ระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์

Thai Syntax Analyser System by Computer

นายสมศักดิ์ จันวัน

Mr. SOMSAK JUNWUN

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ ดร.รัตติกร วรากูลศิริพันธุ์

ADVISOR

ASSISTANT PROFESSOR DR.RUTTIKORN VARAKULSIRIPUNTH

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์มหาบัณฑิต

สาขาวิศวกรรมไฟฟ้า

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2534

หัวข้อวิทยานิพนธ์	ระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์
ผู้วิจัย	นายสมศักดิ์ จันวัน
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผู้ช่วยศาสตราจารย์ ดร.รัตติกร วรากูลศิริพันธุ์
ระดับการศึกษา	วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า
ปีการศึกษา	2534

บทคัดย่อ

ในภาษาไทยนั้น หน่วยคำของภาษาที่ประกอบกันเป็นประโยค มีการเขียนต่อเนื่องกันโดยไม่มีช่องว่าง หรือเครื่องหมายบอกการสิ้นสุดของหน่วยคำในประโยค ทำให้การวิเคราะห์โครงสร้างประโยคภาษาไทย หรือการประมวลผลภาษาไทยโดยระบบคอมพิวเตอร์เป็นไปด้วยความยากลำบาก เพราะไม่รู้จุดสิ้นสุดของหน่วยคำที่แน่นอน โดยปกติแล้วการที่จะใช้ระบบคอมพิวเตอร์ในการวิเคราะห์หน่วยคำหรือประมวลผลภาษาไทยได้นั้น มนุษย์ต้องเป็นผู้ทำการแยกหน่วยคำเสียก่อน หลังจากนั้นจึงป้อนให้ระบบคอมพิวเตอร์วิเคราะห์หรือประมวลผล ต่อมาได้มีนักวิจัยคิดค้นวิธีการแยกหน่วยคำภาษาไทยออกจากประโยคด้วยระบบซอฟท์แวร์คอมพิวเตอร์ เช่น ระบบการแยกหน่วยคำ โดยการประมวลผลจากการผสมพยัญชนะกับสระหรือระบบการเลือกหน่วยคำที่ยาวที่สุด เป็นต้น แต่ระบบเหล่านี้ ยังมีประสิทธิภาพไม่ดีนัก ให้ผลลัพธ์ที่ล่าช้าและมีความถูกต้องน้อย

ดังนั้น วิทยานิพนธ์ฉบับนี้จึงขอเสนอผลงานวิจัยด้านระบบการแยกแยะหน่วยคำจากประโยคในภาษาไทย เรียกว่าระบบ "Fast Word Matching" ซึ่งจากผลการวิจัยพบว่าเป็นระบบการแยกแยะหน่วยคำที่สมบูรณ์ทั้งในด้านความถูกต้องและความเร็วในการทำงาน เพราะได้สร้างฐานข้อมูลที่เป็นพจนานุกรมคำศัพท์ภาษาไทย เป็นข้ออ้างอิงในการเปรียบเทียบแยกหน่วยคำโดยที่คำศัพท์ทั้งหมดภายในพจนานุกรมไทย ได้ถูกจัดแบ่งเป็นหมวดหมู่ เพื่อให้การค้นหาหน่วยคำของอัลกอริทึมของระบบได้ผลรวดเร็วและถูกต้องยิ่งขึ้น นอกจากนั้นงานวิจัยในวิทยานิพนธ์นี้ ยังได้เสนอระบบการวิเคราะห์โครงสร้างประโยคภาษาไทย เพื่อนำผลของการแยกแยะหน่วยคำจากระบบ Fast Word Matching มาประมวลผลหาความสัมพันธ์ของหน่วยคำที่ถูกต้องตามหลักไวยากรณ์ภาษาไทย เนื่องจากผลของการแยกแยะหน่วยคำของระบบ Fast Word Matching อาจได้หลายแบบ ดังนั้นระบบการวิเคราะห์โครงสร้าง

ประโยคภาษาไทย จะตัดสินการแยกหน่วยคำที่ถูกต้องได้โดยใช้ฐานความรู้ทางด้านไวยากรณ์ภาษาไทยเป็นหลักในการเลือกความสัมพันธ์ทางด้านโครงสร้างที่ถูกต้องของหน่วยคำ ฐานความรู้ที่สร้างขึ้นนี้ได้มีการจัดแบ่งชนิดของหน่วยคำแบบใหม่ โดยยึดหลักการจัดเรียงตำแหน่งของหน่วยคำและการผสมหน่วยคำในประโยคภาษาไทย แล้วสร้างเป็นความสัมพันธ์ของหน่วยคำในรูปของ **"โครงข่ายM-ATN"**

ทั้งสองระบบที่กล่าวมาได้แก่ ระบบการแยกแยะหน่วยคำ Fast Word Matching และระบบการประมวลผลโครงสร้างประโยคด้วยโครงข่าย M-ATN จะรวมกันเป็น **"ระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์"** ของงานวิจัยนี้

Thesis Title	Thai Syntax Analyser System by Computer
Author	Mr. Somsak Junwun
Thesis Advisor	Assistant Professor Dr. Ruttikorn Varakulsiripunth
Graduate Level	Master of Engineering in Electrical Engineering
Academic Year	1991

ABSTRACT

In Thai language, words in a sentence are written continuously without spaces between words or without any marks at the end of each word. This will cause the syntactical analysis and processing on Thai language with a computer system to be very complicated. Because in general, we have to separate each word in a sentence by hand before inputting them into a computer system for the furture processing. However, there are some researches on automatic segmentation of Thai words by using computer software techniques such as segmentation by considering on the combination of consonants and vowels, or by longest matching algorithm, etc. But these techniques have low performance since it takes to much time to obtain a correct result.

