บทที่ 2
แนวคิดและทฤษฎี

ค่าของจุดภาพในการประมวลผลรูปภาพ

ในวิทยานิพนธ์ฉบับนี้จะใช้รูปภาพที่มีจำนวนระดับความเข้มของภาพ 2 ระดับ (Binary Image) ซึ่งการประมวลผลรูปภาพในวิทยานิพนธ์ฉบับนี้จะแทนค่าของจุดสีดำด้วย 0 และแทนค่าจุดสีขาวด้วย 1

การพิจารณาจุดภาพที่อยู่ติดกันในการประมวลผลรูปภาพ

การพิจารณาว่าจุดภาพที่กำลังสนใจนั้น อยู่ติดกับจุดภาพอื่นที่มีสีเดียวกันหรือไม่ จะพิจารณาจากจุดภาพใกล้เคียง 8 จุด (8 – Connected Component ของ Gonzalez and Woods [16]) ว่ามีจุดใดที่มีสีเดียวกับจุดที่สนใจอยู่หรือไม่ ซึ่งจุดภาพใกล้เคียง 8 จุด ก็คือจุดภาพ 8 จุดที่ล้อมรอบจุดที่สนใจ ตามรูปที่ 2.1

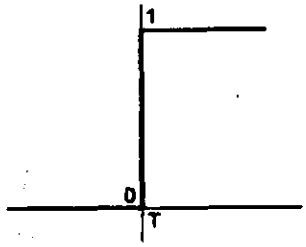
P1	P2	P3
P4	P5	P6
P7	P8	P9

รูปที่ 2.1 แสดงจุดภาพ 9 จุด โดยจุดที่สนใจคือจุด P5
และ จุด P1, P2, P3, P4, P6, P7 P8 และ P9 คือจุดภาพใกล้เคียง 8 จุด

เช่น จุดที่สนใจคือจุด P5 ซึ่งมีสีดำ การพิจารณาว่าจุด P5 นั้น อยู่ติดกับจุดสีดำอื่นหรือไม่ ก็จะพิจารณาที่ จุด P1, P2, P3, P4, P6, P7 P8 และ P9 ว่ามีสีดำหรือไม่ หากพบว่าจุดใดจุดหนึ่งมีสีดำก็แสดงว่า จุด P5 นั้นอยู่ติดกับจุดดำอื่น แต่หากทั้ง 8 จุดนั้นไม่มีสีดำเลยก็แสดงว่าจุด P5 นั้นไม่ติดกับจุดดำอื่น

การแปลงระดับความเข้มสีของรูปภาพจากหลายระดับเป็นสองระดับ (Binarization)

การแปลงระดับความเข้มสีของรูปภาพจากหลายระดับเป็นสองระดับ ใช้วิธีการกำหนดค่า Threshold (Gonzalez and Woods [16]) เป็นตัวแบ่งระดับ ดังรูปที่ 2.2



รูปที่ 2.2 แสดงการกำหนดค่าระดับความเข้มของภาพ
เมื่อต้องการรูปที่มีจำนวนระดับความเข้มเพียง 2 ระดับ

จากรูปที่ 2.2 แกน X คือค่าระดับความเข้มของรูปต้นแบบ และ แกน Y คือค่าระดับความเข้มของรูปใหม่ และ T คือค่า Threshold จะพบว่าจุดที่มีระดับความเข้มมากกว่าหรือเท่ากับค่า T จะถูกปรับให้เป็น 1 และจุดที่มีระดับความเข้มน้อยกว่าค่า T จะถูกปรับให้เป็น 0

การลดจุดภาพรบกวน (Noise Reduction)

วิธีการที่ใช้ในการลดจุดภาพรบกวน คือ การหาผลรวมของค่าในจุดภาพใกล้เคียง 8 จุด จากรูปที่ 2.1 เมื่อพบว่าจุด P5 เป็นจุดภาพสีดำ (P5 มีค่าเท่ากับ 0) จะใช้สมการ

$$sum = \sum_{n=1}^{9} val(P_n) \qquad (1)$$

ถ้าหากค่าของ sum มีค่าเท่ากับ 8 แสดงว่าจุดใกล้เคียงจุด P5 ทั้ง 8 จุดนั้นมีสีขาว (จุดขาวมีค่าเป็น 1) จะถือว่า จุด P5 ไม่อยู่ติดกับจุดดำอื่น ซึ่งถือว่าเป็นจุดภาพรบกวน และจะทำการเปลี่ยนเป็นจุดสีขาวแทน

การหมุนภาพ (Rotation)

การหมุนภาพจะให้ผู้ใช้เป็นผู้กำหนดองศาความเอียงของเอกสาร แล้วโปรแกรมจะทำการหมุนภาพตามสมการการหมุนภาพ 2 มิติ (Heam and Baker [17])

- เมื่อหมุนภาพในทิศทางทวนเข็มนาฬิกา

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (2)$$

และจะได้ค่า x' และ y' ตามสมการ

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta \qquad (2.1), (2.2)$$

- เมื่อหมุนภาพในทิศทางตามเข็มนาฬิกา

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3)$$

และจะได้ค่า x' และ y' ตามสมการ

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\sin\theta + y\cos\theta \quad (3.1), (3.2)$$

การกลับภาพ (Flipping ,Reflection) ใช้สมการการกลับภาพ 2 มิติ (Hearn and Baker [17])

