การสร้างดีเอฟเอเพื่อการรู้จำตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์

DFA INDUCTION FOR ON-LINE THAI HANDWRITTEN RECOGNITION

สมาสันต์ สุวรรณนิตย์
SAMASANT SUWANNIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2549

ISBN 974-15-2657-1

การสร้างดีเอฟเอเพื่อการรู้จำตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์

DFA INDUCTION FOR ON-LINE THAI HANDWRITTEN RECOGNITION

สมาสันต์ สุวรรณนิตย์
SAMASANT SUWANNIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2549

ISBN 974-15-2657-1

DFA INDUCTION FOR ON-LINE THAI HANDWRITTEN RECOGNITION

SAMASANT SUWANNIT

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

2006

ISBN 974-15-2657-1

หัวข้อวิทยานิพนธ์	การสร้างดีเอฟเอเพื่อการรู้จำตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์
นักศึกษา	นาย สมาสันต์ สุวรรณนิตย์
รหัสนักศึกษา	44061616
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2549
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร. บุญธีร์ เครือตราชู

บทคัดย่อ

วิทยานิพนธ์นี้ได้นำเสนอการใช้ดีเทอร์มินิสติกไฟไนท์ออโตมาต้า (DFA) เพื่อสร้างเป็นโมเดลการรู้จำของตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์ โดยมีจุดมุ่งหมายให้มีการสร้าง DFA แบบอัตโนมัติจากข้อมูลชุดฝึกสอนเพื่อให้เกิดการรู้จำลักษณะของตัวอักษรแต่ละตัว ซึ่งโมเดล DFA ของตัวอักษรแต่ละตัวจะต้องมีความ generalize เพียงพอที่จะจดจำลักษณะของตัวอักษรที่นำมาทดสอบซึ่งไม่ใช่ตัวที่เคยนำมาสอนก่อนหน้า แต่จะต้องไม่ over-generalize จนทำให้ยอมรับการทดสอบการรู้จำของอักษรตัวอื่นๆที่ไม่ตรงกับโมเดลของตัวอักษรของตัวมัน โดยเราจะทำการสร้างโมเดลการรู้จำของ DFA จากข้อมูลชุดฝึกสอนเดียวกัน แล้วสร้างเป็น DFA 2 แบบ แบบแรกใช้ข้อมูลอินพุทในรูปแบบของเชนโค้ด 8 ทิศ และมีการใช้กระบวนการทำงานของ loop pair policy ซึ่งเป็นวิธีการที่นำมาจากงานวิจัยก่อนหน้า ส่วนอีกแบบ จะสร้าง DFA จากเชนโค้ด 16 ทิศพร้อมค่าขนาดและตำแหน่ง และไม่ใช้กระบวนการทำงานแบบ loop pair policy ซึ่งเป็นวิธีการที่คิดขึ้นมาใหม่สำหรับงานวิจัยนี้ จากนั้นทำการทดสอบการรู้จำของ DFA ทั้ง 2 แบบ ผลการรู้จำของ DFA 8 ทิศ self accept = 87.97%, accept false = 24.92% ส่วนผลของ DFA 16 ทิศ self accept = 76.52%, accept false = 9.11% ซึ่งแสดงให้เห็นว่า DFA 8 ทิศ มีความ generalized มากเกินไป ส่วน DFA 16 ทิศ ก็มีความ specific มากเกินไป ดังนั้นจึงได้เพิ่มความยืดหยุ่นในการทดสอบให้กับ DFA 16 ทิศ ด้วยการเพิ่มค่าทิศทาง +1,-1,+2,-2 ซึ่งทำให้ผลของ recognition rate ของ DFA 16 ทิศ ออกมาเท่ากับ 84.40%

Thesis Title	DFA induction for on-line Thai handwritten recognition
Student	Mr. Samasant Suwannit
Student ID.	44061616
Degree	Master of engineering
Programme	Computer engineering
Year	2006
Thesis Advisor	Assoc.Prof. Boontee Kruatrachue

Abstract

This thesis proposes method to automatically generate the Deterministic Finite Automata (DFA) to learn on-line Thai handwritten recognition from training set of each character. So the DFAs must be generalize enough to accept same character but not to over-generalize to accept other written-alike characters. In this thesis we made DFA 2 type from only one training set, first type made DFA from chain code 8 directions and use loop pair policy, second type made DFA from chain code 16 directions with value of length and position but not use loop pair policy. The results of DFA 8 directions is self accept = 87.79%, accept false = 24.92% and DFA 16 directions is self accept = 76.52%, accept false = 9.11%. Those results shown the DFA 8 directions is more generalize and the DFA 16 directions is more specific. So we use flexible value +1,-1,+2,-2 to DFA 16 directions for increase recognition rate = 84.40%

กิตติกรรมประกาศ

คุณความดีอันใดที่บังเกิดจากวิทยานิพนธ์ฉบับนี้ ขอมอบแด่บิดาและมารดาของผู้วิจัย ผู้ซึ่งให้โอกาสและสนับสนุนค่าใช้จ่ายในการศึกษาเล่าเรียนและการทำวิจัยตลอดมา ทำให้ผู้วิจัยมีโอกาสได้แสวงหาความรู้พัฒนาตนเองและทำการศึกษาวิจัยจนทำให้เกิดวิทยานิพนธ์ฉบับนี้ขึ้นมาได้ ผู้วิจัยขอสำนึกถึงพระคุณนี้อันเป็นที่สุด

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

ขอขอบพระคุณกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำในทุกๆเรื่อง ทั้งวิธีการแก้ปัญหาที่เกิดขึ้นในวิทยานิพนธ์และมุมมองในเชิงวิศวกรรมอื่นๆที่เกี่ยวข้องกับงานวิจัยนี้ ซึ่งช่วยให้ผู้วิจัยมีวิสัยทัศน์ที่กว้างไกลขึ้น

ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้โอกาสผู้วิจัยได้เข้ารับการศึกษาในระดับบัณฑิตศึกษา สาขาวิศวกรรมคอมพิวเตอร์ ทำให้ผู้วิจัยได้ศึกษาหาความรู้และทำการวิจัยจนเกิดเป็นวิทยานิพนธ์ฉบับนี้ขึ้นมา

ขอขอบพระคุณครูบาอาจารย์ทุกท่านตั้งแต่เล็กจนเติบใหญ่ที่ประสิทธิ์ประสาทวิชาความรู้ให้แก่ผู้วิจัย ซึ่งเป็นพื้นฐานความรู้ต่างๆให้แก่ผู้วิจัย ทำให้ผู้วิจัยมีความรู้ความสามารถที่จะทำการศึกษาวิจัย และทำการศึกษาวิจัยได้จนสำเร็จ ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูง

ผู้วิจัยขอขอบคุณสำหรับการเสียสละเวลาอันมีค่าของเพื่อนๆทุกคน ที่ช่วยเหลือผู้วิจัยในการเขียนตัวอักษรภาษาไทยขึ้นมาเพื่อเป็นชุดข้อมูลที่ใช้ในการสอนและทดสอบสำหรับงานวิจัยนี้ โดยการเขียนชุดข้อมูลขึ้นมาใหม่นี้เพื่อนๆแต่ละคนต้องเสียสละเวลาประมาณ 1 ชั่วโมงต่อบุคคล และต้องใช้ความอดทนในการเขียนซ้ำหลายๆรอบและหลายๆแบบ ผู้วิจัยซาบซึ้งในน้ำใจไมตรีและการให้ความช่วยเหลือจากเพื่อนๆทุกคนในครั้งนี้เป็นอย่างสูง

ผู้วิจัยขอขอบคุณสำหรับเพื่อนๆและเพื่อนรุ่นน้องๆบางคน ที่ช่วยเหลือผู้วิจัยในการแปลบทความงานวิจัยจากภาษาไทยให้เป็นภาษาอังกฤษ ซึ่งต้องนำไปใช้เป็นบทความงานวิจัยที่ส่งไปตีพิมพ์ในการประชุมวิชาการในระดับนานาชาติ ผู้วิจัยขอขอบคุณและซาบซึ้งในการช่วยเหลือครั้งนี้อย่างเป็นสูง

สุดท้ายนี้ หากวิทยานิพนธ์ฉบับนี้มีข้อผิดพลาดประการใดผู้วิจัยขอน้อมรับไว้เพียงผู้เดียว

สมาสันต์ สุวรรณนิตย์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การรู้จำลายมือเขียนเป็นอีกกระบวนการหนึ่งที่จะทำให้มนุษย์สามารถติดต่อกับคอมพิวเตอร์ได้สะดวกมากขึ้น ไม่จำเป็นต้องใช้ทักษะอย่างอื่นที่ต้องฝึกฝนมามากนัก เช่น ความสามารถในการพิมพ์ตัวอักษรจากแป้นพิมพ์ ซึ่งจะต้องมีการฝึกฝนจนเกิดความชำนาญจึงจะสามารถใช้แป้นพิมพ์ได้อย่างมีประสิทธิภาพ เพราะฉะนั้นหากการรู้จำลายมือเขียนมีประสิทธิภาพมากพอ เราก็สามารถที่จะเขียนคำสั่งลงบนอุปกรณ์อินพุทของเครื่องคอมพิวเตอร์ แล้วคอมพิวเตอร์ก็จะสามารถทำงานตอบสนองต่อคำสั่งนั้นๆได้ ซึ่งจะทำให้เราสามารถทำงานติดต่อกับคอมพิวเตอร์ได้สะดวกสบายมากยิ่งขึ้น

การรู้จำลายมือเขียนยังแบ่งออกเป็น 2 ประเภท ตามลักษณะที่มาของข้อมูลที่ใช้ในการรู้จำ ได้แก่ การรู้จำแบบออฟไลน์ (Offline Recognition) ซึ่งจะใช้ข้อมูลที่เป็นภาพที่ได้มาจากการสแกนเพื่อใช้ในการรู้จำ ส่วนการรู้จำลายมือเขียนอีกแบบหนึ่ง คือ การรู้จำแบบออนไลน์ (Online Recognition) ซึ่งจะใช้ลำดับของจุดข้อมูลที่ได้มาจากปากกาอิเล็กทรอนิกส์ (tablet) ที่ใช้ในการเขียนตัวอักษรแล้วนำมาใช้เป็นข้อมูลในกระบวนการรู้จำ ซึ่งในวิทยานิพนธ์นี้ได้มุ่งประเด็นของการวิจัยไปที่การรู้จำตัวอักษรที่เป็นลายมือเขียนภาษาไทยแบบออนไลน์และต้องการให้การรู้จำทำงานแบบอัตโนมัติ โดยขั้นแรกจะทำการสอนในเกิดการรู้จำลักษณะของตัวอักษรแต่ละตัว จากนั้นจะทำการทดสอบการรู้จำของตัวอักษรแต่ละตัว

งานวิจัยด้านการรู้จำลายมือเขียนแบบออนไลน์เท่าที่ผ่านมามีการทำวิจัยและตีพิมพ์นำเสนอออกมาอยู่มากพอสมควร ซึ่งได้ใช้วิธีการที่แตกต่างกันออกไป ตัวอักษรของภาษาแต่ละภาษาที่ถูกใช้ในกระบวนการสร้างการรู้จำก็แตกต่างกันออกไป ซึ่งจะยกตัวอย่างงานวิจัยบางชิ้นไว้ ณ ที่นี้ เช่น งานวิจัยของ ปองเกษม พลสันติกุล [1] ทำการรู้จำลายมือเขียนภาษาไทยแบบออนไลน์โดยใช้คอนแท็กฟรีแกรมม่าจากการเรียนรู้เพิ่มเติม เป็นวิทยานิพนธ์ที่ทำการรู้จำลายมือเขียนภาษาไทยแบบออนไลน์ ที่ใช้รหัสลูกโซ่แบบ GCC แบบ 8 ทิศทาง และใช้ loop pair policy มาช่วยในการสร้างโมเดลการรู้จำด้วย CFG, งานวิจัยของ H. Yuen [2] ทำการใช้รหัสลูกโซ่แบบ GCC ในการเข้ารหัสตัวอักษรเพื่อทำการรู้จำตัวอักษรลายมือเขียนภาษาอังกฤษแบบออนไลน์, งานวิจัยของ M. Okamoto และ K. Yamamoto [3] เป็นการนำเสนอการรู้จำลายมือเขียนตัวอักษรภาษาญี่ปุ่นแบบออนไลน์โดยใช้ทิศทางของ feature และการเปลี่ยนทิศทางของ feature ตามเข็มนาฬิกาและทวนเข็มนาฬิกามาเป็นลักษณะเด่นในการรู้จำ, งานวิจัยของ Xiaolin Li, Plamondon R. และ Parizeau M. [4]

เป็นการนำเสนอวิธีการใช้ Hidden Markov Model และลำดับของสโตร์ก (stroke) ในการรู้จำลายมือเขียนของตัวเลขแบบออนไลน์, งานวิจัยของ Joe, M. J. Lee และ H. Joo [5] เป็นการนำเสนอการรู้จำตัวอักษรเกาหลีแบบออนไลน์และแบบออฟไลน์ในระบบเดียวกัน โดยใช้วิธีการแยกสโตร์กออกมา แล้วทำการจัดเรียงสโตร์กขึ้นมาใหม่จากนั้นทำการแบ่งเซ็กเมนต์จากชุดของสโตร์ก

ปัญหาหลักของงานวิจัยทางด้านนี้ คือ การที่จะสร้างโมเดลการรู้จำอย่างไรให้มีความ generalize ที่มากเพียงพอที่จะรู้จำตัวอักษรตัวเดียวกันในลักษณะการเขียนที่แตกต่างกันได้ แต่จะต้องไม่ generalize มากเกินไป (over-generalize) จนทำให้รู้จำตัวอักษรตัวอื่นๆที่มีลักษณะใกล้เคียงกัน โดยโมเดลการรู้จำจะต้องมีความ specific ที่เหมาะสมที่จะบ่งบอกลักษณะเฉพาะตัวของตัวอักษรแต่ละตัวได้ และจะต้องไม่ specific มากเกินไปจนไม่ยอมรับตัวอักษรตัวเดียวกันแต่มีลักษณะการเขียนที่แตกต่างกันออกไป

ในงานวิจัยนี้มุ่งเน้นที่จะพัฒนางานวิจัยเดิมของ ปองเกษม พลสันติกุล [1] ซึ่งมีข้อด้อยของโมเดลการรู้จำ คือ มีความ generalize มากเกินไป เพราะมีการใช้การทำงานแบบ loop pair policy โดยในงานวิจัยนี้จะไม่มีการใช้งาน loop pair policy เพื่อต้องการสร้างโมเดลการรู้จำให้มีความ specific มากขึ้น และในงานวิจัยนี้มีการแบ่งทิศทางของเชนโค้ดซึ่งเป็นข้อมูลอินพุทตัวอักษรให้มีทิศทางที่ละเอียดขึ้นอีกหนึ่งเท่าตัวจากงานวิจัยเดิม [1] คือ มีการแบ่งทิศทางใหม่ของงานวิจัยนี้เป็น 16 ทิศทาง (งานวิจัยเดิม [1] มี 8 ทิศทาง) และในงานวิจัยนี้ยังได้นำค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาช่วยวิเคราะห์ในขั้นตอนการสร้าง DFA เพิ่มเติมด้วย เพื่อให้ DFA ที่ถูกสร้างขึ้นมามีความ specific ที่บ่งบอกลักษณะความแตกต่างของตัวอักษรแต่ละตัวได้มากขึ้น

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ในงานวิจัยนี้มีวัตถุประสงค์ที่จะหาวิธีการแบบใหม่ สำหรับการสร้างโมเดลการรู้จำด้วย DFA เพื่อให้โมเดลการรู้จำมีประสิทธิภาพมากขึ้นในขั้นตอนการสอนการรู้จำ โดยมุ่งหวังที่จะแก้ปัญหาของงานวิจัยเดิม [1] ที่มีความ generalize มากเกินไป

1.3 สมมติฐานของการศึกษา

ในงานวิจัยเดิม [1] มีการสร้าง CFG โดยใช้เชนโค้ด 8 ทางทิศ และใช้การทำงานของ loop pair policy มาช่วยในการรู้จำตัวอักษร ผลลัพธ์ที่ได้จากการทดลองพบว่ามีการเกิด over-generalize ซึ่งทำให้เกิดการรู้จำที่ผิดพลาด (accept false) ในเปอร์เซ็นต์ที่สูง

ในงานวิจัยนี้ผู้วิจัยได้คิดวิธีการที่จะลดความเป็น generalize เพื่อต้องการลดการเกิดการรู้จำที่ผิดพลาด (accept false) โดยการไม่นำเอากระบวนการ loop pair policy มาใช้ และมีการแบ่ง

ทิศทางของเชนโค้ดที่ละเอียดมากขึ้นเป็น 16 ทิศทาง และนำค่าของขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาช่วยในขั้นตอนการสร้าง DFA เพื่อการรู้จำตัวอักษร โดยมีสมมติฐานว่าวิธีการนี้จะลดการเกิด over-generalize ของงานวิจัยเดิม [1] และยังคงให้ผลการรู้จำที่ใกล้เคียงกับงานวิจัยเดิม [1]

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

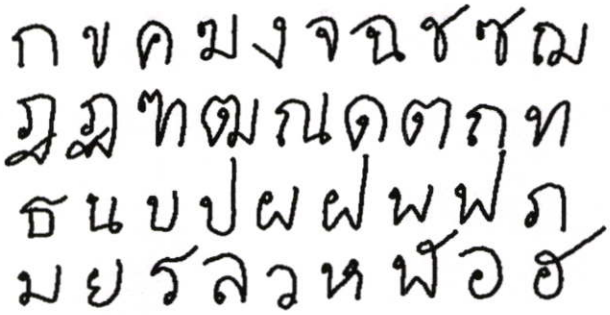
งานวิจัยนี้ได้นำเอาค่าลำดับของจุดข้อมูลที่เขียนด้วยปากกาอิเล็กทรอนิกส์ของตัวอักษรแต่ละตัว แล้วนำค่าจุดเหล่านั้นมาเข้ากระบวนการสร้างรหัสลูกโซ่แบบ GCC [2] ที่กำหนดทิศทางของเชนโค้ดให้มี 8 และ 16 ทิศทาง จากนั้นก็คำนวณหาขนาดและตำแหน่งของเชนโค้ดแต่ละตัวสำหรับเชนโค้ดแบบ 16 ทิศ ดังนั้นข้อมูลของตัวอักษรแต่ละตัวจะถูกแบ่งเป็น 2 ชุด ชุดแรกเป็นข้อมูลในรูปแบบที่เป็นเชนโค้ด 8 ทิศ ส่วนอีกชุดจะเป็นข้อมูลของเชนโค้ดแบบ 16 ทิศพร้อมค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัว

นำข้อมูลอินพุทของตัวอักษรตัวเดียวกันที่ถูกเขียนซ้ำมาหลายๆครั้ง ที่ถูกแปลงอยู่ในรูปแบบของ เชนโค้ด ขนาด และตำแหน่ง มาเข้ากระบวนการสร้างโมเดลการรู้จำด้วย DFA ซึ่งจะสร้างแยกเป็น DFA จากเชนโค้ด 8 ทิศซึ่งใช้เฉพาะค่าของเชนโค้ด และ DFA จากเชนโค้ด 16 ทิศพร้อมค่าขนาดและตำแหน่ง โดยแต่ละแบบก็จะสร้างแยกเป็น DFA ของแต่ละโมเดลตัวอักษรทั้ง 37 ตัวอักษร โดยอาศัยหลักการที่ว่าข้อมูลของตัวอักษรที่มีลักษณะคล้ายคลึงกันจะมีค่าของ เชนโค้ด ขนาด และตำแหน่ง ณ ตำแหน่งนั้นๆที่ใกล้เคียงกัน ซึ่งเราก็จะให้ข้อมูล ณ ตำแหน่งนั้นๆใช้โหนด (node) ของ DFA ร่วมกัน แต่ถ้าข้อมูลของตัวอักษร ณ ตำแหน่งนั้นๆมีลักษณะที่ไม่ใกล้เคียงกันคือแตกต่างกันออกไปเลย ก็จะมีการสร้างโหนดใหม่ขึ้นมาแทน เพื่อสร้างการรู้จำลักษณะที่แตกต่างกันออกไปเพิ่มเติม

1.5 ขอบเขตการวิจัย

งานวิจัยนี้ต้องการที่จะสร้างโมเดลการรู้จำด้วย DFA แบบอัตโนมัติจากข้อมูลตัวอย่างของตัวอักษรแต่ละตัว และต้องการให้โมเดลของตัวอักษรแต่ละตัวมีลักษณะที่บ่งบอกลักษณะเฉพาะตัวของตัวอักษรแต่ละตัวได้

ในวิทยานิพนธ์นี้มุ่งเน้นไปที่การวิจัยการรู้จำตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์ โดยใช้เฉพาะตัวอักษรภาษาไทยที่ใช้งานอยู่ในปัจจุบันที่มีลักษณะการเขียนแบบครั้งเดียวตั้งแต่จดปากกาลงไปแล้วลากเส้นเขียนตัวอักษรนั้นๆจนจบภายในครั้งเดียว โดยไม่ต้องยกปากกามาลากเขียนใหม่เพิ่มเติมเป็นครั้งที่สอง ซึ่งจะมีตัวอักษรภาษาไทยที่มีลักษณะการเขียนดังกล่าวอยู่ 37 ตัวอักษร ดังที่แสดงในรูปที่ 1.1



รูปที่ 1.1 ตัวอักษรลายมือเขียนภาษาไทยแบบที่เขียนครั้งเดียวจบที่ใช้งานอยู่ในปัจจุบัน ทั้งหมด 37 ตัวอักษร

1.6 ขั้นตอนของการศึกษา

1. ศึกษาทฤษฎีและวิธีการรู้จำตัวอักษรลายมือเขียนแบบออนไลน์จากงานวิจัยที่เคยมีผู้นำเสนอมาก่อนหน้านี้
2. ศึกษาทฤษฎีการทำงานของการเข้ารหัสลูกโซ่แบบ GCC เพื่อจะได้กำหนดจำนวนทิศทางที่เหมาะสม และคำนวณหาค่าขนาดและตำแหน่ง ของเชนโค้ดแต่ละตัวเพิ่มเติมขึ้นมา
3. ศึกษาทฤษฎีการทำงานของดีเทอร์มินิสติกไฟไนท์ออโตมาต้า (DFA) เพื่อจะได้นำมาประยุกต์ใช้กับงานวิจัยชิ้นนี้
4. ผู้วิจัยได้ทำการสร้างข้อมูลตัวอักษรที่ใช้วิจัยขึ้นมาใหม่อีกชุดหนึ่ง เพิ่มเติมจากของเดิมที่เคยมีผู้วิจัยก่อนหน้านี้เคยทำไว้ โดยเป็นการเขียนของคน 20 คน เขียนตัวอักษรทั้ง 37 ตัว แต่ละคนเขียนตัวอักษรทุกตัวๆละ 20 ครั้ง
5. นำข้อมูลตัวอักษรที่สร้างขึ้นมาใหม่แยกเป็น 2 ชุด โดยมีสัดส่วนเป็น 1:4 ชุดที่มีสัดส่วนเป็น 1 นำไว้ใช้สำหรับการทดสอบการรู้จำในภายหลัง ชุดที่มีสัดส่วนเป็น 4 ถูกนำมาใช้ในการสอนให้เกิดการรู้จำ
6. ข้อมูลตัวอักษรเก่าที่เคยมีผู้วิจัยก่อนหน้านี้เคยทำไว้ ก็ถูกแบ่งออกเป็น 2 ชุด เช่นกัน ในสัดส่วน 1:3 นำชุดที่มีสัดส่วนน้อยไปรวมกับชุดที่เอาไว้ใช้สำหรับทดสอบของชุดที่เป็นข้อมูลใหม่ ชุดที่มีสัดส่วนมากกว่าก็นำไปรวมกับชุดที่ใช้สำหรับสอนให้เกิดการรู้จำของชุดข้อมูลใหม่
7. นำข้อมูลของตัวอักษรที่ได้จากการเขียนซึ่งจะเป็นข้อมูลที่อยู่ในลักษณะของจุดข้อมูลมาเข้ากระบวนการสร้างเชนโค้ดแบบ GCC แบบ 16 ทิศทาง

8. นำค่าของจุดข้อมูลและเชนโค้ด มาหาค่าของขนาดและตำแหน่ง ของเชนโค้ดแต่ละตัวเพิ่มเติมขึ้นมา
9. นำข้อมูลเดิมทั้งหมดมาเข้ากระบวนการสร้างเชนโค้ดแบบ GCC แต่เป็นแค่ 8 ทิศทาง และไม่นำมาคำนวณหาค่าของขนาดและตำแหน่ง
10. นำค่าเชนโค้ด 16 ทิศ ที่มีค่าของขนาดและตำแหน่งของตัวอักษรแต่ละตัว มาสร้างโมเดลการรู้จำโดยใช้ DFA สร้างเป็นโมเดลการรู้จำตัวของอักษรแต่ละตัวแยกกัน
11. นำค่าเชนโค้ด 8 ทิศ ที่ไม่มีค่าขนาดและตำแหน่ง มาสร้างโมเดลการรู้จำโดยใช้ DFA สร้างเป็นโมเดลการรู้จำตัวของอักษรแต่ละตัวแยกกัน ซึ่งใช้กระบวนการของ loop pair policy มาช่วยในการสร้าง DFA
12. ทดสอบการรู้จำเพื่อวัดประสิทธิภาพของ DFA ที่สร้างจากเชนโค้ดทั้ง 2 แบบ โดยเอาข้อมูลของตัวอักษรของแต่ละแบบที่ไม่เคยใช้สอนโมเดลการรู้จำมาก่อนนำมาใช้ทดสอบ ซึ่งก็คือข้อมูลที่มีสัดส่วนน้อยกว่าที่ถูกแยกไว้ตั้งแต่ต้น
13. ประเมินผลจากการทดลองเปรียบเทียบ วิเคราะห์หาข้อดี-ข้อเสีย

1.7 รายละเอียดในแต่ละบท

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 6 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัย ความมุ่งหมายและวัตถุประสงค์ สมมติฐาน ทฤษฎีที่ใช้ ขอบเขตของการวิจัย และขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงการสร้างโมเดลการรู้จำ DFA ของงานวิจัยก่อนหน้า [1] การเข้ารหัสลูกโซ่ GCC แบบ 8 ทิศทาง ข้อมูลอินพุทที่นำมาใช้สร้างโมเดลการรู้จำ DFA กระบวนการสร้างโมเดลการรู้จำด้วย DFA การทำงานของ loop pair policy และปัญหาของการใช้กระบวนการทำงานของ loop pair policy

บทที่ 3 กล่าวถึงการสร้างโมเดลการรู้จำ DFA จากเชนโค้ด 16 ทิศ พร้อมค่าขนาดและตำแหน่ง อธิบายภาพรวมของระบบ การเข้ารหัสลูกโซ่ GCC แบบ 16 ทิศทาง การ Rename และ Reduce เชนโค้ด การหาค่าขนาด (Length) และตำแหน่ง (Position) ของเชนโค้ด ยกตัวอย่างข้อมูลอินพุทของตัวอักษรที่จะนำมาสร้างการรู้จำ DFA อธิบายถึงการยกเลิกการใช้งาน loop pair policy อธิบายการสร้าง DFA 16 ทิศพร้อมค่าขนาดและตำแหน่ง แสดงข้อมูลของ DFA จากการสอนแบบกราฟของตัวอักษรบางตัว และแสดงข้อมูลของ DFA ที่เป็นตารางจากการสอนของตัวอักษรทุกๆ ตัว

บทที่ 4 กล่าวถึงการทดลองและผลการทดลอง มีการยกตัวอย่างลักษณะลายมือเขียนของตัวอักษรแต่ละตัวที่ใช้ในการทดลองมาให้ดู แสดงจำนวนของตัวอักษรที่ใช้ train และ test ของ

ตัวอักษรแต่ละตัว แสดงผลการทดลองจากการทดสอบการรู้จำของตัวอักษรแต่ละตัวกับทุกๆ โมเดลตัวอักษร ของทั้งโมเดลที่สร้างจากเชนโค้ดแบบ 8 และ 16 ทิศ แล้ววิเคราะห์ผลการทดลอง

บทที่ 5 กล่าวถึงบทสรุป

เอกสารอ้างอิง

ภาคผนวกและงานวิจัยที่ได้รับตีพิมพ์

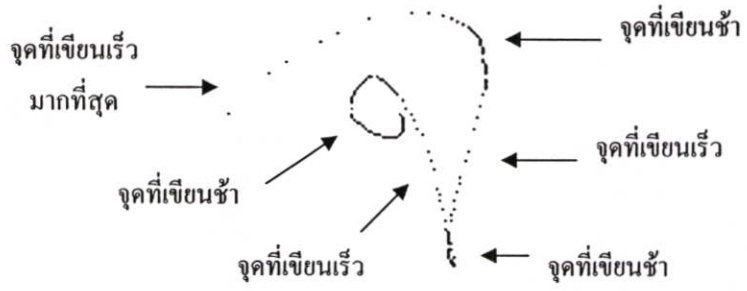
บทที่ 2
การสร้างโมเดลการรู้จำด้วย DFA หลักการและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงขั้นตอนการสร้างโมเดลการรู้จำ DFA ของงานวิจัยก่อนหน้าของ ปองเกษม พลสันติกุล [1] ซึ่งทำการรู้จำลายมือเขียนภาษาไทยแบบออนไลน์โดยใช้คอนแท็กฟรีแกรมม่าจากการเรียนรู้เพิ่มเติม เป็นวิทยานิพนธ์ที่ทำการรู้จำลายมือเขียนภาษาไทยแบบออนไลน์ ที่ใช้รหัสลูกโซ่แบบ GCC แบบ 8 ทิศทาง และใช้ loop pair policy มาช่วยในการสร้างโมเดลการรู้จำด้วย CFG และเนื่องด้วย CFG สามารถแปลงกลับไปมาเป็น DFA ได้ ในงานวิจัยนี้ จึงขออธิบายการทำงานของงานวิจัยก่อนหน้า [1] ด้วย DFA แทน ซึ่งจะสามารถเปรียบเทียบขั้นตอนกระบวนการทำงานของการสร้าง DFA ที่ทางผู้วิจัยได้คิดค้นเพิ่มเติมขึ้นมาในงานวิจัยนี้ได้สะดวกมากขึ้น ซึ่งจะอธิบายโดยละเอียดในบทถัดๆไป

ในข้อหัวย่อยของบทนี้จะอธิบายถึง การได้มาของข้อมูลอินพุทที่ได้มาจากจุดข้อมูลที่เขียนด้วยปากกาอิเล็กทรอนิกส์ การเข้ารหัสลูกโซ่ GCC แบบ 8 ทิศทาง ข้อมูลอินพุทที่นำมาใช้สร้างโมเดลการรู้จำ DFA กระบวนการสร้างโมเดลการรู้จำด้วย DFA การทำงานของ loop pair policy และปัญหาของการใช้กระบวนการทำงานของ loop pair policy