In this thesis, we have presented our research on a new approach of Thai word's segmentation system called "Fast Word Matching". As described in this thesis that our system shew very perfect and good result in both correctness and processing time. This is because that we have constructed a dictionary of Thai vocabulary and use it as a reference database for word segmentation. Moreover, all the words in this dictionary are divided in groups in accordance with the searching algorithm of the system. We have

also presented a new approach of Thai syntactical analysis system. This syntactical analysis system will make a decision to select a correct result of Thai word's segmentation outputs those obtained from Fast Word Matching system. The segmented sentences will be analized based on the knowledge base of Thai syntax. This will result in selection of correct grammatical relation of Thai words in a sentence. The knowledge base of Thai syntax in our system is constructed based on the grammatical ordering structure of Thai words in a sentence. And this grammatical knowledge base is introduced in form of "Modified-Augmented Transition Network (M-ATN)"

Both systems, i.e., Fast Word Matching and M-ATN, are the main parts of our research on "Thai Syntax Analyser System by Computer".

สารบัญ

สารบัญรูปประกอบ

หน้า

สารบัญตารางประกอบ

หน้า

จุดประสงค์ และโครงสร้างของวิทยานิพนธ์

ภาษาเป็นพื้นฐานในการสื่อความหมายกันของมนุษย์ ซึ่งเกิดจากการสร้างสมประสบการณ์ ความรู้ ความเข้าใจในภาษานั้นๆ เพื่อสร้างกระบวนการในการทำความเข้าใจและการใช้ภาษานั้นๆ ซึ่งประกอบด้วย ความรู้ในตัวภาษา เช่นการพูด (Linguistic Knowledge) เมื่อมนุษย์พูด ผู้พูดเองจะต้องใช้กฎเกณฑ์ที่เกี่ยวกับเสียง (Phonological Rules) เพื่อเปลี่ยนประโยคที่คิดไว้เป็นรูปแบบของเสียง หรือเมื่อเขียน ผู้เขียนจะต้องเปลี่ยนประโยคที่คิดไว้ให้อยู่ในรูปแบบที่สามารถมองเห็น และเข้าใจได้สำหรับผู้สื่อสารด้วย

รูปแบบของการพูด (Sound Pattern) และรูปแบบของการเขียน (Written Pattern) เราเรียกว่า โครงสร้างทางไวยากรณ์ (Syntactic Structure) โดยการใช้คำศัพท์ (Vocabulary) และความรู้เกี่ยวกับการประกอบคำศัพท์ให้ถูกต้องตามไวยากรณ์ ผู้ฟังและผู้อ่านสามารถที่จะจับใจความได้ ดังนั้นจึงจำเป็นมากที่จะต้องเข้าใจในภาษาธรรมชาติ (Natural Language) ซึ่งมนุษย์เก็บความเข้าใจเหล่านี้ไว้ในสมอง ในทำนองเดียวกัน เครื่องคอมพิวเตอร์ก็ทำงานคล้ายกับมนุษย์ในแง่ของความสามารถที่จะประมวลผลเกี่ยวกับสัญลักษณ์ และทำการตัดสินใจด้วยฐานความรู้ (Knowledge Base) ที่มีอยู่ในหน่วยความจำ แต่เครื่องคอมพิวเตอร์ก็ยังมีข้อแตกต่างกับมนุษย์ คือ เครื่องคอมพิวเตอร์ทำงานตามโปรแกรมที่สร้างขึ้น ส่วนมนุษย์นั้นทำตามความรู้ที่เกิดจากประสบการณ์และกระบวนความคิดจากสมอง ทำให้นักวิจัยพยายามสร้างเครื่องคอมพิวเตอร์ให้มีความสามารถเช่นเดียวกับมนุษย์ โดยการพยายามจำลองกระบวนการในสมอง (Mental Process) มาใช้กับระบบคอมพิวเตอร์ อันเป็นศาสตร์หนึ่งทางด้านปัญญาประดิษฐ์ (Artificial Intelligence) และจากการค้นคว้าของนักวิจัยทำให้ทราบว่าต้องมีกระบวนการอย่างน้อย 2 กระบวนการที่จะสามารถทำให้เกิดฐานความรู้ในการที่จะทำให้คอมพิวเตอร์เรียนรู้ภาษามนุษย์ได้แก่ **กระบวนการทางโครงสร้าง (Syntactic Analysis)** กับ กระบวนการทางความหมาย (Semantic Analysis)

กระบวนการทางโครงสร้าง คือ กฎเกณฑ์ หรือกระบวนการที่รวมเอาหน่วยคำ (Word) ต่างๆ มาประกอบกันขึ้นเป็นกลุ่มคำที่มีความหมายในภาษาหนึ่งๆ กลุ่มของกฎเกณฑ์ที่ใช้อธิบายโครงสร้างของกลุ่มหน่วยคำในภาษาหนึ่งๆนี้เราเรียกว่า ไวยากรณ์ (Grammar) ซึ่งทฤษฎีทางโครงสร้างนี้ได้มีผู้

สร้างและพัฒนาเรื่อยมา โดยใช้หน้าที่ของหน่วยคำเมื่อปรากฏในประโยค (Part of Speech) และความรู้ด้านอื่นๆในแต่ละภาษาสร้างเป็นทฤษฎี หรือกฎเกณฑ์ขึ้น อาทิเช่น Pattern Matching, Transition Network, Context-Free Grammar, Transformational Grammar และ Augmented Transition Network Grammar เป็นต้น

กระบวนการทางความหมาย เป็นกระบวนการให้ความหมายแก่ประโยคที่ผ่านการวิเคราะห์ทางโครงสร้างแล้ว