- เมื่อกลับภาพในแนวนอน

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (4)$$

และจะได้ค่า x' และ y' ตามสมการ

$$x' = -x$$
$$y' = y \quad (4.1), (4.2)$$

- เมื่อกลับภาพในแนวตั้ง

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (5)$$

และจะได้ค่า x' และ y' ตามสมการ

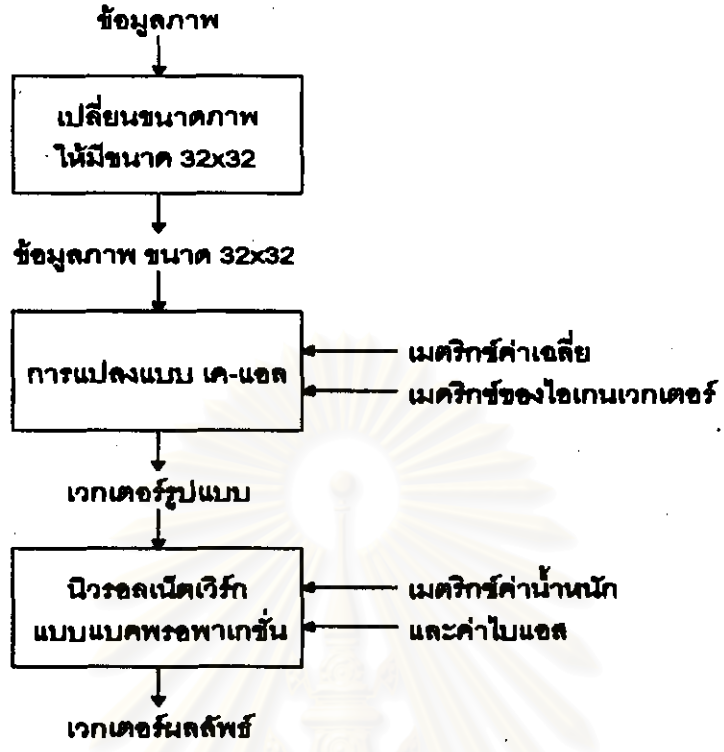
$$x' = x$$
$$y' = -y \quad (5.1), (5.2)$$

การกลับสีของจุดภาพ (Inversion)

การกลับสีของจุดภาพ เป็นการเปลี่ยนสีของจุดที่ต้องการ โดยจุดที่มีสีดำจะถูกเปลี่ยนเป็นสีขาว (เปลี่ยนค่าของจุดจาก 0 เป็น 1) และ จุดที่มีสีขาวจะถูกเปลี่ยนเป็นสีดำ (เปลี่ยนค่าของจุดจาก 1 เป็น 0)

แผนภาพการทำงานของการรู้จำตัวอักษร

การรู้จำตัวอักษรในวิทยานิพนธ์ฉบับนี้นำแนวคิดและทฤษฎีจาก การรู้จำตัวอักษรพิมพ์ภาษาไทยโดยใช้เทคนิคด้านการวิเคราะห์ตัวประกอบสำคัญและนิวรอลเน็ตเวิร์กของธเนศ[14] โดยมีแผนภาพการทำงานตามรูปที่ 2.3



รูปที่ 2.3 แสดงแผนภาพการทำงานของการรู้จำตัวอักษรพิมพ์ภาษาไทย
โดยใช้เทคนิคด้านการวิเคราะห์ตัวประกอบสำคัญและนิวรอลเน็ตเวิร์ก

ข้อมูลภาพ

จากรูป 2.3 ข้อมูลภาพคือเวกเตอร์ของแต่ละจุดภาพ โดยให้แทนค่าจุดขาวด้วย 1 (ใช้ค่า 1 เหมือนค่าที่ใช้ในขั้นตอนการประมวลผลรูปภาพ) และแทนค่าจุดดำด้วย -1 (เปลี่ยนจากค่า 0 จากค่าที่ใช้ในขั้นตอนการประมวลผลรูปภาพ เป็นค่า -1) โดยเริ่มจากจุดมุมซ้ายล่างของภาพไปยังจุดถัดมาในแถวเดียวกันจนครบทุกจุดบนแถวเดียวกัน และเริ่มจุดซ้ายสุดของภาพของแถวถัดขึ้นไปด้านบนดังรูปที่ 2.4

รูปที่ 2.4 ลักษณะการแทนจุดภาพด้วยเวกเตอร์

การเปลี่ยนขนาดภาพ

เนื่องจากภาพตัวอักษรที่ปรากฏในเอกสารมีอยู่หลายขนาด ซึ่งจะได้ข้อมูลภาพตัวอักษร (เวกเตอร์ของภาพตัวอักษร) มีขนาดไม่เท่ากัน ทำให้การคำนวณในขั้นตอนต่อไปนั้นยุ่งยาก จึงต้องทำการแปลงขนาดภาพตัวอักษรให้อยู่ในขนาดเดียวกันเสียก่อน ซึ่งขนาดที่ใช้ในงานวิจัยของธเนศ

[14] คือ 32 x 32 (หรือเมตริกซ์ขนาด 1024 x 1) โดยรักษาอัตราส่วนจำนวนแถวที่มากที่สุดของภาพต่อจำนวนหลักที่มากที่สุดของจุดภาพเดิม

การหาเมตริกซ์ค่าเฉลี่ยและเมตริกซ์ของไอเกนเวกเตอร์

เมตริกซ์ค่าเฉลี่ยเป็นเมตริกซ์ที่ได้จากการนำเวกเตอร์ของจุดภาพขนาด 32 x 32 ที่ได้จากทุกภาพของตัวอักษรมาทำการเฉลี่ย ตามสมการ

$$m_x = \frac{1}{M}\sum_{i=1}^{M} x_i \qquad (6)$$

เมื่อ m_x คือเมตริกซ์ค่าเฉลี่ย (ขนาด 1024 x 1)

M คือจำนวนเวกเตอร์ของจุดภาพทั้งหมดที่เป็นตัวอย่างที่ใช้ขณะเรียนรู้

x_i คือเวกเตอร์ภาพขนาด 32 x 32 (เมตริกซ์ขนาด 1024 x 1)