2.1 จุดข้อมูลที่ได้จากการเขียนด้วยปากกาอิเล็กทรอนิกส์

ข้อมูลลายมือเขียนที่ถูกใช้มาทำการวิจัยในงานวิจัยก่อนหน้า [1] ได้มาจากการเขียนด้วยปากกาอิเล็กทรอนิกส์ซึ่งจะให้ลำดับของจุดข้อมูลออกมา โดยจะมีการตั้งอัตรา Sampling ตามเวลาที่คงที่ คือ 100 จุดต่อ 1 วินาที ส่วนขนาดของตัวอักษรผู้เขียนบางคนอาจเขียนตัวเล็ก บางคนอาจเขียนตัวโต ขึ้นอยู่กับความถนัดของการเขียนของแต่ละบุคคล แต่โดยการประมาณแล้วขนาดของตัวอักษรจะอยู่ที่ความสูงและความกว้างตั้งแต่ 1 เซนติเมตร จนถึง 5 เซนติเมตร ดังนั้นลักษณะจุดข้อมูลที่ออกมาจะมีระยะห่างของแต่ละจุดที่ไม่เท่ากัน ซึ่งขึ้นอยู่กับขนาดและความเร็วในการเขียน ถ้าความเร็วในการเขียนช้าลำดับของจุดข้อมูลแต่ละจุดจะมีระยะห่างที่น้อย คือลักษณะของจุดข้อมูลจะออกมาติดๆกัน แต่ถ้าเขียนด้วยความเร็วที่สูงระยะห่างระหว่างของจุดข้อมูลก็จะห่างขึ้น ซึ่งถ้าเขียนด้วยความเร็วที่มากขึ้นระยะห่างระหว่างจุดข้อมูลก็จะยิ่งห่างกันมากขึ้นด้วย ดังที่แสดงในรูปที่ 2.1 ส่วนขนาดของตัวอักษรจะมีผลต่อจุดข้อมูลคือ ถ้าเขียนตัวอักษรด้วยขนาดที่โตกว่าก็จะมีจำนวนของจุดข้อมูลที่มากกว่า เพราะต้องใช้เวลาในการเขียนนานกว่าตัวอักษรที่มีขนาดเล็ก



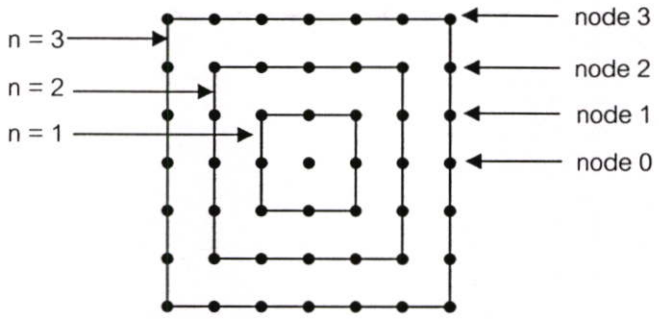
รูปที่ 2.1 แสดงลำดับจุดข้อมูลของตัวอักษรที่ได้จากการเขียนด้วยปากกาอิเล็กทรอนิกส์ ซึ่งในแต่ละส่วนของตัวอักษรถูกเขียนด้วยความเร็วที่แตกต่างกันไป

จุดข้อมูลลายมือเขียนที่เห็นจากในรูปตัวอย่าง 2.1 จะถูกเก็บอยู่ในรูปของข้อมูลคู่ลำดับ (x,y) ต่อเนื่องกันไป โดยเริ่มตั้งแต่การจดปากกาลงเขียนจนเขียนตัวอักษรนั้นเสร็จแล้วยกปากกาขึ้น จากนั้นก็นำค่า (x,y) เหล่านั้นมาเข้ารหัสเพื่อสร้างเชนโค้ดต่อไป

2.2 การเข้ารหัสลูกโซ่ GCC แบบ 8 ทิศทาง

รหัสลูกโซ่แบบ GCC (Generalize chain code) [2] เป็นการเข้ารหัสที่ใช้กลุ่มของรหัสที่เป็นวงรหัส (Coding ring) ที่มีจุดกึ่งกลางเพียงจุดเดียว สำหรับการเข้ารหัสแบบนี้จุดประสงค์ก็เพื่อให้มีการเข้ารหัสของข้อมูลที่ได้จากการ Sampling ตามเวลาให้น้อยสุด ในการเข้ารหัส GCC นั้น รหัส GCC จะประกอบด้วยตัวแปรสองตัว ตัวแรกจะเป็นตัวที่กำหนดวงรหัสที่มีลำดับเป็น n และวงรหัสจะประกอบไปด้วย M node เมื่อ M = (8 x n) และ n = 1,2,3,... และวงรหัสลำดับที่ n สามารถที่จะคำนวณได้จากค่าสูงสุดของระยะห่างระหว่างจุดสองจุดที่มีลำดับติดกันในแนวแกน x หรือแกน y แกนใดแกนหนึ่ง ส่วนตัวแปรตัวที่สองคือ node ที่มีอยู่ในแต่ละวงรหัส ซึ่ง node มีลำดับเป็น i เมื่อ i = 0,1,2,...(8 x n) นับ node 0 เป็นตัวเริ่มต้นและนับทวนเข็มนาฬิกาจนวนครบวงรอบ ดังที่แสดงในรูป 2.2

โดยค่า i หาได้จากองศาที่ทำมุมกันของจุดกึ่งกลางกับจุดที่ต้องการหาค่า ว่ามีค่ามุมเข้าใกล้ i ตัวใดมากที่สุดในวงรหัสที่ n ที่คำนวณได้จากข้างต้น ก็จะได้ว่า node ตัวที่ต้องการหามีค่าเป็นลำดับที่ i ตัวนั้น โดยทุกๆจุดข้อมูลที่ได้มาจากการเขียนด้วยปากกาอิเล็กทรอนิกส์ยกเว้นจุดเริ่มต้นสามารถที่จะกำหนดเป็นรหัสของสองตัวแปรนี้ได้ บนพื้นฐานว่าจุดกึ่งกลางของวงรหัสคือจุดที่มีการเข้ารหัสก่อนหน้า และหลังจากหาค่า n และ node ได้แล้วก็นำมาคำนวณหาค่าทิศทางอีกครั้ง



รูปที่ 2.2 แสดงส่วนประกอบของรหัสลูกโซ่แบบ GCC ที่มีวงรหัส n และจำนวน node ที่มีในแต่ละวงรหัส

ในการเข้ารหัสลูกโซ่แบบ GCC สิ่งที่ต้องคำนวณหาก็คือ เวกเตอร์ node ที่อยู่ใกล้ที่สุด ที่ตัดกับเส้นรอบวงซึ่งเป็นวงรหัสสี่เหลี่ยมที่มีหลายวงรหัส (Ring) โดยทุกวงรหัสจะมีจุดกึ่งกลางเพียงจุดเดียวคือจุดที่ทำการเข้ารหัสก่อนหน้านี้ ผลที่ได้ว่าจะออกมาเป็นวงรหัสใดนั้นขึ้นอยู่กับจุดนั้นๆมีระยะห่างจากจุดกึ่งกลางเป็นเท่าใด เช่น ถ้าจุดนั้นห่างออกไปมีค่าเป็นสามวงรหัสที่ถูกเลือกก็จะเป็นวงที่สาม ซึ่งจะสามารถคำนวณหาได้ตามสมการที่ (2.1)

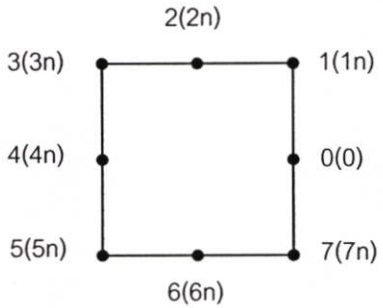
$$n = \left| \max (x_1 - x_2) \right| \text{ หรือ } n = \left| \max (y_1 - y_2) \right| \qquad (2.1)$$

เมื่อได้ค่า n เรียบร้อยแล้ว ก็มาหาค่า node ว่าเป็นลำดับที่ i ใด โดยการคำนวณหาค่าองศาของมุมจากจุดข้อมูลทั้งสอง จุดแรกจะเป็นจุดกึ่งกลางของวงรหัสซึ่งเป็นจุดที่มีการเข้ารหัสก่อนหน้า ส่วนอีกจุดจะเป็นจุดที่ต้องการหาค่า node ออกมา เมื่อคำนวณหาค่ามุมมาได้แล้วก็นำมาเทียบเคียงว่าค่ามุมที่ได้เข้าใกล้กับ i ลำดับที่เท่าไรของวงรหัสที่ n นั้น ค่า node ที่ต้องการหาก็จะเป็นค่าของ i ตัวนั้น

จากนั้นเราจะทำการ Normalized GCC (NGCC) ให้เป็นรหัส Ci ซึ่งสามารถคำนวณได้ตามสมการที่ (2.2) เพื่อเป็นการหาค่าทิศทางของเชนโค้ด

$$Ci = node / n \qquad (2.2)$$

NGCC จะมีค่าเป็น Dynamic โดยจะมีขอบเขตอยู่ในช่วง $0 < Ci < 8$ เพราะทำการลด (reduce) วงรอบของวงรหัสลงมาให้เหลือเพียงวงเดียว ดังนั้น NGCC จะมีเพียง 8 node หลักๆ เท่านั้นดังแสดงรูปที่ 2.3



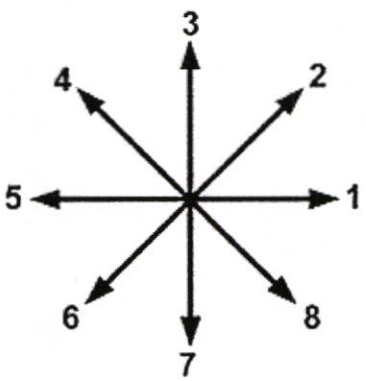
รูปที่ 2.3 แสดง NGCC หรือ GCC ที่ทำการ Normalized ซึ่งจะเหลือเพียง 8 node ต่อวงรหัส

การทำ NGCC เพื่อหาค่า Ci จะใช้อ้างถึงค่าของทิศทางที่ทำการเปรียบเทียบกับทิศทางที่ได้กำหนดขึ้นมา ซึ่งแบ่งออกเป็น 8 ทิศทาง ดังแสดงอยู่ในรูปที่ 2.4 การหาค่าของ Ci จะถูกคำนวณออกมาได้ลงตัวเป็นเลขจำนวนเต็มหรือสามารถจะออกมาไม่ลงตัวซึ่งจะมีค่าเป็นเลขทศนิยมด้วยก็ได้ สามารถเป็นไปได้ทั้ง 2 แบบ แต่จะอยู่ในช่วง $0 < Ci < 8$ แต่เนื่องจากเรากำหนดทิศทางให้มีค่าเริ่มต้นเป็นทิศ 1 แทนที่จะเป็นทิศ 0 เพราะฉะนั้นเรากำหนดให้ $Ci\_new = Ci + 1$ ขอบเขตของ Ci_new จะเป็น $1 < Ci\_new < 9$

ตัวอย่างที่ 2.1 คำนวณค่า Ci ได้ออกมาเป็น 0.78 เราก็นำมาหาค่า Ci_new ใหม่ได้เป็น $0.78 + 1 = 1.78$ ซึ่งค่าที่ออกมาใหม่ก็ถูกเพิ่มค่าทิศทางไปอีก 1 แต่ค่าที่ได้มาก็ยังเป็นค่าที่เป็นทศนิยมอยู่ จะเห็นว่าค่าส่วนที่เป็นทศนิยมมีค่าที่เกินครึ่งขึ้นไป คือถ้ามีค่าตั้งแต่ 0.5 เป็นต้นไปก็จะถูกปัดทิศทางขึ้นไป ดังนั้นเราจะเรียกเชนโค้ดตัวนี้ว่าอยู่ในทิศ 2

ตัวอย่างที่ 2.2 คำนวณค่า Ci ได้ออกมาลงตัวซึ่งเท่ากับ 4 จากนั้นหาค่า $Ci\_new = 4 + 1 = 5$ ดังนั้นเชนโค้ดตัวนี้อยู่ในทิศ 5

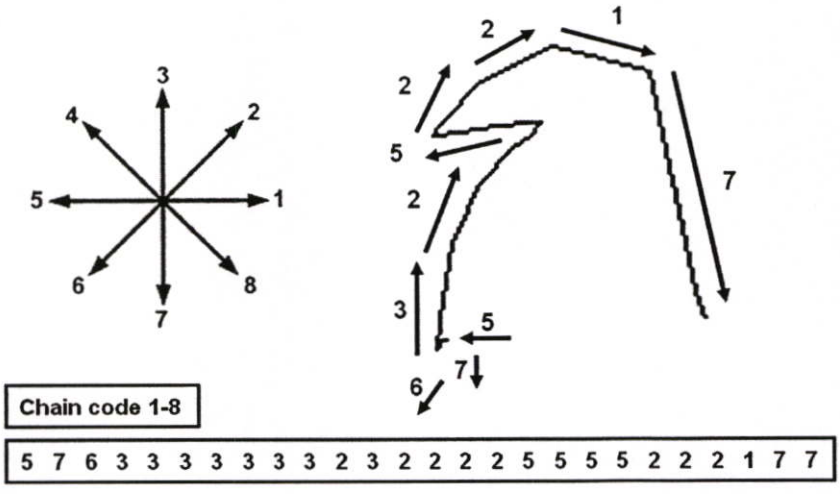
ตัวอย่างที่ 2.3 คำนวณค่า Ci ได้ 7.23 จากนั้นหาค่า $Ci\_new = 7.23 + 1 = 8.23$ ดังนั้นเชนโค้ดตัวนี้อยู่ในทิศ 8



รูปที่ 2.4 แสดงทิศทางของเชนโค้ดที่ได้กำหนดขึ้นมาแบบ 8 ทิศทาง

2.3 ข้อมูลอินพุทที่นำมาใช้สร้างโมเดลการรู้จำ DFA

ข้อมูลอินพุทของตัวอักษรแต่ละตัวที่นำมาสร้างเป็นโมเดลการรู้จำ DFA ของงานวิจัยเดิม [1] หลังจากเข้าสู่กระบวนการสร้างเชนโค้ดแบบ 8 ทิศ เรียบร้อยแล้ว ก็จะได้ข้อมูลของตัวอักษรแต่ละตัวออกมาเป็นสาย string ของเชนโค้ดเรียงต่อการกันไป ตั้งแต่เริ่มต้นจดปากกาอิเล็กทรอนิกส์ลงเริ่มเขียนจดเขียนตัวอักษรเสร็จแล้วยกปากกาขึ้น จะได้เป็นสาย string ต่อเนื่องกันไปที่มีทิศทางตั้งแต่ 1-8 ทิศ เรียงต่อกันไป ขนาดของสาย string จะสั้นหรือยาวมากน้อยเพียงใด ขึ้นอยู่กับลักษณะการเขียนของผู้เขียนที่เขียนตัวอักษรแต่ละตัว ดังที่ได้อธิบายไปแล้วในหัวข้อก่อนหน้า รูปตัวอย่างของสาย string ของเชนโค้ดของตัวอักษร “ก” ที่จะใช้เป็นอินพุทเพื่อจะส่งต่อไปยังกระบวนการสร้างโมเดลการรู้จำ DFA ต่อไป ได้แสดงอยู่ในรูปที่ 2.5



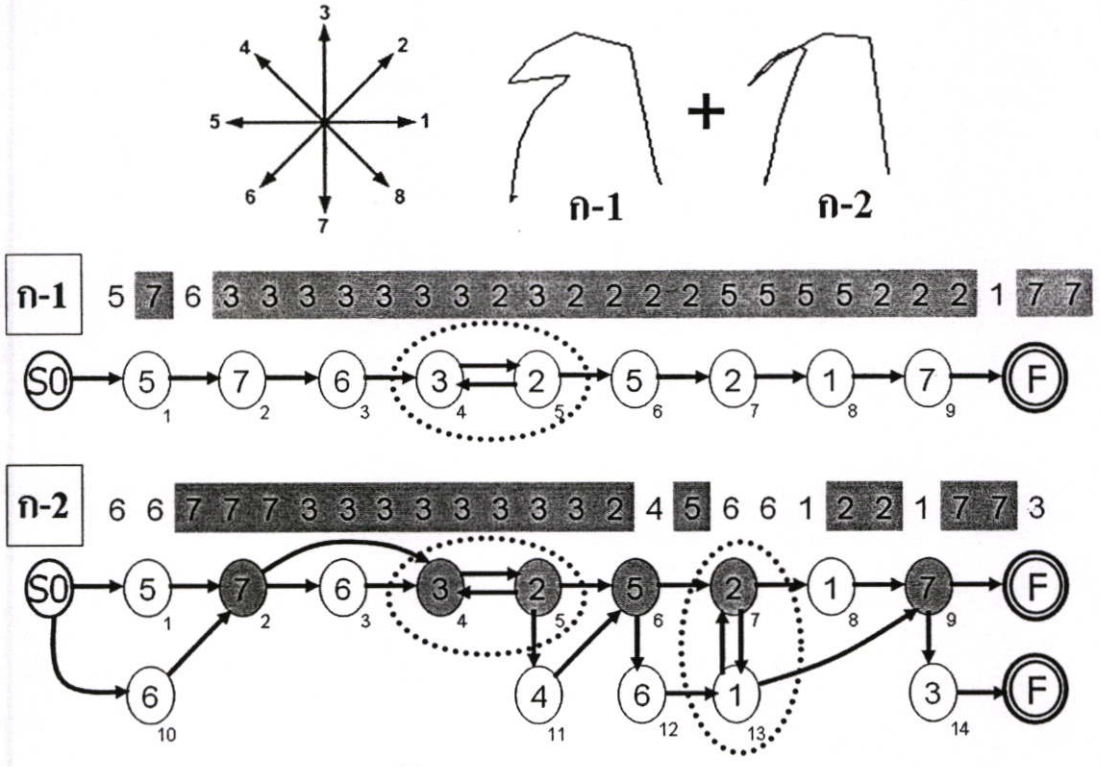
รูปที่ 2.5 ตัวอย่างเชนโค้ดแบบ 8 ทิศของตัวอักษร “ก”

2.4 การสร้างโมเดลการรู้จำ DFA จากเชนโค้ด 8 ทิศ

การสร้างโมเดลรู้จำ DFA จากการใช้เชนโค้ด 8 ทิศ จะนำเอาค่าเชนโค้ดของตัวอักษรนั้นๆ ในแต่ละครั้งของการเขียนนำมาสร้างรวมกันเป็นโมเดลของ DFA เพียงตัวเดียว เช่น โมเดลของ “ก” ก็จะมีโมเดลเดียว โมเดล “ข” ก็มีเพียงโมเดลเดียว โดยเราจะนำเชนโค้ดของตัวอักษรนั้นๆตั้งแต่ ก-1, ก-2, … ถึง ก-n มาสร้างรวมกันเป็นโมเดลของ “ก” เพียงโมเดลเดียว

โดยมีหลักในการสร้างรวมของ DFA คือจะทำการสร้าง link เชื่อมต่อไปยังโหนดที่มีค่าทิศทางเหมือนกันกับเชนโค้ดที่รับเข้ามา (การใช้โหนดร่วมกัน) แล้วให้ค่า cost เป็น 1 จากนั้นจะทำการสร้างโหนดใหม่ของทิศทางนั้นเพิ่มขึ้นอีกมาด้วยแล้วให้ค่า cost เป็น 2 ทำทั้ง 2 ขั้นตอนในทุกๆ state ของการรับเชนโค้ดแต่ละตัวมาสร้างเพิ่มเข้าไปใน DFA เดิม ทำให้ในการสร้างรวมของหลายๆ

state ก็จะมีหลายๆเส้นทางเกิดขึ้นมา เมื่อทำการสร้างรวมจนจบก็หาผลรวมในของแต่ละเส้นทาง แล้วเลือกเส้นทางที่มีค่า cost ที่น้อยที่สุดเพียงเส้นทางเดียวเป็นโมเดลของ DFA ที่ถูกสร้างรวม เรียบร้อยแล้ว และมีการนำวิธีการ loop pair policy มาใช้จับคู่โหนดที่มีทิศทางเป็นขั้นบันได ตัวอย่างของการสร้างรวม DFA ของเชนโค้ด 2 สายแรกที่สมมติว่าเลือกสายที่มีค่า cost ที่น้อยที่สุด เรียบร้อยแล้ว แสดงอยู่ในรูปที่ 2.6



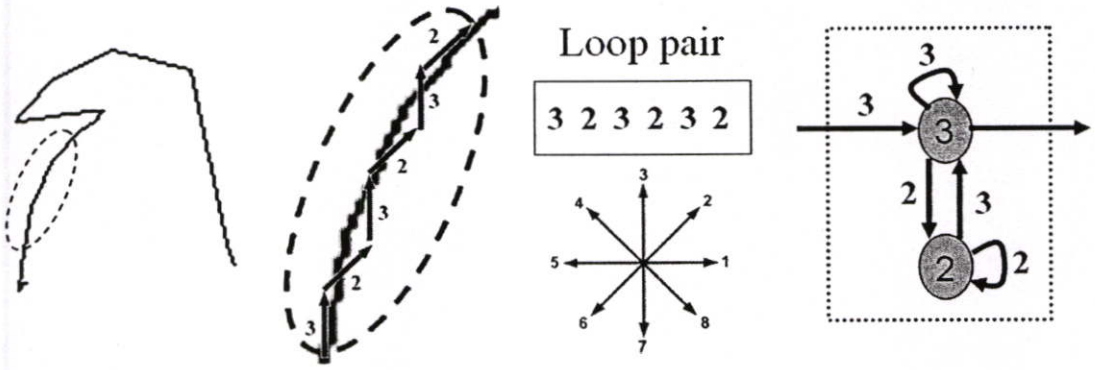
รูปที่ 2.6 ตัวอย่างการสร้างรวมโมเดล DFA ของ ก-1 และ ก-2 ของเชนโค้ดแบบ 8 ทิศ

จากในรูปที่ 2.6 จะเห็นว่าค่าเชนโค้ดและโหนดตัวที่มีการไฮไลท์สีเข้ม จะหมายถึงโหนดนั้นๆมีการใช้โหนดร่วมกันของทั้ง ก-1 และ ก-2 ซึ่งได้แก่โหนดหมายเลข 2, 4, 5, 6, 7 และ 9 สำหรับคู่โหนดที่อยู่ภายในเส้นประ คือโหนดหมายเลข 4 กับ 5 และโหนดหมายเลข 7 กับ 13 ก็จะเป็นลักษณะของโหนดที่เป็น loop pair จากการนำกระบวนการ loop pair policy มาใช้ ซึ่งจะอธิบายการทำงานและปัญหาของ loop pair policy ในหัวข้อถัดไป

2.5 Loop pair policy

Loop pair policy ที่ใช้ในงานวิจัยของ ปองเกษม พลสันติกุล [1] คือการใช้คู่ลูป (loop pair) ของโหนดเพียง 2 ตัวที่ใช้แทนทิศทาง 2 ทิศที่มีค่าทิศทางต่างกันแค่ 1 ค่า ซึ่งเป็นคู่ของทิศทางที่เป็น

ส่วนโค้งของตัวอักษรที่มีลักษณะของเชนโค้ดซิกแซ็กเหมือนขั้นบันไดต่อเนื่องกันไป loop pair จะสร้างเพียง 2 โหนดที่ชี้หมุนวนกันไปแทนที่การใช้โหนดหลายๆตัวในการสร้าง DFA เพื่อการรู้จำ ดังแสดงอยู่ในรูป 2.7



รูปที่ 2.7 แสดงการทำงานของ loop pair policy

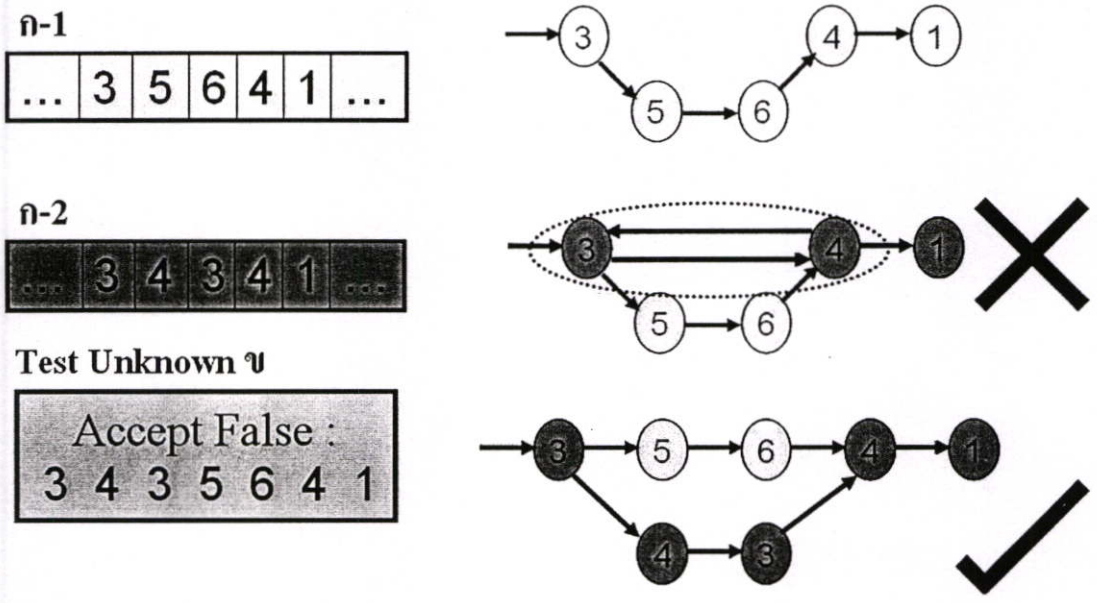
ข้อดีของการใช้ loop pair คือการลดจำนวนโหนดที่ใช้สร้างในการรู้จำ แต่ข้อเสียคือจะมีการรู้จำที่ generalize มากจนเกินไป (over-generalize) ทำให้เกิดการจำจดลักษณะนอกเหนือไปจากที่ได้เคยสอนไว้ ทำให้เกิดการรู้จำที่ผิดพลาด (accept false) ในขั้นตอนการทดสอบการรู้จำ ซึ่งจะอธิบายโดยละเอียดในหัวข้อถัดไป

2.6 การเกิด over-generalize จากการใช้ loop pair policy

ในขั้นตอนการสร้างโมเดลการรู้จำของ DFA แบบ 8 ทิศ เราได้นำกระบวนการของ loop pair policy ซึ่งมีการสร้าง loop pair ของเชนโค้ดคู่ที่มีลักษณะซิกแซ็กเป็นขั้นบันได ปัญหาในการสร้าง loop pair คือจะเกิด over-generalize ทำให้เกิดการรู้จำลักษณะที่นอกเหนือไปจากลักษณะที่เคยสอนไว้ ทำให้เกิดการรู้จำที่ผิดพลาด (accept false) เนื่องจากตัวอักษรอื่นที่ไม่ใช่ตัวอักษรที่ตรงกับโมเดลที่ทดสอบสามารถผ่านการทดสอบจากโมเดลนั้นได้

ยกตัวอย่างเช่น ในขั้นตอนการสร้างโมเดล DFA ของตัว “ก” ถ้า sting สายแรกที่รับมาของ “ก-1” มีเชนโค้ด เป็น ... 3, 5, 6, 4, 1, ... ซึ่งนำมาสร้างเป็นโมเดลตั้งต้น หลังจากนั้นนำ string สายที่สองคือ “ก-2” ที่มีเชนโค้ดเป็น ... 3, 4, 3, 4, 1, ... มาสร้างรวมกับ string สายแรก ก็จะเกิดลักษณะของ loop pair ขึ้นมา ทำให้โมเดล DFA ของตัว “ก” ตอนนี้สามารถ ยอมให้ string ที่มีเชนโค้ดเป็น ... 3, 4, 3, 5, 6, 4, 1, ... ผ่านการทดสอบได้ ซึ่งเป็น string สายที่ไม่ได้ตั้งใจให้เกิดขึ้นจากการสอน แต่เกิดขึ้นมาเพราะการสร้าง loop pair หลังจากนั้นสมมติว่าเรานำ unknown ที่เป็นข้อมูลของตัวอักษร “ข” ตัวหนึ่งมาทดสอบกับโมเดลของตัว “ก” ข้างต้น ซึ่ง unknown “ข” ตัวนั้นมีค่า string

เป็น ... 3, 4, 3, 5, 6, 4, 1, ... ซึ่งเป็นลักษณะการเขียนปกติของตัว “ข” อยู่แล้ว ทำให้ unknown “ข” ตัวนี้ผ่านการทดสอบจากโมเดลของตัว “ก” และอาจมี unknown ของตัวอักษรตัวอื่นๆของตัว “ข” อีกหลายๆตัวผ่านการทดสอบนี้ด้วยเช่นกัน ซึ่งทำให้เกิดผล accept false เป็นจำนวนมากจากกระบวนการนี้ หากไม่มีการสร้าง loop pair ขึ้นมา ก็จะไม่มีเกิด accept false ขึ้นมาในกระบวนการนี้ จากตัวอย่างที่สมมติขึ้นมาข้างต้นจะแสดงอยู่ในรูปที่ 2.8



รูปที่ 2.8 การเกิด accept false จากการสร้าง loop pair

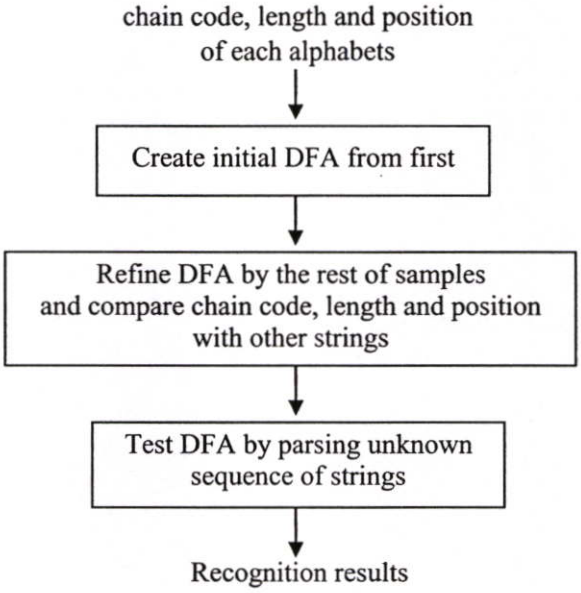
บทที่ 3
การสร้างโมเดลการรู้จำด้วย DFA

ในบทนี้จะกล่าวถึงขั้นตอนการสร้างโมเดลการรู้จำ DFA ของงานวิจัยชิ้นนี้ ซึ่งได้คิดกระบวนการสร้างโมเดลการรู้จำเพิ่มเติมขึ้นมา โดยมุ่งหวังที่จะพัฒนางานวิจัยก่อนหน้า [1] ให้มีประสิทธิภาพมากขึ้น โดยการใช้ข้อมูลอินพุทที่เป็นเชนโค้ด 16 ทิศ พร้อมทั้งค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาช่วยวิเคราะห์ในกระบวนการสร้างการรู้จำ

โดยขั้นตอนต่างๆก่อนการนำมาสร้างโมเดลการรู้จำ DFA จะคล้ายคลึงกันกับขั้นตอนของในงานวิจัยก่อนหน้า [1] ซึ่งได้อธิบายไว้โดยละเอียดแล้วในบทที่ผ่านมา แต่ในบางขั้นตอนจะมีความแตกต่างกันออกไป ซึ่งจะได้อธิบายขั้นตอนที่แตกต่างกันออกไปจากงานวิจัยก่อนหน้าในหัวข้อย่อยต่อไปในบทนี้