ภาษาต่างๆในโลกล้วนมีรูปแบบของตัวเอง ไม่เหมือนกันในแต่ละภาษา ภาษาไทยก็เช่นเดียวกัน ถ้าจะทำให้เครื่องคอมพิวเตอร์เข้าใจความหมายต่างๆของภาษาไทย เราต้องสร้างโปรแกรมออกมาเป็น กฎเกณฑ์ หรือสอนกฎเกณฑ์ทางไวยากรณ์ให้เครื่องคอมพิวเตอร์ แต่ประโยคภาษาไทยนั้นมีปัญหาทางด้านการประมวลผลด้วยคอมพิวเตอร์อยู่ว่า หน่วยคำของภาษาไทยที่ประกอบกันเป็นประโยคนั้น เชียนติดกันไม่มีช่องว่าง หรือเครื่องหมายบอกการสิ้นสุดหน่วยคำในประโยค ทำให้การวิเคราะห์เป็นไปด้วยความยากลำบาก ซึ่งเมื่อก่อนการที่จะทำให้คอมพิวเตอร์วิเคราะห์หน่วยคำในประโยคภาษาไทยได้นั้น มนุษย์ต้องเป็นผู้มาแยกหน่วยคำเอง แล้วจึงป้อนให้คอมพิวเตอร์ทำการวิเคราะห์ทางโครงสร้าง ต่อมาได้มีนักวิจัยคิดค้นวิธีการแยกหน่วยคำในประโยคภาษาไทยไว้หลายวิธี เช่น กระบวนการแยกหน่วยคำออกจากประโยคโดยการประมวลผลจากการผสมพยัญชนะกับสระ[1] และ กระบวนการที่ใช้วิธีเลือกหน่วยคำที่ยาวที่สุด[2] (Longest Mapping) เป็นต้น แต่ระบบเหล่านี้ยังมีประสิทธิภาพไม่ดีนัก ให้ผลลัพธ์ที่ล่าช้าและมีความถูกต้องน้อย

ดังนั้น วิทยานิพนธ์ฉบับนี้จึงขอเสนอผลงานวิจัยด้านระบบการแยกแยะหน่วยคำจากประโยคในภาษาไทย เรียกว่าระบบ **"Fast Word Matching"** ซึ่งจากผลการวิจัยพบว่าเป็นระบบการแยกแยะหน่วยคำที่สมบูรณ์กว่าระบบที่กล่าวมาแล้วข้างต้น ทั้งในด้านความถูกต้องและความเร็วในการทำงาน เพราะได้สร้างฐานข้อมูลที่เป็นพจนานุกรมคำศัพท์ภาษาไทย เป็นข้ออ้างอิงในการเปรียบเทียบแยกหน่วยคำโดยที่คำศัพท์ทั้งหมดภายในพจนานุกรมไทย ได้ถูกจัดแบ่งเป็นหมวดหมู่ เพื่อให้การค้นหาหน่วยคำของอัลกอริทึมของระบบได้ผลรวดเร็วและถูกต้องยิ่งขึ้น นอกจากนั้นงานวิจัยในวิทยานิพนธ์นี้ ยังได้เสนอระบบการประมวลผลโครงสร้างประโยคภาษาไทย เพื่อนำผลของการแยกแยะหน่วยคำจากระบบ Fast Word Matching มาประมวลผลหาความสัมพันธ์ของหน่วยคำที่ถูกต้องตามหลักไวยากรณ์ภาษาไทย เนื่องจากผลของการแยกแยะหน่วยคำของระบบ Fast Word Matching อาจได้หลายแบบ ดังนั้นระบบการวิเคราะห์โครงสร้างประโยคภาษาไทย จะตัดสินการแยกหน่วยคำที่ถูกต้องได้โดยใช้ฐานความรู้ทางด้าน

ไวยากรณ์ภาษาไทยเป็นหลักในการเลือกความสัมพันธ์ทางด้านโครงสร้างที่ถูกต้องของหน่วยคำ ฐานความรู้ที่สร้างขึ้นนี้ได้มีการจัดแบ่งชนิดของหน่วยคำแบบใหม่ โดยยึดหลักการจัดเรียงตำแหน่งของหน่วยคำและการผสมหน่วยคำในประโยคภาษาไทย แล้วสร้างเป็นความสัมพันธ์ของหน่วยคำในรูปของ **"โครงข่าย M-ATN"**

ทั้งสองระบบที่กล่าวมาได้แก่ ระบบการแยกแยะหน่วยคำ Fast Word Matching และระบบการประมวลผลโครงสร้างประโยคด้วยโครงข่าย M-ATN จะรวมกันเป็น **"ระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์"** ของงานวิจัยนี้ ดังแสดงในรูปที่ 1.1 และลำดับขั้นตอน เนื้อหาของวิทยานิพนธ์ในบทต่อๆไปเป็นดังนี้

บทที่ 2 กล่าวถึงการแยกหน่วยคำออกจากประโยคด้วยวิธี Fast Word Matching
- โครงสร้างของพจนานุกรมไทยในระบบ Fast Word Matching
- กระบวนการแยกแยะหน่วยคำจากประโยคของระบบ Fast Word Matching
- ตัวอย่างการทำงานของระบบ Fast Word Matching

บทที่ 3 กล่าวถึงฐานความรู้เกี่ยวกับไวยากรณ์ภาษาไทย
- ชนิดของคำในภาษาไทย
- สัญลักษณ์ของกลุ่มคำภาษาไทย
- โครงสร้างของวลีในภาษาไทย

บทที่ 4 กล่าวถึงการประมวลผลทางโครงสร้างของประโยคภาษาไทยด้วยระบบ M - ATN
- ส่วนประกอบของโครงข่ายในระบบต่างๆ
- การทำงานของระบบประมวลผลโครงสร้างประโยค : TN, RTN และ ATN
- หลักการทำงานของระบบ M-ATN ในการประมวลผลโครงสร้างประโยคภาษาไทย
- ตัวอย่างการประมวลผลประโยคภาษาไทยด้วยระบบ M-ATN

