดขึ้นเพื่อเปลี่ยนสตริง x ใน 1 k-tail เป็นสตริง x' อื่นใน k-tail อื่น

จากนั้นใช้อัลกอริทึมของการจัดกลุ่มแบบลำดับชั้น (hierarchical clustering algorithm) หรืออัลกอริทึมแบบ bottom–up เพื่อทำซ้ำกับจุดที่ใกล้เคียงที่สุด 2 จุด จนกระทั่งไม่มี 2 จุดที่จะเป็น k-tail ได้อีกต่อไป อัลกอริทึมนี้เริ่มที่ k=t-2 โดย t คือจำนวนของโหนดในพรีฟิกซ์ทรีแล้วลดมันลงเรื่อยๆในแต่ละขั้น ซึ่งวิธีการนี้จะมีประสิทธิภาพมากกว่า k-tail ธรรมดาในการทดสอบ DFA ที่เล็กและ ซับซ้อน

3.3.5 วิธีการอื่นๆ

3.3.5.1 การเรียน DFAs แบบ active ด้วย oracles

ในการเรียน DFAs แบบ passive ที่มีอินพุตแบบใดๆ มีงานวิจัยก่อนหน้าที่ทำการเรียนโดยใช้เครื่องมือคืออออร์ราเคิล (oracle) ซึ่งการเรียนสามารถถามคำถามได้ วิธีการเรียนแบบนี้เรียกว่า active learning หรือการเรียนด้วยการ queries ตัวอย่างเช่น [23]

ในโมเดลของออร์ราเคิล สมมติว่า DFA เป้าหมายต้องรีเทิร์นกลับยังรีเซต (reset) สเตจก่อนที่จะตอบคำถามว่า x อยู่ในภาษาของมันหรือไม่ ถ้าไม่จะเรียกว่า no-reset model การเรียนก็จะต้องยากขึ้นไปอีกทั้งนี้เพราะการเรียนเริ่มต้นที่สเตจใดๆ อย่างไรก็ตาม [12] ได้แสดงให้เห็นว่าเราสามารถจัดการกับกรณีที่เกิดขึ้นนี้ได้โดยการใช้ "homing sequence" ในการไกด์นำทาง ในงานวิจัยของ [29] ได้แสดงให้เห็นว่าเราสามารถประยุกต์ใช้วิธีการเหล่านี้กับการเรียน typical DFAs จากการสุ่มเดินได้

3.3.5.2 วิธีการใช้ Neural Network

ได้มีงานวิจัยของ Minsky [31] แสดงให้เห็นว่า ทุก FSM สามารถสร้างให้สมมาตรได้โดยใช้นิวรอลเน็ตเวิร์ก (Neural Network:NN) แต่อย่างไรก็ตามไม่ได้หมายความว่าการสร้างโดยวิธีนี้จะเป็นวิธีที่ง่ายในการเรียนโครงสร้างที่ไม่ทราบของ FSM ปัญหาหลักก็คือ FSM เป็นแบบไม่ต่อเนื่อง (discrete) แต่ว่า NN เป็นโมเดลแบบต่อเนื่อง (continuous) ดังนั้นการเรียนด้วยวิธีนี้จึงไม่ค่อยเป็นที่นิยมเท่าใดนัก อีกทั้งยังเสียเวลาและยุ่งยากในการทดสอบอีกด้วย

3.4 Learning NPFAs (HMMs)

3.4.1 การใช้อัลกอริทึมแบบ Baum-Welch

วิธีการที่เป็นที่นิยมอย่างแพร่หลายในการเรียนโทโปลีจีของ HMM คือการสร้างกราฟที่เชื่อมต่อกันแบบ fully connected graph (clique) ที่มี n โหนด (โดย n คือ ขอบเขตที่มากที่สุดของจำนวนสเตจในโมเดลที่เราพยายามเรียน) จากนั้นใช้อัลกอริทึมของ Baum-Welch ในการสอนโดย

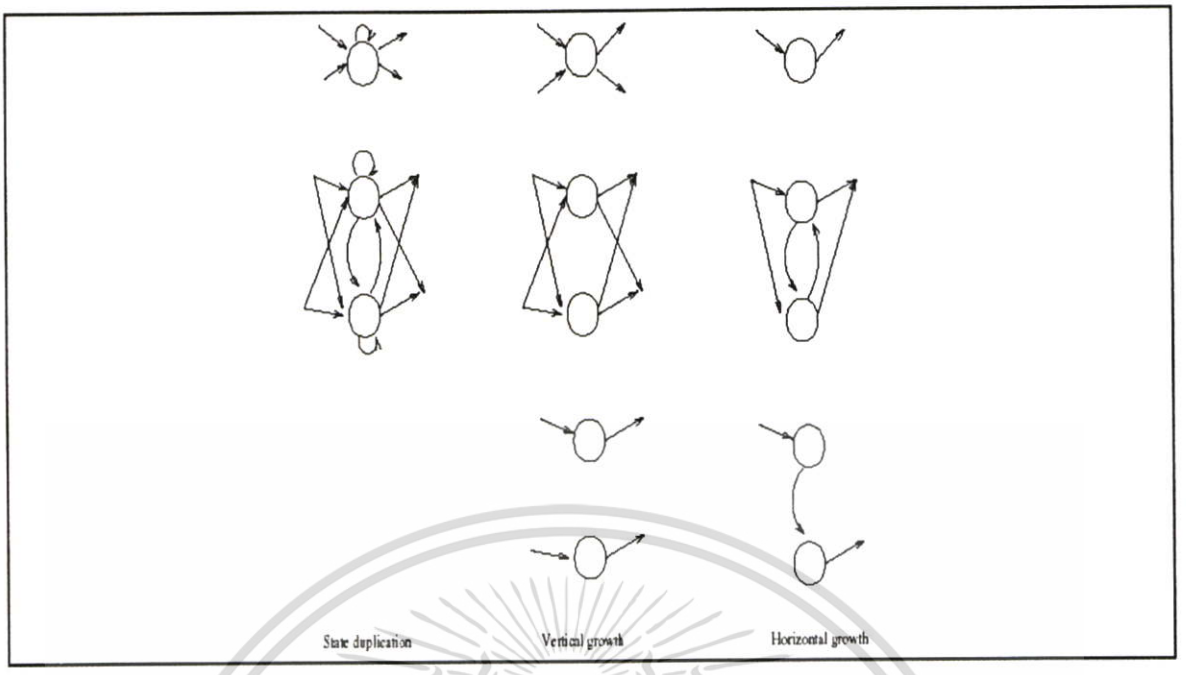
อัลกอริทึมจะกำหนดค่าของน้ำหนักที่เป็น 0 ให้กับเส้นทางที่ไม่สำคัญ วิธีการนี้มีข้อเสีย 2 ข้อคือ อย่างแรก n มักจะเป็นสิ่งที่ไม่ทราบค่ามาก่อน แต่ว่าข้อนี้ก็ไม่ใช่ปัญหาหลักเพราะเราสามารถลองเพิ่มค่า n จาก n=1, n=2,... ได้เรื่อยๆ จนกว่าคิดว่าผลลัพธ์น่าจะยอมรับได้แล้วจึงหยุด ปัญหาที่ 2ซึ่งเป็นปัญหาที่หนักกว่าคือกราฟแบบ fully connected นี้มีเส้นทางทั้งหมด $O(n^2)$ เส้นทางซึ่งเป็นพารามิเตอร์ที่มีค่าใหญ่พอสมควรในการจัดการและอาจมีเส้นทางที่ซ้ำซ้อนเกิดขึ้นด้วย โมเดลที่มีค่าพารามิเตอร์ใหญ่ๆแบบนี้ต้องใช้เวลาในการสอนนานและต้องใช้ข้อมูลมากด้วย

3.4.2 Iterative state duplication

พิจารณาวิธีการแบบ "ad hoc" ที่ใช้ในอัลกอริทึมของ Baum-Welch ซึ่งจะสร้างลูปภายในหรือ "inner loop" ในงานวิจัยของ [34] ได้ใช้ HMM ในการหารูปแบบทั่วไปของโปรตีนที่เรียกว่า motifs วิธีการเรียนโทโปโลจีของ [34] เริ่มจากกราฟแบบ fully connected บน n โหนด จากนั้นทำการสอนโดยใช้อัลกอริทึมแบบ Baum-Welch คือตัด (prune) ทรานซิชันที่มีความน่าจะเป็นที่คาดว่าไม่สำคัญออก (ป้องกันไม่ให้โมเดลใหญ่เกินไป) จากนั้นทำการหาโหนด (เรียกว่า q) ที่มีจำนวนเส้นทางที่เข้ามายังโหนดนั้นและออกจากโหนดนั้นมากที่สุด (อาจเลือกแบบสุ่มเอาหากโมเดลซับซ้อนมาก) แล้วทำสำเนา q จากนั้นวนทำซ้ำจนกว่าจะได้ค่าความเที่ยงตรงที่ถูกต้องเพียงพอ โดยสัญชาตญาณแล้วส่วนที่มีความหนาแน่นที่สุดมักจะเชื่อมโหนด 2 โหนดที่แต่ต่างกันอย่างสิ้นเชิง ดังนั้นจึงทำสำเนาเพื่อลองสอนและตัดทางเชื่อมที่ไม่สำคัญออก ดังรูปที่ 3.10 ซึ่งมีการทำสำเนาสเตจจากหนึ่งสเตจที่มีเส้นทางผ่านเข้าออกทั้งหมด 5 เส้นทาง การทำสำเนาจะต้องทำเส้นทางทั้งแนวนอน (Horizontal growth) และแนวตั้ง (Vertical growth) และเนื่องจากวิธีการของ Baum-Welch ติดความยุ่งยากที่ local optima ดังนั้น [34] จึงแก้โดยจะเริ่มทำซ้ำใหม่ทั้งหมดเมื่อทำการสุ่มครั้งใหม่

3.4.3 Model surgery

อีกวิธีการหนึ่งคือการสร้างโทโปโลจีด้วยมือแล้วทำ "fine tune" หลังจากสอนโทโปโลจีแล้ว ตัวอย่างวิธีการนี้คือโมเดลที่ใช้ในการผ่าตัด [16] ซึ่งสร้างโมเดลให้กับลำดับของโปรตีน วิธีการนี้เริ่มต้นโดยมีโมเดลสายหลักหรือ "backbone" ของ n สเตจ แต่มีทรานซิชันวนกลับเข้าหาตัวเองและทรานซิชันข้ามไปยังสเตจอื่น ในขั้นตอนการสอนเมื่อเจอลำดับโปรตีนที่ต้องข้ามไปก็จะทำการสร้างเส้นทางเพื่อข้ามไปยังโหนดถัดไป แต่หากว่าต้องเพิ่มโหนดเข้าไป โมเดลสายหลักก็จะมีโหนดเพิ่มขึ้น การสอนกระทำไปจนกระทั่งได้โมเดลที่ยอมรับว่าเหมาะสม.



รูปที่ 3.10 แสดงขั้นตอนการจำลองสเตจ

3.4.4 การใช้การรวมสเตจแบบ Bayesian state-merging

วิธีการที่เป็นหลักการและเป็นที่รู้จักในการเรียน HMM ได้นำเสนอใน [7][8] โดยการเรียน PFAs นั้นปัญหาสำคัญคือเรื่องการรวมสเตจ โดย [7][8]ได้ทำการวนทำซ้ำในขั้นตอนการรวมสเตจในพรีฟิกซ์ทรี (นั่นคือ maximum likelihood ของ HMM) แต่ว่าไม่ได้ทำการวนซ้ำในการรวมโหนดลูกของทั้ง 2 สเตจที่นำมารวมกัน ดังนั้นโมเดลที่ได้จึงกลายเป็น non-deterministic นั่นหมายความว่าเราไม่สามารถหาค่าความคล้ายคลึงกันของทั้ง 2 สเตจโดยใช้สับทรีของมันได้ เพราะมันจะมีหลายเส้นทางที่มีเลเบลเดียวกันภายใต้แต่ละโหนด แต่จะใช้การพิจารณาคู่สเตจแทนแล้วคำนวณหาค่า posterior probability ของโมเดลเมื่อทำการรวมคู่สเตจดังกล่าว คู่การรวมสเตจที่ทำให้ค่า posterior probability ของโมเดลหลังการรวมมีค่ามากที่สุดจะถูกเลือก (ดังนั้นวิธีการนี้จึงเป็น greedy หรืออัลกอริทึมแบบ best-first ซึ่งอาจทำให้ติดปัญหาที่ local optima ได้) จากนั้นก็จะทำการรวมคู่สเตจไปเรื่อยๆจนค่า posterior probability ไม่เพิ่มอีกต่อไป

ในงานวิจัยของ stolcke [7][8] ยังได้ใช้ค่าความน่าจะเป็นของการทรานซิชันก่อนหน้าและการกระจายก่อนหน้ามาพิจารณาด้วย ซึ่งเป็นแนวทางที่ดีในการจัดการกับเซตของการสอนที่มีขนาดเล็ก เพราะค่าความน่าจะเป็นเหล่านี้เป็น multinomial [7][8] ยังได้แสดงให้เห็นว่าวิธีการ Bayesian state-merging นี้สามารถให้ผลลัพธ์ที่ดีในการรวมสเตจในอัลกอริทึมแบบ Baum-Welch อีกด้วย

บทที่ 4

การเรียนไวยากรณ์จากกลุ่มตัวอย่าง

4.1 บทนำ

วิธีการพิสูจน์ไวยากรณ์โดยการยกกรณีตัวอย่างซึ่งเรียกว่า "Grammar Induction" หมายถึง การเรียนรู้ไวยากรณ์ (Grammar Learning) สำหรับภาษาหรือรูปแบบที่มีแบบแผน (formal) จากเซตของตัวอย่าง ซึ่งการเรียนแกรมม่าได้ถูกนำมาประยุกต์ใช้ในงานต่างๆมากมาย เช่น ในกระบวนการทางภาษาธรรมชาติ (Natural Language Processing) และการดึงข้อมูล (Information Retrieval) โดยคอนเท็กฟรีแกรมม่ามักจะถูกนำไปประยุกต์ใช้เนื่องจากสามารถจับโครงสร้างทางธรรมชาติและความฉลาดทางภาษาได้

จากการศึกษาการพิสูจน์ไวยากรณ์ในงานวิจัยของ Stolcke Andreas[7][8] ซึ่งเป็นการพิสูจน์ไวยากรณ์ที่ใช้วิธีการแบบ Bayesian model merging framework เพื่อเรียนไวยากรณ์ โดยในงานวิจัยมีแนวทางที่น่าสนใจ คือ การประยุกต์ใช้ไวยากรณ์ที่เรียกว่า สต๊อกเซสติกคอนเท็กฟรี-แกรมม่า (Strochastic Context Free Grammar, SCFG), ซึ่งเป็นการสร้างแกรมม่าที่นำเอาค่าความน่าจะเป็นที่กฎภายในแกรมม่าจะถูกใช้ในการดีไรว์มาช่วยทำให้ประสิทธิภาพการดีไรว์ดียิ่งขึ้น ในงานด้าน computational linguistics นั้นก็มีการนำคอนเท็กฟรีแกรมม่ามาใช้อย่างกว้างขวาง แต่ในงานด้านการรู้จำ ยังไม่ค่อยมีการประยุกต์ใช้งานด้านนี้มากนัก

จากงานวิจัยก่อนหน้านี้ได้แสดงให้เห็นว่า SCFG สามารถใช้ประโยชน์ได้ดีในด้านการรู้จำ หากนำไปประยุกต์ใช้กับงานที่เหมาะสม เช่น Jurafsky [9] นำ SCFG มาใช้ในงานด้าน speech recognition แสดงให้เห็นว่า SCFG ที่สร้างมาจาก hand-crafted rules ที่มีการกำหนดค่าความน่าจะเป็นให้ โดยประมาณมาจากคอปัส (corpus) สามารถเพิ่มประสิทธิภาพการรู้จำให้กับโมเดลทางภาษาที่เรียกว่า n-gram ได้

งานวิจัยนี้ได้ทดลองนำเอาโมเดล SCFG ที่เสนอโดย Stolcke Andreas [7][8] ไปประยุกต์ใช้กับการเรียนไวยากรณ์สำหรับเซนโค้ดของตัวอักษรเขียน โดยในบทนี้จะกล่าวถึงการทดลองและแกรมม่าที่สร้างได้จากการเรียนโมเดลแบบ SCFG

4.2 Stochastic Context-free Grammars

Definition 1 Stochastic context free grammar (SCFG) M ประกอบด้วย

1. เซตของ nonterminal symbols, N
2. เซตของ terminal symbols หรือ alphabet Σ
3. start nonterminal $S \in N$
4. เซตของโปรดักชั่นหรือกฎ R
5. ค่าความน่าจะเป็นของแต่ละโปรดักชั่น P(r) สำหรับทุก $r \in R$

แต่ละโปรดักชั่นอยู่ในรูปแบบ

$$X \rightarrow \lambda$$

โดยที่ $X \in N$ และ $\lambda \in (N \cup \Sigma)^*$,

X เรียกว่า left-hand side (LHS)

λ เรียกว่า right-hand side (RHS)

หมายเหตุ : ตัวอักษรพิมพ์ใหญ่จะแสดงถึง nonterminal symbol และตัวอักษรพิมพ์เล็กจะแสดงถึง terminal symbols ส่วนสตริง (string) ที่เขียนผสมกันระหว่าง nonterminal และ terminal symbol จะแทนด้วยอักขระ λ, μ และ ν และ empty string จะแทนด้วย ε.

SCFG จะคล้ายคอนเท็กฟรีแกรมม่าทั่วๆไปแต่จะมีส่วนที่แตกต่างคือแต่ละโปรดักชันจะมีการกำหนดพารามิเตอร์ที่ต่อเนื่องกันเรียกว่าพารามิเตอร์ของค่าความน่าจะเป็น (probability parameters) ให้การแปลความของ $P(X \rightarrow \lambda)$ คือจะพิจารณาการดีไรว์ (Derivation) จากบนลงล่างของ SCFG โดย RHS จะถูกเลือกเพื่อบ่งชี้ค่าความน่าจะเป็นของการกระจาย nonterminal X นั่นคือ $P(X \rightarrow \lambda)$ คือสภาวะความน่าจะเป็นของ X ดังนี้

$$\sum_{\lambda} P(X \rightarrow \lambda) = 1 \qquad (4.1)$$

โดยผลรวมทั้งหมดที่ได้มาจากโปรดักชันทั้งหมดที่มี LHS เป็น X

Definition 2 a) เซนเทนเชียลฟอร์ม (sentential form) ของ M คือ สตริง (string) ν ของนอนเทอร์มินอลและเทอร์มินอล โดยที่ $\nu = S$ หรือมีประโยคที่อยู่ในรูป μ จากที่ ν สามารถถูกสร้างโดยการแทนที่นอนเทอร์มินอลหนึ่งตัวโดยการแทนนี้จะขึ้นกับโปรดักชันของ M นั่นคือ $\mu = \mu_1 X \mu_2$, $\nu = \mu_1 \lambda \mu_2$ และ $X \rightarrow \lambda \in R$

b) (left-most) derivation ใน M คือ ลำดับของรูปแบบของประโยคที่ขึ้นต้นด้วย S โดยแต่ละประโยคจะถูกดีไรว์ (derive) โดยแอพลิเคชันแบบหนึ่งกฎ (single rule) จากกฎก่อนหน้านั้น นั้น

คือแต่ละขั้นในการแทนที่นอนเทอร์มินอล X จะทำที่นอนเทอร์มินอลที่อยู่ left-most ในรูปประโยคนั้นนั่นคือ $\mu_1 \in \Sigma^*$ ดังนิยามข้างต้น เราจะสามารถเขียนการดีไรว์ดังนี้

$s \Rightarrow \nu_2 \Rightarrow \ldots \nu_k$ โดยที่ $\nu_2 \ldots \nu_k$ คือรูปประโยคที่เกี่ยวข้อง

c) พลอบบาบิลิตี้ของการดีไรว์ $s \Rightarrow \nu_2 \Rightarrow \ldots \nu_k$ ได้ถูกนิยามไว้ดังนี้

1) $P(s) = 1$

2) $P(S \ldots \Rightarrow \nu_k) = P(S \ldots \Rightarrow \nu_{k-1}) P(X \rightarrow \lambda)$

โดยที่ $X \rightarrow \lambda$ คือโปรดักชันที่ใช้ในขั้นตอนการทำ $\nu_{k-1} \Rightarrow \nu_k$

d) ค่าความน่าจะเป็นของสตริง x ใน M คือ

$$P(x \mid M) = \sum_{S \Rightarrow \ldots \Rightarrow x} P(S \Rightarrow \ldots. \Rightarrow x) \qquad (4.2)$$

ค่าของผลรวมของความน่าจะเป็นทั้งหมดนี้มาจากการดีไรว์ที่จบลงด้วย x หรือให้ผลลัพธ์เป็น x

จากนิยามจะเห็นได้ว่าการใช้ SCFG สามารถสร้างพาร์สทรีได้เหมือน CFG ทั่วไปแต่จะมีส่วนของค่าความน่าจะเป็นของกฎในการมีส่วนร่วมเพื่อสร้างพาร์สทรีนั้นๆ และจากนิยามจะเห็นได้ว่า SCFG มีความคล้ายคลึงกับ HMM (Hidden Markov Model) [8] แต่ว่า SCFG อาจจะมีความซับซ้อนในการสร้างแกรมม่ามากกว่าก็ตาม แต่จากวิธีนี้จะสามารถช่วยจัดการในเรื่องความกำกวมของภาษาได้ดีกว่า

4.3 การสร้าง SCFG จากกลุ่มตัวอย่าง

จากงานวิจัยของ Stolcke Andreas [7][8] ที่นำเสนอการเรียนแกรมม่าจากกลุ่มตัวอย่างโดยใช้ SCFG ซึ่งก็คือ คอนเท็กฟรีแกรมม่าที่ได้ทำการกำหนดค่าความน่าจะเป็นให้กับแต่ละกฎในแกรมม่า

หลักการที่มีอยู่เดิม

ในการสร้าง SCFG กระบวนการรวมกฎหรือที่เรียกว่า merging มีองค์ประกอบหลักๆดังนี้

1. วิธีการสร้างโมเดลเริ่มต้น (initial model) จากข้อมูลที่มี
2. วิธีการรวมโมเดลย่อยๆเข้าด้วยกัน

3. การวัดความผิดพลาด (error measurement) เพื่อเปรียบเทียบคุณภาพการรวมโมเดลในแต่ละแบบที่ได้เป็นการกำหนดขอบเขตให้การรวมนั้นยังคงครอบคลุมเป้าหมายของไวยากรณ์
4. กลยุทธ์ในการเลือกวิธีการรวม นั่นคือกลยุทธ์ในการหา model space เพื่อทำการรวม

4.3.1 การกำหนดกฎเริ่มต้น

ตัวอย่างการกำหนดกฎดริ่มต้น สมมติตัวอย่างของสตริงคือ $a_1a_2a_3...a_l$ ทำการรวมสตริงตัวอย่างให้เป็นไวยากรณ์ที่ใช้รู้จำ สามารถสร้างกฎเริ่มต้นคือ $S \longrightarrow a_1a_2a_3...a_l$ เพื่อรักษารูปแบบของไวยากรณ์ตามนิยามของ SCFG เราจะกำหนดนอนเทอร์มินอลใหม่ $X_1X_2X_3...X_l$ แล้วเพิ่มโปรดักชันนี้เข้าไปให้กับไวยากรณ์

$$S \longrightarrow X_1X_2X_3... X_l$$
$$X_1 \longrightarrow a_1$$
$$X_2 \longrightarrow a_2$$
$$X_3 \longrightarrow a_3$$
$$\cdots\cdots$$
$$X_l \longrightarrow a_l$$

โดย S คือสตริงเริ่มต้น (Starting symbol) ของไวยากรณ์นี้

4.3.2 การรวมนอนเทอร์มินอล (Nonterminal Merging)

ขั้นตอนการรวม 2 นอนทอร์มินอล X_1 และ X_2 ให้เป็น 1 นอนเทอร์มินอล Y สามารถแสดงโดย

$$\text{merge}(X_1 , X_2) = Y \qquad (4.3)$$

โดยการรวมนี้จะมี 2 กรณีคือ

1. RHS ที่มี X_1 และ X_2 เกิดขึ้นพร้อมๆกันจึงสามารถรวมและแทนที่ด้วย Y

$$Z_1 \longrightarrow \lambda_1 X_1 \mu_1 \qquad (c_1)$$
$$Z_2 \longrightarrow \lambda_2 X_2 \mu_2 \qquad (c_2)$$
$$\Downarrow \quad \text{merge}(X_1 , X_2) = Y$$
$$Z_1 \longrightarrow \lambda_1 Y \mu_1 \qquad (c_1)$$
$$Z_2 \longrightarrow \lambda_2 Y \mu_2 \qquad (c_2)$$

หมายเหตุ: การรวมกันนี้สามารถนำไปสู่การรวมกันทั้งโปรดักชั่นถ้า $Z_1 = Z_2$, $\lambda_1 = \lambda_2$ และ $\mu_1 = \mu_2$ ซึ่งในกรณีนี้ค่าการนับจำนวนของกฎหลังการรวมจะเป็น $c_1 + c_2$

2. โปรดักชั่น 2 อันที่มี LHSs เป็น X_1 และ X_2 สามารถรวมกันแล้วแทนด้วยนอนเทอร์มินอล Y

$$X_1 \rightarrow \lambda_1 \qquad (c_1)$$

$$X_2 \rightarrow \lambda_2 \qquad (c_2)$$

$$\Downarrow \quad \text{merge}(X_1 , X_2) = Y$$

$$Y \rightarrow \lambda_1 \qquad (c_1)$$

$$Y \rightarrow \lambda_2 \qquad (c_2)$$

สมมติว่า $\lambda_1 = \lambda_2$ การรวมกันแบบ LHS ข้างต้นสามารถกลายเป็น 1 โปรดักชั่นได้ โดยค่าจำนวนการนับของกฎที่ได้หลังการรวมจะเป็น $c_1 + c_2$

ในทำนองเดียวกันกับการรวม state ใน HMM ที่สามารถเกิดการวนทำการรวมภายในโปรดักชั่นซ้ำก็สามารถทำได้ เช่นกรณีที่มี X_1 และ X_2 อยู่ใน LHS และ RHS ของโปรดักชันเดียวกัน

$$X_1 \rightarrow \lambda\, X_2 \mu$$

$$\Downarrow \quad \text{merge}(X_1 , X_2) = Y$$

$$Y \rightarrow \lambda Y \mu$$

4.3.3 การจัดกลุ่มนอนเทอร์มินอล (nonterminal chunking)