3.1 ภาพรวมของระบบ

เป้าหมายของงานวิจัยนี้ คือการนำเอาข้อมูลของตัวอักษรลายเขียนในรูปแบบของเชนโค้ดที่ผู้วิจัยได้กำหนดทิศทางให้มีมากกว่าในงานวิจัยเก่าเป็น 2 เท่าตัว คือจากในงานวิจัยของ ปองเกษม พลสันติกุล [1] ที่ใช้เชนโค้ดแบบ 8 ทิศ ส่วนในงานวิจัยนี้กำหนดให้เป็น 16 ทิศ พร้อมทั้งหาค่าขนาดและตำแหน่งเพิ่มเติมขึ้นมาเพื่อให้มีข้อมูลที่ใช้ในการวิเคราะห์มากขึ้นในกระบวนการสร้างโมเดลการรู้จำด้วย DFA การสร้างโมเดลการรู้จำด้วย DFA ลำดับแรกนำค่าของเชนโค้ดตัวแรกมาสร้างเป็นโมเดล DFA ตัวตั้งต้น จากนั้นก็นำเชนโค้ดของตัวอักษรตัวนั้นๆที่ได้จากการเขียนหลายๆครั้งจากหลายๆคน มาสร้างรวมเข้าไปกับโมเดลตัวตั้งต้น ทำเช่นนี้ไปเรื่อยๆจนครบจำนวนของข้อมูลตัวอักษรที่จะนำมาใช้สอนโมเดลการรู้จำของตัวอักษรแต่ละตัว หลังจากสร้างโมเดลการรู้จำสำหรับตัวอักษรแต่ละตัวเสร็จเรียบร้อยแล้ว ก็จะทำการวัดผลการรู้จำโดยการนำข้อมูลของตัวอักษรที่ไม่เคยถูกใช้สอนมาก่อนมาใช้ในการทดสอบ ซึ่งกระบวนการที่ได้กล่าวมาทั้งหมดจะแสดงอยู่ในรูปที่ 3.1



รูปที่ 3.1 ภาพรวมของระบบ

3.2 การเข้ารหัสลูกโซ่ GCC แบบ 16 ทิศทาง

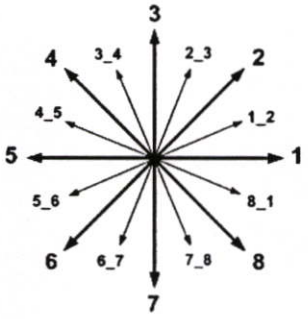
การเข้ารหัสลูกโซ่แบบ 16 ทิศ มีขั้นตอนเบื้องต้นเหมือนกันกับการเข้ารหัสแบบ 8 ทิศ ทุกประการ ซึ่งได้อธิบายกระบวนการโดยละเอียดแล้วในบทที่ 2 หัวข้อที่ 2.2 แต่สิ่งที่ต่างกันคือ ผู้วิจัยได้กำหนดทิศทางของเชนโค้ดใหม่ให้ละเอียดมากขึ้นอีกหนึ่งเท่าตัว คือมี 16 ทิศทาง ดังที่แสดงไว้ในรูปที่ 3.2

ดังนั้นในหัวข้อนี้ จึงขอยกเอาตัวอย่างการคำนวณหาค่า Ci_new เพื่อจะนำมาระบุทิศทางใหม่แบบ 16 ทิศ ของค่าเชนโค้ดแต่ละตัวที่คำนวณได้ โดยจะอ้างถึงตัวอย่างเดิมในบทที่ 2 หัวข้อที่ 2.2 ของตัวอย่างที่ 2.1, 2.2 และ 2.3 ว่าหากนำค่า Ci_new จากตัวอย่างเดิมเหล่านั้นมากำหนดทิศทางใหม่แบบ 16 ทิศ จะได้ทิศทางใหม่เป็นทิศใดบ้าง โดยจะแสดงอยู่ในตัวอย่างที่ 3.1, 3.2 และ 3.3 ซึ่งจะตรงกับตัวอย่างเดิมในตัวอย่างที่ 2.1, 2.2 และ 2.3 ตามลำดับ

ตัวอย่างที่ 3.1 คำนวณค่า Ci ได้ออกมาเป็น 0.78 เราก็นำมาหาค่า Ci_new ใหม่ได้เป็น 0.78 + 1 = 1.78 ซึ่งค่าที่ออกมาใหม่ก็ถูกเพิ่มค่าทิศทางไปอีก 1 แต่ค่าที่ได้มาก็ยังเป็นค่าที่เป็นทศนิยมอยู่ ซึ่งเป็นค่าที่อยู่ในช่วงระหว่างทิศทางหลักคือทิศ 1 กับทิศ 2 ดังนั้นเราจะเรียกเชนโค้ดตัวนี้ว่าอยู่ในทิศ 1_2 ของเชนโค้ดแบบ 16 ทิศ

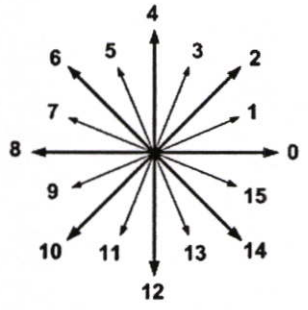
ตัวอย่างที่ 3.2 คำนวณค่า Ci ได้ออกมาลงตัวซึ่งเท่ากับ 4 จากนั้นหาค่า Ci_new = 4 + 1 = 5 ดังนั้นเชนโค้ดตัวนี้อยู่ในทิศ 5 ของเชนโค้ดแบบ 16 ทิศ

ตัวอย่างที่ 3.3 คำนวณค่า Ci ได้ 7.23 จากนั้นหาค่า Ci_new = 7.23 + 1 = 8.23 ดังนั้นเชนโค้ดตัวนี้อยู่ในทิศ 8_1 ของเชนโค้ดแบบ 16 ทิศ



รูปที่ 3.2 แสดงทิศทางของเชนโค้ดที่ได้กำหนดขึ้นมา 16 ทิศ แบบทิศหลักและทิศรอง

เพื่อความสะดวกให้การอ้างถึงทิศทางที่ได้กำหนดขึ้น ผู้วิจัยจึงได้ทำการ rename ชื่อของทิศทางแบบ 16 ทิศเสียใหม่ โดยทิศเก่าที่เป็นทิศ 1, 1_2, 2, 2_3, 3, ..., 8_1 จะถูกเปลี่ยนใหม่เป็นทิศ 0, 1, 2, 3, 4, ...,15 ดังแสดงในรูป 3.3



รูปที่ 3.3 แสดงทิศทางใหม่ของเชนโค้ดแบบ 16 ทิศ โดยมีทิศ 0-15

ต่อไปจะเป็นตัวอย่างการหารหัสลูกโซ่ (chain code) แบบ GCC [2] ของตัวอักษรลายมือเขียนโดยสมมติว่าจุดข้อมูลที่ได้จากการ Sampling มีจุดดังต่อไปนี้ P1 (188,344); P2 (187,342); P3 (186,341); P4 (184,340) รูปที่ 3.4 (ก) เป็นการหารหัสลูกโซ่ระหว่างจุด P1 และ P2 ก่อนอื่นต้องคำนวณหาวงรหัสก่อนว่าอยู่ในวงรหัสที่เท่าไร ซึ่งสามารถคำนวณดังได้นี้

$$n = \left| \max (x_1 - x_2) \right| = \left| \max (188 - 187) \right| = 1 \quad \text{หรือ}$$
$$n = \left| \max (y_1 - y_2) \right| = \left| \max (344 - 342) \right| = 2$$

จะได้ว่าค่าวงรหัสเป็น 2 ซึ่งมาจากค่าในแนวแกน y เพราะในแนวแกน y มีค่าของจุดห่างกันเป็น 2 ซึ่งมากกว่าจากแนวแกน x ซึ่งห่างกันแค่ 1 จากนั้นก็จะทำการหาว่าจุดที่ได้นั้นตกอยู่ใน node ใดในวงรหัสที่ 2 ซึ่งในวงรหัสที่ 2 จะมีจำนวน node ทั้งหมดเท่ากับ 8*2 = 16 node ซึ่งในที่นี้เมื่อคำนวณหาองศาจากค่ามุมของจุดข้อมูล P1 ซึ่งเป็นจุดกึ่งกลางวงรหัสกับจุด P2 ซึ่งเป็นจุดที่

ต้องการนำมาหาค่า node เมื่อคำนวณหาค่ามุมได้แล้วก็นำมาเทียบเคียงกับค่า i ในวงรหัสที่ 2 ซึ่งมีทั้งหมด 16 node แต่ค่ามุมที่ได้มาเทียบเคียงแล้วเข้าใกล้ node ที่ 11 มากที่สุด (เริ่มต้นนับ node 0 ในวงรหัสที่ 2 จากแกน x แล้วนับทวนเข็มนาฬิกา) ดังนั้นจึงได้ค่า node = 11 ดังรูปที่ 3.4 (ข) เมื่อได้ค่า node แล้วก็จะทำการคำนวณหาค่ารหัส Ci ซึ่งจะคำนวณได้จาก

$$Ci = node / n = 11 / 2 = 5.5$$

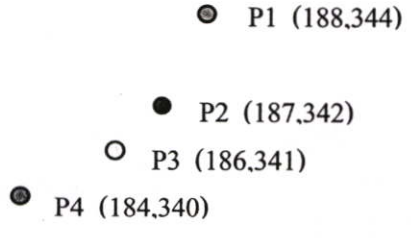
แต่เนื่องจากทิศทางของเราเริ่มต้นที่ 1 ไม่ใช่ 0 ดังนั้นจึงต้องบวก 1 เพิ่มเข้าไป คือ

$$Ci\_new = Ci + 1 = 5.5 + 1 = 6.5 = 6\_7$$

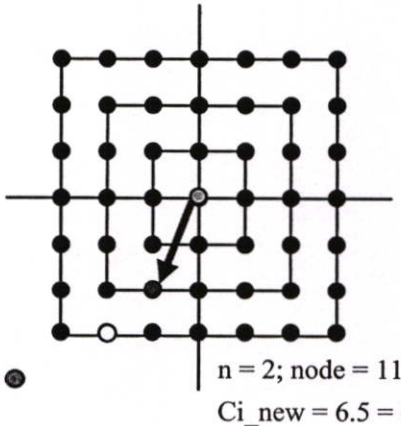
เพื่อความสะดวกในการอ้างถึงเชนโค้ดจึงทำการ rename ทิศทางใหม่ให้เป็น 0-15 ซึ่งจะได้เชนโค้ดทิศทางใหม่เป็น

$$\text{chain code} = 6\_7 \text{ (ทิศ 1-8)} = 11 \text{ (ทิศ 0-15)}$$

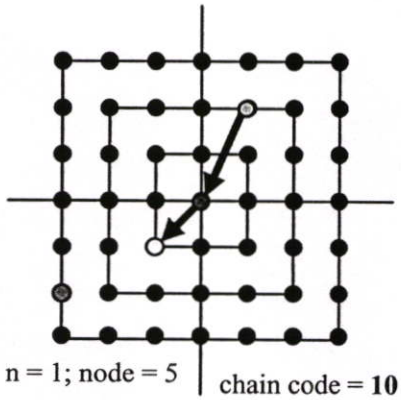
จากกระบวนการดังกล่าวข้างต้นทำให้เราสามารถหาค่าเชนโค้ดระหว่างจุด P1 และ P2 ได้ หลังจากนั้นจุดต่อๆไปก็จะทำเช่นเดียวกันนี้ไปเรื่อยๆไปจนครบทุกจุดที่ได้ในจากการเขียนตั้งแต่ต้นจนจบของตัวตัวอักษรแต่ละตัว จากรูปที่ 3.4 (ค) เป็นการหาค่าเชนโค้ดระหว่างจุด P2 และ P3 ส่วนรูปที่ 3.4 (ง) เป็นการหาค่าเชนโค้ดของจุด P3 และ P4 ส่วนรูปที่ 3.4 (จ) เป็นภาพที่แสดงเชนโค้ดทั้ง 3 ตัว ที่ได้มาจากการเข้ารหัสจากจุดข้อมูลตัวอย่างทั้ง 4 จุด



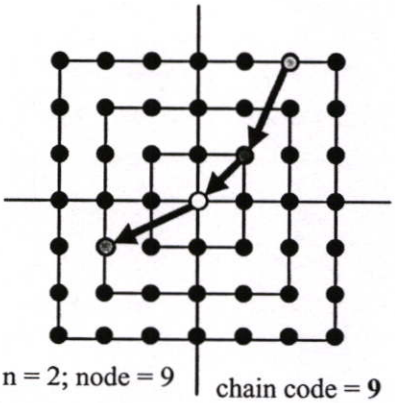
(ก) จุดข้อมูลตัวอย่างที่จะทำการหาเชนโค้ด



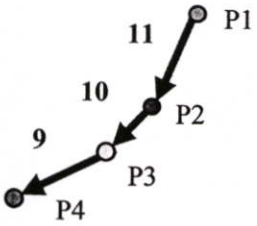
(ข) การหาค่าเชนโค้ดของจุด P1 และ P2



(ค) การหาค่าเชนโค้ดของจุด P2 และ P3



(ง) การหาค่าเชนโค้ดของจุด P3 และ P4

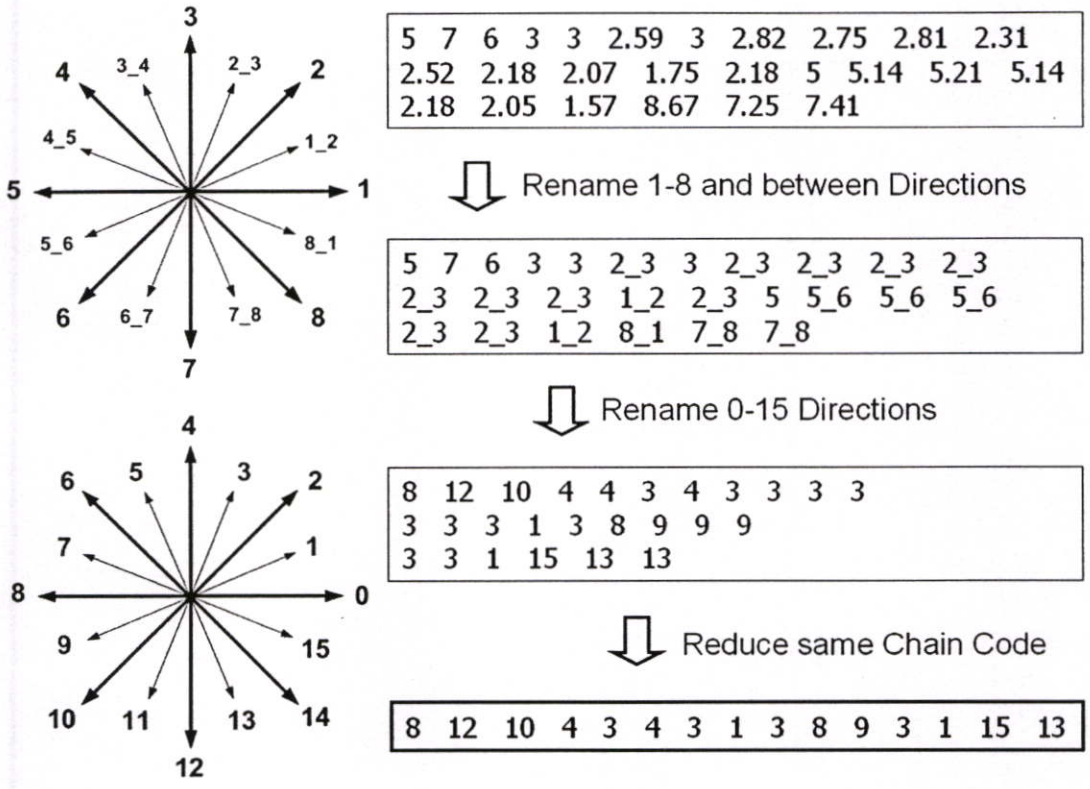


(จ) แสดงเชนโค้ดทั้ง 3 ตัว ที่สร้างมาจากจุด P1, P2, P3 และ P4

รูปที่ 3.4 แสดงตัวอย่างการเข้ารหัสลูกโซ่ (chain code) แบบ GCC ของจุด P1, P2, P3 และ P4

3.3 การ Rename และ Reduce เชนโค้ด

จากกระบวนการเข้ารหัสเพื่อสร้างเชนโค้ด ข้อมูลที่รับมาเพื่อเข้ารหัสจะเป็นจุดข้อมูลจากการเขียน สำหรับการสร้างเชนโค้ดแบบ 16 ทิศ เมื่อเข้ารหัสเสร็จจนถึงขั้นตอนสุดท้ายจะได้เป็นเชนโค้ดที่มีค่าทิศทางเป็น 0-15 ออกมา ซึ่งจริงๆผลที่ออกมาจากการเข้ารหัสครั้งแรกจะเป็นค่าของ Ci ที่มีค่าอยู่ในช่วง $0 < Ci < 8$ จากนั้นนำค่า $Ci + 1$ ก็จะได้เป็นค่า Ci_new ซึ่งจะมีค่าอยู่ในช่วง $1 < Ci\_new < 9$ จากนั้นก็แปลงให้เป็นเชนโค้ด 16 ทิศแบบที่เป็นทิศหลักและทิศรอง ซึ่งทิศรองเป็นทิศที่อยู่ระหว่างทิศหลัก 2 ทิศ แต่เพื่อความสะดวกในการอ้างถึงจึงได้ Rename เชนโค้ดโดยมีทิศทางใหม่เป็นทิศ 0-15 หลังจากนั้นเราก็จะได้เชนโค้ดที่เป็นทิศ 0-15 แต่เนื่องจากมีเชนโค้ดบางตัวที่มีค่าเหมือนกันอยู่ในตำแหน่งที่ติดกัน เราเลยยุบรวมเชนโค้ดที่ติดกันนั้นให้เป็นเชนโค้ดเพียงตัวเดียว (Reduce) เพื่อให้มีข้อมูลที่น้อยลง เพื่อความสะดวกในการนำข้อมูลเชนโค้ดเหล่านั้นเข้าไปเป็นข้อมูลอินพุทของกระบวนการรู้จำในภายหลัง ซึ่งกระบวนการที่ได้กล่าวมาข้างต้นนั้นได้แสดงไว้ในรูปที่ 3.5



รูปที่ 3.5 แสดงตัวอย่างการ Rename และ Reduce ของเชนโค้ดแบบ 16 ทิศ

3.4 การหาค่าขนาด (Length) และตำแหน่ง (Position) ของเชนโค้ด

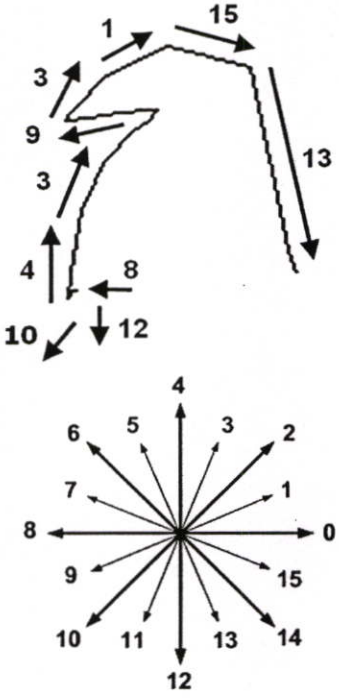
เชนโค้ดแต่ละตัวถูกสร้างมาจากค่าของจุดข้อมูล 2 จุดที่อยู่ติดกัน ดังนั้นเชนโค้ดแต่ละตัวก็จะมีค่าของขนาด (length) ด้วยเช่นกัน ซึ่งสามารถคำนวณได้จากระยะห่างระหว่างจุดทั้งสอง ตามที่ได้แสดงในสมการที่ (3.1)

$$length = \sqrt{(|x_1 - x_2|)^2 + (|y_1 - y_2|)^2} \qquad (3.1)$$

เมื่อเราได้ค่าขนาดของเชนโค้ดทุกตัวของการเขียนตัวอักษรนั้นๆแล้ว ก็นำค่าขนาดของเชนโค้ดทั้งหมดที่ได้มารวมกัน แล้วคิดสัดส่วนรวมทั้งหมดออกมาใหม่ให้มีค่าเป็น 100% แล้วคำนวณหาขนาดของเชนโค้ดแต่ละตัวใหม่ ว่าเชนโค้ดแต่ละตัวมีขนาดความยาวเป็นกี่เปอร์เซ็นต์เมื่อเทียบกับขนาดความยาวรวมของการเขียนจนจบตัวอักษรนั้นๆคือ 100%

เมื่อเราทราบขนาดของเชนโค้ดที่เทียบสัดส่วนใหม่ออกมาเป็นเปอร์เซ็นต์เรียบร้อยแล้ว เราสามารถนำค่า length ของเชนโค้ดแต่ละตัวตั้งแต่ตัวแรกนำมาบวกรวมกับค่า length ของเชนโค้ดตัว

ถัดไป บวกรวมกันไปทีละตัวไปเรื่อยๆจนไปถึงค่าของเชนโค้ดตัวสุดท้ายซึ่งจะต้องได้ค่าเป็น 100% พอดีเพราะเป็นเชนโค้ดตัวสุดท้าย ค่าที่บวกรวมกันนี้จะนำมาเป็นค่าที่ใช้อ้างถึงตำแหน่ง (position) ว่าเชนโค้ดตัวนั้นๆอยู่ในตำแหน่งใดภายในตัวอักษรนั้นๆ ซึ่งจะแสดงตัวอย่างอยู่ในรูปที่ 3.6



chain code	%length	%position
8	0.71	0.71
12	0.71	1.43
10	0.50	1.93
4	1.07	3.00
3	1.13	4.13
4	1.07	5.20
3	21.51	26.72
1	2.57	29.29
3	1.78	31.07
8	1.43	32.50
9	10.80	43.28
3	8.09	51.37
1	9.09	60.46
15	11.07	71.52
13	28.48	100.00

รูปที่ 3.6 แสดงตัวอย่าง chain code แบบ 16 ทิศ, length และ position ของตัวอักษร "ก"

เพราะฉะนั้นข้อมูลที่จะถูกส่งไปเป็นข้อมูลอินพุทสำหรับกระบวนการรู้จำในลำดับต่อไปของเชนโค้ดแบบ 16 ทิศ จะมีทั้ง 3 ส่วน คือ chain code, length และ position

3.5 ข้อมูลอินพุทที่นำมาใช้สร้างโมเดลการรู้จำ DFA

ในวิทยานิพนธ์ฉบับนี้จะทำการสร้างโมเดลการรู้จำ DFA ออกเป็น 2 แบบ เพื่อจะใช้เปรียบเทียบโมเดลการรู้จำของทั้งสองแบบ แบบแรกจะสร้างโดยใช้ข้อมูลอินพุทที่เป็นเชนโค้ด 8 ทิศทางเท่านั้น ซึ่งเป็นแบบของงานวิจัยก่อนหน้า [1] ส่วนแบบที่สองจะสร้างโมเดลการรู้จำโดยใช้เชนโค้ดแบบ 16 ทิศ พร้อมกับค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาช่วยวิเคราะห์ในการสร้างโมเดลการรู้จำ ซึ่งเป็นแบบที่ผู้วิจัยได้คิดเพิ่มเติมขึ้นมา

ข้อมูลอินพุทของเชนโค้ด 8 ทิศ จะมีค่าทิศทางที่อยู่ระหว่าง 1-8 เท่านั้น ต่อเนื่องกันไป ซึ่งจะไม่มีการ reduce เชนโค้ดตัวที่มีค่าเหมือนกันที่อยู่ในตำแหน่งที่ติดกันเหมือนของเชนโค้ด 16 ทิศ

เพื่อจะได้เห็นจำนวนของเชนโค้ดทั้งหมดซึ่งจะแสดงถึงลักษณะของการเขียนและความเร็วในการเขียนของตัวอักษรนั้นๆได้

ส่วนข้อมูลอินพุทของเชนโค้ด 16 ทิศ ก็จะมีค่าทิศทางอยู่ระหว่าง 0-15 และมีการ reduce เชนโค้ดตัวที่มีค่าเหมือนกันที่อยู่ในตำแหน่งติดกันเรียบร้อยแล้ว และเชนโค้ดแต่ละตัวก็ได้ทำการหาค่าของขนาดและตำแหน่งซึ่งเทียบออกมาอยู่ในรูปของเปอร์เซ็นต์เรียบร้อยแล้ว ซึ่งลักษณะของข้อมูลอินพุททั้งสองแบบของตัวอย่างตัวอักษร “ก” ตัวเดียวกัน ได้ถูกแสดงอยู่ในรูปที่ 3.7

chain code 1-8	5	7	6	3	3	3	3	3	3	3	2	3	2	2	2	2	5	5	5	5	2	2	2	1	7	7

chain code 0-15	8	12	10	4	3	4	3	1	3	8	9	3	1	15	13
length	0.71	0.71	0.50	1.07	1.13	1.07	21.52	2.57	1.78	1.43	10.79	8.09	9.09	11.07	28.48
position	0.71	1.43	1.93	3.00	4.15	5.20	26.72	29.29	31.07	32.50	43.28	51.37	60.46	71.52	100.00

รูปที่ 3.7 ตัวอย่างเชนโค้ดแบบ 8 และ 16 ทิศของตัวอักษร “ก” ตัวเดียวกัน พร้อมทั้งค่าขนาดและตำแหน่งของเชนโค้ดแบบ 16 ทิศ

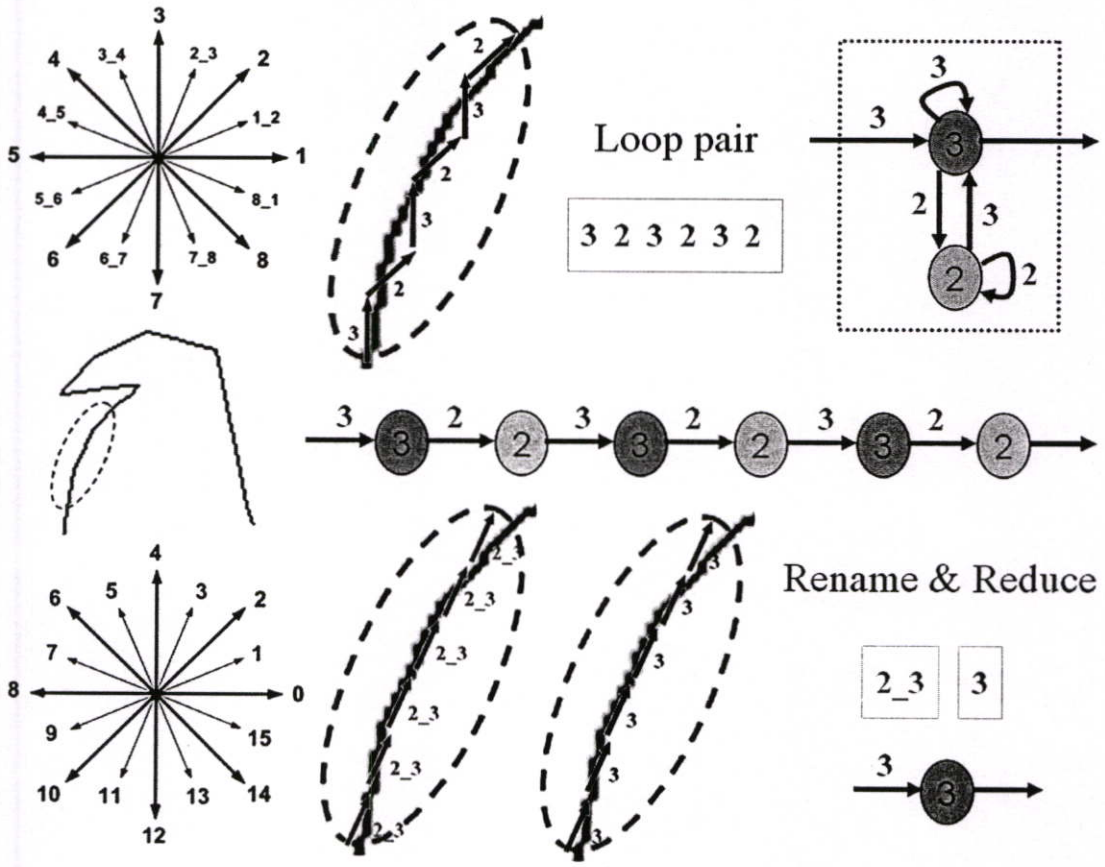
3.6 การยกเลิกการใช้งาน Loop pair policy ของ DFA 16 ทิศ

Loop pair policy ที่ใช้ในงานวิจัยของ ปองเกษม พลสันติกุล [1] คือการใช้คู่ลูป (loop pair) ของโหนดเพียง 2 ตัวที่ใช้แทนทิศทาง 2 ทิศที่มีค่าทิศทางต่างกันแค่ 1 ค่า ซึ่งเป็นคู่ของทิศทางที่เป็นส่วนโค้งของตัวอักษรที่มีลักษณะของเชนโค้ดซิกแซ็กเหมือนขั้นบันไดต่อเนื่องกันไป loop pair จะสร้างเพียง 2 โหนดที่ชี้หมุนวนกันไปแทนที่การใช้โหนดหลายๆตัวในการสร้าง DFA เพื่อการรู้จำ ข้อดีของการใช้ loop pair คือการลดจำนวนโหนดที่ใช้สร้างในการรู้จำ แต่ข้อเสียคือจะมีการรู้จำที่ generalize มากจนเกินไป (over-generalize) ทำให้เกิดการจำจดลักษณะนอกเหนือไปจากที่ได้เคยสอนไว้ ทำให้เกิดการรู้จำที่ผิดพลาด (accept false) ในขั้นตอนการทดสอบการรู้จำ

แต่เนื่องจากในงานวิจัยนี้เราได้กำหนดทิศทางใหม่ให้มากขึ้นเป็น 16 ทิศ ซึ่งมากกว่าในงานวิจัยของ ปองเกษม พลสันติกุล [1] ที่ใช้เพียง 8 ทิศ เพิ่มขึ้นไปอีก 1 เท่าตัว เพราะฉะนั้นโอกาสที่เชนโค้ดจะออกมาเป็นลักษณะซิกแซ็กที่เป็นขั้นบันไดก็จะลดน้อยลง จากเชนโค้ดเดิมของตัวอักษรตัวเดียวกันจากที่เคยแบ่งเพียงแค่ 8 ทิศ เชนโค้ดที่เคยเป็นแบบขั้นบันไดตรงตำแหน่งนั้นก็อาจจะ

กลายมาเป็นเชนโค้ดที่อยู่ในทิศทางที่อยู่กึ่งกลางระหว่างทิศของเชนโค้ดเก่าทั้ง 2 ตัว ซึ่งจะกลายมาเป็นเชนโค้ดใหม่ที่มีเพียงทิศทางเดียว และเมื่อทำการ reduce เชนโค้ดแล้ว ก็จะกลายมาเป็นเชนโค้ดเพียงตัวเดียว ซึ่งเมื่อนำมาสร้างเป็นโหนดของ DFA ก็จะสร้างเพียงแค่โหนดเดียวเท่านั้น แทนการสร้าง loop pair ที่ต้องมี 2 โหนด