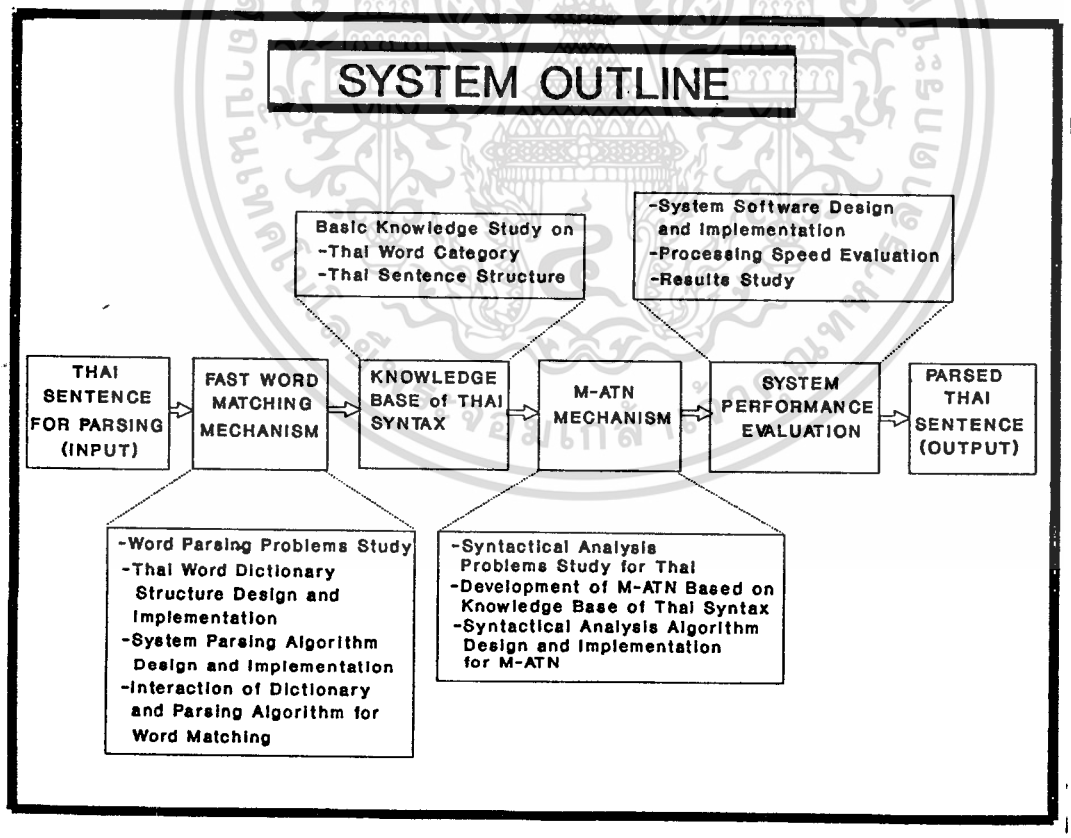
บทที่ 5 กล่าวถึงการออกแบบซอฟต์แวร์
- การออกแบบและการสร้างซอฟท์แวร์ของระบบโครงสร้างภาษาไทย
- ซอฟท์แวร์ของระบบแยกแยะหน่วยคำ Fast Word Matching
- ซอฟท์แวร์ของระบบฐานข้อมูลโครงสร้างภาษาไทย
- ซอฟท์แวร์ของระบบ M-ATN

- การใช้ซอฟท์แวร์วิเคราะห์โครงสร้างภาษาไทย

บทที่ 6 กล่าวถึงการประเมินประสิทธิภาพของระบบ

- ประสิทธิภาพของระบบต่างๆในระบบวิเคราะห์โครงสร้างภาษาไทย
- การวิเคราะห์ทางสถิติของการแยกแยะหน่วยคำด้วยระบบ Fast Word Matching
- การประยุกต์ใช้งานของระบบ

จุดประสงค์ของงานวิจัยนี้เพื่อพัฒนาระบบการแยกแยะหน่วยคำไทยในประโยค และระบบการประมวลผลโครงสร้างประโยคภาษาไทยแบบอัตโนมัติด้วยซอฟท์แวร์คอมพิวเตอร์ ซึ่งผลของงานวิจัยนี้สามารถที่จะนำไปใช้เป็นส่วนหนึ่งของระบบการแปลภาษาด้วยคอมพิวเตอร์ ระบบการตรวจสอบตัวสะกดภาษาไทยด้วยคอมพิวเตอร์ ระบบการเรียงพิมพ์ตัวอักษรด้วยคอมพิวเตอร์ ระบบการสังเคราะห์เสียงจากข้อความภาษาไทยด้วยคอมพิวเตอร์ และงานประยุกต์คอมพิวเตอร์เพื่อการประมวลผลภาษาไทยอื่นๆ อีกมากมาย



รูป 1.1 แผนผังของระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์ของงานวิจัยนี้

บทที่ 2

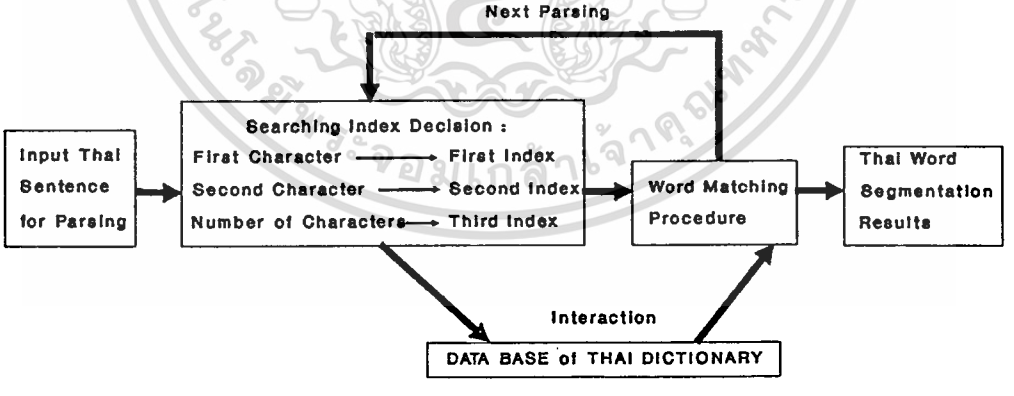
การแยกแยะหน่วยคำออกจากประโยคด้วยระบบ Fast Word Matching