เป็นขั้นตอนที่ต้องอาศัยวิธีการรวมนอนเทอร์มินอลมาใช้ด้วย โดยทำการรวมกลุ่มนอนเทอร์มินอลภายในกฎเดียวกัน จากลำดับของนอนเทอร์มินอล $X_1X_2X_3\ldots X_k$ ในกฎที่มี สร้างนอนเทอร์มินอล Y ใหม่มา โดย Y กระจายได้ $X_1X_2X_3\ldots X_k$ เราจะแทนที่ลำดับของสตริงทั้งหมดด้วยนอนเทอร์มินอล Y ซึ่งขั้นตอนที่กล่าวมาทั้งหมดนี้เรียกว่า "Chunking operation" เขียนแทนด้วย $\text{chunk}(X_1X_2X_3\ldots X_k) = Y$.

$$Z \rightarrow \lambda\, x_1x_2x_3\ldots x_k\, \mu \qquad (c)$$

$$\Downarrow \quad \text{chunk}(X_1X_2X_3\ldots X_k) = Y$$

$$Z \rightarrow \lambda\, Y\, \mu \qquad (c)$$

$$Y \rightarrow X_1X_2X_3\ldots X_k \qquad (c')$$

ค่าการนับกฎ c' ของโปรดักชั่นใหม่คือผลรวมของทุกโปรดักชั่นที่มีการแทนที่ด้วย Y เกิดขึ้น

เราสามารถสร้างไวยากรณ์ใหม่ได้จากวิธีการ merging และ chunking nonterminal จากล่างขึ้นบน (bottom-up) ซึ่งจะได้กฎใหม่ๆในไวยากรณ์ แต่การทำวิธีการข้างต้นจะต้อง

พิจารณาถึงค่าความผิดพลาดระหว่างไวยากรณ์ใหม่ที่ได้ด้วยกันเองกับไวยากรณ์เดิมก่อนการเปลี่ยนแปลงด้วย

วิธีการเรียนรู้จากกลุ่มตัวอย่างแบบเดิมที่นำเสนอมานั้นในขั้นตอนการกำหนดกฎเริ่มต้น หากเรามีสตริงของเซนโค้ดยาว จะทำให้ทางด้านขวาของกฎ (Right hand side) มีความยาวมาก ทำให้เสียเวลาในการ merging และ chunking แกรมม่าที่ได้ยังคงซับซ้อนอยู่และยากต่อการเข้าใจ ทั้งยังไม่สามารถรับประกันได้ว่าหลังจากการทำงานทั้ง 2 แล้วแกรมม่าที่ได้จะเป็นแกรมม่าที่มีความโดยรวมทั่วไป (generalize) สำหรับตัวอักษรนั้นๆด้วย

จุดอ่อนของวิธีการสร้างแกรมม่าของ Stolcke, A. [7][8] คือ

1. วิธีการสร้างแกรมม่าเริ่มต้นของงานวิจัยที่ได้นำเสนอจะทำให้ได้แกรมม่าที่มีลำดับทางขวาของกฎที่ยาว
2. จำนวนกฎในแกรมม่าเริ่มต้นนั้นจะขึ้นกับจำนวนตัวอย่างที่ใช้สร้างแกรมม่าด้วย เช่น หากมีชุดสตริงตัวสอนทั้งหมด 100 ตัว แกรมม่าเริ่มต้นจะมีทั้งหมด 100 กฎ
3. เนื่องจากด้านขวาของกฎยาวทำให้การใช้วิธีการ merging และ Chunking ใช้เวลานาน
4. การปรับปรุงแกมม่าเริ่มต้นให้มีความเหมาะสม (Optimization) ไม่สามารถรับประกันได้ว่าจะได้แกรมม่าที่ง่ายและเป็นแกรมม่าที่เหมาะสมจริงๆ
5. วิธีการสร้างแกรมม่าข้างต้นไม่เหมาะกับงานประยุกต์ทางด้านการรู้จำเนื่องจากตัวอย่างเซนโค้ดของตัวอักษรที่ใช้มีความยาวมาก

4.4 ผลการสร้างแกรมม่า

จากการเรียนไวยากรณ์ที่ใช้วิธีการสร้างตามแบบของ Stolcke Andreas [7][8] โดยการทดลองได้ใช้ตัวอย่างของเซนโค้ดตัวอักษรจำนวน 100 สตริง ซึ่งสตริงของเซนโค้ดเหล่านี้เป็นเซนโค้ดที่ทำการปรับปรุงและลด noise แล้ว การสร้างแกรมม่าที่ทำการทดลอง ผลการทดลองใช้กลุ่มตัวอย่างเซนโค้ดของแต่ละตัวอักษรจำนวนอย่างละ 100 เซนโค้ดโดยแต่ละสายของเซนโค้ดมีสตริงทั้งหมด 100 ตัว แกรมม่าเริ่มต้นที่สร้างจากกลุ่มตัวอย่างนี้เริ่มจาก start symbol "S0" ดีไรว์ไปยัง RHS ที่เป็นสายสตริงของเซนโค้ดจำนวน 100 สาย จากหลักการสร้างแกรมม่าข้างต้นแกรมม่าที่ได้จึงมีกฎในแกรมม่าเริ่มต้นทั้งหมด 100 กฎ ซึ่งผลลัพธ์การสร้างแกรมม่านี้ไม่เป็นที่น่าพอใจ เนื่องจากลำดับนอนเทอร์มินอลใน RHS ของกฎและกฎในแกรมม่ามีจำนวนมาก ทำให้การรวมกฎใช้เวลานานเวลาที่ใช้ในการทดสอบหาทางเลือกที่ดีที่สุดในการรวมกฎใช้เวลา $O(n^n)$.

ตัวอย่างกฎของแกรมม่าเริ่มต้นที่ได้จากอัลกอริทึมของ STOLCKE , A. [6]

S → CCCBBCCCCCCCCBCBBBCCCBBBBCCBBBCEEEEEECCCCCBAAAAAAAAA

AAAAAAAHAAAAGGGGGHHGGGGGGGGGGGGGGGGGGGGGGGGGC

S → CCCCCBCBBBCCBBBCCCBBCCCCCCCCCBEEEEEECECCCCCBBAAAAAAAA

AAAAAAAAHHHHHHHAGGGGHGGAHHGGGGGGGGGGGGGGGGGGGC

S → CCCCCCCCCBBCCCCCCCBCCCCCCCCCCCBCEEEEECCCCAAAABAAAAAA

AAAAAAAAAHHAAAGAGAGGGGGGGGHHGGGGGGGGGGGGGGGGGGG

A → 1

B → 2

C → 3

D → 4

E → 5

F → 6

G → 7

H → 8

จากตัวอย่างแกรมม่าเริ่มต้นที่เริ่มจาก start symbol "S" ดีไรว์ไปยังนอนเทอร์มินอลด้าน RHS ทั้ง 3 กฎที่แสดงเป็นตัวอย่างนี้ เป็นส่วนหนึ่งที่มาจากแกรมม่าเริ่มต้นที่สร้างจากเซนโค้ดจำนวณ 100 สาย แต่ในตัวอย่างนี้นำมาแสดงเพียง 3 สายเท่านั้น แกรมม่าเริ่มต้นที่ได้จะเริ่มจาก start symbol S ด้านซ้ายมือที่ดีไรว์ไปยัง RHS ซึ่งเป็นนอนเทอร์มินอลที่ยาวเท่ากับ 100 ตัว ตามสตริงของเซนโค้ด ทำให้รูปแบบการ merge และ chunk นอนเทอร์มินอลภายในกฎเหล่านี้ใช้เวลานานในการค้นหาเพื่อให้ได้แกรมม่าที่เหมาะสม และจากการทดลองวิธีการนี้หลังจากใช้เวลารันโปรแกรมทั้งหมด 20 ชั่วโมงสำหรับตัวอักษรหนึ่งตัว การทดลองรวมแกรมม่าที่ได้ยังคงไม่มีแนวโน้มที่จะนำไปสู่แกรมม่าที่คาดว่าจะได้สำหรับโมเดลการรู้จำ ดังนั้นวิธีการดังกล่าวจึงไม่เหมาะกับงานด้านการรู้จำตัวอักษร

บทที่ 5

การสร้างโมเดลการรู้จำด้วยคอนเท็กฟรีแกรมม่า

5.1 บทนำ

ในการเขียนตัวอักษรของมนุษย์ ถึงม้าจะเขียนตัวอักษรเดียวกันแต่เราก็สามารถรู้ว่ามาจากลายมือผู้เขียนคนละคนกัน ในวิทยานิพนธ์นี้เราจะใช้ทิศทางของตัวอักษรมาเป็นลักษณะที่ใช้ในการสร้างโมเดลแบบไดนามิกสำหรับการรู้จำตัวอักษร โดยวิธีการที่นำเสนอคือกระบวนการเรียนแกรมม่าหรือที่เรียกว่า Grammar Induction จากกลุ่มตัวอย่าง

การพิสูจน์ไวยากรณ์โดยการยกกรณีตัวอย่าง (Grammatical Induction) คืองานเกี่ยวกับการเรียนรู้ไวยากรณ์ (learning a grammar) สำหรับภาษาที่มีแบบแผน (formal language) จากเซตของตัวอย่างประโยค ซึ่งได้ถูกนำมาทำเป็นแอพพลิเคชันอย่างจริงจังในเรื่องของการรู้จำโครงสร้าง (structural pattern recognition), เรื่องกระบวนการทางภาษาธรรมชาติ (natural language processing), เรื่องการดึงข้อมูล (information retrieval) โดยคอนเท็กฟรีแกรมม่ามักถูกใช้เพราะมันสามารถจับโครงสร้างทางธรรมชาติและความฉลาดทางภาษาได้ [35]

โมเดลทางด้านของความน่าจะเป็นได้ถูกนำมาใช้กันอย่างกว้างขว้างและยิ่งมีความสำคัญมากในงานทางด้านการรู้จำตัวอักษร โมเดลต่างๆถูกคิดค้นและนำมาวิจัย ตัวอย่างเช่น วิธีการกำหนดน้ำหนักการรู้จำ ในการสอนและการทดลองจะมีวิธีการปรับค่าของน้ำหนักในการรู้จำตัวอักษร อาทิเช่น Hidden Markov Models (HMMs), Genetic Algorithm และ Neural Network เป็นต้น วิธีการเหล่านี้เป็นการประยุกต์ใช้ปัญญาประดิษฐ์ สร้างโมเดลทางด้านของความน่าจะเป็นสามารถแบ่งลักษณะรวมเป็น 2 ส่วนใหญ่ๆคือ discrete structure (เช่น ส่วนที่เป็นโทโปโลจีของ HMM, คอนเท็กฟรีกรมม่าหลักของ SCFG) และเซตของพารามิเตอร์ที่ต่อเนื่องซึ่งใช้ตัดสินความน่าจะเป็นของสตริงที่ต้องการรู้จำ แต่ยังไม่ปรากฏงานวิจัยใดที่นำไวยากรณ์หรือแกรมม่ามาช่วยสร้างโมเดลสำหรับการรู้จำลายมือเขียน การใช้ไวยากรณ์เข้ามาช่วยจะเป็นอีกแนวทางหนึ่งที่สามารถประยุกต์ใช้และให้ผลลัพธ์ที่ดีได้

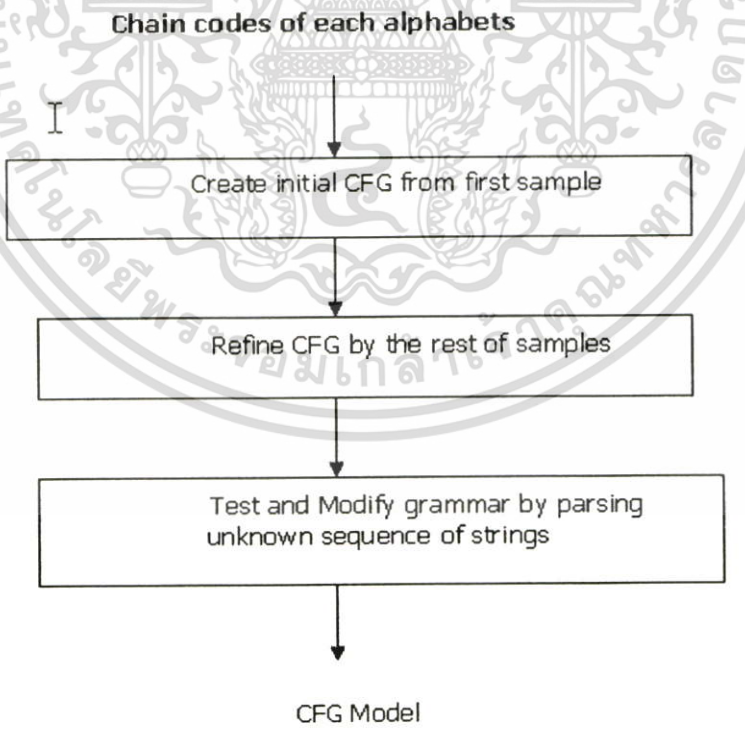
สำหรับวิทยานิพนธ์นี้ เป็นการนำเสนอการสร้างไวยากรณ์สำหรับใช้ในการรู้จำตัวอักษรภาษาไทยที่เป็นลายมือเขียนจากเชนโค้ด (Chain code) แสดงทิศทางการเขียนของตัวอักษรนั้นๆแบบ 8 ทิศทาง ดังนั้นเป้าหมายของงานวิจัยนี้คือ การสร้างไวยากรณ์แบบอัตโนมัติสำหรับตัวอักษรที่เป็นลายมือเขียนได้อย่างไร เพื่อให้ได้ไวยากรณ์ที่เป็นของตัวอักษรนั้นๆโดยต้องครอบคลุมตัวอักษรอื่นนอกเหนือจากกลุ่มตัวอักษรที่ใช้สอนด้วย ไวยากรณ์แรกเริ่มจะถูกสร้างมา

จากกลุ่มตัวอย่างของเชนโค้ดของตัวอักษรนั้นๆ จากนั้นจะทำการปรับแต่งไวยากรณ์เริ่มต้นนี้ให้มีความเหมาะสม เมื่อได้ไวยากรณ์ที่เหมาะสมแล้ว ก็จะสามารถนำไปใช้ในการรู้จำตัวอักษรโดยใช้หลักการกระจายไวยากรณ์ (parse) ได้เหมือนกับการใช้ในงานวิจัยเกี่ยวกับภาษาธรรมชาติได้ สิ่งที่ต้องคำนึงถึงสำหรับแกรมม่าที่สร้าง คือแกรมม่าจะต้องสมารถยอมรับตัวอักษรตัวนั้นที่เขียนในรูปแบบต่างๆได้แต่ก็ต้องมีความเฉพาะสำหรับตัวอักษรตัวนั้นเท่านั้นด้วย

การใช้คอนเท็กฟรีแกรมม่า (Context Free Grammar: CFG) ในการรู้จำตัวอักษรนี้จะทำให้เห็นโครงสร้างการกระจายของเชนโค้ดได้ชัดเจนขึ้น อีกทั้งยังสามารถที่จะพัฒนาต่อเป็นงานวิจัยอื่นในอนาคตให้อยู่ในรูปแบบของโมเดลทางภาษาในรูปแบบอื่นๆเช่น regular expression ที่ง่ายและสั้นสำหรับตัวอักษรตัวนั้นๆได้

5.2 ภาพรวมของระบบ

เป้าหมายของวิทยานิพนธ์นี้ คือทำการสร้างคอนเท็กฟรีแกรมม่าแบบอัตโนมัติจากกลุ่มตัวอย่าง ซึ่งกลุ่มตัวอย่างจะอยู่ในรูปของเชนโค้ดที่แสดงทิศทาง 8 ทิศทาง การสร้างแกรมม่ามีขั้นตอนหลัก 3 ขั้นตอน ดังรูป 5.1



รูปที่ 5.1 แสดงภาพรวมของระบบ

ขั้นตอนแรกคือการสร้างแกรมม่าเริ่มต้นจากสตริงแรกของตัวอักษรนั้นๆ แกรมม่าที่ได้จากสตริงสายแรกนี้จะมีความเฉพาะ (Specialize) สำหรับสตริงนี้เท่านั้น จึงต้องใช้ขั้นตอนที่ 2 คือการปรับปรุงแกรมม่าเพื่อให้มีลักษณะโดยรวม (generalize) มากขึ้นโดยใช้เซตของสตริงที่เป็นตัวสอนที่เหลือ ขั้นตอนสุดท้ายคือการทดสอบแกรมม่าโดยการนำไปทดลองกระจายหรือเรียกว่าการพาร์ส (parse) สตริงเพื่อดูผลการรู้จำ

จากการศึกษาการเรียนไวยากรณ์จากกลุ่มตัวอย่างในบทที่ 4 ของ Andreas [8] วิธีการที่นำเสนอในวิทยานิพนธ์นี้มีความเหมาะสมกับงานด้านการรู้จำตัวอักษรมากกว่า เมื่อเปรียบเทียบกับวิธีการในบทที่ 4 ดังนี้

1. การสร้างแกรมม่าเริ่มต้นง่ายและรวดเร็ว
2. วิธีการค้นหากฎในขั้นตอนการปรับปรุงแกรมม่าใช้เวลาน้อยกว่า
3. วิธีการที่นำเสนอนี้สามารถประยุกต์ใช้กับงานด้านรู้จำลายมือได้ดีกว่า

5.3 การสร้างไวยากรณ์เริ่มต้น

เพื่อให้เห็นภาพชัดเจน การอธิบายจะใช้ Regular Expression (RE) มาช่วยในการอธิบายและแสดงส่วนของลำดับเซนโค้ด

5.3.1 Regular Expression (RE)

Regular expression เป็นเครื่องมือแสดงภาษา โดยการปฏิบัติงานและสัญลักษณ์ที่ใช้ในการแสดง RE ที่ใช้ในวิทยานิพนธ์นี้มีดังนี้

1. ε แสดง สตริงว่างหรือ "empty string" ซึ่งมีความยาวสตริงเป็น 0
2. Union ของ 2 ภาษา L และ M แสดงโดย $L \cup M$ เป็นเซตของสตริงที่อยู่ใน L หรือ M หรืออยู่ในทั้ง L และ M เช่น ถ้า $L = \{001,10,111\}$ และ $M = \{\varepsilon,001\}$ แล้ว $L \cup M = \{\varepsilon,10,001,111\}$
3. Concatenation ของ 2 ภาษา L และ M แสดงโดย $L \cap M$ เป็นเซตของสตริงที่อยู่ในทั้ง L และ M เช่น ถ้า $L = \{001,10,111\}$ และ $M = \{\varepsilon,001\}$ แล้ว $L \cap M = \{001\}$
4. Closure (*) (หรือ star หรือ Kleene closure) ของภาษา L แสดงโดยใช้สัญลักษณ์ $L*$ หมายถึงการแสดงเซตของสตริงเหล่านั้นกี่ตัวก็ได้ (ตั้งแต่ศูนย์ตัวขึ้นไป) นั่นคือ $L*$ เป็น infinite union $\cup_{i\geq 0} L^i$ โดย $L^0 = \{\varepsilon\}, L^1 = L$ และ L^i สำหรับ $i > 1$ คือ $LL...L$ (เป็นการ concatenation ของ L จำนวน i ครั้ง)

5. $L^+ = LL^*$ หมายถึงการแสดงเซตของสตริงเหล่านั้นตั้งแต่ 1 ตัวขึ้นไป นั่นคือ L^+ เป็น infinite union $\cup_{i>0} L^i$ โดย $L^1 = L$ และ L^i สำหรับ $i > 1$ คือ $LL...L$ (เป็นการ concatenation ของ L จำนวน i ครั้ง)

วิธีการสร้างภาษาของ RE อาจแตกต่างจากวิธีการของไฟไนท์ออโตเมตอน (finite-automaton) แต่อย่างไรก็ตามการนิยามของทั้งสองได้ผลลัพธ์ที่แสดงภาษาที่เหมือนกันที่เรียกว่า regular language และจากไฟไนท์ออโตเมตอน เราสามารถสร้างแกมม่าที่แสดงภาษาเดียวกันกับภาษาของไฟไนท์ออโตเมตอนนั้นได้ [6].

เนื่องจากการแสดงด้วย RE ทำให้ลำดับสตริงสั้นลงและจัดกลุ่มสตริงที่เหมือนกันเข้าด้วยกันดังนั้นจะช่วยให้เข้าใจง่ายและมองเห็นรูปแบบการเรียงของสตริงชัดเจน

5.3.2 แม่แบบพิเศษที่ใช้ในการสร้างไวยากรณ์

เพื่อให้การสร้างแกรมม่าเป็นไปตามแนวคิดที่วางไว้ คือมีความเป็นลักษณะโดยรวมของแกรมม่านั้นๆ (generalize) มากพอสำหรับตัวอักษรที่จะสร้างแกรมม่า แต่ต้องไม่มากจนยอมรับสตริงอื่น จึงต้องมีกลยุทธ์พิเศษที่ใช้สร้าง แม่แบบที่ใช้เป็นแนวทางเบื้องต้นในการสร้างไวยากรณ์มี 3 แบบ คือ "แม่แบบชนิดลูป", "แม่แบบชนิดคู่ลูป" และ "แม่แบบชนิดกับดักคู่ลูป" โดย 2 กลยุทธ์แรกเป็นกลยุทธ์ที่ทำให้แกรมม่ามีความเจนเอร์รอไรซ์ ส่วนกลยุทธ์แบบกับดักคู่ลูปมีไว้เพื่อไม่ให้แกรมม่าที่สร้างมีลักษณะโดยรวมมากจนเกินไป

- แม่แบบชนิดลูป (Loop) สร้างขึ้นเพื่อรองรับการกระจายวนกลับมายังสตริงตัวเดิม ดังตัวอย่าง

  ตัวอย่างลำดับสตริง : 4, 44, 444, 44444...

  RE : 4^+

  CFG ลูป : S1 $\rightarrow$ 4 S1

หมายเหตุ แกรมม่าดังกล่าวยอมรับลำดับของเซนโค้ด '4' ตั้งแต่ 0 ตัวขึ้นไป โดยในวิทยานิพนธ์นี้จะเรียกว่า "ลูป 4"

- แม่แบบชนิดคู่ลูป (Loop Pair)

  ตัวอย่างลำดับสตริง : 434, 4 43334, 44343, 4443333434...

  RE : $(4^+3^+)^+$

  CFG "loop" : S1 $\rightarrow$ 4 S1 | 3 S2

  S2 $\rightarrow$ 3 S2 | 4 S1

หมายเหตุ แกรมม่าดังกล่าวยอมรับลำดับของเซนโค้ดของลูป4 ('4' ตั้งแต่ 0 ตัวขึ้นไป) และลูป3 ('3' ตั้งแต่ 0 ตัวขึ้นไป) สลับกันไปมา โดยในวิทยานิพนธ์นี้จะเรียกว่า "คู่ลูป 43"

- แม่แบบชนิดกับดักคู่ลูป (Trap Loop Pair)

  ตัวอย่างลำดับสตริง : 5656323, 565323, 5565333223...

  RE : $(5^{\cdot}6^{\cdot})^{\cdot}\ (3^{+}2^{\cdot})^{\cdot}$

  CFG "loop" : S3 ⟶ 5 S3 | 6 S4 | 3S5

  S4 ⟶ 6 S4 | 5 S3 | 3S5

  S5 ⟶ 3 S5 | 2 S6

  S6 ⟶ 2 S6 | 3 S5

หมายเหตุ แกรมม่าดังกล่าวยอมรับลำดับของเซนโค้ดของลูป56 และลูป32 โดยเมื่อเข้าสู่ลูป32 แล้วจะไม่สามารถกลับไปยังลูป56 ได้อีก โดยในวิทยานิพนธ์นี้จะเรียกว่า "กับดักลูป 5643"

5.3.3 ตัวอย่างการสร้างไวยากรณ์เริ่มต้น

ในการสร้างแกรมม่าจากกลุ่มตัวอย่าง เราจะใช้หลักการจากแม่แบบพิเศษที่กล่าวมาในหัวข้อก่อนหน้านี้ทำการสร้างแกรมม่าเริ่มต้นโดยใช้ลำดับเซนโค้ดสายแรก วิธีการสร้างคือจะเริ่มจากการอ่านเซนโค้ดเข้าไปทีละสตริงเพื่อสร้างแกรม่าให้กับสตริงนั้นไปตามลำดับจนครบทุกสตริง ดังตัวอย่างสตริง 4 4 3 3 3 4 5 6 5 6 3 2 3 จะสามารถสร้างแกรมม่าเริ่มต้นตามแม่แบบที่กำหนดได้ดังตาราง 5.1

จากตาราง 5.1 เป็นการแสดงวิธีการสร้างแกรมม่าเริ่มต้นโดยเริ่มจากการอ่านสตริงทีละสตริงแล้วทำการสร้างกฎสำหรับสำหรับการดีไรว์ เพื่อให้ได้สตริงตัวนั้นเพิ่มเข้าไปในแกรมม่าทีละสตริง จากตัวอย่างสตริงตัวแรกคือ "4" ทำการสร้างกฎสำหรับการดีไรว์เพื่อให้ได้สตริงตัวนี้โดยเริ่มจากนอนเทอร์มินอลที่เป็นตัวเริ่มต้น (start nonterminal) "S0" เป็นนอนเทอร์มินอลทางซ้ายมือเรียกว่า Left Hand Side (LHS) แล้วดีไรว์ได้สตริงทางขวา คือ "4" ตามด้วยนอนเทอร์มินอลอีก 1 ตัวสำหรับเป็นนอนเทอร์มินอล "S1" ลำดับถัดไปที่จะดีไรว์ไปยังเซนโค้ดตัวต่อๆไป สตริงทั้งหมดที่อยู่ทางขวาของกฎเรียกว่า Right Hand Side (RHS) ดังนั้นจากสตริง "4" ตัวแรกนี้จะได้กฎแรกของแกรมม่าคือ "S0 ⟶ 4 S1".

