บทที่ 2
การวิเคราะห์คำและการวิเคราะห์ไวยากรณ์

ในบทนี้จะกล่าวถึงรายละเอียดของหลักการที่ใช้แก้ปัญหาในงานวิจัยได้แก่ การวิเคราะห์คำและการวิเคราะห์ไวยากรณ์

2.1 การวิเคราะห์คำ

2.1.1 การตัดคำ (Word Segmentation)

การตัดคำ คือการแบ่งสายอักขระ (String) ของตัวอักษรเพื่อหาขอบเขตของแต่ละหน่วยคำ ซึ่งการตัดคำมีด้วยกันหลายวิธีการดังนี้

2.1.1.1 วิธีการตัดคำแบบยาวที่สุด (Longest Matching)

เมื่อเราต้องการโปรแกรมให้คอมพิวเตอร์รู้จักคำในภาษาไทย เราก็จะค้นหาคำ โดยเริ่มจากตัวอักษรซ้ายสุดของข้อความนั้นไปยังตัวอักษรถัดไป จนกว่าจะพบว่าเป็นคำที่มีอยู่ในพจนานุกรมหลังจากนั้น ก็ค้นหาคำถัดไปจนกว่าจะจบข้อความในกรณีที่เราพบว่าเป็นคำในพจนานุกรมจากจุด เริ่มต้นเดียวกัน เราจะเลือกคำที่ยาวที่สุดตัวอย่างเช่นการแบ่งคำในประโยค “ฉันนั่งตากลมที่หน้าบ้าน ” จะเริ่มจากตัวอักษร “ฉ” และคำแรกที่แบ่งได้คือ “ฉัน” หลังจากนั้น ก็ค้นหาตัวอักษรถัดไปและนำมาเปรียบเทียบคำในพจนานุกรม ก็จะแบ่งคำว่า “นั่ง” เป็นคำต่อไป ตัวอักษรถัดไปคือ ต จากตัวอักษรนี้ เราจะได้คำว่า “ตา” กับคำว่า “ตาก” แนวคิดนี้ให้เลือกคำที่ยาวที่สุดที่ค้นพบจึงเลือกคำว่า “ตาก” หลังจากนั้น ก็จะค้นหาและเปรียบเทียบต่อไป ซึ่งจะได้ผลลัพธ์ออกมาคือ “ฉัน นั่ง ตาก ลม ที่ หน้า บ้าน”

2.1.1.2 วิธีการตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching)

วิธีการตัดคำแบบนี้เป็นการหาวิธีในการตัดคำที่สามารถจะเป็นไปได้ทั้งหมด เช่น เมื่อมีข้อความว่า "ไปหามเหสี" ก็จะตัดคำได้ 2 แบบ ดังตัวอย่างในตารางที่ 2.1

ตารางที่ 2.1 ผลลัพธ์การตัดคำแบสอดคล้องมากที่สุด

ข้อความที่ถูกต้อง	*ไป หา มเหสี*
แบบที่ 1	ไป หาม เห สี
แบบที่ 2	ไป หา มเหสี

วิธีการนี้จะให้เลือกข้อความที่แบ่งแล้วมีจำนวนคำน้อยที่สุดคือ แบบที่ 2 ถ้าเกิดกรณีที่มีจำนวนคำที่จะใช้วิธีการตัดคำแบบยาวที่สุด (Longest Matching) มาร่วมในการพิจารณา เช่น

ตารางที่ 2.2 ผลลัพธ์การตัดคำแบบสอดคล้องมากที่สุดโดยผลลัพธ์มีจำนวนคำเท่ากัน

ข้อความที่ถูกต้อง	*ฉัน นั่ง ตาก ลม ที่ หน้า บ้าน*
แบบที่ 1	ฉัน นั่ง ตาก ลม ที่ หน้า บ้าน
แบบที่ 2	ฉัน นั่ง ตา กลม ที่ หน้า บ้าน

ทั้ง 2 แบบมีจำนวนคำที่เท่ากัน จึงเลือกแบบที่ 1 โดยเปรียบเทียบจากคำที่ต่างกันคือ (ตา / ตาก) จะเห็นได้ว่า ตากมีตัวอักษรมากกว่า

2.1.1.3 วิธีการตัดคำแบบคำนวณเชิงสถิติเพื่อหาความเป็นไปได้

วิธีการนี้นำเอาค่าสถิติการเกิดของคำ และลำดับของหน้าที่ของคำ (Part of Sentence) เข้ามาช่วยในการคำนวณหาความน่าจะเป็น เพื่อที่จะใช้เลือกแบบที่มีโอกาสการเกิดมากที่สุด วิธีการนี้สามารถจะตัดคำได้ดีกว่า 2 แบบแรก แต่ข้อจำกัดของวิธีการนี้คือ จะต้องมีฐานข้อมูลที่มีการตัดคำที่ถูกต้อง และกำหนดหน้าที่ของคำให้ เพื่อที่จะได้นำไปใช้ในการสร้างสถิติ

