การศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก

นาย สนธยา เมรินทร์

วิทยานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
ภาควิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย
พ.ศ. 2537

ISBN 974-584-382-2

A STUDY ON RECOGNITION OF PRINTED THAI CHARACTERS BY THE SYNTACTIC METHOD

Mr. Sonthaya Menin

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering
Department of Electrical Engineering
Graduate School
Chulalongkorn Universty
1994

หัวข้อวิทยานิพนธ์ การศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก

โดย นาย สนธยา เมรินทร์

ภาควิชา วิศวกรรมไฟฟ้า

อาจารย์ที่ปรึกษา รศ. ดร. สมชาย จิตะพันธ์กุล

บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีบัณฑิตวิทยาลัย

(ศาสตราจารย์ ดร. ถาวร วัชราภัย)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร. ณรงค์ อยู่ถนอม)

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. สมชาย จิตะพันธ์กุล)

..... กรรมการ

(อาจารย์ ดร. วาทิต เบญจพลกุล)

..... กรรมการ

(ดร. บวรกูล จิตต์ประเสริฐ)

พิมพ์ต้นฉบับบทคัดย่อวิทยานิพนธ์ภายในกรอบสีเขียวนี้เพียงแผ่นเดียว

สนธยา เมรินทร์ : การศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก (A STUDY ON RECOGNITION OF PRINTED THAI CHARACTERS BY THE SYNTACTIC METHOD)
อ.ที่ปรึกษา : รศ. ดร. สมชาย จิตะพันธ์กุล, 66 หน้า. ISBN 974-584-382-2.

การวิจัยครั้งนี้มีจุดมุ่งหมาย เพื่อศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก วิธีที่ใช้แบ่งออกเป็น 2 ขั้นตอนคือ การวิเคราะห์โครงร่างของต้นไม้ และการวิเคราะห์ทาง feature หลังจากที่ข้อมูลภาพถูกทำให้เป็นเวกเตอร์เรียบร้อยแล้วจะถูกนำไปทำการรู้จำ ขั้นตอนแรกเป็นการจำแนกขั้นต้นประกอบด้วยการแปลงเวกเตอร์ให้เป็นต้นไม้ของ primitive, การวัดค่าระยะระหว่างต้นไม้ของตัวอักษรที่ต้องการรู้จำกับตัวอักษรต้นแบบ โดยเลือกเปรียบเทียบเฉพาะตัวอักษรต้นแบบที่มีหัวของตัวอักษรอยู่ในบริเวณเดียวกับหัวของตัวอักษรที่ต้องการรู้จำเท่านั้น ส่วนขั้นตอนหลังเป็นการจำแนกโดยละเอียด โดยนำเอาลักษณะเด่นของตัวอักษรมาวิเคราะห์เพื่อเพิ่มความถูกต้องของการรู้จำให้มากขึ้น หากผลการรู้จำไม่อยู่ในเกณฑ์ที่ยอมรับได้ เวกเตอร์ของตัวอักษรจะถูกนำไปปรับปรุงเพื่อตัดส่วนเกินออก หรือต่อส่วนขาดของตัวอักษรเข้าด้วยกัน จากนั้นจึงถูกนำไปรู้จำโดยวิธีเดิมอีกจนกว่าผลการรู้จำจะอยู่ในเกณฑ์ที่น่าพอใจ หรือจนกว่าไม่สามารถปรับปรุงเวกเตอร์ได้อีก ผลการรู้จำที่ได้มีค่าประมาณ 97% จากจำนวนตัวอักษรที่ทำการทดสอบ 966 ตัว (ใช้เป็นตัวอักษรต้นแบบ 101 ตัว) ใช้เวลาในการประมวลผลโดยเฉลี่ย 1.09 วินาทีต่อตัวอักษร บนเครื่อง IBM-PC 486sx ที่ความเร็ว 33 เมกกะเฮิรตซ์

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....
สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2536 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

C515703 : MAJOR ELECTRICAL ENGINEER
KEY WORD: RECOGINITION / SYNTACTICIC METHOD / TREE GRAMMAR / THAI CHARACTERS

SONTHAYA MERIN : A STUDY ON RECOGNITION OF PRINTED THAI CHARACTERS BY THE SYNTACTIC METHOD. THESIS ADVISOR : ASST. PROF. SOMCHAI JITAPUNKUL, Dr.Ing. 66 pp. ISBN 974-584-382-2.

This study proposes a method for recognition of Thai printed characters by the syntactic method. The recognition method combines techniques of syntactic tree structure analysis and global feature analysis. After vectorization, the former method is used for rough classification, including vector to postfix tree of primitive transformation and the distance between tree of input character and template character that the head of character addresses in the same zone as input character. Next, the latter method is used for fine classification. Dominant characteristics of character are analyzed to increase accuracy of recognition. If a character is not recognized, the vector is enhanced by using the joint of two closely broken lines or the cutting of shortest line of input character. Then the recognition procedure is processed again until result of recognition is accepted otherwise it is rejected. The recognition rate is about 97% of 966 test characters, 101 characters are used for templates, and average recognition time is 1.09 second per character on the IBM-PC 486sx, 33 MHz clock speed machine.

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....
สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2536 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ได้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือเป็นอย่างดียิ่งของ รองศาสตราจารย์ ดร. สมชาย จิตะพันธ์กุล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่างๆ ของการวิจัยมาโดยตลอด ขอขอบคุณ คุณรานีพร จันทร์เจริญ ที่ได้ให้ยืมเครื่องคอมพิวเตอร์สำหรับใช้ในการพัฒนาโปรแกรม และฝ่ายระบบสารสนเทศฯ บริษัทเงินทุนหลักทรัพย์ ธนชาติ (มหาชน) จำกัด ที่ได้ให้ใช้เครื่องคอมพิวเตอร์และเครื่องสแกนเนอร์เพื่อใช้ในการเก็บตัวอย่างตัวอักษรและทดสอบโปรแกรม ตลอดจนเพื่อนและน้องๆ ทุกคนที่คอยให้กำลังใจมาโดยตลอด

ท้ายนี้ ผู้วิจัยใคร่ขอกราบขอบพระคุณ บิดา-มารดา ซึ่งได้ให้การสนับสนุนและเป็นกำลังใจแก่ผู้วิจัยจนสำเร็จการศึกษา

สารบัญ

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

ความเป็นมาและความสำคัญของปัญหา

การรู้จำตัวอักษรด้วยคอมพิวเตอร์ เริ่มเข้ามามีบทบาททั้งในชีวิตประจำวัน และในเชิงธุรกิจเป็นอย่างมาก ปัจจุบันเริ่มมีผลิตภัณฑ์คอมพิวเตอร์ที่สามารถอ่านข้อความจากเอกสาร หรือสามารถรับรู้ข้อมูลจากการเขียนที่จอภาพโดยตรง ทำให้มนุษย์ไม่จำเป็นต้องป้อนข้อมูลผ่านทางแป้นพิมพ์อย่างเดียวอีกต่อไป แนวโน้มผลิตภัณฑ์ประเภทนี้สามารถนำมาใช้แทนสมุดบันทึก, พจนานุกรมที่สามารถพกได้โดยสะดวก

K. S. Fu [1],[2] ได้จำแนกวิธีการทางคณิตศาสตร์ที่ใช้เพื่อการรู้จำออกเป็น 2 ประเภทใหญ่ๆ คือ การวิเคราะห์ทาง statistical และ วิธีการทางซินแทกติก

1. การวิเคราะห์ทาง statistical เป็นการวัดและเปรียบเทียบทาง feature เช่น การเปรียบเทียบทางตำแหน่ง, การเปรียบเทียบจุดตัด, จุดแยกของภาพ หรือ การเปรียบเทียบความเข้มของภาพ เป็นต้น แล้วนำข้อมูลเหล่านั้นมาทำการวิเคราะห์และเปรียบเทียบทางสถิติ วิธีการนี้มีข้อจำกัดที่ไม่สามารถแยกแยะรูปภาพที่มีโครงสร้างที่ซับซ้อน และมีส่วนประกอบเป็นจำนวนมากออกจากกันได้อย่างชัดเจน

2. วิธีการทางซินแทกติก วิธีการนี้สามารถเปลี่ยนโครงร่างของภาพที่มีลักษณะซับซ้อนให้อยู่ในรูปของ primitive และใช้ไวยกรณ์ที่กำหนดขึ้นสำหรับใช้อธิบายการเชื่อมต่อ primitive เหล่านั้น ทำให้อธิบายภาพที่มีโครงสร้างซับซ้อนได้อย่างชัดเจนมากยิ่งขึ้น

จากแนวคิดการรู้จำที่กล่าวมา ได้มีผู้นำหลักการดังกล่าวไปประยุกต์ใช้ ควบคู่ไปกับแนวคิดของตนเอง งานวิจัยการรู้จำตัวอักษร และตัวเลขเท่าที่พบก็มี

Shin-Yee Lu [3] ได้นำเสนอวิธีการเปรียบเทียบเพื่อหาค่าระยะระหว่างต้นไม้ของ primitive Shin ได้เสนอการคำนวนแบบ ไดนามิกโปรแกรมมิ่ง เพื่อคำนวนหาค่าระยะที่น้อยที่สุดในการแปลงจาก ต้นไม้ของ primitive ต้นหนึ่งไปยังอีกต้นหนึ่ง ค่าที่ได้ก็คือค่าความแตกต่างระหว่างต้นไม้ทั้งสองนั่นเอง

Louisa Lam and Ching Y. Suen [4] ได้เสนอการรู้จำตัวเลขอารบิกแบบลายมือเขียนของรหัสไปรษณีย์ที่ปรากฏบนซองจดหมาย วิธีที่ใช้แบ่งออกเป็น 2 วิธีการหลักคือ การวิเคราะห์ทางด้านโครงสร้าง เพื่อให้สามารถเปรียบเทียบและรู้จำตัวเลขให้ได้ไวที่สุด แต่ถ้าไม่สามารถรู้จำได้ก็จะใช้วิธีที่สอง คือ การเปรียบเทียบแบบ relaxation เพื่อนำไปคำนวนเปรียบเทียบกับรูปแบบของตัวเลขที่เป็นต้นแบบ โดยใช้วิธีการคำนวนเพื่อหาค่าระยะที่น้อยที่สุด

สุรพันธ์ [5] เสนอวิธีการรู้จำตัวอักษรลายมือเขียนภาษาไทย โดยการพิจารณาหัวของตัวอักษร วิธีนี้ตัวอักษรจะถูกแบ่งออกเป็นกลุ่มย่อย โดยที่แต่ละกลุ่มจะเป็นตัวอักษรที่มีหัวอยู่ในบริเวณเดียวกัน จากนั้นแต่ละกลุ่มก็จะมีวิธีการพิจารณาเปรียบเทียบตัวอักษรที่อยู่ในกลุ่มของตนเองแตกต่างกันไป ทั้งนี้ขึ้นอยู่กับลักษณะเด่นของตัวอักษรที่อยู่ในกลุ่มนั้นๆ ข้อดีของการเปรียบเทียบแบบนี้คือทำการเปรียบเทียบได้อย่างรวดเร็ว แต่การนำไปประยุกต์ใช้งานกับตัวอักษรประเภทอื่นๆต้องมีการปรับปรุงโปรแกรมบางส่วน เพื่อให้สามารถรับเงื่อนไขของตัวอักษรใหม่ๆได้

พิพัฒน์ และมนลดา [6] ได้นำเสนอการรู้จำตัวอักขระไทยหลายรู้แบบโดยวิธีไดนามิกโปรแกรมมิ่ง วิธีการที่ใช้เป็นการวิเคราะห์เส้นแสดงขอบของตัวอักษร เพื่อหาส่วนโค้งที่มีทั้งเว้าและนูนของตัวอักษร แล้วนำไปใช้ในการเปรียบเทียบแบบไดนามิกโปรแกรมมิ่งกับของตัวอักษรต้นแบบ เพื่อหาค่าความคล้ายของตัวอักษร

ในการศึกษาครั้งนี้ เนื่องจากตัวอักษรภาษาไทยมีโครงสร้างเป็นเส้นตรงผสมเส้นโค้งและวงกลม บางครั้งมีการตัดกันของเส้นทำให้มีโครงสร้างที่ค่อนข้างซับซ้อน การรู้จำโดยวิธีซินแทกติกค่อนข้างเหมาะกับการนำมาใช้กับตัวอักษรภาษาไทย เพราะวิธีการซินแทกติกเป็นการพิจารณาที่โครงสร้างของตัวอักษรเท่านั้น โดยมีการอธิบายโครงสร้างของตัวอักษรในรูปของประโยคที่ประกอบด้วย primitive ทำให้สามารถจำแนกตัวอักษรที่มีโครงสร้างที่แตกต่างกันออกจากกันได้อย่างง่ายและรวดเร็ว แต่วิธีการทางซินแทกติก เพียงวิธีการเดียวไม่สามารถจำแนกได้อย่างชัดเจนนัก เนื่องจากตัวอักษรภาษาไทยบางกลุ่มมีลักษณะที่คล้ายคลึงกันมาก จึงจำเป็นต้องอาศัยวิธีการทางเปรียบเทียบทาง feature ควบคู่กันไป เพื่อให้ได้ผลการรู้จำที่ถูกต้องมากที่สุด

แนวคิดและทฤษฎี

1. ตัวอักษรภาษาไทย เนื่องจากเสียงในภาษาไทยมี 3 ชนิด คือ เสียงแท้ เสียงแปร และเสียงดนตรี ดังนั้นตัวอักษรภาษาไทยที่ใช้สำหรับแทนเสียงในภาษาไทยทั้ง 3 ชนิดได้แก่ ตัวสระ ตัวพยัญชนะ และ ตัววรรณยุกต์ ดังปรากฏในหนังสือของ ยศ [7]

1.1 สระ ภาษาไทยมีรูปสระทั้งหมด 21 รูป มีชื่อเรียกดังต่อไปนี้

1.1.1 ะ เรียกว่า วิสรรชนีย์ ใช้เขียนข้างหลังพยัญชนะ เช่น นะคะ และประสมกับสระรูปอื่น เช่น เอะ โอะ อัวะ

1.1.2 อั เรียก ไม้ผัด หรือ ไม้หันอากาศ ใช้เขียนบนพยัญชนะแทนวิสรรชนีย์ในคำที่มีตัวสะกด เช่น ป+ะ+น = ปัน

1.1.3 อ็ เรียก ไม้ไต่คู้ ใช้เขียนบนพยัญชนะแทนวิสรรชนีย์ ในสระเสียงสั้นบางคำที่มีตัวสะกด เช่น เปะ+น = เป็น และผสมกับ ก เป็น ก็

1.1.4 า เรียก ลากข้าง ใช้เขียนข้างหลังพยัญชนะออกเสียงเป็น อา เช่น กา และประสมกับสระรูปอื่นๆ เช่น เอาะ อำ

1.1.5 อิ เรียก พันทุ์อิ ใช้เขียนบนพยัญชนะออกเสียงเป็น อิ เช่น ติ และประสมกับสระรูปอื่นๆ เช่น อี อึ เอีย หรือใช้แทนตัว อ ของสระ เออ เมื่อมีตัวสะกดเป็น เกอน เช่น เกอน = เกิน

1.1.6 ่ เรียก ฝนทอง ใช้เขียนบนพันทุ์อิ เป็นสระอี เช่น ตี และประสมกับสระรูปอื่นๆ เช่น เกียะ

1.1.7 ° เรียก นฤคหิต หรือ หยาดน้ำค้าง ใช้เขียนบนพันทุ์อิ เป็นสระ อึ และใช้เขียนบนลากข้างเป็น สระอำ เช่น ตำ

1.1.8 ๊ เรียก ฟันหนู ใช้เขียนบนพันทุ์อิ เป็นสระ อึ เช่น มือ และประสมกับสระรูปอื่นๆ เช่น เอือ

1.1.9 อุ เรียก ตีนเหยียด ใช้เขียนข้างล่างพยัญชนะ ออกเสียงเป็น อุ เช่น ดุ

1.1.10 อู เรียก ตีนคู้ ใช้เขียนข้างล่างพยัญชนะ ออกเสียงเป็น อู เช่น ดู

1.1.11 เ เรียก ไม้หน้า ใช้เขียนข้างหน้าพยัญชนะ รูปเดี่ยวเป็น สระเอ ออกเสียงเป็น เอ ถ้ามีสองรูปเป็นสระ แอ ออกเสียงเป็น แอ

1.1.12 ไ เรียก ไม้ม้วน ใช้เขียนข้างหน้าพยัญชนะ ออกเสียงเป็น ไอ ในภาษาไทยมีคำที่ใช้สระไออยู่ 20 คำ

1.1.13 ไ เรียก ไม้มลาย ใช้เขียนข้างหน้าพยัญชนะ ออกเสียงเป็น ไอ

1.1.14 โ เรียก ไม้โอ ใช้เขียนข้างหน้าพยัญชนะ ออกเสียงเป็น โอ เช่น โก่

1.1.15 อ เรียก ตัวออ ใช้เขียนข้างหลังพยัญชนะ ออกเสียงเป็น ออ เช่น งอ และประสมกับสระรูปอื่นๆ เช่น เออะ สื่อ

1.1.16 ย เรียก ตัวยอ ใช้เขียนประสมกับสระรูปอื่นๆ เช่น เอียะ เอีย

1.1.17 ว เรียก ตัววอ ใช้เขียนประสมกับสระรูปอื่นๆ เช่น อัวะ

1.1.18 ฤ เรียก ตัวรึ ใช้เขียนประสมกับพยัญชนะออกเสียงเป็น ริ รึ เรอ เช่น ตฤณ ทฤษฎี พฤกษ์ และใช้เขียนต้นคำออกเสียงเป็น รึ เช่น ฤดู ฤดี

1.1.19 ฦ เรียก ตัวรือ ใช้เขียนต้นคำเช่น ฦๅษี นอกนั้นใช้เฉพาะการแต่งคำประพันธ์ เช่น ฦๅพี่ ดังฦๅ

1.1.20 ฤ เรียก ตัวลึ ปัจจุบันเลิกใช้แล้ว

1.1.21 ฦ เรียก ตัวลือ ปัจจุบันเลิกใช้แล้ว

1.2 พยัญชนะ พยัญชนะไทยมีทั้งหมด 44 ตัว ถ้าจัดเป็นวรรค จัดได้ดังนี้

1.2.1 วรรคที่ 1 ก ข ฃ ค ฅ ฆ ง

1.2.1 วรรคที่ 2 จ ฉ ช ซ ฌ ญ

1.2.1 วรรคที่ 3 ฏ ฏ ฐ ฑ ฒ ณ

1.2.1 วรรคที่ 4 ด ต ถ ท ธ น

1.2.1 วรรคที่ 5 บ ป ผ ฝ พ ฟ ภ ม

1.2.1 วรรคที่ 6 ย ร ล ว ศ ษ ส ห ฬ อ ฮ

1.3 วรรณยุกต์ คือ อักษรที่ใช้กำหนดให้คำออกเสียงสูงๆ ต่ำๆ ทั้งนี้เพราะคำไทยมีเสียงสูงต่ำ คล้ายเสียงดนตรี วรรณยุกต์ โดยที่เสียงสูงต่ำนี้จะเป็นเครื่องจำแนกความหมายของคำแต่ละคำ โดยสามารถเขียนได้ 4 รูป ดังนี้

1.3.1 วรรณยุกต์เอก เช่น พี่

1.3.2 วรรณยุกต์โท เช่น น้อง

1.3.3 วรรณยุกต์ตรี เช่น โต๊ะ

1.3.4 วรรณยุกต์จัตวา เช่น โอ๋ ต่อง

1.4 ลักษณะพิเศษของภาษาไทย

1.4.1 ส่วนหัว ตัวอักษรส่วนมากขึ้นต้นด้วยส่วนหัวที่เป็นวงกลมที่มีลักษณะแตกต่างกันซึ่งมี วงกลมหัวเข้า, วงกลมหัวออก หรือวงกลมหัวหยัก (เช่น ป ผ ช) อีกทั้งตำแหน่งของหัวก็มีทั้งด้านบนซ้าย ล่างซ้าย ขวาบน และกึ่งกลางตัวอักษร (เช่น บ ถ ง ด)

1.4.2 ระดับของตัวอักษรภาษาไทย การแบ่งระดับของตำแหน่งของตัวอักษรทำให้การจำแนกตัวอักษรที่มีลักษณะคล้ายกันแต่อยู่ต่างระดับกันทำได้ง่ายขึ้น เช่น ตัวอักษร ข,สระอู และไม้โท ดังนั้น ถ้าสามารถกำหนดเส้นระดับล่างและเส้นระดับของตัวอักษร ก็จะสามารถจำแนกตัวอักษรตามระดับได้ ระดับของตัวอักษรภาษาไทยมี 3 ระดับดังนี้

1.4.2.1 ระดับบน เป็นตัววรรณยุกต์ และสระที่อยู่เหนือพยัญชนะ เช่น ไม้โท, สระอิ เป็นต้น

1.4.2.2 ระดับกลาง เป็นตัวพยัญชนะ, ตัวเลข, สัญลักษณ์พิเศษ หรือสระที่มีระดับเดียวกับ พยัญชนะเช่น เ โ ไ ใ นอกจากนี้พยัญชนะบางตัวยังมีบางส่วนของตัวอักษรอยู่นอกระดับกลาง เช่น ป ญ ฐ ฏ

1.4.2.3 ระดับล่าง เป็นตัวสระที่มีตำแหน่งต่ำกว่าพยัญชนะ ซึ่งมีสระอุ และสระอู

2. ไวยกรณ์ของต้นไม้ ในการวิเคราะห์โครงร่างของตัวอักษร ตัวอักษรจะถูกอธิบายให้อยู่ในรูปของประโยค โดยใช้ไวยกรณ์ที่สามารถอธิบายลักษณะทางโครงสร้างของตัวอักษรในรูปของ primitive ซึ่งไวยกรณ์ที่ใช้กันมีอยู่หลายรูปแบบ เช่น Context-free Programmed Grammar [2], Picture Description Language [2] ส่วนวิธีที่เลือกใช้สำหรับการวิจัยนี้เรียกว่า ไวยกรณ์ของต้นไม้