จากกฎ "S1 ⟶ 4 S1" เป็นกฎของแกรมม่าตามแม่แบบชนิดลูปที่เรียกว่า "ลูป4" สำหรับการดีไรว์เพื่อให้ได้สตริงของเซนโค้ดที่เป็น "4" ซ้ำๆกัน ในกรณีที่มีจำนวนเซนโค้ด "4" มากกว่าตัวอย่างในตารางนี้ กฎนี้จะครอบคลุมการดีไรว์สำหรับเซนโค้ด "4" ที่อยู่ติดกันทั้งหมดโดยไม่ต้องเพิ่มกฎเข้าไปในแกรมม่าอีก

ตาราง 5.1 แสดงตัวอย่างการสร้างแกรมม่าเริ่มต้น

4 4 3 3 3 4 5 6 5 6 3 2 3	**Sample sequence**
4	S0 → 4 S1 ← Loop
4 4	S0 → 4 S1 S1 → 4 S1
4 4 3	S0 → 4 S1 S1 → 4 S1 | 3 S2
4 4 3 3 3	S0 → 4 S1 S1 → 4 S1 | 3 S2 S2 → 3 S2
4 4 3 3 3 4	S0 → 4 S1 S1 → 4 S1 | 3 S2 S2 → 3 S2 | 4 S1 ← Loop Pair
4 4 3 3 3 4 5	S0 → 4 S1 S1 → 4 S1 | 3 S2 | 5 S3 S2 → 3 S2 | 4 S1 S3 → 5 S3
4 4 3 3 3 4 5 6	S0 → 4 S1 S1 → 4 S1 | 3 S2 | 5 S3 S2 → 3 S2 | 4 S1 S3 → 5 S3 | 6 S4 S4 → 6 S4
4 4 3 3 3 4 5 6 5 4 4 3 3 3 4 5 6 5 6	S0 → 4 S1 S1 → 4 S1 | 3 S2 | 5 S3 S2 → 3 S2 | 4 S1 S3 → 5 S3 | 6 S4 S4 → 6 S4 | 5 S3
4 4 3 3 3 4 5 6 5 6 3 4 4 3 3 3 4 5 6 5 6 3 2 4 4 3 3 3 4 5 6 5 6 3 2 3	S0 → 4 S1 S1 → 4 S1 | 3 S2 | 5 S3 S2 → 3 S2 | 4 S1 S3 → 5 S3 | 6 S4 S4 → 6 S4 | 5 S5 | 3 S5 S5 → 3 S5 | 2 S6 S6 → 2 S6 | 3 S5 ← Trap Loop Pairs

การสร้างแกรมม่าจะดำเนินไปเรื่อยๆตามลำดับเซนโค้ดโดยจากนั้นจะทำการสร้าง "ลูป4" และ "ลูป3" แต่เมื่อเราเจอเซนโค้ด "4" อีกครั้งหนึ่งลูปทั้งสองจะกลายเป็น "คู่ลูป43" ตามกลยุทธ์แบบคู่ลูป และในทำนองเดียวกันเมื่อเจอลำดับเซนโค้ด "...5 6 5 6 3 2 3" ก็จะทำการสร้างคู่ลูป56 และลูป32 เพื่อเป็นกับดักเฉพาะลำดับเซนโค้นของ "5 6" ตามด้วยลำดับเซนโค้ด "3 2" เท่านั้น นั่นคือ$(5^{'}6^{'})^{'}$ $(3^{'}2^{'})^{'}$ ไม่ใช่ลำดับอื่น เช่นหากเกิดกรณีที่มีเซนโค้ดที่ผิดซึ่งอาจเป็น 5 6 สลับไปมาแล้วพบ 3 จากนั้นกลับไปเป็น 5 หรือ 6 อีกครั้งแกรมม่านี้จะได้ไม่สามารถดีไรว์ได้ลำดับเซนโค้ดดังกล่าว โดยสองคู่ลูปนี้จะกลายเป็นกับดักของ "กับดักลูป5632"

เมื่อสร้างกฎสำหรับแกรมม่าโดยอ่านเซนโค้ดไปจนถึงสตริงตัวสุดท้ายแล้วการสิ้นสุดจะใช้สัญลักษณ์นอนเทอร์มินอล "F" เพื่อบอกถึงการสิ้นสุดสตริง โดยนอนเทอร์มินอล F ดีไรว์ได้ empty string ε การสร้างแกรมม่าที่สามารถสร้างโค้ดดังกล่าวแกรมม่าเริ่มต้นที่สร้างขึ้นนี้จะมีลักษณะเฉพาะสำหรับเซนโค้ดที่ใช้สร้างมันขึ้นมาเท่านั้น ยังไม่ครอบคลุมถึงเซนโค้ดแบบอื่นๆของตัวอักษรนั้นๆ ดังนั้นเพื่อให้มันรู้จักลำดับเซนโค้ดของอักษรนั้นๆในแบบอื่นด้วยเราต้องทำการปรับปรุงแกรมม่าเริ่มต้นนี้เพื่อให้เป็นแกรมม่าที่เจนเอนร์รอลไรซ์มากขึ้น โดยใช้ตัวสอนซึ่งเป็นกลุ่มตัวอย่างเซนโค้ดของตัวอักษร

5.4 การปรับปรุงไวยากรณ์

หลังจากสร้างแกรมม่าเริ่มต้นแล้ว เพื่อให้แกรมม่าดังกล่าวเป็นแกรมม่าที่มีลักษณะโดยรวม (Generalize) สำหรับตัวอักษรนั้นๆมากกว่า 1 แบบเราจะใช้กลุ่มตัวสอนซึ่งเป็นตัวอย่างเซนโค้ดของอักษรตัวนั้นในรูปแบบอื่นๆมาปรับปรุงแกรมม่าให้รู้จักตัวอักษรนั้นในรูปแบบอื่นๆนอกเหนือจากสตริงเริ่มต้น

เป้าหมายของการปรับปรุงแกรมม่าคือ การสร้างแกรมม่าที่สั้นกระทัดรัดสำหรับอักษรตัวนั้นโดยแกรมม่านี้จะต้องมีลักษณะโดยรวมมากพอที่จะยอมรับเซนโค้ดของตัวอักษรนั้นๆเท่านั้นและไม่ยอมรับเซนโค้ดของอักษรตัวอื่นจึงจะถือว่าเป็นแกรมม่าที่เหมาะสมสำหรับอักษรนั้น

การอ่านเซนโค้ดเพื่อปรับปรุงแกรมม่าจะทำเช่นเดียวกับการสร้างคืออ่านสตริงของเซนโค้ดเข้ามาทีละตัวแล้วทำการพาร์สแกรมม่า โดยเริ่มต้นจากกฎที่มี LHS เป็น start symbol "S0" ทำการพาร์สแกรมม่าไปเรื่อยๆ เมื่อไปถึงกฎใดๆที่ไม่มีทางเลือกให้พาร์สสำหรับสตริงนั้น เพื่อให้แกรมม่านั้นสามารถพาร์สแกรมม่าต่อไปจนสุดสตริงของเซนโค้ดได้จึงต้องทำการปรับปรุงแกรมม่าโดยการเพิ่มทางเลือกให้กับกฎนั้น สำหรับการเลือกทางเลือกที่จะเพิ่มเข้าไปใหม่นั้นมี 2 วิธีคือ

1. เพิ่มกฎใหม่เข้าไปในแกรมม่าสำหรับการดีไรว์สตริงตัวใหม่นั้น
2. เพิ่มทางเลือกให้เพื่อให้แกรมม่าสามารถกระจายไปยังกฎใดกฎหนึ่งที่อยู่ในแกรมม่านั้นและมีความสามารถดีไรว์สตริงนั้นต่อไปได้

อัลกอริทึมการค้นหาเป็นปัญหาหลักอีกปัญหาหนึ่งที่ต้องพิจารณาเป็นพิเศษ การเลือกวิธีการค้นหาที่เหมาะสมจะทำให้การปรับปรุงแกรมม่ามีประสิทธิภาพมากขึ้น เวลาที่ใช้ส่วนใหญ่จะหมดไปกับการค้นหากฎภายในแกรมม่า อัลกอริทึมการค้นหาที่ใช้ในงานวิจัยนี้คือการค้นหาแบบฮิวริสติก (Heuristic Search) โดยจะทำการค้นไปในแกรมม่าเพื่อหากฎที่เป็นไปได้ทุกกฎที่มีความสามารถในการดีไรว์สตริงตัวนั้น เรียกว่ากฎทางเลือกหรือ candidate rule เมื่อพบกฎที่เป็นไปได้นั้นแล้วเราจะทำการเพิ่มให้เป็นทางเลือกของการกระจายจากกฎปัจจุบันไปยังกฎทางเลือกนั้นๆ วิธีการเพิ่มทางเลือกทำได้โดยการเพิ่มนอนเทอร์มินอลที่เป็น LHS ของกฎทางเลือกลงไปยัง RHS ของกฎปัจจุบัน การเพิ่มทางเลือกลงไปแต่ละครั้งจะได้แกรมม่าที่เปลี่ยนแปลงไปจากเดิม ซึ่งถือว่าเป็นแกรมม่าใหม่ ดังนั้นเมื่อเพิ่มกฎทางเลือกทั้งหมดลงไป เราจะได้แกรมม่าที่สามารถดีไรว์ได้สตริงตัวนั้นหลายแบบ

แต่แกรมม่าที่ต้องการคือแกรมม่าที่มีความเหมาะสมที่สุดเพียงแกรมม่าเดียวเท่านั้น จึงต้องใช้พารามิเตอร์บางตัวมาช่วยในการตัดสิน โดยเราจะทำการกำหนดค่าการปรับปรุง (Cost) ให้กับการเปลี่ยนแปลงที่เกิดขึ้นเพื่อใช้เป็นเกณฑ์สำหรับเลือกแกรมม่าที่ดีที่สุด เนื่องจากเราต้องการแกรมม่าที่สั้นกระทัดรัดดังนั้นค่าการปรับปรุงของการเพิ่มทางเลือกที่เป็นกฎซึ่งมีอยู่แล้วในแกรมม่านั้นจะน้อยกว่าค่าการเปลี่ยนแปลงของการเพิ่มกฎใหม่เข้าไปในแกรมม่า นั่นคือ

$$cost_{jump} < cost_{add} \qquad (5.1)$$

ซึ่ง $cost_{jump}$ คือ ค่าการปรับปรุงของการเพิ่มทางเลือกที่เป็นกฎซึ่งมีอยู่แล้วในแกรมม่านั้น และ $cost_{add}$ คือ ค่าการเปลี่ยนแปลงของการเพิ่มกฎใหม่เข้าไปในแกรมม่า

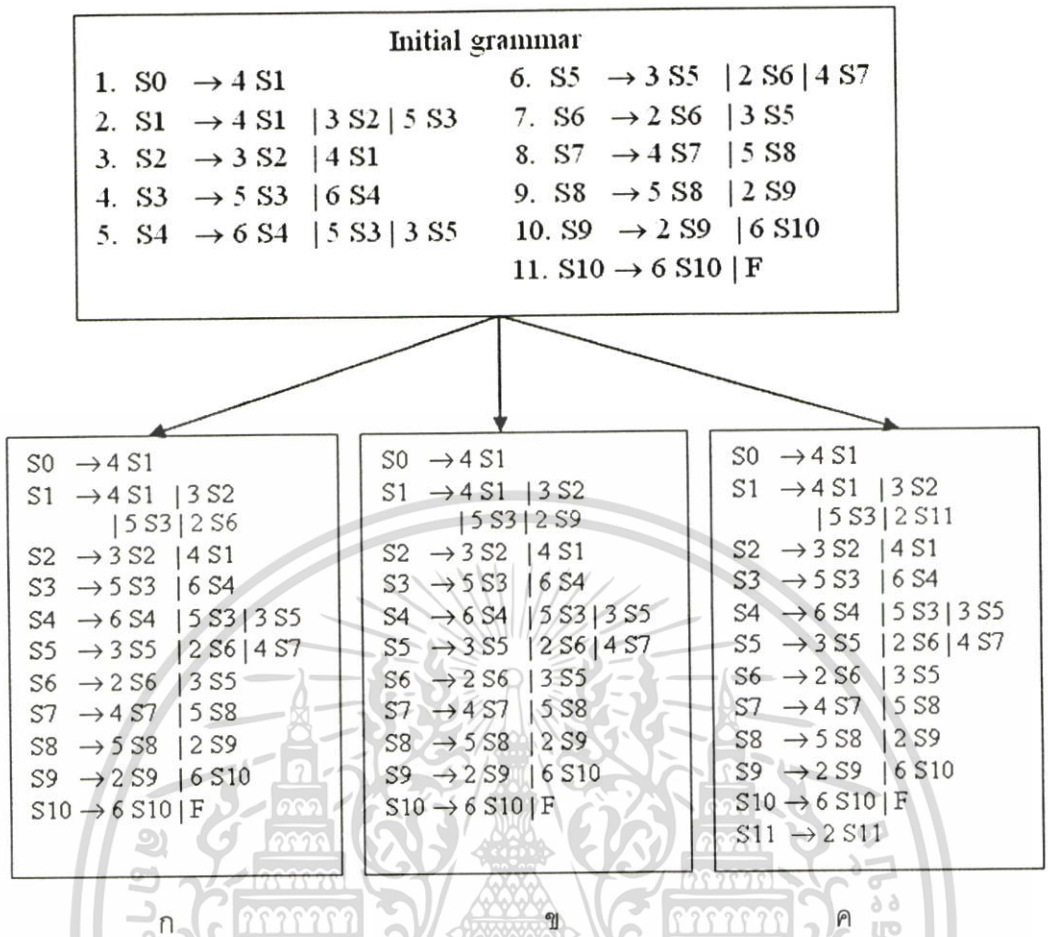
ตัวอย่างการปรับปรุงแกรมม่า กำหนดแกรมม่าเริ่มต้นดังตาราง 5.2

ตาราง 5.2 ตัวอย่างแกรมม่าเริ่มต้นที่จะใช้ปรับปรุง

1. S0 → 4 S1
2. S1 → 4 S1 | 3 S2 | 5 S3
3. S2 → 3 S2 | 4 S1
4. S3 → 5 S3 | 6 S4
5. S4 → 6 S4 | 5 S3 | 3 S
6. S5 → 3 S5 | 2 S6 | 4 S7
7. S6 → 2 S6 | 3 S5
8. S7 → 4 S7 | 5 S8
9. S8 → 5 S8 | 2 S9
10. S9 → 2 S9 | 6 S10
11. S10 → 6 S10 | F

กำหนดสตริงที่เป็นตัวสอนสำหรับปรับปรุงแกรมม่าเริ่มต้นคือ 4 3 4 2 3 2 4 5 2 6

การเริ่มปรับปรุงแกรมม่าเริ่มจากการพาร์สกฎแรก S0 จากแกรมม่าในตาราง 5.2 การพาร์สตามเซนโค้ดที่ใช้ปรับปรุงจะเห็นว่าลำดับเซนโค้ด "4 3 4" สามารถพาร์สได้จนกระทั่งถึงเซนโค้ด "2" ซึ่งไม่สามารถกระจายจาก S1 ต่อไปได้ เราต้องทำการเพิ่มทางเลือกเข้าไปเพื่อให้แกรมม่าสามารถกระจายต่อไปได้ โดยจากการค้นหาในแกรมม่าจะได้ทางเลือกการปรับปรุงแกรมม่าดังรูป 5.2



รูปที่ 5.2 แสดงแกรมม่าใหม่ทางเลือกที่ปรับปรุง

จากกรูป 5.2 จะเห็นว่าสามารถปรับปรุงแกรมม่าโดยทำการเพิ่มทางเลือกการดีไรว์ให้กับกฎที่ 1 ด้าน RHS ได้ 3 แบบ ในรูป 5.2ก เป็นการเพิ่ม "2 S6" รูป 5.2ข เป็นการเพิ่ม "2 S9" และรูป 5.2ค เป็นการเพิ่มกฎใหม่ให้กับแกรมม่าคือ "2 S11" จากแกรมม่าทางเลือกทั้ง 3 จะเห็นว่าแกรมม่าของรูปที่ 5.2ก เป็นแกรมม่าที่เหมาะสมที่สุดเนื่องจาก จากนอลเทอร์มินอล "S6" ในกฎที่ 7 สามารถพาร์สตริงต่อไปเรื่อยได้จนถึงเซนโค้ดตัวสุดท้าย

หลักการพิจารณากฎที่จะกลายเป็นกฎทางเลือกสำหรับกระจายได้สตริงตัวนั้นคือ กฎนั้นๆ ต้องมี LHS กระจายได้ RHS ที่มีเทอร์มินอลเป็นตัวเดียวกับสตริงของเซนโค้ดนั้นตามด้วยนอนเทอร์มินอลตัวเดียวกับ LHS ของกฎนั้น ดังตัวอย่างในรูปที่ 5.2 เมื่อเซนโค้ดขณะนั้นคือ "2" ซึ่งไม่สามารถกระจายต่อได้เมื่อมาถึงกฎที่ 2 เมื่อพิจารณากฎภายในแกรมม่าเพื่อเป็นกฎทางเลือกจะเห็นว่า กฎที่ 7 คือ S6 → 2 S6 | 3 S5 และ กฎที่ 10 คือ S9 → 2 S9 | 6 S10 เป็นกฎทางเลือก ทั้งนี้เพราะจากกฎที่ 7 มี LHS คือ S6 ที่สามารถเลือกทางเลือกกระจายด้าน RHS ได้ "2 S6" ซึ่งมี เทอร์มินอลคือ "2" เหมือนกับเซนโค้ดขณะนั้นตามด้วยนอนเทอร์มินอล "S6" ที่เป็นตัว

เดียวกันกับนอนเทอร์มินอลทาง LHS ของกฎ เช่นเดียวกับกฎที่ 10 ที่มี RHS คือ "2 S9" อัลกอริทึมการปรับปรุงแกรมม่าเป็นดังรูป 5.3

```
For each chain in a sample

    For each grammar
        - Find candidate rules:
            → all nonterminal with terminal, as same as current chain
            → new nonterminal with terminal, as same as current chain

        - add all candidates to be new choices in the right hand side of current rule

        - construct new grammars by replace current rule with these new rules
    End For

End For
```

รูปที่ 5.3 แสดงอัลกอริทึมการปรับปรุง

การเพิ่มทางเลือกการกระจายต้องทำตามแม่แบบพิเศษดังกล่าวข้างต้นด้วย เพื่อให้คงความลักษณะโดยรวมไว้ โดยต้องทำการเช็คลำดับเซนโค้ดใหม่ว่าสามารถสร้างแม่แบบพิเศษใดได้บ้างกับแกรมม่าเดิมที่มีอยู่หรือต้องเพิ่มใหม่เข้าไป หากกฎนั้นเป็นส่วนหนึ่งของกลยุทธ์ใดแม่แบบหนึ่งอยู่แล้ว ก็จะไม่นำมาสร้างอีก เพราะหากนำกฎที่เป็นส่วนหนึ่งของแม่แบบแล้วมาสร้างแม่แบบร่วมกับกฎอื่นอีกจะทำให้เกิดการดีไรว์ของเซนโค้ดที่ผิดลำดับ

5.5 การค้นหากฎที่เหมาะสมแบบ heuristic

การค้นหาจะทำในขั้นตอนของการปรับปรุงแกรมม่า เมื่อใดก็ตามที่การพาร์สของแกรมม่าตามสตริงเซนโค้ดที่เป็นอินพุตไม่สามารถกระจายเพื่อให้ได้สตริงตัวนั้น ก็จะต้องทำการค้นหากฎภายในแกรมม่าที่สามารถจะพาร์สอินพุตตัวนั้นเพื่อให้สามารถกระจายได้ผลลัพธ์ตามสตริง

อัลกอริทึมการค้นหาเป็นปัญหาหลักอีกปัญหาหนึ่งที่ต้องพิจารณาเป็นพิเศษ ในการเลือกวิธีการค้นหาที่เหมาะสมจะทำให้การปรับปรุงแกรมม่ามีประสิทธิภาพมากขึ้น เวลาที่ใช้ส่วนใหญ่จะใช้ไปกับการค้นหากฎภายในแกรมม่า วิธีการค้นหาที่ใช้ในงานวิจัยนี้คือการค้นหาแบบฮิวริสติก

(Heuristic search) ทำการค้นหากฎในแกรมม่า เริ่มพิจารณา ณ กฎปัจจุบันที่การพาร์สหยุดอยู่ การค้นหาจะเริ่มค้นหาจากกฎนี้โดยพิจารณาไปตามกฎทุกกฎที่อยู่ทางด้าน RHS ของกฎจนเจอกฎที่ตรงตามความต้องการก็จะนำนอนเทอร์มินอลที่อยู่ด้าน LHS ของกฎนั้นๆไปเพิ่มเป็นทางเลือกการดีไรว์ให้กับกฎปัจจุบัน จำนวนแกรมม่าที่เกิดขึ้นมาใหม่นี้จะขึ้นกับจำนวนทางเลือกที่มี

แกรมม่าที่เกิดขึ้นใหม่นี้จะมีแกรมม่าใดแกรมม่าหนึ่งที่จะเป็นแกรมม่าที่เหมาะสมที่สุด ที่เราต้องการสำหรับอักษรนั้นๆ แต่ขณะทำการปรับปรุงอยู่นั้นเราไม่สามารถทราบได้ว่าแกรมม่าที่เกิดขึ้นนี้แกรมม่าใดเป็นแกรมม่าที่เหมาะสมที่สุด ดังนั้นระบบจึงจำเป็นต้องทำการเก็บแกรมม่าทางเลือกทั้งหมดนี้ไว้ในลักษณะของฮิวริสติกทรี แกรมม่าทางเลือกที่เกิดขึ้นมาใหม่นี้จะถูกกำหนดค่าของการปรับปรุง (cost) ค่าการปรับปรุงแกรมม่าของการเพิ่มกฎใหม่เข้าไปจะมากกว่าค่าการปรับปรุงแกรมม่าที่เพิ่มทางเลือกไปยังกฎที่มีอยู่แล้วในแกรมม่า และเมื่อทำการปรับปรุงจนถึงเซนโค้ดตัวสุดท้าย ระบบจะทำการพิจารณาแกรมม่าที่ดีที่สุดจากแกรมม่าที่อยู่ในโหนดล่างสุดของทรีเท่านั้น หลักการเลือกแกรมม่าจะดูจากค่าการปรับปรุง แกรมม่าใดที่มีค่าการปรับปรุงน้อยที่สุดแสดงว่าแกรมม่านั้นมีการเพิ่มกฎใหม่น้อยที่สุดจึงเป็นแกรมม่าที่กระชับและเป็นแกรมม่าที่ดีที่สุดสำหรับอักษรตัวนั้น

กำหนดฟังก์ชันของฮิวริสติกคือ

$$f(n) = g(n) + h(n) \qquad (5.2)$$

โดยที่ $f(n)$ คือ ฟังก์ชันสำหรับการวิเคราะห์ของฮิวริสติกหรือเรียกว่า Evaluation function

$g(n)$ คือ ค่าการปรับปรุง (Cost) เพื่อไปอยู่ในสเตจปัจจุบัน

$h(n)$ คือ ค่าฟังก์ชันของฮิวริสติกหรือที่เรียกว่า Heuristic function นั่นคือ ค่าการปรับปรุงเพื่อให้ไปถึงยังเป้าหมาย (goal)

ตัวอย่างการวิเคราะห์ปัญหาตามหลักการของฮิวริสติก และการสร้างฮิวริสติกทรีของแกรมม่าเริ่มต้นในตารางที่ 5.2 กำหนดให้

State: ขั้นตอนแต่ละขั้นในการพยายามพาร์สสตริงของตัวสอนโดยอาจเพิ่มกฎใหม่หรือเลือกใช้กฎเดิมที่มีอยู่ในแกรมม่า

Initial state: จากแกรมม่าเริ่มต้นจะเริ่มจาก start state

Successor function: การได้แกรมม่าที่เพิ่มกฎใหม่น้อยที่สุดที่สามารถพาร์สสตริงของเซนโค้ดที่เป็นตัวสอนได้

Goal test: เช็คโดยการพาร์สสตริงของตัวสอนได้

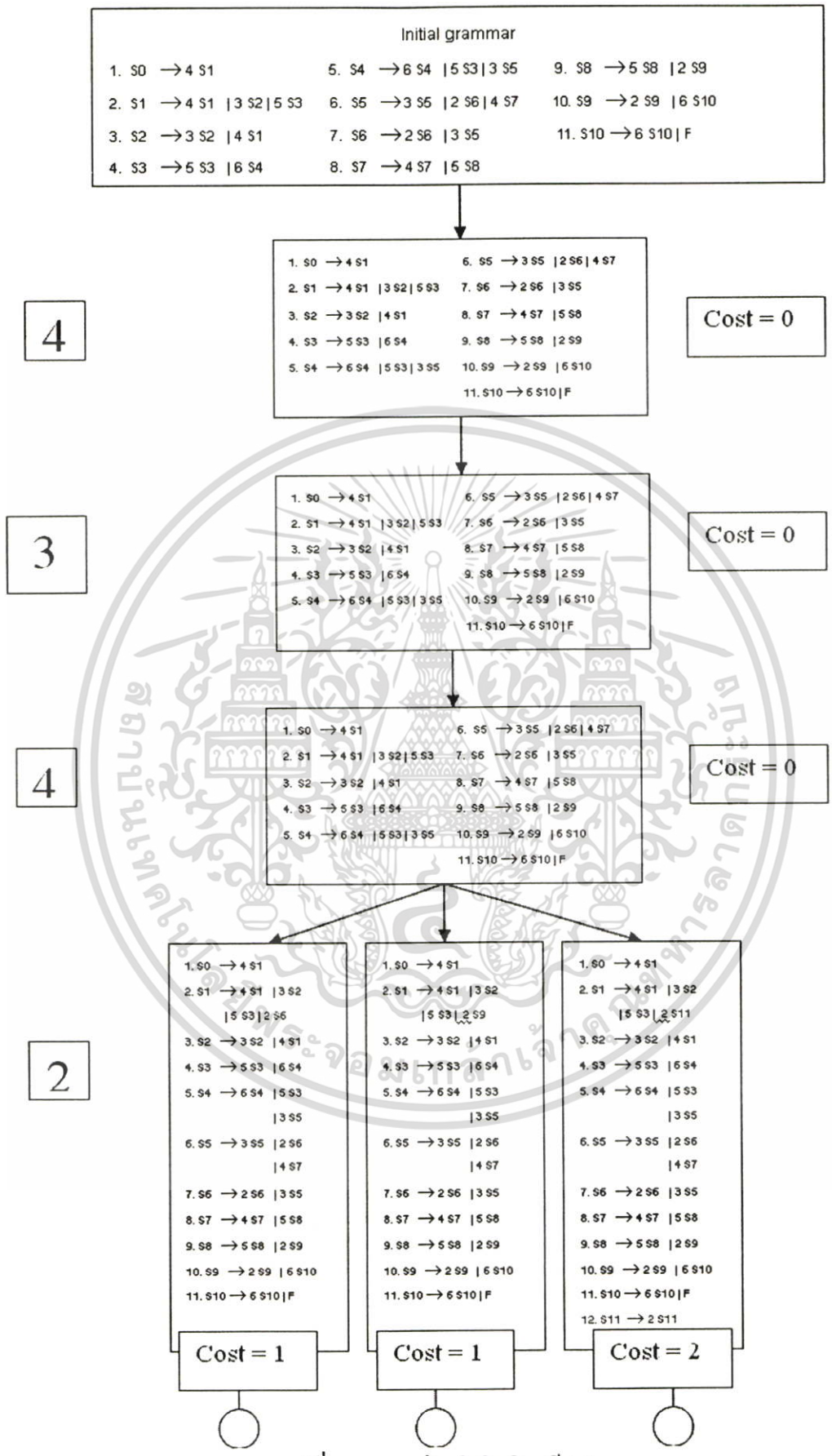
Path cost: กำหนดค่าการเพิ่มของการพยายามพาร์สสตริง โดย