แต่อย่างไรก็ตามไม่ว่าเราจะแบ่งทิศทางให้ละเอียดมากขึ้นเพียงใด การเกิดเชนโค้ดแบบขั้นบันไดก็ยังจะสามารถเกิดขึ้นได้อยู่ดี ซึ่งถ้าเชนโค้ดที่ออกมาใหม่ยังมีลักษณะที่เป็นขั้นบันได หลังจากการแบ่งเชนโค้ดเป็น 16 ทิศแล้ว เราก็จะทำการสร้างโหนดเรียงต่อเนื่องกันไปโดยจะไม่มีการสร้าง loop pair ขึ้นมาสำหรับการสร้าง DFA จากเชนโค้ด 16 ทิศ แต่สำหรับกระบวนการสร้าง DFA จากเชนโค้ด 8 ทิศ เรายังจะทำการสร้าง loop pair เช่นเดิมเหมือนกับในงานวิจัยของ ปองเกษม พลสันติกุล [1] เพื่อจะได้วัดประสิทธิภาพจากกระบวนการทำงานที่ต่างกัน ซึ่งกระบวนการทั้งหมดที่ได้กล่าวมาข้างต้นได้แสดงไว้ในรูป 3.8

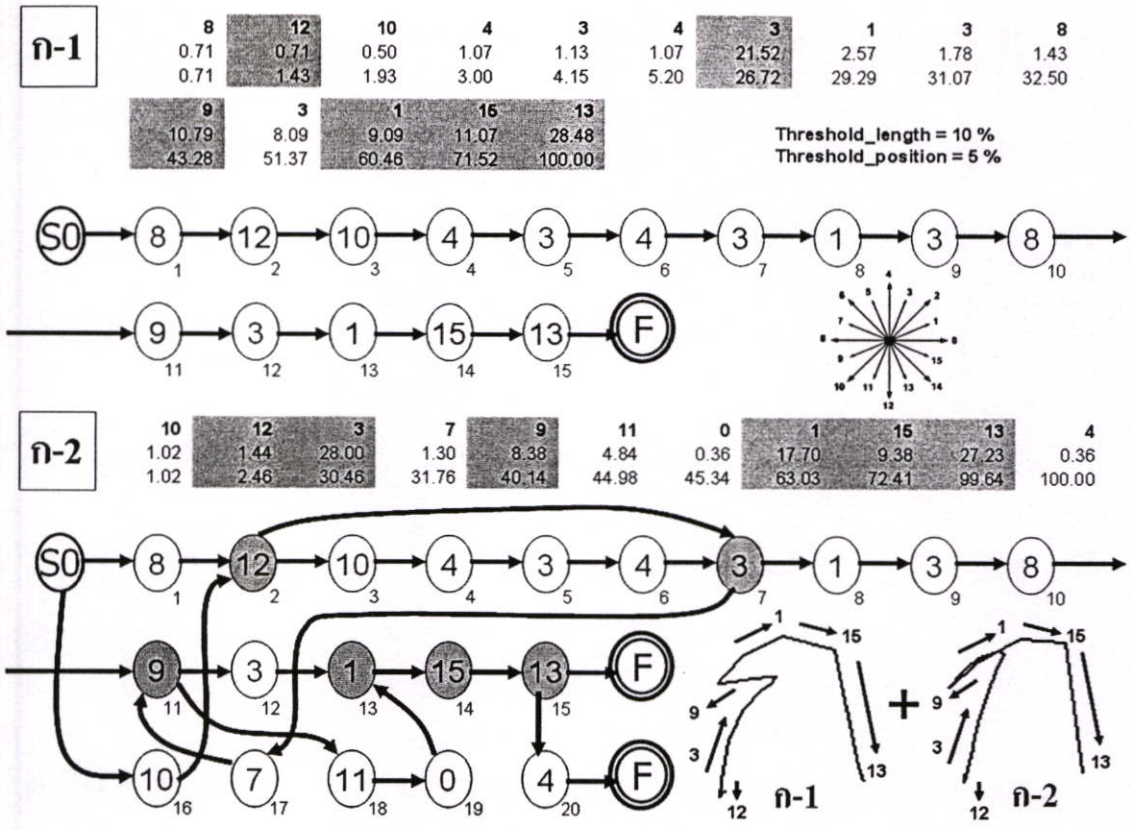


รูปที่ 3.8 แสดงการสร้าง loop pair ระหว่างโหนดที่มีค่าทิศทางเป็น 3 และ 2 (ทิศ 1-8) และการที่ไม่นำกระบวนการนี้มาใช้ เนื่องจากการแบ่งทิศทางที่ละเอียดขึ้น ซึ่งอาจกลายมาเป็นทิศทางใหม่ที่เป็นทิศ 3 (ทิศ 0-15) เพียงทิศเดียวเท่านั้น

3.7 การสร้างโมเดลการรู้จำ DFA จากเชนโค้ด 16 ทิศ พร้อมทั้งค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัว

การสร้างโมเดลการรู้จำ DFA จากเชนโค้ด 16 ทิศ ซึ่งเป็นเป้าหมายหลักของงานวิจัยนี้ ก็คล้ายคลึงกับการสร้างโมเดลการรู้จำของเชนโค้ดแบบ 8 ทิศ เพียงแต่ว่าเรามีข้อมูลของค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาด้วย ซึ่งจะทำให้เราสร้างโมเดลการรู้จำได้มีประสิทธิภาพมากขึ้น โดยนอกเหนือจากที่เราจะวิเคราะห์จากค่าทิศทางที่ละเอียดมากขึ้นแล้ว ยังมีค่าของขนาดและตำแหน่งมาช่วยวิเคราะห์ในขั้นตอนการสร้างโมเดลการรู้จำของ DFA เพิ่มขึ้นด้วย

ขั้นตอนการสร้างโมเดล เริ่มแรกนำค่าของเชนโค้ดของตัวอักษรตัวแรกมาสร้างเป็นโมเดลตั้งต้นก่อน โดยสร้างเรียงกันไปตามลำดับตั้งแต่ตัวแรกจนถึงตัวสุดท้าย โดยไม่ต้องพิจารณาถึงค่าของขนาดและตำแหน่ง หลังจากสร้างโมเดลตั้งต้นเสร็จ ก็นำเอาเชนโค้ดของตัวอักษรตัวต่อๆมาที่จะสร้างรวมกัน มาสร้างรวมกันไปทีละสายจนครบจำนวนตัวอักษรที่ต้องการใช้สอนโมเดลให้เกิดการรู้จำ แต่การสร้างรวมกันของโมเดลจะมีการพิจารณาถึงค่าของขนาดและตำแหน่งของเชนโค้ดแต่ละตัวร่วมด้วย โดยมีหลักการที่ว่า เชนโค้ดที่จะใช้โหนดร่วมกันจะต้องมีค่าทิศทางที่เหมือนกันและค่าขนาดและตำแหน่งของเชนโค้ดตัวนั้นๆจะต้องมีค่าที่ใกล้เคียงกัน และจะไม่มีการสร้าง loop pair ขึ้นมาในการสร้างโมเดลนี้ ซึ่งได้ยกตัวอย่างการสร้างรวมโมเดล DFA ของเชนโค้ดแบบ 16 ทิศ ตามรูปที่ 3.9



รูปที่ 3.9 ตัวอย่างการสร้างรวมโมเดล DFA ของ ก-1 และ ก-2 ของเชนโค้ดแบบ 16 ทิศ พร้อมค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัว

จากในรูป 3.9 จะเห็นว่าโหนดตัวที่มีการไฮไลท์สีเข้ม และข้อมูลของทิศทางขนาดและตำแหน่งตรงที่มีการไฮไลท์สีเข้ม จะหมายถึงโหนดนั้นๆมีการใช้โหนดร่วมกันของทั้ง 2 ตัวอักษร ซึ่งได้แก่โหนดหมายเลข 2, 7, 11, 13, 14 และ 15 โดยได้กำหนดเงื่อนไขไว้ว่าจะใช้โหนดร่วมกันได้จะต้องมีค่าทิศทางที่เหมือนกัน และมีการกำหนดค่าของ Threshold_length = 10% และ Threshold_position = 5% ซึ่งหมายถึงนอกจากจะมีค่าทิศทางที่เหมือนกันแล้วจะต้องมีค่าของขนาดและตำแหน่งที่จะต้องแตกต่างกันไม่เกิน 10% และ 5% ตามลำดับด้วย ถึงจะมีการใช้โหนดร่วมกันได้ ถ้าไม่เป็นไปตามเงื่อนไขข้างต้นที่ได้กำหนดไว้ ก็จะไม่มีการใช้โหนดร่วมกัน แต่จะมีการสร้างโหนดใหม่ขึ้นมาแทนเพื่อจดจำลักษณะที่แตกต่างกันออกไป

สำหรับโหนดที่มีการใช้โหนดร่วมกันแล้ว จะต้องมีการปรับค่า length และ position ของโหนดนั้นทุกๆครั้งด้วย โดยการหาค่าเฉลี่ยตามสมการที่ (3.2) และ (3.3)

$$AvgLength = \frac{\sum_{n} Length_n}{n} \quad (3.2)$$

$$AvgPosition = \frac{\sum_{n} Position_n}{n} \quad (3.3)$$

เมื่อ n คือจำนวนครั้งที่มีการใช้โหนดร่วมกัน

$Length_n$ และ $Position_n$ คือค่าของ length และ position ของแต่ละโหนดที่ n ที่มีการใช้โหนดร่วมกัน

$AvgLength$ และ $AvgPosition$ คือค่า length และ position ค่าใหม่ ที่ได้จากการหาค่าเฉลี่ยหลังจากมีการใช้โหนดร่วมกันแล้ว

3.8 โมเดล DFA 16 ทิศ ของตัวอักษรแต่ละตัวที่ได้จากการสอน

หัวข้อนี้จะแสดงผลจากการสร้างโมเดลการรู้จำของ DFA 16 ทิศ ที่ได้จากการสอนโดยจะยกตัวอย่างโมเดลของตัวอักษร “ก” , “ข” และ “ค” ซึ่งจะแสดงอยู่ในรูปของกราฟที่แสดงอยู่ในรูปที่ 3.10 ถึง 3.15

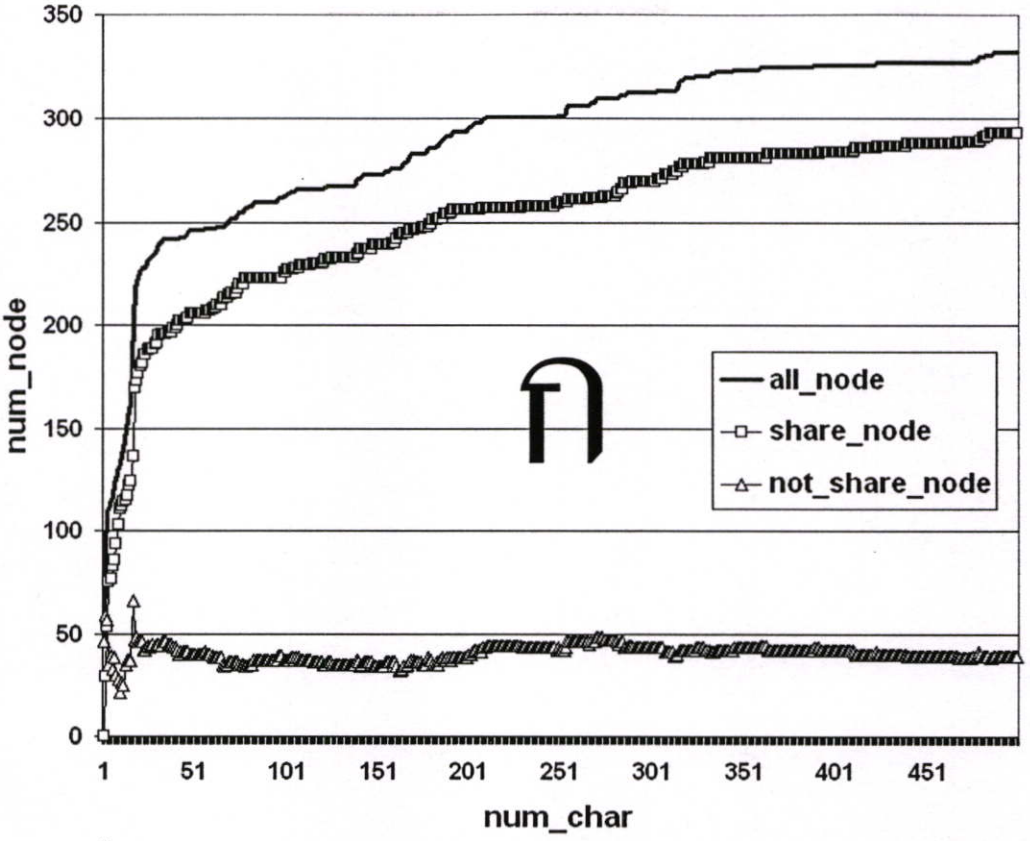
ในรูปที่ 3.10, 3.12 และ 3.14 จะเป็นกราฟที่แสดงจำนวนโหนด (num_node) ของโหนดที่เกิดขึ้นทั้งหมด (all_node), จำนวนโหนดที่มีการใช้งานร่วมกัน (share_node) และโหนดที่ไม่ได้ใช้งานร่วมกัน (not_share_node) ซึ่งจะเป็นกราฟแบบสะสมค่าตามจำนวนตัวอักษร (num_char) ที่นำมาใช้สอนโมเดล ซึ่งค่าของ share_node เมื่อรวมกับค่า not_share_node จะได้เป็นค่าของ all_node

ส่วนในรูปที่ 3.11, 3.13 และ 3.15 จะเป็นกราฟที่แสดงจำนวนโหนด (num_node) ของจำนวนโหนดที่ไม่เท่ากันของสาย string ของตัวอักษรแต่ละตัวที่นำมาใช้สอน (num_char) ซึ่งจะมีค่าได้ตั้งแต่ 10 กว่าโหนด จนถึง 100 กว่าโหนด ขึ้นอยู่กับลักษณะการเขียนของแต่ละบุคคล โดยเมื่อนำมาสร้างรวมเป็น DFA แล้วก็จะมีการสร้างโหนดใหม่ขึ้นมา ซึ่งโหนดที่ถูกสร้างขึ้นใหม่ในช่วงแรกๆ จะมีการสร้างโหนดใหม่ขึ้นมามาก แต่ต่อมาจะมีการสร้างโหนดใหม่ที่น้อยลง ซึ่งอาจสร้างโหนดใหม่เพียงแค่ 2-3 โหนด หรืออาจจะไม่มีการสร้างโหนดใหม่ขึ้นมาเลยก็ได้ เพราะโมเดลได้รู้จำรูปแบบจากตัวอักษรก่อนหน้าเรียบร้อยแล้ว

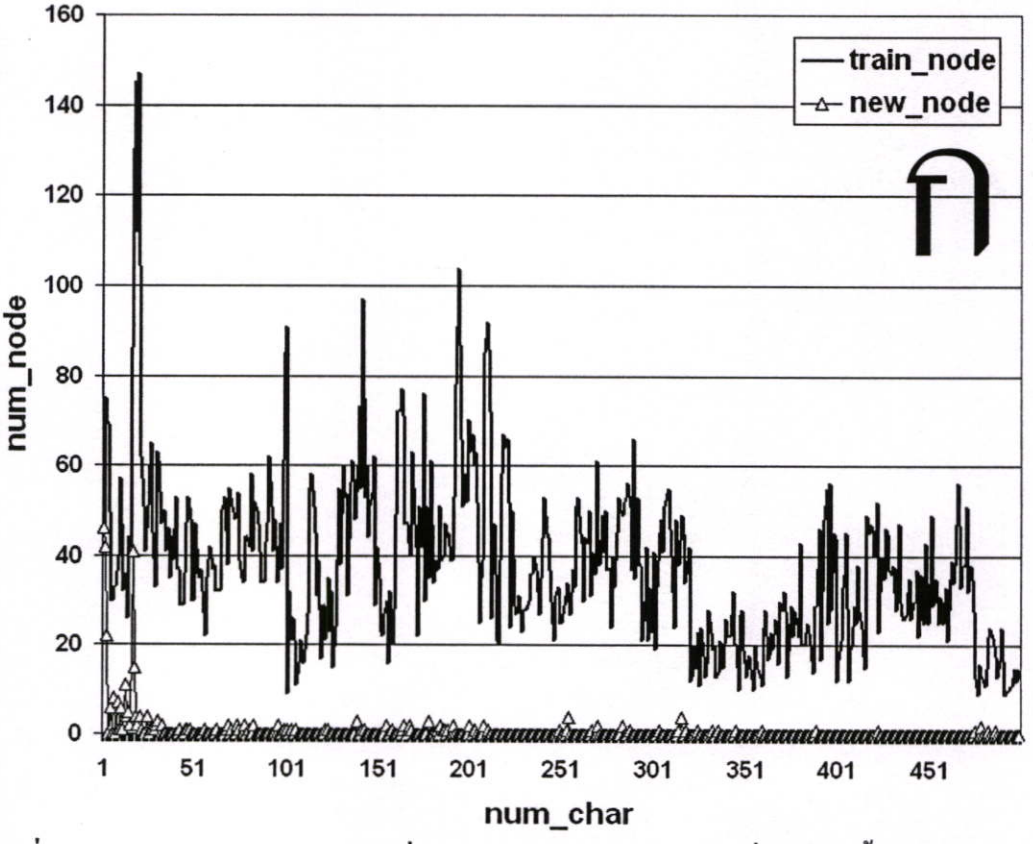
ซึ่งผลจากการสร้าง DFA ของตัวอักษรทั้ง 37 ตัวได้แสดงค่าต่างๆ ของ DFA ไว้ ดังแสดงอยู่ในตารางที่ 3.1 โดยค่า train_char หมายถึง จำนวนตัวอักษรที่นำมาใช้สอนโมเดลของตัวอักษรแต่ละตัว ค่า all_node คือ จำนวนโหนดที่เกิดขึ้นทั้งหมดของ DFA ของตัวอักษรนั้นๆ ค่า

share_node คือ จำนวนโหนดที่มีการใช้โหนดร่วมกัน not_share_node คือ จำนวนของโหนดที่ไม่ได้ใช้งานร่วมกัน percent_share_node กับ percent_not_share_node คือ เปอร์เซ็นต์ของโหนดที่มีการใช้งานร่วมกันและไม่ได้ใช้งานร่วมกันของ DFA ของตัวอักษรแต่ละตัว ส่วนค่า count_link_all คือ จำนวนของลิงค์เชื่อมโยงระหว่างโหนดแต่ละโหนดภายใน DFA นั้นๆ สุดท้ายค่า link_per_node คือ อัตราส่วนระหว่างจำนวนลิงค์ที่เชื่อมโยงหารตัวจำนวนโหนดทั้งหมดของ DFA ตัวนั้นๆ

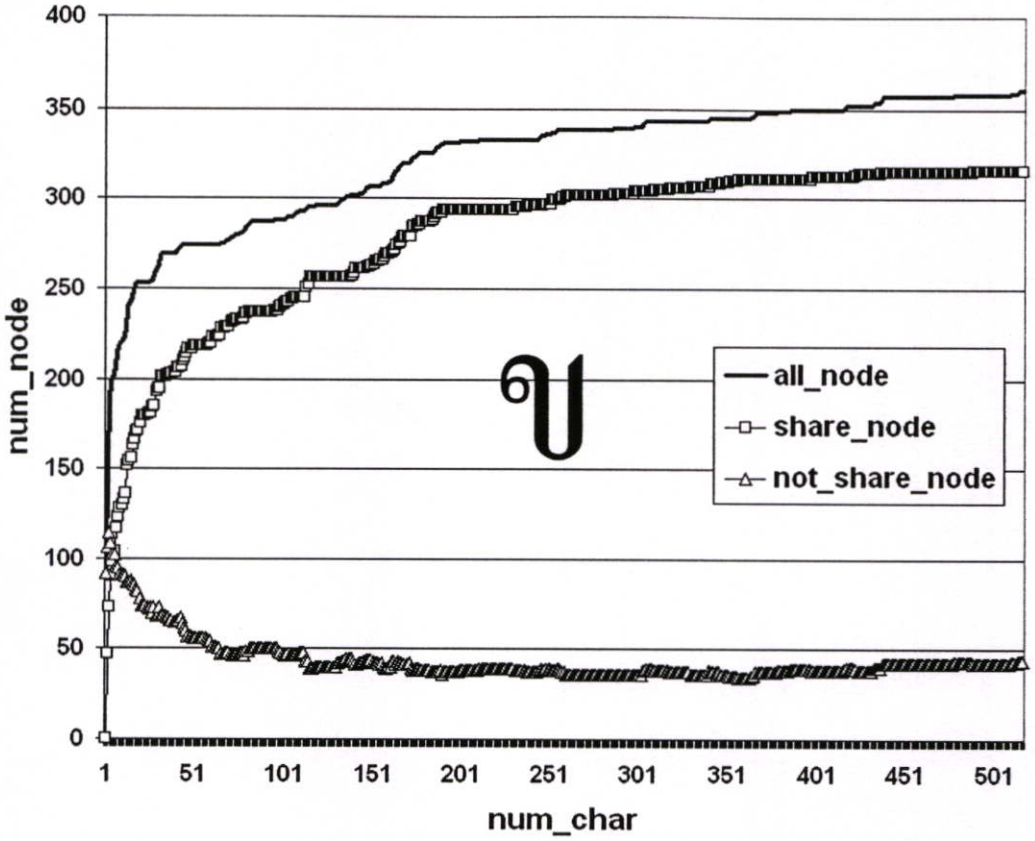
จากผลการสร้าง DFA ที่แสดงเป็นกราฟในรูปที่ 3.10 ถึง 3.15 และในตารางที่ 3.1 จะเห็นได้ว่าโมเดลของ DFA ของตัวอักษรทุกๆตัว จะมีค่าต่างๆที่ใกล้เคียงกัน โดยจะมีโหนดที่ใช้งานร่วมกันอยู่ในเปอร์เซ็นต์ที่สูง คือ ประมาณ 90% ซึ่งแสดงให้เห็นว่าโมเดล DFA 16 ทิศ ในงานวิจัยนี้มีประสิทธิภาพการทำงานที่ดี



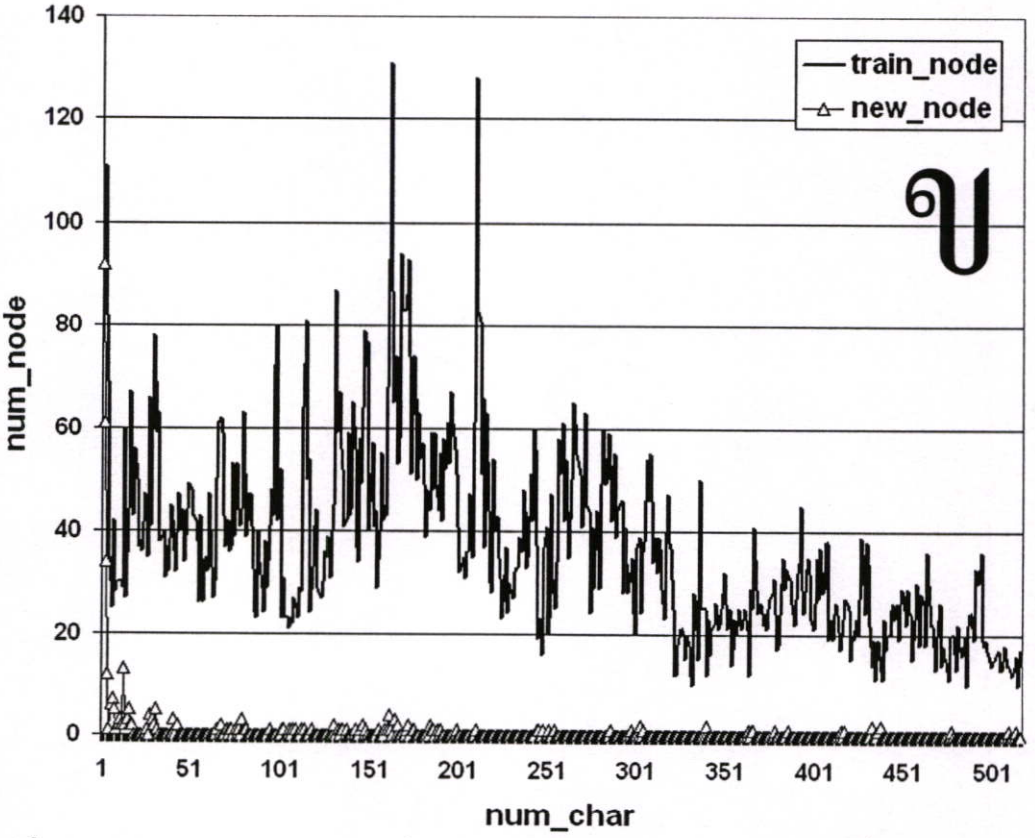
รูปที่ 3.10 กราฟแสดงจำนวนโหนดแต่ละแบบของ DFA ที่สร้างจากตัวอักษร “ก”



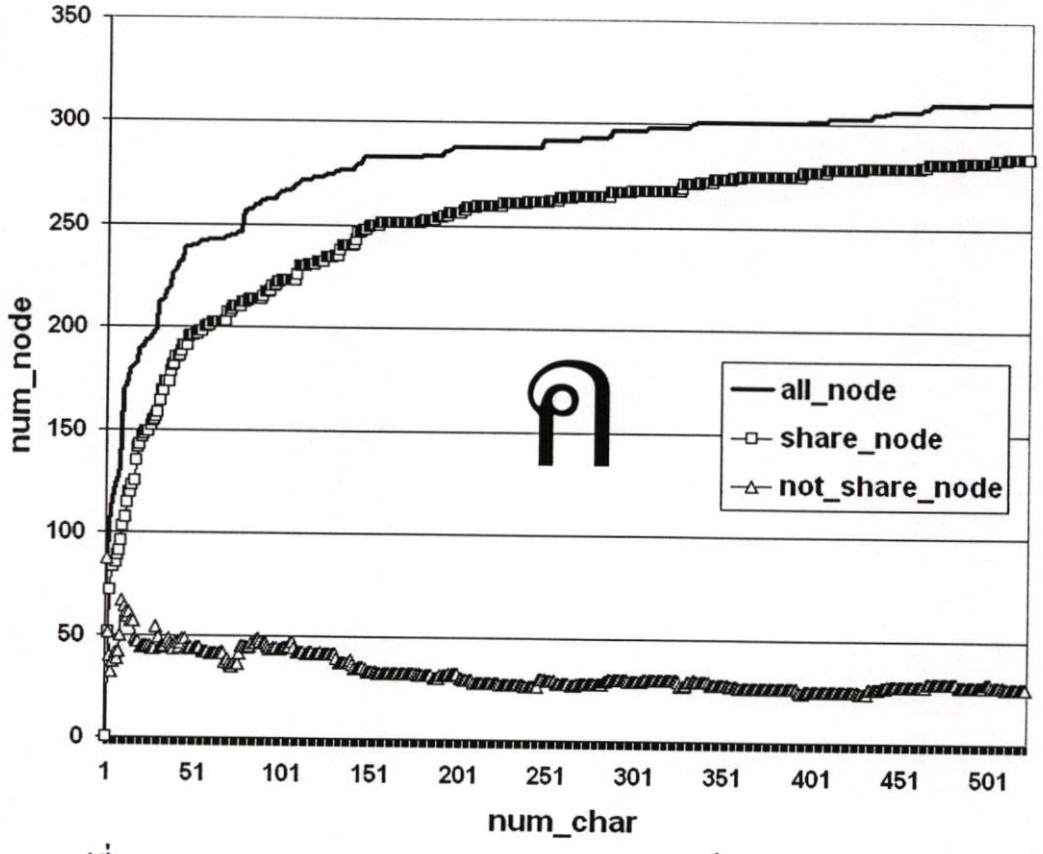
รูปที่ 3.11 กราฟแสดงจำนวนโหนดที่ใช้สอนเปรียบเทียบกับโหนดที่ถูกสร้างขึ้นใหม่ของ “ก”



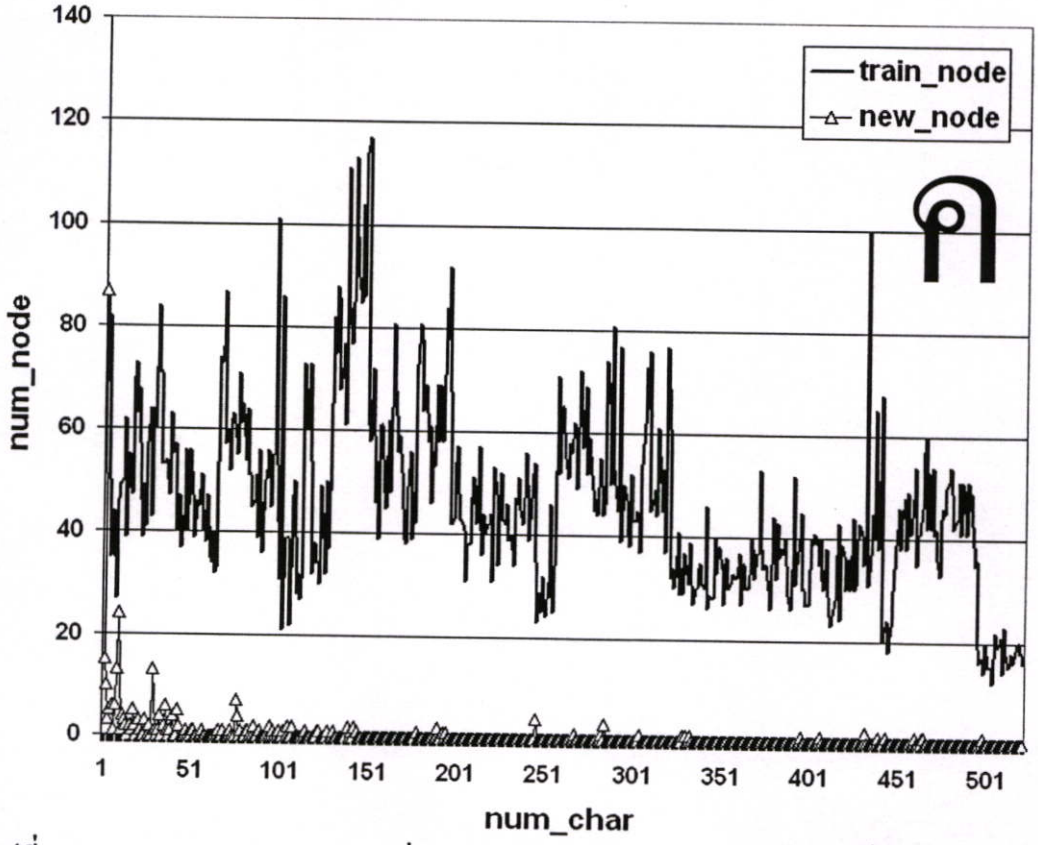
รูปที่ 3.12 กราฟแสดงจำนวนโหนดแต่ละแบบของ DFA ที่สร้างจากตัวอักษร "ข"



