บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะนำเสนอ ทฤษฎีและงานวิจัยที่เกี่ยวข้องในวิทยานิพนธ์ฉบับนี้ โดยแบ่งออกเป็น 4 ส่วน คือ การตัดคำภาษาไทย การตรวจสอบและแก้ไขคำผิด หุ่นยนต์สนทนา และเอไอเอ็มแอล ซึ่งมีรายละเอียดของแต่ละส่วน ดังนี้

2.1 การตัดคำภาษาไทย

การตัดคำได้มีการพัฒนาอย่างต่อเนื่องมากว่า 20 ปี แต่ก็ยังไม่มีวิธีการใดที่ตัดคำได้ถูกต้องทั้งหมด และงานทางการด้านการประมวลผลภาษาธรรมชาติของภาษาไทยนั้น มีความจำเป็นที่ต้องใช้การตัดคำที่มีประสิทธิภาพมาก จากงานวิจัยที่ผ่านมาได้มีการนำเอากฎ พจนานุกรม ค่าสถิติ ไวยากรณ์ เข้ามาช่วยในการตัดคำ แต่ก็ยังไม่มีวิธีการใดที่สามารถตัดคำได้ถูกต้องทั้งหมด โดยสาเหตุที่ทำให้การตัดคำโดยพจนานุกรมนั้นไม่สามารถตัดคำได้ถูกต้องเนื่องมาจากสาเหตุดังต่อไปนี้

1. ปัญหาความกำกวม ซึ่งสามารถตัดได้หลายแบบ ทำให้เกิดความสับสนขึ้นว่าแบบไหนจะเป็นแบบที่ถูกต้องที่สุด เช่น "ตากลม" สามารถตัดคำได้เป็น "ตา-กลม" หรือ "ตาก-ลม"

2. ปัญหาคำศัพท์ที่ไม่ปรากฏในพจนานุกรม (เช่น คำเฉพาะที่เป็นชื่อคน ชื่อสถานที่) คำศัพท์ที่ไม่มีในพจนานุกรมนั้นเป็นสาเหตุทำให้ตัดคำนั้นผิด และอาจจะยังส่งผลให้คำรอบข้างมีการตัดคำผิดด้วย เช่น "ไมโครซอฟท์" สามารถตัดได้เป็น "ไม-โค-ร-ซอ-ฟท์"

วิธีการตัดคำภาษาไทยนั้นมีอยู่หลายวิธี แต่วิธีที่ให้ความถูกต้องสูงและนิยมใช้อยู่ในปัจจุบันมีอยู่ 2 วิธีด้วยกัน ได้แก่

2.1.1 การตัดคำภาษาไทยโดยใช้คุณลักษณะ

งานนี้เป็นของไพศาล เจริญพรสวัสดิ์ [4] โดยเป็นวิทยานิพนธ์ปีการศึกษา 2541 ซึ่งต่อมาได้นำมารวมเป็นส่วนหนึ่งของโปรแกรม SWATH (Smart Word Analysis for Thai) [5] โดยในงานวิจัยของไพศาลได้สรุปเกี่ยวกับงานด้านการตัดคำภาษาไทยไว้ว่า แนวคิดในการตัดคำนั้นมีอยู่หลายแนวคิด อันได้แก่

1. การตัดคำแบบเลือกคำยาวที่สุด (Longest Matching)
2. การตัดคำแบบเลือกแบบเหมือนที่สุด (Maximum Matching)
3. การตัดคำโดยโมเดลไตรแกรม (Trigram Model)

แต่อย่างไรก็ตามแนวคิดต่างๆ เหล่านั้นไม่สามารถให้ความถูกต้องที่สูงได้ ในการแก้ปัญหาการตัดคำ เนื่องจากวิธีการตัดคำ 2 แบบแรกจะใช้เพียงวิทยาการศึกษาสำนึก (Heuristics) และวิธีไตรแกรมนั้นก็มีการพิจารณาแค่บริบทก่อนหน้าแค่เพียง 2 คำเท่านั้น ส่วนความถูกต้องในการแก้ปัญหาความกำกวมนั้น มีความถูกต้องประมาณ 53 และ 73 เปอร์เซ็นต์สำหรับการตัดคำแบบที่สองและสาม ตามลำดับ

ไพศาลจึงได้เสนอแนวคิดการนำการเรียนรู้ของเครื่อง (Machine Learning) 2 แบบ คือ ริปเปอร์และวินโนว์ [6] มาทดสอบใช้ในการแก้ปัญหาการตัดคำภาษาไทย โดยคุณลักษณะที่นำมาใช้แก้ปัญหาคือ คำบริบท และสิ่งที่เกิดร่วมกันโดยมีลำดับ ซึ่งในการทดลองได้นำคลังข้อความ 80 เปอร์เซ็นต์ในการเรียนรู้ และใช้ที่เหลือในการทดสอบ โดยทำการวัดประสิทธิภาพใน 2 ด้าน คือ