- ค่าการปรับปรุงของการเพิ่มทางเลือกที่เป็นกฎซึ่งมีอยู่แล้วในแกรมม่า $\cos t_{jump}$ = +1
- ค่าการเปลี่ยนแปลงของการเพิ่มกฎใหม่เข้าไปในแกรมม่า $\cos t_{add}$ = +2

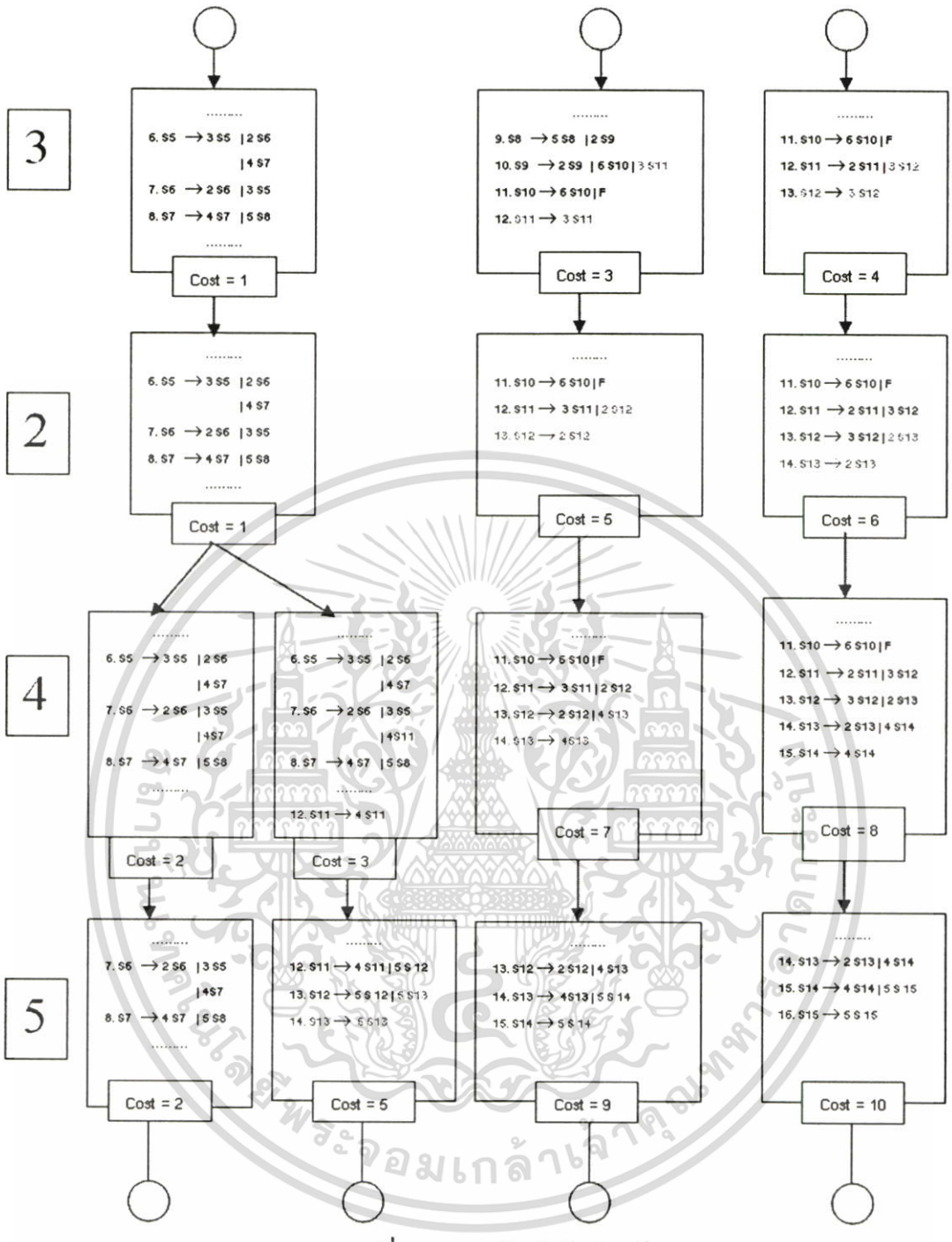
สตริงที่เป็นตัวสอนสำหรับปรับปรุงแกรมม่าเริ่มต้นคือ 4 3 4 2 3 2 4 5 2 6
แกรมม่าเริ่มต้นในตารางที่ 5.2แกรมม่าเริ่มต้นในตารางที่ 5.2
ฮิวริสติกทรีของการปรับปรุงดังแสดงในรูปที่ 5.4, 5.5 และ 5.6

เมื่อทำการปรับปรุงแกรมม่าจนจบเซนโค้ดแล้ว กฎสุดท้ายของแกรมม่าต้องจบลงด้วยนอนเทอร์มินอล "F" เพื่อแสดงถึงการจบลำดับของเซนโค้ด

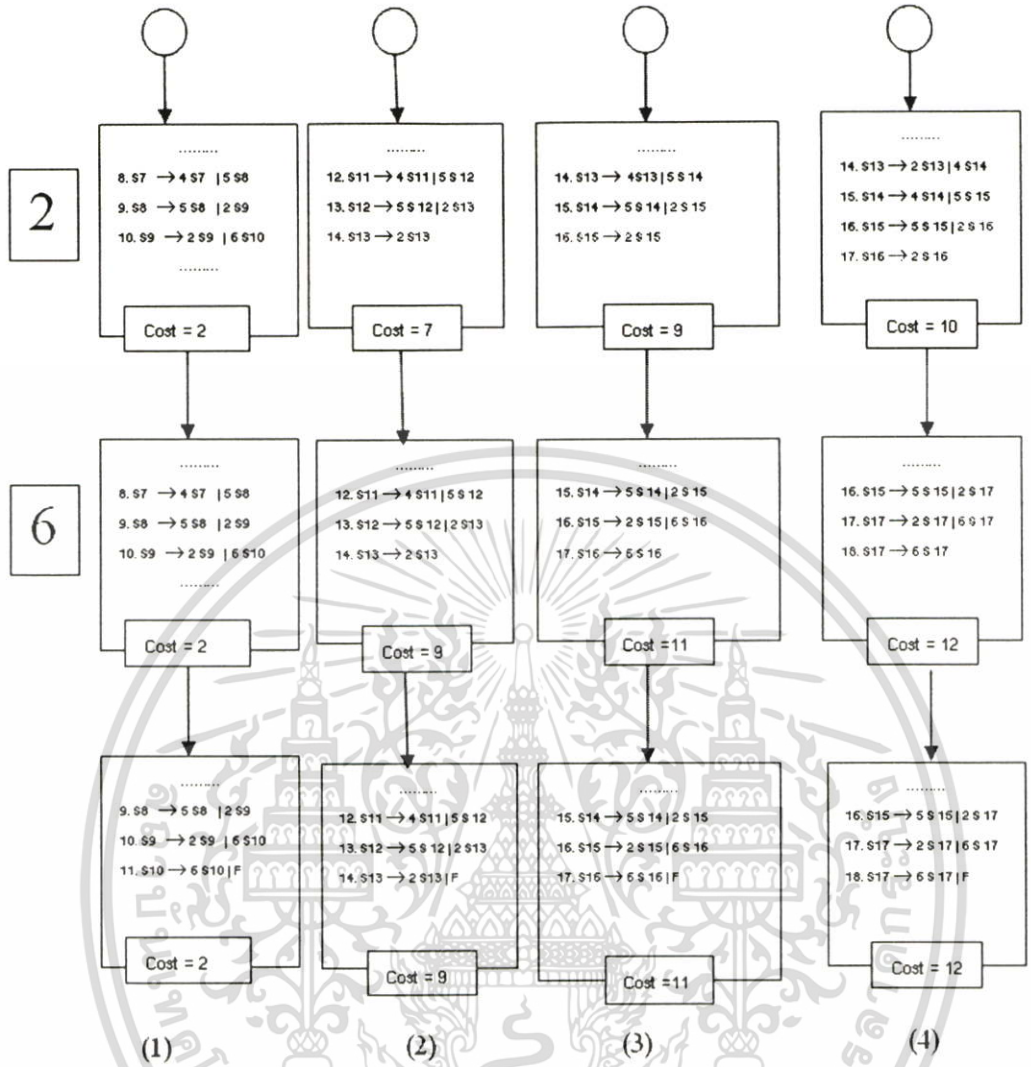
จากตัวอย่างการสร้างฮิวริสติกทรีในรูปที่ 5.6 การเลือกจากแกรมม่าที่เหมาะสมสำหรับอักษรตัวนี้จะพิจารณาจากแกรมม่าที่อยู่ในโหนดสุดท้าย โดยดูจากค่าการปรับปรุงของแกรมม่า จากรูปที่ 5.6 จะได้แกรมม่าที่ (1) ดังแกรมม่าในตารางที่ 5.3 เป็นแกรมม่าที่เหมาะสมที่สุดสำหรับสตริงของอักษรนี้ เนื่องจากค่าการปรับปรุงของแกรมม่าที่ (1) มีค่าเท่ากับ 2 ซึ่งน้อยที่สุด และแกรมม่านี้จะกลายเป็นแกรมม่าเริ่มต้นสำหรับการสอนเซนโค้ดสายต่อไป.



รูปที่ 5.4 การสร้างฮิวริสติกทรี (1)



รูปที่ 5.5 การสร้างฮิวริสติกทรี (2)



รูปที่ 5.6 การสร้างฮิวริสติกทรี (3)

ตาราง 5.3 แกรมม่าที่ปรับปรุงแล้ว

1. S0 → 4 S1
2. S1 → 4 S1 | 3 S2 | 5 S3 | 2 S6
3. S2 → 3 S2 | 4 S1
4. S3 → 5 S3 | 6 S4
5. S4 → 6 S4 | 5 S3 | 3 S
6. S5 → 3 S5 | 2 S6 | 4 S7
7. S6 → 2 S6 | 3 S5 | 4 S7
8. S7 → 4 S7 | 5 S8
9. S8 → 5 S8 | 2 S9
10. S9 → 2 S9 | 6 S10
11. S10 → 6 S10 | F

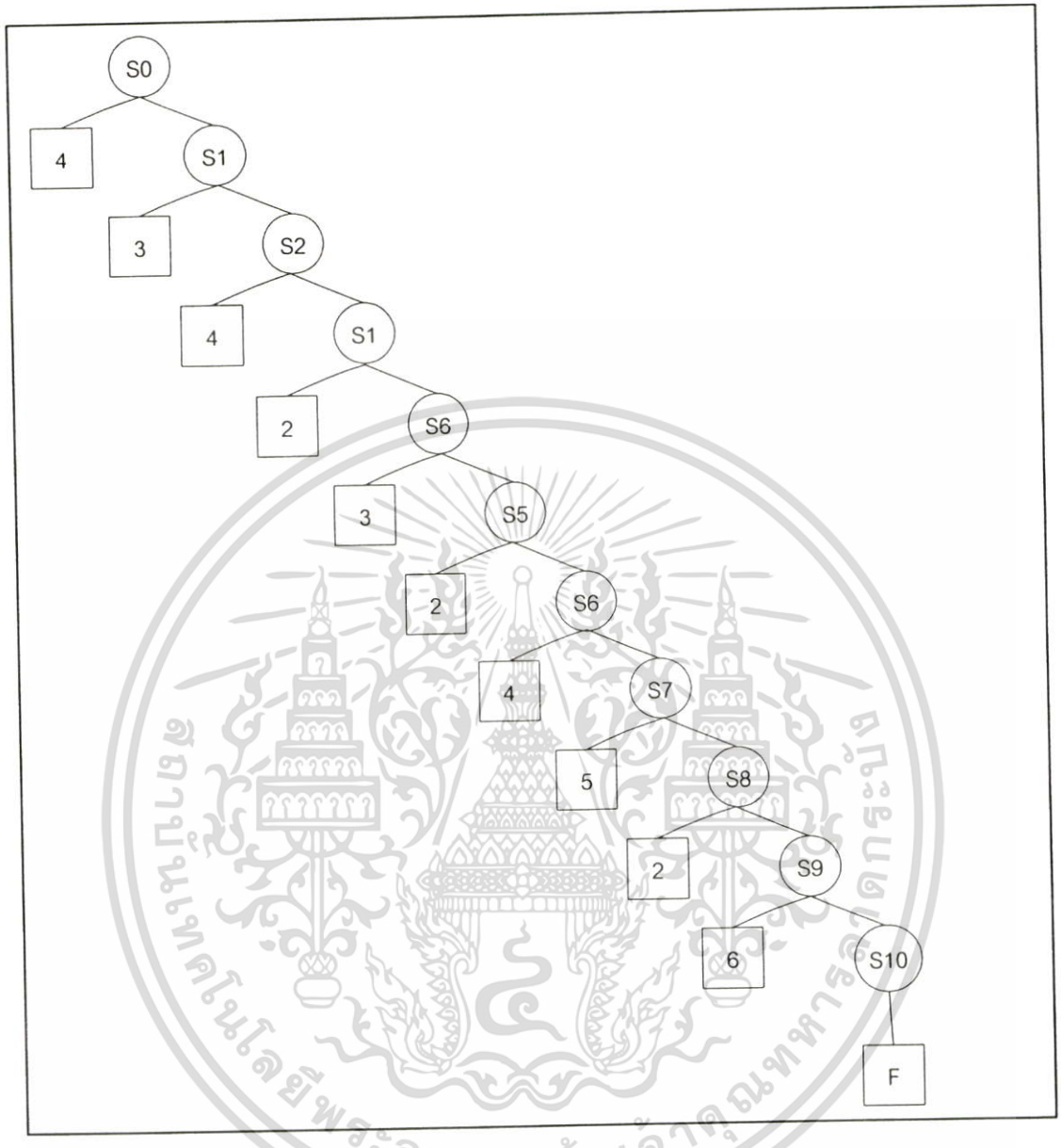
5.6 การพาร์สแกรมม่า

ในการปรับปรุงแกรมม่าและการทดสอบการรู้จำของแกรมม่าต้องทำการพาร์สแกรมม่าเพื่อให้ได้ตามสตริงที่เป็นอินพุต การพาร์สหรือการกระจายแกรมม่าจะมี 2 แบบคือ แบบ bottom-up และแบบ top-down

- แบบ bottom-up เป็นการพาร์สแกรมม่าโดยเริ่มจากเทอร์มินอล แล้วพยายามโยงไปหานอนเทอร์มินอลที่ดีไรว์ได้เทอร์มินอลตัวนั้น การทำงานจะสิ้นสุดเมื่อพาร์สไปถึงนอนเทอร์มินอลเริ่มต้น "S0"
- แบบ top-down เป็นการพาร์สแกรมม่าในทิศทางที่ตรงข้ามกับแบบ bottom-up นั่นคือจะเริ่มจากนอนเทอร์มินอลเริ่มต้น "S0" แล้วกระจายแกรมม่าเพื่อให้ได้เทอร์มินอลทั้งหมด

ในวิทยานิพนธ์นี้ใช้การกระจายแกรมม่าแบบ top-down และเนื่องจากต้องการแสดงลำดับการสิ้นสุดของเซนโค้ดด้วย ดังนั้นการกระจายแกรมม่าจะต้องจบลงที่นอนเทอร์มินอลสุดท้ายคือ "F" จึงจะถือว่าแกรมม่านั้นสามารถกระจายได้ตามเซนโค้ดที่เป็นอินพุตได้

รูปที่ 5.7 แสดงตัวอย่างการกระจายแกรมม่าแบบ top-down ที่ใช้ในวิทยานิพนธ์นี้ตามเซนโค้ด 4 3 4 2 3 2 4 5 2 6 โดยใช้แกรมม่าเริ่มต้นจากตาราง 5.3



รูปที่ 5.7 แสดงการกระจายแกรมม่า

บทที่ 6

การทดลองและผลการทดลอง

6.1 บทนำ

วิทยานิพนธ์นี้เป็นการเสนอวิธีการสร้างโมเดลการรู้จำโดยใช้คอนเท็กฟรีแกรมม่า ซึ่งสามารถเป็นอีกทางหนึ่งที่นำมาใช้ในการรู้จำ ก่อนหน้านี้ในงานวิจัยด้านการรู้จำมีหลายวิธีที่ใช้ เช่น ใช้การ Matching ทางโครงสร้าง [36], การใช้ Fuzzy [37] ในการรู้จำ, การใช้ Rough set [38] และ/หรือ การใช้ Hidden markov model [39] ในการสอนและการรู้จำ และการใช้ Neural Network [40] ในการสอนและการรู้จำ เป็นต้น

ในการทดลองการรู้จำตัวอักษรภาษาไทย จะเริ่มจากการนำค่าพิกัด XY (Coordinate X Y) ที่ได้จากอุปกรณ์อิเล็กทรอนิกส์ มาเข้ากฎลูกโซ่ตามทิศทางแบบ 8 ทิศทาง (8-connected grid) เพื่อให้ได้เชนโค้ดของตัวอักษรแต่ละแบบออกมา จากนั้นจึงจะนำเชนโค้ดของตัวอักษรที่ได้มาทำการรู้จำ

6.2 ข้อมูลที่นำมาเป็นอินพุต

เชนโค้ดที่นำมาทำการทดสอบมีความยาวของสตริงเท่ากับ 100 ซึ่งเป็นเชนโค้ดที่ถูกกำจัด noise แล้ว แกรมม่าเริ่มต้นจะสร้างมาจากเชนโค้ดสายแรก จากนั้นจะนำเชนโค้ดต่างๆที่เหลือของตัวอักษรเดียวกันนั้นมาใช้ในการสอนเพื่อสร้างโมเดลตัวอักษรสำหรับอักษรนั้นๆ โมเดลที่ได้จะอยู่ในรูป context free grammar โดยแกรมม่าที่ได้จะมีลักษณะโดยรวมสำหรับอักษรตัวนั้น

ตัวอักษรภาษาไทยที่ใช้ในการทดลองจะต้องเป็นตัวอักษรที่มีลักษณะการเขียนแบบรวดเดียว ไม่มีการยกมือเท่านั้น ทั้งนี้เพราะลำดับของเชนโค้ดที่นำมาใช้ในการทดลองเป็นลำดับทิศทางตามเส้นของตัวอักษรเรียงต่อกันไปตามการเขียน

ขั้นตอนการทดสอบการรู้จำของโมเดลคือ การนำแกรมม่าที่ได้นี้ไปทำการพาร์สสตริงของเชนโค้ดที่เป็นตัวทดสอบ การพาร์สแกรมม่าจะใช้วิธีการพาร์สแบบบนลงล่าง (Top-Down) โดยเริ่มต้นจากนอนเอทร์มินอลเริ่มต้น "S0" ซึ่งเป็นกฎแรกของแกรมม่า ทำการดีไรว์นอนเทอร์มินอลที่อยู่ทางด้านขวาของกฎปัจจุบันไปเรื่อยๆ โดยผลลัพธ์ที่ต้องการคือเทอร์มินอลหรือสตริงตามเชนโค้ดที่เป็นอินพุต หากการดีไรว์จบลงที่นอนเทอร์มินอลตัวสุดท้าย (Final nonterminal) "F" แสดง

ว่าแกรมม่านั้นสามารถสร้างสายของเซนโค้ดสายนั้นได้ เราจะสรุปว่าแกรมม่านี้มีความสามารถในการรู้จำเซนโค้ดสายนั้น

เพื่อทดสอบประสิทธิภาพของผลลัพธ์การรู้จำของแกรมม่า เราจะนำผลการรู้จำที่ได้มาเปรียบเทียบกับการรู้จำของโมเดลที่สร้างโดยใช้ Hidden makov model

6.3 การทดสอบการรู้จำ

สำหรับการทดสอบการรู้จำตัวอักษรภาษาไทย ประกอบด้วยตัวอักษรจำนวน 5 ชุด ซึ่งมีตัวอักษรทั้งหมด 38 ตัวดังรูปที่ 6.1

รูปที่ 6.1 แสดงตัวอักษรไทยที่นำมาทำการรู้จำ

ข้อมูลที่เป็น Test pattern

- ข้อมูลชุดที่หนึ่งจำนวน 3736 ตัว เป็น Test Pattern 1
- ข้อมูลชุดที่สองจำนวน 1596 ตัว เป็น Test Pattern 2
- ข้อมูลชุดที่สามจำนวน 2265 ตัว เป็น Test Pattern 3
- ข้อมูลชุดที่สี่จำนวน 1319 ตัว เป็น Test Pattern 4

ข้อมูลที่เป็น Train pattern

- ข้อมูลชุดที่หนึ่งจำนวน 4653 ตัว เป็น Train Pattern

รูปที่ 6.2 แสดงตัวอย่างของตัวอักษรที่ใช้ในการสอน

ตารางที่ 6.1 แสดงผลการรู้จำชุดการสอน

โมเดลตัวอักษร		ชุดอักษรที่ใช้ทดสอบ																																		
		118	135	136	125	146	147	134	127	129	139	122	112	122	118	133	119	112	135	123	107	114	111	111	109	111	111	117	118	104	106	106	99	115	119	113	211
		ก	ข	ค	ฆ	ง	จ	ล	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ฉ	ว	ห	ฬ	อ	ฮ
	ก	77	23	15	24	4	17	21	7	55	28	9	8	67	23	98	110	86	32	13	12	17	14	9	8	22	13	15	20	16	8	5	19	3	9	6	11
	ข	87	92	66	48	61	49	28	55	56	76	64	16	53	65	79	62	104	97	43	45	65	67	71	69	75	84	82	47	55	42	23	14	45	54	17	82
	ห	45	75	37	12	22	4	38	0	0	13	6	7	1	16	44	16	21	116	0	30	36	25	27	32	40	43	32	26	8	8	3	5	6	5	4	0
	ร	28	3	45	16	5	14	15	92	100	32	77	61	29	36	16	10	29	3	110	1	3	3	56	56	6	2	59	5	52	81	6	7	46	78	32	192
	น	28	78	34	45	48	60	97	79	29	38	67	65	58	75	61	48	47	55	31	105	65	44	42	51	47	54	40	15	31	36	29	21	44	33	48	82
	บ	12	64	31	86	12	22	17	22	35	102	15	18	72	72	22	38	39	49	56	47	100	81	33	34	53	54	22	92	45	21	23	7	45	26	16	40
	ป	34	81	68	58	24	48	40	64	66	70	40	39	67	78	78	88	73	111	49	34	95	108	57	61	70	80	50	83	68	22	31	23	93	74	44	84
	ผ	5	41	16	24	22	27	29	25	38	58	19	35	58	32	14	25	6	33	23	28	36	24	106	95	41	43	28	33	71	2	34	7	13	36	32	92
	ฝ	1	3	12	14	1	4	7	2	13	23	19	42	26	18	4	16	2	6	10	5	7	9	92	98	13	16	30	10	68	8	4	4	6	35	27	140
	พ	33	84	33	88	66	71	47	32	59	87	28	35	92	66	50	43	23	48	13	38	50	31	46	53	108	98	30	94	32	31	44	4	25	51	12	12
	ฟ	33	61	43	117	39	27	121	23	113	108	60	43	110	101	50	54	27	47	33	98	56	41	68	77	104	106	27	90	63	25	46	3	20	101	25	80
	ภ	9	9	62	10	22	11	26	8	6	22	69	71	28	36	25	37	25	28	12	12	5	8	17	6	6	10	97	5	2	2	6	2	10	9	6	8
	ม	10	22	21	105	20	20	51	30	68	108	19	18	83	68	40	30	36	30	26	28	8	12	22	16	63	68	13	114	14	14	53	0	10	81	17	34
	ย	11	28	24	47	15	10	3	35	45	61	30	18	52	35	17	11	9	29	30	17	38	27	61	68	23	27	48	24	94	40	0	15	19	23	11	24
	ร	2	19	36	32	27	20	27	45	33	15	64	66	42	11	35	33	7	18	59	14	7	4	40	28	18	7	74	17	26	105	4	6	9	18	12	134
	ฉ	3	4	9	3	19	77	84	25	29	11	51	48	7	8	1	1	11	0	18	1	2	0	29	31	0	0	12	3	28	16	99	59	37	66	74	140
	ว	1	2	18	7	11	34	64	32	27	20	38	52	7	2	9	5	2	8	46	8	10	5	14	15	3	1	31	11	23	76	27	97	28	29	100	138
	ห	19	16	56	27	4	68	56	11	17	47	51	x	15	41	2	1	33	1	15	17	18	17	69	55	23	22	40	3	47	22	31	18	111	89	29	162
	ฬ	18	22	21	10	8	6	36	21	25	19	44	64	22	20	7	12	14	2	15	8	47	36	14	13	10	9	28	1	24	14	20	24	16	113	20	58
	อ	6	3	0	2	4	59	56	19	17	0	37	36	2	5	0	1	8	0	15	0	0	0	4	0	0	1	26	1	4	49	38	85	14	12	108	62
	ฮ	0	0	5	1	0	10	8	0	3	19	51	51	6	10	0	0	2	0	12	0	0	0	21	20	2	0	12	0	25	37	5	4	9	63	11	202