2.1 บทนำ

ในปัจจุบันเทคโนโลยีทางด้านระบบไมโครคอมพิวเตอร์ ได้รับการพัฒนาให้มีประสิทธิภาพมากขึ้น ทำให้มีการใช้คอมพิวเตอร์อย่างกว้างขวางในงานที่แตกต่างกันออกไปโดยเฉพาะการพัฒนาซอฟท์แวร์ประเภทปัญญาประดิษฐ์ (Artificial Intelligence) ด้านการแปลภาษาด้วยคอมพิวเตอร์ (Machine Translation) อย่างไรก็ตาม การแปลภาษาระหว่างประเทศต่างๆด้วยระบบคอมพิวเตอร์ยังพัฒนาไปได้ไม่มากนัก ทั้งนี้เพราะภาษาของแต่ละประเทศต่างมีโครงสร้างและระบบการใช้ที่เป็นเอกลักษณ์เฉพาะตัว รวมทั้งข้อยกเว้นทางไวยากรณ์ที่เกิดจากความเคยชินในการใช้ภาษาของประชาชนในประเทศนั้นๆ หรืออิทธิพลจากประเทศที่อยู่ข้างเคียง ภาษาไทยก็เช่นกัน การแปลภาษาด้วยเครื่องคอมพิวเตอร์จากภาษาไทยไปเป็นภาษาชาติอื่นๆนั้น ยังมีอุปสรรคอยู่เกี่ยวกับลักษณะพื้นฐานของภาษาไทยเองโดยเฉพาะหน่วยคำ (Morpheme) ในภาษาไทยที่เขียนติดต่อกันโดยไม่มีเครื่องหมายหรือช่องว่างบอกการจบของหน่วยคำในประโยค ปัญหานี้ทำให้เกิดความยุ่งยากในการประมวลผลภาษาไทยด้วยคอมพิวเตอร์ จึงได้มีผู้วิจัยศึกษาและเสนออัลกอริทึม (Algorithm) สำหรับแยกหน่วยคำออกจากประโยคไว้หลายวิธี สำหรับงานวิจัยที่เสนอในวิทยานิพนธ์นี้ก็ได้คิดค้นและสร้างอัลกอริทึมใหม่ซึ่งเรียกว่า ระบบ **Fast Word Matching** สำหรับแยกหน่วยคำออกจากประโยค โดยใช้พจนานุกรมไทย (Thai Dictionary) ที่เก็บบันทึกไว้ในระบบคอมพิวเตอร์เป็นฐานข้อมูลในการเปรียบเทียบ เพื่อแยกหน่วยคำในประโยค การค้นหาคำศัพท์ (Searching) ในพจนานุกรมไทยเพื่อนำมาเปรียบเทียบนี้จะใช้ดรรชนี (Index) จำนวน 3 ค่า โดยใช้ค่าแอสกีนัมเบอร์ (ASCII Number) ของอักขระตัวที่ 1 (First Character) แอสกีนัมเบอร์ของอักขระตัวที่ 2 (Second Character) และจำนวนอักขระทั้งหมด (Total Number of Characters) ของส่วนประโยคที่จะทำการแยกหน่วยคำเป็นดรรชนีตามลำดับ ดังนั้นพจนานุกรมไทย ที่สร้างขึ้นในระบบของงานวิจัยนี้ จะมีการจัดเก็บบันทึกคำศัพท์ทั้งหมดเป็นหมวดหมู่ตามลำดับของค่าดรรชนีทั้ง 3 ตัวดังกล่าว ด้วยวิธีการนี้จะทำให้ค้นพบหน่วยคำที่สามารถแยกออก

จากประโยคได้อย่างรวดเร็วเพราะการใช้ดรรชนีทั้ง 3 ตัว นอกจากนั้นหน่วยคำที่แยกได้จะถูกต้องเพราะเราได้แยกโดยการเปรียบเทียบคำศัพท์ที่มีบรรจุอยู่ในฐานข้อมูล หรือพจนานุกรมไทยแล้ว

ในบทนี้จะกล่าวถึง โครงสร้างพจนานุกรมไทยที่จัดเก็บบันทึกเป็นฐานข้อมูลและกระบวนการหรืออัลกอริทึมที่ใช้พจนานุกรมดังกล่าว ในการแยกแยะหน่วยคำออกจากประโยคในภาษาไทยของระบบ Fast Word Matching ของงานวิจัยนี้ โดยยกตัวอย่างการทำงานจริงของระบบเพื่อแสดงให้เห็นถึงประสิทธิภาพที่เชื่อถือได้สูง (High Reliability) นอกจากนั้นซอฟแวร์ของระบบ Fast Word Matching ของงานวิจัยนี้ได้พัฒนาขึ้นด้วยภาษาซี (C Language) โดยคำนึงถึงความถูกต้อง และความเร็วในการแยกหน่วยคำ ตลอดจนความสะดวกของผู้ใช้ ระบบนี้จะมีการใช้งานที่ไม่ยุ่งยาก และสามารถที่จะนำไปประยุกต์ใช้กับงานด้านเวิร์ดโปรเซสซิ่งภาษาไทย เช่นการตรวจสอบตัวสะกดในภาษาไทย (Spelling Check) และการจัดเรียงพิมพ์ตัวอักษรของหนังสือพิมพ์เป็นต้น โครงสร้างการทำงานของระบบ Fast Word Matching แสดงไว้ในรูปที่ 2.1