- การแก้ปัญหาความกำกวม : ให้ความถูกต้องมากกว่า 85 และ 90 เปอร์เซ็นต์
- การแก้ปัญหาคำศัพท์ที่ไม่ปรากฏในพจนานุกรม : ให้ความถูกต้องมากกว่า 70 และ 80 เปอร์เซ็นต์ (ด้วยวิธีการริปเปอร์และวินโนว์ ตามลำดับ)

เนื้อหาวิทยานิพนธ์ของไพศาลได้แสดงให้เห็นว่า การตัดคำโดยใช้คุณลักษณะการเรียนรู้ของเครื่องนั้นให้ความถูกต้องที่มากกว่าการตัดคำโดยใช้โมเดลไตรแกรมและการตัดคำโดยเลือกแบบเหมือนที่สุด และยังแสดงให้เห็นว่าวินโนว์สามารถตัดคำได้ดีกว่าริปเปอร์

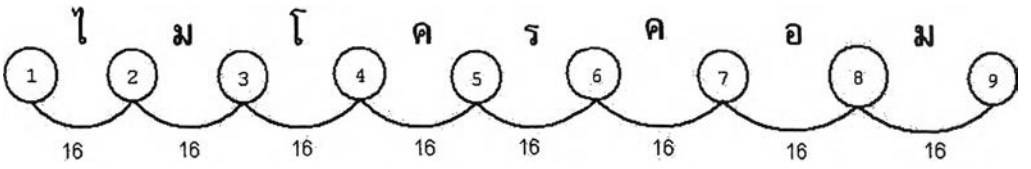
แม้งานวิจัยของไพศาลจะได้แก้ปัญหาทั้ง 2 อย่างของการตัดคำด้วยพจนานุกรมซึ่งได้ผลดีขึ้นแล้ว แต่จากผลการทดลองใช้โปรแกรม SWATH ก็ยังพบข้อเสีย คือ ผลของการตัดคำที่ได้บางครั้งจะได้คำที่มีอักษรเพียงตัวเดียว ซึ่งในหลักภาษาไทย [7] นั้นตัวอักษรเพียงตัวเดียวไม่สามารถเกิดเป็นคำได้ ยกเว้น ณ และ ธ

2.1.2 โปรแกรมตัดคำภาษาไทยเวิร์ดคัท (Wordcut)

งานนี้เป็นของวีร์ สัตยมาศ [8] ในปี 2546 ซึ่งมีการใช้กฎทางอักขระวิธีร่วมกับการใช้พจนานุกรม เพื่อรองรับคำที่ไม่ปรากฏในพจนานุกรม ซึ่งเหมาะสมสำหรับการตรวจสอบตัวสะกด สามารถอธิบายได้ดังนี้

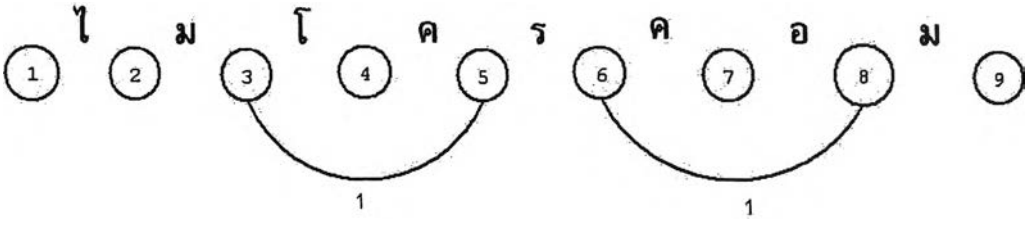
ให้ S เท่ากับสตริงข้อความนำเข้า โดยสมมติให้เท่ากับ "ไมโครคอม"

1) สร้างบัพ (Node) แทรกระหว่างแต่ละอักขระของ S และหัวท้าย เชื่อมทุกบัพด้วยเส้นเชื่อมซึ่งมีต้นทุน (Cost) เท่ากับ 2 เท่าของความยาวของ S จากคำว่า "ไมโครคอม" จะได้ดังรูปที่ 2.1



รูปที่ 2.1 กราฟแสดงการเพิ่มเส้นเชื่อมเพื่อแทนคำที่ไม่รู้จัก

2) ค้นหาคำซึ่งเป็นส่วนหนึ่งของ S ที่ปรากฏในพจนานุกรม สร้างเส้นเชื่อมแทนคำที่พบในพจนานุกรมและให้ต้นทุนเท่ากับ 1 ในกรณีที่ S เท่ากับ "ไมโครคอม" จะพบคำว่า "โค" และ "คอ" ปรากฏในพจนานุกรม ทำให้แสดงกราฟได้ดังรูปที่ 2.2