เมื่อหาเมตริกซ์ค่าเฉลี่ยได้แล้ว จะสามารถหาค่าเมตริกซ์โคแวเรียน (Covariance Matrix) ซึ่งแสดงถึงความแปรผันของข้อมูลทั้งหมดโดยรวมว่าเป็นอย่างไร เมตริกซ์โคแวเรียนหาได้จาก

$$C_x = \frac{1}{M}\sum_{i=1}^{M} x_i x_i^T - m_x m_x^T \qquad (7)$$

เมื่อ C_x คือเมตริกซ์โคแวเรียน (ขนาด 1024 x 1024)

x_i^T คือทรานสโพสเมตริกซ์ของ x_i

m_x^T คือทรานสโพสเมตริกซ์ของ m_x

เมตริกซ์ของไอเกนเวกเตอร์หาได้จากการนำเอาไอเกนเวกเตอร์ทั้ง 1024 เวกเตอร์ที่ได้จากเมตริกซ์โคแวเรียน (เมตริกซ์โคแวเรียนเป็นเมตริกซ์ที่สมมาตร ขนาด 1024 x 1024) โดยที่แต่ละแถวของเมตริกซ์ของไอเกนเวกเตอร์ จะประกอบด้วยสมาชิกที่ได้จากไอเกนเวกเตอร์ ซึ่งเรียงไอเกนเวกเตอร์ที่มีค่าของไอเกน (Eigen value) มากที่สุดอยู่แถวแรก และไอเกนมากรองมาอยู่ในแถวถัดไปตามลำดับ

การแปลงแบบเค-แอล (Karhunen Loeve Transform)

เป็นขั้นตอนที่ใช้ในการวิเคราะห์ตัวประกอบสำคัญ ซึ่งจะทำการแปลงข้อมูลภาพตัวอักษรขนาด 32 x 32 โดยใช้เมตริกซ์ค่าเฉลี่ยและเมตริกซ์ของไอเกนเวกเตอร์ที่ได้คำนวณเก็บไว้แล้วมาเป็นส่วนประกอบ ให้กลายเป็นข้อมูลเวกเตอร์รูปแบบของภาพตัวอักษรนั้น จากสมการ

$$y = A(x - m_x) \qquad (8)$$

เมื่อ y คือเวกเตอร์รูปแบบ (เมตริกซ์ขนาด 1024 x 1)

A คือเมตริกซ์ของไอเกนเวกเตอร์

x คือข้อมูลภาพตัวอักษรขนาด 32 x 32 (เมตริกซ์ขนาด 1024 x 1)

m_x คือเมตริกซ์ค่าเฉลี่ย (เมตริกซ์ขนาด 1024 x 1)

เนื่องจากการคำนวณโดยใช้ เมตริกซ์ A ขนาด 1024 x 1024 จะทำให้สิ้นเปลืองเวลาในการคำนวณมาก ถ้าทำการสร้างเมตริกซ์ของไอเกนเวกเตอร์ A_k จากไอเกนเวกเตอร์เพียง k เวกเตอร์ ซึ่งมีค่าของไอเกนมากที่สุด โดยให้สมาชิกในเมตริกซ์ A_k ในแถวที่ k + 1 ถึงแถวที่ 1024 เป็น 0 จะได้เวกเตอร์รูปแบบซึ่งสมาชิกในแถวที่ k + 1 ถึงแถวที่ 1024 เป็น 0 ซึ่งจะนำเวกเตอร์รูปแบบที่เป็นเมตริกซ์ขนาด k x 1 ไปใช้ในขั้นตอนต่อไป การทำเช่นนี้เสมือนว่าข้อมูลภาพขนาด 32 x 32 จะมีสัญญาณรบกวนซึ่งมีค่าความผิดพลาดกำลังสองเฉลี่ย (Mean Square Error) เท่ากับค่าของไอเกน ตัวที่ไม่ได้ใช้ (ตัวที่ k + 1 ถึงตัวที่ 1024) รวมกันตามสมการ

$$MSE = \sum_{i=k+1}^{N} l_i \qquad (9)$$

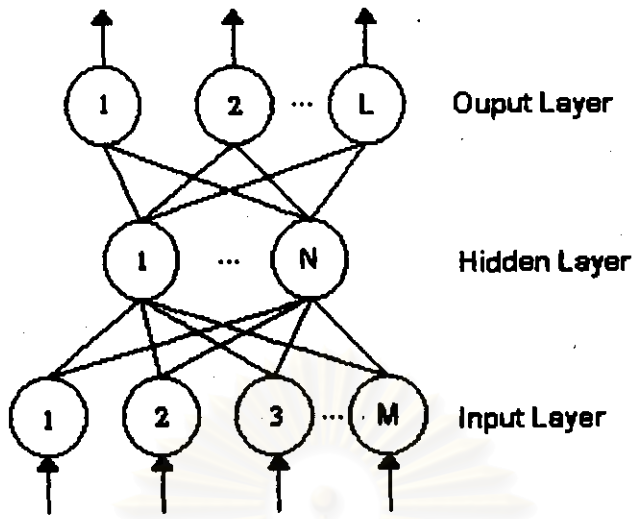
เมื่อ N คือจำนวนสมาชิกของเวกเตอร์ภาพขนาด 32 x 32 (N = 1024)

l_i คือค่าของไอเกนตัวที่ i

การใช้การแปลงแบบเค-แอลนี้มีแนวคิดว่ากลไกการมองเห็น (perception mechanism) ไม่มีรูปแบบ แต่มีการเชื่อมโยงกับข้อมูลลักษณะเด่นทางสถิติซึ่งประกอบขึ้นเป็นตัวอักษร ค่าไอเกนเวกเตอร์ของเมตริกซ์โคแวเรียนที่ได้จากภาพของตัวอักษรเป็นตัวประกอบสำคัญทางสถิติของความแปรผันจากภาพตัวอักษรต้นแบบ ค่าของไอเกน (Eigen value) แสดงถึงความสำคัญของไอเกนเวกเตอร์ตัวที่เกี่ยวข้อง กับตัวอักษรต้นแบบ ค่าของไอเกน ที่น้อยแสดงถึงความไม่เกี่ยวข้องหรือเกี่ยวข้องน้อยมากกับตัวอักษรต้นแบบ

การใช้นิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชัน (Backpropagation Neural Network)

เป็นขั้นตอนที่นำเอาเวกเตอร์รูปแบบของตัวอักษรที่ได้จากขั้นตอนการแปลงแบบเค-แอลมาทำการพิจารณาและแยกแยะตัวอักษร