2.1.1.4 วิธีการตัดคำแบบใช้คุณลักษณะ (Feature - Based Approach)

วิธีการนี้จะพิจารณาจากบริบท(context words) และการเกิดร่วมกันของคำ หรือประเภทของคำ(Part of Speech) เข้ามาช่วยในการตัดคำตัวอย่างเช่น "ตากลม" ถ้าพบคำว่า “แป๋ว” ในบริบทก็จะสามารถตัดคำได้ว่า “ตา” “กลม” ตัวอย่างต่อมาคำว่า “มากว่า” ถ้าในบริบทที่ตามมาเป็นตัวเลขก็สามารถตัดคำได้ว่า “มา” “กว่า” การตัดคำแบบนี้จำเป็นที่จะต้องมีฐานข้อมูลเป็น

จำนวนมาก และจะต้องมีการเรียนรู้การสร้างคำในบริบท หรือการเกิดร่วมกันของคำแต่ละคำ เพื่อให้มีข้อมูลที่จะนำมาใช้ในการตัดคำ

2.1.2 ประเภทของคำศัพท์ภาษาไทย (Part of Speech)

หลังจากทำการตัดคำแล้ว ระบบต้องทำการกำหนด หน้าที่ของคำศัพท์ ซึ่งในภาษาไทยได้ระบุหน้าที่ของคำในประโยคไว้อย่างละเอียด ดังที่แสดงในตารางที่ 2.3

ตารางที่ 2.3 ประเภทของคำศัพท์ในประโยค

No.	POS	Description	Examples
1	NPRP	Proper noun	วินโดวส์ 95, โคโรน่า, โค้ก, พระอาทิตย์
2	NCNM	Cardinal number	หนึ่ง, สอง, สาม, 1, 2, 3
3	NONM	Ordinal number	ที่หนึ่ง, ที่สอง, ที่สาม, ที่1, ที่2, ที่3
4	NLBL	Label noun	1, 2, 3, 4, ก, ข, a, b
5	NCMN	Common noun	หนังสือ, อาหาร, อาคาร, คน
6	NTTL	Title noun	ดร., พลเอก
7	PPRS	Personal pronoun	คุณ, เขา, ฉัน
8	PDMN	Demonstrative pronoun	นี่, นั่น, ที่นั่น, ที่นี่
9	PNTR	Interrogative pronoun	ใคร, อะไร, อย่างไร
10	PREL	Relative pronoun	ที่, ซึ่ง, อัน, ผู้
11	VACT	Active verb	ทำงาน, ร้องเพลง, กิน
12	VSTA	Stative verb	เห็น, รู้, คือ
13	VATT	Attributive verb	อ้วน, ดี, สวย
14	XVBM	Pre-verb auxiliary, before negator "ไม่"	เกิด, เกือบ, กำลัง
15	XVAM	Pre-verb auxiliary, after negator "ไม่"	ค่อย, น่า, ได้
16	XVMM	Pre-verb, before or after negator "ไม่"	ควร, เคย, ต้อง
17	XVBB	Pre-verb auxiliary, in imperative mood	กรุณา, จง, เชิญ, อย่า, ห้าม
18	XVAE	Post-verb auxiliary	ไป, มา, ขึ้น
19	DDAN	Definite determiner, after noun without classifier in between	นี่, นั่น, โน่น, ทั้งหมด

20	DDAC	Definite determiner, allowing classifier in between	นี้, นั้น, โน้น, นู้น
21	DDBQ	Definite determiner, between noun and classifier or preceding quantitative expression	ทั้ง, อีก, เพียง
22	DDAQ	Definite determiner, following quantitative expression	พอดี, ถ้วน
23	DIAC	Indefinite determiner, following noun; allowing classifier in between	ไหน, อื่น, ต่างๆ
24	DIBQ	Indefinite determiner, between noun and classifier or preceding quantitative expression	บาง, ประมาณ, เกือบ
25	DIAQ	Indefinite determiner, following quantitative expression	กว่า, เศษ
26	DCNM	Determiner, cardinal number expression	หนึ่งคน, เสือ 2 ตัว
27	DONM	Determiner, ordinal number expression	ที่หนึ่ง, ที่สอง, ที่สุดท้าย
28	ADVN	Adverb with normal form	เก่ง, เร็ว, ช้า, สม่ำเสมอ
29	ADVI	Adverb with iterative form	เร็วๆ, เสมอๆ, ช้าๆ
30	ADVP	Adverb with prefixed form	โดยเร็ว
31	ADVS	Sentential adverb	โดยปกติ, ธรรมดา
32	CNIT	Unit classifier	ตัว, คน, เล่ม
33	CLTV	Collective classifier	คู่, กลุ่ม, ฝูง, เชิง, ทาง, ด้าน, แบบ, รุ่น
34	CMTR	Measurement classifier	กิโลกรัม, แก้ว, ชั่วโมง
35	CFQC	Frequency classifier	ครั้ง, เที่ยว
36	CVBL	Verbal classifier	ม้วน, มัด
37	JCRG	Coordinating conjunction	และ, หรือ, แต่
38	JCMP	Comparative conjunction	กว่า, เหมือนกับ, เท่ากับ
39	JSBR	Subordinating conjunction	เพราะว่า, เนื่องจาก, ที่, แม้ว่า, ถ้า
40	RPRE	Preposition	จาก, ละ, ของ, ใต้, บน
41	INT	Interjection	โอ้ย, โอ้, เออ, เฮ่, อ่อ