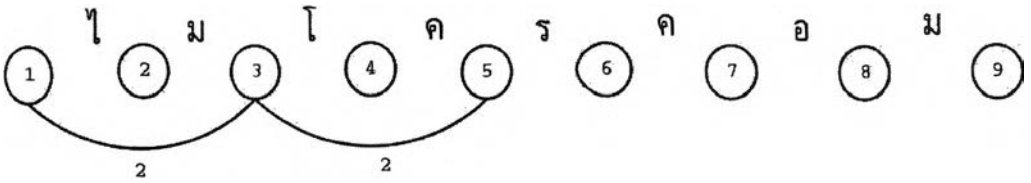
รูปที่ 2.2 กราฟแสดงการเพิ่มเส้นเชื่อมเพิ่มแทนคำที่มีในพจนานุกรม

3) เพื่อแก้ไขปัญหาคำที่ไม่ปรากฏในพจนานุกรมจึงใช้กฎทางอักขระวิธีร่วมกับการใช้พจนานุกรม โดยมีกฎในการจับกลุ่มอักขระ ซึ่งแสดงในรูปเรคกูลาร์เอกเพรสชันได้ดังรูปที่ 2.3

```
[เ-แ][ก-ฮ][ั-ื][ ็- ์]?.
เ[ ^ก]ื.
กื
[เ-ไ][ก-ฮ]
. .
.[ ็- ์]
...
[ก-ฮ]ื[ ็- ์]..
[ก-ฮ]ื[ก-ฮ].
[ก-ฮ][ะ- ู]
```

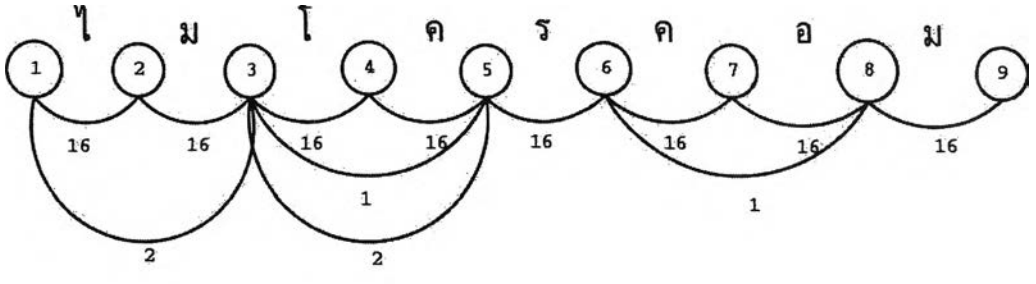
รูปที่ 2.3 กฎในการจับกลุ่มอักขระ

สร้างเส้นเชื่อมแทนกลุ่มคำที่ตรงตามกฎทางอักขระวิธีและให้ต้นทุนเท่ากับ 2 ซึ่งในกรณี S เท่ากับ "ไมโครคอม" จะได้คำว่า "ไม" และ "โค" ทำให้แสดงกราฟได้ดังรูปที่ 2.4



รูปที่ 2.4 กราฟแสดงการเพิ่มเส้นเชื่อมเพื่อแทนกลุ่มอักขระตามกฎการจับกลุ่มอักขระ

4) นำกราฟได้จากข้อ 1 2 และ 3 มาเขียนร่วมกัน เพื่อแสดงกราฟวิธีที่เป็นไปได้ในการตัดคำดังรูปที่ 2.5



รูปที่ 2.5 กราฟวิธีที่เป็นไปได้ในการตัดคำ

จากนั้นหาเส้นทางที่สั้นที่สุด โดยให้ต้นทุนในเส้นทางนั้นน้อยที่สุดจากบัพซ้ายสุด ไปยังบัพขวาสุด แล้วตัดคำตามเส้นทางที่ได้จะได้ "ไม-โค-ร-คอ-ม"

5) เนื่องจากในภาษาไทยเมื่อมีคำที่มีอักขระเดียวจะถูกแบ่งด้วยเครื่องหมายวรรคตอนเช่น "ณ ระนอง" "ธ ประสงค์ใด" ดังนั้นอักษรเดี่ยวที่พบได้ในผลลัพธ์ของการตัดคำจึงมีโอกาสเป็นส่วนหนึ่งของคำเมื่อรวมกับกลุ่มอักขระข้างหน้าหรือข้างหลัง

ดังนั้นเมื่อพบอักขระเดี่ยวจึงทำการรวมอักขระนั้นเข้ากับกลุ่มคำข้างหน้า หรือรวมกับกลุ่มคำข้างหลังเมื่อไม่มีกลุ่มคำข้างหน้า ทั้งนี้เนื่องจากการรวมอักขระเดี่ยวเข้ากับกลุ่มคำข้างหน้ามีโอกาสถูกต้องมากกว่าการรวมอักขระเข้ากับกลุ่มคำข้างหลัง จากการสังเกตจากคลังข้อมูลตัวอย่างเช่น "ไม-โค-ร-คอ-ม" เมื่อรวมอักขระเดี่ยว จะได้เป็น "ไม-โคร-คอม"

โดยโปรแกรม Wordcut นี้มีข้อดีตรงที่สามารถกำจัดตัวอักษรตัวเดียวทิ้งได้ แต่ก็มีข้อเสียคือ ไม่สามารถแก้คำกำกวมได้ดีเท่าวิธีการของโปรแกรม SWATH