ไวยกรณ์ของต้นไม้มีความสามารถในด้านการอธิบายโครงสร้างที่ซับซ้อนได้ค่อนข้างดี โดยแทนแต่ละ node ของต้นไม้ด้วย primitive และกิ่งของต้นไม้แสดงถึงการเชื่อม primitive เข้าด้วยกัน แต่การใช้ต้นไม้ก็มีข้อเสียที่สำคัญคือ ไม่สามารถอธิบายโครงร่างที่เป็นวงวนได้ เพราะต้นไม้มีลักษณะที่เป็นการแตกกิ่งจากบนลงล่างเท่านั้นไม่สามารถเชื่อมกับ node ที่อยู่นอกเหนือจาก node ที่อยู่ต่ำกว่า ดังนั้นการนำไปใช้งานถ้ามีการเชื่อมโยงในลักษณะวงวน เกิดขึ้นก็ถือเสมือนว่าวงวนเหล่านั้นถูกทำให้ขาดออกจากกัน Shin ได้ให้นิยามของไวยกรณ์ของต้นไม้ ไว้ดังนี้

2.1 นิยาม 1 Tree Domain กำหนดให้ N^+ เป็นเซ็ตของจำนวนเต็มบวก และให้ U เป็น Universal Tree Domain มีลักษณะดังรูปที่ 2 (โดยมีจุดเริ่มต้นที่ 0 และกำหนด " . " เป็นเครื่องหมายการกระทำการแบบไบนารี่) D จะเป็น Tree Domain ก็ต่อเมื่อ D เป็นเซ็ตย่อยที่จำกัดของ U ที่เป็นไปตาม

2.1.1 ถ้า $a.x \in D$ โดยที่ $a \in U$ และ $x \in U$ แล้วถือได้ว่า $a \in D$

2.1.1 ถ้า $a_j \in D$ และ $1 < j$ ใน N^+ แล้ว จะถือได้ว่า $a.1 \in D$

2.1.1 0 เป็นส่วนเริ่มของ D ที่เรียกว่า รากของต้นไม้

รูปที่ 1.1 Universal Tree Domain

2.2 นิยาม 2 การกำหนดชื่อ node การกำหนดชื่อ node ของต้นไม้เป็นไปตาม α:D $\rightarrow \Sigma$ โดย Σ เป็นเซ็ทที่จำกัดของสัญลักษณ์ และให้ N(α) เป็นการระบุจำนวน node ของต้นไม้ α

ตัวอย่างที่ 1 ให้ D = { 0, 1, 2, 1.1, 1.2 } และ Σ = {$, s, t } และ node ของต้นไม้ α เป็น

$\alpha(0)$ = $

$\alpha(1)$ = s

$\alpha(2)$ = t

$\alpha(1.1)$ = t

$\alpha(1.2)$ = t

ดังนั้นจะได้ N(α) มีค่าเป็น 5 ต้นไม้ α D_α

2.3 นิยาม 3 node บน และ node ล่าง ให้ a,b เป็น 2 node ใน D กำหนดให้ a $\leq$ b ก็ต่อเมื่อ a.x = b โดย x $\in$ U แต่ถ้า x=0 แสดงว่า a = b ดังนั้นจะเห็นได้ว่า a < b เมื่อ a เป็น node ที่มาก่อน node b และเรียก b ว่าเป็น node ล่าง

2.4 นิยาม 4 ลำดับที่ของ node ลำดับที่มีรูปแบบ r:$\Sigma \rightarrow$ N โดยที่ N = {0} $\cup$ N^+ ได้เป็น r(a)=n เรียกว่าเป็นลำดับที่ของ a โดยที่ n เป็นจำนวนของ node ล่างของ node a

ตัวอย่างที่ 2 ให้ D มีค่าตามตัวอย่างที่ 1 จะได้ ลำดับของ node ต่างๆ ดังนี้

r(0) = r(1) = 2

r(2) = r(1.1) = r(1.2) = 0

2.5 นิยาม 5 การเรียงตามลำดับ postfix เป็นไปตาม h_α:D $\rightarrow$ N โดยเป็นการกำหนดค่าลำดับของ node ใน D_α ให้เป็นแบบ postfix order

ตัวอย่างที่ 3 ให้ D = { 0, 1, 2, 1.1, 1.2 } ดังนั้น

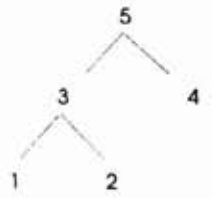
$h\alpha(1.1)$ = 1

$h\alpha(1.2)$ = 2

$h\alpha(2)$ = 3

$h\alpha(1)$ = 4

$h\alpha(0)$ = 5



2.6 นิยาม 6 ต้นไม้ย่อย ให้ α เป็นชื่อของต้นไม้D_α และ $a \in D_\alpha$ ดังนั้น α/a หมายถึงเป็นต้นไม้ย่อยที่ node a ของต้นไม้ α

ตัวอย่างที่ 4 ให้ต้นไม้α เป็นดังที่กำหนดไว้ในตัวอย่างที่ 1

ดังนั้นต้นไม้ย่อย $\alpha/1$ จะได้ดังรูปขวามือ

2.7 นิยาม 7 การแปลงต้นไม้ ให้ α และ β เป็น ต้นไม้ 2 ต้นใด ๆ การแปลง $T:D_\beta \rightarrow D_\alpha$ T จะประกอบด้วย 3 รูปแบบต่อไปนี้ ε , ϕ , และ σ โดย

$$\varepsilon(b) \rightarrow \lambda, p$$

$$\phi(\lambda) \rightarrow a, q$$

$$\sigma(b) \rightarrow a, r$$

โดยที่ $b \in D_\beta$, $a \in D_\alpha$, λ เป็นสัญลักษณ์ว่าง และ p , q , และ r เป็น cost ที่เกิดขึ้นเพื่อใช้ในการแปลงที่สอดคล้องกับ ε , ϕ , and σ ตามลำดับ

การแปลง ε หรือ ϕ จะเป็นผลให้เกิด การตัดออก (deiete) หรือ การแทรกเข้า (insertion) ตามลำดับ ส่วนการแปลง σ จะเป็นผลให้เกิดการแทนที่ (substitution) ในกรณีที่ node ค และ a ไม่เหมือนกัน $[\beta(b) \neq \alpha(a)]$ ถ้า p และ q เป็นค่าใช้จ่ายที่สอดคล้องกับการแปลง ε และ ϕ ตามลำดับ ดังนั้น p และ q จะเป็นค่าใช้จ่ายของการตัดออก และการแทรกเข้า ตามลำดับ เราจะให้ $r>0$ ถ้าการแปลง σ ก่อให้เกิดการแทนและให้ $r = 0$ ในกรณีนอกจากนี้

ให้ T^{-1} แทนการแปลงย้อนกลับของ T, $T^{-1}:D_\alpha \rightarrow D_\beta$ ซึ่งจะได้ว่า $\varepsilon^{-1} = \phi$ และ $\sigma^{-1} = \sigma$ ค่าระยะระหว่างต้นไม้ α และ β เขียนด้วย $d(\alpha,\beta)$ คือ ค่าใช้จ่ายที่น้อยที่สุดที่จำเป็นเพื่อที่จะได้ α

2.8 นิยาม 8 ข้อจำกัดของการคำนวณค่าระยะ การทำการแปลงต้นไม้จาก β ไปเป็น α โดยใช้ชุดของการแปลง T ที่สอดคล้องกับเงื่อนไขต่อไปนี้

2.8.1 ถ้า a < b ใน D_β ดังนั้น T(a) < T(b) และถ้า a และ b ไม่สามารถเปรียบเทียบกันได้ ดังนั้น T(a) และ T(b) ก็จะไม่สามารถเปรียบเทียบกันได้

2.8.2 ถ้า a = b ใน D_β ดังนั้น T(a)=T(b) และถ้า a≠b ดังนั้น T(a) ≠ T(b)

2.8.3 ถ้า $h_\beta(a) < h_\beta(b)$ ดังนั้น $h_\alpha(T(a)) < h_\alpha(T(b))$ โดยมีข้อแม้ว่า T(a) หรือ T(b) ต้องไม่เป็นสัญลักษณ์ null

2.9 ค่าระยะระหว่างต้นไม้ ถ้ากำหนดให้ค่าใช้จ่ายที่สอดคล้องกับการแปลง ε และ ϕ เป็น 1 และ ค่าใช้จ่ายที่สอดคล้องกับการแปลง σ เป็น 1 ถ้ามีการแทนที่เกิดขึ้น และเป็น 0 ในกรณีนอกจากนี้ ดังนั้นค่าระยะจะกลายเป็นผลรวมของ cost ที่เกิดจากการตัดออก, การแทรกเข้า และการแทนที่ที่น้อยที่สุดที่จำเป็นเพื่อที่จะแปลงต้นไม้อันหนึ่งจากไปเป็นต้นไม้อีกอันหนึ่ง

ตัวอย่างที่ 5 กำหนดต้นไม้ 2 ต้นเป็น α และ β

พิจารณาเซตของการแปลง T_1 :

T_1 : $D_\beta \rightarrow D_\alpha$

$\sigma(0) \rightarrow 0, 0$

$\varepsilon(1) \rightarrow \lambda, 1$ (การตัดออก)

$\sigma(1.1) \rightarrow 1.1, 0$

$\sigma(1.2) \rightarrow 2, 0$

$\phi(\lambda) \rightarrow 1, 1$ (การแทรกเข้า)

$\sigma(2) \rightarrow 3, 1$ (การแทนที่)

$\varepsilon(2.1) \rightarrow \lambda, 1$ (การตัดออก)

ให้ cost ของการตัดออก, การแทรกเข้า และการแทนที่ เป็น 1 ทั้งหมด ดังนั้นชุดของ cost ที่น้อยที่สุดที่สอดคล้องกับ เงื่อนไข (1) - (3)

h_β :	(1)		(2)	(3)	(4)	(5)	(6)
D_β :	1.1	λ	1.2	1	2.1	2	0
	↓	↓	↓	↓	↓	↓	↓
h_α :	(1)	(2)	(3)			(4)	(5)
$D_\alpha = T_1(D_\beta)$:	1.1	1	2	λ	λ	3	0
cost :	0	1	0	1	1	1	0

ดังนั้นค่าระยะ $d(\alpha,\beta) = 4$

พิจารณาเซ็ตอีกอันหนึ่ง

T_2 : D_β	→	D_α	
$\sigma(0)$	→	0 . 0	
$\sigma(1)$	→	1.1 . 0	
$\sigma(1.1)$	→	3 . 0	
$\sigma(1.2)$	→	1 . 0	
$\sigma(2)$	→	2 . 1	(การแทนที่)
$\varepsilon(2.1)$	→	λ . 1	(การตัดออก)

แม้ว่าค่าใช้จ่ายของ T_2 จะน้อยกว่าแต่มันไม่สอดคล้องตามเงื่อนไขข้อที่ 1 คือ $T_2(1) = 1.1$ และ $T_2(1.2) = 1$ โดยที่ $T_2(1) > T_2(1.2)$ แต่ $1 < 1.2$

วัตถุประสงค์

1. เพื่อศึกษาวิธีการรู้จำตัวอักษรภาษาไทยโดยวิธีซินแทกติก

2. เพื่อพัฒนาโปรแกรมสำหรับการรู้จำตัวอักษรพิมพ์ภาษาไทย

ขอบเขตของการวิจัย

1. เครื่องคอมพิวเตอร์ที่ใช้ : เครื่อง IBM PC แบบ PS/ValuePoint รุ่น 433SX/S ใช้ไมโครโปรเซสเซอร์ 80486 ความเร็ว 33 เมกกะเฮิรตซ์ มีหน่วยความจำ 4 เมกกะไบท์ และจานแม่เหล็กแบบแข็งขนาด 120 เมกกะไบท์

2. ภาษาคอมพิวเตอร์ที่ใช้ : ภาษา C

3. อุปกรณ์อ่านเอกสาร : เครื่องสแกนเนอร์ Microtek รุ่น ScanMaker II ใช้ความละเอียดของการอ่านเอกสารที่ 300 dpi

4. รูปแบบตัวอักษรที่ใช้ : ตัวอักษรพิมพ์ภาษาไทยชื่อ EUCROSIA และ CORDIA ที่เป็นตัวธรรมดา ไม่เป็นตัวเอียง หรือมีเส้นใต้ เก็บลงไฟล์ 1 ตัวอักษรต่อ 1 ไฟล์ในรูปแบบ BMP หรือ PCX โดยมีตัวอักษรที่สามารถรู้จำได้ 68 ตัวต่อ 1 รูปแบบ (ดังแสดงในภาคผนวก ก และ ข) ดังนี้

4.1 พยัญชนะ 44 ตัว

4.2 สระ 20 ตัว

4.3 วรรณยุกต์ 4 ตัว

5. ขนาดตัวอักษรที่ใช้

5.1 ขนาดของตัวอักษรที่อ่านโดยเครื่องสแกนเนอร์ 20, 22, 24, 28, 32 และ 36 point

5.2 ขนาดของตัวอักษรที่เก็บจากโปรแกรมโดยตรง 48 point

ขั้นตอนและวิธีดำเนินการวิจัย

1. ศึกษาความเป็นไปได้จากผลงานวิจัยที่ผ่านมา
2. กำหนดขอบเขต และเป้าหมายของวิทยานิพนธ์
3. ศึกษาทฤษฎี และเลือกวิธีการที่เหมาะสม
4. พัฒนาโปรแกรม
5. เก็บตัวอย่างตัวอักษรพิมพ์ภาษาไทย
6. ทดสอบและแก้ไขโปรแกรม
7. ประเมินผล

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถรู้จำตัวอักษรพิมพ์ภาษาไทยที่มีรูปแบบและขนาดของตัวอักษรตามที่กำหนดไว้ โดยมีอัตราการรู้จำ 80% ขึ้นไป

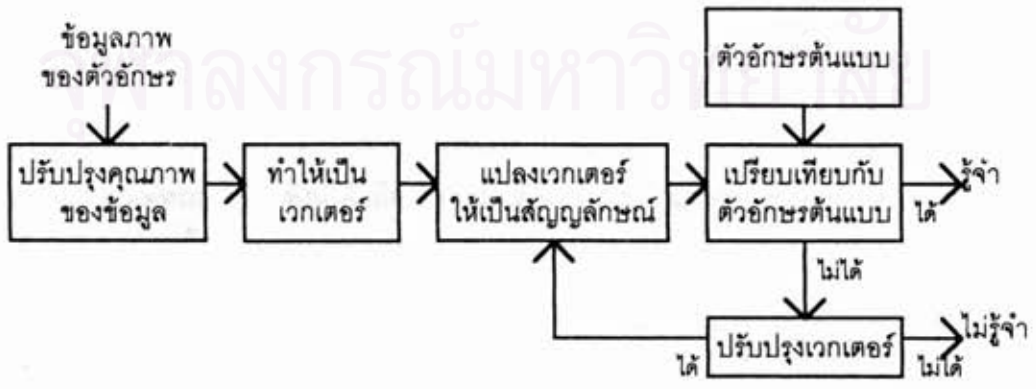
2. สามารถนำไปประยุกต์ใช้กับตัวอักษรพิมพ์ภาษาไทยรูปแบบอื่นๆ

บทที่ 2

การรู้จำโดยวิธีซินแทกติก

โครงสร้างของระบบ

การศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยในครั้งนี้มุ่งเน้นที่ความถูกต้องของการรู้จำเป็นหลัก ดังนั้นวิธีการที่ใช้จึงเป็นวิธีที่ค่อนข้างจะซับซ้อน พิจารณาโครงสร้างของระบบดังแสดงในรูปที่ 2.1 ข้อมูลภาพของตัวอักษรที่รับเข้ามาจะอยู่ในรูปของจุดภาพ จะถูกนำมาปรับปรุงคุณภาพของข้อมูลโดยการกำจัดสัญญาณรบกวน และทำตัวอักษรให้บางเพื่อให้เหลือความกว้างของเส้นเพียง 1 จุดภาพ จากนั้นข้อมูลจุดภาพของตัวอักษรจะถูกนำไปเข้ารหัสเพื่อหาทิศทางของจุดภาพแล้วจึงแปลงให้เป็นเวกเตอร์ โดยนำจุดภาพที่อยู่ติดกันและมีทิศทางไปในทางเดียวกันมารวมเข้าด้วยกันเป็นเส้นตรงที่มีขนาดและทิศทางหนึ่ง เวกเตอร์เหล่านี้จะถูกนำมาเชื่อมโยงกันเป็นรูปต้นไม้ เพื่อให้ยังคงแสดงรูปร่างใกล้เคียงตัวอักษรเดิมที่สุดแต่เวกเตอร์ที่ได้จะมีการเก็บข้อมูลที่มีรายละเอียดค่อนข้างมาก การนำไปใช้เปรียบเทียบกับชุดของตัวอักษรต้นแบบเพื่อรู้จำตัวอักษรจะใช้เวลาในการประมวลผลค่อนข้างมาก ดังนั้นเพื่อการเปรียบเทียบที่รวดเร็วจึงเปลี่ยนเวกเตอร์ให้เหลือเพียงทิศทางของเวกเตอร์เพียงอย่างเดียว เรียกว่า primitive โดย primitive ที่ได้ยังคงมีการเชื่อมโยงเป็นรูปต้นไม้คล้ายกับรูปต้นไม้ของเวกเตอร์ แต่มีการจัดเรียงลำดับ node ของต้นไม้เสียใหม่ตามลำดับ postfix เพื่อให้สามารถนำไปหาค่าระยะระหว่างต้นไม้ได้ และเป็นการทำให้หัวของตัวอักษรเป็นรากของต้นไม้เสมอ



รูปที่ 2.1 โครงสร้างของระบบการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก

แฟ้มของตัวอักษรต้นแบบเป็นการนำต้นไม้ของเวกเตอร์และต้นไม้ของ primitive ของตัวอักษรที่เลือกไว้เป็นต้นแบบเก็บไว้ในแฟ้ม วิธีการรู้จำก็เพียงแต่เลือกเอาเฉพาะต้นไม้ของตัวอักษรต้นแบบที่มีส่วนหัวอยู่ในเซตเดียวกันกับตัวอักษรที่ต้องการรู้จำนำมาเปรียบเทียบกัน ถ้าผลการเปรียบเทียบกับต้นไม้ของตัวอักษรต้นแบบตัวไหนให้ค่าความแตกต่างน้อยที่สุด และมีค่าความแตกต่างที่ต่ำกว่าค่าที่ยอมรับได้ แสดงว่าตัวอักษรที่นำมารู้จำมีความคล้ายกับตัวอักษรต้นแบบนั้นมากที่สุด

ในกรณีที่ผลการเปรียบเทียบไม่เป็นที่น่าพอใจ ต้นไม้ของเวกเตอร์จะถูกนำไปปรับปรุงคุณภาพของเวกเตอร์ โดยการตัดเวกเตอร์ที่มีจุดปลายที่มีความยาวน้อยที่สุด หรือต่อเชื่อมเวกเตอร์ที่มีจุดปลายอยู่ใกล้กันมากๆเข้าด้วยกัน ถ้าสามารถปรับปรุงเวกเตอร์ได้ ต้นไม้ของเวกเตอร์จะถูกนำไปแปลงให้เป็นต้นไม้ของ primitive และนำกลับไปเปรียบเทียบกันตัวอักษรต้นแบบ แต่ถ้าไม่สามารถปรับปรุงเวกเตอร์ได้ แสดงว่าไม่สามารถรู้จำได้ตัวอักษรที่รับเข้ามา

การปรับปรุงคุณภาพของข้อมูลภาพ

ข้อมูลภาพตัวอักษรอาจจะได้มาจากเครื่องถ่ายภาพหรือจากเครื่องสแกนเนอร์ แล้วเก็บเป็นแฟ้มข้อมูลภาพ โดยแฟ้มข้อมูลภาพที่ได้อาจมีรูปแบบการเก็บภาพได้หลายรูปแบบเช่น รูปแบบ BMP, รูปแบบ PCX, รูปแบบ TIF เป็นต้น แต่ในการศึกษาครั้งนี้กำหนดให้รับข้อมูลภาพจากแฟ้มข้อมูลที่มีรูปแบบของข้อมูลภาพแบบสีเดียวเท่านั้น การเก็บข้อมูลภาพแบบนี้จะใช้ 1 บิตข้อมูลต่อ 1 จุดภาพ นั่นคือ 1 จุดภาพจะมีค่าเพียง 2 ค่าเท่านั้น คือ 0 แสดงว่าไม่ใช่จุดภาพ และ 1 แสดงว่าเป็นจุดภาพของตัวอักษร แต่เพื่อให้ง่ายต่อการนำข้อมูลภาพไปใช้งาน จึงต้องเปลี่ยนเป็นข้อมูล 1 ไบต์ต่อ 1 จุดภาพ ถึงแม้ว่าจะต้องใช้หน่วยความจำมากกว่าก็ตาม เพราะการเข้าถึงข้อมูลที่เป็น 1 บิตต่อ 1 จุดภาพนั้นทำได้ยาก และใช้เวลาประมวลผลมากกว่า

ข้อมูลภาพของตัวอักษรที่รับเข้ามาจากเครื่องถ่ายภาพหรือจากเครื่องสแกนเนอร์ มักจะเกิดจุดภาพที่ไม่ต้องการปะปนเข้ามา หรือบางจุดภาพขาดหายไป ซึ่งอาจเกิดจากมีสัญญาณรบกวนปะปนเข้ามากับข้อมูลภาพ หรือเกิดจากประสิทธิภาพของอุปกรณ์เอง ทำให้ข้อมูลภาพไม่สมบูรณ์เหมือนต้นฉบับ ซึ่งอาจมีผลทำให้การรู้จำผิดพลาดได้ ดังนั้นจึงมีความจำเป็นต้องปรับปรุงคุณภาพของข้อมูลภาพของตัวอักษรก่อนเสมอ โดยมีวิธีการดังนี้

1. การกำจัดสัญญาณรบกวน [5] สัญญาณรบกวนที่พบในข้อมูลภาพของตัวอักษร ที่สามารถตรวจพบและกำจัดได้โดยที่ข้อมูลภาพไม่เปลี่ยนรูปร่างไปจากเดิม คือสัญญาณรบกวนที่ทำให้ข้อมูลภาพของตัวอักษรมีลักษณะเป็น จุดเดี่ยวและรูเดี่ยว วิธีการกำจัดสัญญาณรบกวนทั้ง 2 ลักษณะจะใช้วิธีการที่นิยม