รูปที่ 2.5 แสดงตัวอย่างของนิวรอลเน็ตเวิร์ก

นิวรอลเน็ตเวิร์กทำหน้าที่เป็น mapping network ซึ่งสามารถคำนวณหาลักษณะบางอย่างที่ก่อให้เกิดความสัมพันธ์ระหว่าง input และ output ได้ โดยที่นิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชัน เป็น Feed Forward Network ซึ่งจะไม่มีการป้อนผลลัพธ์ที่ได้ในแต่ละโหนดย้อนกลับไปยังโหนดที่ส่งข้อมูลมาให้ (แต่ในขั้นตอนการเรียนรู้จะมีการปรับปรุงค่าน้ำหนักของแต่ละโหนดเมื่อตอบคำตอบผิด โดยส่งข้อมูลย้อนกลับไปจึงเรียกว่า Backpropagation) โดยมีการเชื่อมโยงแบบ Fully Connected จากรูปที่ 2.5 แสดงให้เห็นเน็ตเวิร์กที่มี 3 layer คือ input layer, hidden layer และ output layer (hidden layer สามารถมีได้มากกว่า 1 layer)

ดังนั้น input ของ นิวรอลเน็ตเวิร์กก็คือเวกเตอร์ผลลัพธ์ที่ได้รับจากการแปลงแบบเค-แอล ซึ่งหากใช้สมาชิกของเวกเตอร์ผลลัพธ์ทั้ง 1024 ตัว ก็จะต้องใช้ input node จำนวน 1024 node หรือถ้าใช้สมาชิกของเวกเตอร์ผลลัพธ์ k ตัว ก็จะต้องใช้ input node จำนวน k node

การทำงานของนิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชันในขั้นตอนการรู้จำ

ในขั้นตอนการรู้จำ (Recognition Mode) จะมีการรับข้อมูล (สมาชิกของเวกเตอร์ผลลัพธ์) เข้าทาง input node (ในชั้น input layer) จากนั้นแต่ละ input node จะทำการส่งข้อมูล ไปยังทุกๆ hidden node (ในชั้น hidden layer) จากนั้นแต่ละ hidden node จะทำการคำนวณค่าที่ได้รับมาตามสมการ

$$O_j = F(\theta_j + \sum_{i=1}^{n} w_{ji}\, o_i) \qquad (10.1), (10.2)$$

$$F(a) = \frac{1}{1+e^{-a}}$$

เมื่อ O_j คือผลลัพธ์ของ node ที่ j

θ คือค่า bias ของ node ที่ j

w_{ji} คือค่าน้ำหนักของ node ที่ i ซึ่งส่งข้อมูล o_i มาให้ node ที่ j

o_i คือผลลัพธ์ของ node ที่ i

F() คือ Activation Function ซึ่งใช้การคำนวณแบบ Sigmoid Function

จากนั้น hidden node จะส่งผลลัพธ์ของตัวเอง ต่อไปยังทุกๆ output node (ในชั้น output layer) ซึ่งแต่ละ output node ก็จะคำนวณค่าที่ได้รับมาตามสมการข้างบน แล้วตอบผลลัพธ์ของ node ตัวเอง ซึ่งเราสามารถพิจารณาผลลัพธ์ของทุกๆ output node แล้วหาผลลัพธ์สุดท้ายได้ เช่น ถ้าผลลัพธ์ของ output node ที่ 1 มีค่าเท่ากับ 1 และผลลัพธ์ของ output node ตัวที่เหลือ เท่ากับ 0 ทั้งหมด แสดงว่า ข้อมูล input ที่ได้รับคือ ข้อมูลของตัวอักษร ก เป็นต้น

การทำงานของนิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชันในขั้นตอนการเรียนรู้

ก่อนที่จะนำนิวรอลเน็ตเวิร์กไปใช้งานในการรู้จำนั้น จะต้องทำการสอนนิวรอลเน็ตเวิร์ก (Train) ให้สามารถหาความสัมพันธ์ของ input และ output ให้ได้ถูกต้องหรือผิดพลาดน้อยที่สุดเสียก่อน (network convergence) ซึ่งเรียกว่าขั้นตอนการเรียนรู้ (Learning Mode, Training Mode) และมีวิธีการทำงานดังนี้ (Limin Fu [18])

- การกำหนดค่าเริ่มต้นแก่ค่าน้ำหนัก (weight) และ ค่า bias โดยการสุ่มตัวเลขที่มีค่าน้อยๆ เช่นระหว่าง –0.05 ถึง 0.05
- ส่งข้อมูลที่จะทำการสอนนิวรอลเน็ตเวิร์ก เข้าสู่ input node แล้วดำเนินการเหมือนในขั้นตอนการรู้จำ คือคำนวณค่าผลลัพธ์ของแต่ละ node (ยกเว้น input node) ตามสมการ

$$O_j = F(\theta_j + \sum_{i=1}^{n} w_{ji}\, o_i) \qquad (11.1), (11.2)$$

$$F(a) = \frac{1}{1+e^{-a}}$$

เมื่อ O_j คือผลลัพธ์ของ node ที่ j

θ คือค่า bias ของ node ที่ j

w_{ji} คือค่าน้ำหนักของ node ที่ I ซึ่งส่งข้อมูล o_i มาให้ node ที่ j

o_i คือผลลัพธ์ของ node ที่ I

F(a) คือ Activation Function ซึ่งใช้ Sigmoid Function

- ปรับค่าน้ำหนัก

เริ่มต้นที่ output node แล้วค่อยทำย้อนกลับไปยัง hidden node ในชั้น hidden layer ชั้นไป โดยคำนวณค่าน้ำหนักใหม่ ($W_{ji}(t+1)$) ได้ตามสมการ

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji} \tag{12}$$

เมื่อ $W_{ji}(t)$ คือค่าน้ำหนักของ node ที่ i ซึ่งเชื่อมกับ node ที่ j เมื่อเวลาที่ t (หรือ iteration ที่ t)