รูปที่ 3.13 กราฟแสดงจำนวนโหนดที่ใช้สอนเปรียบเทียบกับโหนดที่ถูกสร้างขึ้นใหม่ของ "ข"



รูปที่ 3.14 กราฟแสดงจำนวนโหนดแต่ละแบบของ DFA ที่สร้างจากตัวอักษร "ค"



รูปที่ 3.15 กราฟแสดงจำนวนโหนดที่ใช้สอนเปรียบเทียบกับโหนดที่ถูกสร้างขึ้นใหม่ของ "ค"

ตารางที่ 3.1 แสดงค่าต่างๆ ของ DFA ของตัวอักษรแต่ละตัว

ตัวอักษร	train_char	all_node	share_node	not_share_node	percent_share_node	percent_not_share_node	count_link_all	link_per_node
ก	500	332	293	39	88.25	11.75	1669	5.03
ข	517	360	313	47	87.78	12.22	1785	4.96
ค	521	311	284	27	91.32	8.68	1581	5.08
ม	507	371	346	25	93.26	6.74	2045	5.51
ง	540	383	338	45	88.25	11.75	1995	5.21
จ	536	359	330	29	91.29	8.71	1814	5.05
ฉ	521	344	320	24	93.02	6.98	2074	6.03
ช	509	358	335	23	93.58	6.42	2022	5.65
ซ	509	371	341	30	91.91	8.09	2196	5.92
ฌ	527	383	353	30	92.17	7.83	2247	5.87
ญ	505	377	352	25	93.37	6.63	2199	5.83
ฐ	489	392	367	25	93.62	6.38	2528	6.45
ฑ	462	383	360	23	93.99	6.01	2155	5.63
ฒ	503	366	347	19	94.81	5.19	2179	5.95
ณ	499	351	335	16	95.44	4.56	2130	6.07
ด	506	323	292	31	90.40	9.60	1729	5.35
ต	500	319	283	36	88.71	11.29	1778	5.57
ถ	488	301	265	36	88.04	11.96	1615	5.37
ท	526	341	313	28	91.79	8.21	1766	5.18
ธ	503	317	284	33	89.59	10.41	1746	5.51
น	481	312	293	19	93.91	6.09	1671	5.36
บ	491	307	271	36	88.27	11.73	1561	5.08
ป	488	285	268	17	94.04	5.96	1503	5.27
ผ	486	304	281	23	92.43	7.57	1612	5.30
ฝ	486	305	273	32	89.51	10.49	1646	5.40
พ	488	323	289	34	89.47	10.53	1811	5.61
ฟ	492	320	295	25	92.19	7.81	1780	5.56
ภ	498	295	266	29	90.17	9.83	1630	5.53
ม	497	318	298	20	93.71	6.29	1756	5.52
ย	476	307	286	21	93.16	6.84	1703	5.55
ร	478	298	277	21	92.95	7.05	1585	5.32
ล	480	314	283	31	90.13	9.87	1601	5.10
ว	473	290	255	35	87.93	12.07	1293	4.46
ห	497	334	303	31	90.72	9.28	1888	5.65
ฬ	500	337	308	29	91.39	8.61	1989	5.90
อ	491	310	289	21	93.23	6.77	1484	4.79
ฮ	479	278	252	26	90.65	9.35	1477	5.31

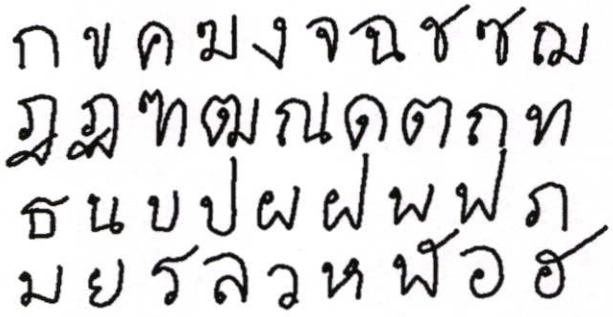
บทที่ 4

การทดลองและผลการทดลอง

การทดลองในงานวิจัยนี้ คือการนำข้อมูลตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์มาทำการสร้างการรู้จำด้วยโมเดล DFA แบบ 8 และ 16 ทิศแยกกันของตัวอักษรแต่ละตัว หลังจากนั้นก็จะทำการทดสอบการรู้จำโดยการนำเอาข้อมูลตัวอักษรส่วนที่ไม่เคยใช้สอนโมเดลรู้จำนั้นมาก่อนมาใช้ทดสอบ โดยในรายละเอียดต่างๆจะได้กล่าวถึงในหัวข้อย่อต่อไปในบทนี้

4.1 ข้อมูลที่ใช้เป็นอินพุทในการทดลอง

เป็นข้อมูลของตัวอักษรลายมือเขียนภาษาไทย ของตัวอักษรที่เขียนแบบครั้งเดียวจบโดยไม่ต้องยกมือมาเขียนเป็นครั้งที่สอง และเป็นตัวอักษรที่มีใช้อยู่กันในปัจจุบัน ตัวอักษรที่อยู่ในขอบเขตที่กำหนดนี้มีทั้งหมด 37 ตัวอักษร ดังที่จะแสดงไว้ในรูปที่ 4.1



รูปที่ 4.1 ตัวอักษรลายมือเขียนภาษาไทยแบบที่เขียนครั้งเดียวจบที่ใช้งานอยู่ในปัจจุบัน ทั้งหมด 37 ตัวอักษร

ในการทดลองผู้วิจัยได้นำข้อมูลของตัวอักษรเก่าที่มีผู้ทำวิจัยก่อนหน้าได้ทำเอาไว้ รวมกับข้อมูลตัวอักษรใหม่ที่ผู้วิจัยได้จัดทำขึ้นมาเอง โดยได้จากการเขียนจากบุคคล 20 คน โดยให้แต่ละคนเขียนตัวอักษรทั้ง 37 ตัว ตัวละ 20 ครั้ง เพราะฉะนั้นจะได้ข้อมูลการเขียนของตัวอักษรแต่ละตัวคือ 400 ครั้ง หลังจากนั้นก็นำข้อมูลของทั้งตัวอักษรเก่าและใหม่มาทำการแยกเป็น 2 ชุด แล้วนำมารวมกันอีกครั้งเพื่อใช้เป็นข้อมูลสำหรับชุดสอนและข้อมูลสำหรับชุดทดสอบต่อไป

จากข้อมูลตัวอักษรเก่าจะถูกนำมาแยกเป็นสัดส่วน 3:1 โดยการนับเรียงตามจำนวนตัวอักษร โดยนับตัวที่ 1, 2, 3 แล้วจัดเก็บเป็นข้อมูลที่ใช้เป็นชุดสอน ส่วนตัวที่ 4 จะถูกนำเอามาเก็บ

ไว้ในชุดทดสอบ จากนั้นก็นับตัวต่อไป ตัวที่ 5, 6, 7 นำมาเก็บไว้ในชุดสอนเพิ่มเติม ส่วนตัวที่ 8 ก็จะเก็บไว้ในชุดทดสอบเพิ่มเข้าไป ทำไปเช่นนี้เรื่อยๆจนกว่าจะครบตามจำนวนครั้งที่มีการเขียนตัวอักษรทั้งหมดที่มีในตัวอักษรแต่ละตัว ซึ่งจำวนวนของการเขียนสำหรับตัวอักษรแต่ละตัวจะไม่เท่ากันเพราะผู้วิจัยก่อนหน้าได้ทำการคัดแยกตัวอักษรบางตัวที่มีลักษณะไม่ดีออกไปแล้ว แต่ก็จะมีจำนวนการเขียนประมาณ 200 ครั้งสำหรับตัวอักษรแต่ละตัว

สำหรับข้อมูลตัวอักษรใหม่ก็นำมาแยกสัดส่วนเช่นเดียวกัน เพียงแต่สัดส่วนของชุดสอนกับชุดทดสอบจะเป็น 4:1 แทน คือ นับตัวที่ 1, 2, 3, 4 แล้วเก็บเป็นชุดสอน ส่วนตัวที่ 5 จะเป็นชุดทดสอบ ทำเช่นนี้ไปเรื่อยๆจนกว่าจะครบทั้ง 400 ตัวสำหรับจำนวนครั้งในการเขียนของตัวอักษรแต่ละตัว

จากนั้นก็นำข้อมูลที่แยกเป็น 2 ชุด ของทั้งตัวอักษรเก่าและใหม่มารวมกัน กลายเป็นมีข้อมูลสำหรับชุดสอนเพียงชุดเดียว และมีชุดสำหรับทดสอบเพียงชุดเดียวเท่านั้น ซึ่งจะมีสัดส่วนของชุดสอน (train) กับชุดทดสอบ (test) ประมาณ 4:1 เพราะข้อมูลตัวอักษรใหม่จะมีจำนวนมากกว่าของตัวอักษรเก่า ซึ่งจำนวนของตัวอักษรที่ใช้ train และ test ของตัวอักษรทุกตัวได้แสดงไว้ในตารางที่ 4.1

ตารางที่ 4.1 แสดงจำนวนตัวอักษรที่ใช้ train และ test ของตัวอักษรแต่ละตัว

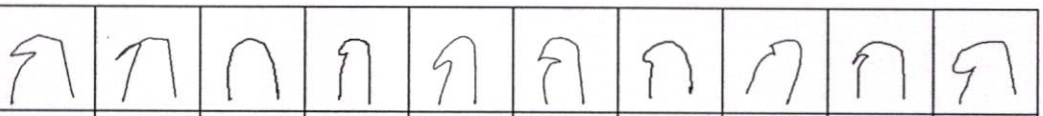
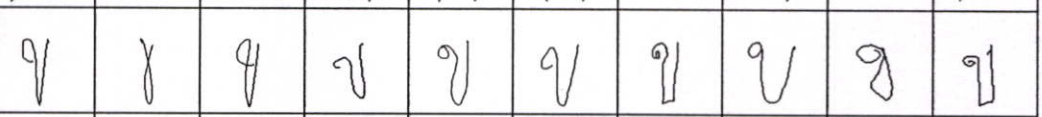
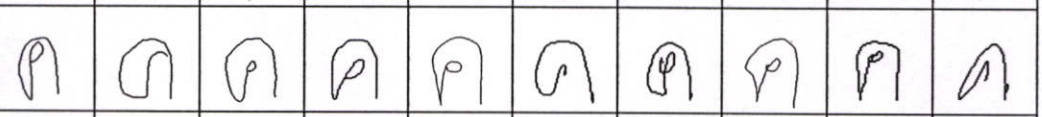
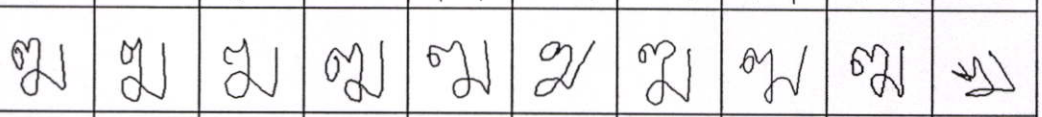
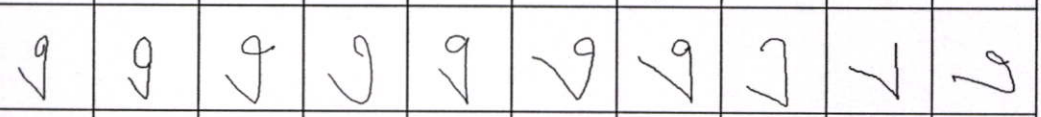
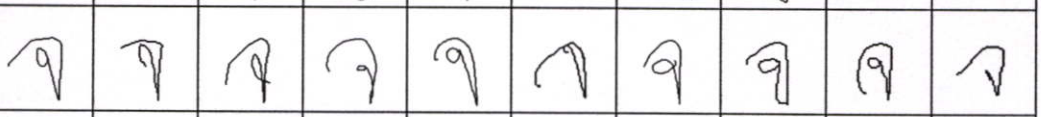
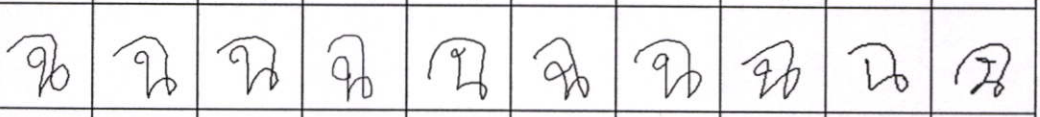
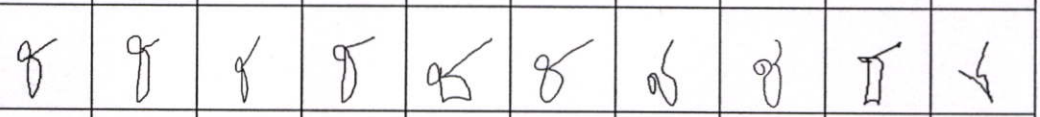
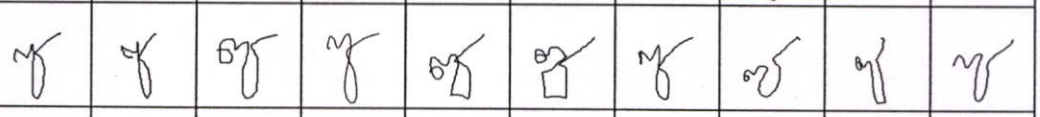
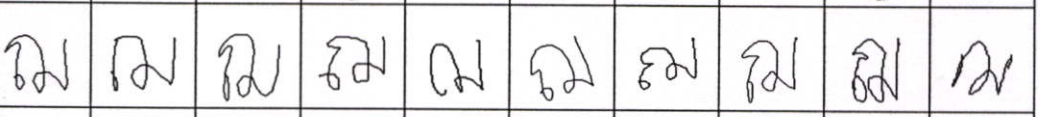
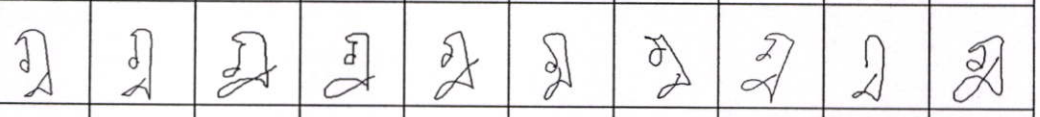
อักษร	train	test	อักษร	train	test	อักษร	train	test	อักษร	train	test
ก	500	139	ฎ	505	141	น	481	133	ร	478	132
ข	517	145	ฏ	489	136	บ	491	136	ล	480	133
ค	521	146	ฑ	462	127	ป	488	135	ว	473	130
ฆ	507	142	ฒ	503	140	ผ	486	135	ห	497	138
ง	540	153	ณ	499	139	ฝ	486	135	พ	500	139
จ	536	151	ด	506	142	พ	488	135	อ	491	136
ฉ	521	147	ต	500	139	ฟ	492	137	ฮ	479	133
ช	509	142	ถ	488	145	ภ	498	139			
ซ	509	143	ท	526	148	ม	497	139			
ฌ	527	149	ธ	503	141	ย	476	132			

หากนับรวมจำนวนของตัวอักษรทุกตัวของทั้ง 37 ตัวรวมกัน จะได้จำนวนของตัวอักษรที่ใช้ train และ test คือ

- จำนวนที่ใช้ train ของตัวอักษรทั้งหมดรวมกัน คือ 18,449 ตัวอักษร
- จำนวนที่ใช้ test ของตัวอักษรทั้งหมดรวมกัน คือ 5,152 ตัวอักษร

โดยตัวอย่างลักษณะลายมือเขียนของทั้งข้อมูลตัวอักษรเก่าและใหม่ ที่นำมาเป็นข้อมูลของทั้งชุดสอนและชุดทดสอบในงานวิจัยนี้ ได้แสดงไว้ในตารางที่ 4.2

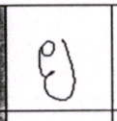
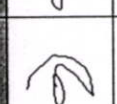
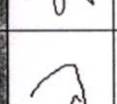
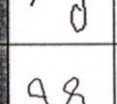
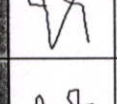
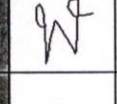
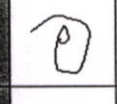
ตารางที่ 4.2 แสดงตัวอย่างลักษณะลายมือเขียนของตัวอักษรแต่ละตัวที่นำมาใช้ในงานวิจัยนี้

ก										
ข										
ค										
ฆ										
ง										
จ										
ฉ										
ช										
ซ										
ฌ										
ฎ										
ฏ										

ตารางที่ 4.2 (ต่อ)

ฑ										
ฒ										
ณ										
ด										
ต										
ถ										
ท										
ธ										
น										
บ										
ป										
ผ										
ฝ										
พ										
ฟ										
ภ										
ม										

ตารางที่ 4.2 (ต่อ)

ย										
ร										
ล										
ว										
ห										
พ										
อ										
ฮ										

4.2 การทดลอง

การทดลองเราจะทำการทดสอบการรู้จำโดยการนำข้อมูลตัวอักษรชุดทดสอบของตัวอักษรแต่ละตัวมาทดสอบกับโมเดลตัวอักษรของแต่ละโมเดล หากข้อมูลตัวอักษรที่นำมาทดสอบนั้นสามารถวิ่งได้ตาม link ที่เชื่อมต่อระหว่างแต่ละโหนด โดยเริ่มต้นจาก starting state (S0) และสามารถวิ่งไปได้จนจบ จนถึง final state (F) ของโมเดล DFA ของตัวอักษรนั้นๆได้ เราก็จะให้ตัวอักษรตัวนั้นผ่านการทดสอบการรู้จำของโมเดลตัวอักษรนั้นๆ

ข้อมูลที่นำมาทดสอบใช้เฉพาะที่เป็นค่าทิศทางของเชนโค้ดที่เป็นแบบ 8 และ 16 ทิศเท่านั้น จะไม่นำข้อมูลที่เป็นค่าขนาดและตำแหน่งของเชนโค้ดแบบ 16 ทิศมาทดสอบด้วย (ค่าขนาดและทิศทางใช้เฉพาะขั้นตอนการสร้างโมเดล DFA เท่านั้น)

ตัวอย่างการทดสอบการรู้จำได้แสดงไว้ในรูปที่ 4.2 และ 4.3 ในรูปที่ 4.2 คือการนำเอา unknown ก-30 มาทดสอบกับโมเดล DFA ก ซึ่งค่าเชนโค้ดของ ก-30 สามารถวิ่งได้ตาม link เชื่อมต่อของ state ทุกๆ state ของโมเดล DFA ก ตั้งแต่ starting state (S0) ไปจนถึง final state

(กำหนดค่า final state ในโปรแกรมเป็นค่า 9999) เพราะฉะนั้นผลการทดสอบคือ ก-30 สามารถผ่านการทดสอบจากโมเดล DFA ก ได้

ส่วนรูปที่ 4.3 คือการนำ unknown ข-50 มาทดสอบกับโมเดล DFA ก ซึ่งปรากฏว่าค่าเซนโค้ดของ ข-50 ไม่สามารถวิ่งไปตาม link ที่เชื่อมต่อของโมเดล DFA ก ได้ทุก state และไม่สามารถไปสิ้นสุดที่ final state ได้ เพราะฉะนั้น ข-50 ไม่สามารถผ่านการทดสอบจากโมเดล DFA ก ได้

```
Unknown ก-30
5          3          1          15         13         11         12         4          5
DFA ก
L[0][11] S0 -> V[265] = 5
L[265][3] 5 -> V[329] = 3
L[329][1] 3 -> V[330] = 1
L[330][2] 1 -> V[35] = 15
L[35][1] 15 -> V[36] = 13
L[36][3] 13 -> V[122] = 11
L[122][1] 11 -> V[39] = 12
L[39][8] 12 -> V[45] = 4
L[45][1] 4 -> V[46] = 5
+++++++
L[46][1] 5 =-> 9999          +++ Final State +++      Recog
+++++++
```

รูปที่ 4.2 แสดงการทดสอบ unknown ก-30 กับโมเดล DFA ก

```
Unknown ข-50
6        4        13       12       13       12       7        5        4        12
DFA ก
L[0][10] S0 -> V[263] = 6
L[263][1] 6 -> V[3] = 4
L[3][6] 4 -> V[48] = 12
L[48][9] 12 -> V[126] = 5
L[126][1] 5 -> V[5] = 4
+++++++
L[5][10] 4  => V[94] = 9          Non_Recog
+++++++
```

รูปที่ 4.3 แสดงการทดสอบ unknown ข-50 กับโมเดล DFA ก

4.3 ผลการทดลอง

4.3.1 ผลการทดลองของ DFA ทั้งแบบ 8 และ 16 ทิศ

ผลการทดลองจากการวัดประสิทธิภาพการรู้จำของโมเดล DFA จากเชนโค้ด 8 ทิศ ได้แสดงไว้ในตารางที่ 4.3 และผลการทดลองของโมเดล DFA จากเชนโค้ด 16 ทิศพร้อมทั้งข้อมูลขนาดและตำแหน่ง ได้แสดงไว้ในตาราง 4.4

โดยผลการทดลองจะแสดงจำนวนของตัวอักษรแต่ละตัวที่นำมาทดสอบกับโมเดลของตัวอักษรแต่ละตัว ว่าตัวอักษรแต่ละตัวเมื่อทำการทดสอบการรู้จำแล้วจะมีจำนวนตัวอักษรกี่ตัวที่สามารถผ่านการทดสอบการรู้จำของแต่ละโมเดลได้

ในผลการทดสอบการรู้จำของตารางที่ 4.3 และ 4.4 เป็นการทดสอบว่าเมื่อนำตัวอักษรชุดทดสอบของตัวอักษรแต่ละตัวมาทดสอบกับทุกๆโมเดลตัวอักษร แล้วจะมีตัวอักษรนั้นๆจำนวนกี่ตัวที่ผ่านการทดสอบของแต่ละโมเดล ซึ่งอาจมีตัวอักษรบางตัวผ่านการทดสอบจากหลายๆโมเดลได้ อันเนื่องมาจากโมเดลตัวอักษรนั้นมีความ generalize มากเกินไป (over-generalize) ซึ่งเป็นผลให้เกิดการรู้จำที่ผิดพลาด (accept false) ยกตัวอย่างเช่น ตัวอักษรที่นำมาทดสอบของตัว “ก” อาจจะผ่านการทดสอบการรู้จำจากทั้งโมเดลของ “ก” , “ถ” และ “ภ” ได้ เนื่องมาจากลักษณะของตัวอักษรที่มีคล้ายคลึงกัน

ตารางที่ 4.3 ผลการรู้จำจากการสร้าง DFA จากเชนโค้ดแบบ 8 ทิศ

โนดจาก DFA ของตัวอักษรที่สร้างจากเชนโค้ดแบบ 8 ทิศ	ชุดข้อมูลของตัวอักษรที่ใช้ทดสอบ																																					
จำนวน		139	145	146	142	153	151	147	142	143	149	141	136	127	140	139	142	139	135	148	141	133	136	135	135	135	135	137	139	139	132	132	133	130	138	139	136	133
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฎ	ฏ	ฑ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ฉ	ว	ห	พ	อ	ฮ
โนดจาก DFA ของตัวอักษรที่สร้างจากเชนโค้ดแบบ 8 ทิศ	ก	125	66	103	55	67	23	28	22	34	47	16	24	38	68	73	99	73	89	83	8	69	50	58	17	25	61	62	57	52	30	38	42	11	39	31	35	27
	ข	63	132	57	69	44	98	76	107	69	75	19	27	23	67	65	26	12	38	103	49	78	99	89	39	44	93	92	36	81	35	56	50	42	67	68	34	35
	ค	46	49	134	31	48	56	85	54	3	43	22	35	13	24	38	51	44	58	71	3	37	16	28	10	11	42	36	55	35	3	8	34	2	18	19	12	15
	ฆ	63	34	23	129	9	11	11	8	9	14	38	28	101	33	2	37	47	43	42	4	10	19	33	36	27	25	43	44	22	24	2	26	19	14	15	18	16
	ง	12	16	3	42	141	84	91	65	27	38	59	41	5	59	76	8	13	15	0	30	81	22	18	17	44	2	0	18	36	44	25	62	25	55	48	32	22
	จ	11	12	38	18	2	93	86	62	59	54	52	36	7	4	47	32	21	54	63	27	5	8	6	18	25	14	4	60	3	32	37	83	60	65	59	75	17
	ฉ	67	14	52	13	3	63	130	53	52	23	58	44	20	13	34	36	26	48	58	38	32	21	31	28	16	18	20	47	16	25	29	74	85	51	75	80	13
	ช	58	43	76	35	61	72	83	130	110	28	47	34	43	38	81	48	14	61	89	55	89	50	35	44	41	54	47	71	35	53	28	73	60	59	58	70	45
	ซ	29	107	27	58	43	71	80	121	136	27	46	23	74	27	34	36	28	13	14	62	76	32	30	53	29	38	55	42	36	52	39	68	25	29	44	51	34
	ฌ	37	69	20	51	42	66	88	63	52	106	49	34	11	79	82	22	49	34	72	50	92	61	55	84	79	60	53	40	62	71	15	72	94	27	23	83	21
	ญ	48	98	66	67	49	24	34	78	63	66	138	111	9	73	35	79	73	44	98	84	39	60	63	58	58	72	68	52	66	69	72	20	46	49	50	18	66
	ฏ	35	21	14	5	53	3	8	14	16	4	13	109	8	6	12	17	18	12	46	33	18	33	19	13	10	20	20	9	10	16	16	4	0	25	26	3	27
	ฑ	23	57	69	102	7	5	12	11	52	10	8	13	108	22	90	35	57	32	99	21	12	25	23	12	13	15	19	22	43	21	14	18	7	9	32	6	3
	ฒ	64	74	71	98	10	84	29	49	61	64	28	32	23	113	91	74	26	58	75	38	88	86	95	97	98	95	95	56	79	98	63	88	80	40	66	86	60
	ณ	67	81	42	17	46	27	22	45	27	25	84	26	18	37	121	43	79	59	60	35	91	66	63	46	45	29	79	72	18	55	36	19	5	56	18	19	43
	ด	24	44	24	62	82	19	49	5	20	38	16	15	6	50	53	122	112	27	73	17	63	24	45	15	11	67	44	15	66	7	9	36	0	26	22	10	17
	ต	48	39	39	33	68	8	38	11	9	46	7	19	32	31	32	104	120	94	19	6	32	35	38	25	22	30	43	38	48	11	14	3	5	25	10	9	8
	ถ	79	77	63	43	43	16	20	58	48	65	48	9	4	54	70	69	76	129	96	32	53	72	66	72	65	68	68	80	53	47	31	19	9	55	27	22	33
	ท	35	84	15	21	22	3	57	0	14	26	8	17	97	4	21	5	6	15	142	0	61	45	32	26	29	50	41	21	27	15	7	6	5	3	8	10	1

ตารางที่ 4.3 (ต่อ)

โมเดล DFA ของตัวอักษร ที่สร้างจากเซนโค้ดแบบ 8 ทิศ	ชุดข้อมูลของตัวอักษรที่ใช้ทดสอบ																																					
จำนวน		139	145	146	142	153	151	147	142	143	149	141	136	127	140	139	142	139	135	148	141	133	136	135	135	135	135	137	139	139	132	132	133	130	138	139	136	133
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฎ	ฏ	ฑ	ฒ	ฌ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ฉ	ว	ห	พ	อ	ฮ
	ก	12	0	25	10	2	9	9	67	77	28	85	55	0	33	32	35	11	12	0	115	0	1	0	29	35	0	0	66	0	36	75	4	7	50	44	26	84
	ข	3	58	4	39	20	24	103	88	39	24	44	34	27	67	76	18	32	22	30	33	126	48	30	36	44	29	48	28	8	27	44	19	36	18	48	28	39
	ค	2	32	19	80	7	6	3	14	22	72	7	18	15	58	48	3	37	18	11	34	24	116	99	39	41	54	47	24	77	44	21	25	13	45	35	15	7
	ฆ	27	47	41	52	24	18	27	57	61	41	26	37	13	56	63	75	75	67	92	36	25	108	127	37	63	59	66	58	86	63	27	36	25	91	60	53	53
	ง	4	39	0	21	21	33	22	17	29	78	8	19	35	81	41	16	36	9	43	32	35	42	33	126	100	47	45	27	54	77	6	58	7	20	36	64	68
	จ	1	0	2	3	0	0	6	1	7	27	5	24	33	17	16	6	9	0	3	15	2	0	4	102	120	6	7	17	9	75	0	4	0	3	2	52	66
	ฉ	11	99	18	75	41	67	19	46	35	84	7	13	28	98	48	13	34	12	32	23	30	47	35	38	55	129	119	34	96	32	40	40	6	13	43	18	12
	ช	10	64	9	93	27	15	84	21	89	98	40	18	31	87	95	11	29	21	31	46	89	38	31	60	72	120	122	29	72	58	48	25	7	16	87	32	42
	ซ	23	7	68	8	8	2	18	3	2	22	79	71	6	22	45	27	32	26	7	10	11	0	3	0	2	5	3	114	4	7	0	3	0	6	1	7	13
	ฌ	4	18	1	105	7	17	28	12	53	112	2	14	22	78	43	14	20	11	28	18	14	2	1	18	18	84	79	20	130	21	7	45	2	15	83	20	19
	ญ	3	12	16	33	7	4	1	23	36	43	15	18	38	49	39	8	12	5	9	16	13	39	27	53	60	18	20	45	26	88	38	0	23	24	24	13	9
	ฐ	2	41	21	64	31	37	43	64	54	5	53	69	7	34	8	26	18	8	18	83	22	3	13	15	25	22	9	57	6	23	116	8	0	7	8	79	57
	ฑ	10	0	5	0	3	45	59	35	38	4	42	41	39	0	15	0	0	18	0	6	3	0	0	19	24	0	0	19	0	24	8	110	64	21	61	93	58
	ฒ	1	4	7	12	16	26	71	31	49	19	38	33	34	3	2	4	3	1	1	69	8	2	4	6	11	0	0	60	3	27	93	53	124	4	17	94	65
	ห	35	11	26	42	3	29	25	16	22	69	33	0	48	11	58	3	2	55	0	10	7	20	19	61	61	15	27	61	4	36	17	19	12	130	95	24	76
	ฬ	12	7	15	8	7	11	42	11	27	18	57	38	27	3	22	5	4	17	0	11	9	33	37	18	15	2	5	35	0	25	25	28	34	18	131	25	44
	อ	7	3	0	3	2	29	71	30	28	0	41	12	11	2	2	0	0	18	0	7	1	0	2	0	0	0	0	46	1	4	44	49	102	7	14	116	46
	ฮ	1	0	0	2	0	4	3	1	4	24	32	40	4	0	21	0	0	5	0	14	0	0	0	9	8	0	0	28	0	12	36	0	0	9	42	4	119

ตารางที่ 4.4 ผลการรู้จำจากการสร้าง DFA จากเชนโค้ดแบบ 16 ทิศ พร้อมค่าขนาดและตำแหน่ง