2.2 การตรวจสอบและแก้ไขคำผิด

จากงานวิจัยของ Karen Kukich [9] ได้รวบรวมปัญหาและวิธีการที่ใช้จัดการกับความผิดพลาดของข้อความภาษาอังกฤษ ซึ่งสามารถแยกออกเป็น 3 ส่วน ตามกลุ่มของการวิจัย คือ

2.2.1 **การตรวจสอบส่วนที่ไม่เป็นคำ** (Nonword Error Detection)

เป็นการตรวจสอบหาข้อความที่ไม่ปรากฏในรายการของคำ พจนานุกรม หรือคลังศัพท์ วิธีที่นิยมใช้มี 2 วิธีคือ

- การวิเคราะห์เอ็นแกรม (N-gram Analysis) เหมาะสำหรับการตรวจสอบความผิดพลาดที่เกิดจากเครื่องมือ เช่น จากเครื่องรู้จำอักขระ (Optical Character Recognizer: OCR) แต่มีความแม่นยำน้อยสำหรับการตรวจสอบความผิดพลาดที่เกิดจากมนุษย์

- การค้นหาในพจนานุกรม (Dictionary Lookup) เป็นวิธีที่นิยมใช้สำหรับการตรวจสอบตัวสะกด โครงสร้างที่ใช้สร้างพจนานุกรม ได้แก่ ตารางแฮช (Hash table) หรือทรัย (Trie)

แต่ในงานวิจัยของกลุ่มนี้ ก็ยังไม่ได้คำนึงถึงเรื่องสาเหตุอื่นที่ทำให้เกิดคำผิด เช่น ความผิดพลาดที่เกิดจากมนุษย์ เข้ามาใช้เป็นองค์ประกอบในการจัดลำดับรายการคำใกล้เคียง

2.2.2 **การแก้ไขคำผิดโดยไม่คำนึงถึงสภาพรอบข้าง** (Isolated-word Error Correction)

จากงานวิจัยของ Damerau [10] ได้ค้นพบว่า ประมาณ 80 เปอร์เซ็นต์ของคำที่พิมพ์ผิดทั้งหมดเกิดขึ้นจากการประกอบกันของความผิดพลาด 4 ชนิด ดังนี้

1. การแทรก (Insertion) เช่น นักเรียน พิมพ์เกินเป็น นักเรียนน
2. การลบ (Deletion) เช่น อาหาร พิมพ์ตกเป็น อหาร
3. การแทนที่ (Substitution) เช่น อักษร พิมพ์ผิดเป็น อัการ
4. การสลับ (Transposition) เช่น สมัคร พิมพ์สลับเป็น สมัรค

จากงานวิจัยของ Grudin [11] ได้กล่าวถึงการศึกษารูปแบบของการพิมพ์ผิดในระดับความชำนาญต่างๆ กัน ได้ผลคือ ผู้ที่พิมพ์ชำนาญมักพิมพ์เกินใน 2 คีย์ที่อยู่ติดกัน ในขณะที่ผู้ไม่ชำนาญมักผิดพลาดจากการพิมพ์ผิด และการพิมพ์ผิดนั้นมักเกิดจากคีย์ที่อยู่ติดกัน และได้ค้นพบอีกว่า การพิมพ์ผิดนั้น ผู้พิมพ์มักพิมพ์อักษรที่มีการใช้งานสูง (High-Frequency Letter) แทนที่อักษรที่มีการใช้งานต่ำ (Low-Frequency Letter)

ระยะแก้ไข (Edit Distance) [10] ระหว่างสตริง 2 สตริง หมายถึงจำนวนครั้งที่น้อยที่สุดที่ต้องการในการแก้ไขโดยการเพิ่ม ลบ หรือเปลี่ยน อักขระของสตริงหนึ่งให้เท่ากับอีกสตริงหนึ่ง เช่น

- นักเรียน กับ นักเรียนน มีระยะทางของการแก้ไขเท่ากับ 1
- สมัคร กับ สมัรค มีระยะทางของการแก้ไขเท่ากับ 2

การแก้ไขคำผิดที่ได้จากการหาส่วนที่ไม่เป็นคำ วิธีที่นิยมใช้คือ การหาคำในพจนานุกรมที่ใกล้เคียงกับคำที่ผิดมากที่สุด คือมีระยะแก้ไขน้อยที่สุด (Minimum Edit Distance) มาใช้เป็น

ปัจจัยหลักในการจัดลำดับรายการคำใกล้เคียงคำผิด แต่ก็ยังไม่สามารถแก้ปัญหาในกรณีที่ได้ผลลัพธ์เป็นคำที่มีระยะทางเท่ากันหลายรายการได้

2.2.3 **การแก้ไขคำผิดโดยขึ้นกับรูปประโยค** (Context-dependent Word Correction)

กลุ่มนี้เป็นการแก้ไขคำผิดโดยการวิเคราะห์ถึงระดับไวยากรณ์ของประโยค ซึ่งค่อนข้างซับซ้อนและอยู่นอกเหนือขอบเขตของงานวิจัยนี้