ΔW_{ji} คือค่าน้ำหนักที่ต้องการปรับ

ค่าน้ำหนักที่ต้องการปรับคำนวณได้จากสมการ

$$\Delta W_{ji} = \eta \delta_j O_i \tag{13}$$

เมื่อ η คือค่า Learning Rate ($0 < \eta < 1$)

δ_j คือค่าความผิดพลาดของผลลัพธ์ของ node ที่ j

แต่ในบางครั้งต้องการที่จะให้นิวรอลเน็ตเวิร์กนั้น convergence เร็วขึ้น ก็สามารถทำได้โดยเพิ่มค่า inertia หรือ momentum ตามสมการ

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_j O_i + \alpha(W_{ji}(t) - W_{ji}(t-1)) \tag{14}$$

เมื่อ α คือค่า Inertia หรือ momentum ($0 < \alpha < 1$)

ค่าความผิดพลาดของผลลัพธ์หาได้ดังนี้

หากเป็น output node ใช้สมการ

$$\delta_j = O_j(1 - O_j)(T_j - O_j) \tag{15}$$

เมื่อ T_j คือผลลัพธ์ที่ต้องการได้ (Target) จาก output node ที่ j

O_j คือผลลัพธ์ที่คำนวณได้ (Actual) จาก output node ที่ j

หากเป็น hidden node ใช้สมการ

$$\delta_j = O_j(1 - O_j)\sum_k \delta_k W_{kj} \tag{16}$$

เมื่อ δ_k คือค่าความผิดพลาดของผลลัพธ์ของ node ที่ k (ซึ่งเชื่อมต่อจาก node ที่ j)

W_{kj} คือค่าน้ำหนักของ node ที่ j ซึ่งเชื่อมกับ node ที่ k

ทำการปรับค่าน้ำหนักใน node อื่นจนครบ แล้วทำการสอนนิวรอลเน็ตเวิร์กในตัวอย่างถัดไป (iteration ถัดไป) จนกระทั่ง network convergence

เมื่อทำการเรียนรู้ของนิวรอลเน็ตเวิร์กเสร็จสิ้นลง จะทำการเก็บค่าน้ำหนักและค่า bias ของทุกๆ node เพื่อนำไปใช้ในการรู้จำต่อไป

การแก้ไขคำผิดที่เกิดจากโปรแกรมโอซีอาร์ด้วยวิธีการไตรแกรมของประเภทของคำ

การแก้คำผิดที่เกิดจากโปรแกรมโอซีอาร์ด้วยวิธีการไตรแกรมของประเภทของคำนั้น สามารถให้คำจำกัดความได้ว่าเป็นการหาลำดับของคำ $W = w_1 w_2 ... w_n$ ที่มีความน่าจะเป็น $P(W|S)$ สูงสุด เมื่อให้ $S = c_1 c_2 .. c_n$ เป็นสายอักขระที่เป็นผลลัพธ์จากโปรแกรมโอซีอาร์ ซึ่งสามารถหาได้จากสมการต่อไปนี้

$$\arg\max_W (P(W|S)) = \arg\max_W \left(\frac{P(W) \cdot P(S|W)}{P(S)} \right) \qquad (17.1)$$

และเนื่องจาก $P(S)$ เป็นค่าคงที่ของประโยคที่ได้จากโอซีอาร์ จะได้

$$\arg\max_W (P(W|S)) = \arg\max_W (P(W) \cdot P(S|W)) \qquad (17.2)$$

ค่าความน่าจะเป็น $P(W)$ สามารถหาได้โดยใช้แบบจำลองทางภาษา ซึ่งในโครงงานของอนันต์ลดา และ ชลวิช [15] ได้ใช้แบบจำลองประเภทของคำแบบไตรแกรม (POS trigram model) ในการหาค่าสถิติ ดังสมการต่อไปนี้

$$P(W) = P(W,T) = \prod P(t_i | t_{i-2} t_{i-1}) \cdot P(w_i | t_i) \qquad (17.3)$$

เนื่องจากการหาความน่าจะเป็นของลำดับของคำที่เป็นไปได้ทั้งหมดนั้น ต้องอาศัยฐานความรู้ขนาดใหญ่มาก จึงใช้วิธีการแทนลำดับของคำด้วยลำดับของประเภทของคำ $T = t_1 t_2 .. t_n$ แล้วหาความน่าจะเป็นของลำดับของประเภทของคำ $P(t_n | t_1 t_2 .. t_{n-1})$ และความน่าจะเป็นของคำเมื่อรู้ประเภทของคำ $P(w_i | t_i)$ แทน แต่เนื่องจากการหาความน่าจะเป็นของลำดับของประเภทของคำทั้งหมด ก็ยังต้องอาศัยฐานความรู้ขนาดใหญ่ จึงใช้ลำดับของประเภทของคำเพียง 3 ลำดับ (Part of speech Tri-gram) เพื่อหาความน่าจะเป็น $P(t_i | t_{i-2} t_{i-1})$ ซึ่งก็คือความน่าจะเป็นของประเภทของคำตัวที่ i เมื่อรู้ประเภทของคำตัวที่ i-2 และ i-1

สำหรับค่าความน่าจะเป็น $P(S|W)$ เป็นค่าที่แสดงถึงลักษณะเฉพาะของโปรแกรมโอซีอาร์แต่ละตัว โดยสามารถเก็บค่าสถิติได้จาก การเปรียบเทียบเอกสารต้นฉบับกับเอกสารผลลัพธ์ที่ได้จากโปรแกรมโอซีอาร์

ในขอบเขตงานวิจัยของอนันต์ลดา และ ชลวิช [15] และ ขอบเขตงานวิจัย Meknavin et al. [19] นั้นมุ่งที่จะแก้ไขคำผิดเฉพาะคำภาษาไทยเท่านั้น จึงไม่สามารถแก้ไขคำผิดที่เป็นภาษาอังกฤษได้