ตารางที่ 6.1 (ต่อ) แสดงผลการรู้จำชุดการสอน

โมเดลตัวรักษ		ชุดอักษรที่ใช้หลอลอบ																																				
		118	135	136	125	146	147	134	127	129	139	122	112	122	118	133	119	112	135	123	107	114	111	111	109	111	111	117	118	104	106	106	99	115	119	113	211
		ภ	ช	ค	ม	ง	จ	ล	ย	ร	ณ	ฏ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ห	อ	ฮ	
	ห	45	75	37	12	22	4	38	0	0	13	6	7	1	16	44	16	21	116	0	30	36	25	27	32	40	43	32	26	8	8	3	5	6	5	4	0	
	ธ	28	3	45	16	5	14	15	92	100	32	77	61	29	36	16	10	29	3	110	1	3	3	56	56	6	2	59	5	52	81	6	7	46	78	32	192	
	น	28	78	34	45	48	60	97	79	29	38	67	65	58	75	61	48	47	55	31	105	65	44	42	51	47	54	40	15	31	36	29	21	44	33	48	82	
	บ	12	64	31	86	12	22	17	22	35	102	15	18	72	72	22	38	39	49	56	47	100	81	33	34	53	54	22	92	45	21	23	7	45	26	16	40	
	ป	34	81	68	58	24	48	40	64	66	70	40	39	67	78	78	88	73	111	49	34	95	108	57	61	70	80	50	83	68	22	31	23	93	74	44	84	
	ผ	5	41	16	24	22	27	29	25	38	58	19	35	58	32	14	25	6	33	23	28	36	24	106	95	41	43	28	33	71	2	34	7	13	36	32	92	
	ฝ	1	3	12	14	1	4	7	2	13	23	19	42	26	18	4	16	2	6	10	5	7	9	92	98	13	16	30	10	68	8	4	4	6	35	27	140	
	พ	33	84	33	88	66	71	47	32	59	87	28	35	92	66	50	43	23	48	13	38	50	31	46	53	108	98	30	94	32	31	44	4	25	51	12	12	
	ฟ	33	61	43	117	39	27	121	23	113	108	60	43	110	101	50	54	27	47	33	98	56	41	68	77	104	106	27	90	63	25	46	3	20	101	25	80	
	ภ	9	9	62	10	22	11	26	8	6	22	69	71	28	36	25	37	25	28	12	12	5	8	17	6	6	10	97	5	2	2	6	2	10	9	6	8	
	ม	10	22	21	105	20	20	51	30	68	108	19	18	83	68	40	30	36	30	26	28	8	12	22	16	63	68	13	114	14	14	53	0	10	81	17	34	
	ย	11	28	24	47	15	10	3	35	45	61	30	18	52	35	17	11	9	29	30	17	38	27	61	68	23	27	48	24	94	40	0	15	19	23	11	24	
	ร	2	19	36	32	27	20	27	45	33	15	64	66	42	11	35	33	7	18	59	14	7	4	40	28	18	7	74	17	26	105	4	6	9	18	12	134	
	ล	3	4	9	3	19	77	84	25	29	11	51	48	7	8	1	1	11	0	18	1	2	0	29	31	0	0	12	3	28	16	99	59	37	66	74	140	
	ว	1	2	18	7	11	34	64	32	27	20	38	52	7	2	9	5	2	8	46	8	10	5	14	15	3	1	31	11	23	76	27	97	28	29	100	138	
	ห	19	16	56	27	4	68	56	11	17	47	51	x	15	41	2	1	33	1	15	17	18	17	69	55	23	22	40	3	47	22	31	18	111	89	29	162	
	ห	18	22	21	10	8	6	36	21	25	19	44	64	22	20	7	12	14	2	15	8	47	36	14	13	10	9	28	1	24	14	20	24	16	113	20	58	
	อ	6	3	0	2	4	59	56	19	17	0	37	36	2	5	0	1	8	0	15	0	0	0	4	0	0	1	26	1	4	49	38	85	14	12	108	62	
	ฮ	0	0	5	1	0	10	8	0	3	19	51	51	6	10	0	0	2	0	12	0	0	0	21	20	2	0	12	0	25	37	5	4	9	63	11	202	

ตารางที่ 6.2 แสดงผลการรู้จำชุดการทดสอบชุดที่ 1

โมเดลตัวอักษร	ชุดอักษรที่ใช้ทดสอบ																																			
	จำนวน	91	96	87	98	96	97	94	98	98	100	99	96	100	101	100	100	99	100	105	100	100	101	101	100	100	100	100	102	107	106	104	105	103	105	102
		ก	ช	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ฬ	อ	ฮ
	ก	82	55	75	46	56	19	23	19	30	45	13	22	65	70	91	69	67	81	7	67	48	56	15	24	59	59	55	50	28	36	39	10	37	29	32	25
	ช	59	88	39	62	41	82	69	85	64	72	15	26	64	62	23	10	37	102	46	77	81	87	37	42	91	90	35	79	32	54	48	39	85	65	32	33
	ค	42	45	80	27	43	47	77	48	x	39	20	33	20	34	47	41	56	68	2	37	14	27	8	9	41	34	53	33	2	6	32	1	45	17	10	12
	ฆ	59	28	18	89	8	7	7	6	8	12	36	25	30	1	34	44	42	40	3	8	18	31	34	25	22	41	42	19	21	1	23	17	27	13	16	13
	ง	8	10	2	35	89	80	87	57	23	34	56	38	57	72	6	10	13	0	29	77	13	17	35	41	1	0	16	33	42	22	58	23	52	46	29	20
	จ	7	7	32	16	2	60	78	53	53	49	49	53	3	44	29	18	51	62	25	4	7	5	16	22	12	3	57	2	29	35	80	57	62	57	73	74
	ฉ	31	9	49	10	2	52	83	44	49	20	55	40	12	32	31	23	47	56	37	30	20	29	25	14	16	18	44	14	22	27	71	83	49	72	78	70
	ช	63	31	71	31	53	69	70	90	83	24	45	33	36	78	44	x	58	87	52	88	47	32	42	39	51	45	68	31	49	27	70	58	57	56	68	42
	ซ	26	71	21	53	31	69	70	90	93	24	45	20	26	31	33	27	11	12	59	74	29	28	51	27	36	52	39	32	49	37	65	23	27	41	48	31
	ฌ	35	55	17	48	38	60	75	56	45	71	47	33	76	79	20	47	33	68	47	88	59	52	82	77	57	51	38	58	68	12	69	92	45	21	80	18
	ฏ	46	70	62	62	41	19	30	74	61	64	97	79	71	32	77	71	41	98	82	37	58	60	55	56	69	65	50	63	66	69	18	44	47	48	16	64
	ฏ	33	14	11	3	8	2	7	10	13	3	10	77	5	10	16	16	10	45	31	17	31	18	11	9	18	18	8	9	14	13	3	0	22	24	2	24
	ฒ	58	65	64	95	34	71	25	45	58	59	26	29	81	88	69	55	56	73	35	83	84	91	95	94	93	92	53	77	94	59	84	78	37	63	84	58
	ณ	60	67	40	11	29	23	19	41	23	21	81	23	35	88	41	24	57	58	33	84	63	61	44	43	64	77	68	16	52	32	17	3	54	16	17	41
	ด	21	33	20	58	73	17	44	4	18	34	12	13	48	49	86	77	35	71	15	59	22	43	14	9	39	41	13	64	6	8	32	0	23	19	8	15
	ต	45	30	35	28	63	7	34	9	8	43	5	17	28	29	76	86	25	17	5	29	32	37	23	19	27	41	36	45	9	12	2	4	23	9	8	6
	ถ	74	63	55	38	38	15	18	57	46	62	44	6	51	67	65	74	92	93	30	51	69	65	70	63	65	66	78	49	45	29	17	8	53	25	20	30

ตารางที่ 6.2 (ต่อ) แสดงผลการรู้จำชุดการทดสอบชุดที่ 1

โมเดลตัวอักษร	ชุดอักขรที่ใช้ทดสอบ																																				
	จำนวน	91	96	87	98	96	97	94	98	98	100	99	96	100	101	100	100	99	100	105	100	100	101	101	100	100	100	100	100	102	107	106	104	105	103	105	102
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ฬ	อ	ฮ
	ห	31	61	12	17	18	2	51	0	13	23	6	15	2	18	4	5	13	98	0	58	42	31	25	27	47	39	19	25	12	6	4	5	2	7	9	1
	ธ	8	0	21	8	1	7	7	63	75	25	83	51	31	29	33	10	13	0	86	0	1	0	27	32	0	0	64	0	34	73	3	6	48	42	24	82
	น	2	41	2	32	17	21	89	80	34	22	40	31	64	74	16	29	19	28	30	95	44	28	34	41	27	46	26	7	25	42	17	34	16	46	26	37
	บ	2	22	16	74	5	4	2	12	19	69	6	15	55	46	2	35	15	9	31	22	85	76	32	39	52	46	22	75	42	19	23	11	43	33	12	5
	ป	25	36	37	47	22	16	25	53	57	39	24	33	54	59	71	73	64	91	34	23	78	95	52	61	56	64	55	82	60	25	32	22	88	57	51	50
	ผ	2	26	0	19	16	28	19	15	25	72	6	17	75	39	13	33	8	41	31	33	37	31	94	91	44	42	25	53	73	5	55	6	18	33	61	66
	ฝ	1	0	1	2	0	0	4	1	6	25	4	22	15	14	5	8	0	2	12	1	0	3	80	89	5	6	16	8	72	0	3	0	2	1	51	64
	พ	9	83	15	71	37	60	17	42	32	80	5	11	96	45	11	33	11	30	19	28	45	30	36	52	96	90	32	92	30	38	38	5	11	41	16	10
	ฟ	9	58	7	91	20	12	92	19	88	92	38	16	85	91	9	27	19	30	43	87	36	28	57	68	95	89	27	68	56	45	23	5	13	85	32	40
	ภ	20	2	63	6	6	1	16	3	1	18	75	67	20	43	23	29	24	6	9	9	0	2	0	1	4	2	82	3	6	0	2	0	5	1	6	11
	ม	3	11	1	86	5	14	24	9	50	83	1	10	75	41	12	19	10	26	16	11	1	1	17	16	82	77	18	94	19	5	43	1	13	81	18	16
	ย	2	6	12	25	5	3	1	21	31	40	13	13	44	37	7	10	4	8	14	11	37	25	51	58	16	18	43	24	68	34	0	22	22	21	11	7
	ร	1	35	18	51	28	30	41	57	51	4	49	65	33	7	22	16	7	16	80	20	3	10	13	21	20	8	55	6	21	94	6	0	5	6	6	56
	ล	8	0	3	0	1	40	56	32	34	4	39	38	0	11	0	0	16	0	4	2	0	0	17	21	0	0	17	0	21	6	88	61	33	59	77	56
	ว	1	2	4	9	11	21	67	29	46	17	36	29	2	2	2	2	1	1	66	7	2	3	5	9	0	0	58	2	24	89	50	103	19	15	91	63
	ส	0	0	1	1	0	3	6	13	14	4	1	11	1	5	1	0	10	0	0	0	0	0	1	0	0	0	3	0	0	1	0	0	3	2	1	43
	ห	31	8	24	37	2	26	22	13	20	65	31	0	10	56	2	1	53	0	8	6	18	18	59	58	13	25	59	3	34	16	17	10	99	87	23	74
	ฬ	10	4	13	5	5	8	39	10	24	15	55	36	1	20	4	3	14	0	9	9	31	35	16	13	1	4	32	0	23	23	27	31	16	100	23	41
	อ	6	2	0	1	2	56	67	28	26	0	39	10	1	1	0	0	15	0	6	1	0	1	0	0	0	0	43	1	3	42	47	95	6	11	98	43
	ฮ	1	0	0	1	0	3	2	1	3	23	31	38	0	19	0	0	4	0	12	0	0	0	7	7	0	0	27	0	11	34	0	0	7	39	3	91

ตารางที่ 6.3 แสดงผลการรู้จำชุดการทดสอบชุดที่ 2

โมเดลตัวอักษร	ตัวอักษรชุดทดสอบที่ 2																																			
		56	75	83	63	89	66	68	62	81	62	63	43	47	41	52	46	46	55	23	21	19	25	34	27	29	39	29	30	21	18	25	22	23	24	18	24
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฎ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ห	อ	ฮ
	ก	50	32	64	40	21	17	8	11	36	34	8	11	23	30	48	22	31	42	2	4	10	9	2	4	9	10	20	12	1	7	5	3	9	3	0	0
	ข	29	64	32	36	44	30	23	41	59	27	41	33	21	25	15	7	20	23	13	11	13	17	15	13	25	27	11	20	8	7	15	9	9	16	8	8
	ค	21	15	72	12	14	23	30	7	9	19	19	16	18	17	35	35	24	39	0	4	5	3	1	2	7	6	13	3	3	1	13	4	6	9	5	4
	ฆ	32	17	16	53	12	7	1	7	10	5	8	3	2	2	3	10	20	18	0	1	5	5	9	11	9	11	16	2	6	1	1	2	5	1	1	4
	ง	7	22	14	20	80	53	54	38	36	21	44	31	23	29	11	12	10	12	9	15	10	8	13	11	9	11	6	9	12	5	19	3	10	11	8	12
	จ	5	33	35	13	8	59	57	14	24	14	54	36	8	4	9	12	12	24	14	0	1	0	14	10	12	3	4	2	6	3	18	6	16	24	12	19
	ล	12	17	25	9	17	34	62	14	26	12	28	21	6	6	21	13	3	23	7	2	1	4	3	1	1	3	5	3	2	5	8	14	6	1	11	3
	ช	40	48	43	38	37	30	39	51	51	29	42	25	28	20	13	12	26	26	13	13	7	13	9	11	15	17	17	8	7	8	17	9	13	12	8	13
	ซ	14	34	20	44	26	11	0	54	78	0	0	0	4	2	5	37	9	0	2	5	7	1	4	0	0	0	2	1	6	4	0	0	0	0	1	2
	ฌ	29	31	40	39	31	26	25	16	21	56	24	9	25	27	31	19	28	25	9	16	7	6	11	16	14	8	10	11	11	7	8	11	8	2	8	8
	ญ	1	9	2	23	1	1	13	10	15	12	2	2	25	4	2	2	2	0	5	7	2	4	3	3	10	16	0	2	0	7	3	0	4	5	0	2
	ฏ	30	54	33	54	50	6	15	43	63	45	50	30	37	16	32	21	25	39	10	11	12	11	11	10	20	20	15	11	11	9	2	8	4	17	4	10
	ฏ	17	15	18	11	2	1	2	5	10	5	1	25	4	1	10	4	5	15	1	3	1	4	0	2	8	3	1	1	1	1	1	0	1	1	0	0
	ฐ	0	0	0	1	0	0	0	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	ฑ	26	46	51	48	28	40	35	25	29	50	37	21	39	25	35	30	24	36	19	15	12	22	31	22	24	34	12	26	13	14	20	18	10	14	13	13
	ฒ	33	31	35	12	34	37	29	40	56	11	42	0	14	24	14	12	24	31	3	14	7	13	11	10	11	18	15	7	7	8	8	3	14	20	3	12
	ด	21	19	54	28	37	4	25	10	21	37	10	0	24	23	45	34	21	42	9	11	7	12	2	2	2	7	8	15	1	1	11	4	3	2	0	3
	ต	11	2	3	2	4	0	2	5	0	0	0	0	7	7	44	43	2	1	1	2	8	5	5	5	5	3	3	0	11	2	1	5	7	2	0	0
	ถ	33	37	33	17	35	22	12	22	31	14	31	16	18	21	22	12	29	34	9	9	2	11	4	14	15	27	16	7	9	12	4	1	5	16	1	10
	ท	24	21	25	3	16	4	7	0	0	2	2	0	0	4	19	19	7	47	1	0	2	3	1	4	4	10	4	1	0	3	3	0	1	0	0	0

ตารางที่ 6.3 **(ต่อ)** แสดงผลการรู้จำชุดการทดสอบชุดที่ 2

โมเดลตัวอักษร	ตัวอักษรชุดทดสอบที่ 2																																			
		56	75	83	63	89	66	68	62	81	62	53	43	47	41	52	46	46	55	23	21	19	25	34	27	29	39	29	30	21	18	25	22	23	24	18	24
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ฬ	ฮ
	ก	6	5	35	9	7	2	9	45	57	12	31	31	8	7	5	4	18	0	21	0	2	5	19	10	10	5	11	2	14	12	3	3	7	23	11	19
	ข	11	49	28	14	41	39	36	28	20	8	45	32	6	27	15	10	17	17	3	20	9	13	15	10	5	5	6	8	5	7	7	1	8	1	6	7
	ค	2	36	24	42	13	8	11	18	14	43	8	9	26	25	10	10	16	21	7	10	14	14	9	5	4	14	6	21	4	10	5	0	5	1	7	7
	ฆ	9	52	41	38	21	30	25	26	25	26	31	16	20	24	14	25	28	41	17	8	11	20	17	12	19	27	14	18	14	8	5	5	14	19	7	17
	ง	4	19	22	19	8	4	20	13	30	21	16	12	12	9	2	2	3	8	5	11	6	7	33	25	17	20	7	9	12	1	5	2	8	10	5	6
	จ	0	0	13	7	1	0	6	0	10	11	13	23	15	4	2	6	4	3	5	0	3	2	20	21	9	13	6	4	10	0	1	0	6	15	1	18
	ฉ	18	47	29	35	41	20	31	13	30	50	20	15	28	18	27	21	22	26	2	7	7	8	13	13	27	28	6	13	2	5	14	0	6	6	2	4
	ช	17	31	44	61	47	19	52	17	74	55	33	27	47	30	29	17	22	26	15	19	14	15	13	16	23	28	5	14	7	5	19	0	3	19	4	8
	ซ	8	7	37	7	18	8	28	2	7	15	31	22	15	7	17	15	15	17	0	10	1	3	2	0	2	3	24	4	0	0	3	3	4	0	2	0
	ฌ	5	16	11	44	31	13	25	26	25	38	19	16	21	15	11	11	21	10	13	5	8	4	8	5	4	6	8	28	1	11	13	0	2	3	2	6
	ญ	5	17	17	31	16	1	1	21	36	20	10	5	27	10	6	2	7	8	9	6	8	11	22	18	16	18	9	14	17	7	1	0	4	4	3	11
	ร	1	6	25	10	12	1	7	20	8	15	27	16	12	4	7	9	8	8	10	1	2	0	11	8	5	7	21	9	9	13	4	3	5	9	4	20
	ล	1	3	11	0	7	45	56	6	7	5	20	14	0	0	0	1	1	0	4	0	0	0	12	8	1	0	2	2	7	2	19	8	10	13	8	13
	ว	0	0	22	5	11	29	28	15	4	17	13	15	7	0	20	6	3	8	12	1	1	1	12	10	5	4	3	2	10	7	6	18	11	12	15	17
	ส	0	0	5	0	1	1	1	0	0	5	40	38	1	0	0	0	1	0	8	1	1	0	0	1	6	1	1	1	0	2	1	0	2	20	0	2
	ห	4	4	37	7	5	23	27	1	1	10	34	0	9	9	4	5	9	0	10	3	3	3	13	13	17	12	4	1	5	4	12	5	21	21	8	17
	ฮ	3	15	18	10	11	0	10	12	12	8	14	24	7	6	3	6	2	3	6	3	5	7	7	8	13	14	9	1	2	2	4	1	6	22	3	4
	ฬ	1	5	0	1	2	21	8	0	2	0	15	15	0	0	1	0	0	0	11	0	0	0	3	0	1	2	0	0	3	11	6	7	2	3	12	6
	อ	0	0	9	0	0	4	6	0	1	1	20	16	1	0	0	0	0	0	7	0	0	0	4	8	4	1	2	0	7	2	3	4	2	14	7	20

ตารางที่ 6.4 แสดงผลการรู้จำชุดการทดสอบชุดที่ 3

โมเดลตัวอักษร	ตัวอักษรชุดทดสอบที่ 3																																			
	จำนวน	59	60	62	60	59	60	59	59	59	65	58	61	55	62	54	59	61	61	60	59	63	61	61	59	62	61	64	63	68	63	64	61	64	72	64	60
		ก	ข	ค	ฆ	ง	จ	ล	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ห	จ	ฮ
	ก	57	29	58	27	6	16	21	41	18	23	15	18	21	56	50	44	56	47	11	17	47	39	28	26	38	35	53	41	17	12	30	2	23	6	2	0
	ข	23	55	24	29	40	42	19	29	19	40	35	27	39	38	4	10	26	26	16	15	58	47	26	9	48	46	19	48	27	32	13	15	37	42	12	8
	ค	36	7	62	15	16	34	31	8	x	18	19	20	13	26	15	20	40	20	7	19	36	40	29	20	45	47	45	10	3	4	18	11	38	46	9	3
	ฆ	44	17	34	48	2	4	5	6	4	7	3	8	5	3	20	10	21	29	9	0	25	16	25	30	18	14	19	7	22	0	9	6	12	13	14	9
	ง	1	9	6	26	51	48	54	50	45	49	41	50	39	51	10	22	16	3	24	55	34	19	12	15	13	5	7	7	34	13	42	15	43	28	36	9
	จ	15	3	13	4	4	46	57	9	8	1	40	18	1	2	12	11	2	47	1	3	8	2	19	26	5	3	1	2	23	27	47	16	41	60	51	24
	ล	5	4	30	10	8	22	49	16	21	18	16	34	7	9	24	20	30	6	30	3	14	7	4	5	0	1	15	7	9	24	26	31	36	22	24	17
	ช	48	22	25	18	46	44	37	41	40	16	32	23	16	42	27	x	43	55	33	42	23	26	35	30	38	29	48	10	34	30	28	34	36	38	31	25
	ซ	0	55	23	21	1	35	3	44	56	7	0	0	0	22	14	17	1	21	47	33	20	15	3	7	8	3	6	23	11	0	5	16	14	22	19	21
	ฌ	21	50	31	34	16	32	46	9	6	48	8	3	28	46	26	17	17	30	22	51	38	45	43	41	34	29	23	33	45	4	44	41	24	4	56	6
	ฏ	21	27	25	24	22	6	2	23	28	13	48	21	15	20	16	16	20	27	22	7	45	46	39	46	40	44	26	40	53	25	6	3	21	44	6	35
	ฏ	20	12	6	0	0	2	2	1	1	3	0	53	5	0	3	7	4	24	5	12	13	20	5	7	11	6	0	2	3	0	2	0	4	6	1	2
	ฌ	31	47	39	60	18	36	14	24	31	62	22	16	49	56	23	22	26	21	16	47	59	61	49	55	31	43	21	59	67	30	42	43	22	24	47	36
	ฌ	37	37	33	9	11	11	6	43	31	17	42	0	21	46	22	16	24	34	21	42	29	34	45	35	48	56	39	7	45	14	7	14	23	49	7	43
	ด	24	20	22	27	15	5	47	5	24	20	16	0	40	27	49	55	33	40	5	41	17	22	23	12	27	35	38	29	3	0	28	1	21	29	2	0
	ต	13	7	2	45	7	0	11	2	21	5	9	0	44	32	40	51	21	2	12	0	0	1	6	6	3	5	2	8	19	3	12	2	4	13	7	7
	ถ	39	48	9	36	18	10	7	17	17	39	22	x	39	23	28	36	48	41	23	17	32	39	39	36	50	52	40	32	36	38	16	15	24	39	2	30
	ท	9	23	13	2	2	2	10	0	x	4	0	0	4	10	5	6	8	42	1	5	12	6	31	28	15	18	17	5	1	0	3	4	1	0	1	0

ตารางที่ 6.4 (ต่อ) แสดงผลการรู้จำชุดการทดสอบชุดที่ 3

โมเดลตัวอักษร	ตัวอักษรชุดทดสอบที่ 3																																				
	จำนวน	59	60	62	60	59	60	59	59	59	65	58	61	55	62	54	59	61	61	60	59	63	61	61	59	62	61	64	63	68	63	64	61	64	72	64	60
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ต	ถ	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ห	อ	ฮ
	ธ	30	0	27	7	0	0	2	55	53	29	37	25	13	32	0	4	24	5	52	2	1	0	49	49	2	1	33	4	44	57	0	3	36	58	19	60
	น	32	38	26	36	25	2	39	25	9	32	21	37	18	26	30	46	43	38	12	54	30	24	27	31	50	45	21	8	16	19	30	4	27	14	27	27
	บ	19	49	26	48	5	14	12	19	28	59	7	8	41	60	32	33	38	36	53	37	54	47	25	20	37	35	9	54	32	4	20	2	15	5	17	19
	ป	28	41	32	22	5	11	22	27	36	37	23	16	31	52	22	49	36	47	26	22	50	58	43	39	51	57	17	47	48	13	17	10	51	38	21	21
	ผ	3	26	8	10	16	12	19	20	18	16	15	28	27	12	17	10	0	5	1	19	15	4	52	53	17	17	11	6	48	1	4	2	4	14	2	5
	ฝ	0	5	5	12	0	4	5	3	4	6	18	18	24	15	3	8	0	2	0	11	7	7	55	52	15	14	6	2	46	13	3	0	1	34	2	30
	ห	11	32	5	46	39	31	31	4	45	36	17	32	48	45	9	27	0	5	0	25	39	28	37	25	58	52	11	44	23	13	30	3	13	30	5	3
	ฟ	20	30	13	54	7	0	58	10	48	51	24	29	52	59	10	40	0	0	4	56	40	33	48	45	60	60	12	51	47	6	40	0	0	60	19	16
	ภ	1	6	21	3	13	11	15	8	7	5	34	45	21	10	27	29	3	23	6	18	7	6	24	6	8	10	52	1	0	2	4	1	9	13	0	0
	ม	13	13	9	52	3	19	32	11	53	56	10	10	43	58	10	16	30	0	19	20	13	17	11	15	23	33	6	61	5	6	50	0	2	42	11	12
	ย	10	28	6	25	9	1	0	17	21	31	15	9	36	19	3	8	5	21	15	8	15	8	29	35	6	5	20	8	60	26	1	10	14	6	6	6
	ร	0	5	25	3	8	8	9	17	2	6	47	28	30	6	32	34	4	4	20	6	4	1	31	24	4	0	48	9	21	58	0	8	11	20	3	39
	ล	0	3	5	3	23	43	46	13	6	10	33	33	11	6	2	0	7	0	16	2	5	0	20	19	0	0	2	3	26	17	63	35	25	48	30	47
	ว	0	3	12	0	1	12	28	21	4	2	27	44	2	1	2	0	0	1	9	6	18	5	10	11	3	0	6	19	13	44	10	57	22	19	57	47
	ห	8	10	34	6	3	14	24	6	11	18	22	x	5	20	0	0	6	1	3	18	3	1	46	40	11	9	9	0	43	17	16	22	55	33	12	47
	ห	19	21	9	10	1	1	10	21	16	15	14	27	19	7	7	19	15	1	7	3	27	23	5	3	1	4	8	1	15	2	6	11	6	57	2	7
	อ	1	0	0	0	5	31	34	2	3	0	24	28	2	3	0	1	2	0	0	0	0	1	1	3	2	1	2	0	1	39	23	51	14	9	63	5
	ฮ	0	0	3	1	0	2	2	0	0	11	40	34	7	3	0	0	1	0	7	0	0	0	28	19	0	0	1	0	28	32	0	2	7	52	6	53