2.3 หุ่นยนต์สนทนา

จากการสำรวจเทคนิคต่างๆ ที่ใช้สร้างหุ่นยนต์สนทนาจากทางอินเทอร์เน็ตและจากหนังสือ Virtual Humans [12] พบว่าเทคนิคที่นิยมอยู่ในปัจจุบัน (ตุลาคม 2548) มีอยู่ด้วยกัน 2 เทคนิค คือ

1. ภาษาวายไอเอ็มแอล (YIML) : ใช้สร้างหุ่นยนต์ชื่อ Yapanda [3] แต่ภาษาวายไอเอ็มแอลนั้นได้หยุดการพัฒนาต่อแล้ว

2. ภาษาเอไอเอ็มแอล (AIML) : ใช้สร้างหุ่นยนต์ชื่อ ALICE [2] ซึ่งได้รับรางวัล Loebner Prize ในปี 2001 2003 และ 2004 โดย ALICE นั้นมีต้นแบบมาจากหุ่นยนต์ชื่อ ELIZA [13]

โดยภาษาเอไอเอ็มแอล มีข้อดี คือ สามารถเขียนคำสั่ง (Script) ได้ง่าย มีคนใช้เทคนิคนี้อยู่จำนวนมาก และมี Open source tool ให้ใช้ แต่ก็ยังมีข้อเสีย คือ ไม่สามารถจัดการกับประโยคของภาษาที่ไม่มีการใช้สัญลักษณ์หรือตัวอักษรคั่นระหว่างคำ เช่น ภาษาไทย จีน ญี่ปุ่น ได้

สัญลักษณ์คั่นระหว่างคำนั้นเป็นคุณสมบัติสำคัญที่ ALICE ต้องพึ่งพาในการแยกแยะวิธีการโต้ตอบกลับ ดังนั้นภาษาที่ไม่มีสัญลักษณ์คั่นระหว่างคำในประโยคจึงจำเป็นที่จะต้องมีการตัดคำเสียก่อน จึงจะทำให้สามารถนำ ALICE และเอไอเอ็มแอลไปใช้สร้างหุ่นที่โต้ตอบด้วยภาษาไทยได้

2.4 เอไอเอ็มแอล (AIML-Artificial Intelligence Markup Language) [14]

เอไอเอ็มแอล (AIML) ย่อมาจาก Artificial Intelligence Markup Language ซึ่งถูกพัฒนาขึ้นมาโดยทีมของ The Alicebot free software community และ Dr. Richard S. Wallace เมื่อปีคริสตศักราช 1995-2000

ภาษาเอไอเอ็มแอล มีหน่วยพื้นฐาน (Basic Unit) ของฐานความรู้ (Knowledge Base) เรียกว่า แคตากอรี (Category) โดยแต่ละแคตากอรีประกอบด้วยส่วนประกอบหลัก ดังนี้

- คำถามนำเข้า (Input Question) เรียกว่า แพทเทิร์น (pattern)

- คำตอบส่งออก (Output Answer) เรียกว่า เทมเพลต (template)
- บริบททางเลือก (Optional Context) มีอยู่ 2 ชนิด คือ แดท (that) และ ทอพิค (topic)

รูปแบบของภาษาเอไอเอ็มแอลนั้นประกอบไปด้วย คำ(Words) ช่องว่าง(Spaces) สัญลักษณ์ไวล์คาร์ด (Wildcard symbols : _) และสตาร์ (*) โดยที่คำนั้นสามารถเป็นตัวอักษรหรือตัวเลขก็ได้ แต่ต้องไม่ใช่อักขระอื่นๆ ซึ่งคำในประโยคจะแบ่งแยกระหว่างคำโดยใช้ช่องว่างเพียง 1 ตัว (a Single Space)

มาตรฐานเอไอเอ็มแอล อนุญาตให้มีไวล์คาร์ด (Wildcards) หลายที่ได้ในแต่ละแพทเทิร์น (Pattern) แต่ภาษาได้ถูกออกแบบมาเพื่อให้ง่ายเท่าที่จะเป็นไปได้สำหรับงานที่ใช้มือ ซึ่งจะง่ายกว่านิพจน์ปกติ (Regular Expression)

เทมเพลต (Template) คือ แท็กคำตอบของเอไอเอ็มแอล โดยอยู่ในรูปแบบที่ง่าย ซึ่งประกอบไปด้วย เพลนเท็กซ์ (Plain text) และอันมาร์คเท็กซ์ (Unmarked text)

โดยทั่วไปแล้วแท็ก (tag) ของเอไอเอ็มแอลไม่ได้มีความสามารถเพียงตอบคำถามได้เท่านั้น เนื่องจากสามารถเก็บข้อมูล เรียกใช้งาน (Activate) โปรแกรมอื่น ให้เงื่อนไขการตอบ (Conditional Responses) เรียกซ้ำและแทรกคำตอบที่นำมาจากแคตากอรีอื่นได้