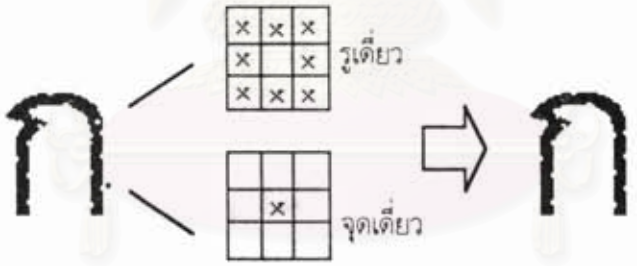
การที่นิยมใช้กันมากๆคือ ใช้ตารางหน้าต่างขนาด 3x3 จุดภาพ (ดังในรูปที่ 2.2) ตรวจสอบกับทุกๆจุดภาพของข้อมูลภาพตัวอักษร โดยที่

1.1 ถ้า X เป็นจุดภาพ และจำนวนจุดข้างเคียงของจุดภาพ X ทั้ง 8 ช่องหน้าต่าง ไม่มีจุดภาพใดๆอยู่เลย แสดงว่าจุด X เป็นจุดเดี่ยว ให้ลบจุดภาพนี้ทิ้งไป

1.2 ถ้า X ไม่ใช่จุดภาพ และจำนวนจุดข้างเคียงของ X ทั้ง 8 ช่องหน้าต่างเป็นจุดภาพทั้ง8 ช่อง แสดงว่าจุด X เป็นรูเดี่ยว ให้เปลี่ยนจุด X เป็นจุดภาพ

x3	x2	x1
x4	X	x0
x5	x6	x7

รูปที่ 2.2 หน้าต่างขนาด 3*3



ก. ก่อนการกำจัดสัญญานรบกวน ข. หลังการกำจัดสัญญานรบกวน

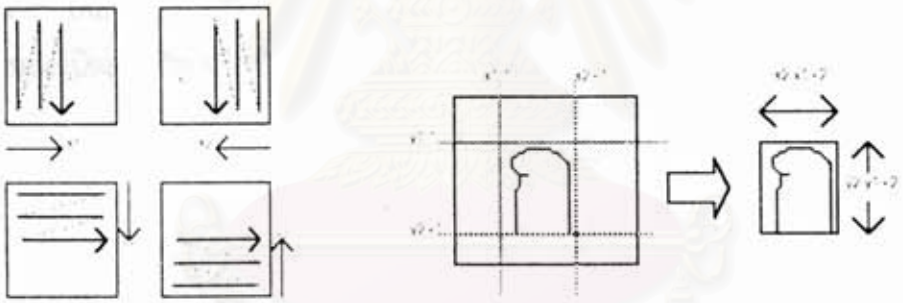
รูปที่ 2.3 การกำจัดสัญญาณรบกวน

2. การทำตัวอักษรให้บาง [8],[9] จุดประสงค์เพื่อทำให้ข้อมูลภาพของตัวอักษรมีความกว้างของเส้นเพียง 1 จุดภาพเท่านั้น ข้อมูลภาพที่มีความกว้างของเส้นเพียง 1 จุดภาพเรียกอีกชื่อว่า โครงร่างของภาพ (skeleton) วิธีการทำตัวอักษรให้บาง ได้มีผู้ศึกษามาก่อนหน้านี้แล้วเป็นจำนวนมาก วิธีการที่เลือกใช้ในการศึกษานี้เรียกว่า Sequential Processing Thinning หรือ Classical Thinning ของ Pavlidis [9] วิธีการนี้สามารถทำข้อมูลภาพของตัวอักษรให้เป็นโครงร่างตัวอักษรที่มีความกว้างของเส้นเพียง 1 จุดภาพได้ดี และมีการต่อเชื่อมของเส้นโครงร่างคล้ายกับตัวอักษรต้นฉบับค่อนข้างมาก แต่ข้อเสียของวิธีการนี้คือใช้เวลาในการประมวลผลค่อนข้างมาก เพราะ 1 จุดภาพต้องมีการประมวลผลซ้ำถึง 4 ครั้งต่อการแสกนตัวอักษร 1 รอบ ผลการทำให้บางแสดงในรูปที่ 2.4

ก. ตัวอักษรก่อนการทำให้บาง ข. ตัวอักษรหลังการทำให้บาง

รูปที่ 2.4 การทำตัวอักษรให้บาง

3. การปรับกรอบของตัวอักษร หลังการทำตัวอักษรให้บางแล้ว จะได้ขอบภาพที่ว่างเป็นจำนวนมากทั้งนี้เพราะความกว้างของตัวอักษรมีขนาดลดลง ตำแหน่งของตัวอักษรอาจจะอยู่เยื้องไปยังมุมใดมุมหนึ่งของกรอบภาพ เพื่อให้ง่ายต่อการกำหนดตำแหน่งของจุดใดๆในภาพ จึงจำเป็นต้องปรับขนาดของกรอบให้พอดีกับขนาดของภาพ โดยการตรวจสอบหาจุดภาพจากขอบภาพจนกระทั่งพบจุดภาพแรกในแนวขอบนั้นๆ (ดังแสดงในรูปที่ 2.5) ทำจนครบทั้ง 4 ขอบภาพ



ก. ทิศทางการตรวจสอบกรอบของตัวอักษร ข. การปรับกรอบของตัวอักษรให้พอดีกับตัวอักษร

รูปที่ 2.5 การปรับกรอบของตัวอักษร

จะได้ตำแหน่งของภาพใหม่ที่ตำแหน่ง X_1, X_2, Y_1, Y_2 แต่เพื่อให้สะดวกในการนำข้อมูลภาพไปใช้งาน เช่นการตรวจสอบด้วยตารางหน้าต่างขนาด 3x3 จึงต้องเหลือขอบนอกสุดของภาพทั้ง 4 ด้านให้ว่างไว้ด้านละ 1 จุดภาพ จะได้ข้อมูลภาพใหม่ที่มีกรอบภาพเป็น

3.1 ขอบซ้ายที่ตำแหน่ง $X_1 - 1$

3.2 ขอบขวาที่ตำแหน่ง $X_2 + 1$

3.3 ขอบบนที่ตำแหน่ง $Y_1 - 1$

3.4 ขอบล่างที่ตำแหน่ง $Y_2 + 1$

ดังนั้นข้อมูลภาพจะมีความกว้างของภาพเท่ากับ X_2-X_1+2 และมีความสูงของภาพเท่ากับ Y_2-Y_1+2

การทำให้เป็นเวกเตอร์

เนื่องจากข้อมูลภาพที่ได้มาเป็นภาพโครงร่างตัวอักษรที่ประกอบไปด้วยจุดภาพเรียงต่อกันไป เพื่อให้ง่ายในการนำไปใช้งานในการประมวลผลต่อๆไป จากแนวคิดของ Lam และ Suen [4] ที่ต้องการหาทิศทางของจุดภาพที่เรียงต่อเนื่องกันไปโดยการเปลี่ยนจุดภาพให้เป็นรหัสเงื่อนไข แล้วนำไปเปลี่ยนให้จุดภาพที่มีจุดภาพมีทิศทางไปในทางเดียวกันให้เป็นส่วนของเส้นตรงหรือเส้นโค้งเรียงต่อเนื่องกันไป แต่เพื่อให้สามารถใช้ได้กับตัวอักษรภาษาไทย ควรเปลี่ยนจุดภาพที่เรียงต่อๆกันไปในทิศทางเดียวกันให้อยู่ในรูปของเส้นตรง หรือเปลี่ยนจุดภาพที่มีการวนมาบรรจบกันเป็นวงกลมให้อยู่ในรูปของวงกลม เรียกเส้นตรงหรือวงกลมเหล่านี้ว่า เวกเตอร์ เวกเตอร์เหล่านี้มีการเชื่อมต่อกันในรูปต้นไม้เพื่อให้ยังคงลักษณะที่เหมือนกับโครงร่างเดิมของตัวอักษรมากที่สุด โดยที่แต่ละเวกเตอร์จะยังคงเก็บตำแหน่งเริ่มและปลายของส่วนของเส้นตรงนั้นไว้ เพื่อให้สามารถนำเวกเตอร์เหล่านี้มาปรับปรุงได้ในภายหลังถ้าพบว่ามีส่วนหนึ่งส่วนใดของตัวอักษรมีลักษณะเป็นส่วนเกิน หรือเป็นส่วนขาด

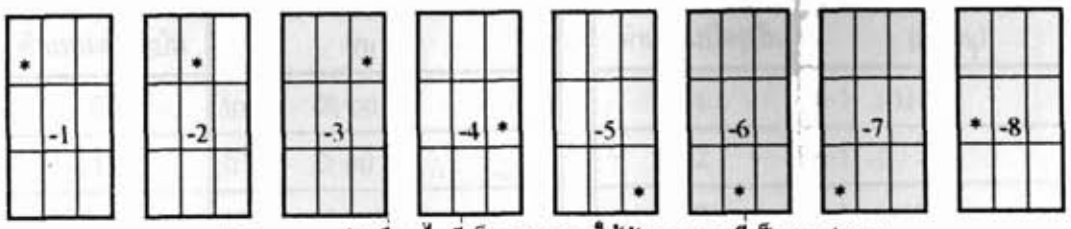
1. ข้อดีของการเปลี่ยนจากโครงร่างตัวอักษรไปเป็นต้นไม้ของเวกเตอร์ มีดังนี้

1.1 สามารถเข้าถึงตำแหน่งต่างๆของตัวอักษรได้อย่างรวดเร็ว

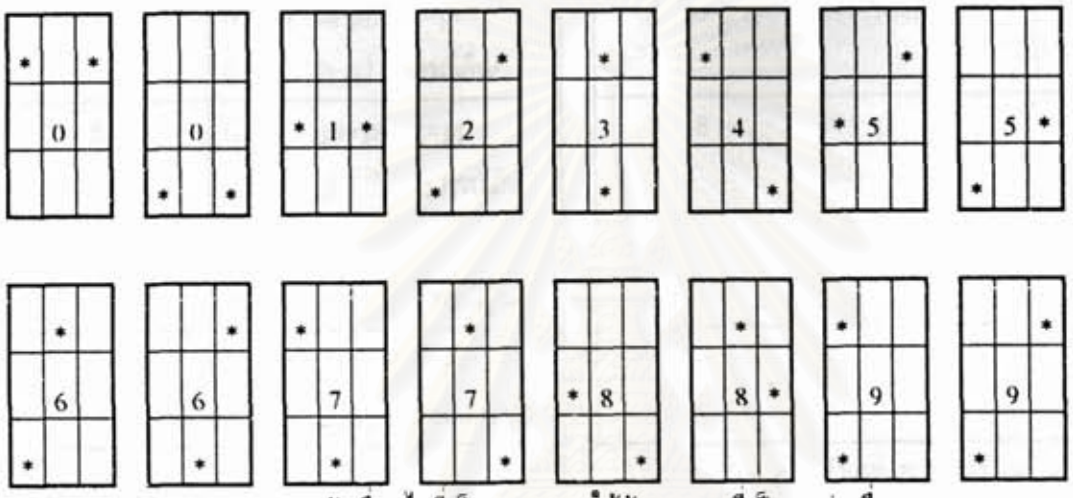
1.2 สามารถตรวจสอบหาส่วนหัวของตัวอักษรได้โดยง่าย

1.3 สามารถตัดส่วนเกินของตัวอักษรออก หรือเชื่อมส่วนที่ขาดออกจากกันเข้าด้วยกันได้โดยง่าย

2. การเข้ารหัสจุดภาพของตัวอักษร การทำจุดภาพของโครงร่างตัวอักษรให้เป็นเวกเตอร์นั้น ก่อนอื่นต้องทราบว่าแต่ละจุดภาพมีมุมที่สัมพันธ์กับจุดภาพข้างเคียงเป็นอย่างไร มีจุดที่เป็นจุดปลายและจุดทางแยกอยู่ที่ตำแหน่งใดบ้าง การหามุมสัมพันธ์ของจุดภาพสามารถทำได้ง่ายๆ โดยการแทนค่าแต่ละจุดภาพด้วยรหัสเงื่อนไข (condition code) โดยใช้วิธีของ Lam and Suen [4] ทำได้โดยใช้ตารางหน้าต่างขนาด 3x3 จุดภาพ ตรวจสอบกับทุกๆจุดภาพและทำการเปรียบเทียบกับจุดภาพรอบข้างทั้ง 8 จุดภาพในตารางหน้าต่าง (ดังแสดงในรูปที่ 2.6) โดยมีเงื่อนไขการแทนค่าด้วยรหัสเงื่อนไข ดังนี้



รูป 2.6 ก. รหัสเงื่อนไขที่เป็นเลขลบ ให้กับจุดภาพที่เป็นจุดปลาย



รูป 2.6 ข. รหัสเงื่อนไขที่เป็นเลขบวก ให้กับจุดภาพที่เป็นจุดต่อเนื่อง

2.1 ถ้า X เป็นจุดภาพ และมีจุดภาพรอบข้างมากกว่า 2 จุดขึ้นไป แสดงว่าเป็นจุดทางแยก ให้แทนจุดนั้นด้วยรหัสเงื่อนไข -9

2.2 ถ้า X เป็นจุดภาพ และมีจุดภาพรอบข้างเพียง 1 จุด แสดงว่าเป็นจุดปลายของภาพ ให้แทนด้วยเลขลบตามที่แสดงไว้ในรูปที่ 2.7 ก

2.3 ถ้า X เป็นจุดภาพ และมีจุดภาพรอบข้างเพียง 2 จุด ดังที่แสดงในรูปที่ 2.7ข แสดงว่า จุด X เป็นจุดต่อเนื่อง ให้แทนด้วยรหัสเงื่อนไขที่เป็นเลขบวก

2.4 ถ้า X ไม่ใช่จุดภาพ หรือเป็นจุดภาพที่มีจุดภาพรอบข้างนอกเหนือจากที่ระบุไว้ ให้แทนด้วยรหัสเงื่อนไข -128

รหัสเงื่อนไขที่ไม่ใช่เลข -9 และ -128 สามารถคำนวนหาตำแหน่งของจุดถัดไปโดยใช้สูตรตามตารางที่ 2.1ก สำหรับจุดต่อเนื่อง และ ตาราง 2.1ข สำหรับจุดปลาย

รหัสเงื่อนไขของตำแหน่งปัจจุบัน	ตำแหน่งของจุดถัดไป (n_i, n_j)
0	(pi ,2j-pj)
1	(i ,2j-pj)
2,4	(2i-pi ,2j-pj)
3	(2i-pi ,j)
5	(i-j+pj ,2j-pj) ; i=pi (i ,2j-pj) ; กรณีอื่นๆ
6	(2i-pi ,j-i+pi) ; j=pj (2i-pi ,j) ; กรณีอื่นๆ
7	(2i-pi ,i+1-pi) ; j=pj (2i-pi ,j) ; กรณีอื่นๆ
8	(i+j-pj ,2j-pj) ; i=pi (i ,2j-pj) ; กรณีอื่นๆ
9	(2i-pi ,pj)

รหัสเงื่อนไขของตำแหน่งปัจจุบัน	ตำแหน่งของจุดถัดไป (n_i, n_j)
-1	(i-1 ,j-1)
-2	(i-1 ,j)
-3	(i-1 ,j+1)
-4	(i ,j+1)
-5	(i+1 ,j+1)
-6	(i+1 ,j)
-7	(i+1 ,j-1)
-8	(i ,j-1)

(i,j) = ตำแหน่งของจุดปัจจุบัน

(pi,pj) = ตำแหน่งของจุดก่อนหน้า

ตารางที่ 2.1 ก. สูตรคำณวนหาจุดถัดไปของจุดต่อเนื่อง ข. สูตรคำณวนหาจุดถัดไปของจุดปลาย

รหัสเงื่อนไขของแต่ละจุดภาพที่เป็นจำนวนบวกที่มีค่า 1 ถึง 8 แสดงถึงจุดภาพนั้นทำมุมในทิศทางต่างๆกันดังแสดงในตาราง 2.2 รหัส 1 ถึง 4 เรียกว่ารหัสเงื่อนไขหลัก ทำมุมเป็นทวีคูณของมุม 45 องศา รหัส 5 ถึง 8 เรียกว่ารหัสเงื่อนไขรอง ทำมุมเป็นทวีคูณของมุม 30 องศา ส่วนรหัสเงื่อนไข 0 และ 9 เป็นจุดหักเหของเส้นโครงร่าง

จากตาราง 2.2 แสดงถึงจุดภาพที่เป็นรหัสเงื่อนไขเดียวกันเรียงต่อเนื่องกัน แต่ถ้าเป็นจุดภาพที่มีการเรียงของรหัสเงื่อนไขหลักและรหัสเงื่อนไขรองคละกันไปจะได้มุมของเส้นดังที่แสดงในตารางที่ 2.3 ข้อมูลภาพที่เข้ารหัสเงื่อนไข แสดงในรูปที่ 2.7

รหัส เงื่อนไข	มุม (องศา)
1	0
2	45
3	90
4	135
5	30
6	60
7	120
8	150

ตารางที่ 2.2 มุมของรหัสเงื่อนไข

รหัสเงื่อนไข เรียงต่อเนื่อง	มุมที่ได้ (องศา)
1-5-1	$0 < \theta < 30$
2-5-2	$30 < \theta < 45$
2-6-2	$45 < \theta < 60$
3-6-3	$60 < \theta < 90$
3-7-3	$90 < \theta < 120$
4-7-4	$120 < \theta < 135$
4-8-4	$135 < \theta < 150$
1-8-1	$150 < \theta < 180$

ตารางที่ 2.3 มุมที่ได้จากการเรียงรหัสเงื่อนไขคละกัน

ก. ข้อมูลภาพที่ถูกทำให้บาง ข. ข้อมูลภาพที่เข้ารหัสเงื่อนไข

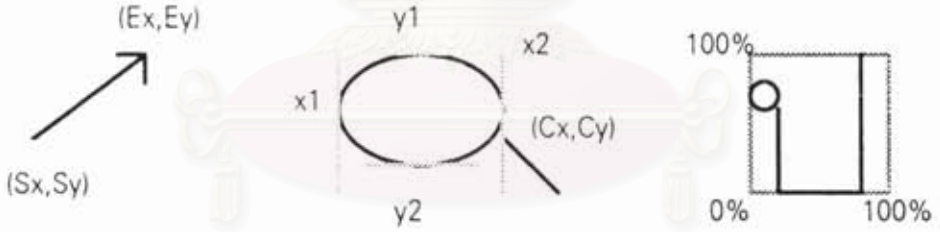
รูปที่ 2.7 การเข้ารหัสเงื่อนไข

3. ประเภทของเวกเตอร์ เวกเตอร์ที่ใช้สามารถแบ่งออกเป็น 2 ประเภทคือ

3.1 เวกเตอร์เส้นตรง แทนจุดภาพที่มีทิศทางเป็นเส้นตรง เวกเตอร์เส้นตรงประกอบด้วย ตำแหน่งของจุดแรก และตำแหน่งของจุดท้าย เท่านั้น ดังแสดงในรูปที่ 2.8 ก

3.2 เวกเตอร์วงกลม แทนจุดภาพที่มีตำแหน่งเริ่มต้นและตำแหน่งสุดท้ายวนมาบรรจบกัน เนื่องจากเวกเตอร์วงกลมถูกออกแบบมาสำหรับให้เป็นส่วนหัวของตัวอักษร เวกเตอร์วงกลมจึงถูกกำหนดให้มีจุดต่อเชื่อมเพียง 1 จุดให้สามารถต่อเชื่อมกับเวกเตอร์เส้นตรงได้ 1 เวกเตอร์ ดังนั้นเวกเตอร์วงกลมประกอบด้วย ตำแหน่งของจุดขอบซ้ายสุด, ขอบขวาสุด, ขอบบนสุด, ขอบล่างสุด และตำแหน่งของจุดที่เวกเตอร์วงกลมต่อเชื่อมกับเวกเตอร์อื่น ดังแสดงในรูปที่ 2.8 ข

เพื่อให้ง่ายต่อการใช้งาน จึงได้กำหนดให้เวกเตอร์มีขนาดความกว้างและความยาวของเวกเตอร์สูงสุดเป็น 100*100 เสมอเรียกว่า กรอบของเวกเตอร์ ดังนั้นตำแหน่งของจุดต่างๆในเวกเตอร์จึงมีขนาดที่เป็นสัดส่วนของ 100 เสมอ ดังแสดงในรูปที่ 2.8 ค



ก. เวกเตอร์เส้นตรง ข. เวกเตอร์วงกลม ค. กรอบของเวกเตอร์

รูปที่ 2.8 ประเภทและกรอบของเวกเตอร์

4. การแปลงจุดภาพที่เข้ารหัสให้เป็นเวกเตอร์เส้นตรง เป็นการรวมเอาจุดภาพที่มีรหัสเงื่อนไขเหมือนกัน หรือเป็นรหัสเงื่อนไขที่ต่อเนื่องกัน (ดังแสดงในตารางที่ 2.4) เข้าด้วยกันเป็น 1 เวกเตอร์เส้นตรง การแปลงจุดภาพที่เข้ารหัสเงื่อนไขให้เป็นเวกเตอร์ มีการทำงานดังแสดงใน algorithm 1

รหัสเงื่อนไข	รหัสต่อเนื่อง	รหัสไม่ต่อเนื่อง
1	1,5,8	-
2	2,5,6	9,0
3	3,6,7	-
4	4,7,8	9,0
5	1,2,5,6	0,8,9
6	2,3,5,6	0,7,9
7	3,4,7,8	0,6,9
8	2,4,7,8	0,5,9
9	2,4,5,6,7,8	ทุกรหัส
0	2,4,5,6,7,8	ทุกรหัส

ตารางที่ 2.4 รหัสต่อเนื่อง และรหัสไม่ต่อเนื่อง ของรหัสเงื่อนไข

5. การทำเส้นตรงให้เป็นเวกเตอร์วงกลม ในกรณีที่จุดทางแยกหนึ่งมีทางแยก 3 ทาง และมีอยู่ 2 ทางแยกที่สามารถเดินถึงกันและกันได้ โดยที่ไม่มีจุดทางแยกอื่นๆในระหว่างเส้นทางที่เดินถึงกันนั้น แสดงว่าทางแยกทั้ง 2 นั้นมีการเชื่อมต่อเป็นรูปวงกลมอยู่ ดังนั้นทางแยกที่เหลืออีก 1 เส้นทางก็จะเป็นจุดต่อเชื่อมของวงกลมกับเวกเตอร์อื่น ดังแสดงในรูปที่ 2.9 การทำให้เป็นเวกเตอร์วงกลมทำได้ดังนี้