ตารางที่ 6.5 แสดงผลการรู้จำชุดการทดสอบชุดที่ 4

โมเดลตัวอักษร	ตัวอักษรชุดทดสอบที่ 4																																		
	จำนวน	33	31	35	28	49	64	47	32	14	49	26	25	41	34	74	34	17	48	56	34	45	36	25	35	32	31	44	43	17	22	19	16	43	40	40	26
		ก	ข	ค	ฆ	ง	จ	ล	ช	ซ	ฌ	ฎ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ฬ	อ
	ก	30	11	33	16	2	12	9	22	5	18	4	2	21	24	66	23	14	28	2	6	10	11	11	8	12	19	40	18	3	3	5	1	12	9	0	0
	ข	24	30	25	13	8	57	37	29	8	36	18	9	23	10	61	17	13	42	41	25	43	33	8	15	23	22	16	37	3	6	18	3	31	34	14	4
	ค	16	9	35	14	6	4	14	9	x	13	2	11	21	22	42	22	4	28	1	12	8	15	5	5	12	16	10	5	1	4	3	0	10	15	1	0
	ฆ	15	13	3	5	1	12	13	7	0	3	6	2	3	1	7	4	7	16	4	0	8	16	9	10	8	4	17	3	5	4	3	2	14	8	19	13
	ง	7	9	7	14	40	23	23	20	4	18	19	9	18	13	32	13	5	2	7	16	15	5	2	2	3	2	2	19	3	2	13	4	23	5	1	1
	จ	0	13	6	5	5	58	27	10	2	9	18	16	19	9	0	5	4	1	5	1	4	8	13	21	20	21	1	13	7	2	10	7	26	37	16	23
	ล	1	10	0	6	6	12	41	14	4	12	11	10	6	8	1	1	0	2	13	5	3	9	5	12	3	5	3	2	3	7	11	8	6	18	29	8
	ช	11	15	17	11	31	43	24	22	13	26	14	15	20	22	35	x	4	16	17	20	20	17	22	28	19	24	24	20	8	9	14	11	20	24	18	11
	ซ	13	23	11	15	2	32	12	27	14	0	0	0	32	12	5	26	6	0	43	0	12	11	2	4	4	6	2	11	11	1	4	x	12	0	0	0
	ฌ	23	20	11	19	12	28	16	18	5	38	18	17	25	27	50	15	11	35	31	33	37	27	23	24	23	17	36	31	13	18	9	12	26	25	32	9
	ฏ	21	28	25	24	30	45	11	16	9	37	25	13	32	26	48	27	13	33	26	30	33	30	21	29	28	27	26	37	16	19	9	14	18	31	12	24
	ฐ	12	9	2	0	6	12	8	2	2	7	2	1	3	7	5	9	1	11	3	9	14	9	4	11	6	5	3	6	0	2	2	0	9	4	0	0
	ฒ	25	27	29	26	9	34	20	15	7	41	16	10	33	18	62	19	16	43	23	7	41	32	19	30	28	25	39	42	11	8	16	14	26	27	37	21
	ณ	24	9	21	9	7	36	22	6	5	17	9	0	15	24	33	17	14	35	35	24	22	18	9	9	13	8	33	5	1	12	11	10	19	19	13	6
	ด	22	8	31	17	24	3	23	8	1	15	5	0	14	15	72	28	9	45	9	6	8	17	2	1	18	22	23	28	3	1	4	1	13	24	0	2
	ต	14	2	25	4	0	12	2	0	7	8	0	0	30	11	61	34	7	2	3	8	0	0	2	1	2	4	4	3	1	1	0	0	6	0	2	6
	ถ	29	22	27	5	24	48	17	15	5	26	18	x	7	26	52	11	16	42	14	25	37	32	24	30	25	21	39	9	11	3	6	5	19	26	13	7
	ท	19	23	24	1	5	3	14	1	x	4	1	0	3	4	47	8	8	31	0	1	11	24	1	4	12	8	16	13	1	1	1	0	6	2	0	0

ตารางที่ 6.5 **(ต่อ)** แสดงผลการรู้จำชุดการทดสอบชุดที่ 4

โมเดลตัวอักษร	ตัวอักษรชุดทดสอบที่ 4																																			
	จำนวน	33	31	35	28	49	64	47	32	14	49	26	25	41	34	74	34	17	48	56	34	45	36	25	35	32	31	44	43	17	22	19	16	43	40	40	26
		ก	ข	ค	ฆ	ง	จ	ล	ช	ซ	ฌ	ฏ	ฏ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ล	ว	ห	ห	อ	ฮ
	ธ	3	0	10	4	0	19	8	22	10	0	3	15	10	3	0	0	1	0	51	0	0	0	9	21	0	0	7	0	10	16	4	5	8	31	16	26
	น	9	22	14	10	5	44	23	20	5	7	17	23	20	25	61	14	15	37	22	27	34	22	11	10	15	18	25	6	8	3	12	9	27	7	33	3
	บ	8	18	6	16	1	13	7	8	3	36	4	7	22	15	2	9	3	21	15	6	37	31	8	12	19	17	7	34	3	1	2	2	11	8	3	6
	ป	7	30	16	15	3	34	24	24	7	34	11	13	30	21	52	26	14	39	26	6	36	32	12	17	16	22	14	31	11	2	12	4	31	30	16	11
	ผ	0	5	2	1	1	9	7	2	1	4	2	16	2	6	0	2	2	17	0	3	12	5	22	28	8	4	26	2	9	1	3	0	3	13	0	6
	ฝ	0	0	1	4	0	2	1	0	2	0	2	24	0	0	0	3	2	5	0	0	0	0	25	29	0	5	34	0	11	1	1	5	1	10	0	22
	ห	25	12	24	21	6	21	18	5	7	17	10	13	15	25	50	17	15	39	0	12	16	3	8	10	29	29	20	33	6	1	2	0	16	23	0	1
	ฟ	23	12	23	25	6	21	43	6	9	25	18	20	26	32	61	23	17	46	0	30	14	4	16	28	30	29	23	38	12	1	4	1	18	38	4	13
	ภ	0	3	2	8	1	0	1	4	2	4	1	9	2	18	1	4	1	6	2	0	0	1	3	1	0	3	18	1	0	2	1	0	1	0	0	0
	ม	1	1	20	26	1	1	18	7	8	34	4	2	25	14	53	21	12	26	2	7	0	2	5	6	16	12	0	40	1	5	7	1	6	30	0	1
	ย	5	6	9	9	1	10	2	6	8	15	14	18	5	11	20	9	3	23	17	7	22	4	17	17	15	12	21	7	14	1	1	1	5	16	2	6
	ร	3	0	8	3	0	2	0	5	0	3	3	21	12	2	12	4	1	1	9	1	1	4	14	15	3	1	27	7	6	19	0	5	0	2	7	19
	ล	0	0	0	0	6	26	19	6	2	1	8	11	2	2	0	0	0	0	12	0	0	0	5	7	0	0	6	1	3	3	17	6	8	20	25	18
	ว	0	1	1	0	2	2	1	3	1	2	1	13	2	2	4	1	0	1	4	0	0	0	0	1	0	0	0	3	2	3	0	11	7	3	36	17
	ห	2	8	17	3	0	54	31	5	0	1	7	x	2	3	0	0	2	0	6	0	11	18	8	10	1	1	11	1	6	1	12	6	35	34	10	23
	ห	2	5	1	1	0	2	17	5	0	4	3	14	10	4	0	1	1	1	16	2	25	4	1	5	1	1	6	0	3	5	3	6	3	35	10	3
	อ	0	0	0	0	2	14	6	9	2	2	0	4	1	2	1	0	0	0	8	0	0	0	0	0	0	0	0	0	2	4	5	14	8	2	38	2
	ฮ	0	0	0	0	0	17	7	1	0	0	6	16	0	3	0	0	0	0	1	0	0	0	1	9	0	0	0	0	5	1	3	2	4	20	6	21

6.4 ผลการทดลอง

จากการทดสอบการรู้จำของแกรมม่า โดยทดสอบกับชุดที่ใช้สอนจำนวน 1 ชุดและชุดทดสอบจำนวน 4 ชุด ให้ผลดังตารางตารางที่ 6.1 ถึง ตารางที่ 6.5 ตามลำดับ โดยสามารถสรุปเป็นผลการรู้จำได้ดังตารางที่ 6.6

ตารางที่ 6.6 แสดงผลการรู้จำของแกรมม่า

	ชุดตัวอักษรที่ใช้ในการทดลอง	จำนวนตัวอักษรทั้งหมด	โมเดลแบบ CFG: จำนวนที่รู้จำถูก	(%)	จำนวนตัวอักษรทั้งหมด	โมเดลแบบ HMM: จำนวนที่รู้จำถูก	(%)
ตัวอักษรไทย	Train character	4654	4563	98.05	4654	4563	98.05
	Test character	4265	3444	80.75	4265	3122	73.19
	All character test	8919	8007	89.4	8919	7685	86.08

หมายเหตุ จำนวนตัวอักษรที่มีในชุดสอนและชุดทดสอบในแต่ละวิธีอาจจะไม่เท่ากัน เนื่องจากว่าได้ตัดตัวอักษรบางตัวออกไป

6.5 การวิเคราะห์

จากผลการทดลองการรู้จำอักษรภาษาไทยโดยใช้โมเดลทางภาษาแบบคอนเท็กฟรีแกรมม่าดังที่ได้นำเสนอในวิทยานิพนธ์นี้ จะเห็นว่าผลการรู้จำที่ได้ในตารางที่ 6.1 มีเปอร์เซนต์การรู้จำที่ใกล้เคียงกับการรู้จำโดยใช้ Hidden makov model ซึ่งเป็นโมเดลที่ใช้อย่างแพร่หลายในด้าน speech recognition เปอร์เซนต์การรู้จำที่ได้ยังไม่เป็นที่น่าพอใจนัก ทั้งนี้อาจมีสาเหตุจากคอนเท็กฟรีแกรมม่าที่สร้างมีความลักษณะทั่วไปมากเกินไป ทำให้ยอมรับสตริงอื่นเข้ามาด้วย

ข้อผิดพลาดของการรู้จำตัวอักษรโดยใช้คอนเท็กฟรีแกรมม่าที่เกิดขึ้น สามารถวิเคราะห์ความผิดพลาดที่เกิดได้ดังนี้

1. คอนเท็กฟรีแกรมม่านี้ยังมีจุดอ่อนคือ ไม่สามารถแยกแยะตัวอักษรที่มีลักษณะที่ใกล้เคียงกัน เช่นตัวอักษรที่มีความแตกต่างกันเพียงแค่ความยาวของหาง ซึ่งมีค่าของลำดับเชนโค้ดที่ใกล้เคียงกัน นั่นคือไม่สามารถแยก "บ- ป", "ผ-ฝ" และ "พ-ฟ" ได้ ดังตัวอย่างในรูปที่ 6.3

(a) บ-ป

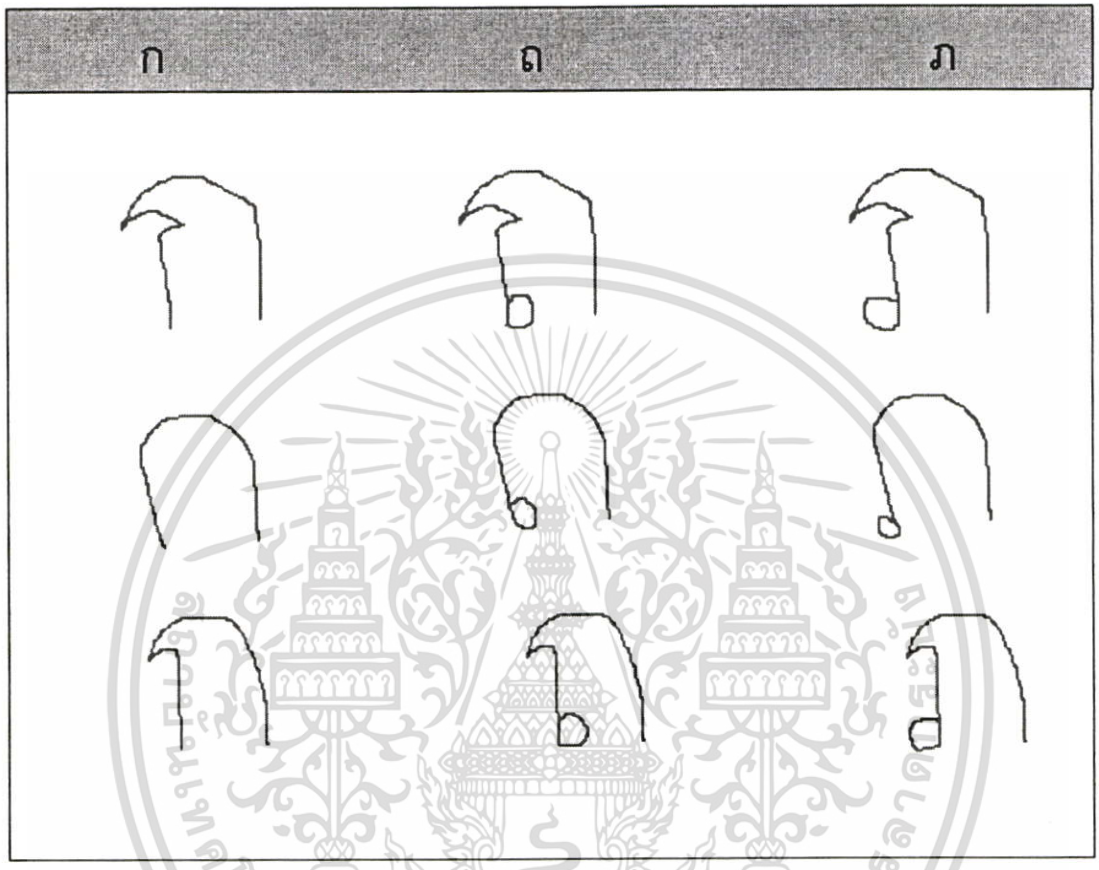
(b) ผ-ฝ

(c) พ-ฟ

รูปที่ 6.3 แสดงตัวอักษรที่ใกล้เคียงกันแต่แตกต่างกันเพียงความยาว

จากรูป 6.3 แสดงตัวอักษรที่โมเดลแบบคอนเท็กฟรีแกรม่าที่สร้างขึ้นมาไม่สามารถแยกได้ว่าเป็นอักษรตัวใด เพราะคอนเท็กฟรีแกรมม่าแสดงในรูปแบบของกฎเพื่อดีไรว์เซนโค้ด ตัวอักษรเหล่านี้มีลักษณะที่คล้ายกัน แตกต่างกันเพียงความยามของหางเท่านั้น กฎในแกรมม่าที่ได้ไม่สามารถระบุได้ว่าตัวอักษรเหล่านี้ต้องมีเซนโค้ดในส่วนของหางยาวเท่าใด เมื่อทำการดีไรว์เพื่อให้ได้เทอร์มินอลหรือเซนโค้ดที่เป็นส่วนหางนี้จึงไม่สามารถบอกความแตกต่างได้

2. การรู้จำผิดพลาดในกรณีที่ตัวอักษรนั้นๆ มีลักษณะเซนโค้ดที่เป็นสับเซตของตัวอักษรอีกตัวหนึ่ง เช่น "ก", "ถ" และ "ภ" โดยตัวอักษรทั้ง 3 ตัวนี้มีลักษณะเซนโค้ดที่คล้ายกัน คือ มีส่วนเซนโค้ดที่เป็น "ก" เป็นสับเซตปรากฏอยู่ในทั้ง "ถ" และ "ภ" ดังรูป 6.4



รูปที่ 6.4 แสดงตัวอักษรที่มีส่วนของเซนโค้ดเป็นสับเซตกัน

จากรูป 6.4 แสดงลักษณะตัวอักษรที่มีส่วนของเซนโค้ดที่เป็นสับเซตกัน โดยทั้ง "ถ" และ "ภ" มีส่วนที่ต่อจากหัวของพยัญชนะทั้ง 2 เหมือนกับ "ก" ดังนั้นส่วนของเซนโค้ดที่ต่อจากเซนโค้ดส่วนหัวของพยัญชนะทั้ง 2 จะเหมือนกัน การรู้จำจึงอาจผิดพลาดได้ การแก้ไขสามารถทำได้โดยทำตั้งแต่การเลือกตัวสอน เช่นในการเลือกตัวสอนสำหรับ "ถ" และ "ภ" นั้นเมื่อเริ่มจาก "S0" จะต้องมีเซนโค้ดที่เป็นส่วนหัวชัดเจน แต่ทั้งนี้ก็ขึ้นกับลายมือเขียนที่นำมาทดสอบด้วย ผู้เขียนบางคนอาจเขียนพยัญชนะทั้ง 2 แบบไม่มีหัว หรือเขียนหัวไม่ชัด ข้อผิดพลาดก็อาจเกิดขึ้นได้

3. การรู้จำผิดพลาดในกรณีที่ตัวอักษรนั้นมีค่าลำดับของเซนโค้ดที่ใกล้เคียงกัน เนื่องจากคอนเท็กฟรีแกรมม่าไม่สามารถบอกได้ว่าเมื่อถึงลำดับของเซนโค้ดที่เท่าใดแล้วควรจะเปลี่ยนจากเส้นตรงเป็นเส้นเฉียง ดังตัวอย่างตัวอักษรในรูป 6.5

รูปที่ 6.5 แสดงตัวอักษรที่ผิดเพราะลำดับเซนโค้ดที่ใกล้เคียงกัน

จากรูป 6.5 "ฑ" และ "ต" มีลักษณะของ การหยักต่อจากส่วนหัวพยัญชนะคล้ายกัน และพยัญชนะ "ป" กับ "ช" หากส่วนที่หยักของ "ช" ไม่ชัดเจนหรือหยักน้อยจนเกือบจะเป็นเส้นตรงคล้าย "ป" ลำดับของเซนโค้ดของตัวอักษรเหล่านี้จึงมีความใกล้เคียงกันทำให้การรู้จำผิดพลาดได้

4. การรู้จำผิดพลาดแบบที่คาดไม่ถึง ทั้งนี้อาจเป็นเพราะลักษณะของการเขียนนั้นไม่ชัดเจนซึ่งหากเป็นการบอกด้วยมนุษย์ก็อาจไม่แน่ใจว่าตัวอักษรนั้นคือตัวอักษรใด ดังรูป 6.6

รูปที่ 6.6 แสดงตัวอักษรที่ผิดแบบคาดไม่ถึง

5. การรู้จำผิดพลาดเพราะน่าจะผิดเพราะลักษณะของเซนโค้ดของตัวอักษรมีความคล้ายคลึงกันมาก ดังรูป 6.7

รูปที่ 6.7 แสดงตัวอักษรที่รู้จำผิดพลาดเพราะน่าจะผิด

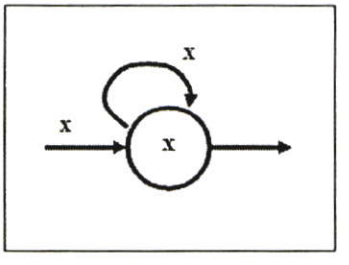
ตัวอักษรจากรูป 6.7 เป็นตัวอักษรที่รู้จำผิดพลาดเพราะน่าจะผิด เช่น "ธ" และ "ร" จะเห็นว่าหากมองด้วยตามนุษย์ก็ยากที่จะบอกได้ว่าเป็นตัวอักษร "ธ" หรือ "ร"

จากข้อผิดพลาดที่เกิดขึ้นข้างต้นสามารถลดปริมาณการเกิดข้อผิดพลาดได้ ทั้งนี้ขึ้นกับกลุ่มตัวอย่างเซนโค้ดที่ใช้เป็นตัวสอนด้วย หากกลุ่มตัวอย่างที่นำมาสอนนี้มีลักษณะของตัวอักษรนั้นๆครบและครอบคลุม แกรมม่าที่สร้างขึ้นมาก็จะเป็นแกรมม่าที่มีความแม่นยำและมีประสิทธิภาพในการรู้จำดี แต่หากกลุ่มตัวสอนไม่ครบคลุมความผิดพลาดในการรู้จำก็มีโอกาสเกิดขึ้นสูงเช่นกัน

6.6 กรณีที่โมเดลเกิด Over Generalized

การสร้างและการสอนโมเดลทางภาษาตามหลักการในวิทยานิพนธ์นี้ มีวัตถุประสงค์เพื่อเพิ่มลักษณะทั่วไป (Generalized) ของตัวอักษรให้กับโมเดล แต่อาจมีบางกรณีที่ทำให้โมเดลมีลักษณะดังกล่าวนี้มากเกินไปหรือที่เรียกว่า Over Generalized ดังนี้

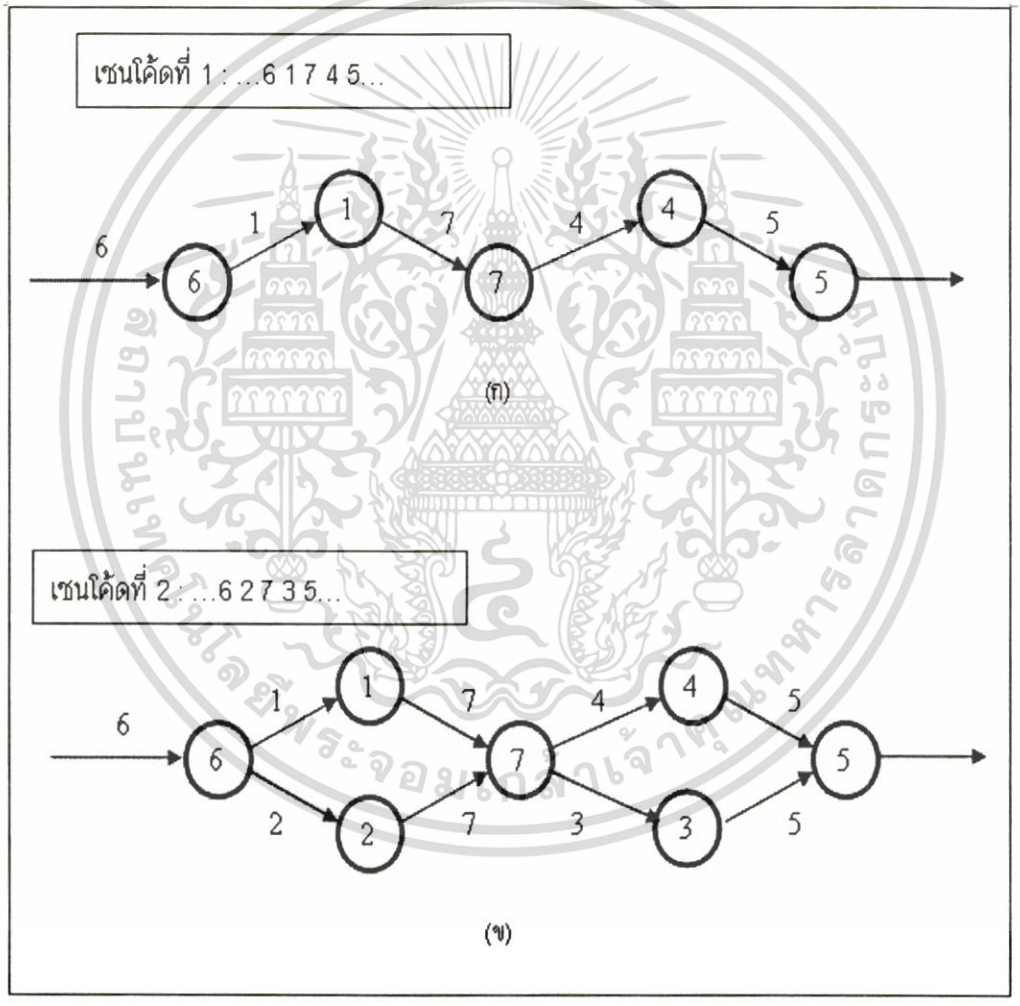
1. การสร้างโมเดลที่แสดงลำดับตามแม่แบบลูป ที่สร้างขึ้นเพื่อรองรับการกระจายแบบวนกลับมายังสตริงตัวเดิมเพื่อแทนลำดับของสตริงแบบ $x*$ โดย x คือ ทิศทางของเซนโค้ดซึ่งมีค่าได้ตั้งแต่ 1, 2, 3,..., 8 ดังรูป 6.8



รูปที่ 6.8 แสดงการสร้างโมเดลตามแม่แบบลูปเพื่อแทนลำดับของสตริง $x*$

การสร้างโมเดลดังกล่าวนี้ ทำให้การแสดงการรู้จำไม่สามารถแยกแยะตัวอักษรที่มีลักษณะที่ใกล้เคียงกัน เช่น ตัวอักษรที่มีความแตกต่างกันเพียงแค่ความยาวของหาง นั่นคือ ไม่สามารถแยก "บ- ป", "ผ-ฝ" และ "พ-ฟ" ได้

2. กรณีการสอนโมเดลที่ทำให้เกิดการเพิ่มลักษณะทั่วไปที่มากเกินไป (Over generalized) ในลักษณะแนวราบดังรูปที่ 6.9 โดยเริ่มจากการสร้างโมเดลจากส่วนของสตริงสายแรกคือ ..., 6, 1, 7, 4, 5,... ดังโมเดล (ก) จากนั้นทำการสอนด้วยส่วนของสตริงสายที่สองคือ คือ ..., 6, 2, 7, 3, 5,... ซึ่งจะทำการเพิ่มโหนดใหม่เข้าไปในโมเดลเดิมในเชิงแนวราบนั่นคือจาก 6 เพิ่มสเตจสำหรับ 2 และจาก 7 เพิ่มสเตจสำหรับ 3 ต่อ ตามลำดับ ดังโมเดล 6.9 (ข)



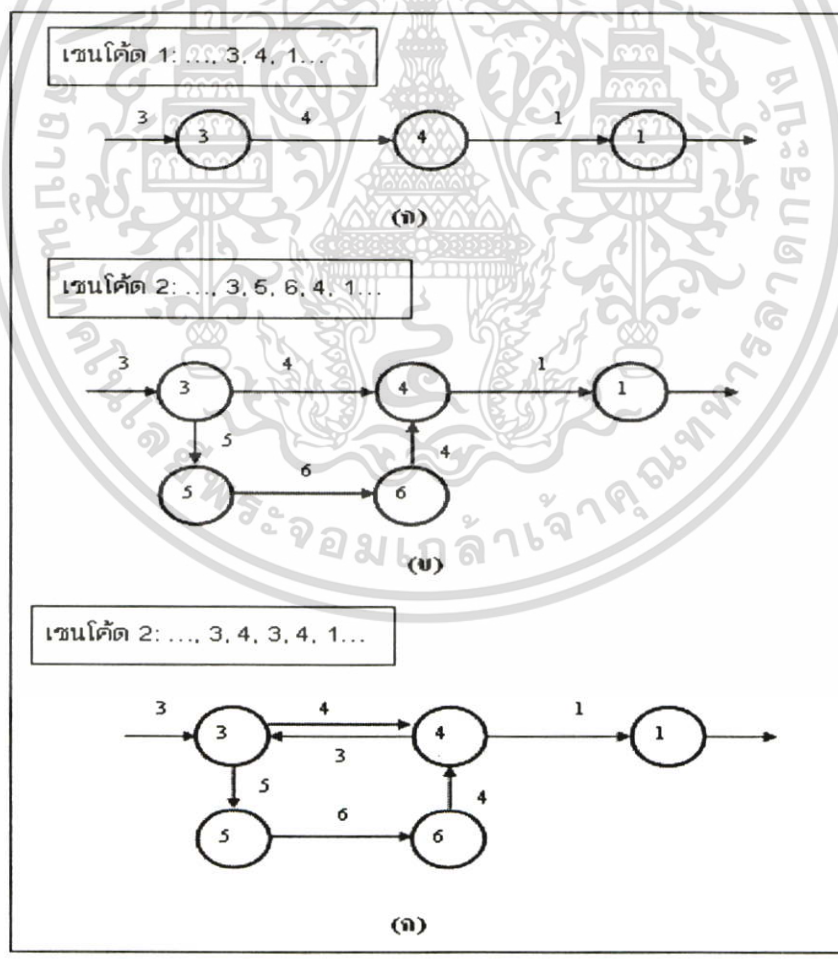
รูปที่ 6.9 แสดงการสอนด้วยสตริง 2 สายที่ทำให้เกิด Over Generalized แบบแนวราบ

จากรูปที่ 6.9 หลังจากทำการสอนด้วยสตริงทั้งสองสายแล้วจะเห็นว่าส่วนของโมเดล (ข) ซึ่งเป็นโมเดลที่เกิดจากการสอนด้วยเซนโค้ดสองสาย สามารถยอมรับส่วนของสตริงได้ 4 แบบดังนี้

1. ส่วนของเซนโค้ดที่ 1 : ..., 6, 1, 7, 4, 5,...
2. ส่วนของเซนโค้ดที่ 2 : ..., 6, 2, 7, 3, 5,...
3. ส่วนของเซนโค้ดที่เพิ่มขึ้นมา : ..., 6, 1, 7, 3, 5,...
4. ส่วนของเซนโค้ดที่เพิ่มขึ้นมา : ..., 6, 2, 7, 4, 5,...