บริบททางเลือกของแคตากอรีมีให้เลือกใช้อยู่ 2 แท็ก คือ <that> และ <topic> ซึ่ง แดท ทำหน้าที่จำสิ่งที่พูดไปแล้ว และ ทอพิค ทำหน้าที่รวมแคตากอรีหลายๆ อัน ให้เป็นกลุ่มเดียวกัน โดยที่ แดท จะเขียนอยู่ภายในแท็กแคตากอรี แต่ ทอพิค จะเขียนอยู่ภายนอกแท็กแคตากอรี

ตัวอย่างการเขียนคำถาม-คำตอบอย่างง่าย ด้วยเอไอเอ็มแอล

```
<category>
<pattern>How are you</pattern>
<template>Fine, thanks</template>
</category>
```

เอไอเอ็มแอลไม่ได้เป็นฐานข้อมูลของคำถาม-คำตอบเท่านั้น เพราะว่าใช้วิธีการที่เรียกว่าการจับคู่รูปแบบ (Pattern Matching) ซึ่งคล้ายกับภาษาสอบถาม (Query Language) แต่จะง่ายกว่า และคำตอบนั้นไม่จำเป็นว่าจะจับคู่ได้เพียงแคตากอรีเดียว เนื่องจากเอไอเอ็มแอลสามารถยอมให้คำตอบในแคตากอรีมีแท็กเรียกซ้ำ (Recursive Tags) ที่เรียกว่า srai ได้ โดยเอไอเอ็มแอลนั้นจะใช้การเรียกซ้ำ (Recursion) เป็นหลักสำคัญ

ตัวอย่างการใช้งานหลักๆ ของการเขียนเอไอเอ็มแอล มีดังต่อไปนี้

1) Symbolic Reduction

ประโยคที่ยาวๆ บางประโยค สามารถเขียนเป็นประโยคที่มีใจความเดียวกันแต่สั้นกว่ากันได้ เช่น "DO YOU KNOW WHO SOCRATES IS" อาจเขียนได้เป็น "WHO IS SOCRATES" ซึ่งจะให้ความหมายในทำนองเดียวกัน แต่ประโยคหลังจะสั้นกว่า ดังนั้นทั้ง 2 ประโยคอินพุตก็ควรจะมีรูปแบบคำตอบที่เหมือนกัน ทำให้สามารถใช้แท็ก <srai> ในการเรียกซ้ำคำตอบของประโยคที่สองมาเป็นคำตอบของประโยคแรกได้ ดังตัวอย่าง

```
<category>
<pattern>DO YOU KNOW WHO * IS</pattern>
<template><srai>WHO IS <star/></srai></template>
</category>
```

ในบางตำแหน่งของประโยคที่เป็นชื่อเฉพาะ เช่น ชื่อคน อาจลดรูปได้โดยใส่ * ลงไปแทนที่ ซึ่งทำให้อินพุตอะไรก็ตามที่จับคู่ (match) ได้กับแพทเทิร์นนี้ จะถูกนำมาแทรกคำตอบในส่วนที่เป็น * ลงไปในแพทเทิร์นอีกทีหนึ่ง

2) Divide and Conquer

ประโยคสามารถแยกออกได้เป็นส่วนย่อยของประโยคได้ หากมีส่วนย่อยหลายส่วนจะแยกกันทำแล้วนำคำตอบมารวมกัน ดังตัวอย่างข้างล่างนี้เป็นการใช้คำตอบของกฎ Yes มาต่อกับคำตอบของกฎที่ใช้กับ *

```
<category>
<pattern>YES *</pattern>
<template><srai>YES</srai> <sr/></template>
</category>
```

หมายเหตุ <sr/> เป็นแท็กย่อของ <srai><star/></srai>

3) Synonyms

มาตรฐานของเอไอเอ็มแอล เวอร์ชัน1.0.1 ไม่อนุญาตให้มีแพทเทิร์นมากกว่า 1 อันในแต่ละแคตากอรี แต่ถ้าหากมีแคตากอรีที่มีแพทเทิร์นที่มีความหมายในทำนองเดียวกันหลายอัน ซึ่งจะ

มีรูปแบบคำตอบในทำนองเดียวกัน ก็สามารถใช้แท็ก <srai> เพื่อเรียกซ้ำใช้งานคำตอบของแคตากอรีที่มีอยู่แล้วได้ ดังในตัวอย่าง

```
<category>
<pattern>HELLO</pattern>
<template>Hi there!</template>
</category>

<category>
<pattern>HI</pattern>
<template><srai>HELLO</srai></template>
</category>

<category>
<pattern>HI THERE</pattern>
<template><srai>HELLO</srai></template>
</category>

<category>
<pattern>HOWDY</pattern>
<template><srai>HELLO</srai></template>
</category>

<category>
<pattern>HOLA</pattern>
<template><srai>HELLO</srai></template>
</category>
```