รูปที่ 2.1 แสดงส่วนของ Fast Word Matching ในระบบการวิเคราะห์ประโยคภาษาไทย

2.2 โครงสร้างของพจนานุกรมไทยในระบบ Fast Word Matching

2.2.1 การกำหนดดรรชนีของกลุ่มคำศัพท์

คำศัพท์ในพจนานุกรมไทยทั้งหมดจะถูกจัดให้เป็นหมวดหมู่ และเรียงเป็นลำดับก่อนหลังในการเก็บบันทึกไว้ในฐานข้อมูลตามค่าของดรรชนี 3 ตัวที่คิดจากค่าแอสกีนัมเบอร์ของอักขระตัวที่ 1 ค่าแอสกีนัมเบอร์ของอักขระตัวที่ 2 และจำนวนอักขระทั้งหมดของคำศัพท์นั้นๆ โดยที่กำหนดให้ค่าแอสกีนัมเบอร์ของอักขระตัวแรกของคำศัพท์เป็นดรรชนีที่แบ่งคำศัพท์ออกเป็นกลุ่มใหญ่ และให้ค่าแอสกีนัมเบอร์ของอักขระตัวที่สองของคำศัพท์นั้นเป็นดรรชนีตัวที่สองที่แบ่งคำศัพท์ภายในกลุ่มใหญ่ ให้เป็นกลุ่มย่อย และคำศัพท์ภายในกลุ่มย่อยจะถูกจัดเรียงเป็นลำดับตามจำนวนของคำศัพท์นั้นๆ โดยให้จำนวนอักขระมากอยู่ในตำแหน่งต้นๆของกลุ่มคำศัพท์นั้นๆตามลำดับ

ในที่นี้จะกำหนดให้ V เป็นคำศัพท์ทั้งหมดที่บรรจุอยู่ในพจนานุกรมไทยของระบบ

$I_1(j)$ เป็นดรรชนีตัวแรก (First Index) ที่แสดงกลุ่มคำศัพท์ใหญ่กลุ่มที่ j (j= 1, 2, ...)

$I_{j,2}(k)$ เป็นดรรชนีตัวที่ 2 (Second Index) ที่แสดงกลุ่มคำศัพท์ย่อยกลุ่มที่ k ภายในกลุ่มคำศัพท์ใหญ่กลุ่มที่ j (k=1, 2, ...)

$I_{j,k,3}(N)$ เป็นดรรชนีตัวที่ 3 (Third Index) ที่แสดงกลุ่มคำศัพท์ย่อยกลุ่มที่ N โดยที่คำศัพท์นั้นจะอยู่ในกลุ่มย่อยที่ k และในกลุ่มใหญ่ที่ j

เราจะแสดงความสัมพันธ์ของกลุ่มคำศัพท์ในโครงสร้างพจนานุกรมไทยในรูปของเซ็ท (Set) ได้ดังนี้

$$V = \{ I_1(j) \{ I_{j,2}(k) \{ I_{j,k,3}(n) \}\} \} \quad ...(1)$$

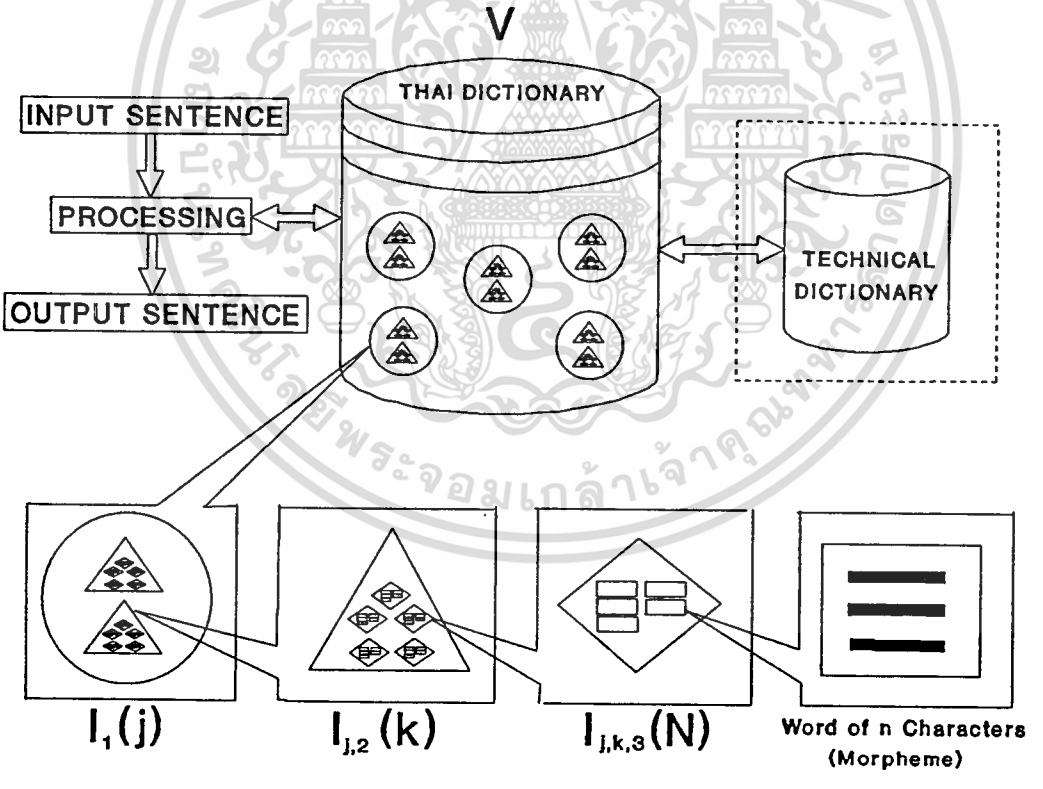
โดยที่

$I_1(j)$ = แอสกีนัมเบอร์ของอักขระตัวแรกของคำศัพท์ ...(2)