42	FIXN	Nominal prefix	การทำงาน, ความสนุกสนาน
43	FIXV	Adverbial prefix	อย่างเร็ว
44	EAFF	Ending for affirmative sentencev	จ๊ะ, จ้ะ, ค่ะ, ครับ, นะ, น่า, เถอะ
45	EITT	Ending for interrogative sentence	หรือ, เหรอ, ไหม, มั้ย
46	NEG	Negator	ไม่, มิได้, ไม่ได้, มิ
47	PUNC	Punctuation	(,), ", ,, ;

จากการศึกษาภาษามือไทย คำศัพท์ภาษามือมีแค่ 8000 คำ การแบ่งประเภทตามภาษาไทยนั้นละเอียดมากเกินไป ทำให้การเขียนกฎไวยากรณ์ทำได้ยาก และซับซ้อนเกินไป ดังนั้นทางผู้วิจัยจึงแบ่งประเภทของคำศัพท์ดังแสดงในตารางที่ 2.4 ซึ่งก็เพียงพอต่อการเขียนกฎไวยากรณ์ในการแปลภาษาสำหรับงานวิจัยนี้

ตารางที่ 2.4 ประเภทของคำศัพท์ที่ใช้ในงานวิจัยนี้

ประเภท	*ประเภทของคำ*
นาม	Noun
ลักษณะนาม	CL
กริยา	Verb
เวลา	Time
กริยาวิเศษณ์	Advb
คุณศัพท์	Adjt
บุพบท	PreA,PreB
สันธาน	ConA, ConB, ConC, ConD, ConE, ConF
กริยานุเคราะห์	Aux
ปฏิเสธ	Neg
คำถาม	Quet

2.1.3 การจำแนกคำศัพท์ภาษามือไทย (Word Segmentation)

จากการศึกษาเรื่องการตัดคำขั้นตอนวิธีที่เหมาะที่จะนำมาใช้ในงานวิจัยนี้คือ วิธีการตัดคำแบบยาวที่สุด ถึงแม้ว่าผลลัพธ์การตัดคำในบางครั้งจะมีการตัดคำที่ผิดพลาดอยู่บ้าง ในงานวิจัยฉบับนี้จะแก้ไขด้วยการเพิ่มฐานข้อมูลพิเศษ จุดประสงค์มีด้วยกันสองอย่างคือ

1. รวมคำศัพท์ที่ตัดคำผิด เช่น ข้าวมันไก่(ข้าวมัน|ไก่) ไข่นกกระทา(ไข่นก|กระทา)
2. รวมคำศัพท์ที่ภาษามือไทยถือว่าเป็นคำศัพท์คำเดียว เช่น มาถึง เดินไป จับชีพจร

โดยโครงสร้างการเรียกใช้งานฐานข้อมูลพิเศษ ดังรูป 2.1

รูปที่ 2.1 การเรียกใช้งานฐานข้อมูลพิเศษ

ตัวอย่างของคำศัพท์ที่บันทึกไว้ในฐานข้อมูลพิเศษแสดงในตารางที่ 2.4

ตารางที่ 2.4 ตัวอย่างคำศัพท์ที่บันทึกไว้ในฐานข้อมูลพิเศษ

ลำดับที่	คำศัพท์	คำศัพท์ที่รวมแล้ว
1	ภาษา+มือ	ภาษามือ
2	สวัสดี+ครับ	สวัสดีครับ
3	วิ่ง+แข่ง	วิ่งแข่ง
4	อาจ+จะ	อาจจะ
5	คาด+ว่า	คาดว่า