การหาค่าสถิติของความผิดพลาดที่เกิดขึ้นจากโปรแกรมโอซีอาร์

ค่าสถิติของความผิดพลาดที่เกิดขึ้นจากโปรแกรมโอซีอาร์หรือค่าความน่าจะเป็น $P(S|W)$ นอกจากเป็นค่าที่แสดงถึงลักษณะเฉพาะของโปรแกรมโอซีอาร์แต่ละตัวแล้ว ยังสามารถบอกความคล้ายคลึงกันระหว่างคำที่จะแก้ กับคำใกล้เคียงที่เลือกขึ้นมาได้ด้วย ซึ่งสามารถคำนวณหาความน่าจะเป็น $P(S|W)$ ได้ดังนี้

ให้ลำดับของคำ W ประกอบด้วยตัวอักษร $v_1, v_2, v_3, \ldots, v_m$ และผลลัพธ์จากโปรแกรมโอซีอาร์ S ประกอบด้วยตัวอักษร $c_1, c_2, c_3, \ldots, c_n$ ค่าความน่าจะเป็น $P(S|W)$ จะคำนวณได้จากการหาผลรวมของความน่าจะเป็น ในการแทนที่ตัวอักษรตัวหนึ่งด้วยอีกตัวอักษรหนึ่ง, การเพิ่มตัวอักษร และการลบตัวอักษร ที่จะทำให้ลำดับของตัวอักษรทั้ง 2 ตรงกัน

ให้ S_i เป็นอุปสรรคลำดับที่ i (i-prefix) ที่ประกอบด้วยอักษรตัวแรกจนถึงตัวที่ i ของสายอักขระ $S = (c_1, c_2, c_3, \ldots, c_i)$ และในทำนองเดียวกัน W_j เป็นอุปสรรคลำดับที่ j (j-prefix) ของ $W = (v_1, v_2, v_3, \ldots, v_j)$ โดยการใช้ขั้นตอนวิธีแบบ Dynamic Programming จะสามารถคำนวณ $P(S|W) = P(S_m, W_n)$ ได้จากสมการต่อไปนี้

$$\begin{aligned} P(S_i|W_j) = \max\{ & P(S_{i-1}|W_j)\cdot P(ins(c_j)), \\ & P(S_i|W_{j-1})\cdot P(del(v_j)), \\ & P(S_{i-1}|W_{j-1})\cdot P(c_i|v_j)\} \end{aligned} \quad (18)$$

โดยที่ $P(ins(c))$ คือค่าความน่าจะเป็นที่ตัวอักษร c จะถูกเพิ่ม

$P(del(v))$ คือค่าความน่าจะเป็นที่ตัวอักษร v จะถูกลบ

$P(c|v)$ คือค่าความน่าจะเป็นที่ตัวอักษร v จะถูกแทนที่ด้วยตัวอักษร c

การคำนวณแสดงได้ตามตัวอย่างดังนี้

P(พยายม|พยายาม) = P(พ|พ)· P(ย|ย)· P(า|า)· P(ย|ย)· P(del(า))· P(ม|ม)

แต่เนื่องจากคำผิดที่เกิดจากโปรแกรมโอซีอาร์ภาษาไทยมักเป็นคำผิดที่เกิดจากการเชื่อมต่อกันของตัวอักษร ดังนั้นตัวอักษรก่อนหน้าและตัวที่ตามมาอาจมีผลต่อความผิดพลาดของตัวอักษรที่ทำการพิจารณาด้วย อย่างไรก็ดีความผิดพลาดที่เกิดกับตัวอักษรบางตัว ก็ไม่มีผลกระทบมาจากตัวอักษรที่อยู่ใกล้เคียง ดังนั้นการนำเอาตัวอักษรใกล้เคียงมาพิจารณาต้องคำนึงด้วยว่าความผิดพลาดนั้นเกิดกับตัวอักษรอะไร โดย Meknavin et al. [19] ได้เสนอการแบ่งกลุ่มตัวอักษรออกเป็น 6 กลุ่ม ตามประเภทของตัวอักษรที่มีผลต่อความผิดพลาดของตัวอักษรในแต่ละกลุ่ม ดังแสดงในตาราง 2.1

กลุ่มที่	ระดับการวางของตัวอักษร	ตัวอักษร	ตัวอักษรที่มีผลกระทบ		ตัวอย่าง
			ตัวก่อน	ตัวตาม	
1	ระดับกลาง	ก ข ฃ ค ฅ ฆ ง จ ฉ ช ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ ท ธ น บ ผ พ ฟ ภ ม ย ร ล ว ษ ห อ ฯ ะ า เ แ	-	-	ค → ด
2	ระดับบน	อิ อี อึ อื อั อ่ อ้ อ๊ อ๋ อ์	✓	✓	ที่ → ที, ชิ้ → ชิ
3	ระดับล่าง	ุ ู	✓	-	ญู → ญ
4	ระดับกลางและบน กลุ่ม1	ฃ ฅ ป ฝ ฟ ศ ส ห ฮ อำ	-	✓	ส์ → ม
5	Upper Middle2	โ ไ ใ	✓	-	ห้ไ → หฟ
6	Lower Middle	ญ ฎ ฏ ฐ ฤ ๅ ๆ	-	✓	ฎู → ภ [7]

ตารางที่ 2.1 แสดงกลุ่มของตัวอักษรโดยแบ่งตามตัวอักษรที่มีผลกระทบต่อความผิดพลาด

[7] ลำดับของตัวอักษรที่ยกเป็นตัวอย่าง ไม่เกิดขึ้นจริงในภาษาไทย แต่สามารถเขียนได้ตามหลักการสะกด

ดังนั้นค่าความน่าจะเป็นในการเพิ่ม ลบ หรือแทนที่ตัวอักษรในสมการ (18) จะต้องพิจารณาตัวอักษรข้างเคียงประกอบด้วยซึ่งสามารถคำนวณได้ตามสมการ

$$P(ins(c \mid context)) = \frac{num(ins(c) \cap context)}{num(context)} \quad (19)$$

$$P(del(v \mid v \cap context)) = \frac{num(del(v) \cap v \cap context)}{num(v \cap context)} \quad (20)$$

$$P(c \mid v \cap context) = \frac{num(c \cap v \cap context)}{num(v \cap context)} \quad (21)$$