จะเห็นได้ว่าแพทเทิร์น "HELLO" , "HI" , "HI THERE" , "HOWDY" , "HOLA" จะมีความหมายในทำนองเดียวกัน ในเมื่อมีคำตอบของแพทเทิร์น "HELLO" อยู่ก่อนแล้ว ก็สามารถเรียกซ้ำใช้คำตอบของแคตากอรีที่มีอยู่แล้วได้

4) Spelling and Grammar correction

ผู้ใช้มักจะสะกดผิดโดยนำภาษาพูดมาใช้เขียนแทนภาษาเขียน เช่น ใช้ "your" แทนที่ควรจะเป็น "you're" หรือ "you are" ซึ่งสามารถใช้แท็ก <srai> ในการช่วยเรียกซ้ำคำตอบที่น่าจะเป็นมาแทนที่ประโยคที่ผิดหลักไวยากรณ์นั้นได้ ดังในตัวอย่างจะเป็นการใช้กฎของ "YOU ARE A" มาตอบคำถามที่มี "YOUR A" แทน

```
<category>
<pattern>YOUR A *</pattern>
<template>I think you mean "you're" or "you are" not "your." <srai>YOU ARE A
<star/></srai>
</template>
</category>
```

5) Keywords

หากต้องการสร้างคำตอบของเทมเพลตโดยการดูจากคีย์เวิร์ด (keyword) ที่มีอยู่ในประโยคนำเข้า (input sentence) จะสามารถใช้ <srai> ช่วยสร้างได้ ดังในตัวอย่าง มีแคตากอรีอยู่ 4 อัน แล้วทุกแคตากอรีต้องการตอบเหมือนกันโดยที่ดูจากคีย์เวิร์ดคำว่า "MOTHER" ซึ่งไม่ว่าจะปรากฏคีย์เวิร์ดที่ตำแหน่งใดๆ ของประโยคนำเข้าก็ตาม โดยแคตากอรีที่สองจะมีคีย์เวิร์ดเป็น suffix ส่วนแคตากอรีที่ 3 จะมีคีย์เวิร์ดเป็น prefix และสุดท้ายแคตากอรีที่ 4 จะมีคีย์เวิร์ดเป็น infix ในประโยค ตามลำดับ ซึ่งทั้ง 4 แคตากอรีจะให้คำตอบเหมือนกัน คือ ตอบตามกฎของแคตากอรี "MOTHER"

```
<category>
<pattern>MOTHER</pattern> <template> Tell me more about your family. </template>
</category>

<category>
<pattern>_ MOTHER</pattern> <template><srai>MOTHER</srai></template>
</category>
```

```
<category>
<pattern>MOTHER _</pattern> <template><srai>MOTHER</srai></template>
</category>

<category>
<pattern>_ MOTHER *</pattern><template><srai>MOTHER</srai></template>
</category>
```

6) Conditionals

สามารถสร้างกิ่งเงื่อนไข (conditional branches) โดยใช้แท็ก <srai> ได้ ซึ่งพิจารณาได้ดังแคตากอรีทั้ง 3 ดังในตัวอย่างนี้

```
<category>
<pattern>WHO IS HE</pattern>
<template><srai>WHOISHE <get name="he"/></srai></template>
</category>

<category>
<pattern>WHOISHE *</pattern>
<template>He is <get name="he"/>.</template>
</category>

<category>
<pattern>WHOISHE UNKNOWN</pattern>
<template>I don't know who he is.</template>
</category>
```

หากค่าที่ได้จากแท็ก <get> ไม่ใช่ null (ใน AIML คือ UNKNOWN) ก็จะทำในแคตากอรีที่สาม แต่ถ้าหากได้ค่าจากแท็ก <get> เป็นค่าอื่นๆ (คือมีการป้อนค่าพารามิเตอร์ name เก็บเอาไว้อยู่ก่อนหน้า) ก็จะทำในแคตากอรีที่สองแทน

7) That

แท็ก <that> จะใช้ในการอ้างอิงว่าก่อนหน้านี้หุ่นยนต์ได้ตอบอะไรออกไป เพื่อช่วยในกรณีของการตอบคำถามถัดมาที่ผู้ใช้ตอบมาเพียงแค่ว่า "ใช่-ไม่ใช่" ซึ่งหากไม่รู้ว่าก่อนหน้านี้คุยอะไรกันอยู่จะทำให้ยากต่อการตัดสินใจว่าจะคุยอะไรต่อไป และนอกจากนี้ยังทำให้การคุยนั้นมีความต่อเนื่องเป็นบทสนทนาเรื่องเดียวกันต่อได้ ซึ่งมักจะพบแท็กนี้ได้ใน ประโยคนำเข้าประเภทใช่-ไม่ใช่ (yes-no questions) เป็นหลัก

<category>
<pattern>YES</pattern>
<that>DO YOU LIKE MOVIES</that>
<template>What is your favorite movie?</template>
</category>

จากในตัวอย่าง หากมีประโยคนำเข้าว่า YES ก็จะต้องมาดูอีกทีว่าก่อนหน้านี้หุ่นได้ตอบสนองกับผู้ใช้ไว้ว่าอย่างไร ถ้าหากก่อนหน้านี้หุ่นยนต์ได้ถามไปว่า DO YOU LIKE MOVIES จึงจะนำคำตอบในแคตากอรีนี้มาตอบสนองออกไปให้ผู้ใช้ทราบเป็นคำตอบที่อยู่ในแคตากอรีนี้ว่า What is your favorite movie?