6	ตอน+เช้า	ตอนเช้า
7	ตอน+บ่าย	ตอนบ่าย
8	ตอน+กลางคืน	ตอนกลางคืน
9	ตอน+ค่ำ	ตอนค่ำ
10	ตอน+เที่ยงคืน	ตอนเที่ยงคืน
11	ข้าวมัน+ไก่	ข้าวมันไก่
12	หมี+แพน+ด้า	หมีแพนด้า
13	ตอน+เย็น	ตอนเย็น
14	วัน+พฤหัสบดี	วันพฤหัสบดี
15	ยิง+ธนู	ยิงธนู
16	ปั่น+จักรยาน	ปั่นจักรยาน
17	ไข่+เป็ด	ไข่เป็ด
18	เห็ด+ฟาง	เห็ดฟาง
19	ยิง+ธนู	ยิงธนู
20	จับ+ชีพจร	จับชีพจร
21	โรงพยาบาล+รัฐบาล	โรงพยาบาลรัฐบาล
22	ช่างเชื่อม+โลหะ	ช่างเชื่อมโลหะ
23	ช่างซ่อม+เครื่องยนต์	ช่างซ่อมเครื่องยนต์

2.2 การวิเคราะห์ไวยากรณ์ (Syntax analysis)

การวิเคราะห์ไวยากรณ์คือการแจงโครงสร้างของข้อมูลที่ได้รับแล้วทำการเทียบกับกฎที่ตั้งไว้เพื่อดำเนินการสังเคราะห์ต่อไป จากการศึกษาพบว่าการแจงโครงสร้างมีหลายรูปแบบ แต่รูปแบบที่นิยมในปัจจุบันมีดังต่อไปนี้

2.2.1 ไวยากรณ์แบบ Backus Naur Form (BNF)

เป็นไวยากรณ์ที่ใช้สำหรับวิเคราะห์โครงสร้างของประโยค BNF มักจะนำไปใช้ในส่วนของการตรวจโครงสร้างไวยากรณ์ของโปรแกรมจำพวก Syntax Analyzer หรือ Parser เนื่องจาก BNF มีข้อดีคือสามารถเข้าใจได้ง่าย วิธีการเขียนไม่ซับซ้อน แต่จุดด้อยคือ BNF ไม่มีเครื่องหมาย

พิเศษที่ช่วยในการเขียนกฎไวยากรณ์ให้สั้นลง ดังนั้นเมื่อจำนวนไวยากรณ์เพิ่มมากขึ้น BNF จะดูซับซ้อนและแก้ไขได้ยาก ตัวอย่างการเขียน BNF สามารถเขียนได้ดังรูปที่ 2.2

```
ประโยค        ::=   ประธาน กริยา
ประธาน        ::=   คำนาม
กริยา         ::=   อกรรมกริยา | สกรรมกริยา นามวลี
นามวลี        ::=  คำนาม    คำวิเศษณ์ | คำนาม
คำนาม         ::=   ข้าว | แม่ | ขนม | น้ำ | หนู
คำวิเศษณ์     ::=   สีแดง
อกรรมกริยา    ::=   นอน | เดิน
สกรรมกริยา    ::=   กิน | ดื่ม | เห็น
```

รูปที่ 2.2 ตัวอย่าง BNF สำหรับประโยคภาษาไทยอย่างง่าย

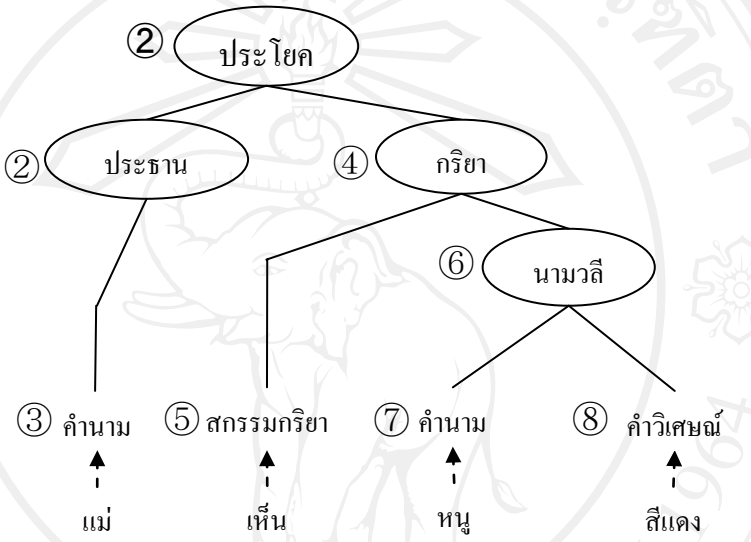
จาก BNF ตามรูปที่ 2.2 สามารถสร้างประโยคตัวอย่างที่ถูกต้องตามไวยากรณ์ได้ดังตารางที่ 2.5