Algorithm 1. การแปลงจุดภาพที่เข้ารหัสเงื่อนไขให้เป็นเวกเตอร์

1. If พบจุดปลาย หรือจุดทางแยกของภาพ (จุดภาพที่มีรหัสเงื่อนไข -1 ถึง -8 หรือ -9)
   then
2. ไปยังจุดปลายอันแรกสุด ถ้าไม่มีให้ไปที่จุดทางแยกอันแรกสุด
3. Do จนกว่าไม่สามารถไปยังจุดปลาย หรือจุดทางแยกได้อีก
   begin
   Trace ไปยังจุดถัดไป
4. เก็บค่ารหัสเงื่อนไขของจุดภาพปัจจุบัน เป็นรหัสเงื่อนไขเริ่มต้น
5. Trace ไปยังจุดถัดไป, เก็บค่ารหัสเงื่อนไขของจุดภาพ
6. If รหัสเงื่อนไขของจุดภาพปัจจุบันเป็นรหัสต่อเนื่องกับรหัสเงื่อนไขเริ่มต้น
   then ทำ step 5
   end if
7. สร้างเวกเตอร์เส้นตรง โดย
   ตำแหน่งเริ่มต้นของเวกเตอร์ได้จากจุดภาพที่ step 4.
   ตำแหน่งปลายของเวกเตอร์ ได้จากจุดภาพปัจจุบัน
   คำนวนตำแหน่งของเวกเตอร์ให้สัมพันธ์กับขนาดกรอบเวกเตอร์
8. If จุดปัจจุบันยังไม่เป็นจุดปลายหรือจุดทางแยก
   then ทำ step 4
   else ไปยังจุดปลายหรือจุดทางแยกถัดไป
   end if
   end do
9. else
   สร้างเวกเตอร์วงกลมที่ไม่มีจุดต่อเชื่อม โดยใช้ขนาดของภาพ เป็นขนาดของเวกเตอร์
10. end if

5.1 สร้างเวกเตอร์วงกลมโดยนำตำแหน่งของขอบบนสุด, ขอบล่างสุด, ขอบซ้ายสุด, ขอบขวาสุด จากเวกเตอร์ที่เชื่อมต่อถึงกันระหว่าง 2 ทางแยก และนำตำแหน่งที่เชื่อมกับเวกเตอร์ของทางแยกที่ 3 มาเป็นจุดต่อเชื่อมวงกลม

5.2 ลบเวกเตอร์ที่เชื่อมต่อถึงกันระหว่าง 2 ทางแยกออกให้หมด

ก. ก่อนทำให้เป็นวงกลม ข. หลังการทำให้เป็นวงกลม

รูปที่ 2.9 การแปลงจากเวกเตอร์เส้นตรงเป็นเวกเตอร์วงกลม

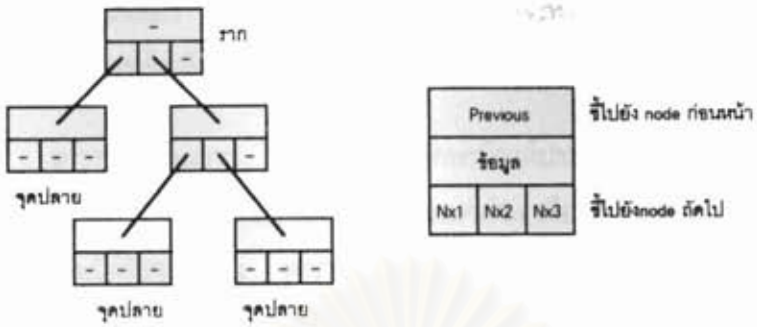
6. การเชื่อมเวกเตอร์เป็นรูปต้นไม้ เวกเตอร์ที่ได้จากการแปลงจุดภาพที่เข้ารหัสเงื่อนไข จะต้องนำมาเชื่อมโยงเข้าด้วยกันเป็นรูปต้นไม้ เพื่อต่อเชื่อมเวกเตอร์ต่างๆให้เหมือนกับรูปของโครงร่างตัวอักษรตามเดิม โดยใช้รูปแบบต้นไม้ที่ใช้เรียกว่า Binary-Tree of Order 3 ที่มีการเชื่อมโยงระหว่าง node แบบ doubly link list ดังแสดงในรูปที่ 2.10 ก เหตุที่กำหนดให้มีเพียง 3 ลำดับก็เพราะว่าข้อมูลภาพหลังการทำให้บางแล้ว จุดภาพที่เป็นจุดทางแยกจะมีทางแยกได้สูงสุดไม่เกิน 3 ทางเสมอ

ต้นไม้ประกอบด้วยสมาชิกที่เรียกว่า node เชื่อมโยงกัน โดยที่แต่ละ node มีรายละเอียดดังในรูปที่ 2.10 ข ซึ่งประกอบด้วย

6.1 ข้อมูลของเวกเตอร์ ระบุประเภท และขนาดของเวกเตอร์

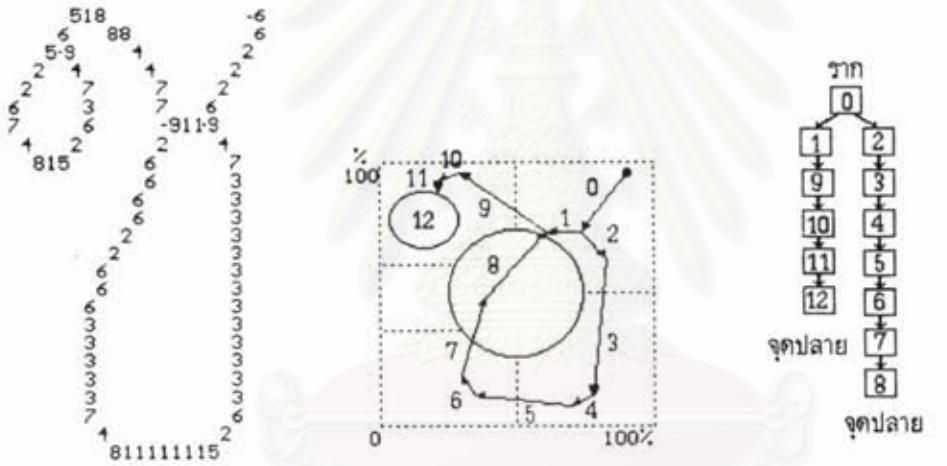
6.2 ตัวชี้ไปยัง node บน 1 ตัวชี้ เพื่อให้สามารถเดินย้อนขึ้นไปยัง node ต่างๆที่อยู่เหนือขึ้นไปได้ ในกรณีที่เป็นรากของต้นไม้จะไม่มี node บน ดังนั้นตัวชี้ไปยัง node บนของรากจะถูกกำหนดให้มีค่าเป็น -1 เสมอ

6.3 ตัวชี้ไปยัง node ล่าง 3 ตัวชี้ เพื่อให้สามารถเดินลงไปยัง node ที่อยู่ต่ำลงไป ในกรณีที่ตัวชี้ไปยัง node ล่างตัวใดไม่ได้ถูกใช้งานเนื่องจากไม่มี node ล่างถัดลงไป ให้กำหนดให้มีค่าเป็น -1 เช่นกัน ดังนั้นถ้าเป็น node ที่เป็นจุดปลาย ตัวชี้ทั้ง 3 จะมีค่าเป็น -1



ก. โครงสร้างของต้นไม้ ข. โครงสร้างของ node

รูปที่ 2.10 โครงสร้างของต้นไม้



ก. ข้อมูลภาพที่เข้ารหัสเงื่อนไข ข. เวกเตอร์ของตัวอักษร ค. ต้นไม้ของเวกเตอร์

รูปที่ 2.11 การแปลงจากข้อมูลภาพของตัวอักษรที่เข้ารหัสเงื่อนไขให้เป็นต้นไม้ของเวกเตอร์

การเลือก node ที่จะมาเป็นรากของต้นไม้ของเวกเตอร์ ปกติจะกำหนดให้จุดปลายของตัวอักษรจุดใดจุดหนึ่งเป็นราก แต่ถ้าในตัวอักษรนั้นไม่มีจุดปลายเลยก็ให้ใช้จุดทางแยกจุดใดจุดหนึ่งเป็นรากของต้นไม้แทน จากรูปที่ 2.11 ก. ข้อมูลภาพที่ถูกเข้ารหัสแล้ว พบจุดปลาย 1 จุดที่มุมขวาบนของภาพ ดังนั้นหลังจากทำให้เป็นเวกเตอร์ได้แล้ว ในรูปที่ 2.11 ข. จึงให้จุดปลายนั้นเป็นรากของต้นไม้ของเวกเตอร์ (node หมายเลข 0) แล้วเชื่อมแต่ละ node ของเวกเตอร์เข้าด้วยกันเป็นต้นไม้ แสดงในรูปที่ 2.11 ค.

7. ข้อมูลอื่นๆของเวกเตอร์ ชุดของเวกเตอร์นอกจากจะทำการเก็บ ต้นไม้ของเวกเตอร์แล้ว ยังมีข้อมูลที่สำคัญที่ใช้ช่วยในการรู้จำอีก เช่น

7.1 ตารางตำแหน่งราก ในกรณีที่ตัวอักษรมีองค์ประกอบมากกว่า 1 ส่วนที่ไม่ติดกัน เช่นตัวอักษร ฐ, ฉ เป็นต้น ต้นไม้ของเวกเตอร์สามารถมีได้หลายๆส่วน เพื่อให้สามารถเก็บองค์ประกอบของตัวอักษรแต่ละส่วนได้อย่างเป็นอิสระต่อกัน ตารางนี้ใช้เก็บตำแหน่งรากที่เป็นตำแหน่งเริ่มต้นของต้นไม้แต่ละชุด เพื่อให้สามารถเข้าถึงส่วนย่อยอื่นๆของตัวอักษรได้อย่างรวดเร็ว

7.2 ตารางตำแหน่งจุดปลายของตัวอักษร ใช้เก็บตำแหน่งของจุดปลายของตัวอักษรไม่ว่าจุดปลายนั้นจะเป็นเวกเตอร์เส้นตรง หรือเวกเตอร์วงกลมก็ตาม เพื่อใช้ในขั้นตอนการเปรียบเทียบตำแหน่งของจุดปลายของตัวอักษรระหว่างตัวอักษรที่ต้องการรู้จำกับตัวอักษรต้นแบบ ในการเปรียบเทียบตัวอักษรทาง feature

7.3 ตารางตำแหน่งจุดปลายที่เป็นวงกลม ใช้เก็บตำแหน่งของจุดปลายของตัวอักษรที่เป็นเวกเตอร์วงกลมเท่านั้น เพื่อให้สามารถค้นหาส่วนหัวของตัวอักษรได้โดยง่าย

7.4 จำนวนจุดทางแยกของตัวอักษร ใช้ช่วยในการตัดสินใจในการเปรียบเทียบตัวอักษรว่ามีความคล้ายของตัวอักษรมากน้อยเพียงใด

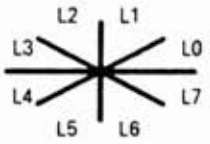
การเปลี่ยนเวกเตอร์ใช้เป็น primitive

เนื่องจากการเก็บข้อมูลของเวกเตอร์เก็บในลักษณะที่เป็นตำแหน่งของจุดเริ่มต้นและตำแหน่งของจุดปลายของเวกเตอร์ ดังนั้นเพื่อให้สามารถนำไปเปรียบเทียบกับตัวอักษรต้นแบบได้อย่างรวดเร็ว จึงต้องเปลี่ยนเวกเตอร์เหล่านี้ให้อยู่ใน primitive เช่น จากเวกเตอร์เส้นตรงให้เป็นรูปเส้นตรงที่มีทิศทางต่างๆกัน เป็นต้น จุดประสงค์ก็เพื่อลดการคำนวณในระหว่างการเปรียบเทียบตำแหน่งของเวกเตอร์ลงให้มากที่สุด

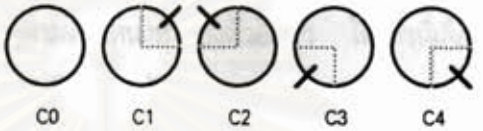
1. ลักษณะของ primitive ในการศึกษาครั้งนี้ ได้กำหนดไว้ดังนี้

1.1 primitive เส้นตรง มี 8 รูปแบบเหมือนกับของ freeman chain code เพื่อใช้แทนเวกเตอร์เส้นตรงที่ทำมุมกับทิศทั้ง 8 โดยที่แต่ละ primitive ครอบคลุมพื้นที่ 45 องศา ดังแสดงในรูปที่ 2.12 ก. เช่น primitive เส้นตรงหมายเลข 0 แทนเวกเตอร์เส้นตรงที่ทำมุมตั้งแต่ 0 ถึง 45 องศา เป็นต้น

1.2 primitive วงกลม มี 5 รูปแบบ โดย primitive แรกเป็นวงกลมที่ไม่มีจุดต่อเชื่อมกับเวกเตอร์อื่น ใช้แทนเวกเตอร์วงกลมที่ไม่มีจุดต่อเชื่อมกับเวกเตอร์อื่นเลย ส่วน primitive ที่เหลือเป็นวงกลมที่มีจุดต่อเชื่อมที่ครอบคลุมพื้นที่ 1 ใน 4 ของวงกลม แทนเวกเตอร์วงกลมที่มีจุดต่อเชื่อมอยู่ในบริเวณดังกล่าว ดังแสดงในรูปที่ 2.12 ข.



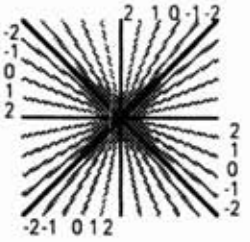
ก. primitive เส้นตรง

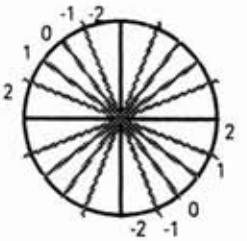


ข. primitive วงกลม

รูปที่ 2.12 primitive

2. มุมเบี่ยงเบน จากที่กล่าวมาว่า primitive ของเส้นตรง 1 ตัว แทนเส้นตรงที่มีมุมอยู่ในช่วง 45 องศา และ primitive วงกลมที่มีจุดต่อเชื่อม 1 ตัว แทนวงกลมที่มีจุดต่อเชื่อมในพื้นที่ถึง 90 องศา ซึ่งค่อนข้างไม่ละเอียดเมื่อเทียบกับลักษณะของตัวอักษรภาษาไทยที่มีลักษณะเป็นเส้นโค้ง และถ้าจุดที่กำลังพิจารณามีมุมที่อยู่คาบระหว่างมุมที่เป็นจุดตัดของ primitive การแทน primitive อาจจะผิดไปได้ เพื่อช่วยแก้ปัญหาดังกล่าว นอกจากจะแทนเวกเตอร์ด้วย primitive แล้ว ยังมีการแบ่งมุมของแต่ละ primitive ออกเป็น 5 ส่วนย่อย เรียกว่า มุมเบี่ยงเบน แทนด้วยตัวเลขตั้งแต่ -2 ถึง +2 { -2,-1,0,+1,+2 } ให้กับแต่ละส่วนย่อย เรียงทวนเข็มนาฬิกา (ดังแสดงในรูปที่ 2.13) ดังนั้น primitive จึงสามารถระบุมุมได้ละเอียดขึ้นอีก 5 เท่า เช่น เวกเตอร์เส้นตรงที่ทำมุม 148 องศา จะได้ primitive เส้นตรงหมายเลข 3 และมีมุมเบี่ยงเบนเป็น +1 เป็นต้น มุมเบี่ยงเบนนี้จะไม่ถูกนำไปใช้ในการเปรียบเทียบโดยการหาระยะของต้นไม้ของตัวอักษร แต่จะถูกนำไปใช้ในการเปรียบเทียบทาง feature ที่ต้องการข้อมูลสำหรับการเปรียบเทียบที่ละเอียดมากขึ้น

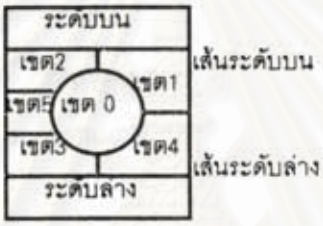




ก. มุมเบี่ยงเบนของเวกเตอร์เส้นตรง ข. มุมเบี่ยงเบนของเวกเตอร์วงกลม

รูปที่ 2.13 มุมเบี่ยงเบนของ primitive

3. การแบ่งระดับและแบ่งเขตหัวของตัวอักษร เนื่องจากในการศึกษาครั้งนี้ การเปรียบเทียบระหว่างตัวอักษรที่ต้องการรู้จำ กับตัวอักษรที่เป็นต้นแบบที่มีเป็นจำนวนมาก การที่จะเปรียบเทียบกับตัวอักษรต้นแบบทั้งหมดเพื่อหาตัวอักษรที่ใกล้เคียงที่สุดนั้นจึงเป็นเรื่องที่เสียเวลาเป็นอย่างมาก วิธีการที่สามารถนำมาใช้เพื่อช่วยลดจำนวนตัวอักษรต้นแบบที่ต้องถูกนำมาเปรียบเทียบ เป็นจำนวนมากคือการจัดหมวดหมู่ของตัวอักษรภาษาไทยตามระดับของตัวอักษรแล้วเลือกเปรียบเทียบเฉพาะตัวอักษรที่อยู่ในระดับเดียวกันเท่านั้น ซึ่งตามปกติตัวอักษรภาษาไทยมีการจัดระดับอยู่ 3 ระดับ คือ ระดับบน, ระดับกลาง และ ระดับล่าง แต่เนื่องจากตัวพยัญชนะและสระที่อยู่ในระดับกลางนั้นมีเป็นจำนวนมาก จึงต้องมีการแบ่งเขตของตัวอักษรในระดับกลางออกเป็นส่วนย่อยๆตามตำแหน่งหัวของตัวอักษร ในการนี้เลือกแบ่งเขตของตำแหน่งหัวตัวอักษรออกเป็น 6 ส่วน (ดังแสดงในรูปที่ 2.14) ดังนี้



รูปที่ 2.14 เขตย่อยของตัวอักษร

3.1 เขต0 คือพื้นที่วงกลมมีจุดศูนย์กลางอยู่กึ่งกลางของระดับกลาง มีรัศมี 25% ของความสูงของระดับกลาง เป็นเขตส่วนหัวของพยัญชนะ เช่น ค,ต หรือตัวเลข เช่น ๑ เป็นต้น

3.2 เขต1 คือพื้นที่มุมขวาบน เป็นเขตส่วนหัวของพยัญชนะ ง เป็นต้น

3.3 เขต2 คือพื้นที่มุมซ้ายบน เป็นเขตส่วนหัวของพยัญชนะ ข.ข เป็นต้น

3.4 เขต3 คือพื้นที่มุมซ้ายล่าง เป็นเขตส่วนหัวของพยัญชนะ ล,ส หรือสระ โ

3.5 เขต4 คือพื้นที่มุมขวาล่าง เป็นเขตส่วนหัวของพยัญชนะ ว เป็นต้น

3.6 เขต5 คือพื้นที่ซ้ายกลาง อยู่ระหว่างระดับความสูงจาก 35% ถึง 60% ของระดับกลาง เป็นเขตส่วนหัวของพยัญชนะ อ,ฮ,ฉ เป็นต้น

4. การหาส่วนหัวของตัวอักษร ตัวอักษรภาษาไทยมีเอกลักษณ์พิเศษตรงที่มีหัวของตัวอักษรและเกือบทั้งหมดของส่วนหัวมีลักษณะเป็นวงกลม จะมีส่วนหัวที่เป็นเส้นตรงอยู่บ้าง แต่ก็ไม่เป็นอุปสรรคต่อการค้นหา ทั้งนี้เพราะส่วนหัวของตัวอักษรภาษาไทยมักจะเป็นส่วนที่เป็นจุดปลายของตัวอักษรเสมอ (เป็นส่วนที่มีจุดเชื่อมต่อเพียง 1 จุดเท่านั้น) ดังนั้นการค้นหาตำแหน่งหัวของตัวอักษรจึงสามารถค้นหาได้จากจุด

ปลายของตัวอักษร จากขั้นตอนแรกสุดจนถึงปัจจุบัน เราได้ตารางตำแหน่งจุดปลายของตัวอักษร ซึ่งรวบรวมจุดปลายของตัวอักษรทั้งหมดไม่ว่าจะเป็นเส้นตรงและเป็นวงกลม การที่จะทราบว่าจุดปลายใดเป็นหัวของตัวอักษร สามารถหาได้จากข้อสมมติฐานตามนี้

4.1 ส่วนหัวของตัวอักษรภาษาไทย เป็นส่วนที่เป็นจุดปลายของตัวอักษรเสมอ

4.2 ให้จุดปลายที่เป็นวงกลม เป็นหัวของตัวอักษรเสมอ

4.3 ถ้าตัวอักษรมีจุดปลายจุดใดจุดหนึ่งที่เป็นวงกลม ให้ถือว่าวงกลมเป็นส่วนหัวของตัวอักษร ถ้าพบวงกลมมากกว่า 1 จุด ให้จุดที่พบก่อนตามลำดับดังนี้เป็นส่วนหัว เขต5, เขต2, เขต3, เขต0, เขต4,เขต 1, ระดับบน และ ระดับล่าง ดังแสดงในรูป 2.15 ก

4.3 ถ้าไม่พบวงกลม ให้หาจุดปลายที่เป็นเส้นตรงที่พบก่อนตามลำดับดังนี้เป็นส่วนหัว เขต5, เขต3, เขต0, เขต2, เขต4,เขต 1, ระดับบน และ ระดับล่าง ดังแสดงในรูป 2.15 ข

ก. จุดปลายที่เป็นเวกเตอร์วงกลม ข. จุดปลายที่เป็นเวกเตอร์เส้นตรง

รูปที่ 2.15 ลำดับความสำคัญของเขตหัวของตัวอักษรในการค้นหาหัวตัวอักษร

5. ต้นไม้ของ primitive หลังการแปลงเวกร์ให้กลายเป็น primitive แล้ว primitive ที่ได้จะถูกนำมาจัดเรียงในลักษณะต้นไม้คล้ายกับต้นไม้ของเวกเตอร์ แต่มีการจัดเรียงลำดับของ node ในต้นไม้เสียใหม่ให้มีการจัดเรียงตามลำดับ postfix เพื่อเตรียมสำหรับการนำไปใช้เปรียบเทียบกับตัวอักษรต้นแบบ รูปที่ 2.16 แสดงการแปลงจากเวกเตอร์ที่แสดงในรูป 2.11

5.1 node ของต้นไม้ primitive จะแตกต่างจาก node ของเวกเตอร์ ประกอบด้วย

5.1.1 primitive

5.1.2 ค่ามุมเบี่ยงเบน

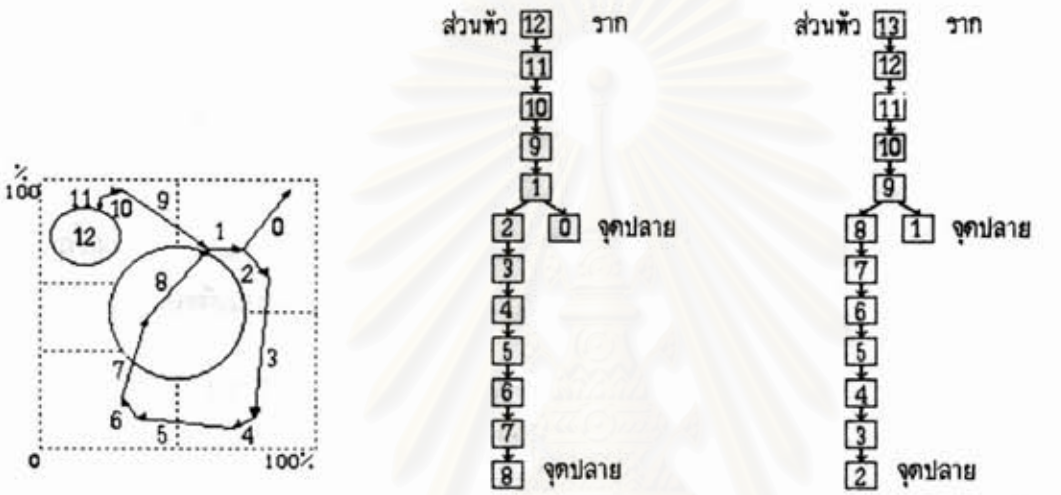
5.1.3 เขตของส่วนหัวและส่วนท้ายของ primitive

5.1.4 หมายเลข node ตามลำดับ postfix

5.1.5 ตัวชี้ไปยัง node บน 1 ตัวชี้

5.1.6 ตัวชี้ไปยัง node ล่าง 3 ตัวชี้

5.2 การแปลงจากต้นไม้ของเวกเตอร์เป็นต้นไม้ของ primitive ดังใน algorithm 2



ก. เวกเตอร์ของตัวอักษร ที่จัดเรียงแบบ postfix

ข. ต้นไม้ของเวกเตอร์ ที่จัดเรียงแบบ postfix

ค. ต้นไม้ของ primitive แสดงหมายเลข postfix ของ node

รูปที่ 2.16 การแปลงจากต้นไม้ของเวกเตอร์ให้เป็นต้นไม้ของ primitive

Algorithm 2. การแปลงจากต้นไม้ของเวกเตอร์เป็นต้นไม้ของ primitive

1. หาส่วนหัวของตัวอักษร
2. จัดเรียงต้นไม้ของเวกเตอร์แบบ postfix
3. กำหนดหมายเลข postfix ให้กับทุกๆ node ของต้นไม้ของเวกเตอร์
4. Do ทุกๆ node ของต้นไม้ของเวกเตอร์ ตามลำดับ postfix
   begin
5. สร้าง node ของ primitive
6. โยง node ของ primitive เข้ากับ node ใกล้เคียงให้เหมือนกับการโยงในต้นไม้ของเวกเตอร์
7. คำนวนหา primitive และค่ามุมเบี่ยงเบนของ primitive
8. คำนวนหาเขตหัวของตัวอักษร
9. end do

ตัวอักษรต้นแบบ

เนื่องจากการศึกษาครั้งนี้เลือกใช้วิธีการรู้จำตัวอักษรภาษาไทย โดยการเปรียบเทียบระหว่างตัวอักษรที่ต้องการรู้จำกับตัวอักษรต้นแบบ ดังนั้นเพื่อให้สามารถรู้จำตัวอักษรได้หลายๆรูปแบบจึงจำเป็นต้องเก็บตัวอักษรต้นแบบไว้หลายๆรูปแบบที่ไม่ซ้ำกัน ข้อมูลของตัวอักษรต้นแบบที่จัดเก็บไม่ได้เก็บเป็นข้อมูลภาพของตัวอักษรต้นแบบโดยตรง เพราะเมื่อต้องการนำตัวอักษรต้นแบบมาใช้ในการเปรียบเทียบต้องมีการประมวลผลใหม่ทุกครั้ง ดังนั้นข้อมูลที่เก็บจึงเป็นข้อมูลที่ผ่านการประมวลผลให้อยู่ในรูปของต้นไม้ของ primitive ,ต้นไม้ของเวกเตอร์แล้วมาจัดเก็บลงในแฟ้มข้อมูลของตัวอักษรต้นแบบ

1. แฟ้มตัวอักษรต้นแบบ แฟ้มตัวอักษรต้นแบบมีการจัดเก็บข้อมูลของตัวอักษรต้นแบบเพื่อให้สามารถนำไปใช้ในการเปรียบเทียบกับตัวอักษรที่ต้องการรู้จำได้ทันทีไม่ต้องนำไปประมวลผลใหม่ การเก็บข้อมูลในแฟ้มตัวอักษรต้นแบบจะประกอบด้วยตัวอักษรต้นแบบเรียงกันไปตามลำดับของการนำเข้าไปจัดเก็บ โดยตัวอักษรต้นแบบ 1 ตัวอักษรประกอบไปด้วย

1.1 ความยาวของข้อมูลตัวอักษรต้นแบบ มีขนาดแตกต่างกันไปตามตัวอักษรต้นแบบ เพื่อให้สามารถข้ามไปยังตัวอักษรต้นแบบถัดไปได้อย่างถูกต้อง

1.2 หมายเลขดัชนี เป็นตัวเลขที่ไม่ซ้ำกันใช้กำกับตัวอักษรต้นแบบทุกๆตัว เพื่อใช้อ้างอิงร่วมกับแฟ้มดัชนีของตัวอักษรต้นแบบ

1.3 ตำแหน่งเส้นฐานและเส้นระดับบนของตัวอักษร เพื่อให้ทราบระดับของตัวอักษรต้นแบบ

1.4 จำนวนจุดทางแยกและจุดปลายของตัวอักษร เพื่อใช้ช่วยในการเปรียบเทียบทาง feature

1.5 ต้นไม้ของ primitive เพื่อใช้เปรียบเทียบกับตัวอักษรที่ต้องการรู้จำ

1.6 ตารางส่วนย่อยของต้นไม้ primitive เพื่อใช้บอกจุดเริ่มต้นของต้นไม้ primitive ในทุกๆส่วนย่อยของตัวอักษร

1.7 ต้นไม้ของเวกเตอร์ เพื่อใช้แสดงรูปร่างของตัวอักษรต้นแบบ และใช้เปรียบเทียบทาง feature

1.8 ตารางส่วนย่อยของต้นไม้เวกเตอร์ เพื่อบอกจุดเริ่มต้นของต้นไม้เวกเตอร์ในทุก ๆ ส่วนย่อยของตัวอักษร

1.9 เงื่อนไขการตรวจสอบพิเศษ ใช้สำหรับตรวจสอบตัวอักษรที่มีความคล้ายกับตัวอักษรอื่นๆ เพื่อให้สามารถรู้จำตัวอักษรได้ถูกต้องมากยิ่งขึ้น

1.10 รหัสแอสกี เพื่อระบุรหัสแอสกีของตัวอักษรต้นแบบ ภาษาไทยใช้รหัส สมอ. หรือรหัสอื่นๆ ก็ได้ขึ้นอยู่กับการป้อนขณะที่เก็บตัวอักษรต้นแบบ

2. แฟ้มดัชนีของตัวอักษรต้นแบบ เนื่องจากการเปรียบเทียบระหว่างตัวอักษรที่ต้องการรู้จำกับตัวอักษรต้นแบบนั้นจะเลือกเฉพาะตัวอักษรต้นแบบที่มีระดับของตัวอักษรตรงกัน และมีตำแหน่งหัวของตัวอักษรอยู่ในเขตเดียวกันเท่านั้น แต่ว่าการจัดเก็บตัวอักษรต้นแบบที่อยู่ในแฟ้มตัวอักษรต้นแบบไม่ได้มีการจัดเรียงตามความต้องการดังกล่าว ดังนั้นจึงต้องมีการสร้างแฟ้มข้อมูลสำหรับเป็นดัชนีให้กับแฟ้มตัวอักษรต้นแบบโดยเฉพาะ เพื่อให้การค้นหาและการอ่านตัวอักษรต้นแบบจากแฟ้มทำได้รวดเร็วยิ่งขึ้น

2.1 การจัดเรียงตัวอักษรต้นแบบในแฟ้มดัชนีมีการเรียงตามลำดับดังนี้

2.1.1 จัดเรียงตามหมายเลขเขตของส่วนหัวของตัวอักษรเป็นอันดับแรก

2.1.2 จัดเรียงตามจำนวนของวงกลมที่ปรากฏในตัวอักษรเป็นอันดับต่อไป

2.1.3 จัดเรียงตามจำนวนของจุดปลายที่ปรากฏในตัวอักษรเป็นอันดับสุดท้าย

2.2 ดัชนีที่ใช้กำกับตัวอักษรต้นแบบ 1 ตัวอักษร ในแฟ้มดัชนีของตัวอักษรประกอบด้วย

2.2.1 หมายเลขดัชนี กำหนดให้ตรงกับหมายเลขดัชนีของตัวอักษรต้นแบบ เพื่อใช้ตรวจสอบว่าดัชนีนั้นชี้ไปยังตัวอักษรต้นแบบได้ถูกต้องหรือไม่

2.2.2 ตำแหน่งเริ่มต้นของตัวอักษรในแฟ้มตัวอักษรต้นแบบ เพื่อให้สามารถเข้าถึงข้อมูลของตัวอักษรต้นแบบได้

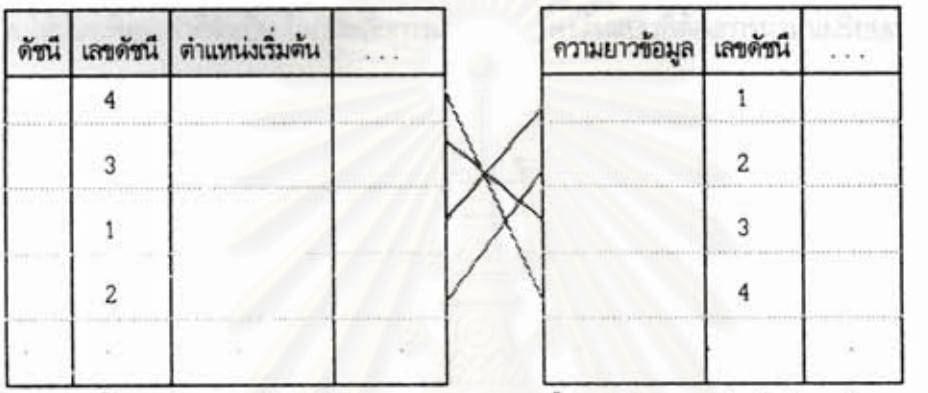
2.2.3 ความยาวของตัวอักษรต้นแบบ

2.2.4 ค่าดัชนีของตัวอักษรต้นแบบ มีจำนวนตามส่วนย่อยของตัวอักษร ในที่นี้กำหนดจำนวนของส่วนย่อยของตัวอักษรสูงสุดเพียง 2 ส่วนย่อย เนื่องจากตัวอักษรภาษาไทยไม่ปรากฏตัวอักษรที่มีมากกว่า 2 ส่วนย่อยเลย และในแต่ละส่วนย่อยของตัวอักษรประกอบด้วย

2.2.4.1 หมายเลขเซตของส่วนหัวของตัวอักษร

2.2.4.2 จำนวนของจุดปลายที่เป็นวงกลม

2.2.4.3 จำนวนของจุดปลายทั้งหมด



ก. โครงสร้างของแฟ้มดัชนี ข. โครงสร้างของแฟ้มตัวอักษรต้นแบบ

รูปที่ 2.17 โครงสร้างของแฟ้มตัวอักษรต้นแบบ

การเปรียบเทียบตัวอักษรเพื่อการรู้จำ

จากที่กล่าวมาแล้วว่าวิธีการรู้จำตัวอักษรภาษาไทยของการศึกษาครั้งนี้เลือกใช้ การเปรียบเทียบระหว่างตัวอักษรที่ต้องการรู้จำกับตัวอักษรต้นแบบ ซึ่งมีการเปรียบเทียบอยู่ 2 แบบ แบบแรกเป็นการวิเคราะห์โครงร่างของตัวอักษร โดยวิธีหาค่าระยะระหว่างต้นไม้ของตัวอักษรทั้งสอง ได้ผลการเปรียบเทียบเรียกว่า ค่าระยะ และอีกวิธีเป็นการเปรียบเทียบทาง feature เป็นการนำเอาลักษณะเด่นของตัวอักษรมาทำการเปรียบเทียบเพื่อให้การรู้จำทำได้ถูกต้องมากยิ่งขึ้น ได้ผลการเปรียบเทียบเรียกว่า ค่าความต่าง ดังมีรายละเอียดดังนี้

1. การวิเคราะห์โครงร่างของตัวอักษร เป็นการเปรียบเทียบเพื่อหาค่าความแตกต่างระหว่างตัวอักษรต้นแบบกับตัวอัก ี่ต้องการรู้จำ วิธีที่ใช้ในการศึกษาครั้งนี้ใช้วิธีการคำนวณหาค่าระยะ (Tree-to-Tree Distance) ระหว่างต้นไม้ primitive ของตัวอักษรที่ต้องการรู้จำกับของตัวอักษรต้นแบบ วิธีการคำนวนหาค่าระยะใช้วิธีที่พัฒนาโดย Shin [3] ซึ่งเป็นการคำนวนแบบไดนามิกโปรแกรมมิง

1.1 เงื่อนไขการเปรียบเทียบเพื่อหาค่าระยะ เพื่อให้สามารถรู้จำทำได้ถูกต้องและรวดเร็วมากที่สุด จำเป็นต้องตั้งเงื่อนไขของการเปรียบเทียบไว้ ทั้งนี้เพื่อลดจำนวนตัวอักษรที่ต้องนำมาเปรียบเทียบให้น้อยที่สุดเท่าที่จำเป็น โดยมีหลักการเลือกตัวอักษรต้นแบบที่ต้องการนำมาเปรียบเทียบดังนี้

1.1.1 เปรียบเทียบเฉพาะตัวอักษรต้นแบบที่เป็นตัวอักษรระดับเดียวกันเท่านั้น ดังนั้นตัวอักษรที่นำมาทำการรู้จำต้องกำหนด เส้นฐาน และเส้นระดับบนให้ถูกต้อง

1.1.2 เปรียบเทียบเฉพาะตัวอักษรที่มีส่วนหัวของตัวอักษรอยู่ในเขตเดียวกัน

1.1.3 เปรียบทียบเฉพาะตัวอักษรที่มีจำนวนส่วนย่อยของตัวอักษรเท่ากัน

1.2 การหาค่าระยะ (Tree-to-Tree Distance) [3],[8] เพื่อหาค่าระยะของแมทริกซ์ D(i,j) โดยที่ i และ j เป็นค่าเลขลำดับ postfix ของต้นไม้ของตัวอักษรต้นแบบ (β) กับต้นไม้ของตัวอักษรที่ต้องการรู้จำ (α) ตามลำดับ ให้ $a = h^{-1}_{\alpha}(j)$ และ $b = h^{-1}_{\beta}(i)$ โดย a และ b เป็น node ของ α และ β , $h_{\alpha}(a) = j$ และ $h_{\beta}(b) = i$ ดังนั้น D(i,j) จะเป็นค่าระยะที่ต่ำที่สุด เพื่อที่จะให้ได้ต้นไม้ย่อย α/a จาก ต้นไม้ย่อย β/b กำหนดให้ D เป็นแมทริกซ์ ขนาด m*n โดย m และ n เป็นจำนวน node ของต้นไม้ α และ β ตามลำดับ algorithm ของการหาค่าระยะแสดงใน algorithm 3

1.3 cost ที่ใช้ใน algorithm 3. กำหนดไว้ดังนี้

1.3.1 การแทรกเข้า ให้ cost เป็น 1 เหมือนกับค่าที่กำหนดไว้โดย Shin [3]

1.3.2 การตัดออก ให้ cost เป็น 1 เช่นเดียวกับ Shin [3]

1.3.3 การแทนที่ primitive กำหนดไว้แตกต่างกับของ Shin ซึ่งเดิม Shin กำหนดให้การแทนที่ primitive ใดๆ มี cost เป็น 1 เสมอ แต่การศึกษาครั้งนี้ มีทั้ง primitive เส้นตรง และ primitive วงกลม ดังนั้นจึงควรกำหนดให้การแทนที่ primitive ที่ใกล้เคียงกันมีค่า cost ต่ำกว่าการแทนที่ primitive ที่แตกต่างกันมาก cost ของการแทนที่ primitive แสดงไว้ในตารางที่ 2.5

Algorithm 3. Tree-to-Tree Distance Algorithm

D(0,0) =

Do j=1 TO m

D(0,j) = N(α/a)*q, Where a = $h^{-1}\alpha(j)$

END DO

DO i=1 TO n

D(i,0) = N(β/b)*p, Where b = $h^{-1}\beta(i)$

END DO

DO i=1 TO n

DO j=1 TO m

E(0,0) = 0

DO L=1 TO t, Where t = r(a)

E(0,L) = E(0,L-1)+N(α/a.L)*q

END DO

E(0,t+1) = E(0,t)+q

DO k=1 TO s, Where s = r(b)

E(k,0) = E(k-1,0)+N(β/b.k)*p

END DO

E(s+1,0) = E(s,0)+p

DO k=1 TO s, Where s = r(b)

DO L=1 TO t { E(k-1,L)+N(β/b.k)*p
E(k,L) = min { E(k,L-1)+N(α/a.L)*q
END DO { E(k-1,L-1)+D(u,v), Where u=hβ(b.k),v=hα(a.L)

END DO

DO L=1 TO t { E(s+1,L-1)+N(α/a.L)*q
E(s+1,L) = min { E(s,L)+p
END DO { E(0,L-1)+D(i,v), Where v=hα(a.L)

DO k=1 TO s { E(k-1,t+1)+N(β/b.k)*p
E(s+1,L) = min { E(k,s)+q
END DO { E(k-1,0)+D(u,j), Where u=hβ(b.k)

{ E(s,t+1) + p
D(i,j) = min { E(s+1,t) + q
{ E(s,t) + r, Where r = subsitution cost of primitive α(a) & β(b)

END DO

END DO

ตาราง 2.5 cost ของการแทนที่ primitive

primitive	L 0	L 1	L 2	L 3	L 4	L 5	L 6	L 7	C 0	C 1	C 2	C 3	C 4
เส้นตรง L 0	0	1	2	3	4	3	2	1	5	5	5	5	5
เส้นตรง L 1	1	0	1	2	3	4	3	2	5	5	5	5	5
เส้นตรง L 2	2	1	0	1	2	3	4	3	5	5	5	5	5
เส้นตรง L 3	3	2	1	0	1	2	3	4	5	5	5	5	5
เส้นตรง L 4	4	3	2	1	0	1	2	3	5	5	5	5	5
เส้นตรง L 5	3	4	3	2	1	0	1	2	5	5	5	5	5
เส้นตรง L 6	2	3	4	3	2	1	0	1	5	5	5	5	5
เส้นตรง L 7	1	2	3	4	3	2	1	0	5	5	5	5	5
วงกลม C 0	5	5	5	5	5	5	5	5	0	5	5	5	5
วงกลม C 1	5	5	5	5	5	5	5	5	5	0	1	2	1
วงกลม C 2	5	5	5	5	5	5	5	5	5	1	0	1	2
วงกลม C 3	5	5	5	5	5	5	5	5	5	2	1	0	1
วงกลม C 4	5	5	5	5	5	5	5	5	5	1	2	1	0

2. การวิเคราะห์ทาง feature จากข้อเสียของการวิเคราะห์โครงร่างของตัวอักษรโดยวิธีการเปรียบเทียบแบบหาค่าระยะคือ เป็นการเปรียบเทียบเพื่อหาจำนวนของ primitive ที่คล้ายคลึงกันเป็นหลัก ดังนั้นถ้าตัวอักษรทั้ง 2 ตัวมีลักษณะโครงร่างที่เหมือนกันทุกประการ แต่มีจำนวนของ primitive ที่แตกต่างกัน ค่าระยะที่ได้อย่างน้อยที่สุดก็คือค่าผลต่างระหว่างจำนวนของ primitive ของตัวอักษรทั้ง 2 นั้น จากหลักการคำนวนดังกล่าว อาจกล่าวได้ว่าการเปรียบเทียบโดยการหาค่าระยะระหว่างต้นไม้ของ primitive จะได้ผลการเปรียบเทียบที่ถูกต้อง ก็ต่อเมื่อตัวอักษรทั้ง 2 มีรูปร่างที่แตกต่างกันค่อนข้างมากเท่านั้น แต่ถ้าเป็นการเปรียบเทียบตัวอักษรที่คล้ายกันมากๆ เช่น ค กับ ด หรือ ข กับ ข ค่าระยะที่ได้แทบไม่แตกต่างกันเลย ทำให้ยากต่อการจำแนกตัวอักษร ดังนั้นหลังจากการเปรียบเทียบโดยหาค่าระยะของตัวอักษรแล้ว จึงต้องมีการเปรียบเทียบทาง feature เพื่อหาความแตกต่างของตัวอักษรทั้ง 2 ออกได้อย่างถูกต้องมากยิ่งขึ้น

Algorithm 4. การเปรียบเทียบตำแหน่งของจุดปลายระหว่างตัวอักษรต้นแบบกับตัวอักษรที่ต้องการรู้จำ

```
1.  Do ทุกๆ ตัวอักษรต้นแบบที่ถูกนำมาเปรียบเทียบ
      begin
2.      ให้จำนวนจุดปลายที่อยู่ต่างเขต มีค่าเป็น 0
        ให้จำนวนจุดปลายที่ห่างกันเกินไป มีค่าเป็น 0
        ให้ค่าความต่างของตัวอักษรต้นแบบ มีค่าเป็น 0
3.      Do ทุกๆ จุดปลายของตัวอักษรที่ต้องการรู้จำจากตารางจุดปลาย
          begin
4.          เปรียบเทียบทุกๆจุดปลายของตัวอักษรต้นแบบ หาจุดปลายที่ใกล้เคียงที่สุด
5.          If  ระยะห่างระหว่างจุดปลายเกิน 20% หรือ
                cost ของ pirmitive ทั้ง 2 ต่างกันเกิน 1
              then  เพิ่มค่าจำนวนจุดปลายที่ห่างกันเกินไปอีก 1
            end if
6.          If  จุดปลายทั้งสองอยู่ต่างเขตกัน
              then  เพิ่มค่าจำนวนจุดปลายที่อยู่ต่างเขตอีก 1
            end if
7.      end do
8.      เพิ่มค่าความต่าง = 4*จำนวนจุดปลายที่อยู่ต่างเขต
        เพิ่มค่าความต่าง = 2*จำนวนจุดปลายที่ห่างกันเกินไป
9.  end do
```

2.1 การเปรียบเทียบตำแหน่งของจุดปลาย เป็นวิธีการเปรียบเทียบตำแหน่งของจุดปลายของตัวอักษรต้นแบบกับตัวอักษรที่ต้องการรู้จำ เพื่อหาความคล้ายของตัวอักษร โดยนับจำนวนของจุดปลายที่มีตำแหน่งไม่ตรงกัน แล้วนำไปเพิ่มให้กับค่าความต่างของการเปรียบเทียบทาง feature ให้กับตัวอักษรต้นแบบที่นำมาเปรียบเทียบ

2.1.1 algorithm ของการเปรียบเทียบ แสดงใน algorithm 4

2.1.2 จาก ภาคผนวก ค. พบว่าตัวอักษรที่เปรียบเทียบแล้วได้จำนวนจุดปลายที่ห่างกันเกินไปมากกว่า 2 จุด จะเป็นตัวอักษรที่ให้ความแตกต่างกันค่อนข้างมาก และจากข้อ 3. ค่าความต่างที่ยอมรับได้ว่าเป็นตัวอักษรที่ใกล้เคียงกันมีค่าต่ำกว่า 4 ดังนั้นจึงกำหนดให้ค่าความต่างเพิ่มค่าขึ้นจุดละ 2 เพื่อให้ตัวอักษรที่มีจำนวนจุดปลายที่ห่างกันเกินไปมากกว่า 2 จุดเป็นตัวอักษรที่ใช้ไม่ได้

2.1.3 จาก ภาคผนวก ค. พบว่าตัวอักษรที่เปรียบเทียบแล้วได้จำนวนจุดปลายที่อยู่ต่างเซตเกิน 1 จุด จะเป็นตัวอักษรที่ให้ความแตกต่างกันค่อนข้างมาก และจากข้อ 3. ค่าความต่างที่ยอมรับได้ว่าเป็นตัวอักษรที่ใกล้เคียงกันมีค่าต่ำกว่า 4 ดังนั้นจึงกำหนดให้ค่าความต่างเพิ่มค่าขึ้นจุดละ 4 เพื่อให้ตัวอักษรที่มีจำนวนจุดปลายที่อยู่ต่างเซตมากกว่า 1 จุดเป็นตัวอักษรที่ใช้ไม่ได้

2.2 การเปรียบเทียบส่วนหัวของตัวอักษร เพื่อตรวจสอบว่าตัวอักษรต้นแบบที่รู้จำได้นั้นมีส่วนหัวที่เหมือน หรือแตกต่างกับตัวอักษรที่นำมารู้จำมากน้อยเพียงใด โดยใช้ algorithm 5 ค่า cost ของการเปรียบเทียบ primitive ของหัวของตัวอักษรต้นแบบกับตัวอักษรที่ต้องการรู้จำ จะถูกนำไปเพิ่มให้กับค่าความต่างของตัวอักษรต้นแบบนั้นให้มากขึ้น

Algorithm 5. การเปรียบเทียบส่วนหัวของตัวอักษร

1. หาส่วนหัวของตัวอักษรที่ต้องการรู้จำ
2. Do ทุกๆ ตัวอักษรต้นแบบที่ถูกนำมาเปรียบเทียบ
   begin
3. หาส่วนหัวของตัวอักษรต้นแบบ
4. นำ primitive ของหัวตัวอักษรทั้งสอง ไปหา cost โดยใช้ตาราง 2.5
5. นำ cost ที่ได้ไปเพิ่มให้กับค่าความต่างของตัวอักษรต้นแบบ
6. end do

2.3 การเปรียบเทียบกรณีพิเศษ เนื่องจากการเปรียบเทียบที่ผ่านมา ไม่สามารถจำแนกตัวอักษรต้นแบบที่มีความคล้ายกันมากๆออกจากกันได้ จึงต้องมีวิธีที่สามารถอธิบายลักษณะเด่นของตัวอักษรต้นแบบ เพื่อให้สามารถนำลักษณะเด่นของตัวอักษรต้นแบบนั้นไปเปรียบเทียบกับลักษณะของตัวอักษรที่ต้องการรู้จำ ตัวอักษรต้นแบบที่เปรียบเทียบในกรณีพิเศษแล้ว พบว่าให้ผลการเปรียบเทียบว่าแตกต่างจากตัวอักษรที่ต้องการรู้จำ จะถูกเพิ่มค่าความต่างของตัวอักษรต้นแบบนั้นไปอีก 2 เท่าของค่าความต่างที่ยอมรับได้ที่กำหนดไว้ในข้อ 3. เพื่อให้กลายเป็นตัวอักษรต้นแบบที่แตกต่างจากตัวอักษรที่ต้องการรู้จำมากๆ

การเปรียบเทียบทาง feature ทั้ง 2 ข้อที่ผ่านมาจะถูกกำหนดให้กระทำกับทุกๆ ตัวอักษรต้นแบบที่ถูกนำไปเปรียบเทียบ แต่การเปรียบเทียบกรณีพิเศษนี้จะต้องเลือกใช้ให้เหมาะสมกับตัวอักษรต้นแบบที่ค่อนข้างจะไม่แตกต่างจากตัวอักษรต้นแบบอื่นๆมากนัก จนเป็นผลให้การรู้จำผิดพลาดเท่านั้น ทั้งนี้ก็เพื่อประหยัดเวลาในขบวนการรู้จำ

2.3.1 การเปรียบเทียบประเภทหัวของตัวอักษร เนื่องจากหัวของตัวอักษรภาษาไทยมีหลายรูปแบบ และยังมีตำแหน่งที่แตกต่างกันอีกด้วย โดยสามารถแบ่งได้ดังนี้

2.3.1.1 เขตหัวหมายเลข 0 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบขวาของวงกลม เช่นตัวอักษร ค, ต

2.3.1.2 เขตหัวหมายเลข 0 ประเภทวงกลม มีเส้นต่อเชื่อมที่ขอบซ้ายของวงกลม เช่นตัวอักษร ค, ต

2.3.1.3 เขตหัวหมายเลข 1 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบขวาของวงกลม เช่นตัวอักษร ง

2.3.1.4 เขตหัวหมายเลข 2 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบบนของวงกลม เช่นตัวอักษร ข, ช

2.3.1.5 เขตหัวหมายเลข 2 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบขวาของวงกลม เช่นตัวอักษร บ, ป

2.3.1.6 เขตหัวหมายเลข 2 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบซ้ายของวงกลม เช่นตัวอักษร ย, ฝ

2.3.1.7 เขตหัวหมายเลข 3 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบขวาของวงกลม เช่นตัวอักษร ภ, ฏ

2.3.1.8 เขตหัวหมายเลข 3 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบซ้ายของวงกลม เช่นตัวอักษร ฤ, สระ เ

2.3.1.9 เขตหัวหมายเลข 4 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบขวาของวงกลม เช่นตัวอักษร ว, ร

2.3.1.10 เขตหัวหมายเลข 5 ประเภทหัววงกลม มีเส้นต่อเชื่อมที่ขอบซ้ายของวงกลม เช่นตัวอักษร อ, ฮ

2.3.1.11 เขตหัวหมายเลข 5 ประเภทหัวเส้นตรง ทำมุมชี้ลงล่าง เช่นตัวอักษร ธ

2.3.2 การเปรียบเทียบส่วนหัวของตัวอักษร

2.3.2.1 เขตหัวหมายเลข 2 จากหัวตัวอักษรเป็นเส้นโค้ง ไม่มีหยัก เช่นตัวอักษร ข, ช

2.3.2.2 เขตหัวหมายเลข 2 จากหัวตัวอักษรเป็นเส้นโค้งมี 1 หยัก เช่นตัวอักษร ข, ช

2.3.3 การเปรียบเทียบส่วนบนของตัวอักษร

2.3.3.1 เขตหัวหมายเลข 1 หรือ 2 จากซ้ายไปขวา มีลักษณะโค้งลง ไม่มีหยักกลาง เช่นตัวอักษร ค, ด

2.3.3.2 เขตหัวหมายเลข 1 หรือ 2 จากซ้ายไปขวา มีลักษณะโค้งลง มี 1 หยักกลาง เช่นตัวอักษร ค, ด

2.3.4 การเปรียบเทียบส่วนล่างของตัวอักษร

2.3.4.1 ที่ระดับล่าง จากขวาไปซ้ายมี 1 หยัก เช่นตัวอักษร ฎ

2.3.4.2 ที่ระดับล่าง จากขวาไปซ้ายมี 2 หยัก เช่นตัวอักษร ฎ

2.3.5 การเปรียบเทียบลักษณะเฉพาะของตัวอักษร การเปรียบเทียบแบบนี้ต้องการจำแนกตัวอักษรที่มีลักษณะคล้ายกับตัวอักษรอื่นมากๆ ดังนั้นจึงต้องหาลักษณะเด่นทาง feature ของตัวอักษรที่ไม่ซ้ำกับตัวอักษรอื่นออกมาใช้เปรียบเทียบ การเปรียบเทียบแบบนี้จะทำกับบางตัวอักษรเท่านั้น จากผลการทดลองพบว่า มีตัวอักษรที่ต้องการการตรวจสอบเป็นกรณีพิเศษตามลักษณะเฉพาะของตัวอักษร ดังนี้

2.3.5.1 ตรวจสอบสระอิ ที่มีลักษณะเป็นวงกลมเดี่ยวในระดับบนของตัวอักษรมีความคล้ายกับสระอำที่เป็นวงกลมเดี่ยวเช่นกัน แต่ที่แตกต่างคือสระอิมักจะมีสัดส่วนของวงกลมที่มีความยาวต่อความสูงมากกว่า 1.1 เสมอ ดังในรูป 2.18 ก

2.3.5.2 ตรวจสอบสระอำ มีลักษณะเป็นวงกลมเดี่ยวในระดับบนของตัวอักษรเช่นเดียวกันกับสระอิ แต่สระอำมักจะมีสัดส่วนของวงกลมที่มีความยาวต่อความสูงประมาณ 1.1 หรือน้อยกว่า ดังในรูป 2.18 ข

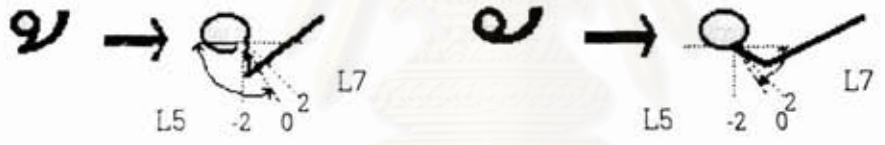
w/h >= 1.1 w/h < 1.1

ก. ลักษณะเฉพาะของสระอิ ข. ลักษณะเฉพาะของสระอำ

รูปที่ 2.18 การเปรียบเทียบลักษณะเฉพาะของสระอิ และสระอำ

2.3.5.3 ตรวจสอบไม้โท มีลักษณะเป็นหัววงกลมแล้วมีเส้นตรงต่อเชื่อมลงไปในแนวขวาล่าง เหมือนกับไม้หันอากาศ แต่แตกต่างที่ primitive ของเส้นตรงเป็น L5 ที่มีมุมเบี่ยงเบน -2 ถึง 1 ดังในรูป 2.19 ก

2.3.5.4 ตรวจสอบไม้หันอากาศ มีลักษณะเป็นหัววงกลมแล้วมีเส้นตรงต่อเชื่อมลงไปในแนวขวาล่าง เหมือนกับไม้โท แต่แตกต่างที่ primitive ของเส้นตรงเป็น L7 ที่มีมุมเบี่ยงเบน -2 ถึง 2 หรือ L6 ที่มีมุมเบี่ยงเบน 0 ถึง 2 ดังในรูป 2.19 ข



ก. ลักษณะเฉพาะของไม้โท ข. ลักษณะเฉพาะของไม้หันอากาศ

รูปที่ 2.19 การเปรียบเทียบลักษณะเฉพาะของไม้โท และไม้หันอากาศ

2.3.5.5 ตรวจสอบการันต์ มีลักษณะเป็นหัววงกลมแล้วมีเส้นตรงต่อเชื่อมขึ้นในแนวขวาบน เหมือนกับสระอี แต่แตกต่างที่ primitive ของวงกลมเป็น C2 ที่มีมุมเบี่ยงเบนของจุดเชื่อมต่อเป็น -2 ถึง 2 หรือ C1 ที่มีมุมเบี่ยงเบนของจุดเชื่อมต่อเป็น 0 ถึง 2 ดังในรูป 2.20 ก

2.3.5.6 ตรวจสอบสระอี มีลักษณะเป็นวงกลมแล้วมีเส้นตรงต่อเชื่อมขึ้นในแนวขวาบน เหมือนกับสระอี แต่แตกต่างที่ primitive ของวงกลมเป็น C1 ที่มีมุมเบี่ยงเบนของจุดเชื่อมต่อเป็น -2 ถึง -1 และมีสัดส่วนของความยาวต่อความสูงมากกว่า 1 เสมอ ดังในรูป 2.20 ข

2.3.5.1 ตรวจสอบไปรยาล (ๆ) เนื่องจากไปรยาลมีค่าระยะที่ไม่ต่างไปจากไม้ยมก (ๆ) มากนัก ไม้ยมกมีวิธีการตรวจสอบโดยตรวจสอบลักษณะหัวของตัวอักษรที่ต้องมี 1 หยัก ส่วนไปรยาล มีลักษณะเป็นหัววงกลมแล้วมีเส้นตรงต่อเชื่อมลงในแนวขวาล่าง มี primitive ของวงกลมเป็น C4 และมีเส้นตรงต่อเชื่อมที่เป็น primitive L6 หรือ L7

ก. ลักษณะเฉพาะของการันต์ ข. ลักษณะเฉพาะของสระอี

รูปที่ 2.20 การเปรียบลักษณะเฉพาะของการันตร์ และสระอี

3. การวิเคราะห์ผลการเปรียบเทียบ จากวิธีการเปรียบเทียบตัวอักษรที่ต้องการรู้จำกับตัวอักษรต้นแบบที่ผ่านมา ผลที่ได้คือค่าระยะและค่าความต่าง เพื่อใช้ในการพิจารณาเลือกตัวอักษรต้นแบบที่มีความคล้ายกับตัวอักษรที่ต้องการรู้จำมากที่สุดได้ โดย

3.1 พิจารณาค่าความต่างของตัวอักษรต้นแบบที่น้อยที่สุด เป็นอันดับแรก

3.2 พิจารณาค่าระยะของตัวอักษรต้นแบบที่มีค่าน้อยที่สุดเป็น อันดับต่อไป

3.3 ค่าความต่างที่ยอมรับได้จะต้องมีค่าไม่เกิน 4 ซึ่งได้จากตัวอักษรที่มีจำนวนจุดปลายที่อยู่ในระดับตัวอักษรเดียวกันต่างกันไม่เกิน 2 จุด หรือตัวอักษรที่มีจำนวนจุดปลายที่อยู่ต่างระดับตัวอักษรไม่เกิน 1 จุด

3.4 ค่าระยะที่ยอมรับได้จะต้องมีค่าไม่เกิน 15 ได้จากค่าสูงสุดของค่าระยะของตัวอักษรที่นำมาทดสอบกับตัวอักษรต้นแบบ แล้วให้ผลการเปรียบเทียบรู้จำได้

การปรับปรุงเวกเตอร์

จากการเปรียบเทียบที่ผ่านมา ถ้าผลการเปรียบเทียบยังไม่อยู่ในเกณฑ์ที่ยอมรับได้ คือมีค่าระยะมากกว่า 15 และต้องมีค่าความต่างมากกว่า 4 อาจเป็นไปได้ว่าข้อมูลภาพของตัวอักษรที่รับเข้ามามีรูปร่างที่ผิดไปจากต้นแบบ เช่น ส่วนหัวของที่เป็นวงกลมของตัวอักษรมีลักษณะไม่ครบวงก็จะมีผลให้รูปลักษณะของส่วนหัวผิดไปและเป็นผลให้จำนวนจุดปลายของตัวอักษรเพิ่มขึ้น หรือตัวอักษรมีเส้นขาดเป็นผลให้จำนวนส่วนย่อยมากขึ้น การเปรียบเทียบกับตัวอักษรต้นแบบจึงถูกนำไปเปรียบเทียบกับตัวอักษรที่มีจำนวนส่วนย่อยที่เท่าๆกันซึ่งอาจจะไม่มีตัวอักษรต้นแบบตามลักษณะที่ต้องการ หรือตัวอักษรมีลักษณะผิดไปบ้าง เช่นตัวอักษร ข,ป มีเส้นเล็กๆเพิ่มที่บริเวณมุมล่างซ้ายของตัวอักษรทำให้จำนวนจุดปลายของตัวอักษรเพิ่มขึ้น เป็นผลให้ค่าระยะของการเปรียบเทียบตัวอักษรเพิ่มมากขึ้นจนอาจทำให้ผลการรู้จำผิดพลาดได้ ดังนั้นการปรับปรุงเวกเตอร์ จึงมีหน้าที่ในการนำเวกเตอร์ของตัวอักษรที่ต้องการรู้จำมาพยายามค้นหาสิ่งที่คาดว่าจะเป็นสิ่งผิดปกติเหล่านี้แล้วแก้ไขให้ได้ใกล้เคียงกับลักษณะของตัวอักษรที่เป็นต้นฉบับมากที่สุด

ในขบวนการรู้จำนั้น การปรับปรุงเวกเตอร์จะถูกกระทำทุกครั้งหลังจากการเปรียบเทียบตัวอักษรสิ้นสุดลง ถ้าค่าระยะและค่าความต่างยังไม่เป็นค่าที่ยอมรับได้ เมื่อปรับปรุงเวกเตอร์ได้ก็จะนำเวกเตอร์ที่ได้ไปทำการเปรียบเทียบกับตัวอักษรต้นแบบ จนกว่าจะได้ค่าระยะและค่าความต่างที่เป็นค่าที่ยอมรับได้ หรือจนกว่าจะไม่สามารถปรับปรุงเวกเตอร์ได้แล้ว การปรับปรุงเวกเตอร์แบ่งเป็น 2 ประเภทคือ

1. การเชื่อมเวกเตอร์ที่อยู่ใกล้กัน ถ้ามีเวกเตอร์เส้นตรงที่เป็นจุดปลายของตัวอักษร 2 เวกเตอร์ที่อยู่ใกล้กันมากที่สุด ที่มีระยะห่างกันของจุดปลายไม่เกิน 7% ของขนาดของกรอบเวกเตอร์ มีโอกาสที่จะเชื่อมเป็นเวกเตอร์เส้นตรงเส้นเดียวกันได้ การเชื่อมเวกเตอร์ให้พิจารณาดังนี้

1.1 ถ้าเวกเตอร์ทั้ง 2 เป็นจุดปลายของส่วนย่อยเดียวกัน หรือกล่าวได้ว่าเวกเตอร์ทั้ง 2 อยู่ในต้นไม้เดียวกัน ให้พิจารณาดังนี้

1.1.1 ถ้าเวกเตอร์ที่เป็นจุดปลายทั้งสองสามารถเชื่อมถึงกันได้ และทุกๆ เวกเตอร์ที่เชื่อมต่อระหว่างเวกเตอร์นั้นไม่ปรากฏกิ่งแยกออกไป แสดงว่าเมื่อเชื่อมเวกเตอร์ทั้งสองเข้าด้วยกันแล้วจะได้เป็น เวกเตอร์วงกลมเดี่ยวที่ไม่มีจุดเชื่อมต่อไปยังเวกเตอร์ใดๆ

1.1.2 ถ้าเวกเตอร์ที่เป็นจุดปลายทั้งสองสามารถเชื่อมถึงกันได้ และระหว่างเส้นทางที่ต่อเชื่อมกันปรากฏกิ่งแยกออกไปต่อเชื่อมกับเวกเตอร์อื่นๆ 1 กิ่ง แสดงว่าเมื่อเชื่อมเวกเตอร์ทั้งสองเข้าด้วยกันแล้วจะได้เป็น เวกเตอร์วงกลมที่มีจุดเชื่อมต่อไปยังเวกเตอร์อื่นด้วย

1.1.3 ถ้าเวกเตอร์ที่เป็นจุดปลายทั้งสองสามารถเชื่อมถึงกันได้ และระหว่างเส้นทางที่ต่อเชื่อมกันปรากฏกิ่งแยกออกไปต่อเชื่อมกับเวกเตอร์อื่นๆมากกว่า 1 กิ่ง ให้นำเวกเตอร์ที่เป็นจุดปลายอันใดอันหนึ่งที่ไม่เป็นรากของต้นไม้มาขยายจุดปลายให้มีตำแหน่งเป็นจุดเดียวกับตำแหน่งจุดปลายของเวกเตอร์อีกอันหนึ่ง แต่ไม่มีการต่อเชื่อมกันใดๆทั้งสิ้น

ก. ไม่มีกิ่งแยก ข. มีกิ่งแยก 1 กิ่ง ค. มีกิ่งแยกมากกว่า 1 กิ่ง

รูปที่ 2.21 การเชื่อมเวกเตอร์ที่มีจุดปลายอยู่ในต้นไม้เดียวกัน

1.2 ถ้าเวกเตอร์ทั้ง 2 เป็นจุดปลายอยู่ต่างส่วนย่อยกัน แสดงว่าเวกเตอร์ทั้ง 2 อยู่ต่างต้นไม้กันและมีตำแหน่งรากที่เป็นของตนเอง ต้องรวมทั้งสองต้นไม้เข้าด้วยกันเป็นต้นเดียว โดยการเชื่อมต่อจะต้องพิจารณาทิศทางและรากของเวกเตอร์ การเชื่อมต่อให้พิจารณาตามประเภทของเวกเตอร์ดังนี้

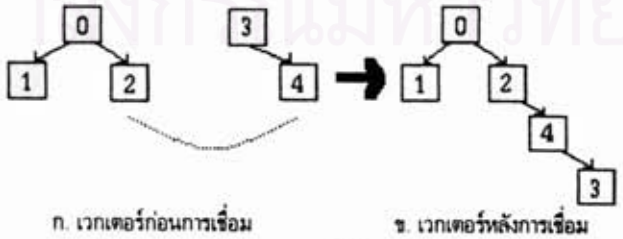
1.2.1 ถ้าเวกเตอร์ที่ต้องการนำมาเชื่อมกันทั้งสองเป็นจุดปลายของต้นไม้ทั้งคู่ ให้ทำการเชื่อมเข้าด้วยกันดัง algorithm 6. ผลการเชื่อมแสดงในรูปที่ 2.22

1.2.2 ถ้าเวกเตอร์ที่ต้องการนำมาเชื่อมกันทั้งสองเป็นจุดปลายอันหนึ่ง และเป็นรากอันหนึ่ง ให้ทำการเชื่อมเข้าด้วยกันดัง algorithm 7. ผลการเชื่อมแสดงในรูปที่ 2.23

1.2.3 ถ้าเวกเตอร์ที่ต้องการนำมาเชื่อมกันทั้งสองเป็นรากของต้นไม้ทั้งคู่ ให้ทำการเชื่อมเข้าด้วยกันดัง algorithm 8. ผลการเชื่อมแสดงในรูปที่ 2.24

Algorithm 6. การเชื่อมเวกเตอร์ที่เป็นจุดปลายทั้งคู่

1. เปลี่ยนเวกเตอร์อันใดอันหนึ่งให้เป็นรากของต้นไม้
2. นำรากที่ได้ไปเชื่อมกับจุดปลายอีกจุดที่เหลือ
3. ลบจุดปลายที่ถูกเปลี่ยนให้เป็นรากออกจากตารางตำแหน่งจุดปลายของตัวอักษร
4. จัดเรียงจุดปลายที่อยู่ในตารางตำแหน่งจุดปลายของตัวอักษร
5. เลือกจุดปลายตำแหน่งแรกสุดของตารางเป็นรากของต้นไม้
5. ลดจำนวนส่วนย่อยของตัวอักษรลง 1
6. จัดเรียงทิศทางของเวกเตอร์ในต้นไม้

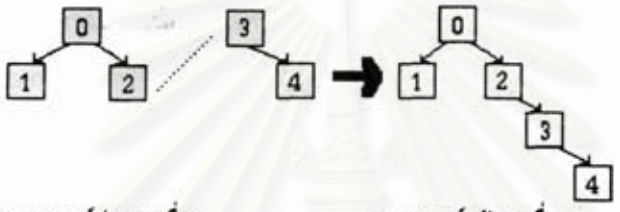


ก. เวกเตอร์ก่อนการเชื่อม ข. เวกเตอร์หลังการเชื่อม

รูปที่ 2.22 การเชื่อมเวกเตอร์ที่อยู่ต่างต้นไม้กัน และเวกเตอร์ทั้งคู่เป็นจุดปลาย

Algorithm 7. การเชื่อมเวกเตอร์ที่เป็นจุดปลายอันหนึ่ง และอีกอันเป็นราก

1. ลบเวกเตอร์ที่เป็นรากออกจากตารางตำแหน่งราก และจุดปลายของตัวอักษร
3. นำรากของเวกเตอร์หนึ่งไปเชื่อมกับจุดปลายของอีกเวกเตอร์หนึ่ง
4. ลดจำนวนส่วนย่อยของตัวอักษรลง 1
5. จัดเรียงทิศทางของเวกเตอร์ในต้นไม้

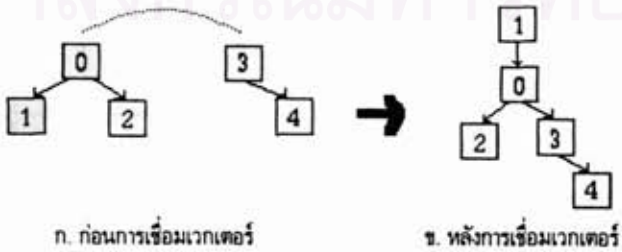


ก. เวกเตอร์ก่อนการเชื่อม ข. เวกเตอร์หลังการเชื่อม

รูปที่ 2.23 การเชื่อมเวกเตอร์ที่อยู่ต่างต้นไม้กัน โดยเวกเตอร์ทั้งสองต่างชนิดกัน

Algorithm 8. การเชื่อมเวกเตอร์ที่เป็นรากทั้งคู่

1. เลือกรากของต้นไม้ต้นใดต้นหนึ่งเปลี่ยนให้เป็นจุดปลาย
2. หาจุดปลายจุดใดจุดหนึ่งในต้นไม้ต้นเดียวกัน กับ step 1 มาเป็นรากแทน
2. นำจุดปลายที่ได้ไปเชื่อมต่อกับรากของอีกเวกเตอร์ที่เหลือ
3. ลบรากของทั้งสองเวกเตอร์ออกจากตารางตำแหน่งราก และตำแหน่งจุดปลาย
4. ลดจำนวนส่วนย่อยของตัวอักษรลง 1
5. จัดเรียงทิศทางของเวกเตอร์ในต้นไม้



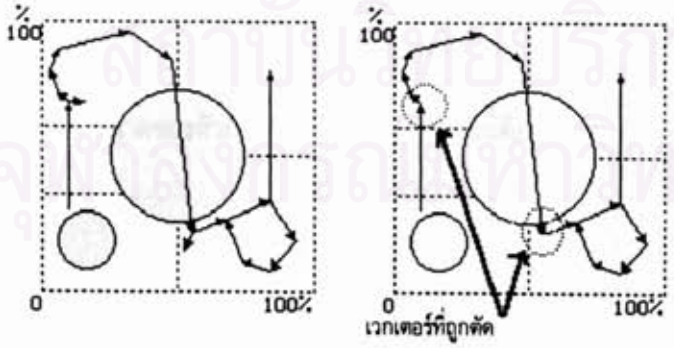
ก. ก่อนการเชื่อมเวกเตอร์ ข. หลังการเชื่อมเวกเตอร์

รูปที่ 2.24 การเชื่อมเวกเตอร์ที่อยู่ต่างต้นไม้กัน และเวกเตอร์ทั้งคู่เป็นรากของต้นไม้

2. การตัดเวกเตอร์ส่วนเกินออก เวกเตอร์ส่วนเกินก็คือเวกเตอร์เส้นตรงที่เป็นจุดปลายของตัวอักษรและมีความยาวของเวกเตอร์ไม่เกิน 15% ของขนาดของกรอบเวกเตอร์ การตัดจะตรวจหาเวกเตอร์ที่สั้นที่สุดที่ต่อเชื่อมกับจุดทางแยก และจะตัดเวกเตอร์นี้ออกก่อนเวกเตอร์ที่สั้นที่สุดที่ต่อกับเวกเตอร์อื่นเป็นเส้นตรง ทั้งนี้เพราะเวกเตอร์ที่ต่อกับจุดทางแยกมีโอกาสเป็นเส้นกิ่งขนาดเล็กที่เป็นส่วนเกินของตัวอักษร การตัดเวกเตอร์ส่วนเกินจะตัดออกครั้งละ 1 จุดเท่านั้น การตัดดังใน algorithm 9

Algorithm 9. การตัดเวกเตอร์ส่วนเกิน

1. If พบเวกเตอร์จุดปลายที่มีขนาดสั้นที่สุดที่ต่อกับจุดทางแยก และมีความยาวน้อยกว่าที่กำหนดไว้
   then
      ทำ step 4
   end if
2. If พบเวกเตอร์จุดปลายที่มีความยาวน้อยกว่าที่กำหนดไว้
   then
      ทำ step 4
   end if
3. จบการทำงานแบบไม่สามารถตัดเวกเตอร์ส่วนเกินได้
4. ตัดเวกเตอร์ที่เป็นส่วนเกินออก
5. จัดเรียงทิศทางต้นไม้
6. จบการทำงานแบบตัดเวกเตอร์ส่วนเกินได้



ก. ก่อนการตัดเวกเตอร์ส่วนเกิน ข. หลังการตัดเวกเตอร์ส่วนเกิน

รูปที่ 2.25 การตัดเวกเตอร์ส่วนเกิน

บทที่ 3

การทดสอบการรู้จำตัวอักษรพิมพ์ภาษาไทย

ที่มาของข้อมูล

ข้อมูลที่ใช้ทดสอบการรู้จำตัวอักษรพิมพ์ภาษาไทย เป็นข้อมูลภาพของตัวอักษรที่เก็บไว้ในไฟล์ข้อมูลเก็บแยก 1 ตัวอักษรต่อ 1 ไฟล์ โดยรูปแบบตัวอักษรที่เลือกใช้ในการศึกษาครั้งนี้ เลือกไว้ 2 รูปแบบ ชื่อ EUCROSIA และ CORDIA ตัวอักษรที่ใช้สำหรับการทดสอบก็มี พยัญชนะ 44 ตัว สระ, วรรณยุกต์ และ ตัวอักษรพิเศษ ทั้งหมด 25 ตัว รวมเป็น 69 ไฟล์ต่อ 1 ชุดตัวอักษร ข้อมูลทั้งหมดได้มาจาก 2 แหล่ง คือ

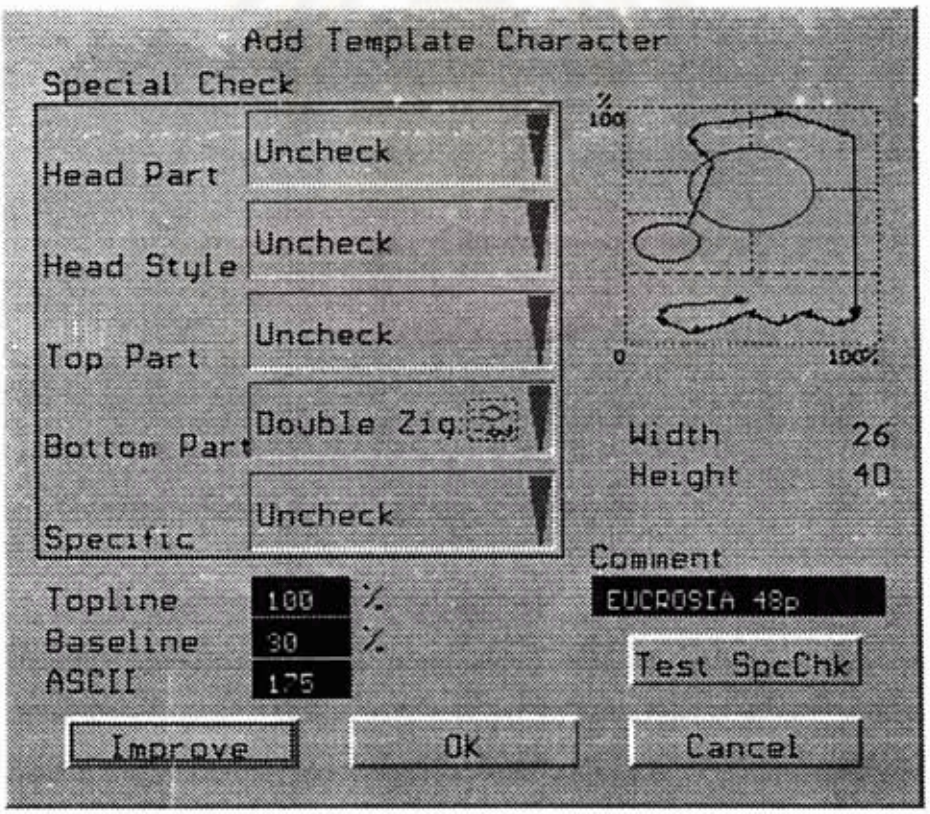
1. สร้างตัวอักษรจากโปรแกรม Paintbrush ของ Microsoft Windows 3.1 เนื่องจากโปรแกรมนี้สามารถพิมพ์ตัวอักษรแล้วเก็บข้อมูลภาพของตัวอักษรลงไฟล์ข้อมูลภาพตระกูล BMP หรือ PCX ได้โดยตรง ข้อมูลภาพที่ได้จากวิธีนี้เป็นข้อมูลภาพที่สมบูรณ์มาก เหมาะสำหรับใช้เป็นตัวอักษรต้นแบบ ในการทดลองครั้งนี้เลือกใช้ขนาด 48 point

2. ป้อนตัวอักษรที่ต้องการทุกรูปแบบและทุกขนาด โดยใช้โปรแกรม Word for Windows 2.0c แล้วพิมพ์ออกมาทางเครื่องพิมพ์เลเซอร์ ที่ความละเอียดขนาด 600 dpi นำเอกสารที่ได้ไปทำการอ่านกลับมาเป็นไฟล์ข้อมูลภาพด้วยเครื่องสแกนเนอร์ ที่ความละเอียดขนาด 300 dpi จากนั้นใช้โปรแกรม Paintbrush ของ Microsoft Windows 3.1 อ่านไฟล์ข้อมูลภาพที่ได้ แล้วตัดตัวอักษรเป็นตัวอักษรเดี่ยวๆ เก็บลงไฟล์ข้อมูลภาพ 1 ไฟล์ต่อ 1 ตัวอักษร ข้อมูลตัวอักษรที่เก็บจะถูกแยกตามรูปแบบและขนาดของตัวอักษรเช่นเดียวกับวิธีแรก ขนาดของตัวอักษรที่เลือกใช้สำหรับวิธีนี้คือ 20, 22, 24, 28, 32 และ 36 point

การสร้างตัวอักษรต้นแบบ

สิ่งที่สำคัญที่สุดของการรู้จำตัวอักษรพิมพ์ภาษาไทยคือ การรวบรวมตัวอักษรต้นแบบ เพื่อใช้ในการเปรียบเทียบ ตัวอักษรที่เป็นต้นแบบจะต้องสามารถครอบคลุมลักษณะของตัวอักษรที่ต้องการรู้จำให้ได้มากประมาท และมากรูปแบบที่สุด รวมถึงต้องให้ครอบคลุมกรณีต่างๆที่เป็นไปได้ เช่นหัวของตัวอักษรที่มีทั้งเป็นวงกลม และไม่เป็นวงกลม, หัวตัวอักษรอยู่นอกเขตที่ควรจะเป็น ซึ่งโดยมากจะเกิดจากตัวอักษรที่มีส่วนหัวอยู่ในเขต 0 และเขต 5 เช่น ฉ อ ฐ เป็นต้น

ในการทดลองครั้งนี้ เลือกใช้ตัวอักษร EUCROSIA ขนาด 48 point เป็นตัวอักษรต้นแบบหลัก และมีตัวอักษรที่มีรูปแบบและขนาดอื่นๆรวมอยู่ด้วยเป็นบางส่วน เพื่อให้สามารถครอบคลุมกรณีต่างๆที่กล่าวมาข้างต้น รวมเป็นตัวอักษรที่เป็นต้นแบบทั้งหมด 101 ตัวอักษร



รูปที่ 3.1 หน้าจอของการสร้างตัวอักษรต้นแบบ

การสร้างตัวอักษรต้นแบบ สามารถระบุ และปรับแต่งข้อมูลได้บ้าง ดังนี้

1. การตรวจสอบทาง feature ของตัวอักษรต้นแบบ มี 5 รูปแบบคือ

1.1 การเปรียบเทียบประเภทของหัวตัวอักษร

1.2 การเปรียบเทียบส่วนหัวของตัวอักษร

1.3 การเปรียบเทียบส่วนบนของตัวอักษร

1.4 การเปรียบเทียบส่วนล่างของตัวอักษร

1.5 การเปรียบเทียบลักษณะเฉพาะตัวของตัวอักษร

2. กำหนดเส้นระดับบน และระดับล่างของตัวอักษร เพื่อกำหนดระดับภาษาไทยให้กับตัวอักษร เช่น

2.1 ตัวอักษร ข มีเส้นระดับบนเป็น 100% และเส้นระดับล่างเป็น 0 %

2.2 ตัวอักษร ฏ มีเส้นระดับบนเป็น 100% และเส้นระดับล่างเป็น 30 %

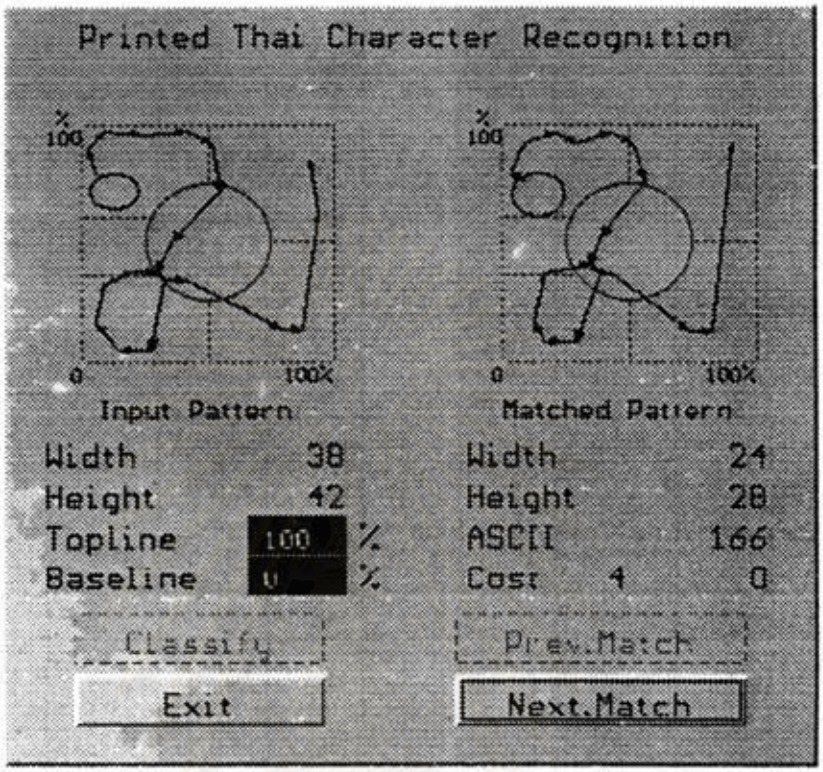
2.3 ตัวอักษร ฟ มีเส้นระดับบนเป็น 80% และเส้นระดับล่างเป็น 0 %

2.4 ไม้โท มีเส้นระดับบนเป็น 10% และเส้นระดับล่างเป็น 0 %

2.5 สระอุ มีเส้นระดับบนเป็น 100% และเส้นระดับล่างเป็น 90 %

3. ระบุรหัสแอสกี และ รหัสภาษาไทย สมอ. ให้ตัวอักษร

4. ในกรณีที่ตัวอักษรปรากฏส่วนเกินของตัวอักษร ในลักษณะเป็นกิ่งเล็กๆเกินออกมาตามมุมต่างๆของตัวอักษรที่เป็นควรกำตัดเส้นส่วนเกินเหล่านี้ออก โดยใช้การปรับปรุงคุณภาพข้อมูลก่อนที่จะเก็บข้อมูลนั้นเป็นตัวอักษรต้นแบบ ทั้งนี้เพราะถ้าเก็บข้อมูลทั้งๆที่มีส่วนเกินของตัวอักษรเป็นตัวอักษรต้นแบบแล้ว เมื่อนำไปเปรียบเทียบกับตัวอักษรเหมือนกันแต่ไม่มีส่วนเกินของตัวอักษร อาจทำให้ผลการรู้จำผิดพลาดได้ แต่ถ้าตัวอักษรต้นแบบถูกกำจัดส่วนเกินออกไปแล้ว เมื่อนำไปเปรียบเทียบกับตัวอักษรที่มีส่วนเกิน ผลที่ได้ในครั้งแรกจะไม่ถูกต้อง แต่โปรแกรมจะจัดการกำจัดส่วนเกินของตัวอักษรออกให้ได้ เป็นผลให้การรู้จำถูกต้อง และสามารถใช้ตัวอักษรต้นแบบเดียวกัน ไปใช้กับตัวอักษรทั้งที่มีส่วนเกิน และไม่มีส่วนเกินได้อีกด้วย



รูปที่ 3.2 หน้าจอของการรู้จำทีละตัวอักษร

วิธีการทดสอบ

1. การทดสอบทีละตัวอักษร เป็นการอ่านไฟล์ข้อมูลภาพตัวอักษรครั้งละ 1 ตัวอักษร จากนั้นจะต้องระบุค่าระดับบน และ ระดับล่างของตัวอักษรก่อนการรู้จำทุกครั้ง ผลที่ได้จะแสดงตัวอักษรต้นแบบที่รู้จำได้ รวมถึง รหัสแอสกี, ค่าระยะ, ค่าความต่าง ที่ได้จากการรู้จำ และสามารถแสดงตัวอักษรต้นแบบอื่นๆที่ถูกนำมาเปรียบในการรู้จำครั้งนั้นด้วย การทดสอบแบบนี้เพื่อต้องการหาข้อผิดพลาดของการรู้จำ หรือของตัวอักษรต้นแบบ

2. การทดสอบกลุ่มตัวอักษร เป็นการทดสอบเพื่อให้ได้ผลของการทำงานโดยรวมของระบบ ก่อนทำการทดสอบจะต้องป้อนชื่อของไฟล์, ค่าระดับบน, ระดับล่างของ และรหัสแอสกีที่ต้องการของตัวอักษรที่ต้องการรู้จำทั้งหมด ผลที่ได้จะเป็นรายงานสรุปการรู้จำ โดยจะแสดงค่าระยะ, ค่าความต่าง, รหัสแอสกีที่ได้, เวลาที่ใช้ นอกจากนี้ยังสรุปค่าความถูกต้องของการรู้จำเป็นเปอร์เซนต์

ผลการทดสอบ

ชื่อรูปแบบตัวอักษร	ขนาด (point)	จำนวน	รู้จำได้	ไม่รู้จำ	รู้จำผิด	ความถูกต้อง (%)	เวลาเฉลี่ย (วินาที)
Eucrosia	20	69	65	1	3	94.20	0.92
	22	69	66	2	1	95.65	0.92
	24	69	68	0	1	98.55	0.95
	28	69	67	1	1	97.10	1.15
	32	69	68	1	0	98.55	1.24
	36	69	67	1	1	97.10	1.26
	*48	69	69	0	0	100.00	0.75
	รวม	483	470	6	7	97.31	1.03
Cordia	20	69	66	2	1	95.65	1.22
	22	69	68	0	1	98.55	1.18
	24	69	67	1	1	97.10	1.05
	28	69	68	0	1	98.55	1.15
	32	69	67	1	1	97.10	1.34
	36	69	64	2	3	92.75	1.21
	48	69	65	0	4	94.20	0.94
	รวม	483	465	6	12	96.27	1.16
	รวม	966	935	12	19	96.79	1.09

* ตัวอักษรต้นแบบ

ตาราง 3.1 ผลการทดสอบการรู้จำ

เครื่องคอมพิวเตอร์ที่ใช้ทดสอบ ใช้เครื่องไมโครคอมพิวเตอร์ IBM PS/ValuePoint รุ่น 433SX/S 80486sx ความเร็ว 33 เมกกะเฮิรตซ์ในการทดสอบ ได้ผลการทดสอบตามตารางที่ 3.1 มีอัตราการรู้จำ 96.79% จากจำนวนตัวอักษรทั้งหมด 966 ตัวอักษร โดยมีการรู้จำผิด 1.96% และไม่สามารถรู้จำได้ 1.24% เวลาที่ใช้ในการรู้จำเฉลี่ย 1.09 วินาทีต่อ 1 ตัวอักษร โดยเวลาที่ใช้จะแปรผันตรงกับจำนวนครั้งของการปรับปรุงเวกเตอร์, จำนวนตัวอักษรต้นแบบที่ถูกนำไปเปรียบเทียบ และความซับซ้อนของเวกเตอร์ของทั้งตัวอักษรที่ต้องการรู้จำและของตัวอักษรต้นแบบ

ปัญหาและข้อจำกัด

ข้อมูลที่นำมาทดสอบที่ได้จากเครื่องสแกนเนอร์ ความละเอียดของจุดภาพมีผลต่อการรู้จำเป็นอย่างมาก ถ้าใช้ที่ความละเอียด 300 dpi จะมีผลต่อตัวอักษรที่มีขนาดเล็กกว่า 22 point โดยเส้นของตัวอักษรจะเลอะ และบางครั้งมีการติดกันกับเส้นข้างเคียง หรือ ทำให้หัวตัวอักษรที่เป็นช่องว่างเต็ม เป็นผลให้รูปโครงร่างของตัวอักษรหลังการทำให้บางผิดไปเป็นอย่างมาก ดังนั้นถ้าต้องการรู้จำตัวอักษรที่ขนาดต่ำกว่า 20 point จึงควรใช้สแกนเนอร์ที่มีความละเอียด 600 dpi จะช่วยแก้ปัญหาได้มาก

การทำตัวอักษรให้บาง ก็มีผลกับตัวอักษรประเภทที่มีบางส่วนของตัวอักษรเป็นเส้นหยักมากพอสมควร เช่น ตัวอักษร ข, ค, ฆ, ช, ฑ, ฏ, ฐ เป็นต้น ในตัวอักษรขนาดเล็กกว่า 22 point เส้นหยักของตัวอักษรประเภทนี้จะกลายเป็นเส้นตรงเมื่อผ่านกระบวนการทำตัวอักษรให้บาง จากการสังเกตุจากไฟล์ข้อมูลของตัวอักษรพบว่า ที่บริเวณหยักของภาพจะเห็นว่ามีการเว้าหรือนูนของเส้นออกจากเส้นปกติของตัวอักษรเพียง 1 จุดภาพเท่านั้น ทำให้เมื่อผ่านการทำให้บางแล้วปรากฏว่าได้ออกมาเป็นเส้นตรง ดังนั้น ตัวอักษร ข ก็จะถูกรู้จำเป็นตัวอักษร ข เป็นต้น

สระอือ มีอัตราการรู้จำได้ต่ำที่สุด คือสามารถรู้จำได้เพียง 5 ตัวของจำนวนสระอือทั้งหมด 14 ตัว ทั้งนี้เพราะถ้าความสูงของเส้นตรงเส้นด้านซ้ายบนสุดของสระอือมีความสูงของเส้นไม่เกิน 4 จุดภาพ หลังการทำให้บางเส้นตรงเส้นนี้จะหายไป หรือเหลือเพียงหยักนูนขึ้นด้านบนเพียงเล็กน้อย และเหลือเพียงเส้นตรงที่อยู่ด้านขวามือเพียงเส้นเดียวเท่านั้น ซึ่งเป็นผลให้สระอือถูกรู้จำเป็นสระอีทันที

ไม้โท และไม้หันอากาศ หลังการทำตัวอักษรให้บางแล้วจะมีลักษณะใกล้เคียงกันมาก แม้จะใช้การเปรียบเทียบทาง feature เข้าช่วยในการจำแนกก็ไม่สามารถจำแนกได้อย่างถูกต้อง 100% ทั้งนี้เพราะมุมของเส้นตรงที่ต่อออกมาจากหัวของตัวอักษรที่ใช้เป็นตัวพิจารณาในการจำแนกไม้โทและไม้หันอากาศนั้นบางครั้งทำมุมไม่แตกต่างกัน ดังนั้นการรู้จำไม้โทและไม้หันอากาศของการวิจัยในครั้งนี้จึงรู้จำได้เฉพาะที่มีลักษณะแตกต่างกันอย่างเด่นชัดเท่านั้น

ตัวอักษร ส กับ ศ ก็มีปัญหาในการรู้จำ โดยเฉพาะ ศ ทั้งนี้เพราะว่า ถ้าตัวอักษรทั้งสองถ้านำไปปรับปรุงเวกเตอร์ เนื่องจากการรู้จำครั้งแรกให้ผลการรู้จำไม่เป็นที่น่าพอใจแล้ว ผลที่ได้หลังการปรับปรุงเวกเตอร์ก็คือ ส่วนหางของ ส และ ศ จะเป็นส่วนของเวกเตอร์ที่สั้นที่สุดที่จะต้องถูกตัดออก เป็นผลให้ ส กลายเป็น ล และ ศ กลายเป็น ค ในทันที ทำให้ผลการรู้จำผิดไปอย่างมาก

ตาราง 3.2 ตัวอักษรที่รู้จำผิด หรือไม่สามารถรู้จำได้

ตัวอักษร	Eucrosia		Cordia	
	รู้จำผิด	ไม่รู้จำ	รู้จำผิด	ไม่รู้จำ
ซ	2	-	-	-
ม	-	1	-	-
ฉ	-	-	1	-
ช	-	-	2	1
ฏ	-	1	-	-
ฐ	-	1	-	2
ถ	-	-	1	-
น	-	-	1	-
ศ	1	-	2	2
ส	1	-	-	-
ห	-	-	-	1
สระ อี	-	-	1	-
สระ อื	-	-	2	-
สระ อื	2	3	-	-
ไม้ เอก	-	-	1	-
ไม้ โท	1	-	1	-
	7	6	13	6

บทที่ 4

ข้อสรุปและข้อเสนอแนะ

ข้อสรุป

การศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก ได้ใช้วิธีการหลักๆอยู่ 2 ขั้นตอนในการจำแนกตัวอักษร คือ การหาค่าระยะระหว่างต้นไม้ของ primitive และการเปรียบเทียบตัวอักษรทาง feature ระหว่างตัวอักษรต้นแบบกับตัวอักษรที่ต้องการรู้จำ โดยขั้นตอนแรกจะช่วยจำแนกตัวอักษรที่มีความแตกต่างกันมากๆออกจากกันได้ ส่วนขั้นตอนที่สองจะช่วยจำแนกตัวอักษรที่มีลักษณะคล้ายคลึงกัน โดยนำเอาลักษณะเด่นของตัวอักษรมาทำการเปรียบเทียบ เป็นผลให้การรู้จำทำได้อย่างถูกต้องและแม่นยำมากยิ่งขึ้น ผลการรู้จำจากตัวอย่างตัวอักษร 2 รูปแบบ ๆ ละ 6 ขนาดตัวอักษร รวมเป็นตัวอักษรทั้งสิ้น 966 ตัวอักษรพบว่ามีอัตราการรู้จำ 96.79% ใช้เวลาในการรู้จำเฉลี่ยต่อตัวอักษร 1.09 วินาที เวลาที่ใช้ไปส่วนมากในขบวนการรู้จำก็คือ การจำแนกตัวอักษร ซึ่งเป็นการอ่านตัวอักษรต้นแบบที่เก็บไว้ใน harddisk ของเครื่องคอมพิวเตอร์ บางครั้งการจำแนกตัวอักษรต้องทำมากกว่า 1 ครั้งเพื่อปรับปรุงคุณภาพของเวกเตอร์จนกว่าผลการรู้จำจะเป็นที่น่าพอใจ

การเก็บตัวอักษรต้นแบบเพื่อการรู้จำ เป็นตัวแปรที่สำคัญอย่างยิ่งในการช่วยให้การรู้จำทำได้อย่างถูกต้องและรวดเร็ว ถ้าสามารถทำให้ตัวอักษรต้นแบบ 1 ตัว สามารถใช้ได้กับตัวอักษรหลายรูปแบบหรือหลายขนาด ก็จะเป็นการทำให้ตัวอักษรต้นแบบที่ต้องเก็บมีจำนวนลดลง การวิเคราะห์ทาง feature ก็เช่นกัน ถ้าเลือกใช้กับตัวอักษรต้นแบบเฉพาะกรณีที่จำเป็นก็จะช่วยให้ระยะเวลาที่ใช้ในการรู้จำลดลงได้มาก

การเลือกใช้การเก็บตัวอักษรต้นแบบในการศึกษาครั้งนี้ ทำให้ระบบมีความยืดหยุ่นมากในการนำไปใช้กับตัวอักษรแบบอื่นๆ เพราะเพียงแต่เก็บตัวอักษรที่มีรูปแบบใหม่ๆเข้าไป โดยแทบจะไม่ต้องแก้ไขส่วนหนึ่งส่วนใดของโปรแกรมเลย หรือถ้าจำเป็นก็เพียงแต่เพิ่มเงื่อนไขในการเปรียบเทียบเป็นกรณีพิเศษเข้าไปก็จะทำให้การรู้จำทำได้อย่างถูกต้องมากยิ่งขึ้น

ข้อเสนอแนะ

การรู้จำตัวอักษรภาษาไทยยังไม่ได้รับความสนใจเท่าที่ควร เท่าที่ศึกษางานวิจัยที่ผ่านมาพบว่าการศึกษาทางด้านนี้ยังไม่เป็นที่แพร่หลายมากนัก การศึกษาในครั้งนี้แสดงให้เห็นว่าสามารถที่จะรู้จำตัวอักษรพิมพ์ภาษาไทยที่มีรูปแบบของตัวอักษรตามที่กำหนดไว้ได้ แต่รูปแบบของตัวอักษรที่สามารถรู้จำได้มีจำนวนรูปแบบที่น้อยมาก เมื่อเทียบกับรูปแบบของตัวอักษรภาษาไทยอื่นๆที่มีใช้กันทั่วไปรวมไปถึงตัวอักษรที่เป็นลายมือเขียนด้วย การพัฒนาในครั้งต่อๆไปน่าจะนำเอาเทคนิคของ Fuzzy Logic เข้ามาช่วยในขั้นตอนการการทำให้บาง หรือการจำแนกตัวอักษร หรือนำ Neural Network เข้ามาช่วยในการรู้จำ หรือการเลือกตัวอักษรต้นแบบให้มีลักษณะไม่แตกต่างจากกลุ่มตัวอักษรที่ต้องการมากนัก เพื่อให้มีการเรียนรู้ตัวอักษรหลายๆรูปแบบ และสามารถตัดสินใจได้ดียิ่งขึ้น และถ้ามีการพัฒนาการรู้จำตัวอักษรภาษาไทยให้สามารถรู้จำลายมือเขียนในลักษณะทำงานแบบ on-line ก็จะเป็นประโยชน์อย่างยิ่งและสามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง เช่นนำไปใช้กับเครื่องช่วยเหลือส่วนบุคคลแบบดิจิตอล (Personal Digital Assistant) กำลังได้รับการพัฒนาอย่างเต็มที่ในต่างประเทศเพื่อให้สามารถรับรู้ลายมือมนุษย์ได้

รายการอ้างอิง

1. K.S.Fu. Sequential Methods in Pattern Recognition and Machine Learning. New York: Academic Press, 1968.

2. K.S.Fu. Syntactic Methods in Pattern Recognition. New York: Academic Press, 1974.

3. Shin-Yee Lu. A Tree-to-Tree Distance and its Application to Cluster Analysis. IEEE Trans. Pattern Anal. Mach. Intell. (April 1979): 219 - 224.

4. Lousia Lam and Ching Y. Suen, Structural Classification and Relaxation Matching of Totally Unconstrained Handwrittend Zip-code Numbers. Pattern Recognition (1988): 19-31.

5. สุรพันธ์ เอื้อไพบูลย์. การจดจำลายมือเขียนภาษาไทยโดยพิจารณาหัวของตัวอักษร. วิทยานิพนธ์ปริญญามหาบัณฑิต สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2531.

6. พิพัฒน์ หิรัณย์วณิชากร และ มนลดา บุญสุวรรณ. การรู้จำอักขระไทยหลายรูปแบบโดยวิธีไดนามิคโปรแกรมมิ่ง. สถาบันพัฒนบริหารศาสตร์.

7. ยศ พนัสสรณ์. รวมหลักภาษาไทย ม.ต้น. กรุงเทพ: สำนักพิมพ์แม็ค, 2531.

8. อนันต์ เอกวงศ์วิริยะ. การรู้จำตัวเลขพิมพ์ภาษาไทยโดยวิธีซินแทกติก. วิทยานิพนธ์ปริญญามหาบัณฑิต บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2537.

8. Theo Pavlidis. Algorithms for Graphics and Image Processing. Marry Land: Computer Science Press, 1982.

9. Takeshi Agui, Masayuki Nakajima, Tar K. Kim and Eduardo T. Takatta Shi. A Method of Recognition and Representation of Korean Characters by Tree Grammars. IEEE Transactions on Pattern Analysis and Machine Intelligence. (July 1979): 245-250.

10. Kameswara Rao, Kenneth Black. Type Classification of Fingerprints: A Synctactic Approch. IEEE Transactions on Pattern Analysis and Machine Intelligence. (May 1980): 223-231.

11. Robert L. Kruse. Data Structures and Program Design. 2nd Edition. New Jersey: Prentice-Hall, 1987.

12. David C. Kay and John R. Levine. Graphics File Formats. First Edition. McGraw-Hill, 1992.

ภาคผนวก

ภาคผนวก ก

ตัวอักษรต้นแบบ Eucrosia

1. ขนาด 20 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ
ทธนบปผฝพฟภมยรลวศษสหฬอฮ
ะาิีึืุูเ่้๊๋็ั์
เโไใำฤๅฦๆฯ

2. ขนาด 22 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ
ทธนบปผฝพฟภมยรลวศษสหฬอฮ
ะาิีึืุูเ่้๊๋็ั์
เโไใำฤๅฦๆฯ

3. ขนาด 24 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ
ทธนบปผฝพฟภมยรลวศษสหฬอฮ
ะาิีึืุูเ่้๊๋็ั์
เโไใำฤๅฦๆฯ

4. ขนาด 28 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ
ทธนบปผฝพฟภมยรลวศษสหฬอฮ
ะาิีึืุูเ่้๊๋็ั์
เโใไอำฤๅฦๆฯ

5. ขนาด 32 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ
ทธนบปผฝพฟภมยรลวศษสหฬอฮ
ะาิีึืุูเ่้๊๋็ั์
เโใไอำฤๅฦๆฯ

6. ขนาด 36 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ
ทธนบปผฝพฟภมยรลวศษสหฬอฮ
ะาิีึืุูเ่้๊๋็ั์
เโใไอำฤๅฦๆฯ

7. ขนาด 48 points

กขฃคฅฆงจฉชซฌญฎฏฐ

ฑฒฌณดตถทธนบปผฝพฟ

ภมยรลวศษสหฬอฮ

ะาอิีีึืุูเ่้๊๋็ั์

เโใไอำฤๅภๆฯ

ภาคผนวก ข

ตัวอักษรต้นแบบ Cordia

1. ขนาด 20 points

กขฃคฅฆงจฉชฌญฎฏฐฑฒณดตถ

ทธนบปผฝพฟภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็ั์

เโไำฤๅฦๆฯ

2. ขนาด 22 points

กขฃคฅฆงจฉชฌญฎฏฐฑฒณดตถ

ทธนบปผฝพฟภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็ั์

เโไำฤๅฦๆฯ

3. ขนาด 24 points

กขฃคฅฆงจฉชฌญฎฏฐฑฒณดตถ

ทธนบปผฝพฟภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็ั์

เโไำฤๅฦๆฯ

4. ขนาด 28 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ

ทธนบปผฝพฟภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็ั์

เโใไอำฤๅฦๆฯ

5. ขนาด 32 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ

ทธนบปผฝพฟภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็ั์

เโใไอำฤๅฦๆฯ

6. ขนาด 36 points

กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถ

ทธนบปผฝพฟภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็ั์

เโใไอำฤๅฦๆฯ

7 ขนาด 48 points

กขฃคฅฆงจฉชซฌญฎฏฐ

ฑฒฆณดตถทธนบปผฝพฟ

ภมยรลวศษสหฬอฮ

ะาิีึืุูเ่้๊๋็์ั

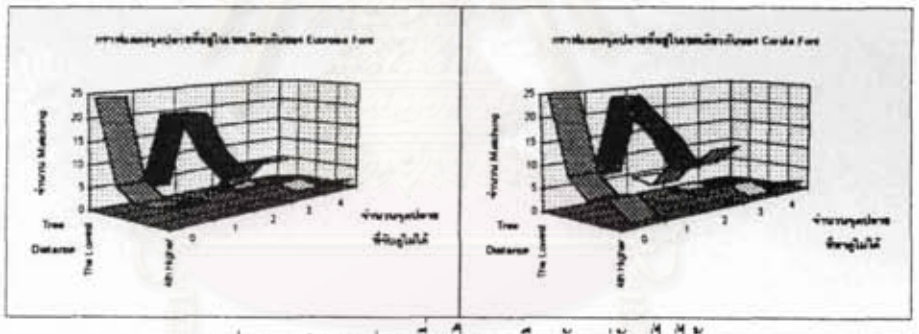
เโใไอำฤๅภๆฯ

ภาคผนวก ค

การวิเคราะห์จุดปลายของตัวอักษร

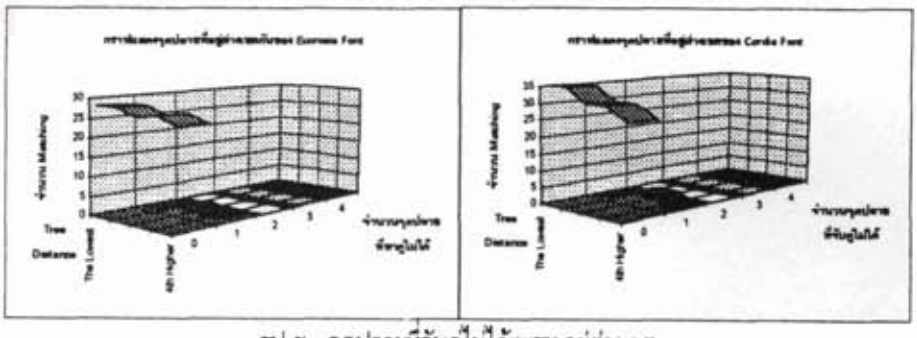
ในการวิเคราะห์ทาง feature นั้น มีการเปรียบเทียบเพื่อหาความแตกต่างระหว่างตัวอักษร เพื่อจับคู่หาตำแหน่งจุดปลายที่อยู่ใกล้กันที่สุดระหว่างตัวอักษรต้นแบบและตัวอักษรที่ต้องการรู้จำ แล้วหาจำนวนจุดปลายที่ไม่สามารถจับคู่ได้ ซึ่งแบ่งออกเป็น 2 ประเภทคือ

1. จุดปลายที่อยู่ในเขตเดียวกันแต่จับคู่ไม่ได้ กราฟรูป ก. ได้จากการนำผลการเปรียบเทียบค่าระยะของตัวอักษร แล้วนำจำนวนครั้งที่ตัวอักษร match, ค่า Tree Distance และจำนวนจุดปลายของตัวอักษรที่อยู่ในเขตเดียวกันแต่จับคู่ไม่ได้มาเขียนกราฟ พบว่าจำนวนตัวอักษรที่ match กันเป็นส่วนมากจะมีค่าของจำนวนจุดปลายที่จับคู่ไม่ได้ระหว่าง 0 ถึง 1 ดังนั้นจึงสรุปได้ว่าตัวอักษรที่ match กันได้ควรจะมีจำนวนจุดปลายที่อยู่ในเขตเดียวกันแต่จับคู่ไม่ได้ไม่เกิน 2 จุด



รูป ก. แสดงจุดปลายที่อยู่ในเขตเดียวกันแต่จับคู่ไม่ได้

2. จุดปลายที่จับคู่ไม่ได้เพราะอยู่ต่างเขต ลักษณะเหมือนในข้อที่ 1 แต่นำจำนวนจุดปลายที่จับคู่ไม่ได้เพราะอยู่ต่างเขตกันมาเขียนกราฟ ดังแสดงในรูป ข. พบว่าจำนวนตัวอักษรที่ match กันเป็นส่วนมากจะมีค่าของจุดปลายที่จับคู่ไม่ได้เพราะอยู่ต่างเขตเป็น 0 ดังนั้นจึงสรุปได้ว่าตัวอักษรที่ match กันได้ควรจะมีจำนวนจุดปลายที่จับคู่ไม่ได้เพราะอยู่ต่างเขตไม่เกิน 1 จุด



รูป ข. จุดปลายที่จับคู่ไม่ได้เพราะอยู่ต่างเขต

ประวัติผู้เขียน

นายสนธยา เมรินทร์ เกิดวันที่ 1 พฤษภาคม พ.ศ. 2510 ที่จังหวัดลำปาง สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาตรบัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมเทคโนโลยี สถาบันเทคโนโลยีราชมงคล เมื่อปี พ.ศ. 2533 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2535 เคยทำงานในตำแหน่งเจ้าหน้าที่ควบคุมช่ายสื่อสาร ที่ธนาคาร สหธนาคาร (มหาชน) จำกัด เมื่อปี พ.ศ. 2530 และทำงานในตำแหน่งเจ้าหน้าที่ควบคุมระบบคอมพิวเตอร์ ที่บริษัท เงินทุนหลักทรัพย์ธนชาติ (มหาชน) จำกัด ตั้งแต่ปี พ.ศ. 2533 จนถึงปัจจุบัน