เมื่อ context คือตัวอักษรใกล้เคียงที่ต้องพิจารณาประกอบ ซึ่งขึ้นกับกลุ่มของตัวอักษรตามที่แสดงในตารางที่ 2.1

num() คือฟังก์ชันจำนวนตัวอักษรตามเงื่อนไขในวงเล็บที่เกิดขึ้นในเอกสารที่เราใช้เก็บค่าสถิติ

การคำนวณแสดงได้ตามตัวอย่างดังนี้

$$P(\text{ที่}|\text{ที}) = P(\text{ท}|\text{ท}) \cdot P(\text{ี}|\text{ที่}) \cdot P(\text{ins}(\text{ }|\text{ ี}))$$

'ท' เป็นตัวอักษรระดับกลางจึงไม่ต้องพิจารณาตัวอักษรอื่นในการหาค่าความน่าจะเป็นในการแก้ไข ส่วนตัวอักษร 'ี' และ ' ' เป็นตัวอักษรระดับบนจึงต้องพิจารณาตัวอักษรทั้งตัวก่อนหน้าและตัวที่ตามหลังในการหาค่าความน่าจะเป็น แต่เนื่องจาก ' ' เป็นตัวสุดท้ายของคำจึงพิจารณาได้เฉพาะตัวก่อนหน้าเท่านั้น

ขั้นตอนในการแก้ไขคำผิดที่เกิดจากโปรแกรมโอซีอาร์ภาษาไทย

ขั้นตอนในการแก้ไขคำผิดที่เกิดจากโปรแกรมโอซีอาร์ภาษาไทย ที่ใช้ในโครงงานของอนันต์ลดา และ ชลวิช [15] มีขั้นตอนดังนี้

1. หาคำที่ถูกต้องที่เป็นไปได้ทั้งหมดในประโยค โดยใช้ขั้นตอนวิธีในการตัดคำ แล้วนำมาต่อกันเพื่อหาสายอักขระทั้งหมดที่เป็นไปได้ ที่ประกอบด้วยคำที่มีอยู่ในพจนานุกรมเท่านั้น
2. หาส่วนของประโยคที่ประกอบด้วยคำผิดที่ไม่เป็นคำ เพื่อทำการแก้ไข
   2.1 ส่วนของประโยคที่ไม่ถูกครอบคลุมด้วยสายอักขระใดๆที่ได้จากข้อ1 จะถูกพิจารณาว่าเป็นสายอักขระที่ประกอบด้วยคำผิดที่ไม่เป็นคำ แต่ขั้นตอนวิธีที่ใช้ในการตัดคำ ไม่สามารถหาขอบเขตของคำผิดที่แน่นอนได้ เนื่องจากคำผิดที่มีอยู่ในประโยคอาจทำให้ตัดคำผิดได้ จึงทำให้ไม่สามารถสรุปได้ว่าสายอักขระที่มีความผิดพลาดนี้คือคำที่ผิด และคำที่ผิดอาจไม่ได้มีขอบเขตอยู่เพียงในสายอักขระนี้เท่านั้น แต่อาจคาบเกี่ยวถึงสายอักขระในข้อ1ด้วย เนื่องจากบางส่วนของคำที่ผิดอาจเป็นคำที่มีอยู่ในพจนานุกรมได้ เช่น 'เท ดโนโล ยี' 'เท' และ 'ยี' เป็นคำที่มีอยู่ในพจนานุกรม จึงถูกรวมอยู่ในสายอักขระที่ถูกต้องที่หาได้ในข้อ 1
   2.2 เพื่อให้การแก้ไขครอบคลุมส่วนที่ผิดพลาดทั้งหมด จึงต้องนำบางส่วนของสายอักขระที่ติดกับสายอักขระที่จะทำการแก้ไข มารวมเพื่อทำการแก้ไขด้วย

เช่น 'เท ดโนโล ยี' โดยที่ 'ดโนโล' เป็นสายอักขระที่ได้จากข้อ 2.1 การที่จะแก้เป็น 'เทคโนโลยี' ได้นั้นจะต้องนำ 'เท' และ 'ยี' ซึ่งเป็นส่วนหนึ่งของสายอักขระที่อยู่ติดกันมารวมด้วย

3. นำสายอักขระที่ประกอบด้วยคำผิดที่ไม่เป็นคำ ที่ได้จากข้อ 2.2 มาทำการแก้ไข โดยใช้สมมติฐานที่ว่าสายอักขระย่อยทุกอันที่เป็นไปได้คือคำที่จะทำการแก้ไข การแก้ไขจะทำโดยเลือกคำที่ใกล้เคียงกับคำที่จะทำการแก้ไขจากพจนานุกรม โดยจะดูจากระยะแก้ไขระหว่างคำที่จะถูกแก้กับคำที่เลือกขึ้นมาจากพจนานุกรม คำที่เลือกขึ้นมาจะต้องมีค่าระยะแก้ไขไม่เกินที่กำหนด ซึ่งค่าที่กำหนดนี้จะแปรผันตามความยาวของคำที่เลือกขึ้นมาจากพจนานุกรม ถ้าคำนั้นมีความยาวมากก็สามารถมีระยะแก้ไขจากคำที่ต้องการจะแก้ได้มาก
4. สร้างลำดับของคำที่เป็นไปได้ทั้งหมด โดยนำคำที่ถูกต้องอยู่แล้วซึ่งได้จากข้อ1 และคำใกล้เคียงซึ่งได้จากข้อ 3 มาต่อกัน จากนั้นนำมาคำนวณหาลำดับที่ดีที่สุดตามสมการที่ 17.2 โดยจะเลือกลำดับที่ดีที่สุดขึ้นมา N ลำดับ ซึ่งในโครงงานของ ชนันต์ลดา และ ขลวิช [15] ได้ให้ N มีค่าเท่ากับ 10 (หากต้องการแก้ไขเฉพาะคำผิดที่ไม่เป็นคำ จะนำเอาประโยคที่มีค่าสถิติดีที่สุดเป็นผลลัพธ์ของโปรแกรม)
5. ทำการแก้ไขคำผิดที่เป็นคำ สำหรับแต่ละลำดับของคำทั้ง N ลำดับ ดังนี้