ตารางที่ 2.5 ตัวอย่างประโยคที่ถูกต้องตามไวยากรณ์ BNF

ลำดับที่	ประโยค
1	ข้าวนอน
2	ขนมเดิน
3	ขนมกินน้ำ
4	แม่กินขนม
5	แม่กินข้าว
6	หนูกินข้าวสีแดง
7	แม่เห็นหนูสีแดง

2.2.2 การแจงโครงสร้างข้อมูลแบบลำดับชั้นบนลงล่าง (Top-Down)

เป็นการแจงโครงสร้างข้อมูลแบบลำดับชั้นโดยเริ่มจา กราก (Root) และมีการท่องไปในโครงสร้างต้นไม้แบบพรีออเดอร์ (Preorder เป็นวิธีการท่องไปในต้นไม้แบบ Root → Left → Right; NLR) จากตัวอย่างประโยคตามไวยากรณ์ BNF ในตารางที่ 2.5 สามารถนำประโยคมาแจงโครงสร้างข้อมูลแบบลำดับชั้นบนลงล่างได้ตามกฎไวยากรณ์ BNF ดังรูปที่ 2.3



รูปที่ 2.3 การแจงประโยครูปแบบชั้นบนลงล่าง

จากรูปที่ 2.3 เป็นการท่องไปในโครงสร้างต้นไม้ โดยมี

N : root node เป็นโหลดเริ่มต้นของโครงสร้าง

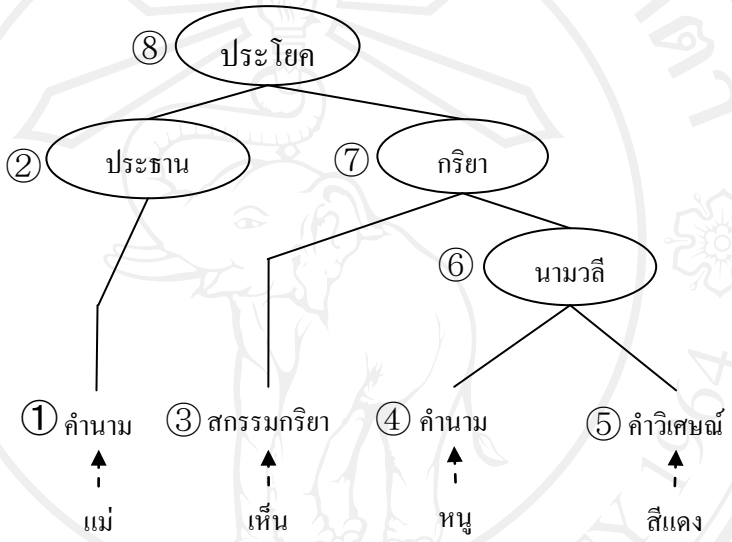
L : Sub node ไปทางซ้าย

R : Sub node ไปทางขวา

ดังนั้นการท่องไปในโครงสร้างต้นไม้แบบพรีออเดอร์จะเป็นการท่องในลักษณะ NLR โดยจะมีการเรียงลำดับของโหนด ดังนี้ ประโยค → ประธาน → คำนาม → กริยา → สกรรมกริยา → นามวลี → คำนาม → คำวิเศษณ์

2.2.3 การแจงโครงสร้างข้อมูลแบบลำดับชั้นล่างขึ้นบน (Bottom-Up)

เป็นการแจงโครงสร้างข้อมูลแบบลำดับชั้นโดยเริ่มต้นจาก โหนดใบ(Leaves Node) และมีการท่องไปในโครงสร้างต้นไม้แบบโพสออร์เดอร์ (Post order เป็นวิธีการท่องไปใน โครงสร้างต้นไม้แบบ Left→Right → Root; LRN) จากตัวอย่างประโยคตามไวยากรณ์ BNF ในตารางที่ 2.5 สามารถนำประโยคมาแจงโครงสร้างข้อมูลแบบลำดับชั้นบนลงล่างได้ตามกฎไวยากรณ์ BNF ดังรูปที่ 2.4