โมเดล DFA ของตัวอักษร ที่สร้างจากเชนโค้ดแบบ 16 ทิศ พร้อมค่าขนาดและตำแหน่ง		ชุดข้อมูลของตัวอักษรที่ใช้ทดสอบ																																			
จำนวน		139	145	146	142	153	151	147	142	143	149	141	136	127	140	139	142	139	135	148	141	133	136	135	135	135	135	137	139	139	132	132	133	130	138	139	136	133
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ญ	ฐ	ฑ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ฤ	ล	ห	พ	อ	ฮ
	ก	103	27	21	1	12	10	5	12	3	3	1	2	3	0	5	22	2	18	20	1	8	18	8	1	1	8	22	7	10	0	2	11	0	5	3	0	0
	ข	18	107	7	27	14	36	12	29	8	12	2	0	18	8	7	24	3	6	35	12	22	53	65	7	6	48	31	5	42	8	8	25	28	11	3	6	4
	ค	26	6	117	1	13	3	1	4	1	0	3	1	0	0	0	10	2	6	7	0	3	3	2	4	3	2	1	15	3	1	2	2	0	2	0	1	1
	ฆ	14	27	11	101	14	11	7	15	14	29	1	1	17	36	10	24	24	12	30	14	33	23	27	5	7	27	23	7	79	3	3	17	6	6	7	4	4
	ง	11	20	12	18	111	22	15	10	1	14	8	7	2	12	9	16	5	8	13	11	17	12	12	7	4	7	4	12	29	2	12	25	14	6	7	8	3
	จ	9	7	0	0	5	116	33	6	1	1	8	1	0	0	1	2	1	2	0	8	1	5	6	5	2	2	1	1	4	5	16	67	79	23	18	41	30
	ฉ	17	9	11	4	14	81	116	6	5	6	20	6	6	3	3	8	2	7	3	7	5	5	6	4	4	2	4	20	4	3	18	63	65	33	28	52	25
	ช	7	38	6	18	12	7	3	102	47	9	5	0	7	11	2	10	2	7	22	34	13	27	43	8	7	18	20	8	27	3	19	6	6	2	6	0	4
	ซ	24	43	7	13	17	12	6	82	111	12	6	5	21	12	5	11	2	10	21	26	10	36	47	22	12	32	39	16	19	4	14	10	10	6	17	3	4
	ฌ	22	26	16	41	25	19	7	11	12	114	6	3	15	42	20	33	11	31	23	6	28	15	25	11	6	22	23	10	57	5	12	15	10	8	7	12	7
	ญ	3	16	5	5	7	13	4	11	10	5	111	20	1	4	6	4	1	5	4	15	5	18	7	14	8	10	10	5	1	10	26	8	15	4	6	11	23
	ฐ	31	27	29	11	13	17	19	14	5	7	75	109	13	8	14	24	5	14	34	16	22	16	20	13	17	20	11	34	13	13	17	13	14	15	14	14	22
	ฑ	33	37	23	21	18	14	10	6	11	7	2	1	85	17	13	39	24	19	65	3	13	26	24	7	9	30	24	15	26	3	0	23	16	16	6	1	1
	ฒ	18	35	19	61	25	11	8	18	13	41	2	2	25	96	14	33	24	20	28	4	24	32	31	9	11	30	38	12	64	3	8	14	5	8	6	5	2
	ณ	14	20	10	19	41	3	13	5	5	9	3	1	9	6	99	37	6	14	21	5	62	18	14	8	9	16	19	8	17	4	4	12	10	9	7	11	3
	ด	38	24	53	12	23	9	7	5	3	17	0	1	23	9	38	91	29	40	70	4	39	25	19	3	4	25	28	10	29	0	5	5	2	5	1	6	0
	ต	54	39	43	33	25	24	23	18	21	21	8	5	70	30	12	76	105	26	85	18	24	29	35	14	9	39	26	28	40	4	9	28	15	15	6	12	2
	ถ	55	23	26	16	8	24	15	13	9	20	3	4	30	10	10	49	21	102	50	12	4	21	10	0	0	18	11	13	22	0	3	32	4	17	8	2	1
	ท	20	22	12	11	3	1	30	10	18	7	1	0	29	10	9	44	13	8	126	0	29	25	27	1	2	37	36	7	34	0	1	9	0	4	3	0	1

ตารางที่ 4.4 (ต่อ)

โมเดล DFA ของตัวอักษร ที่สร้างจากเซนโด้ดแบบ 16 ทิศ พร้อมค่าขนาดและตำแหน่ง		ชุดข้อมูลของตัวอักษรที่ใช้ทดสอบ																																				
	จำนวน	139	145	146	142	153	151	147	142	143	149	141	136	127	140	139	142	139	135	148	141	133	136	135	135	135	135	137	139	139	132	132	133	130	138	139	136	133
		ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ฎ	ฏ	ฑ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม	ย	ร	ฉ	ว	ห	พ	อ	ฮ
	ธ	13	8	11	2	3	5	2	13	6	1	7	3	3	1	1	1	1	2	2	115	4	5	5	10	5	4	3	22	0	0	58	1	8	2	2	4	8
	น	2	5	5	3	20	1	19	3	6	2	0	0	5	10	0	15	2	4	17	2	79	2	6	1	1	5	4	0	8	0	0	6	0	1	1	0	0
	บ	7	66	2	14	6	22	4	34	14	4	1	0	9	11	1	10	5	0	9	8	9	102	80	11	4	44	32	9	29	2	15	13	23	5	5	6	0
	ป	10	65	1	17	11	19	5	16	7	9	0	2	5	12	0	5	1	5	7	8	8	71	97	10	9	27	38	8	44	5	9	16	24	8	7	8	7
	ผ	6	15	6	0	3	12	0	3	5	0	2	1	4	0	1	1	0	1	6	8	1	9	20	106	78	13	17	15	1	47	22	1	33	11	9	43	32
	ฝ	3	13	3	1	5	6	0	7	2	1	2	0	1	0	6	1	0	0	8	7	1	9	18	70	107	9	13	8	1	21	12	1	14	6	8	24	15
	พ	19	51	10	11	6	26	14	31	28	11	2	3	19	17	8	18	12	4	50	17	10	53	56	19	19	104	91	11	38	5	11	23	29	14	20	7	4
	ฟ	11	49	13	11	11	19	6	25	23	9	2	1	21	13	3	13	7	10	30	11	14	49	67	31	27	67	107	12	41	8	12	17	8	14	20	6	4
	ภ	68	13	55	3	7	8	8	10	7	3	9	4	8	0	5	13	1	14	25	12	6	3	4	18	9	13	11	112	8	4	19	7	4	8	1	4	2
	ม	13	49	16	36	20	32	8	14	20	23	3	3	22	29	0	33	15	12	35	10	13	20	26	8	15	38	32	8	113	3	9	42	15	10	9	10	1
	ย	3	12	7	0	5	2	0	2	4	2	2	3	1	0	5	0	0	0	3	7	0	17	17	35	46	12	9	5	1	100	14	3	22	3	1	23	35
	ร	33	41	20	11	12	10	4	20	9	12	5	1	3	3	1	8	0	21	13	53	5	32	22	27	22	9	20	23	14	10	96	11	19	15	7	23	31
	ฉ	0	4	0	0	6	82	49	7	5	0	7	2	0	2	1	1	0	0	0	3	3	3	4	2	0	1	2	0	0	2	3	106	18	15	13	15	8
	ว	1	2	1	0	0	5	0	1	0	0	4	1	0	0	1	0	0	0	1	4	1	4	2	6	1	1	1	1	0	9	13	0	110	2	2	54	28
	ห	5	14	13	4	5	52	10	4	7	0	6	5	6	2	0	2	1	4	9	6	1	18	7	18	20	7	12	4	5	14	3	19	34	107	24	17	23
	พ	3	0	0	1	5	12	10	11	5	0	16	2	0	1	0	2	1	1	0	7	1	1	1	4	1	2	3	4	3	1	4	18	10	21	113	12	18
	อ	0	2	0	0	0	11	1	0	0	0	1	1	0	0	2	0	0	1	0	1	0	1	0	5	2	0	0	1	0	10	2	4	65	7	2	113	41
	ฮ	8	4	3	1	0	5	1	2	1	0	11	2	2	0	1	0	0	1	2	6	0	2	5	11	12	0	6	9	0	20	15	2	8	1	9	19	102

4.3.2 สรุปผลการทดลองจากตาราง 4.3 และ 4.4

จากผลในตารางที่ 4.3 และ 4.4 สามารถสรุปออกมาในรูปของผลการรู้จำแบบ self accept และ accept false ดังที่จะแสดงไว้ในตารางที่ 4.5 และ 4.6 สำหรับ DFA 8 ทิศ และ 16 ทิศ ตามลำดับ

เปอร์เซ็นต์ของการรู้จำในแบบ self accept หมายถึง การทดสอบโดยการนำข้อมูลชุดทดสอบที่เป็นตัวอักษรตัวเดียวกันกับโมเดล DFA ของตัวอักษรนั้นๆ มาทดสอบแล้วผ่านการทดสอบ เช่น เอาชุดทดสอบของตัว “ก” มาทดสอบกับโมเดลของตัว “ก” แล้วผ่านการทดสอบ แล้วคิดออกมาเป็นเปอร์เซ็นต์ก็จะได้ค่าเปอร์เซ็นต์การรู้จำแบบ self accept ของตัว “ก”

ส่วนผลการรู้จำในแบบ accept false คือ การนำเอาชุดทดสอบของตัวอักษรอื่นๆที่ไม่ใช่ตัวอักษรตัวเดียวกันกับโมเดลที่จะทดสอบมาทดสอบแล้วสามารถผ่านการทดสอบได้ เช่น การเอาชุดทดสอบของตัว “ข”, “ค”, “ง”,...,“ฮ” มาทดสอบกับโมเดลตัว “ก” แล้วสามารถผ่านการทดสอบได้ แล้วคิดออกมาเป็นเปอร์เซ็นต์ ก็จะได้เปอร์เซ็นต์การรู้จำแบบ accept false ของตัว “ก”

ตารางที่ 4.5 แสดงผลแบบ self accept กับ accept false ของตัวอักษรแต่ละตัวจาก DFA 8 ทิศ

ตัวอักษร	% Self accept	% Accept false	ตัวอักษร	% Self accept	% Accept false	ตัวอักษร	% Self accept	% Accept false	ตัวอักษร	% Self accept	% Accept false
ก	89.93	19.52	ฎ	97.87	24.27	น	94.74	27.55	ร	87.88	22.47
ข	91.03	27.34	ฏ	80.15	22.92	บ	85.29	24.65	ล	82.71	25.33
ค	91.78	20.53	ฑ	85.04	20.78	ป	94.07	23.97	ว	95.38	19.62
ฆ	90.85	28.76	ฒ	80.71	27.54	ผ	93.33	25.72	ห	94.20	21.72
ง	92.16	16.79	ณ	87.05	32.17	ฝ	88.89	27.28	พ	94.24	27.96
จ	61.59	20.40	ด	85.92	21.22	พ	95.56	27.04	อ	85.29	26.96
ฉ	88.44	28.93	ต	86.33	22.54	ฟ	89.05	27.66	ฮ	89.47	26.34
ช	91.55	26.66	ถ	95.56	23.02	ภ	82.01	29.76			
ซ	95.10	27.14	ท	95.95	27.52	ม	93.53	24.66			
ฌ	71.14	27.24	ธ	81.56	20.78	ย	67.69	27.34			

ตารางที่ 4.6 แสดงผลแบบ self accept กับ accept false ของตัวอักษรแต่ละตัวจาก DFA 16 ทิศ

ตัวอักษร	% Self accept	% Accept false	ตัวอักษร	% Self accept	% Accept false	ตัวอักษร	% Self accept	% Accept false	ตัวอักษร	% Self accept	% Accept false
ก	74.10	12.37	ฎ	78.72	4.61	น	77.44	9.17	ร	72.73	8.42
ข	73.79	16.46	ฏ	80.15	1.92	บ	75.00	14.42	ล	79.70	11.88
ค	80.14	9.08	ฑ	66.93	8.71	ป	71.85	15.80	ว	84.62	13.53
ฆ	71.13	7.75	ฒ	68.57	6.33	ผ	78.52	8.83	ห	77.54	6.80
ง	72.55	7.52	ณ	71.22	4.28	ฝ	79.26	8.07	พ	81.29	5.86
จ	76.82	11.88	ด	64.08	11.52	พ	77.04	13.27	อ	83.09	9.48
ฉ	78.91	6.78	ต	75.54	4.46	ฟ	78.10	13.48	ฮ	76.69	8.23
ช	71.83	9.35	ถ	75.56	6.85	ภ	80.58	7.45			
ซ	77.62	6.53	ท	85.14	14.04	ม	81.29	14.25			
ฌ	76.51	5.54	ธ	81.56	7.21	ย	75.76	4.88			

จากตารางที่ 4.5 และ 4.6 เป็นผลการรู้จำของตัวอักษรแต่ละตัวแยกกัน เพื่อให้เห็นภาพรวมของผลการรู้จำของ DFA ทั้ง 2 แบบ จึงได้หาค่าเฉลี่ยผลการรู้จำของแต่ละแบบ โดยการนำผลของแต่ละแบบรวมกันเฉพาะในแบบของตัวเองแล้วหารด้วยจำนวนของตัวอักษรคือ 37 ตัว ซึ่งผลเฉลี่ยที่ออกมาแสดงอยู่ในตารางที่ 4.7

ตารางที่ 4.7 แสดงผลเฉลี่ยการรู้จำจากตารางที่ 4.5 และ 4.6

ผลทดสอบการรู้จำ	Self accept	Accept false
DFA ของเชนโค้ด 8 ทิศ	87.92 %	24.92 %
DFA ของเชนโค้ด 16 ทิศพร้อมค่าขนาดและตำแหน่ง	**76.52 %**	**9.11 %**

4.3.3 การเพิ่มความยืดหยุ่นในการทดสอบการรู้จำของโมเดล DFA 16 ทิศ

เนื่องจากผลการทดลองของโมเดล DFA 16 ทิศ มีผลแบบ self accept ที่ต่ำกว่าของ DFA 8 ทิศอยู่ประมาณ 10% แต่ว่ามีผลแบบ accept false ที่ต่ำกว่าของ DFA 8 ทิศ อยู่ประมาณ 15% ซึ่งเป็นผลที่ดีขึ้น แต่อย่างไรก็ตามผู้วิจัยยังต้องการให้ผลแบบ self accept ของ DFA 16 ทิศ มีค่าที่สูงขึ้น จึงได้คิดวิธีเพิ่มความยืดหยุ่นในการทดสอบการรู้จำของ DFA 16 ทิศเพิ่มเติมขึ้นมา

โดยเพิ่มเงื่อนไขในการผ่านการทดสอบการรู้จำด้วยการเพิ่มความยืดหยุ่นว่าทิศทางของเชนโค้ดที่นำมาทดสอบไม่จำเป็นต้องมีค่าทิศทางตรงกับของโมเดลโดยตรง แต่ว่าจะต้องมีความคลาดเคลื่อนไม่เกิน +1,-1 และ +2,-2 ยกตัวอย่างเช่น เชนโค้ดของโมเดลตัวอักษร มีค่าเป็น 4 แต่เชนโค้ดของตัวอักษรที่นำมาทดสอบมีค่าเป็น 5 หรือ 3 ก็จะผ่านการทดสอบแบบที่มีความคลาดเคลื่อนที่ +1,-1 (Recog +1,-1) และหากเชนโค้ดของตัวอักษรที่นำมาทดสอบมีค่าเป็น 6 หรือ 2 ก็จะผ่านการทดสอบแบบมีค่าความคลาดเคลื่อน +2,-2 (Recog +2,-2) แต่ถ้าเชนโค้ดที่นำมาทดสอบเหมือนกันกับของโมเดลคือมีค่าทิศทางเป็น 4 ตรงกัน ก็เรียกการผ่านการทดสอบว่าเป็นแบบ Real Recog

ตัวอย่างการทดสอบแบบ Real Recog ได้แสดงตัวอย่างไว้ในหัวข้อก่อนหน้า ในหัวข้อที่ 4.2 รูปที่ 4.2 และ 4.3

ตัวอย่างการทดสอบแบบเพิ่มยืดหยุ่นได้แสดงไว้ในรูปที่ 4.4 และ 4.5 ในรูปที่ 4.4 แสดงการทดสอบแบบเพิ่มความยืดหยุ่นแบบ Recog +1,-1 โดยการนำ unknown ก-6 มาทดสอบกับโมเดล DFA ก ซึ่ง ก-6 สามารถผ่านการทดสอบจากโมเดล DFA ก ได้ เมื่อได้มีการเพิ่มค่าทิศทาง +1 และ -1 เข้าไปในบาง state ของโมเดลของ DFA ก ซึ่งทำให้ค่าเชนโค้ดของ ก-6 สามารถวิ่งไปตาม link เชื่อมต่อของ DFA ก จนไปสิ้นสุดที่ final state ได้

ส่วนในรูปที่ 4.6 แสดงการทดสอบแบบเพิ่มความยืดหยุ่นแบบ Recog +2,-2 โดยการนำ unknown ก-67 มาทดสอบกับโมเดล DFA ก ซึ่ง ก-67 สามารถผ่านการทดสอบจากโมเดล DFA ก เมื่อได้มีการเพิ่มค่าทิศทางเข้าไปโดยการ +1,-1 และ +2,-2 เข้าไปในบาง state ของ DFA ก ซึ่งทำให้ค่าเชนโค้ดของ ก-67 สามารถวิ่งไปตาม link เชื่อมต่อของ DFA ก จนไปสิ้นสุดที่ final state ได้

ในรายละเอียดเชิงลึกของการทดสอบแบบเพิ่มความยืดหยุ่น ผู้วิจัยได้กำหนดให้ทดสอบทิศทางที่ตรงกันก่อนของทั้ง unknown และตัว DFA หากค่าทิศทางของตัว unknown ไม่ตรงกันกับค่า state นั้นๆของ DFA ซึ่งทำให้ไม่สามารถวิ่งไปยัง state ถัดไปได้ ก็จะทำการเพิ่มทิศทาง +1 เข้าไปในโมเดล DFA แล้วทำการทดสอบ ถ้ายังไม่สามารถผ่านการทดสอบใน state นั้นๆได้อีก ก็จะลดทิศทางลง -1 แล้วทำทดสอบ ถ้ายังไม่ผ่านอีกก็จะเพิ่มทิศทาง +2 และ -2 ไปตามลำดับ ในทุกๆ state ที่ทำการทดสอบ จนไปถึง final state หากทุกๆค่าเชนโค้ดของตัว unknown สามารถวิ่งไปตาม link ที่เชื่อมต่อจนไปถึง final state ได้ ก็จะผ่านการทดสอบแบบการเพิ่มความยืดหยุ่นแบบนั้นๆ โดยที่

จะไม่มีการจดจำ state ณ ตำแหน่งใดๆ ที่ทำการเพิ่มหรือลดค่าทิศทาง ทำให้ไม่สามารถย้อนกลับมาทดสอบ ณ ตำแหน่งนั้นๆได้

```
Unknown n-6
7        5        4        3        2        1        0        14       13       12
13       12       15       3
DFAก
L[0][12] S0 -> V[266] = 7
L[266][5] 7 -> V[91] = 5
L[91][1] 5 -> V[92] = 4
L[92][2] 4 -> V[4] = 3
L[4][3] 3 -> V[141] = 2
L[141][3] 2 -> V[145] = 1
L[145][2] 1 -> V[58] = 0
L[58][7] 0 -> V[249] = 13 +1 = 14
L[249][2] 13 -> V[78] = 12 +1 = 13
L[78][2] 12 -> V[122] = 11 +1 = 12
L[122][5] 11 -> V[40] = 13
L[40][1] 13 -> V[41] = 12
L[41][14] 12 -> V[255] = 15
L[255][5] 15 -> V[45] = 4 -1 = 3
+++++++
L[45][4] 4 => 9999          +++ Final State +++      Recog_+-1
+++++++
```

รูปที่ 4.4 แสดงการทดสอบแบบเพิ่มความยืดหยุ่นแบบ Recog +1,-1 ของ unknown ก-6 กับโมเดล DFA ก

```
Unknown n-67
4  3  2  1  2  3  1  3  1  2
3  1  2  0  15 0  15 0  13 0
15 0  15 0  15 0  13 10 12 11
12 4  6  4
DFA n
L[0][8] S0 -> V[241] = 4
L[241][2] 4 -> V[142] = 3
L[142][2] 3 -> V[20] = 2
L[20][4] 2 -> V[222] = 1
L[222][2] 1 -> V[22] = 2
L[22][2] 2 -> V[24] = 3
L[24][3] 3 -> V[29] = 1
L[29][2] 1 -> V[137] = 3
L[137][1] 3 -> V[33] = 1
L[33][2] 1 -> V[101] = 2
L[101][1] 2 -> V[72] = 1 +2 = 3
L[72][1] 1 -> V[73] = 0 +1 = 1
L[73][5] 0 -> V[102] = 2
L[102][2] 2 -> V[178] = 0
L[178][2] 0 -> V[35] = 15
L[35][2] 15 -> V[130] = 0
L[130][1] 0 -> V[76] = 15
L[76][2] 15 -> V[188] = 0
L[188][1] 0 -> V[160] = 13
L[160][2] 13 -> V[189] = 0
L[189][2] 0 -> V[105] = 15
L[189][2] 0 -> V[105] = 15
L[105][2] 15 -> V[188] = 0
L[105][2] 15 -> V[188] = 0
L[188][2] 0 -> V[105] = 15
L[105][2] 15 -> V[188] = 0
L[188][2] 0 -> V[105] = 15
L[105][2] 15 -> V[188] = 0
L[188][1] 0 -> V[160] = 13
L[160][5] 13 -> V[157] = 11 -1 = 10
L[157][3] 11 -> V[82] = 12
L[82][2] 12 -> V[107] = 11
L[107][1] 11 -> V[84] = 12
L[84][3] 12 -> V[45] = 4
L[45][2] 4 -> V[43] = 6
L[43][3] 6 -> V[45] = 4
+++++++
L[45][4] 4 => 9999   +++ Final State +++
+++++++
Recog_+-2
```

รูปที่ 4.5 แสดงการทดสอบแบบเพิ่มความยืดหยุ่นแบบ Recog +2,-2 ของ unknown ก-67 กับโมเดล DFA ก

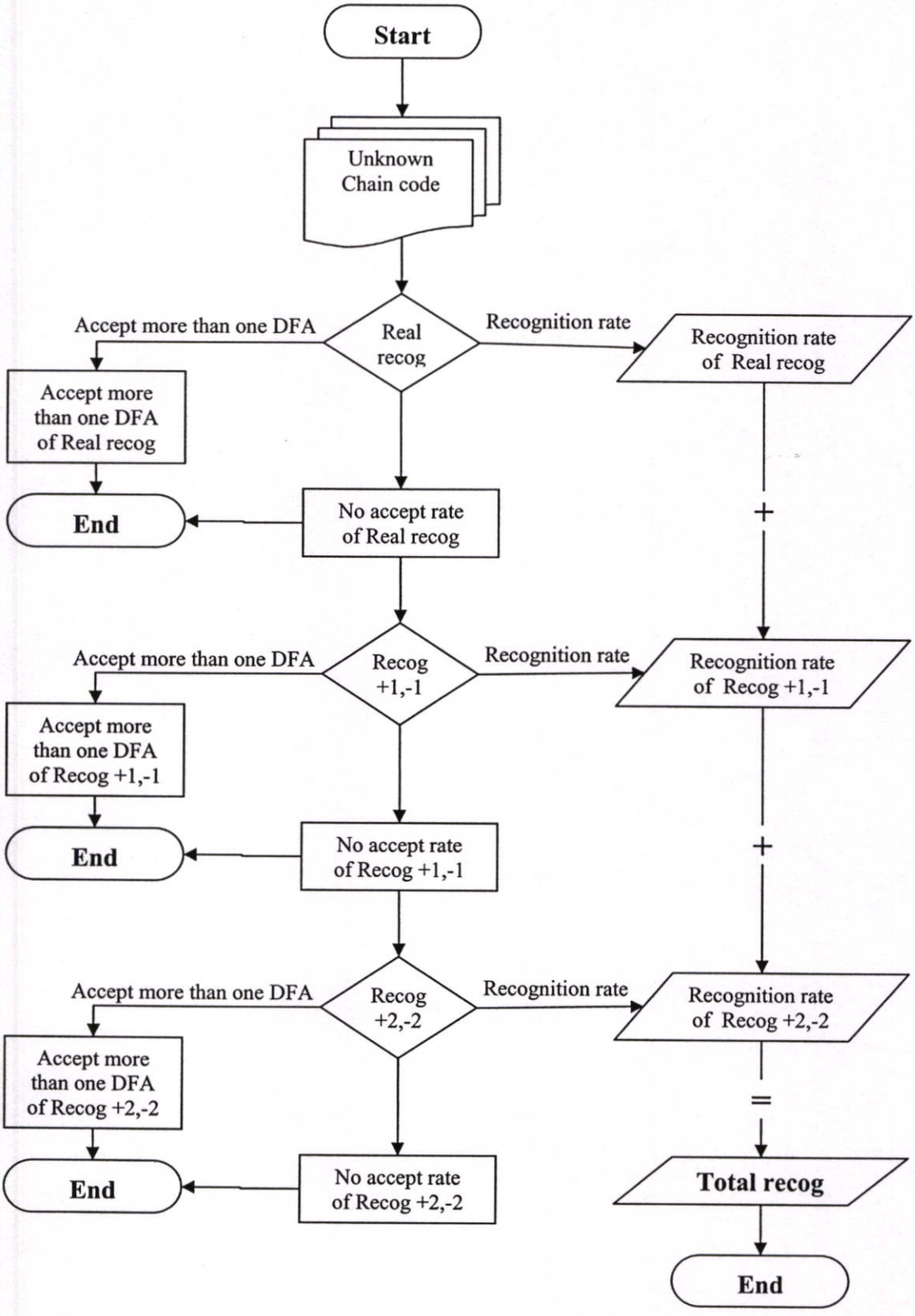
เพื่อให้ผลการทดสอบการรู้จำแบบ Real Recog, Recog +1,-1 และ Recog +2,-2 สามารถบอกรายละเอียดของการทดสอบได้มากขึ้น ผู้วิจัยจึงได้จำแนกการนับผลการผ่านการทดสอบให้ละเอียดขึ้นเป็น 3 ประเภท คือ

1) ถ้าตัวอักษรตัวนั้นผ่านการทดสอบเฉพาะโมเดลของตัวมันเองเพียงโมเดลเดียวเท่านั้น โดยจะไม่ไปผ่านการทดสอบจากโมเดลอื่นอีก และเมื่อคิดเป็นเปอร์เซ็นต์เฉลี่ยของตัวอักษรทั้งหมดแล้ว จะเรียกผลแบบนี้ว่า Recognition rate
2) ถ้าตัวอักษรที่นำมาทดสอบตัวนั้นสามารถผ่านการทดสอบได้มากกว่า 1 โมเดล ซึ่งส่วนใหญ่จะผ่านการทดสอบจากโมเดลของตัวเองและไปผ่านการทดสอบจากโมเดล

ของตัวอักษรอื่นที่มีลักษณะตัวอักษรคล้ายคลึงกับโมเดลของตัวมันเอง เมื่อคิดเป็นเปอร์เซ็นต์เฉลี่ยของตัวอักษรทั้งหมดแล้ว จะเรียกผลแบบนี้ว่า Accept more than one DFA

3) ถ้าตัวอักษรที่นำมาทดสอบตัวนั้นไม่สามารถผ่านการทดสอบจากโมเดลตัวไหนเลย หรือผ่านการทดสอบจากโมเดลของตัวอักษรอื่นที่ไม่ใช่โมเดลของตัวอักษรมันเอง(ซึ่งมีโอกาสเกิดขึ้นได้ แต่เกิดขึ้นจริงน้อยมาก) เมื่อคิดเป็นเปอร์เซ็นต์เฉลี่ยของตัวอักษรทั้งหมดแล้ว จะเรียกผลแบบนี้ว่า No accept rate

โดยในการทดสอบครั้งแรกจะทดสอบแบบ Real Recog เท่านั้น หลังจากนั้นจึงนำผลที่ตัวอักษรเหล่านั้นไม่ผ่านการทดสอบจากโมเดลใดๆเลย (No accept rate) จากการทดสอบแบบ Real Recog มาทดสอบซ้ำแบบ Recog +1,-1 เป็นครั้งที่สอง ซึ่งก็จะยังมีผลบางส่วนที่ไม่ผ่านจากการทดสอบจากแบบ Recog +1,-1 เราก็จะนำผลที่ไม่ผ่านครั้งที่สองนี้มาทดสอบซ้ำเป็นครั้งที่สามโดยทดสอบแบบ Recog +2,-2 สุดท้ายจะได้ผลของ Total recog คือการนำผลของ Real recog + Recog +1,-1 + Recog +2,-2 ซึ่งก็คือผลรวมของ Recognition rate แบบที่เพิ่มความยืดหยุ่นแล้ว โดยการทดสอบแบบเพิ่มความยืดหยุ่นนี้ได้แสดงไว้ใน flow chart ในรูปที่ 4.6 และผลการทดลองดังกล่าวที่ได้กล่าวมาข้างต้นของ DFA แบบ 16 ทิศ ของทุกๆตัวอักษรแยกกันได้แสดงไว้ในตารางที่ 4.8 ส่วนตารางที่ 4.9 เป็นผลของตัวอักษรทุกตัวรวมกัน



รูปที่ 4.6 Flow chart แสดงการทดสอบแบบเพิ่มความยืดหยุ่น

ตารางที่ 4.8 ผลการทดสอบจากการเพิ่มความยืดหยุ่นของ DFA 16 ทิศ ของทุกตัวอักษรตัวแยกกัน