ส่วนของเซนโค้ดที่เพิ่มขึ้นมานี้ถือว่าโมเดลเกิดการเพิ่มลักษณะทั่วไปที่มากเกินไป

3. กรณีการสอนโมเดลที่ทำให้เกิดการเพิ่มลักษณะทั่วไปที่มากเกินไป (Over generalized) ในลักษณะลูปดังรูปที่ 6.10 โดยเริ่มจากการสร้างโมเดลจากส่วนของสตริงสายแรกคือ ..., 3, 4, 1... ดังโมเดล (ก) จากนั้นทำการสอนด้วยส่วนของสตริงสายที่สองคือ คือ ..., 3, 5, 6, 4, 1... ดังโมเดล (ข) และสอนด้วยส่วนของสตริงสายที่สาม คือ ...,3, 4, 3, 4, 1... ดังโมเดล (ค) ซึ่งหลังจากทำการสอนด้วยสตริงทั้งสามสายตามหลักการที่นำเสนอในวิทยานิพนธ์นี้แล้ว ทำให้โมเดลที่ได้เกิดลูปขึ้น โดยลูปนี้จะทำให้โมเดลยอมรับส่วนของเซนโค้ด เช่น ..., 3, 4, 3, 5, 6, 4, 1... เป็นต้น ทำให้เกิดการวนอยู่ในลูปได้.



รูปที่ 6.10 แสดงการเกิด Over Generalized แบบลูป

6.7 การเปรียบเทียบโมเดลตามหลักการของ Stolcke และวิธีการที่นำเสนอ

จากการทดลองการเรียนแกรมม่าตามวิธีการของ Stolcke [7][8] โดยทำการทดลองกับเซนโค้ดชุดการสอน ที่เป็นชุดการสอนเดียวกันกับที่ใช้เรียนแกรมม่าตามหลักการเรียนรู้เพิ่มเติมโดยใช้คอนเท็กฟรีแกรมม่าที่ได้นำเสนอในวิทยานิพนธ์นี้ โดยในแต่ละชุดตัวอักษรมีตัวอย่างเซนโค้ดสำหรับการสอนทั้งหมด 100 ตัวอย่าง และมีความยาวของเซนโค้ดตัวอย่างละ 100 ตัว จากการทดลองในขั้นตอนการรวมกฎนั้น จะทำการรวมกฎโดยพยายามรวมนอนเทอร์มินอลทางด้านขวาของกฎเข้าด้วยกัน โดยจะพยายามรวมในทุกๆแบบที่สามารถทำได้ เริ่มจากการพยายามรวมนอนเทอร์มินอลตั้งแต่ 2 ตัวขึ้นและทำไปเรื่อยๆ โดยเมื่อทำการรวมเสร็จแต่ละครั้งจะต้องทำการทดสอบค่า likelihood ของการรวมทุกครั้ง รูปแบบการรวมใดที่มีค่า likelihood มากกว่าค่ามาตรฐานที่กำหนด (threshold) จะถือว่าเป็นรูปแบบที่ดีสำหรับการรวมในรอบนั้นๆ จากนั้นจะใช้รูปแบบการรวมที่มีค่า likelihood มากสุดเป็นแกรมม่าตั้งต้นสำหรับการรวมแกรมม่าในรอบถัดไป หลังจากทดลองรันโปรแกรมเพื่อสร้างแกรมม่าสำหรับตัวอักษรเพียง 1 ตัวนั้น เมื่อเวลาผ่านไป 20 ชั่วโมง แกรมม่าที่ได้ยังคงไม่มีแนวโน้มที่จะนำไปสู่แกรมม่าที่คาดว่าจะได้สำหรับโมเดลการรู้จำ เนื่องจากแกรมม่าที่ได้ยังคงมีความยาวทางด้านขวาของกฎใกล้เคียงกับแกรมม่าเริ่มต้น ทั้งนี้เพราะลำดับนอนเทอร์มินอลทางด้านขวาของกฎและจำนวนกฎในแกรมม่ามีจำนวนมาก อีกทั้งการทดสอบหาค่า likelihood ในแต่ละครั้งจะต้องทำการทดสอบโดยการพาร์สแกรมม่าที่ทำการเปลี่ยนแปลงนั้นกับเซนโค้ดที่เป็นตัวสอนทั้งหมดด้วย จึงใช้เวลานาน และเมื่อลองนำแกรมม่าที่มีค่า likelihood จากการทดลองนี้ไปทดสอบกับเซนโค้ดสำหรับทดสอบ แกรมม่าที่ได้ไม่สามารถรู้จำตัวทดสอบได้เลย

วิธีการของ Stolcke[7][8] จึงไม่เหมาะสำหรับการเรียนไวยากรณ์สำหรับการรู้จำดังนี้

- จำนวนกฎในแกรมม่าเริ่มต้นนั้นจะขึ้นกับจำนวนตัวอย่างที่ใช้สร้างแกรมม่าด้วย เช่น หากมีชุดสตริงตัวสอนทั้งหมด 100 ตัว แกรมม่าเริ่มต้นจะมีทั้งหมด 100 กฎ
- วิธีการสร้างแกรมม่าเริ่มต้นของงานวิจัยที่ได้นำเสนอจะทำให้ได้แกรมม่าที่มีลำดับทางขวาของกฎที่ยาว
- เนื่องจากด้านขวาของกฎยาวทำให้การใช้วิธีการ merging และ Chunking ใช้เวลานาน
- การปรับปรุงแกมม่าเริ่มต้นให้มีความเหมาะสม (Optimization) ไม่สามารถรับประกันได้ว่าจะได้แกรมม่าที่ง่ายและเป็นแกรมม่าที่เหมาะสมจริงๆ
- วิธีการสร้างแกรมม่าข้างต้นไม่เหมาะกับงานประยุกต์ทางด้านการรู้จำเนื่องจากตัวอย่างเซนโค้ดของตัวอักษรที่ใช้มีความยาวมาก

วิธีการสร้างวิธีนี้มีความจำกัดเรื่องการดึงลักษณะทั่วไป (Generalized) เมื่อทดสอบกับเซนโค้ดสำหรับทดสอบ แกรมม่าที่ได้ไม่สามารถรู้จำตัวทดสอบได้เลย

จากการทดลองพบข้อแตกต่างของวิธีการแบบ Stolcke [7][8] และวิธีการเรียนรู้เพิ่มเติมโดยใช้คอนเท็กฟรีแกรมม่าที่ได้นำเสนอในวิทยานิพนธ์นี้มีดังตารางที่ 6.7

ตารางที่ 6.7 แสดงการเปรียบเทียบข้อแตกต่างของวิธีการแบบ Stolcke [7][8] และวิธีการเรียนรู้เพิ่มเติมโดยใช้คอนเท็กฟรีแกรมม่า

ข้อแตกต่าง	วิธีการแบบ Stolcke [7][8]	วิธีการเรียนรู้เพิ่มเติมโดยใช้คอนเท็กฟรีแกรมม่า
จำนวนกฎในแกรมม่าเริ่มต้น	จำนวนกฎเท่ากับจำนวนตัวสอน จากการทดลองมี100 กฎ	จำนวนกฎน้อยกว่าโดยไม่เกินกว่าจำนวนเซนโค้ดในแต่ละสาย
จำนวนนอนเทอร์มินอลทางด้านขวาของกฎ	เท่ากับจำนวนเซนโค้ดในแต่ละสายที่เป็นตัวสอน	มีเพียง 2 ตัวคือเทอร์มินอลและนอนเทอร์มินอลสำหรับกระจายไปยังกฎถัดไปเท่านั้น
ความเร็วในการสอน	ช้ากว่าวิธีการเรียนรู้เพิ่มเติม	เร็วกว่า แบบ Stolcke
ระยะเวลาที่ใช้ในการสอน	ช้ากว่าวิธีการเรียนรู้เพิ่มเติม	เร็วกว่า แบบ Stolcke มาก (ไม่ถึง 1 นาที)
การรับประกันว่าจะได้โมเดลการรู้จำ	ไม่สามารถรับประกันได้	รับประกันได้เพราะเมื่อทดสอบกับเซนโค้ดตัวทดสอบให้ผลเป็นที่น่าพอใจ
การเพิ่มลักษณะทั่วไป (generalized)ให้กับโมเดล	มีข้อจำกัดมากกว่าเพราะวิธีการนี้ทำการสร้างนอนเทอร์มินอลเท่ากับจำนวนเซนโค้ด	เพิ่มลักษณะทั่วไปได้ดีกว่า แต่อาจมีบางกรณีที่เพิ่มลักษณะทั่วไปมากเกินไป (Over Generalized)
การแสดงผลโมเดล	แกรมม่าที่ได้ยาวและจำนวนกฎขึ้นกับจำนวนสตริงที่ใช้ทดสอบ	กฎในแกรมม่าสั้น จำนวนกฎที่ได้จากการทดลองไม่เกิน 60 กฎ

บทที่ 7

สรุปผลการทดลองและข้อเสนอแนะ

7.1 บทนำ

ก่อนหน้านี้ได้มีงานวิจัยเกี่ยวกับการเรียนแกรมม่าจากกลุ่มอยู่บ้าง แต่ก็ยังไม่เป็นที่นิยมและแพร่หลายมากนัก อีกทั้งยังไม่มีงานวิจัยใดที่นำการเรียนแกรมม่ามาประยุกต์ใช้กับงานด้านการรู้จำตัวอักษร วิทยานิพนธ์นี้นำเสนอทางเลือกอีกทางหนึ่งในการรู้จำลายมือเขียนภาษาไทยแบบออนไลน์โดยใช้คอนเท็กฟรีแกรมม่าที่สร้างจากกลุ่มตัวอย่าง โดยใช้ทิศทางของตัวอักษรนั้นๆ ตามหลักการของเชนโค้ดมาเป็นอินพุตเพื่อสร้างแกรมม่าสำหรับตัวอักษรนั้น จากนั้นทำการปรับปรุงแกรมม่าเพื่อให้มีลักษณะโดยรวม (Generalize) มากพอที่จะครอบคลุมตัวอักษรนั้นๆในรูปแบบอื่นๆที่ไม่ได้นำมาใช้สอนด้วย ในขณะเดียวกันแกรมม่านี้ต้องมีความเฉพาะสำหรับตัวอักษรนั้นๆเท่านั้น

7.2 สรุปผลการทดลอง

จากการทดลอง โดยนำเอาแนวทางงานวิจัยด้านการเรียนแกรมม่าก่อนหน้านี้มาประยุกต์ใช้กับงานด้านการรู้จำตัวอักษร พบว่าวิธีการดังกล่าวนั้นไม่เหมาะสมกับงานด้านการรู้จำตัวอักษร ทั้งนี้เพราะบางวิธีการมีข้อจำกัดมากเกินไป เช่นข้อจำกัดเรื่องกลุ่มตัวอย่างที่ใช้เป็นอินพุต หากอินพุตมีความยาวมากเกินไปจะทำให้ใช้เวลาในการสอนนั้นนานเกินไป และบางวิธีนั้นโมเดลที่ได้มีลักษณะโดยทั่วไปมากเกินไป ทำให้โมเดลนั้นยอมรับตัวอักษรอื่นที่ไม่ใช่ตัวอักษรของโมเดลนั้น

วิทยานิพนธ์นี้ได้นำเสนอวิธีการเรียนแกรมม่าแบบไดนามิกที่เหมาะสมกับงานด้านการรู้จำตัวอักษรมากขึ้น โดยมีเทคนิคพิเศษที่ใช้สร้างแกรมม่าซึ่งมีลักษณะโดยรวมเพียงพอสำหรับสตริงของตัวอักษรตัวนั้นที่ไม่ได้อยู่ในเซตของตัวสอน แต่ว่าไม่มากเกินไปจนยอมรับสตริงที่ไม่ใช่ของตัวอักษรนั้นๆไปด้วย เทคนิคที่ว่านี้จะทำการสร้าง "กับดัก" เพื่อไม่ให้แกรมม่าที่สร้างมีลักษณะโดยรวมมากเกินไป อีกทั้งยังลดข้อจำกัดในบางรูปแบบเช่นกรณีของลำดับสตริงที่ซ้ำกัน เทคนิคพิเศษที่กล่าวถึงนี้ได้แก่ การสร้างลูป คู่ลูป และกับดักลูป

- แม่แบบชนิดลูป สร้างขึ้นเพื่อรองรับการกระจายวนกลับมายังสตริงตัวเดิม เช่น ลักษณะลำดับสตริงแบบ $n*$ โดย n คือเชนโค้ด 1, 2,...,8

- แม่แบบซนิดคู่ลูป สร้างขึ้นเพื่อรองรับการกระจายลำดับของสตริงที่เป็นคู่ของเซนโค้ด 2 เซนโค้ดกระจายกลับไป-มาระหว่างเซนโค้ด 2 เซนนี้ เช่น $(m * n*)*$ โดย m, n คือเซนโค้ด 1, 2,...,8
- แม่แบบซนิดกับดักลูป สร้างขึ้นเพื่อรองรับการกระจายลำดับของสตริงที่เกิดจากคู่ลูป 2 ลูปที่ติดกัน เป็นการดักกรณีที่เมื่อผ่านการกระจายคู่ลูปคู่แรกและเข้าสู่การกระจายของลำดับสตริงที่เป็นคู่ลูปคู่ที่ 2 แล้วสตริงจะต้องไม่วนกลับหลังไปยังคู่ลูปแรกอีก ทั้งนี้เพราะหากคู่ลูปที่ 2 สามารถกระจายกลับไปยังคู่ลูปคู่แรกได้ จะทำให้ลำดับการกระจายของตัวอักษรนั้นผิดได้เนื่องจากมีการข้ามคู่ลูปเกิดขึ้น

"กับดัก" จะทำหน้าที่คงลำดับ (sequence) ก่อนหลังของการกระจายสตริงนั้นๆไว้ เพื่อไม่ให้แกรมม่าที่สร้างมีลักษณะทั่วไปจนเกินไปนั่นเอง

แต่จากการทดลองที่ผ่านมา แม้ว่าผลการทดลองการรู้จำลายมือเขียนที่ได้จะเป็นที่น่าพอใจ แต่ยังคงพบปัญหาในเรื่องลักษณะทั่วไปที่แกรมม่าควรมีอยู่ เนื่องจากแกรมม่ายังคงยอมรับสตริงของตัวอักษรตัวอื่นที่ไม่ใช่ของแกรมม่านั้นๆอยู่ เปอร์เซ็นต์ความถูกต้องของการรู้จำจึงต่ำกว่าที่คาดว่าน่าจะเป็น ซึ่งต้องทำการปรับปรุงต่อไป

ปัญหาเรื่องอัลกอริทึมในการค้นหาในขั้นตอนการปรับปรุงแกรมม่าเป็นปัญหาหลักอันหนึ่งที่ต้องกล่าวถึง ทั้งนี้เพราะการปรับรุงแกรมม่าจะมีประสิทธิภาพดีและใช้เวลาน้อยเพียงใดนั้นขึ้นอยู่กับอัลกอริทึมการค้นหาที่ใช้ด้วย อัลกอริทึมการค้นหาที่ใช้ในงานวิจัยนี้คือการค้นหาแบบ full-search แม้ว่าการค้นหาวิธีนี้จะให้ประสิทธิภาพดี แต่ก็ยังมีจุดอ่อนคือเนื่องจากเป็นวิธีแบบ full-search จึงต้องเก็บทางเลือกทุกทางที่เป็นไปได้ในการปรับปรุงแกรมม่าไว้ แล้วจึงนำทางเลือกนั้นมาพิจารณาทีหลังว่าทางเลือกใดเป็นทางเลือกที่ดีที่สุด จึงต้องใช้พื้นที่มากในการเก็บทางเลือกทั้งหมด

7.3 ข้อเสนอแนะ

วิทยานิพนธ์นี้เป็นการรู้จำลายมือเขียนตัวอักษรภาษาไทยแบบออนไลน์โดยทำการสร้างโมเดลทางภาษาที่เรียกว่าคอนเท็กฟรีแกรมม่า มาใช้เป็นโมเดลในการรู้จำ แต่ยังคงมีบางประเด็นที่ต้องปรับปรุง ซึ่งผู้วิจัยขอเสนอแนะดังนี้

ในงานวิจัยนี้ประเด็นหลักคือเรื่องของความเจนเนอร์รอลไรซ์ของโมเดล จะเห็นได้ว่าโมเดลยังมีลักษณะโดยรวมมากเกินไป อำนาจจำแนกของแกรมม่าน้อย ทำให้ยอมรับสตริงบางสตริงที่ไม่ใช่สตริงของแกรมม่าตัวอักษรนั้น เปอร์เซ็นต์การรู้จำตัวอักษรเป็นตัวเลขที่พอยอมรับได้แต่ยังไม่เป็นที่น่าพอใจนัก ดังนั้นหากทำการเพิ่มลักษณะบางอย่างและปรับปรุงการสร้างแกรมม่าในบาง

ขั้นตอนเพื่อให้แกรมม่ามีลักษณะทั่วไปของสตริงนั้นๆที่จำกัดกว่านี้ เช่น อาจมีการกำหนดค่าความน่าจะเป็นให้แต่ละกฎในแกรมม่า จะทำให้การรู้จำได้ผลลัพธ์ที่ดีขึ้น

ในงานวิจัยนี้ไม่ได้ทำการรู้จำตัวอักษรภาษาไทยได้ทั้งหมด ซึ่งตัวอักษรที่ใช้ทดลองในงานวิจัยนี้เป็นตัวอักษรที่เวลาเขียนพยัญชนะนั้นต้องต่อเนื่องไม่ยกปากกาในขณะเขียน นั่นคือมีเพียงสโตกเดียว ดังนั้นจึงไม่สามารถรู้จำตัวอักษรเช่น ฐ, ศ, ษ, ส ซึ่งต้องมีการยกปากกาในเวลาเขียนหรือตัวอักษรที่มีหลายสโตกได้ ดังนั้นการพัฒนางานวิจัยนี้ต่อไปในอนาคตจะต้องสามารถรู้จำตัวอักษรได้ทั้งหมด

งานวิจัยนี้ไม่สามารถแยกความแตกต่างของตัวอักษรที่มีความยาวของหางพยัญชนะที่แตกต่างกันได้ เช่น "บ" และ "ป", "ผ" และ"ฝ", "พ"และ"ฟ" เป็นต้น กรณีดังกล่าวนี้อาจแก้ปัญหาได้โดยต้องคิดอัตราส่วนความยาวของส่วนหางเทียบกับความยาวของส่วนที่ต่อจากหัวของพยัญชนะเพื่อให้สามารถบ่งชี้ความแตกต่างได้

ในขั้นตอนการปรับปรุงแกรมม่า หากปรับปรุงอัลกอริทึมในการค้นหา (Search Algorithm) ให้มีประสิทธิภาพดีขึ้น จะช่วยลดเวลาที่ใช้ได้

สุดท้ายคืองานวิจัยนี้สามารถพัฒนาต่อไปให้เป็นการแสดงรูปแบบโมเดลตัวอักษรที่ง่ายขึ้น เช่น ทำให้อยู่ในรูปแบบของ Regular expression

เอกสารอ้างอิง

[1] H. Yuen. 1996. "A chain coding approach for real-time recognition of on-line handwritten character". ICASSP'96. Atlanta. USA. pp.3426-3429.

[2] M. Okamoto, K. Yamamoto. 1999. "On-line Handwritten Character Recognition Method using Directional Features and Clockwise/Counterclockwise Direction Change Feature", Fifth International Conference on Document Analysis and Recognition, Bangalore, India, September 20-22. pp. 491-494.

[3] Xiaolin Li, Plamondon R., Parizeau M., "Model-base On-line Handwritten Digit Recognition" Pattern Recognition, 1998. Proceedings, Fourteenth International Conference on, vol. 2, Aug 16-20, 1998 ,pp. 1134-1136

[4] Joe, M. J. Lee, H. Joo, "A combined Method on the Handwritten Character Recognition", Document Analysis and Recognition, 1995, Proceedings of the Third International Conference on, vol. 1, Aug. 14-16, 1995, pp. 112-115.

[5] Andreas Kosmala, Gerhard Rigoll, Stephane Lavirotte, Loic Pottier, "On-line Handwritten Formula Recognition using Hidden Markov Model and Context Dependent Graph Grammars" Fifth International Conference on Document Analysis and Recognition, Bangalore, India, September 20-22, 1999, pp. 107-110.

[6] HOPCROFT, JOHN E., & JEFFREY D. ULLMAN. 1979. Introduction to Automata Theory, Languages, and Computation. Reading, Mass.: Addison-Wesley.

[7] STOLCKE, ANDREAS. 1993. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, Berkeley, CA. To appear in Computational Linguistics.

[8] Stolcke A. & S. Omohundro (1994), Inducing Probabilistic Grammars by Bayesian Model Merging. In Grammatical Inference and Applications, R. C. Carrasco & J. Oncina, editors, Springer, pp. 106-118

[9] JURAFSKY, DANIEL, CHUCK WOOTERS, GARY TAJCHMAN, JONATHAN SEGAL, ANDREAS STOLCKE, ERICFOSLER, & NELSON MORGAN. 1994a. The

Berkeley Restaurant Project. In Proceedings International Conference on Spoken Language Processing, Yokohama.

[10] D. Angluin. And C.H. Smith. Inductive inference: Theory and methods. Computing Surveys, 15(3):247-269, 1983.

[11] A. Oliveria and S. Edwards. Inference of state machines from samples of behavior. Technical report, Dept. of Electrical Engineering, U.C. Berkeley, 1995. Available from http://www.eecs.berkeley.edu~/sedwards

[12] R.L. Rivest and R. E. Schapire. Diversity based inference of finite automata. In IEEE Symposium on the Foundations of Computer Science, pages 78-87, 1987.

[13] R.L. Rivest and R. E. Schapire. Diversity based inference of finite automata. In ACM Symposium on the Theory of Computing, pages411-420, 1989.

[14] D. Ron, Y. Singer and N. Tishby. Learning probabilistic automata with variable memory length. In Proc. Of the workshop on Computational Learning Theory, 1995

[15] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden markov models of biological primary sequence information. Proc. Of the National Academy of Science, USA, 91:1059-1063, 1994

[16] A. Krogh, M. Brown, I.S. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. J. of Moleular Biology, 235:1501-1531, 1994.

[17] A. Krogh, I.S. Mian, and D. Haussler. A Hidden Markov model that finds genes in E. Coli DNA, Technical Report USCS-CRL-93-33, U.C. Santa Cruze, Dept. Comp.Sci.,1993. Available from ftp.cse.ucsc.edu, directory pub/dna.

[18] A. Stolke and S.M. Omohundro. Hidden markov model induction by Bayesian model induction. Technical Report TR-94-03, International Computer Science Institute, 1994. Available from http://www.icsi.berkeley.edu.

[19] A.V. Aho and J.D. Ullman. The Theory of Parsing, Translation and compiling. Vol. I. Prentice Hall, 1972.

[20] A. Paz. Introduction to probabilistic Automata. Academic Press, 1971.

[21] L.R. Rabiner. A toturial in Hidden markov models and selected applications in speech recognition. Proc. Of the IEEE, 77(2):257-286,1989.

[22] D. Angluin. Computational leraning theory: Survey and selected bibliography. In ACM Symposium on the Theory of Computing, pages 351-369,1992.

[23] M.J. Kearns and U.V. Vazirani. An Introduction to Computational Learning Theory. MIT press,1994.

[24] D. Angluin. On the complexity of minimum inference of regular stes. Information and control, 39(3):337-350, 1978.

[25] S. Porat and J.A. Feldman. Learning automata from ordered examples. Machine Learning, 7:109-138, 1991.

[26] B.R. Gaines. Behavior/structure transformations under uncertainty. Intnl. J. of Man-Machine Studies,8:337-365, 1976.

[27] B.A. Trakhtenbrot and Ya. M. Barzdin. Finite Automata: Behavior and Synthesis. North Holland, 1973.

[28] K.J. Lang. Ransom DFAs can be approximately learned from sparse uniform examples. In Proc. of the Workshop on Computational Learning Theory, 1992. Volume 5.