รูปที่ 2.4 การแจงประโยครูปแบบ Bottom-up

จากรูปที่ 2.4 เป็นการท่องไปในโครงสร้างต้นไม้ โดยมี

L : Sub node ไปทางซ้าย

R : Sub node ไปทางขวา

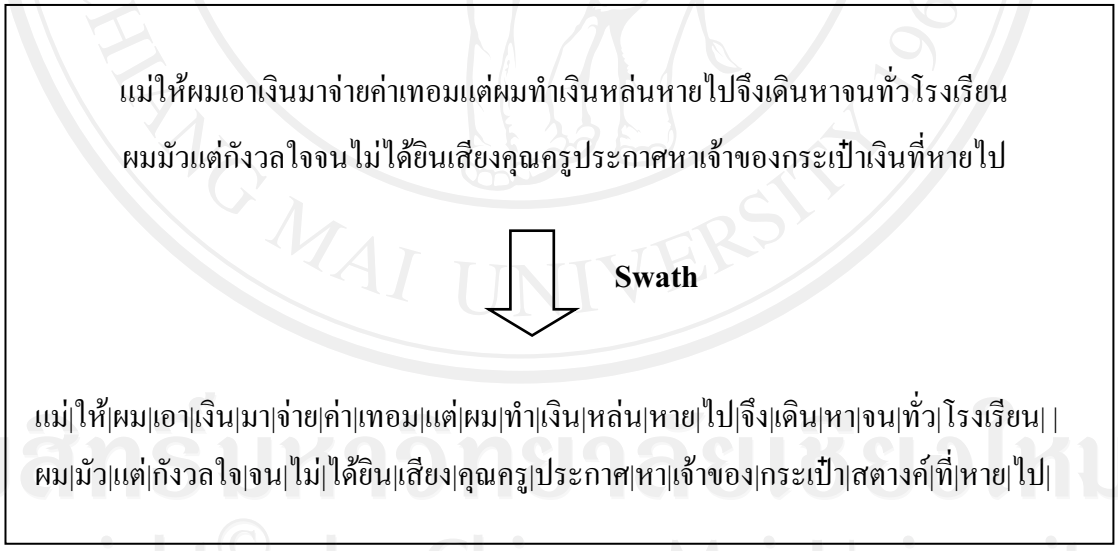
N : root node เป็นโหลดเริ่มต้นของโครงสร้าง

ดังนั้นการท่องไปในโครงสร้างต้นไม้แบบโพสออร์เดอร์จะเป็นการท่องในลักษณะ NLR โดยจะมีการเรียงลำดับของโหนด ดังนี้ คำนาม → ประธาน → สกรรมกริยา → คำนาม → คำวิเศษณ์ → นามวลี → กริยา → ประโยค

2.3 เครื่องมือช่วยวิเคราะห์คำและวิเคราะห์ไวยากรณ์

2.3.1 เครื่องมือช่วยวิเคราะห์คำ

จากที่ได้อธิบายมาในตอนต้นเกี่ยวกับการวิเคราะห์คำ ขั้นแรกของการวิเคราะห์คำจำเป็นต้องตัดคำให้ได้เสียก่อน ซึ่งการจะวิเคราะห์ภาษาไทยก็ต้องตัดคำภาษาไทย แต่ภาษาไทยเป็นภาษาที่ตัดคำได้ยาก เพราะภาษาไทยไม่มีการบอกขอบเขตของคำ มีงานวิจัยเกี่ยวกับการตัดภาษาไทยมากมายได้พัฒนาซอฟต์แวร์ที่ช่วยในการตัดคำภาษาไทย ซึ่งซอฟต์แวร์ที่มีผู้นิยมใช้มากและยังมีการพัฒนาต่อเนื่องอยู่ก็คือ Swath [10] เป็นซอฟต์แวร์สำหรับตัดคำภาษาไทย ที่ สามารถเลือกวิธีการตัดคำได้สองวิธี คือ การตัดคำแบบยาวที่สุด (longest matching) และการตัดคำโดยเลือกแบบสอดคล้องมากที่สุด (maximal matching algorithms) ซึ่งนอกเหนือจากการใช้งานได้ดีกับข้อความที่เป็น text ธรรมดา ซอฟต์แวร์ยังสามารถรองรับไฟล์ในรูปแบบต่างๆ ได้แก่ html, rtf ทางผู้วิจัยจึงได้นำ Swath มาทดลองใช้และได้ผลลัพธ์จากการตัดคำดังตัวอย่างในรูปที่ 2.5



รูปที่ 2.5 การตัดคำด้วย Swath

จากผลลัพธ์ใน การตัดคำในรูปที่ 2.5 จะเห็นได้ ว่าแม้ผู้วิจัยจะเลือกการตัดคำแบบยาวที่สุด แต่ก็ยังมีหลายคำที่ตัดได้ผิดพลาด เช่น ค่าเทอม มัวแต่ กระเป๋าสตางค์ เป็นต้น สำหรับในปัจจุบันยังไม่มีซอฟต์แวร์ตัดคำใดที่มีสัญญาอนุญาตแบบ GNU General Public License (GNU-

GPL) ที่อนุญาตให้บุคคลใดๆทำซ้ำ เผยแพร่ และ/หรือดัดแปลงซอฟต์แวร์นั้นได้อย่างถูกต้องตามกฎหมาย และโดยเสรี ดังนั้นผู้วิจัยจึงต้องหาวิธีแก้ไขการตัดคำที่ผิดพลาดซึ่งได้เลือกที่จะใช้วิธีตามหัวข้อ 2.1.3 ที่ได้กล่าวมาแล้ว