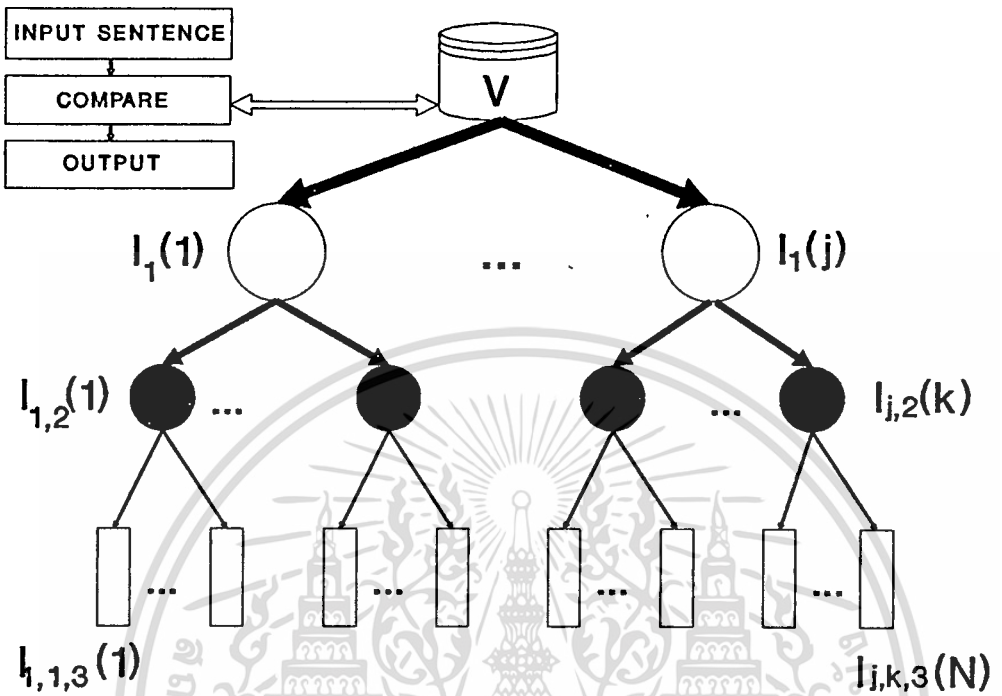
$I_{j,2}(k)$ = แอสกีนัมเบอร์ของอักขระตัวที่สองของคำศัพท์ ...(3)

$I_{j,k,3}(N)$ = n เมื่อ n = จำนวนอักขระของคำศัพท์ ...(4)

จากนิยามของสัญลักษณ์ V, $I_1(j)$, $I_{j,2}(k)$, $I_{j,k,3}(N)$ ข้างต้นทั้งหมด เราสามารถแสดงความสัมพันธ์ของกลุ่มคำศัพท์ และการจัดเรียงคำศัพท์ทั้งหมดที่ประกอบเป็นโครงสร้างพจนานุกรมไทยในระบบ Fast Word Matching ได้ดังแสดงในรูปที่ 2.2ก และ2.2ข โครงสร้างของพจนานุกรมไทยที่แสดงในรูปที่ 2.2 นั้นได้ใช้วิธีการแสดงความสัมพันธ์แบบชาร์ท (Chart) และแบบ ต้นไม้ (Tree) ตามลำดับ เพื่อทำให้เห็นรายละเอียดของโครงสร้างมากขึ้น



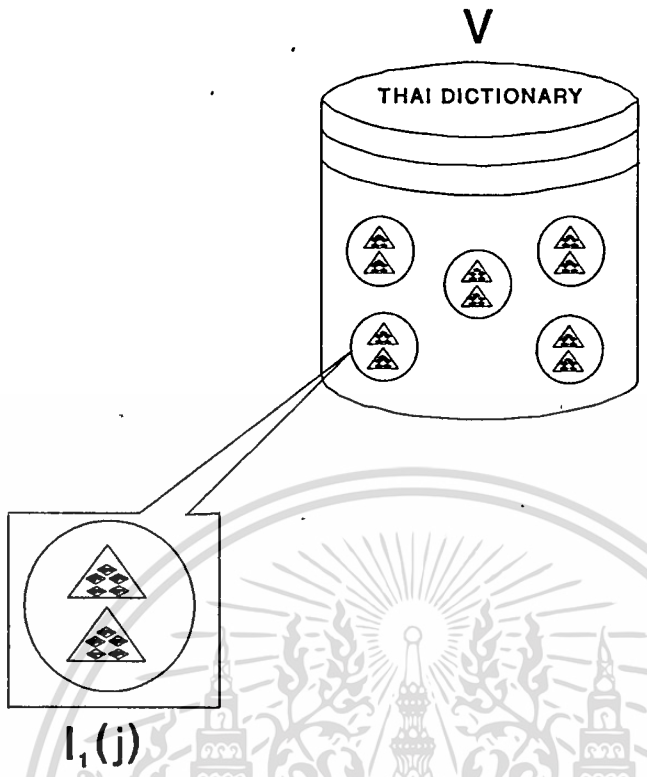
รูปที่ 2.2ก แสดงโครงสร้างของพจนานุกรมไทยในระบบ Fast Word Matching (ในรูปแบบของ Chart)



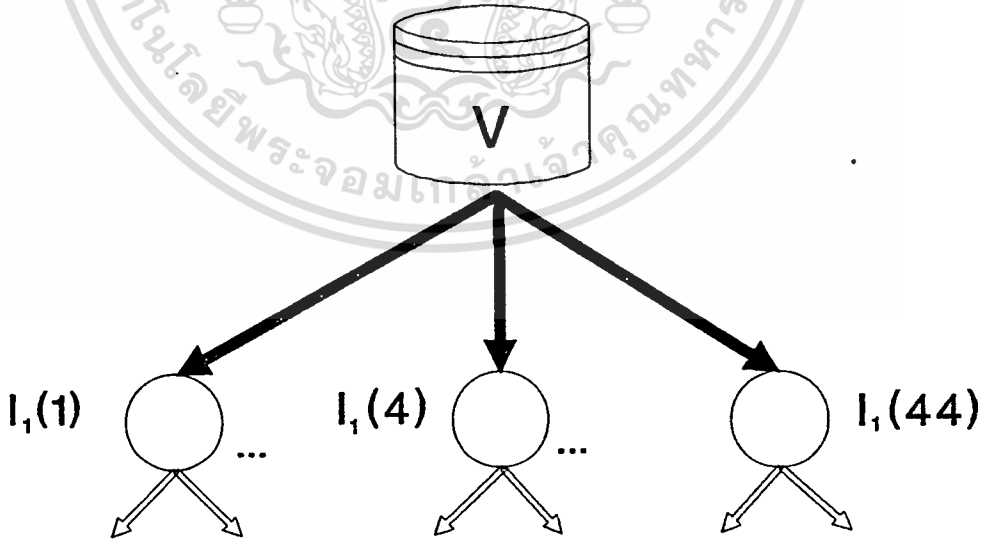
รูปที่ 2.2ข แสดงโครงสร้างของพจนานุกรมไทยในระบบ Fast Word Matching (ในรูปแบบของ Tree)