   5.1 หาคำที่น่าจะเป็นคำที่ผิด โดยหาคำ w_i ที่มีความน่าจะเป็น $P(t_i|t_{i-2},t_{i-1})$ หรือ $P(w_i|t_i)$ ต่ำกว่าค่าที่กำหนด หรือเป็นไตรแกรมของประเภทของคำที่ไม่เคยเจอมาก่อน

   5.2 ทำการแก้ไขคำที่ผิดโดยใช้วิธีการตามข้อที่3

   และเพื่อป้องกันการแก้ไขคำที่ถูกอยู่แล้วให้ผิดไป เนื่องจากคิดว่าเป็นคำผิดที่เป็นคำ จึงได้มีการลดค่าความน่าจะเป็นของคำใกล้เคียงที่เลือกขึ้นมาแก้ด้วยอัตราที่กำหนดไว้ค่าหนึ่ง
6. เลือกลำดับที่ดีที่สุด N ลำดับ จากลำดับที่เป็นไปได้ทั้งหมดในข้อ 5.2 โดยดูจากค่าความน่าจะเป็นตามสมการที่ 17.2 เช่นเดียวกันในข้อ 4 ซึ่งในโครงงานนี้ได้ให้ N มีค่า 10 ซึ่งลำดับที่ดีที่สุดจะถูกเลือกเป็นประโยคที่แก้ไขแล้ว ซึ่งเป็นผลลัพธ์ของโปรแกรม

ระดับของตัวอักษรภาษาไทย

ระดับของตัวอักษรในภาษาไทยที่กำหนดขึ้นในวิทยานิพนธ์ฉบับนี้นั้น กำหนดขึ้นเพื่อใช้เป็นข้อมูลในการแก้ไขผลลัพธ์ที่ได้จากขั้นตอนการรู้จำตัวอักษร (ซึ่งไม่เกี่ยวข้องกับระดับตัวอักษรในการแก้ไขคำผิดที่เกิดจากโปรแกรมโอซีอาร์ภาษาไทย ตามตารางที่ 2.1 ซึ่งเป็นงานวิจัยของ

Meknavin et al. [19]) โดยพิจารณาถึงระดับของตัวอักษรที่ปรากฏในประโยค ซึ่งสามารถอธิบายได้ตามตารางที่ 2.2

ระดับของตัวอักษร	ตัวอักษร	หมายเหตุ
ระดับกลาง (Middle Level)	ก – ฮ ะ า เ ไ ใ ๆ ๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙	เป็นตัวอักษรที่ปรากฏระหว่างระดับบนและระดับล่าง
ระดับบน (Higher Level)	◌ ◌ ◌ ◌ ◌ ◌	เป็นตัวอักษรที่จะต้องปรากฏอยู่เหนือระดับกลาง แต่สามารถอยู่ได้ตัวอักษรในระดับบนสุดได้
ระดับบนสุด (Top Level)	◌ ◌ ◌ ◌ ◌ ◌	เป็นตัวอักษรที่จะต้องปรากฏอยู่ในระดับบนสุดเสมอ ไม่สามารถอยู่ได้ตัวอักษรในระดับอื่นได้
ระดับล่าง (Lower Level)	◌ ◌	เป็นตัวอักษรที่จะต้องปรากฏอยู่ได้ตัวอักษรระดับกลาง

ตารางที่ 2.2 แสดงกลุ่มของตัวอักษรที่ปรากฏอยู่ในระดับต่างๆ

การหาเส้นแบ่งระดับตัวอักษรภาษาไทย

การหาเส้นแบ่งระดับตัวอักษรภาษาไทยนั้นสามารถพิจารณาได้จากจำนวนจุดดำในรูปภาพบรรทัด เนื่องจากประโยคในภาษาไทยมีตัวอักษรในระดับกลางมากกว่าในระดับอื่น ซึ่งจะทำให้ในระดับกลางมีจำนวนจุดดำมากกว่าในระดับอื่น แสดงได้ตามรูปที่ 2.6

ขณะเดียวกัน เทคโนโลยีเด่นอีกอย่างที่กลุ่มสถาบันการเงินนำมาแสดงนั้น

รูปที่ 2.6 แสดงจำนวนจุดดำในแต่ละแถวตามแนวนอนของรูปภาพบรรทัด

การพิจารณาเส้นแบ่งระดับนั้น ดูได้จากความแตกต่างของจำนวนจุดดำในแถวก่อนหน้ากับจำนวนจุดดำในแถวปัจจุบัน โดยคำนวณได้จากสมการ

$$value_i = n_{i-1} - n_i \tag{22}$$

เมื่อ n_{i-1} คือจำนวนจุดดำในแถวที่ i –1

n_i คือจำนวนจุดดำในแถวที่ i

$value_i$ คือผลต่างของจำนวนจุดดำในแถวที่ i –1 กับแถวที่ i

เมื่อคำนวณครบทุกแถวแล้วจะได้ว่าค่า value ของแถวใดมีค่าน้อยที่สุด (เป็นค่าติดลบเสมอ) จะเป็นเส้นแบ่งระดับบนและกลาง และจะได้ว่าค่า value ของแถวใดที่มีค่ามากที่สุด (เป็นค่าบวกเสมอ) จะเป็นเส้นแบ่งระดับกลางและล่าง จากนั้นจะคำนวณหาเส้นกึ่งกลางของตัวอักษรในระดับกลาง โดยหาค่ากึ่งกลางระหว่างเส้นแบ่งระดับบน กับ เส้นแบ่งระดับล่าง