2.3.2 เครื่องมือช่วยวิเคราะห์ไวยากรณ์

เครื่องมือที่ช่วยวิเคราะห์ไวยากรณ์ในปัจจุบันมีให้เลือกใช้มากมาย และสามารถเลือกใช้ได้ตามระบบปฏิบัติการณ์คอมพิวเตอร์ หรือเลือกใช้ตามภาษาคอมพิวเตอร์ที่จะพัฒนาโปรแกรมก็ได้ โดยงานวิจัยฉบับนี้ ผู้วิจัยเลือกใช้ Boost Spirit [11] ซึ่งเป็นเครื่องมือช่วยวิเคราะห์ไวยากรณ์ที่สามารถพัฒนาได้บนระบบปฏิบัติการณ์วินโดว์และสามารถใช้ภาษาคอมพิวเตอร์ C++ ในการเขียนโปรแกรมได้

2.3.2.1 การวิเคราะห์โครงสร้างทางไวยากรณ์ด้วย Boost Spirit

Boost Spirit เป็นกลุ่มคำสั่งที่มีลักษณะการทำงานแบบ Parser generator ชนิดหนึ่ง โดยมีแนวคิดแบบ Object-oriented ที่จะเปลี่ยนการอธิบาย ไวยากรณ์ในรูปแบบ Extended Backus Naur Form (EBNF) ไปเป็น LL parser โดยเปลี่ยนให้อยู่ในรูปแบบของภาษา C++ ซึ่งทำให้เราสามารถวิเคราะห์โครงสร้างของประโยคได้

แม้ว่าในปัจจุบันจะมีกลุ่มคำสั่งที่มีหน้าที่ลักษณะการทำงานเหมือนกันกับ Boost Spiritอยู่มากมายหลายตัวด้วยกัน หนึ่งในนั้นคือ Bison [12] ซึ่งเป็น Parser Generator ชนิดหนึ่งที่จะทำการเปลี่ยนการอธิบาย ไวยากรณ์ ไปเป็น Look Ahead Left-to-right Rightmost (LALR) ให้อยู่ในภาษา C ซึ่งทำให้ Bison สามารถวิเคราะห์ โครงสร้าง ประโยคแบบ ลำดับชั้นล่าง ขึ้นบนได้ นอกจากนี้ ยังมีผู้นิยมใช้ Bison เป็นจำนวนมากในการวิเคราะห์โครงสร้างเนื่องจาก สามารถสร้างการวิเคราะห์แบบ Generalized Left-to-right Rightmost (GLR) สำหรับไวยากรณ์ที่มีความกำกวมได้อีกด้วย

แต่สาเหตุที่งานวิจัยฉบับนี้เลือกใช้ Boost Spirit เนื่องจาก

1. งานวิจัยฉบับนี้พัฒนาระบบบน Visual Studio Express ทำให้ Boost Spirit มีการเรียกใช้ที่ง่ายกว่า รวมทั้งสนับสนุนการพัฒนางานบนฝั่ง Windows
2. การเขียน EBNF มีความง่ายกว่าการเขียน Backus Naur Form (BNF) แบบธรรมดามาก เนื่องจากมีตัวดำเนินการ (Operators) ที่ง่ายต่อการใช้งานมากดังที่ปรากฏในตารางที่ 2.6

ตารางที่ 2.6 ตัวดำเนินการของ Boost Spirit

ชุดคำสั่ง	*คำอธิบาย*
Set Operators	
a | b	a หรือ b ก็ได้
a & b	a และ b
a - b	เจอ a แต่ไม่ใช่ b
a ^ b	เจอ a หรือ b แต่ต้องไม่เจอทั้งสอง
Sequencing Operators	
a >> b	เรียงลำดับจาก a ต่อด้วย b
a && b	มี a และต่อด้วย b ในลำดับ
a || b	มี a >> b หรือ a หรือ b ในลำดับ
Optional and Loops	
*a	มี a ต่อกันตั้งแต่ 0 ตัวขึ้นไป
+a	มี a ต่อกันตั้งแต่ 1 ตัวขึ้นไป
!a	มี a 0 หรือ 1 ตัวเท่านั้น
a % b	ความหมายเดียวกันกับ a >> *(b >> a)