ตัวอักษร	% Real recog			% Recog +1,-1			% Recog +2,-2			% Total recog
	Recognition rate	Accept more than one DFA	No accept rate	Recognition rate	Accept more than one DFA	No accept rate	Recognition rate	Accept more than one DFA	No accept rate	
ก	71.29	10.12	18.59	8.04	3.16	7.39	4.19	0.13	3.07	83.52
ข	70.35	8.79	20.86	7.43	4.85	8.58	4.68	1.35	2.55	82.46
ค	74.21	6.67	19.12	7.59	4.59	6.94	3.52	1.14	2.28	85.32
ฆ	68.10	13.46	18.44	8.69	2.37	7.38	2.84	1.53	3.01	79.63
ง	71.04	9.21	19.75	9.17	3.12	7.46	4.46	0.44	2.56	84.67
จ	73.34	7.33	19.33	10.52	2.50	6.31	3.35	0.75	2.21	87.21
ฉ	74.22	6.75	19.03	7.32	4.95	6.76	2.38	1.12	3.26	83.92
ช	69.38	13.32	17.30	8.25	3.32	5.73	3.86	0.22	1.65	81.49
ซ	71.46	11.82	16.72	7.42	2.25	7.05	4.41	0.32	2.32	83.29
ฌ	70.45	10.31	19.24	9.14	3.42	6.68	3.17	1.20	2.31	82.76
ญ	72.32	8.54	19.14	7.83	3.19	8.12	4.20	1.57	2.35	84.35
ฐ	73.27	7.65	19.08	7.53	3.72	7.83	5.02	0.60	2.21	85.82
ฑ	65.34	12.47	22.19	8.76	4.97	8.46	3.18	1.54	3.74	77.28
ฒ	67.36	9.10	23.54	8.23	6.19	9.12	4.25	0.89	3.98	79.84
ณ	69.15	11.48	19.37	7.39	3.64	8.34	3.84	1.26	3.24	80.38
ด	64.02	13.24	22.74	8.46	5.43	8.85	4.06	1.51	3.28	76.54
ต	71.65	9.48	18.87	8.03	3.53	7.31	2.75	1.34	3.22	82.43
ถ	72.43	8.67	18.90	9.15	2.59	7.16	2.78	1.22	3.16	84.36
ท	80.67	6.13	13.20	6.40	1.86	4.94	2.55	0.43	1.96	89.62
ธ	75.39	8.05	16.56	8.37	1.77	6.42	3.78	0.21	2.43	87.54
น	73.61	8.37	18.02	7.71	3.17	7.14	3.56	0.44	3.14	84.88
บ	72.42	9.45	18.13	7.45	4.03	6.65	3.69	0.35	2.61	83.56
ป	70.28	8.21	21.51	8.34	3.67	9.50	4.51	1.21	3.78	83.13
ผ	73.15	6.94	19.91	8.19	4.53	7.19	3.91	1.13	2.15	85.25
ฝ	74.10	7.53	18.37	9.11	1.78	7.48	4.65	0.28	2.55	87.86
พ	72.29	8.59	19.12	7.49	4.80	6.83	4.53	0.54	1.76	84.31
ฟ	73.16	8.99	17.85	8.74	2.97	6.14	3.59	0.34	2.21	85.49
ภ	74.28	7.38	18.34	8.35	2.67	7.32	4.11	0.81	2.40	86.74
ม	75.33	6.32	18.35	9.21	1.93	7.21	3.25	0.41	3.55	87.79
ย	71.27	8.97	19.76	8.34	3.07	8.35	5.07	0.65	2.63	84.68
ร	69.48	10.31	20.21	7.37	3.73	9.11	5.53	0.37	3.21	82.38
ล	74.32	8.42	17.26	8.39	2.31	6.56	3.56	0.81	2.19	86.27
ว	81.08	2.97	15.95	9.18	1.56	5.21	3.52	0.56	1.13	93.78
ศ	73.27	7.49	19.24	8.76	2.89	7.59	3.59	1.65	2.35	85.62
ษ	75.21	9.22	15.57	7.11	1.92	6.54	3.26	1.14	2.14	85.58
ส	75.25	8.32	16.43	8.67	1.53	6.23	4.42	0.10	1.71	88.34
ห	71.49	9.90	18.61	8.23	2.97	7.41	4.89	0.95	1.57	84.61

ตารางที่ 4.9 ผลการทดสอบจากการเพิ่มความยืดหยุ่นของ DFA แบบ 16 ทิศ รวมของทุกตัวอักษร

	Recognition rate	Accept more than one DFA	No accept rate
Real recog	72.31 %	10.15 %	17.54 %
Recog +1,-1	7.86 %	2.43 %	7.25 %
Recog +2,-2	4.23 %	0.82 %	2.20 %
Total recog	**84.40 %**		

จากผลการทดลองในตารางที่ 4.9 เมื่อนำผลของ No accept rate ที่เป็น 17.54% จากการทดสอบแบบ Real recog ในครั้งแรก มาทำการทดสอบครั้งที่สองแบบ Recog +1,-1 ได้ผลที่ผ่านการทดสอบเพิ่มขึ้นอีก 7.86% และได้ผล No accept rate ของการทดสอบครั้งที่สองเป็น 7.25% จากนั้นเราก็นำผลนี้มาทดสอบใหม่ในครั้งที่สามแบบ Recog +2,-2 ซึ่งได้ผลที่ผ่านการทดสอบเพิ่มขึ้นอีก 4.23% ดังนั้นหากนับรวมผลการทดสอบทั้งหมดจากการเพิ่มยืดหยุ่นในการทดสอบแล้ว ก็จะได้ผลรวมของการผ่านการทดสอบของโมเดล DFA แบบ 16 ทิศ เป็น 17.54% + 7.86% + 4.23% = 84.40% ซึ่งใกล้เคียงกับผล self accept ของโมเดลแบบ 8 ทิศ ที่มีค่า 87.92% แต่ผลการเกิด accept false ของโมเดล DFA ทั้งแบบ 8 ทิศ 16 ทิศ ยังคงเป็นค่าเดิมตามตารางที่ 4.7 เพราะข้อมูลที่ใช้ในการทดสอบผลแบบเพิ่มความยืดหยุ่นก็ยังเป็นข้อมูลชุดเดิม

4.4 วิเคราะห์ผลการทดลอง

จากกระบวนการสร้างโมเดลการรู้จำและทำการทดลองวัดผลการรู้จำของโมเดล DFA ทั้งแบบ 8 และ 16 ทิศ ทำให้เห็นข้อดีข้อด้อยของโมเดลแต่ละแบบ และมองเห็นปัญหาต่างๆทั้งจากกระบวนการสร้างและทดสอบการรู้จำ ซึ่งจะอธิบายในหัวข้อย่อยต่อไป

4.4.1 การเกิด over-generalize จากการใช้ loop pair policy ของโมเดล DFA 8 ทิศ

ในขั้นตอนการสร้างโมเดลการรู้จำของ DFA แบบ 8 ทิศ เราได้นำกระบวนการของ loop pair policy ซึ่งมีการสร้าง loop pair ของเชนโค้ดคู่ที่มีลักษณะซิกแซ็กเป็นขั้นบันได ปัญหาในการสร้าง loop pair คือมันจะเกิด over-generalize ทำให้เกิดการรู้จำลักษณะที่นอกเหนือไปจากลักษณะที่เคยสอนไว้ ทำให้เกิดการรู้จำที่ผิดพลาด (accept false) เนื่องจากตัวอักษรอื่นที่ไม่ใช่ตัวอักษรที่ตรงกับโมเดลที่ทดสอบสามารถผ่านการทดสอบจากโมเดลนั้นได้ ซึ่งได้อธิบายโดยละเอียดไว้ก่อนหน้านี้แล้วในบทที่ 2 หัวข้อที่ 2.6

4.4.2 โมเดล DFA 16 ทิศ มีความยืดหยุ่นน้อย (specific) จากการใช้ค่าขนาดและตำแหน่ง

กระบวนการสร้างโมเดล DFA 16 ทิศ เราได้นำค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาช่วยในกระบวนการสร้างโมเดลการรู้จำ ซึ่ง DFA จะมีการใช้โหนดร่วมการได้นอกจากมีค่าทิศทางที่เท่ากันแล้ว จะต้องมีค่าขนาดและตำแหน่งที่ใกล้เคียงกันด้วย คือจะต้องมีค่าขนาดต่างกันไม่เกิน 10% และค่าตำแหน่งต่างกันไม่เกิน 5%

ในด้านดีจะเห็นว่าโมเดล DFA ที่สร้างมาทิศทางที่ละเอียดขึ้นนำค่าขนาดและตำแหน่งมาช่วยวิเคราะห์ในขั้นตอนการสร้างโมเดล ทำให้มีการจดจำข้อมูลลักษณะของตัวอักษรแต่ละตัวที่มากขึ้น แยกแยะความแตกต่างของตัวอักษรได้มากกว่าการจดจำเพียงแค่ทิศทาง และไม่มีการสร้าง loop pair ทำให้ผลการทดสอบแบบ accept false ของ DFA 16 ทิศ มีค่าน้อยกว่า DFA แบบ 8 ทิศ ถึงประมาณ 15% ซึ่งเป็นผลที่ดีขึ้น เมื่อเทียบกับงานวิจัยเดิม [1]

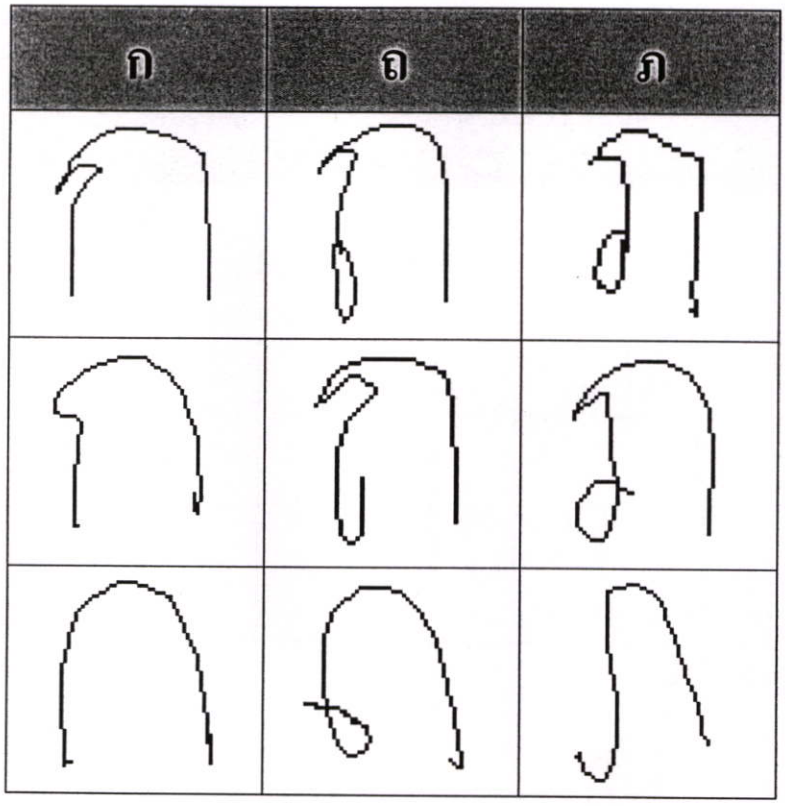
แต่เนื่องจากเงื่อนไขที่ตั้งไว้ในการสร้าง DFA แบบนี้ค่อนข้างเจาะจงมากไป ทำให้ผลการทดสอบการรู้จำแบบ self accept มีค่าน้อยกว่าของโมเดลแบบ 8 ทิศ จึงต้องคิดวิธีเพิ่มความยืดหยุ่นเพื่อเพิ่มผลการทดสอบให้ดีขึ้น ซึ่งก็ทำให้ผลการทดสอบจากการที่เพิ่มความยืดหยุ่นแล้วออกมาใกล้เคียงกับผลแบบ self accept ของ DFA 8 ทิศ ซึ่งวิธีการเพิ่มความยืดหยุ่นให้กับ DFA 16 ทิศ ก็ให้ผลที่น่าพอใจเช่นกัน

โดยรวมแล้วโมเดล DFA 16 ทิศ ถึงจะมีความ specific ที่มากไปแต่เมื่อเพิ่มความยืดหยุ่นในการทดสอบให้แล้ว ก็เป็นโมเดลที่สร้างการรู้จำที่มีประสิทธิภาพและให้ผลการรู้จำที่ดี

4.4.3 การเกิด accept false จากลักษณะการเขียนของตัวอักษรที่คล้ายคลึงกัน

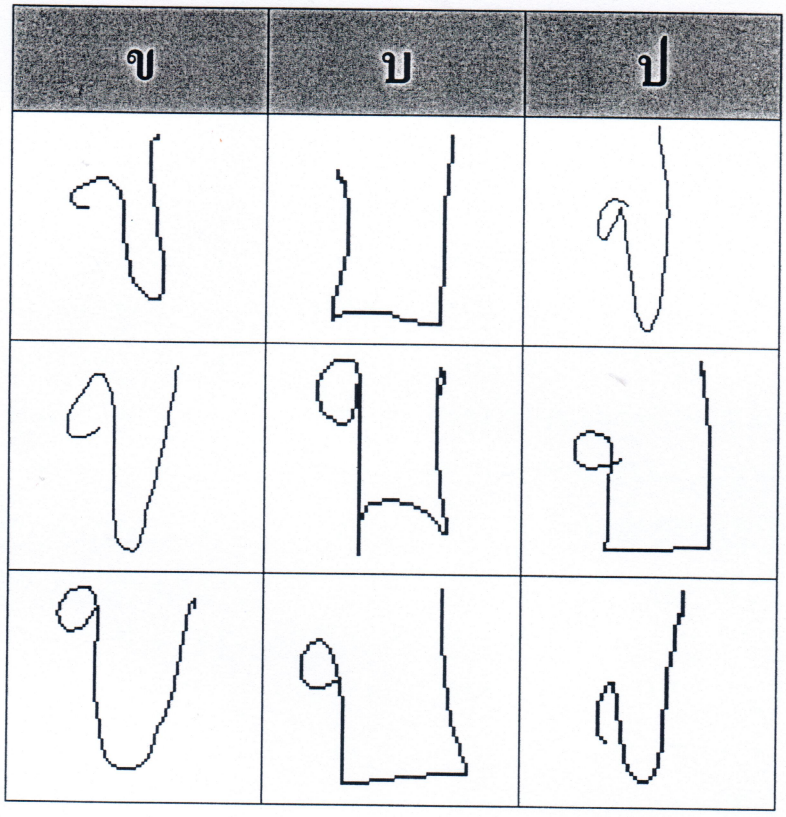
การเกิด accept false ที่หมายถึงเกิดการจดจำที่ผิดพลาด นอกจากเกิดจากการสร้าง loop pair แล้ว ยังสามารถเกิดได้จากการที่ลักษณะเชนโค้ดของข้อมูลที่นำมาสอนโมเดลของตัวอักษรบางตัวมีความคล้ายคลึงกันมาก อันเนื่องมาจากลักษณะของตัวอักษรนั้นมีความคล้ายคลึงกันอยู่แล้ว หรืออาจจะเกิดจากการเขียนในลายมือของแต่ละบุคคลที่ทำให้ตัวอักษรบางตัวมีลักษณะที่คล้ายคลึงกัน

เช่น ลักษณะของตัวอักษร “ก” , “ถ” และ “ภ” จะเห็นว่าตัวอักษรทั้ง 3 ตัว มีรูปแบบที่คล้ายคลึงกันอยู่แล้ว ต่างกันตรงที่ไม่มีหัว มีหัวอยู่ด้านใน และหัวอยู่ด้านนอก ซึ่งลักษณะเชนโค้ดโดยรวมจะเหมือนๆกัน ยิ่งถ้าหากลายมือของบางคนชอบเขียนตัวอักษรแบบไม่มีหัวอยู่แล้ว ก็จะยิ่งแยกความแตกต่างของตัวอักษรได้ยากมาก ดังแสดงในรูปที่ 4.3



รูปที่ 4.7 แสดงลายมือเขียนของตัว “ก” , “ถ” และ “ภ” ซึ่งมีลักษณะคล้ายคลึงกัน

ลักษณะของตัวอักษรที่มีความคล้ายคลึงกันอีก 3 ตัวจากการเขียนคือ “ข” , “บ” และ “ป” ซึ่งตัว “บ” และ “ป” มีความคล้ายคลึงกันตั้งแต่ตัวพิมพ์ต่างกันตรงที่มีไม่หางกับมีหาง ส่วนตัว “ข” ถ้าเป็นลักษณะของตัวพิมพ์อาจจะดูแตกต่างกับ “บ” และ “ป” อย่างชัดเจน แต่ถ้าเป็นลักษณะของตัวเขียนแล้ว บางครั้งในลายมือของแต่ละบุคคลจะดูคล้ายคลึงกันมากของทั้ง 3 ตัวอักษร ดังแสดงในรูปที่ 4.8

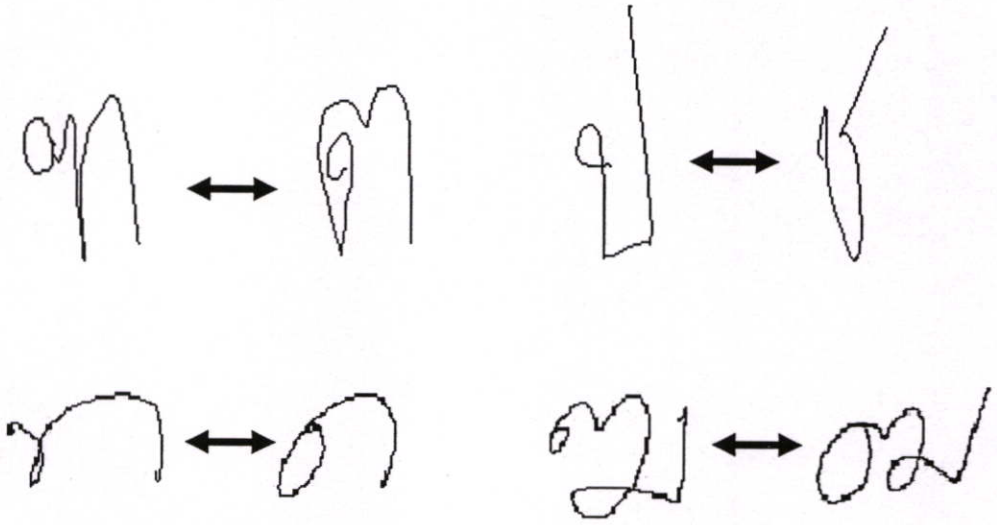


รูปที่ 4.8 แสดงลายมือเขียนของตัว “ข” , “บ” และ “ป” ซึ่งมีลักษณะคล้ายคลึงกัน

นอกจากตัวอักษรชุดของ “ก” , “ถ” และ “ภ” กับตัวอักษรชุดของ “ข” , “บ” และ “ป” แล้วยังมีลักษณะของตัวอักษรที่คล้ายคลึงกันเป็นคู่ๆคือ “บ-ป” , “ผ-ฝ” , “พ-ฟ” , “ฏ-ฏ” , “ช-ซ” , “ท-ฑ” ซึ่งมีความแตกต่างกันเพียงเล็กน้อยตรงที่มีหางกับไม่มีหาง และมีลักษณะของหยักมากกว่าเพียงหนึ่งครั้งเท่านั้น ทำให้ผล accept false ของตัวอักษรเหล่านี้มีค่าสูง เพราะเกิดการจำจดที่ผิดพลาดไปจดจำตัวอักษรใกล้เคียงกันที่เป็นคู่ของมันด้วย

บางครั้งอาจมีการจดจำที่ผิดพลาดเนื่องจากเชนโค้ดของตัวอักษรเหล่านั้นมีความคล้ายคลึงกันโดยบังเอิญ เช่น เชนโค้ดของตัว “ฑ” จะคล้ายกับของตัว “ต” เพราะมีการเขียนหยักตรงส่วนหัวกับส่วนตรงกลางตัวอักษรที่คล้ายกัน ส่วนตัว “ป” และ “ช” ถ้าส่วนที่หยักของ “ช” ไม่ชัดเจนหรือหยักน้อยจนเกือบจะเป็นเส้นตรงซึ่งคล้าย “ป” ส่วนคู่ตัวอักษรของ “ท-ด” และ “ฆ-ฒ” ก็มีเชนโค้ด

คล้ายกับลักษณะที่ได้กล่าวไว้ข้างต้น ทำให้ลำดับเชนโค้ดของตัวอักษรเหล่านี้มีความใกล้เคียงกัน มากจนทำให้เกิดการจดจำผิดพลาดได้ ดังแสดงในรูปที่ 4.9



รูปที่ 4.9 ตัวอักษรที่มีลำดับของเชนโค้ดใกล้เคียงกัน

นอกจากนี้ยังเกิดการจดจำที่ผิดพลาดได้เนื่องจากลักษณะการเขียนที่ไม่ชัดเจนของลายมือเขียนในแต่ละบุคคล เช่น ระหว่างตัว “จ-ล” และ “ธ-ร” ซึ่งบางครั้งแม้จะแยกความแตกต่างด้วยสายตาของคนเอง ก็ยังอาจจะแยกไม่ได้ว่าเป็นตัวอักษรใด ดังแสดงในรูป 4.10

รูปที่ 4.10 ตัวอักษรที่เขียนไม่ชัดเจน แยกความแตกต่างจากสายตายังยาก

บทที่ 5
บทสรุป

ในงานวิจัยนี้ผู้วิจัยได้ใช้ค่าทิศทาง 16 ทิศที่มีค่าขนาดและตำแหน่งของตัวอักษรร่วมด้วย มาใช้สร้างโมเดลการรู้จำ DFA ของตัวอักษรลายมือเขียนภาษาไทยแบบออนไลน์ขึ้นมา โดยมีความมุ่งหวังจะสร้างโดเมลการรู้จำ DFA ให้มีประสิทธิภาพมากขึ้น ต้องการให้โมเดลของตัวอักษรแต่ละตัวมีความเฉพาะเจาะจงมากขึ้น (specific) มากกว่าของงานวิจัยก่อนหน้า [1] โดยที่โมเดลที่สร้างขึ้นมาใหม่นี้ยังคงให้ผลการรู้จำที่ดีกว่าหรือใกล้เคียงกับงานวิจัยเดิม [1]

ข้อมูลที่นำมาเป็นอินพุทให้กับกระบวนการสร้างการรู้จำของ DFA จะได้มาจากเขียนด้วยปากกาอิเล็กทรอนิกส์ ซึ่งจะให้ค่าที่เป็นจุดข้อมูลที่อยู่ในรูปของคู่ลำดับ (x,y) จากนั้นนำมาสู่กระบวนการสร้างเชนโค้ดแบบ GCC [2] ซึ่งเราจะสร้างเชนโค้ดให้ออกมา 2 แบบคือ เชนโค้ดแบบ 8 และ 16 ทิศ และทำการคำนวณหาค่าขนาดและตำแหน่งของเชนโค้ดแบบ 16 ทิศขึ้นมาด้วย ต่อมาก็นำข้อมูลอินพุททั้ง 2 แบบ นำมาสร้างเป็นโมเดลการรู้จำของ DFA ทั้ง 2 แบบ คือ แบบของงานวิจัยก่อนหน้า [1] ที่ใช้เชนโค้ด 8 ทิศซึ่งจะนำกระบวนการของ loop pair policy มาใช้ด้วย และอีกแบบคือวิธีการที่คิดขึ้นมาใหม่ในงานวิจัยนี้ คือใช้เชนโค้ด 16 ทิศ พร้อมค่าขนาดและตำแหน่ง ซึ่งจะไม่นำกระบวนการของ loop pair policy มาใช้ จากนั้นทำการวัดผลการรู้จำของ DFA ทั้ง 2 แบบ แล้ววิเคราะห์หาข้อดีข้อด้อยของแต่ละแบบ

ผลที่ออกมาแสดงให้เห็นว่าการสร้าง DFA จากเชนโค้ด 8 ทิศ ให้ผลการทดสอบแบบ self accept ที่สูงกกว่าแบบ 16 ทิศ แต่ก็มีผลของ accept false ที่สูงกว่าด้วยเช่นกัน ซึ่งเป็นผลมาจากการสร้าง loop pair ของกระบวนการ loop pair policy แสดงให้เห็นว่ากระบวนการสร้างโมเดลแบบ 8 ทิศ ทำให้โมเดลมีความ generalize มากจนเกินไป จนเกิดเป็น over-generalize ทำให้เกิดการรู้จำที่ผิดพลาดมากขึ้นตามมา เลยทำให้ผลของ accept false ออกมาค่อนข้างสูง ซึ่งเป็นผลที่ไม่น่าพอใจนัก ซึ่งเป็นจุดที่ให้งานวิจัยนี้ต้องการจะแก้ให้ปรับปรุงให้ดีขึ้น

ส่วนผลจากการทดสอบของโมเดล DFA แบบ 16 ทิศ ให้ผลแบบ self accept ที่ต่ำกว่าของแบบ 8 ทิศ แต่ว่าก็มีผลของ accept false ที่ต่ำกว่าของ DFA 8 ทิศ อยู่มากด้วยเช่นกัน คือต่ำกว่าประมาณ 15% ซึ่งเป็นผลที่ออกมาดีกว่าในงานวิจัยก่อนหน้า [1] แสดงให้เห็นว่ากระบวนการสร้างโมเดลการรู้จำของ DFA แบบ 16 ทิศ จะมีความเป็น specific ที่สูงกว่าแบบ 8 ทิศ ซึ่งอาจจะมีความเป็น specific ที่สูงเกินไปจนทำให้ผลแบบ self accept ไม่สูงเท่าที่คาดหวังไว้ และเนื่องจากผลแบบ self accept ออกมาไม่สูงมากนัก ผู้วิจัยจึงได้คิดการทดสอบแบบเพิ่มความยืดหยุ่นขึ้นมา โดยใช้ค่าความยืดหยุ่นของ +1,-1 และ +2,-2 มาช่วยปรับทิศทางของเชนโค้ดที่นำมาทดสอบเพื่อให้ผ่านการทดสอบมากขึ้น ซึ่งผลที่ออกมาหลังการเพิ่มความยืดหยุ่นในการทดสอบ ทำให้ผลการรู้จำของ DFA

แบบ 16 ทิศ สูงขึ้นจนใกล้เคียงกับ DFA แบบ 8 ทิศได้ ซึ่งเป็นวิธีการที่แก้ไขปัญหาความ specific ของโมเดล 16 ทิศ ได้ดีพอสมควร

โดยภาพรวมแล้วกระบวนการสร้าง DFA แบบ 16 ทิศ ก็เป็นกระบวนการสร้างการรู้จำที่มีประสิทธิภาพ สามารถแยกแยะตัวอักษรได้อย่างมีประสิทธิภาพพอสมควร เนื่องจากมีการแบ่งทิศทางที่ละเอียดขึ้น ไม่มีการสร้าง loop pair ซึ่งจะไม่เกิด accept false จาก loop pair และมีการใช้ค่าขนาดและตำแหน่งมาช่วยวิเคราะห์ในขั้นตอนการสร้าง DFA ซึ่งทำให้โมเดลที่สร้างออกมามีความ specific ของตัวอักษรแต่ละตัวได้มากขึ้น แต่การเป็น specific ของโมเดลนี้ก็อาจมีมากจนเกินไป จนทำให้ผลแบบ self accept น้อยกว่าของ DFA 8 ทิศ แต่ผู้วิจัยก็ใช้วิธีเพิ่มความยืดหยุ่นในการทดสอบ จึงทำให้ผลการรู้จำของโมเดลแบบ 16 ทิศ ออกมามีค่าใกล้เคียงกับของโมเดล 8 ทิศ

ผู้วิจัยมองเห็นปัญหาจากขั้นตอนการสร้างลายมือเขียนจากการเขียนของบุคคล 20 คน ที่ทำการเขียนตัวอักษรทั้ง 37 ตัว ตัวละ 20 ครั้ง ซึ่งลักษณะการเขียนสำหรับผู้เขียนบางคน จะมีลักษณะการเขียนตัวอักษรที่จะทำให้เกิด noise สูง คือบางครั้งจะเป็น noise ส่วนต้นของตัวอักษรตอนเริ่มจดปากกาลงเขียน และก็บางครั้งจะเป็น noise ของส่วนท้ายของตัวอักษรจากตอนที่ยกปากกาขึ้นหลังเขียนเสร็จ ซึ่งเป็นลักษณะเฉพาะของการเขียนของแต่ละบุคคล ถ้าเรานำข้อมูลของตัวอักษรจากการเขียนมาหาวิธีการเพิ่มเติมในการตัดหรือลดทอน noise บางส่วนออกไปก่อนที่จะนำมาสร้างเชนโค้ดได้ จะทำให้กระบวนการสร้างการรู้จำจะมีประสิทธิภาพมากขึ้น ผลการรู้จำน่าจะสูงขึ้น

ผู้วิจัยเห็นว่าการนำค่าขนาดและตำแหน่งของเชนโค้ดแต่ละตัวมาใช้ ยังสามารถนำมาใช้วิเคราะห์นอกเหนือไปจากนำมาช่วยในขั้นตอนการสร้าง DFA ได้ เช่น ถ้าเรานำค่าขนาดและทิศทางมาช่วยในขั้นตอนการทดสอบผลการรู้จำด้วย ยกตัวอย่างเช่น จะผ่านการทดสอบการรู้จำได้เชนโค้ดตัวที่นำมาทดสอบนอกจากจะมีทิศทางที่ตรงกับของโมเดลแล้วยังต้องมีขนาดและตำแหน่งของเชนโค้ดตัวนั้นที่จะต้องใกล้เคียงกับค่าขนาดและตำแหน่งของโมเดลด้วย จึงจะผ่านการทดสอบ state นั้นไปได้ ซึ่งถ้าเรานำค่าของขนาดและตำแหน่งมาใช้ทดสอบด้วยอาจจะทำให้สามารถแยกความแตกต่างของตัวอักษรที่มีลักษณะใกล้เคียงกันแบบที่เป็นตัวที่ไม่มีหางกับตัวมีหางได้ดีขึ้น เช่น ตัว “บ-ป” , “ผ-ฝ” และ “พ-ฟ” แต่ผลการทดสอบอาจจะไม่สูงขึ้นอย่างที่หวังไว้ก็ได้ เพราะจะยิ่งทำให้การทดสอบมีความ specific มากขึ้นไปอีก แต่อย่างไรก็ตามผู้วิจัยเห็นว่าการใช้ค่าของขนาดและตำแหน่งร่วมกับค่าทิศทางน่าจะช่วยทำการวิเคราะห์แยกแยะความต่างของตัวอักษรได้มากกว่าการใช้ค่าทิศทางเพียงอย่างด้วย เพียงแต่ต้องหากระบวนการที่จะนำมาใช้ให้เหมาะสม จึงจะไม่ทำให้การทดสอบมีความ specific มากจนเกินไป

เอกสารอ้างอิง

[1] ปองเกษม พลสันติกุล, **"การรู้จำลายมือเขียนภาษาไทยแบบออนไลน์โดยใช้คอนแท็กฟรีแกรมม่าจากการเรียนรู้เพิ่มเติม,"** วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2548.

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

[3] M. Okamoto, K. Yamamoto, **"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, pp. 491-494, September 20-22, 1999.

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

[5] 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, pp. 112-115, Aug. 14-16, 1995.

[6] คำเพ็ด บุญนะดี, **"การรู้จำลายมือเขียนตัวอักษรลาวแบบออนไลน์โดยใช้ลักษณะเด่นทางโครงสร้างแบบยืดหยุ่น,"** วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2546.

[7] สุวิมล (ก่อเกิดวิบูลย์) ฮอลล์, **"ทฤษฎีการคณนา Theory of Computation,"** ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2542.

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

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

[10] 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

[11] 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.

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

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

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

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

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

[17] 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.

[18] 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.

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

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

[21] 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, pp. 107-110, 20-22 September 1999.

[22] 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, pp. 605-608, 19-22 April 1994.

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

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

[25] 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.

[26] 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.

[27] 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, pp.33-38, Jan. 22-24, 2003.

[28] 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.

[29] 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.

ภาคผนวก

ภาคผนวก ก.