[29] Y. Freund and M. Kearns, D. Ron, R. Rubinfeld, R.E. Schapire, and L. Sellie. Efficient learning of typical finite automata from random walks. In ACM Symposium on the theory of computing, pages 315-324, 1993.

[30] A.W. Biermann and J.A. Feldman. On the synthesis of finite-state machines from samples of their behavior. IEEE Trans. On Computers, Pages 592-597, 1972.

[31] L. Miclet. Regular inference with a tail-clustering method. IEEE Trans. On Systems, Man, and Cybernetics, SMC-10(11):737-743, 1980.

[32] M.L. Minsky. Computation: Finite and infinite Machines. Prentice Hall, 1967.

[33] Y. Freund and D. Ron. Learning to model sequences generated by switching distributions. In Proc. of the Workshop on Computational Learning Theory, 1996.

[34] Y. Fujiwara, M. Asogawa, and A. Konagaya. Stochastic motif extraction using hidden markov model. In International Conf. On Intelligent Systems for Molecular Biology, 1994.

[35] Bianchi D., Learning Grammatical Rules From Examples Using A Credit Assignment Algorithm, Proceedings of The Fisrt Online Workshop on Soft Computing(WSC1), Nagoya, Aug. 1996,pag.113.

[36] Hirobumi Nishida, "Model-Based Shape Matching with Structural Feature Grouping" IEEETransactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 3, March 1995, pp. 315-320.

[37] A. Malaviya, L. Peters, "Fuzzy Feature Description of Handwriting Patterns", Pattern Recognition Journal, vol. 30, Number 10, 1997, pp. 1591-1604.

[38] Daijin Kim and Sung-Yung Bang, "A Handwritten Numeral Character Classification Using Tolerant Rough Set" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.22, No.9, September 2000, pp. 923-937.

[39] Andreas Kosmala, Gerhard Rigoll, Stephane Lavirotte, Loic Pottier "On-Line Handwritten Formula Recognition using Hidden Makov Model and Context Dependent Graph Grammars" Fifth International Conference on Document Analysis and Recognition, Bangalore, India, 20 - 22 September, 1999, pp. 107-110.

[40] Bellegarda, E.J.; Bellegarda, J.R.; Kim, J.H.; "On-line handwritten character recognition using parallel neural networks", Acoustics, Speech, and Signal Processing, IEEE International Conference on , vol. 2, 19-22 April 1994, pp. 605 -608.

[41] James Power 2002-11-29. Converting an FSA to a Regular Grammar. [Online]. http://www.cs.may.ie/~jpower/Courses/parsing/node17.html

[42] T. Suebsanit and C. Kimpan, "Printed Thai Character Recognition using Multifeature and Multilevel Classification", SCCORED 2001, paper 167, Feb. 20-21, 2001.

[43] B. Kruatrachue, K. Siriboon, W. Chatwiriya and K. Bounnady "Online Handwritten Feature with Propotional Invariant", IASTED 2003, pp. 191-193, July 14-16, 2003.

[44] X. Gao, L.W. Jin, J.X. Jin, J.C. H, "A New Stroke-Base Directional Feature Extraction Approach for Handwritten Chinese Character Recognition" IEEE, 2001.

[45] Daijin Kim and Sung-Yang Bang, "A Handwritten Numeral Character Classification Using Tolerant Rough Set", IEEE Transaction on Pattern Analysis and Machine Intelligence Vol. 20, 2008.

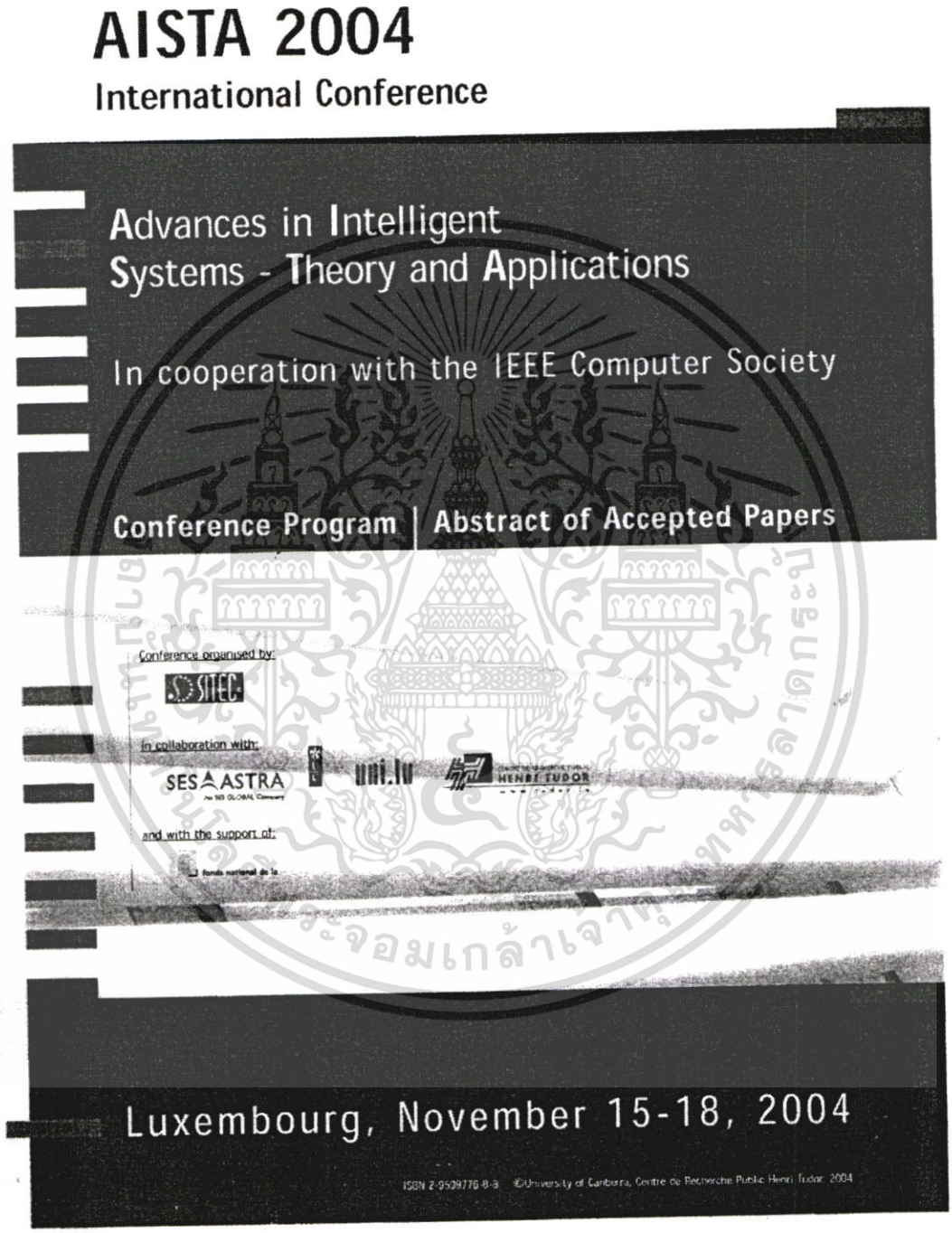
[46] T. Suebsanit, P. Phokharatkul, P. Pantaragphong, O. Pinngern and C. Kimpan, "Recogniton of Printed Thai Characters using Fuzzy and Rough Sets Theory", Proceedings of the International Conference on Robotics, Vision, Information and Signal Processing ROVISP 2003, Malaysia, Jan. 22-24, 2003, pp.33-38.

[47] E. Anquetil and G. Lorette, "Online Cursive Handwritten Character Recognition Using Hidden Markov Model", Traitement dn, Signal, vol.12, no.6, pp.575-583,1995.

ภาคผนวก

งานวิจัยที่ได้รับการตีพิมพ์

[1] Boontee Kruatrachue, Pongkasem Polsuntikul and Kritawan Siriboon, "Dynamic Construction of Context Free Grammar from Sample for On-line Thai Handwriting Recognition", AISTA 2004: International Conference, Luxembourg, Nov. 15 -18, 2004.



AISTA 2004 in Cooperation with the IEEE Computer Society Proceedings / ISBN : 2-9599776-8-8
15-18 November, Kirchberg - Luxembourg

AISTA 2004, ID: 131-04

Dynamic Construction of Context Free Grammar from Sample for On-line Thai Handwriting Recognition

Boontee KRUATRACHUE, Pongkasem POLSUNTIKUL and Kritawan SIRIBOON

Abstract— **This paper proposes a new approach for on-line handwriting recognition. In this study, string of direction information (chain code) of alphabets was used for generating recognition models. Each character model is represented by context free grammar (CFG). The main problem addressed in this paper is how to automatically generate CFG of chain code sequences for each training characters. This CFG has to be generalized enough to classify other similar characters but not too generalized to except other characters. CFG is automatic induction from a selected string of chain code for a character and incrementally refined to be more generalized to accept difference patterns of that character. The induction CFG is shown for a sample training set strings. Also the recognition results of the CFG models are compared to the HMM models.**

Index Terms— Grammar induction, Context free grammar, Handwriting recognition, Dynamic language model.

I. INTRODUCTION

IN human handwriting, people write the same alphabet differently but we still know that it is the same characters.

In this research, we represent handwriting as a string of direction alphabet (1-8 chain code like direction). We implement each character model as context free grammar (CFG). So our main problem is to automatically construct the CFG for each character that fully represent the set of training string of that character. The restriction of this grammar is to be generalized enough to accept the untrained string of the same character but at the same time not to generalize to accept others.

There are many papers on grammar induction or grammatical inference such as Stolcke[2][3] and Stanley F. Chen[4], which are not suitable in this character recognition.

Since the induction grammar is either too restricted or too generalize for the recognition purpose. The grammar in this paper is constructed based on the specific policy so that the grammar will be generalized enough to the unseen strings but not over generalized to accept other character strings. The grammar guided policy create trap for other characters strings for over generalized purpose. The guided policy also create less restricted pattern like 4* to represent sequence of 4, 44, 44....44 for generalized purpose.

Our approach has suggested the dynamic construction of CFG from samples. There are distinguished points of our method. First, construction of initial model is very simple. Second, it takes less time for searching in merging method. Third, it can be applied for learning long-sequence samples better than Stolcke [3].

A. Chain Code

According to our experimental studies, we used the direction information obtained from the pixel maps of handwriting, called chain code.

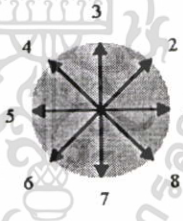


Fig.1. Definition of chain code in the 8-connected grid.

As chain code, an arbitrary curve is represented by sequence of small vectors of length and a limited set of possible directions. Fig.1 showed the sample of chain code in the 8-connected grid which is used in this experiment. A chain is defined as an ordered sequence of links with possible interspersed signal codas. Fig. 2 showed the direction followed by chain code in the 8-connected grid of "ก".

B. The Context Free Grammar (CFG)

In this paper, the terms "Grammar" or "CFG" all mean "Context Free Grammar". There are rules in grammar. A CGF G can be represented by its 4 components, that is, G=(V, T ,P ,S) where V is the set of variables , T the terminals , P the set of productions and S the start symbol.

Manuscript received September 30, 2004.

Boontee KRUATRACHUE is Assoc. Prof. Dr in Department of Computer Engineenng, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang Bangkok, 10520 Thailand (e-mail: booontee@yahoo.com)

Pongkasem POLSUNTIKUL is a master student in graduated school. She enroll as a full time student in Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang Bangkok, 10520 Thailand. (e-mail: s5061031@kmitl.ac.th).

Kritawan SIRIBOON is Asst. Prof. in Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang Bangkok, 10520 Thailand. (e-mail: Kritawan@diamond.ce.kmitl.ac.th)

The language of G, denoted L(G) , is the set of terminal strings that have deviations from the start symbol. We called context-free language, or CFL. [1].

4 4 3 3 2 2 3 3 3 2 2 3 3 2 2 2 3 2 3 2 2 2 3 3 3 3 4 5 5 5 5
6 5 2 2 2 1 2 2 2 1 2 2 2 1 2 2 1 2 1 2 2 1 1 1 1 1 1 1 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 7 7 7 8 7 8 7 7 8 7 7 8 7 7 8 7 7 8 7 7
7 7 8 7 7 7 7 7 8 7 7 7 2

Fig.2. Sequence of chain code of alphabet "ก"

From figure3, showed partial sample of "ก", where "Si" is nonterminal, "→" is production symbol, numbers are terminal and "|" denote choices of derivation. Nonterminal in left hand side of → is called "left hand side :lhs". Terminals and nonterminals in right hand side of → is called "right hand side :rsh"

S0 → 3 S1
S1 → 3 S1 | 1 S2
S2 → 1 S2 | 2 S3
S3 → 2 S3 | 5 S4
S4 → 5 S4 | 6 S5
.....

Fig.3. partial CFG sample of "ก"

II. RELATED WORK

There are many previous researches that adapted language model in speech recognition [7], for example Stolcke[2][3] and Stanley(1995) [4][5] . Stolcke[2] had proposed probabilistic modeling of language which we have got the idea for applying CFG for pattern recognition. The methodology of learning Inducing Probabilistic Grammars[3] showed the construction of grammar from samples and merging method to normalize the initial grammar. This method starts with start symbol in lsh. For rsh, every single sample had been converted to be choices of the derivation. To make it more generalize, this approach try to merge the model. Merging algorithm played an important role in this experiment. It is simple when we adopted this proposed method on our on-line handwriting recognition. However, it took a long time for merging and chunking because the training chain codes is much longer than sample in the proposed research.

Our approach has suggested the dynamic construction of CFG from samples. There are distinguished points of our method. First, construction of initial model is very simple. Second, it takes less time for searching in merging method. Third, it can be applied for learning long-sequence samples more than Stolcke [3].

III. S YSTEM OVERVIEW

The proposed system has 3 main steps as shown in fig. 4. We used the direction information, called chain code, to be the training set. From sequences of chain code for each alphabet, our approach is to construct the context free grammar (CFG). We first construct the initial grammar from first sample. This grammar has a specific pattern of sequence just only for this first character. Then, we try to make the grammar more generalized by refining this initial grammar, using the rest of samples. Last, we test this grammar by parsing unknown sequence of string.

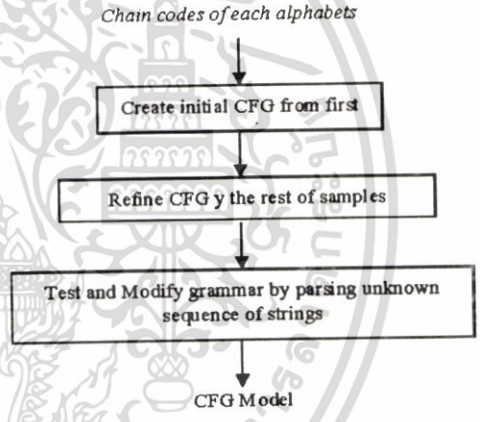


Fig.4. System Overview

IV. CONTEXT FREE GRAMMAR CONSTRUCTION

We construct the initial grammar follow sequence of first sample. For each single chain code, there is a combination of this chain, acted as the terminal, and its nontermianl. Table 1 showed the construction of initial grammar. "Si" is nonterminal. The symbol "|" in rule denote the choices of rule's derivation.

To see more clearly, we would like to explain by using sample of construction initial CFG in table1 and using Regular Expression (RE) to represent partial sequence of chain code.

The grammar start with starting nonterminal "S0" in the left hand side that derived to the first chain code with its noterminal. For example in table1, chain code "4 4 3 3 3 ...",

the first rule is "S0→4 S1" while "S1" is nontermial of chain "4".

For last chain, we ended with nonterminal "F." For example, chain ". . . 4 2 5", the rule for last chain "5" is "5 Si →5 Si | F."

TABLE 1
Initial construction CFG

4 4 3 3 3 4 5 6 5 6 3 2 3	Sample sequence
4	S0 →4 S1
4 4	S0 →4 S1 S1 →4 S1 ← Loop
4 4 3	S0 →4 S1 S1 →4 S1 | 3 S2
4 4 3 3 3	S0 →4 S1 S1 →4 S1 | 3 S2 S2 →3 S2
4 4 3 3 3 4	S0 →4 S1 S1 →4 S1 | 3 S2 S2 →3 S2 | 4 S1 ← Loop Pair
4 4 3 3 3 4 5	S0 →4 S1 S1 →4 S1 | 3 S2 | 5S3 S2 →3 S2 | 4 S1 S3 →5S3
4 4 3 3 3 4 5 6	S0 →4 S1 S1 →4 S1 | 3 S2 | 5S3 S2 →3 S2 | 4 S1 S3 →5 S3 | 6 S4 S4 →6 S4
4 4 3 3 3 4 5 6 5 4 4 3 3 3 4 5 6 5 6	S0 →4 S1 S1 →4 S1 | 3 S2 | 5S3 S2 →3 S2 | 4 S1 S3 →5 S3 | 6 S4 S4 →6 S4 | 5 S3
4 4 3 3 3 4 5 6 5 6 3 4 4 3 3 3 4 5 6 5 6 3 2 4 4 3 3 3 4 5 6 5 6 3 2 3	S0 →4 S1 S1 →4 S1 | 3 S2 | 5S3 S2 →3 S2 | 4 S1 S3 →5 S3 | 6 S4 S4 →6 S4 | 5 S3 | 3S5 ← Trap Loop Pairs S5 →3 S5 | 2 S6 S6 →2 S6 | 3 S5

Our guided policies to construct initial CFG, is the policy of "generalized" and "trap". For Generalized of CFG, there are 2 policies to construct CFG, called "Loop" and "Loop Pair". The "trap" policy is used to trap chain code sequence for not being too generalized, called "trap loop pair".

1) Loop Policy:
Partial sequence : 4, 44, 444, 44444...
RE : 4^+
CFG "loop" : S1 →4 S1

This partial grammar accepts chain code sequence of zero or more '4' (which will be called loop 4).

2) Loop Pair Policy:
Partial sequence : 434, 4 43334, 44343, 4443333434...
RE : $(4^+3^+)^*$
CFG "loop" : S1 →4 S1 | 3 S2
S2 →3 S2 | 4 S1

This partial grammar accepts chain code sequence of "loop4" and "loop3" back and forth (loop43).

3) Trap loop Pair:
Partial sequence : 5656323, 565323, 5565333223...
RE : $(5^+6^+)^*$ $(3^+2^+)^*$
CFG "loop" : S3 →5 S3 | 6 S4 | 3S5
S4 →6 S4 | 5 S3 | 3S5
S5 →3 S5 | 2 S6
S6 →2 S6 | 3 S5

As seen in rule S3 and S4, they are loop56 with the addition of 'jumping' to loop32. The trap created here is the jumping to S5. Once S5 in rule S3, S4 is expanded, there is no going back to loop56. This partial grammar is called "trap loop pair 56, 32

The following table is a step by step initial grammar rule induction from the first sample of 4 4 3 3 3 4 5 6 5 6 3 2 3. Rule S0 is the starting rule. Rule S1 is the CFG of the loop 4. The construction of grammar follows the sequence and constructs loop4 loop3, once we see 4 then they become loop pairs 43. In the same way, there is the following sequence "... 5 6 5 6 3 2 3", we construct loop pair 56 and loop pair 32. In order to trap specific chain code sequence only $(5^+6^+)^*$ $(3^+2^+)^*$, not for other sequences, we construct trap loop pair 5632.

B. Refine Context Free Grammar

Making initial grammar to accept other training sequences, we refine this grammar by using the rest of the samples. For each new training sample string, we parse the string with the current CFG until it reaches the unaccepted alphabet in the string. Since this string must be accepted by the CFG, we need to modify CFG by inserting rule to make this string acceptable. We try to minimize the insertion of the rules to make our CFG simplest by reusing the existing rule. But we may have to search through all the rules to find the appropriate one.

1. S0 →4 S1
2. S1 →4 S1 | 3 S2 | 5 S3
3. S2 →3 S2 | 4 S1
4. S3 →5 S3 | 6 S4
5. S4 →6 S4 | 5 S3 | 3 S5
6. S5 →3 S5 | 2 S6 | 4 S7
7. S6 →2 S6 | 3 S5
8. S7 →4 S7 | 5 S8
9. S8 →5 S8 | 2 S9
10. S9 →2 S9 | 6 S10
11. S10 →6 S10 | F

Fig. 5. Sample of initial grammar

For example, as initial grammar in fig. 5, we want to refine it to accept sequence sample, assumed as:

4 3 4 2 3 2 4 5 2 6

The method starts refining from parsing first rule "S0". From grammar we can still parse this sequence "4 3 4" until current chain code is "2". There is no nonterminal to derive "S1" for chain"2". We have to refine CFG by adding some choice to rule1 for deriving "S1" followed "2". In grammar,

there also will be more than one nonterninals that can be a choice to add in current rule. Fig. 6 show possible choices for deriving "S1" to "2".

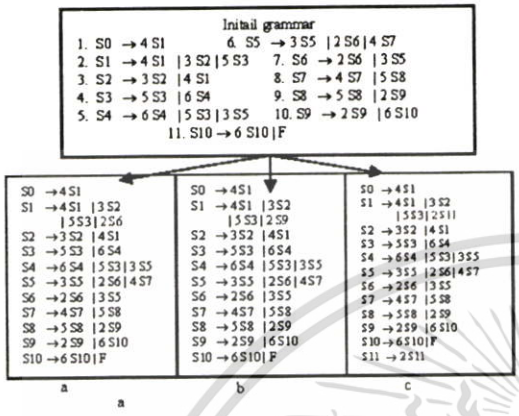


Fig.6. Three different choices to derive "S1" followed chain code "2"

There are 3 patterns to refine, fig.3a added "2 S6", fig.3b added "2 S9", fig.3c added new nonterminal "S11" with its rule. Therefore, the optimize choice is figure 3a because from "S6" in rule7 can continue parsing though the end of the sequence. The refining continues until the end of sequence. Consequently, this grammar is generalized and simplest for this sequence. To search for rules in grammar we look forward through existed rule by using heuristic search.

The algorithm starts from starting rule "S0", and searches for candidate rules that can derive to current chain in a sample. Then, we add these candidate rules with current chain to be a new choice in current rule. While refining grammar, cost will be assigned for grammars. Cost for adding choice, that contains nonterminal already existed in grammar, is cheaper than cost for adding new nonterminal that has never existed in grammar before, as algorithm below.

```
For each chain in a sample
  For each grammar
    - Find candidate rules:
        → all nonterminal with terminal, as same as current
          chain
        → new nonterminal with terminal, as same as current
          chain
    - add all candidates to be new choices in the right hand side
      of current rule
    - construct new grammars by replace current rule with these
      new rules
  End For
End For
```

Refining grammar continue until all chain codes in sequence have been processed. The most optimize grammar is the candidate grammar with the smallest cost. We use this optimize one to be the initial grammar for the next recursive of refining process. Finally, we get the optimize grammar for all these training set.

V. Results and Discussion

The previous works of pattern recognition have been focused on the training algorithm [5]. Most of them used artificial Intelligent (A.I.) model such as, Genetic Algorithm (G.A.) and neural network [6] for training. The major difference of our works is the dynamic construction of model for recognition and represented in form of context free grammar (CFG). This model was trained by heuristic method in searching appropriate rules to refine CFG. The searching algorithm played very important row to find the appropriate grammar. Searching through grammar consumes most of running time. Heuristic search, used in this research, give a good efficiency more than the simple search e.g. Look up n-rules. However, for some alphabets with no varieties of chain codes, the simple search could be used and generated grammar that quite similar to the heuristic search.

Our algorithm adapted a context free grammar to construct dynamic model for On-line Thai Handwriting Recognition. There can be more than one grammar that can derive the same alphabet. The most appropriate one is the grammar with the fewer rules. Thus, we assigned some cost during refining CFG to be a parameter for adjusting the appropriate one.

We generated context free grammar for each character and trained the grammar by their chain codes. The trained characters set consist of 38 letters with the total 3839 characters. The test characters are 5333 characters. The recognition performed by using "parsing algorithm" to derive and construct parse tree. The Number of correct characters recognition is 80.75%.

VI. Conclusion

Model for on-line handwriting can be automatically generated and represented by Context Free grammar. Our experiment shows that model is simple and proportion invariant with high recognition rate.

There are some more future works to improve from our experiment:

1. Since our model is based on language model, the model can be converted in to regular form, which is more simple and easier to represent.
2. In order to improve the recognition rate, it is dispensable to use some off-line techniques with ambiguous chain codes.
3. The probabilistic parser [8] can be used for improving the efficiency of recognition.

ACKNOWLEDGMENT

The authors would like to thank the chair man and committees of AISTA 2004 for organizing this conference.

REFERENCES

[1] Hopcroft, John E., & Jeffrey D. Ullman. 1979. Introduction to Automata Theory, Languages, and Computation. Reading, Mass.: Addison-Wesley.

[2] Stolcke, Andreas. 1993. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, Berkeley, CA. To appear in Computational Linguistics.

[3] Stolcke A. & S. Omohundro (1994), Inducing Probabilistic Grammars by Bayesian Model Merging. In Grammatical Inference and Applications, R. C. Carrasco & J. Oncina, editors, Springer, pp. 106-118.

[4] Stanley F. Chen (1995), Bayesian Grammar Induction for language Modleing, Proc. 33rd Annual Meeting of the ACL, p. 228-235, Cambridge, MA 1995

[5] Kam-Fai Chan, Dit-Yan Yeung(1998), Recognizing on-line handwriting alphanumeric characters through flexible structural matching, The Journal of the pattern recognition society, October27 ,1998.

[6] M. Fatih Amasyah, Nilgun Erdem, Hakan Haberdar, Filiz Koyuncu (2003). *Neuro-Chain : A Handwritten Character Recognition System*, Turkish Symposium on Artificial Intelligence and Neural Networks – TAINN 2003.

[7] Zue V., Glass J., Goodnie D., Hong L., Phillips M., Polifroni J., & Seneff S. 1991. *Integration of speech recognition and natural language processing in the MIT Voyager system*. In Proceedings IEEE Conference on Acoustics, Speech and Signal Processing, volume 1, 713–716, Toronto.

[8] Jones, Mark A., & Eisner J. M., 1992a. *A probabilistic parser and its applications*. In AAAI Workshop on Statistically-Based NLP Techniques, 20–27, San Jose, CA.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นางสาว ปองเกษม พลสันติกุล
วันเดือนปีเกิด	วันศุกร์ ที่ 26 เดือน กันยายน พ.ศ. 2523
ประวัติการศึกษา	สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรวิทยาศาสตร์บัณฑิต สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2544