8) Topic

ภายในเอไอเอ็มแอลอินเทอร์พรีเตอร์ (AIML interpreter) จะเก็บ input pattern, that pattern และ topic pattern ตามพาธเดี่ยว (Single path) ซึ่งมีลักษณะคล้ายดังนี้

INPUT <that> THAT <topic> TOPIC

(หากค่าของ that และ topic ไม่ได้ถูกระบุไว้ ระบบจะตั้งค่าเป็นไวล์คาร์ด * ไว้ให้)

หากอินพุตแพทเทิร์นนั้นสามารถ match ได้หลายแคตากอรี การตัดสินใจว่าจะเลือกแคตากอรีไหนตอบนั้น จะต้องพิจารณาจากแท็ก that อีกทีหนึ่ง แต่ถ้าหากพิจารณาแล้วยังได้แคตากอรีอีกหลายทางเลือกอยู่อีก ก็ให้ทำการเลือกโดยดูจากแท็ก topic เป็นขั้นสุดท้าย

โดยมีข้อแนะนำในการเขียนกฎโครงสร้าง ดังนี้คือ

- อย่าใช้แท็ก <that> จนกว่าจะมีแคตากอรีที่มี <pattern> เหมือนกัน เกิดขึ้น
- อย่าใช้แท็ก <topic> จนกว่าจะมีแคตากอรีที่มี <pattern> และ <that> เหมือนกัน เกิดขึ้น

ตัวอย่างการใช้งานแท็ก topic ซึ่งเป็นการรวมเรื่องในหัวข้อเดียวกันไว้ด้วยกัน ซึ่งในที่นี้หัวข้อคือ CARS

```
<topic name="CARS">
<category>
<pattern>*</pattern>
<template>
<random>
<li>What's your favorite car?</li>
<li>What kind of car do you drive?</li>
<li>Do you get a lot of parking tickets?</li>
<li>My favorite car is one with a driver.</li>
</random>

</template>
</category>
```

หมายเหตุ botmaster สามารถใช้แท็ก <set> ในการเปลี่ยนค่าของ topic เพื่อเปลี่ยนหัวข้อบทสนทนาได้

9) Get

หากต้องการนำหุ่นสนทนาหลายๆ ตัวไปทำงานบนเว็บเซิร์ฟเวอร์ จะต้องมีการเก็บความสัมพันธ์ของแต่ละ ID กับแต่ละผู้ใช้ (client) ซึ่งสามารถใช้แท็ก <get> ทำหน้าที่รับและจำค่าตัวแปรให้กับระบบได้ โดยแท็กนี้มักจะใช้งานคู่กับแท็ก <set>

10) Set

แท็ก <set> ทำหน้าที่กำหนดหรือเปลี่ยนแปลงค่าให้กับตัวแปรที่อยู่ภายในแท็ก <get> อีกทีหนึ่ง เช่น <set name="name">Matthew</set> จะเป็นการเปลี่ยนแปลงค่าในตัวแปร name ให้เป็น Matthew ซึ่งหากมีการเรียกใช้งานแท็ก <get>หลังจากนี้ด้วยรูปแบบคำสั่งว่า <get name="name"/> จะส่งผลให้มีการคืนค่า (return) เป็นคำว่า Matthew

นอกจากนี้ยังสามารถใช้ช่วยจำสรรพนามได้ด้วย ดังในตัวอย่างข้างล่างนี้ซึ่งได้ผลลัพธ์ว่า หากบุคคลที่ชื่อ "Samuel Clemens" กล่าวถึงสรรพนามว่า he แล้วจะหมายถึงว่า he คือ "Mark Twain"

<template>
<set name="he">Samuel Clemens</set> is Mark Twain.
</template>

การคืนค่าของแท็ก <set> นั้นจะคืนค่าที่อยู่ระหว่างแท็ก หรือชื่อของเพรดิเคต (predicate) ออกมา ดังเช่น <set name="it">Opera</set> จะคืนค่า "it" แต่ <set name="likes">Opera</set> จะคืนค่าออกมาเป็น "Opera"

ทั้งนี้หากเพรดิเคตไม่ได้ถูกแปลี่ยนแปลงค่าโดยแท็ก <set> จะมีค่าเริ่มต้น ดังนี้

<get name="she"/> จะคืนค่า (return) เป็น "Unknown"

<get name="has"/> จะคืนค่า (return) เป็น "a mother" (เนื่องจากทุกคนต้องมีแม่)

<get name="wants"/> จะคืนค่า (return) เป็น "to chat"