ทฤษฎีของดีเทอร์มินิสติกไฟไนท์ออโตมาต้า

ดีเทอร์มินิสติกไฟไนท์ออโตมาต้า

Deterministic Finite Automata (DFA)

ดีเทอร์มินิสติกไฟไนท์ออโตมาต้า (Deterministic Finite Automata : DFA) หรือ ออโตมาต้าจำกัดเชิงกำหนด เป็นส่วนหนึ่งของออโตมาต้าจำกัด (Finite Automata : FA) ซึ่ง FA เป็นโมเดลที่ถูกออกแบบมานานแล้ว เพื่อเป็นโมเดลที่ใช้ในการคำนวณซึ่งได้นำแนวคิดนี้มาใช้ในการออกแบบสร้างคอมพิวเตอร์ในยุคแรกๆและได้พัฒนาจนกลายมาเป็นคอมพิวเตอร์ที่มีประสิทธิภาพมากขึ้นที่ใช้งานกันอยู่ในปัจจุบัน และ FA ยังเป็นต้นแบบซึ่งถูกนำมาใช้ออกแบบคอมไพเลอร์และอินเทอร์พรีเตอร์ และยังเป็นต้นแบบที่ใช้ในขั้นตอนวิธีจับคู่แบบ pattern matching algorithm อีกด้วย ส่วน DFA ก็เป็น FA แบบหนึ่งที่เป็นเชิงกำหนด เมื่อมีอินพุทเข้ามาค่าหนึ่งจะต้องเปลี่ยนไปอยู่อีกสเตจหนึ่งเพียงสเตจเดียวเท่านั้น ซึ่งเหมาะที่จะนำมาประยุกต์ใช้เป็นโมเดลที่ใช้ในการรู้จำของงานวิจัยนี้ ซึ่งรายละเอียดและทฤษฎีต่างๆเกี่ยวกับ DFA จะแสดงอยู่ในหัวข้อย่อยต่อไปนี้

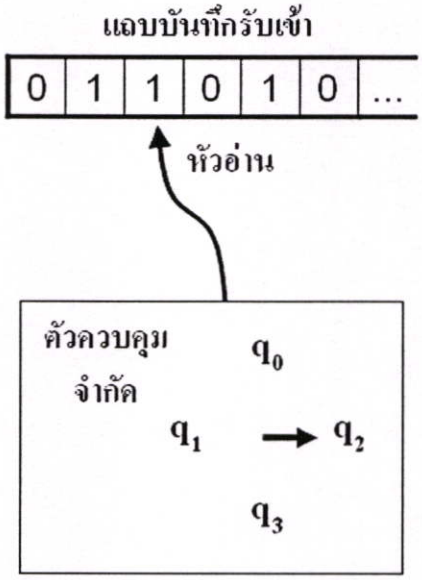
แนวคิดการนำ DFA มาสร้างเป็นเครื่องคอมพิวเตอร์

หากจะนำโมเดลของ FA หรือ DFA มาใช้สร้างเครื่องคอมพิวเตอร์ ก็จะเป็นเครื่องคอมพิวเตอร์ที่มีเฉพาะซีพียู ซึ่งมีความจุเป็นค่าตรึงถาวรที่ขึ้นอยู่กับการออกแบบไว้ตั้งแต่ต้น และมีส่วนที่ทำหน้าที่อ่านข้อมูลเข้าเท่านั้นโดยไม่มีหน่วยความจำช่วย

เราสามารถสร้างภาพของ DFA แต่ละเครื่องว่าเป็นอุปกรณ์ที่มี สายอักขระรับเข้า (input string) ถูกส่งผ่านเข้ามาทาง แถบบันทึกรับเข้า (input tape) ที่ถูกแบ่งออกเป็นช่องๆ โดยแต่ละช่องจะบันทึก สัญลักษณ์ (symbol) เพียงตัวเดียว ตามรูปที่ 1

ส่วนสำคัญของอุปกรณ์นี้คือส่วนที่เรียกว่า ตัวควบคุมจำกัด (finite control) ซึ่ง ณ ขณะหนึ่งๆ จะต้องอยู่ใน สถานะ (state) เพียงสถานะเดียวเท่านั้นจากเซตของสถานะทั้งหมดซึ่งเป็นเซตจำกัด ตัวควบคุมนี้อ่านสัญลักษณ์จากแถบบันทึกรับเข้าโดย หัวอ่าน (reading head) โดยจะอ่านจากสัญลักษณ์ตัวซ้ายสุดจากแถบบันทึกเข้า และเริ่มอ่านจาก สถานะเริ่มต้น (initial state) q_0 โดยในการเคลื่อนของหัวอ่านแต่ละครั้ง DFA จะอยู่ในสถานะใดสถานะหนึ่งเสมอ เช่นในรูปที่ 1 DFA อยู่ในสถานะ q_2 กำลังอ่านสัญลักษณ์ 1 เมื่ออ่านเสร็จหัวอ่านจะเคลื่อนไปทางขวา 1 ช่อง และ DFA อาจจะเปลี่ยนสถานะหรืออยู่ในสถานะเดิมขึ้นอยู่กับสถานะที่อยู่เดิมและสัญลักษณ์ที่เพิ่งอ่านเสร็จ ทำเช่นนี้ไปเรื่อยๆจนกว่าจะจบสายอักขระทั้งสาย เครื่องจะชี้ว่า “ยอมรับ” (accept) หรือ “ไม่ยอมรับ” (reject) โดยดูจากสถานะสุดท้ายเมื่ออ่านสายอักขระเสร็จ โดยเครื่องจะชี้ว่า

ยอมรับ สายอักขระนั้นก็ต่อเมื่อสถานะที่อยู่ขณะนั้นเป็นสมาชิกตัวหนึ่งของกลุ่มที่เครื่องจัดไว้ตั้งแต่แรกว่าเป็นกลุ่มของ สถานะยอมรับ (accepting state)



รูปที่ 1 แนวคิดการนำ DFA มาสร้างเป็นเครื่องคอมพิวเตอร์

นิยามของ DFA

เราจะเขียนแทน DFA แต่ละเครื่องด้วยสัญลักษณ์ (Q,Σ,δ,q_0,F) ซึ่ง DFA แต่ละเครื่องประกอบไปด้วย

1. เซตของ สถานะ (state) ทั้งหมดซึ่งเป็นเซตจำกัด เขียนแทนเซตนี้ด้วย Q
2. เซตของ สัญลักษณ์รับเข้า (input symbol) ทั้งหมดซึ่งเป็นเซตจำกัด เขียนแทนเซตนี้ด้วย Σ
3. สถานะเริ่มต้น (initial state) ซึ่งเป็นสมาชิกของ Q เขียนแทนสถานะนี้ด้วย q_0
4. เซตของ สถานะยอมรับ (accepting state) ทั้งหมดซึ่งเป็นสับเซตของ Q เขียนแทนเซตนี้ด้วย F
5. ฟังก์ชัน δ จาก $Q\times\Sigma$ ไปยัง Q ฟังก์ชันนี้มีชื่อเรียกว่า ฟังก์ชันการผ่าน (transition function) เขียนแทนด้วยสัญลักษณ์ $\delta : Q\times\Sigma \to Q$

เพื่อง่ายต่อการทำความเข้าใจ จะทำการยกตัวอย่าง DFA เครื่องหนึ่งเพื่อแสดงความสัมพันธ์ของสัญลักษณ์ต่างๆจากนิยามของ DFA ซึ่งแสดงอยู่ในตัวอย่างที่ 1

ตัวอย่างที่ 1 ให้ $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$ และ δ เป็นฟังก์ชันนิยามดังนี้

$$\delta(q_0, a) = q_0 \text{ , } \delta(q_0, b) = q_1 \text{ , } \delta(q_1, a) = q_1 \text{ และ } \delta(q_1, b) = q_0$$

จะได้ว่า (Q,Σ,δ,q_0,F) เป็น DFA เครื่องหนึ่ง ซึ่งสามารถเขียนเป็น transition table ได้ ตามตารางที่ 1

ตารางที่ 1 transition table ของตัวอย่างที่ 1

สถานะ \ สัญลักษณ์รับเข้า	a	b
q_0	q_0	q_1
q_1	q_1	q_0

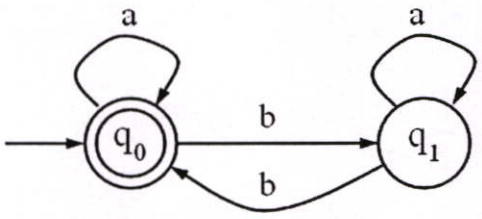
แผนภาพการผ่าน (transition diagram) ของ DFA

เราสามารถบอกรายละเอียดทั้งหมดของ DFA แต่ละเครื่องโดยใช้แผนภาพที่มีชื่อเรียกว่า แผนภาพการผ่าน (transition diagram) ซึ่งนิยามได้ดังนี้

(Q,Σ,δ,q_0,F) แต่ละเครื่อง คือกราฟระบุทิศทางที่

1. มี Q เป็นเซตที่เป็นจุดยอดทั้งหมด และมีลูกศรชี้เข้าที่จุด q_0
2. มีวงกลม 2 วงล้อมรอบจุดยอดทุกจุดใน F
3. จะมีอาร์กเชื่อมจากจุดยอด p ไปยังจุด q โดยมีสัญลักษณ์รับเข้า a เขียนกำกับ ก็ต่อเมื่อ $\delta(p, a) = q$

จากนิยามทั้ง 3 ข้อข้างต้น นำมาสร้างเป็น transition diagram ของ DFA จากตัวอย่างที่ 1 และตารางที่ 1 ได้เป็น transition diagram รูปที่ 2



รูปที่ 2 transition diagram ของ DFA จากตัวอย่างที่ 1 และตารางที่ 1

โครงแบบ (configuration) ของ DFA

นอกจากการใช้แผนภาพการผ่านบอกรายละเอียดทั้งหมดของ DFA แต่ละเครื่องแล้ว เรายังสามารถใช้สัญลักษณ์ทางคณิตศาสตร์แสดงภาพของ DFA ที่กำลังอ่านสายอักขระรับเข้า ณ ขณะใดๆได้ และเนื่องจากหัวอ่านของ DFA เคลื่อนขวาได้เพียงอย่างเดียว เราจึงสามารถตัดส่วนของสายอักขระรับเข้าที่เครื่องได้อ่านไปแล้วออกจากสัญลักษณ์ทางคณิตศาสตร์ที่แสดงภาพของ DFA ที่กำลังอ่านสายอักขระรับเข้า ณ ขณะนั้นได้ สัญลักษณ์นี้มีชื่อเรียกว่า โครงแบบ (configuration) ซึ่งนิยามได้ดังนี้

โครงแบบ (configuration) ของ DFA ซึ่งมี $M = (Q, \Sigma, \delta, q_0, F)$ คือคู่อันดับ $(q, w) \in Q \times \Sigma^*$ (ซึ่งมีความหมายว่าขณะนั้น M กำลังอยู่ในสถานะ q หัวอ่านอยู่ที่สัญลักษณ์ตัวแรกสุดทางซ้ายมือของ w และ w เป็นส่วนที่เหลือของสายอักขระรับเข้าที่เครื่องยังไม่ได้อ่าน) เช่น ในรูปที่ 1 (q_2, 1010) เป็นโครงแบบของ DFA ในขณะนั้น

จะกล่าวว่า (q, w) เป็นโครงแบบถัดจาก (p, x) เขียนแทนด้วย (p, x) $\vdash$ (q, w) ก็ต่อเมื่อ x = aw โดยที่ a $\in \Sigma$, w $\in \Sigma^*$ และ δ (p, a) = q

(q, w) เป็นโครงแบบที่ได้มาจาก (p, x) เขียนแทนด้วย (p, x) $\vdash^*$ (q, w) ก็ต่อเมื่อ (p, x) $\vdash$ (q, w) หรือมีโครงแบบ $c_1, c_2, \ldots, c_n$ ที่ทำให้ c_1 = (p, x) , $c_1 \vdash c_2$, $c_2 \vdash c_3$, ... , $c_{n-1} \vdash c_n$ และ c_n = (q, w)

M ยอมรับ สายอักขระรับเข้า w ก็ต่อเมื่อ มี q $\in F$ ที่ทำให้ (q_0, w) $\vdash^*$ (q, ε) ในกรณีนี้เราเรียก (q, ε) ว่าเป็น โครงแบบยอมรับ (accepting configuration)

และ ภาษาที่ M ยอมรับ เขียนแทนด้วย $L(M)$ คือ {w $\in \Sigma^*$ | M ยอมรับ w} ดังนั้น ภาษา L ใดๆ จะเป็นภาษาที่ M ยอมรับ ก็ต่อเมื่อ $L = L(M)$

เพื่อการทำความเข้าใจที่ง่ายขึ้น จะยกตัวอย่างการตรวจสอบสายอักขระของ M ซึ่งเป็น DFA ที่สร้างมาจากตัวอย่างที่ 1 ว่าจะยอมรับสายอักขระที่นำมาทดสอบหรือไม่ ซึ่งจะตรวจสอบด้วยวิธี โครงแบบ (configuration) ดังแสดงอยู่ในตัวอย่างที่ 2 และจะยกอีกตัวอย่างซึ่งกำหนดให้ต้องออกแบบ DFA ขึ้นมาเพื่อใช้ยอมรับสายอักขระที่กำหนดให้ ซึ่งจะแสดงอยู่ในตัวอย่างที่ 3

ตัวอย่างที่ 2 ให้ $M=(Q,\Sigma,\delta,q_0,F)$ เป็น DFA ในตัวอย่างที่ 1 จงตรวจสอบว่า M ยอมรับสายอักขระ aabba และ abbba หรือไม่

วิธีทำ สำหรับสายอักขระ aabba :

$(q_0, aabba) \vdash (q_0, abba)$ เนื่องจาก $\delta(q_0, a) = q_0$

$\vdash (q_0, bba)$ เนื่องจาก $\delta(q_0, a) = q_0$

$\vdash (q_1, ba)$ เนื่องจาก $\delta(q_0, b) = q_1$

$\vdash (q_0, a)$ เนื่องจาก $\delta(q_1, b) = q_0$

$\vdash (q_0, \varepsilon)$ เนื่องจาก $\delta(q_0, a) = q_0$

ดังนั้น $(q_0, aabba) \vdash^* (q_0, \varepsilon)$ และเนื่องจาก $q_0 \in F$ จึงสรุปได้ว่า M ยอมรับ aabba

สำหรับสายอักขระ abbba จะได้ว่า

$(q_0, abbba) \vdash (q_0, bbba)$

$\vdash (q_1, bba)$

$\vdash (q_0, ba)$

$\vdash (q_1, a)$

$\vdash (q_1, \varepsilon)$

ดังนั้น $(q_0, abbba) \vdash^* (q_1, \varepsilon)$ และเนื่องจาก $q_1 \notin F$ จึงสรุปได้ว่า M ไม่ยอมรับ abbba

ตัวอย่างที่ 3 จงสร้าง DFA ของ M ที่ยอมรับสายอักขระ x $\in$ $\{0, 1\}^*$ ที่มี 0 ปรากฏอยู่ทั้งหมดเป็นจำนวนคู่ครั้ง และ 1 ปรากฏอยู่ทั้งหมดเป็นจำนวนคี่ครั้ง

วิธีทำ ให้ $M = (Q, \Sigma, \delta, q_0, F)$ เป็น DFA ที่มี $Q = \{ q_0, q_1, q_2, q_3\}$ โดยที่

q_0 แทน สถานะที่อ่าน 0 ไปแล้วจำนวนคู่ตัว และอ่าน 1 ไปแล้วจำนวนคู่ตัว

q_1 แทน สถานะที่อ่าน 0 ไปแล้วจำนวนคี่ตัว และอ่าน 1 ไปแล้วจำนวนคู่ตัว

q_2 แทน สถานะที่อ่าน 0 ไปแล้วจำนวนคู่ตัว และอ่าน 1 ไปแล้วจำนวนคี่ตัว

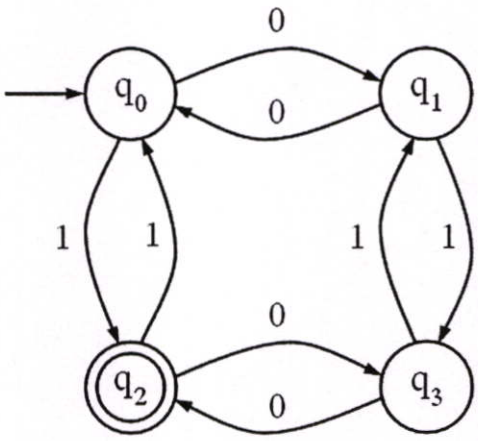
q_3 แทน สถานะที่อ่าน 0 ไปแล้วจำนวนคี่ตัว และอ่าน 1 ไปแล้วจำนวนคี่ตัว

$F = \{q_2\}$, $\Sigma = \{0, 1\}$ และ $\delta : Q \times \Sigma \to Q$ นิยามได้ดังนี้

$$\delta(q_0, 0) = q_1 , \ \delta(q_0, 1) = q_2 , \ \delta(q_1, 0) = q_0 , \ \delta(q_1, 1) = q_3$$

$$\delta(q_2, 0) = q_3 , \ \delta(q_2, 1) = q_0 , \ \delta(q_3, 0) = q_2 , \ \delta(q_3, 1) = q_1$$

ซึ่งสามารถเขียนเป็น transition diagram ของ DFA เครื่องนี้ได้ ตามรูปที่ 3



รูปที่ 3 transition diagram จากการสร้าง DFA ในตัวอย่างที่ 3

จะเห็นได้ว่า DFA ที่สร้างขึ้นมาเครื่องนี้ ยอมรับเฉพาะสายอักขระ x $\in$ $\{0, 1\}^*$ ที่มี 0 ปรากฏอยู่ทั้งหมดจำนวนคู่ครั้ง และ 1 ปรากฏอยู่เป็นจำนวนคี่ครั้งเท่านั้น

ภาคผนวก ข.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. Samasant Suwannit, Boontee Kruatachue, Kritawan Siriboon, **"DFA inducton of chain code sequence for on-line Thai handwritten recognition,"** Proceeding of 2006 Electrical Engineering/ Electronics Computer Telecommunications and Information Technology (ECTI) International Conference (ECTI-CON 2006), Ubon Ratchatani, Thailand, pp. 697-700, May 10-13, 2006.


ECTI-CON 2006
THE 2006 ECTI INTERNATIONAL CONFERENCE
ECTI
Association
ECTI
2006
Enter to the Proceeding
Proceedings of the 2006 Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) International Conference
May 10-13, 2006
Ubonburi Hotel, Ubon Ratchathani, THAILAND
ECTI Association
IEEE-LEOS Thailand Chapter
IEEE MTT/AP/ED THAILAND CHAPTER
NECTEC
Seagate
IEEE CAS Society
LEOS

DFA induction of chain code sequence for on-line Thai handwritten recognition

Department of Computer Engineering, Faculty of Engineering, School of Graduate Studies
King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand

Samasant Suwannit E-mail: samasant@hotmail.com
Boontee Kruatachue E-mail: kkboonte@kmitl.ac.th
Kritawan Siriboon E-mail: kskritaw@kmitl.ac.th

ABSTRACT

This paper proposed method to automatically generate the Deterministic Finite Automata (DFA) to learn strings of chain code. This DFA will take into account the length and relative position of the occurred chain code sequence in order to merge similar nodes in DFA. Since the DFA are generated for recognition task, the DFAs must be generalized enough to accept the chain code sequence of the same character and not to generalized to accept other written-alike characters. The DFA is first generated from a training chain code sequence and further refine from all other sequence in the training set. When test recognition we use flexible value +1,-1 and +2,-2 in order to change chain code to near direction for more generalized.

Keywords: Deterministic Finite Automata (DFA), handwritten recognition

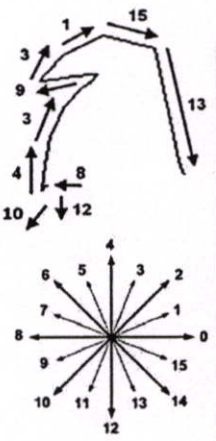
1. INTRODUCTION

In human handwriting, people write the same alphabet differently but we still recognize which character it is. In this research, we represent handwriting as a string of 16 directions alphabet (0-15 chain code like direction), length and relative position of the occurred chain code sequence. We implement each character model as Deterministic Finite Automata (DFA). So our main problem is to automatically construct the DFA for each character that fully represent the set of training string of that character. The restriction of this DFA 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.

2. CHAIN CODE, LENGTH AND POSITION

Online handwritten data is in the form of sequence of the (x,y,t) coordinates which are sampling on an electronic tablet that traces the movement of the pen. As the sampling rate of a tablet is fixed, the distance between sampling points is equal in time but not in length. Then the chain code is used to quantize the change in direction between each sampling point. The length of each chain code are the Euclidean distance between the 2 sampling points and normalize by the sum of all the distance between all sampling. The relative position of chain code in the letter is the summation of the normalized length as shown in Fig. 1.

The selected Thai characters with one-stroke used in the experiment are 37 characters as in Fig. 2.



chain code	%length	%position
8	0.71	0.71
12	0.71	1.43
10	0.50	1.93
4	1.07	3.00
3	1.13	4.13
4	1.07	5.20
3	21.51	26.72
1	2.57	29.29
3	1.78	31.07
8	1.43	32.50
9	10.80	43.28
3	8.09	51.37
1	9.09	60.46
15	11.07	71.52
13	28.48	100.00

Fig. 1: Sample character ก with chain code, length and position

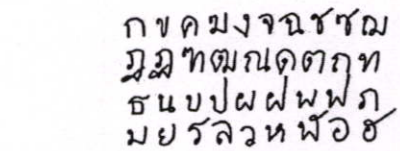


Fig. 2: Thai one-stroke 37 characters

3. DETERMINISTIC FINITE AUTOMATA (DFA)

In the theory of computation, a deterministic finite state machine or deterministic finite automaton (DFA) is a finite state machine where for each pair of state and input symbol there is one and only one transition to a next state. DFAs recognize the set of regular languages and no other languages.

A DFA will take in a string of input symbols. For each input symbol it will then transition to a state given by following a transition function. When the last input symbol has received it will either accept or reject the string depending on if it is in an accepting state or not.

3.1 Formal definition

A DFA is a 5-tuple, (S, Σ, T, s, A), consisting of

- a finite set of states (S)
- a finite set called the alphabet (Σ)
- a transition function ($T : S \times \Sigma \rightarrow S$)
- a start state ($s \in S$)
- a set of accept states ($A \subseteq S$)

Let M be a DFA such that M = (S, Σ, T, s, A), and $X = x_0x_1 \ldots x_n$ be a string over the alphabet Σ. M accepts the string X if a sequence of states, $r_0, r_1, \ldots, r_n$, exists in S with the following conditions:

1. $r_0 = s$
2. $r_{i+1} = T(r_i, x_i)$, for $i = 0, \ldots, n-1$
3. $r_n \in A$.

As shown in the first condition, the machine starts in the start state s. The second condition says that given each character of string X, the machine will transition from state to state as ruled by the transition function T. The last condition says that the machine accepts if the last input of X causes the machine to be in one of the accepting states. Otherwise, it is said to reject the string. The set of strings it accepts form a language, which is the language the DFA recognizes.

3.2 Example

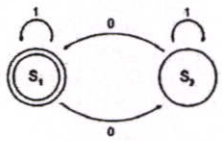
The following example is of a DFA M, with a binary alphabet, which determines if the input contains an even number of 0s.

$M = (S, \Sigma, T, s, A)$ where

- $\Sigma = \{0, 1\}$,
- $S = \{S_1, S_2\}$,
- $s = S_1$,
- $A = \{S_1\}$, and
- T is defined by the following state transition table:

Input / State	1	0
S_1	S_1	S_2
S_2	S_2	S_1

And the state diagram for M is:



Simply put, the state S_1 represents that there has been an even number of 0s in the input so far, while S_2 signifies an odd number. A 1 in the input does not change the state of the automaton. When the input ends, the state will show whether the input contained an even number of 0s or not.

The language of M can be described by the regular language given by this regular expression: 1*(01*01*)*

4. SYSTEM OVERVIEW

The proposed system has 3 main steps is shown in Fig. 3. We used chain code, length and position sequences of the training set to construct each character DFA. First the initial DFA is constructed from the first sequence in the training. Then, we try to make the DFA more generalized by refining this initial DFA using the rest of samples. Each refinement accumulated into the final DFA. Last, we test this DFA by parsing unknown sequence of string.

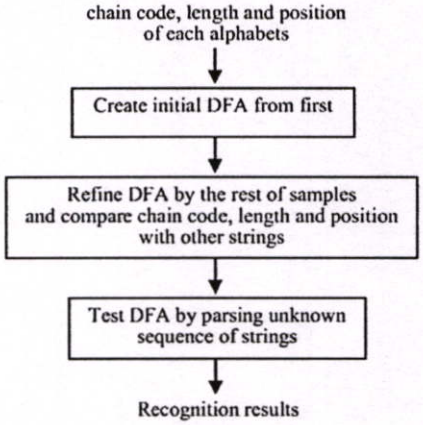


Fig. 3: System overview

5. PROCESS OF CONSTRUCTION DFA

The DFA induction example is shown in Fig. 4. The DFA model n is trained from 2 strings, n-1 and n-2.

At the first time, chain code of the first string (n-1) is used to construct DFA beginning from the starting state (S0) to final state (F). Each chain code becomes a node in the DFA. The node transition is set in the same order as the chain code occurring in the sequence. Each node keeps the following information chain code, length and position.

Next, chain code, length and position of the second string (n-2) is used to refine the initial DFA by merging input chain code with the existing node that has the same chain code, length and position. The length and position are allowed with in certain threshold (threshold_length = 10%, threshold_position = 5%). The length and position of the merge node are update by averaging with the current input, that shown on equations (1) and (2). The unmergable chain code (different chain code, length and/or position) becomes a new node and inserted into the existing DFA.

$$AvgLength = \frac{\sum_{n} Length_n}{n} \quad (1)$$

$$AvgPosition = \frac{\sum_{n} Position_n}{n} \quad (2)$$

In Fig. 4, the highlight is the value of chain code, length and position of shared node. That means the features of character n-1 and n-2 at same position in that node are similar. If the features of node are not similar, the new node will be created.

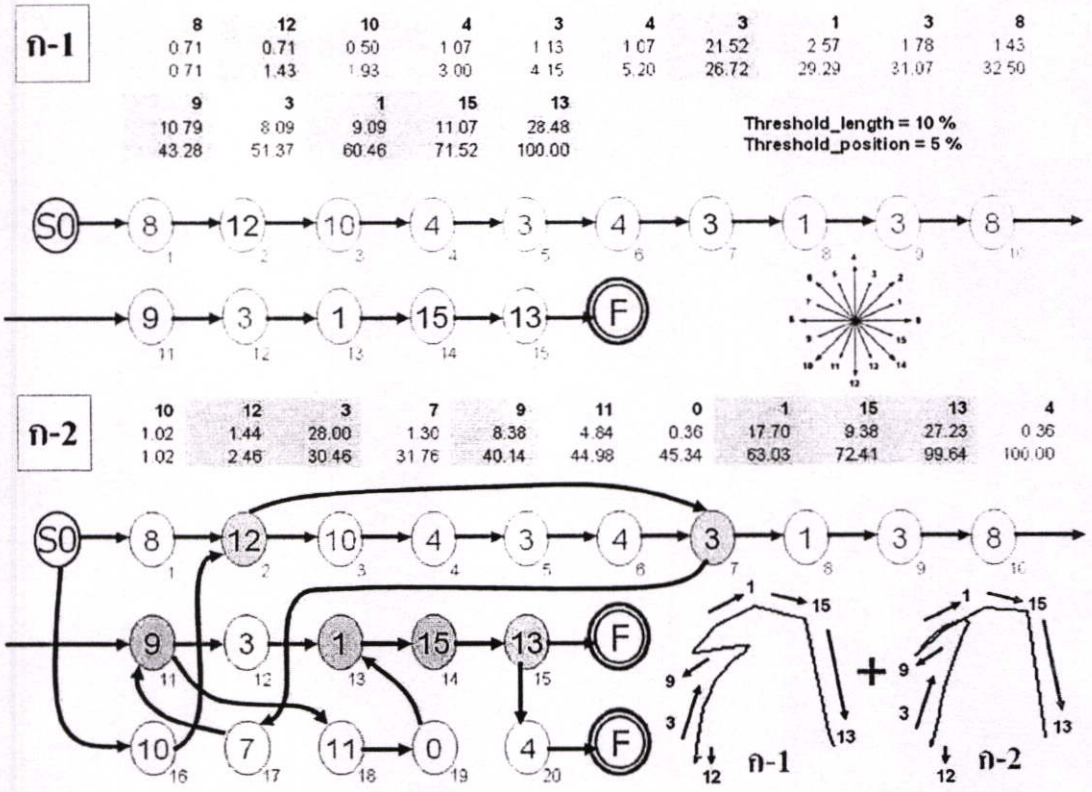


Fig. 4: Construction DFA model ก of ก-1 and ก-1 + ก-2

6. RESULTS AND DISCUSSION

In this section, we describe the experimental result. We use the training set consist of 37 letters with the total 4826 characters and the testing set are 2469 characters. In the recognition process, an unknown handwritten chain code string is parsed with all DFA for each character. If the parse sting reach the final state, that handwritten is classified as the model character (recognition rate). In case that more than one DFA reach final state, the handwritten is un-classified (accept more than one DFA). If no DFA reach final state or accept another DFA (a few case), the handwritten is unknown (no accept rate).

For more generalize, we parsing to reach final state by real direction (Real recog) and near direction of chain code. We change chain code to near direction by flexible value for test recognition. The flexible value is +1,-1 and +2,-2. For example, if the next sequence is chain code 7 and the next states in DFA have chain code 6 or 8. We will parse to state with chain code 6 because 6+1=7. If it reaches final state the DFA accept the sequence by Recog +1,-1. Otherwise, we try on the next state with chain code 8 (8-1=7). If it does not reach final state, then it is unaccepted by this model (no accept rate).

The recognition results are shown in Table 1. The recognition rate is 71.83% with 11.69% accept more than one DFA and 16.48% are no accept rate. From 16.48% of no accept rate we perform recognition again with the more generalize parsing allow +1,-1 in each state chain code. From this 16.48%, we have recognition rate of 7.45%, 2.91% accept more than one DFA and 6.12% no accept rate. We repeat more generalize parsing with +2,-2 from 6.12% of no accept rate, we have recognition rate of 3.84%, 0.69% accept more than one DFA and 1.59% no accept rate. So the effective recognition rate is 71.83% + 7.45% + 3.84% = 83.12%.

***Table 1:** Recognition results*

	Recognition rate	Accept more than one DFA	No accept rate
Real recog	71.83 %	11.69 %	16.48 %
Recog +1,-1	7.45 %	2.91 %	6.12 %
Recog +2,-2	3.84 %	0.69 %	1.59 %
Total recog	83.12 %		

7. CONCLUSIONS

We proposed a way to automatically induction DFA from the chain code strings. The DFA learn grammar of the handwritten character. We try to make the DFA generalized but specific enough for character recognition. The experiment indicates that model is too specific so we made them more generalized by allow "near" symbol parsing for unknown cases.

8. REFERENCES

[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.

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

[3] M. Okamoto, K. Yamamoto, "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, pp. 491-4941, September 20-22, 1999.

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

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

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

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

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

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

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

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

[12] 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, pp. 605-608, 19-22 April 1994.

[13] 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, pp. 107-110, 20-22 September 1999.

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

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

ประวัติผู้เขียน

ชื่อ-นามสกุล	นาย สมาสันต์ สุวรรณนิตย์
วันเดือนปีเกิด	วันศุกร์ที่ 11 เดือน มีนาคม พ.ศ. 2520
ประวัติการศึกษา	พ.ศ. 2543 สำเร็จการศึกษาระดับปริญญาตรี หลักสูตร วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์) จาก คณะสารสนเทศศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยศรีปทุม