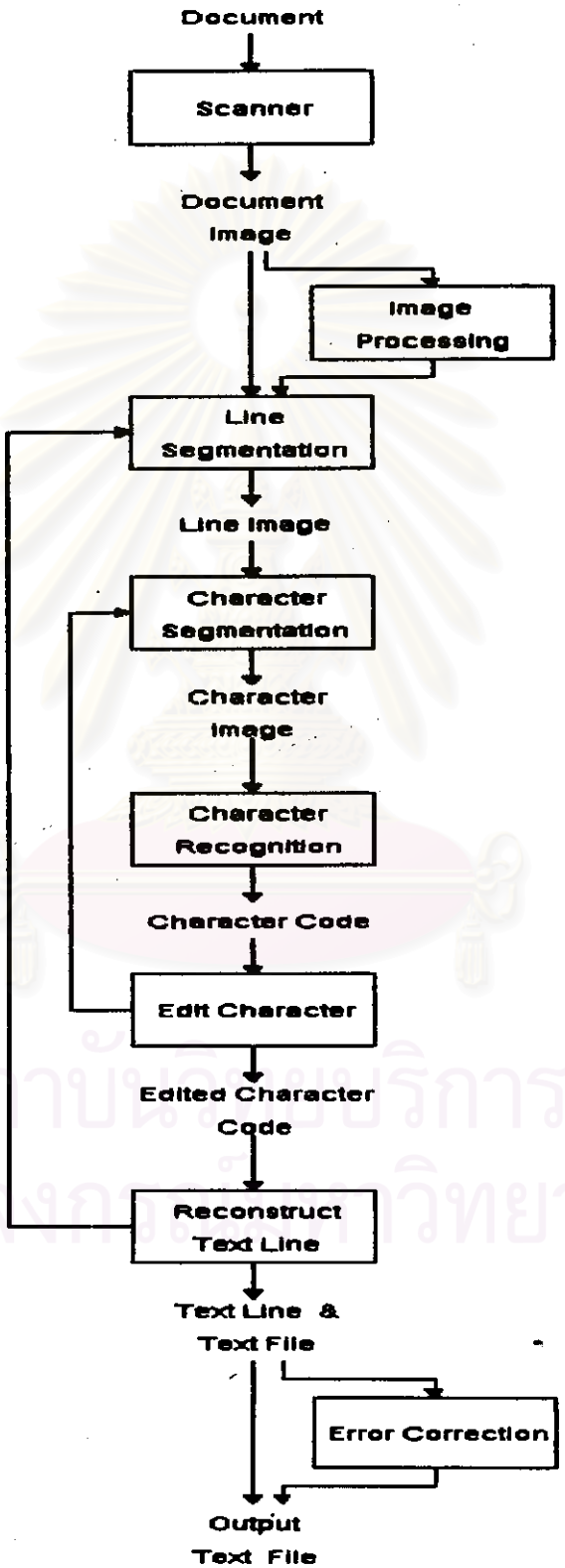
บทที่ 3
การออกแบบและพัฒนา

การทำงานของโปรแกรมไทยโอซีอาร์



รูปที่ 3.1 แผนภาพแสดงการทำงานของโปรแกรมไทยโอซีอาร์

จากแผนภาพในรูปที่ 3.1 ภาพของเอกสารจะได้จากการสแกนเอกสารต้นฉบับด้วยเครื่องสแกนเนอร์ชนิดที่รับข้อมูลเข้าทีละหน้ากระดาษ ซึ่งกำหนดความละเอียดไว้ที่ 300 จุดต่อนิ้ว (DPI) โดยจะอ่านภาพในลักษณะขาว-ดำ หรือ ภาพที่มีระดับความเข้มในโทนสีเทา 256 ระดับ (256 Gray level) หรือเปิดแฟ้มข้อมูลที่เป็นภาพเอกสารในรูปแบบบิตแมพไฟล์ของไมโครซอฟท์วินโดวส์ (Microsoft Windows BMP) ที่ได้ทำการสแกนและเก็บไว้ก่อนแล้ว หากภาพของเอกสารมีระดับความเข้มในโทนสีเทา 256 ระดับ โปรแกรมจะให้ผู้ใช้เป็นผู้กำหนดค่า Threshold เพื่อปรับให้เป็นภาพขาว-ดำ

ถ้ารูปของเอกสารนั้น มีจุดภาพรบกวน, เอียง หรือ ตัวอักษรเป็นสีขาวบนพื้นดำ (โดยปรกติตัวอักษรต้องเป็นสีดำบนพื้นสีขาว) ผู้ใช้สามารถกำหนดให้โปรแกรมโอซีอาร์ทำงานในขั้นตอนเตรียมประมวลผลด้วยวิธีการประมวลผลรูปภาพ (อธิบายไว้ในบทที่ 2) เพื่อให้ได้รูปเอกสารที่เหมาะสมกับขั้นตอนประมวลผลการรู้จำ

เมื่อได้รูปที่เหมาะสมกับการทำงานของโปรแกรมโอซีอาร์แล้ว ก็สามารถเริ่มทำงานในขั้นตอนประมวลผลการรู้จำ โดยโปรแกรมจะเริ่มทำงานที่กระบวนการตัดแยกบรรทัดโดยหาบริเวณที่น่าจะเป็นบรรทัดเดียวกันจนครบทั้งเอกสาร จากนั้นก็จะสำเนารูปของบรรทัดและส่งไปยังกระบวนการตัดแยกตัวอักษรทีละบรรทัด

กระบวนการตัดแยกตัวอักษรจะทำการหาบริเวณที่น่าจะเป็นตัวอักษร จากนั้นก็จะทำการตัดรูปของตัวอักษรออกจากรูปของบรรทัด และส่งไปยังกระบวนการรู้จำตัวอักษร (ที่ได้อธิบายไว้ในบทที่ 2) พร้อมกับตำแหน่งต่างๆ ของตัวอักษรเพื่อนำไปใช้ประโยชน์ในขั้นตอนอื่นต่อไป

กระบวนการรู้จำจะหาความสัมพันธ์ของรูปของตัวอักษรที่ได้รับ กับค่าสถิติต่างๆที่เก็บไว้เพื่อหาคำตอบว่ารูปที่ได้นั้นคือรูปของตัวอักษรอะไร และตอบผลลัพธ์กลับมาเป็นรหัสของตัวอักษร (Character Code)

รหัสของตัวอักษรที่ได้มานั้นอาจไม่ถูกต้องทั้งหมดจึงต้องทำการตรวจสอบและแก้ไขเสียก่อนในกระบวนแก้ไขผลลัพธ์ที่ได้จากขั้นตอนการรู้จำ โดยใช้เส้นแบ่งระดับตัวอักษรที่ได้จากกระบวนการตัดแยกบรรทัดและค่าตำแหน่งของรูปตัวอักษรที่ได้จากกระบวนการตัดแยกตัวอักษร เช่น หากพบว่ามี ' ั ' ต่อกับ ' ั ' และอักษร 2 ตัวนี้อยู่ซ้อนกัน แสดงว่าอักษรนี้เป็น ' ะ '

เมื่อได้รหัสของตัวอักษรแล้วจะเก็บคำตอบนั้นไว้และจะตรวจสอบว่ายังมีรูปตัวอักษรเหลืออยู่ในรูปของบรรทัดนั้นหรือไม่ ถ้ายังมีรูปตัวอักษรเหลืออยู่ก็จะย้อนกลับไปทำงานในกระบวนการตัดแยกตัวอักษร จนกระทั่งไม่มีรูปตัวอักษรเหลืออยู่ในรูปของบรรทัดนั้น แล้วจะไปทำงานต่อในกระบวนการสร้างบรรทัด

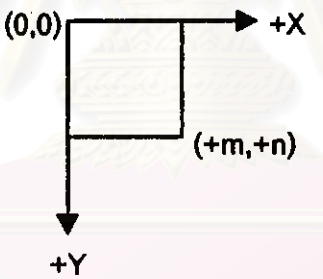
กระบวนการสร้างบรรทัดจะนำเอารหัสของตัวอักษรที่เก็บไว้มาเรียงต่อรวมกันเพื่อให้กลายเป็นประโยคที่ถูกต้องแล้วจัดเก็บต่อจากประโยคที่แล้ว จากนั้นก็จะทำการตรวจสอบว่ายังมี

รูปของบรรทัดที่เหลืออยู่และยังไม่ได้ทำการรู้จำตัวอักษรอีกหรือไม่ ถ้ายังมีก็จะย้อนกลับไปทำงานในกระบวนการตัดแยกบรรทัดต่อ จนกระทั่งครบทั้งเอกสารก็จะได้ผลลัพธ์ที่ของการรู้จำตัวอักษรทั้งเอกสาร ซึ่งสามารถบันทึกเก็บไว้เป็นแฟ้มข้อมูลแบบตัวอักษรเพื่อนำไปใช้งานต่อไปได้ หรือหากพบว่าผลลัพธ์ของการรู้จำตัวอักษรนั้นมีข้อผิดพลาดอยู่มากก็สามารถทำงานในกระบวนการแก้ไขคำผิดได้

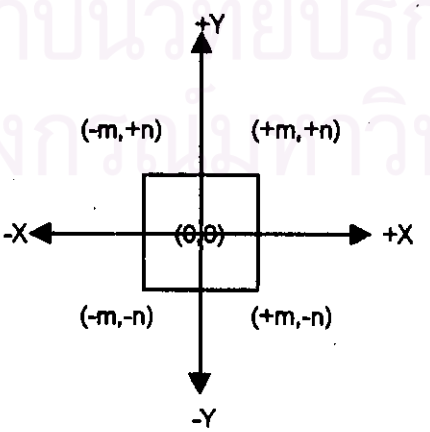
กระบวนการแก้ไขคำผิดจะนำข้อความแต่ละบรรทัดไปทำการแก้ไข (ซึ่งได้อธิบายวิธีการทำงานแล้วในบทที่ 2) จนครบทุกบรรทัด ซึ่งก็สามารถบันทึกผลลัพธ์ที่แก้ไขแล้วเป็นแฟ้มข้อมูลแบบตัวอักษรเพื่อนำไปใช้งานต่อไปได้เช่นกัน

โครงสร้างของข้อมูลในส่วนของการประมวลผลรูปภาพ

เนื่องจากรูปภาพของเอกสารเป็นรูปภาพแบบขาว-ดำ จึงกำหนดให้โครงสร้างของข้อมูลเป็นอาเรย์ของไบต์ (array of BYTE) ซึ่งเป็นอาเรย์ 2 มิติที่มีขนาดความกว้าง x ยาว เท่ากับจำนวนจุด (pixel) ตามความกว้าง x จำนวน จุด (pixel) ตามความยาวของรูปภาพ ค่าที่เก็บลงในอาเรย์คือ 0 แทนจุดสีดำ และ 1 แทนจุดสีขาว โดยที่จุดซ้ายบนสุดของรูปภาพจะเก็บที่ อาเรย์ช่อง [0][0] เรียกว่าระนาบของการแสดงผล ตามรูปที่ 3.2 ซึ่งแตกต่างจากระนาบของกราฟ ตามรูปที่ 3.3



รูปที่ 3.2 แสดงระนาบของการแสดงผล



รูปที่ 3.3 แสดงระนาบของกราฟ

การลดจุดภาพรบกวน

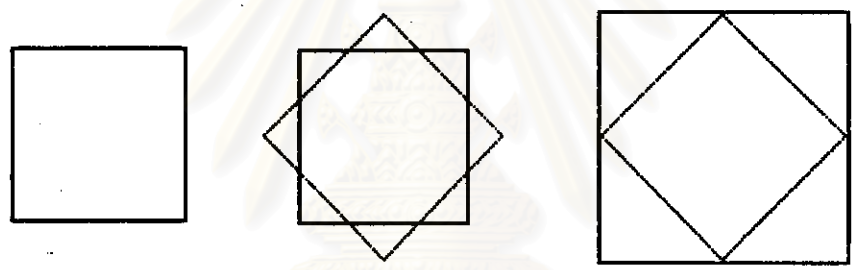
จากสมการ (1) ในบทที่ 2 เราพบว่ามีปัญหาในการหาผลรวมที่จุดภาพที่อยู่บริเวณมุมและขอบเนื่องจากว่ามีจุดใกล้เคียงไม่ครบ 8 จุด ดังนั้นจึงแก้ปัญหาโดยให้ค่าแก่จุดใกล้เคียงที่ไม่มีอยู่นั้น ด้วยค่า 1 (ค่าของสีขาว) แล้วคำนวณหาผลรวมตามสมการเดิม

การหมุนภาพ

เนื่องจาก สมการ (2) และ (3) ในบทที่ 2 นั้น เป็นสมการการหมุนภาพ 2 มิติรอบจุดศูนย์กลางที่จุด X = 0, Y = 0 ในระนาบของกราฟ (รูปที่ 3.3) ดังนั้นในการหมุนภาพในโปรแกรมโอซีอาร์จะต้องทำการหาตำแหน่งของจุดศูนย์กลางการหมุนซึ่งจะใช้จุดกึ่งกลางของภาพเสียก่อน คือ

X = (ความกว้างของภาพ – 1) / 2 และ Y = (ความสูง – 1) / 2

แล้วปรับตำแหน่งของจุดต่างๆ บนภาพจากระนาบของการแสดงผล เป็นระนาบของกราฟ โดยถือว่าจุดกึ่งกลางที่คำนวณได้นั้นคือจุด (0,0) จากนั้นจะต้องคำนวณหาขนาดของภาพใหม่เสียก่อน เนื่องจากการหมุนภาพอาจทำให้ภาพโดนตัดได้ถ้าไม่ขยายขนาดภาพ ตามรูปที่ 3.4 ก, ข, ค



รูปที่ 3.4 ก รูปที่ 3.4 ข รูปที่ 3.4 ค

รูปที่ 3.4 ก แสดงภาพและขนาดของภาพก่อนการหมุนภาพ

รูปที่ 3.4 ข แสดงภาพหลังจากหมุนแล้ว แต่ยังใช้ขนาดภาพเท่าเดิม

รูปที่ 3.4 ค แสดงภาพหลังจากหมุนแล้ว และเปลี่ยนขนาดภาพใหม่

เมื่อได้ขนาดภาพใหม่แล้วก็จะทำการหมุนภาพในระนาบของกราฟ โดยจะคำนวณหาตำแหน่งใหม่ทีละจุดจนครบทั้งภาพ จากนั้นก็จะต้องคำนวณย้อนกลับเพื่อเปลี่ยนตำแหน่งของจุดบนระนาบของกราฟให้กลับมาอยู่ในระนาบของการแสดงผล

การกลับภาพ

การกลับภาพของเอกสาร เช่นเดียวกับการหมุนภาพคือจะต้องคำนึงถึงความแตกต่างของระนาบการแสดงผลและระนาบของกราฟ เพราะสมการที่ (4) และ (5) เป็นสมการการกลับภาพ

โดยอ้างอิงกับแกน Y และแกน X ตามลำดับ ดังนั้นจึงลดความยุ่งยากลงโดยใช้แนวกึ่งกลางของภาพทางแนวตั้งแทนแกน Y และแนวกึ่งกลางของภาพทางแนวนอนแทนแกน X

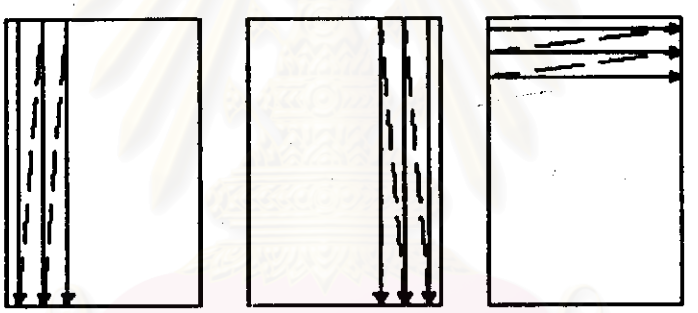
อัลกอริธึมในกระบวนการตัดแยกบรรทัด

เนื่องจากภาษาไทยมีตัวอักษรหลายระดับ ซึ่งทำให้เกิดความยุ่งยากในการระบุว่าบริเวณใดคือบรรทัดเดียวกัน เนื่องจากไม่สามารถใช้ช่องว่างระหว่างบรรทัดเป็นตัวระบุได้ทั้งหมด จึงต้องใช้อัลกอริธึมที่มีความซับซ้อน ดังนี้

1. <u>ตรวจสอบหาตำแหน่งกั้นซ้ายของเอกสาร</u> นำจุดภาพตามแนวตั้งมาทีละหลักจากซ้ายไปขวา (รูปที่ 3.5 ก) เพื่อตรวจสอบหาจุดว่ามีจุดดำอยู่ในหลักนี้หรือไม่ หากไม่พบจุดดำอยู่เลย ก็จะนำหลักถัดไปมาตรวจสอบต่อไป แต่ถ้าหากตรวจสอบพบจุดดำที่หลักใดก็จะทำการบันทึกไว้เป็นค่ากั้นซ้ายของเอกสาร แล้วจะไปทำงานในข้อ 2
2. <u>ตรวจสอบหาตำแหน่งกั้นขวาของเอกสาร</u> นำจุดภาพตามแนวตั้งมาทีละหลักจากขวาไปซ้าย (รูปที่ 3.5 ข) เพื่อตรวจสอบหาจุดว่ามีจุดดำอยู่ในหลักนี้หรือไม่ หากไม่พบจุดดำอยู่เลย ก็จะนำหลักถัดไปมาตรวจสอบต่อไป แต่ถ้าหากตรวจสอบพบจุดดำที่หลักใดก็จะทำการบันทึกไว้เป็นค่ากั้นขวาของเอกสาร แล้วจะไปทำงานในข้อ 3
3. <u>ตรวจสอบหาแถวเริ่มต้นของบรรทัด</u> นำจุดภาพตามแนวนอนมาทีละแถวจากบนลงล่าง (รูปที่ 3.5 ค) แล้วตรวจสอบว่ามีจุดดำอยู่ในแถวนี้หรือไม่ หากไม่พบจุดดำอยู่เลย ก็จะนำแถวถัดไปมาตรวจสอบต่อไป แต่ถ้าหากตรวจสอบพบจุดดำ แสดงว่าแถวที่กำลังตรวจสอบอยู่นี้เป็นแถวเริ่มต้นของบรรทัด จะไปทำงานในข้อ 4
4. <u>ตรวจสอบหาแถวสิ้นสุดของบรรทัด</u> นำจุดภาพตามแนวนอนมาทีละแถวจากบนลงล่าง แล้วตรวจสอบว่ายังมีจุดดำอยู่ในแถวนี้หรือไม่ หากยังพบจุดดำแสดงว่าแถวที่กำลังตรวจสอบอยู่นี้เป็นบรรทัดเดียวกับแถวก่อน ก็จะนำแถวถัดไปมาตรวจสอบต่อไป หากไม่พบจุดดำอยู่เลย ก็จะไปทำงานต่อในข้อ 5
5. <u>ตรวจสอบหาตัวอักษรในระดับอื่นของบรรทัดนี้</u> จากการที่ไม่พบจุดดำในแถวจากข้อ 4 นั้น ยังไม่สามารถระบุได้ว่าหมดบริเวณที่เป็นบรรทัดเดียวกันแล้ว เพราะอาจมีตัวอักษรในระดับอื่นที่ไม่ได้เชื่อมกับตัวอักษรในระดับที่ตรวจสอบในข้อ 4 จึงต้องตรวจสอบต่อไปอีกห้าแถว หากภายในห้าแถวนั้นไม่พบจุดดำอยู่เลยแสดงว่าแถวที่ได้ในข้อ 4 เป็นแถวของการสิ้นสุดบริเวณของบรรทัดนั้น แล้วจะข้ามไปทำงานต่อในข้อ 6 แต่หากพบจุดดำภายในห้าแถวนั้นจะถือว่าบริเวณของบรรทัดที่ได้ในข้อ 4 นั้นยังไม่หมด และจะย้อนกลับไปทำต่อในข้อ 4

6. <u>เก็บข้อมูลบริเวณที่เป็นบรรทัด</u> เมื่อทราบแถวเริ่มต้นและแถวสิ้นสุดของบริเวณที่เป็นบรรทัดแล้ว จะนำมาคำนวณหาเส้นแบ่งระดับตัวอักษรในบรรทัดนั้น (อธิบายไว้ในบทที่ 2) แล้วทำการบันทึกเก็บไว้ จากนั้นก็จะวนกลับไปทำงานในข้อ 3 จนกว่าจะครบทุกแถว (สุดเอกสารทางด้านล่าง) ก็จะจบการทำงาน

สิ่งสำคัญในการเชื่อมระดับของตัวอักษรให้มารวมกันเป็นบรรทัดเดียวกันนั้น ก็เพื่อความถูกต้องในการรู้จำตัวอักษร แต่จากการทดลองกับเอกสารที่สแกนด้วยความละเอียด 300 จุดต่อนิ้ว พบว่า ถ้าจำนวนแถวที่จะตรวจสอบในข้อ 5 หากทำการตรวจสอบมากกว่าห้าแถวแล้ว จะทำให้โปรแกรมทำการรวมเอาบรรทัดบนและล่างเข้าด้วยกัน ซึ่งจะทำให้ผลลัพธ์ผิด แต่การตรวจสอบด้วยจำนวนห้าแถวหรือน้อยกว่าห้าแถวนั้น อาจไม่สามารถรวมเอาระดับของตัวอักษรที่ควรจะเป็นบรรทัดเดียวกันเข้าด้วยกันได้ทั้งหมด จึงต้องมีอัลกอริธึมที่จะกล่าวถึงต่อไปในกระบวนการสร้างบรรทัด มาทำการรวมบรรทัดที่แยกผิดไว้นี้เข้าด้วยกัน



รูปที่ 3.5 ก รูปที่ 3.5 ข รูปที่ 3.5 ค

รูปที่ 3.5 ก แสดงวิธีหากั้นซ้ายของเอกสาร

รูปที่ 3.5 ข แสดงวิธีหากั้นขวาของเอกสาร

รูปที่ 3.5 ค แสดงวิธีหาบรรทัดในเอกสาร

อัลกอริธึมในกระบวนการตัดแยกตัวอักษร

หลังจากที่ทำการตัดแยกบรรทัดแล้ว จะทำการสำเนารูปของบรรทัดเพื่อนำมาหาตัวอักษรที่อยู่ในบรรทัด โดยจะทำการตรวจสอบหาจุดดำในบรรทัดจากบนลงล่างและจากซ้ายไปขวาตามรูปที่ 3.6

รูปที่ 3.6 แสดงวิธีหาตัวอักษรในบรรทัด

เมื่อพบจุดดำที่ตำแหน่งใดก็ตาม โปรแกรมโอซีอาร์จะทำการค้นหาจุดดำที่อยู่ติดกันไปเรื่อยๆ (Connected Component) จนไม่สามารถหาจุดดำที่ติดกันได้อีกตามอัลกอริธึมข้างล่าง ก็จะได้รูปของตัวอักษร จากนั้นก็จะตัดรูปตัวอักษรนั้นออกจากรูปของบรรทัด และส่งรูปของตัวอักษรให้กับขั้นตอนการรู้จำต่อไป

อัลกอริธึมในการค้นหาจุดดำที่อยู่ติดต่อกันสามารถเขียนได้โดยใช้ Recursive Prgramming ได้ดังนี้

```
void Segment_Char_8Conn(int x,int y,BYTE **src_pic,BYTE **des_pic )
{ // look for first image in src_pic array and return in des_pic array
 if (src_pic[x][y] == BLACK_COLOR) // ตรวจสอบว่าเป็นจุดดำหรือไม่
 {  // พบว่าเป็นจุดดำ
  src_pic[x][y] = WHITE_COLOR; // ลบจุดดำออกจากรูปบรรทัด
  des_pic[x][y] = BLACK_COLOR; // ใส่จุดดำลงในรูปของตัวอักษร
  Segment_Char_8Conn(x+1,y,src_pic,des_pic );  // recursive ไปยังจุดถัดไป
  Segment_Char_8Conn(x-1,y,src_pic,des_pic );
  Segment_Char_8Conn(x,y+1,src_pic,des_pic );
  Segment_Char_8Conn(x,y-1,src_pic,des_pic );
  Segment_Char_8Conn(x+1,y+1,src_pic,des_pic );
  Segment_Char_8Conn(x-1,y-1,src_pic,des_pic );
  Segment_Char_8Conn(x-1,y+1,src_pic,des_pic );
  Segment_Char_8Conn(x+1,y-1,src_pic,des_pic );
 }
}
```

การรู้จำตัวอักษร

จากงานวิจัยของชเนศ [14] นั้นใช้จำนวนไอเกนเวกเตอร์ในเมตริกซ์ของไอเกนเวกเตอร์เท่ากับ 128 เวกเตอร์ (A_{128}) และเมตริกซ์ค่าเฉลี่ย (m_x) เพื่อคำนวณหาเวกเตอร์รูปแบบ ซึ่งเป็นคุณลักษณะ (Feature) ของตัวอักษรในทางสถิติ ซึ่งเวกเตอร์รูปแบบที่ได้นั้นจะมีสมาชิกเป็นจำนวนจริง และมีจำนวนสมาชิกเท่ากับ 128 ตัว

ส่วนจำนวนของ hidden node ใน hidden layer ที่ใช้ในงานวิจัยของธเนศ [14] นั้นคือ 128 node และเนื่องจากใช้เวกเตอร์รูปแบบที่มีสมาชิก 128 ตัว จึงต้องกำหนดจำนวนของ input node ใน input layer เป็น 128 node และมีจำนวนของ output node ใน output layer เป็น 68 node (เพราะทำการรู้จำเฉพาะตัวอักษรภาษาไทย 68 ตัว)

แต่ในวิทยานิพนธ์ฉบับนี้จะต้องทำการรู้จำตัวอักษรภาษาไทยและภาษาอังกฤษและยังเพิ่มตัวอักษรที่เป็นสัญลักษณ์อื่นๆเข้าไปอีก จึงต้องกำหนดจำนวนของ output node ใน output layer เป็น 178 node ตามจำนวนตัวอักษรทั้งหมดที่ต้องการรู้จำ และได้ทำการทดลองเพื่อหาจำนวนของ input node และ hidden node เพื่อให้ได้ network มีประสิทธิภาพที่สุด แต่จากการทดลองไม่สามารถหา network ที่ convergence ได้ จึงนำค่าน้ำหนักและค่าไบแอสจากการเรียนรู้ทุกๆ 1 ล้าน iteration ของทุก network มาทำการหาจำนวนความผิดพลาดเพื่อเปรียบเทียบกัน เป็นจำนวน 20 ล้าน iteration ซึ่งได้แสดงผลการทดลองไว้ในตารางที่ 3.1 ดังนี้

Network ที่	จำนวน Node			จำนวนตัวอักษรที่รู้จำผิดของ network	
	Input	Hidden	Output	ข้อมูลชุดทดสอบ (1024 ตัว)	ข้อมูลชุดเรียนรู้ (8544 ตัว)
1	128	128	178	33	74
2	128	150	178	36	51
3	128	250	178	34	39
4	150	128	178	28	84
5	150	150	178	31	77
6	150	250	178	32	65
7	250	128	178	35	82
8	250	150	178	28	50
9	250	250	178	28	76

ตารางที่ 3.1 แสดงผลการทดลองของ network ขนาดต่างๆ และจำนวนความผิดพลาดที่เกิดขึ้น

การทดลองได้ใช้เงื่อนไขในการเรียนรู้เหมือนกันทุก network ดังนี้

1. ค่าอัตราการเรียนรู้ (Learning Rate) เท่ากับ 0.08
2. ค่าอินเนอร์เชียหรือโมเมนตัม (Inertia or Momentum) เท่ากับ 0.95
3. ใช้ Activation Function แบบ Sigmoid Function ซึ่งให้ค่าของฟังก์ชันอยู่ในช่วง (0,1)

4. เงื่อนไขในการ Convergence คือ ต้องเรียนรู้ตัวอย่างให้ถูกต้องเป็นจำนวน 1424 ตัวอย่าง โดยมีค่าผิดพลาดไม่เกิน 0.5
5. ค่าเริ่มต้นของค่าน้ำหนักและไบแอสได้จากการสุ่มค่าระหว่าง [-0.1,0.1]

การพิจารณาผลลัพธ์ของ network นั้น จะดูอันดับของ node ที่มีค่าสูงที่สุดเพียง node เดียว เป็นคำตอบ เช่น กำหนดว่าคำตอบของการรู้จำ รูปตัวอักษร ก นั้น node ที่ 1 จะต้องมีค่าสูงที่สุด ซึ่งถ้า network ใดมี node อื่นที่มีค่าสูงกว่าค่าของ node ที่ 1 แล้วจะถือว่า network นั้นตอบผิด

การพิจารณาเลือกว่าจะนำ network ใดมาใช้งานนั้น จะพิจารณาจากความผิดพลาดของข้อมูลในชุดทดสอบจะต้องผิดพลาดน้อยที่สุด ซึ่งจะมีอยู่ 3 network ที่ได้ผลลัพธ์เท่ากันคือ 28 ดังนั้นจึงต้องพิจารณาความผิดพลาดของข้อมูลในชุดเรียนรู้ของทั้ง 3 network นั้น ซึ่งจะได้ว่าความผิดพลาดของ network ที่ 8 นั้นน้อยที่สุด

ดังนั้นในวิทยานิพนธ์ฉบับนี้ จึงใช้จำนวนไอเกนเวกเตอร์ในเมตริกซ์ของไอเกนเวกเตอร์เท่ากับ 250 เวกเตอร์ (A_{250}) และใช้ network ที่มี input node เท่ากับ 250 node และ hidden node เท่ากับ 150 node และ output node เท่ากับ 178 node

การแก้ไขผลลัพธ์ที่ได้จากขั้นตอนการรู้จำตัวอักษร

เนื่องจากในภาษาไทยมีการใช้ตัวอักษรที่มีลักษณะเดียวกัน แต่วางไว้คนละระดับกัน และถือว่าเป็นตัวอักษรคนละตัวกัน เช่น ' ๋ ' กับ '๘' หรือ ' ๊ ' กับ '๙' หรือการใช้ตัวอักษร 2 ตัวมารวมกันกลายเป็น ตัวอักษรตัวใหม่เช่น 'เ' กับ 'เ' กลายเป็น 'แ' หรือ ' ํ ' รวมกับ 'า' กลายเป็น ' ำ' และยังมีตัวอักษรภาษาอังกฤษและสัญลักษณ์ต่างๆ ที่คล้ายกับภาษาไทยอีก ซึ่งสิ่งเหล่านี้ทำให้ผลลัพธ์ของการรู้จำตัวอักษรนั้นไม่ถูกต้อง ดังนั้นจึงต้องทำการแก้ไขผลลัพธ์ที่ได้จากขั้นตอนการรู้จำตัวอักษร โดยใช้เส้นแบ่งระดับที่ได้เก็บค่าไว้แล้วจากการตัดแยกบรรทัด และตำแหน่งของรูปตัวอักษรนั้น มาใช้พิจารณาร่วมกัน จากนั้นก็จะทำการจัดเก็บตัวอักษรเหล่านี้ไว้ในตาราง ตามระดับของตัวอักษรนั้น (ตามตารางที่ 2.2)