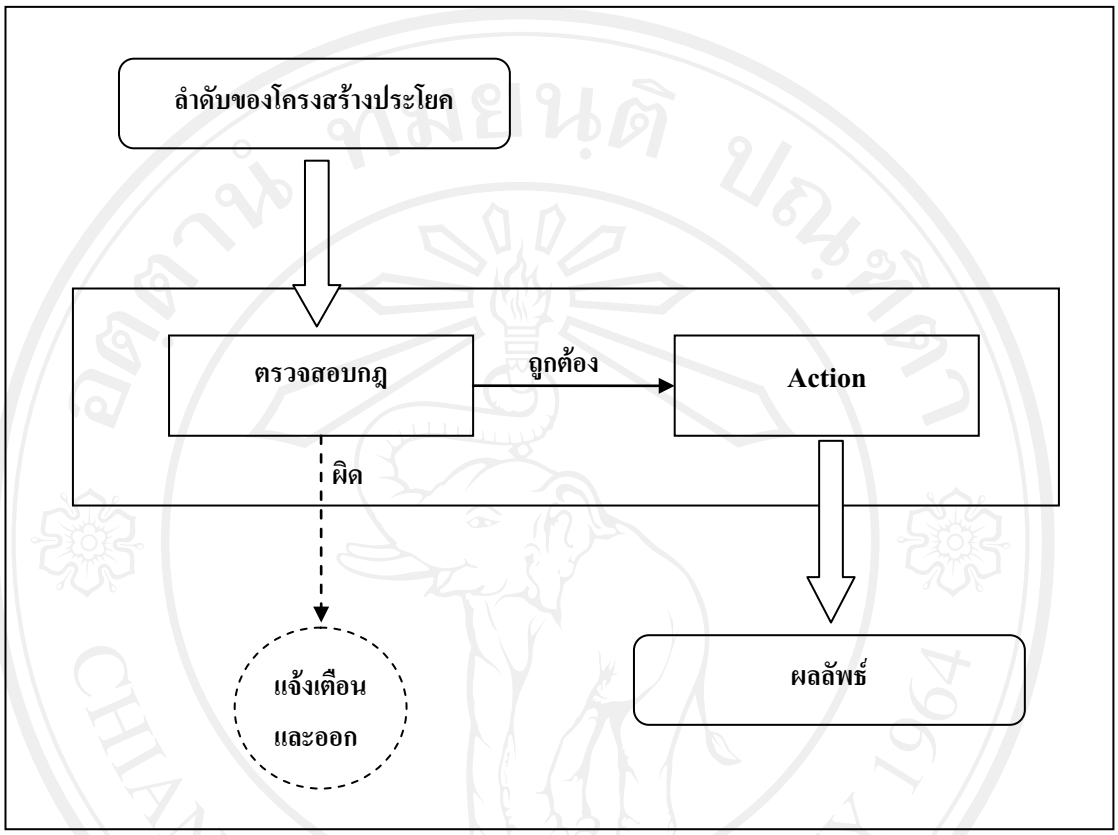
จากตารางที่ 2.6 เมื่อนำตัวดำเนินการมาเขียน EBNF เปรียบเทียบกับการเขียน BNF ธรรมดาจะได้ผลความแตกต่างดังตารางที่ 2.7

ตาราง 2.7 ตัวอย่างของการใช้ตัวดำเนินการใน Boost Spirit

EBNF กับ ตัวดำเนินการ	*BNF*
A >> +(!B >> C)	A >> B >> C >> B >> C >> … A >> C >> C >> C >> …
A >> (B || C || D)	A >> B >> C >> D A >> B >> C A >> C >> D A >> B >> D A >> B A >> C A >> D

(!A >> B >> C)%D	A >> B >> C >> D >> A >> B >> C >> D >>… B >> C >> D >> B >> C >> D >>…
+A >> !(B%C)	A >> B >> C >> B >> C >> … A >> A >> B >> C >> B >> C >> … A >> B >> C >> B A >> A >> A >> … A
A >> !B >> !C >> +D	A >> B >> C >> D >> D >> D >>… A >> B >> C >> D A >> C >> D >> D >> D >> … A >> C >> D A >> B >> D >> D >> D >> … A >> B >> D A >> D >> D >> D >> … A >> D
A >> !(A|B) >> +C	A >> A >> C >> C >> C >> … A >> A >> C >> C >> C >> … A >> C >> C >> C >> … …

2.3.2.2 โครงสร้างพื้นฐานการทำงานของ Boost Spirit



รูปที่ 2.6 โครงสร้างโดยรวมของBoost Spirit

จากรูปที่ 2.6 สามารถอธิบายการทำงานของ Boost Spirit ได้ว่า เมื่อมีการรับลำดับของโครงสร้างประโยคเข้ามาเพื่อทำการตรวจสอบโครงสร้างถ้าโครงสร้างที่รับเข้ามาตรงกับกฎที่ระบุไว้อยู่แล้ว ระบบจะยอมให้ผ่านเพื่อไปทำงานต่อในขั้นตอนต่อไปของการทำงาน ถ้าผิดกฎก็จะแจ้งเตือนและออกจากโปรแกรม