การสร้างบรรทัด

การสร้างบรรทัดเป็นการนำเอาตัวอักษรที่อยู่ในระดับต่างๆ ที่ได้เก็บไว้แล้วมารวมกันเป็นประโยค โดยจะพิจารณาตำแหน่งของตัวอักษรที่อยู่ในระดับกลางก่อนโดยนำเอาตัวอักษรที่มีตำแหน่งของรูปอยู่ทางซ้ายสุดมาใช้ก่อน แล้วจะหาตัวอักษรในระดับล่าง, ระดับบน และ ระดับบนสุด ที่อยู่ตรงกับตัวอักษรในระดับกลางนั้น ตามลำดับแล้วจึงหาตัวอักษรในระดับกลางตัวถัดไปมา

ใช้งาน ซึ่งจะมีการใส่เว้นวรรคลงในประโยคด้วยเมื่อพบว่าตัวอักษรในระดับกลาง 2 ตัวอยู่ห่างกันเกินค่าที่กำหนดค่าหนึ่ง เมื่อนำเอาตัวอักษรในระดับกลางมาใช้หมดแล้วก็จะได้ประโยคที่ครบถ้วนและจัดเก็บประโยคนั้นไว้ใช้งานต่อไป

แต่จากปัญหาในการตัดแยกบรรทัด ที่อาจเกิดการตัดแยกบรรทัดที่ควรจะเป็นบรรทัดเดียวกันให้กลายเป็นหลายบรรทัด (เนื่องจากตัวอักษรในแต่ละระดับนั้นห่างเกินระยะที่กำหนด) ดังนั้นกระบวนการสร้างประโยคจะทำการตรวจสอบก่อนว่า ถ้าตัวอักษรที่จะนำมาสร้างประโยคนี้ มีตัวอักษรในระดับกลางอยู่ ก็จะทำการสร้างประโยคโดยใช้ตัวอักษรจากทุกระดับตัวอักษรที่มีอยู่ ตามปกติ แต่ถ้าไม่มีตัวอักษรในระดับกลางอยู่เลยก็จะตรวจสอบว่าตัวอักษรที่ได้มานั้นอยู่ในระดับใด

1. หากเป็นตัวอักษรในระดับบนหรือตัวอักษรในระดับบนสุด แสดงว่าการตัดแยกประโยคนั้นได้ตัดตัวอักษรในระดับบนหรือบนสุดออกจากระดับกลาง และจะยังไม่สร้างประโยคในบรรทัดนี้ ซึ่งจะทำการเก็บตัวอักษรเหล่านี้ไปรวมกับบรรทัดต่อไป
2. หากเป็นตัวอักษรในระดับล่าง แสดงว่าการตัดแยกประโยคนั้นได้ตัดตัวอักษรในระดับล่างออกจากระดับกลาง ซึ่งจะนำเอาประโยคที่ได้จากการสร้างในบรรทัดที่แล้วมาทำการสร้างใหม่ โดยนำเอาตัวอักษรในระดับล่างนี้ไปร่วมสร้างด้วย ซึ่งจะได้ประโยคใหม่ที่มีตัวอักษรในระดับล่างแล้ว ไปแทนประโยคที่แล้ว

การแก้ไขคำผิดที่เกิดจากโปรแกรมโอซีอาร์ภาษาไทย

จากสมการ (17.2) ในบทที่ 2 นั้นจะต้องใช้ค่าความน่าจะเป็น 2 ค่า คือ $P(W)$ และ $P(S|W)$ ซึ่งค่าความน่าจะเป็น $P(W)$ นั้นจะนำมาจากโครงงานของอนันต์ลดาและชลวิช ซึ่งได้ใช้แบบจำลองประเภทของคำแบบไตรแกรม (อธิบายไว้ในบทที่ 2) แต่ค่าความน่าจะเป็น $P(S|W)$ นั้นจะต้องทำการรวบรวมขึ้นใหม่เนื่องจากเป็นค่าความน่าจะเป็นที่ขึ้นอยู่กับลักษณะของโปรแกรมโอซีอาร์แต่ละตัว ซึ่งในวิทยานิพนธ์ฉบับนี้ได้ทำการหาค่าความน่าจะเป็น $P(S|W)$ ตามวิธีการใหม่ที่ใช้ในงานของ Meknavin et al. [19] ที่พิจารณาถึงความผิดพลาดที่เกิดกับตัวอักษรในแบบต่างๆ โดยใช้ข้อมูลตัวอักษรเพื่อเปรียบเทียบกันระหว่างตัวอักษรในเอกสารต้นฉบับกับตัวอักษรในเอกสารที่ได้จากโปรแกรมโอซีอาร์ เป็นจำนวน 74180 ตัวอักษร (การแก้ไขคำผิดนี้สามารถแก้ไขได้เฉพาะคำในภาษาไทยเท่านั้น)