จากการศึกษาค้นคว้าโครงสร้างของคำในภาษาไทยพบว่า อักขระที่จะเป็นอักษรตัวแรกของคำศัพท์ทั้งหมดในภาษาไทยมีทั้งหมด 44 ตัวอักษร แบ่งเป็นพยัญชนะ 39 ตัวอักษรและสระอีก 5 ตัวอักษร นั่นคือเราจะได้ค่าของ $I_1(j)$ ทั้งหมด 44 ค่า โดยที่ j จะแสดงลำดับของกลุ่มคำศัพท์หรือ j = 1, 2, 3, ..., 44 เป็นผลให้การแบ่งคำศัพท์ภาษาไทยออกเป็นกลุ่มใหญ่ๆตามอักขระตัวแรกนี้จะได้ทั้งหมด 44 กลุ่ม โดยมีดรรชนี $I_1(1)$, $I_1(2)$, ..., $I_1(44)$ เป็นตัวชี้บอกกลุ่มของคำศัพท์เหล่านั้นภายในโครงสร้างของพจนานุกรมไทย ดังแสดงไว้ในตารางที่ 2.1 ซึ่งเป็นตารางกำหนดค่าของ $I_1(j)$ ตามลำดับของอักขระในภาษาไทยและจากสมการที่ (2) เราจะเรียกกลุ่มคำศัพท์ที่ถูกแบ่งเป็นกลุ่มใหญ่ๆตามค่าของแอสกีนัมเบอร์ของอักขระตัวแรกเหล่านี้ว่า **"กลุ่มคำศัพท์ชั้นที่ 1 (First Level Vocaburary)"**

รูปที่ 2.3 แสดงให้เห็นถึงโครงสร้างของกลุ่มคำศัพท์ชั้นที่ 1 ภายในพจนานุกรมไทยของระบบ



รูปที่ 2.3ก แสดง Chart ความสัมพันธ์ของการแบ่งคำศัพท์ชั้นที่ 1



รูปที่ 2.3ข แสดง Tree ความสัมพันธ์ของการแบ่งคำศัพท์ชั้นที่ 1

ตารางที่ 2.1 แสดงการกำหนดค่าดรรชนีตัวที่ 1 ของกลุ่มคำศัพท์ภาษาไทย

อักขระตัวแรกของคำศัพท์	แอสกีนัมเบอร์ของอักขระตัวแรก	กลุ่มดรรชนีตัวที่ 1	ตัวอย่าง
ก	161	$I_1(1)$	การบ้าน, กิน, กลิ่น, กึ่ง
ข	162	$I_1(2)$	ข้าว, ขิง, ขี่, ขาว, ขนุน
ค	164	$I_1(3)$	คอ, ค้อน, คำ, คู่
ฆ	166	$I_1(4)$	ฆ้อง, ฆ่า, ฆาตกร
ง	167	$I_1(5)$	งาน, งอม, ง่วง, งัด
จ	168	$I_1(6)$	จาน, จับ, จอด, จริง
ฉ	169	$I_1(7)$	ฉัน, ฉาก, ฉาย, ฉุน
ช	170	$I_1(8)$	ชาม, ชอบ, ช้อน, ชา
ซ	171	$I_1(9)$	ซอง, ซอก, ซอ
ฌ	172	$I_1(10)$	ฌาปนกิจ, ฌาน
ญ	173	$I_1(11)$	ญัตติ, ญาติ, ญี่ปุ่น
ฎ	174	$I_1(12)$	ฎีกา, ฎีกาธรรมจักร
ฏ	175	$I_1(13)$	ฏาร, ฏางการ
ฐ	176	$I_1(14)$	ฐาน, ฐิติ
ณ	179	$I_1(15)$	ณรงค์
ด	180	$I_1(16)$	ด่าง, ดวงดาว, ดี
ต	181	$I_1(17)$	ตก, ตรง, ตี, ตอง
ถ	182	$I_1(18)$	ถ่าน, ถั่ว, ถือ
ท	183	$I_1(19)$	ท่าน, ท่า, ท่อง, ทำ
ธ	184	$I_1(20)$	ธง, ธรรม

ตารางที่ 2.1 (ต่อ)

อักขระตัวแรกของคำศัพท์	แอสกีนัมเบอร์ของอักขระตัวแรก	กลุ่มดรรชนีตัวที่ 1	ตัวอย่าง
น	185	$I_1(21)$	นาน, นา, นี้, นอน
บ	186	$I_1(22)$	บ้าน, บ่า, บุก
ป	187	$I_1(23)$	ป่า, ปีก, ปาก, ปิด
ผ	188	$I_1(24)$	ผ่า, ผ่าน, ผล
ฝ	189	$I_1(25)$	ฝาก, ฝัน, ฝัง, ฝ้า
พ	190	$I_1(26)$	พยาน, พัก, พัด
ฟ	191	$I_1(27)$	ฟัก, ฟอก, ฟัง, ฟ้อง
ภ	192	$I_1(28)$	ภาพ, ภูมิใจ
ม	193	$I_1(29)$	มา, มาก, มี, มิติ
ย	194	$I_1(30)$	ยอด, ยาง, ยาก, ยาย
ร	195	$I_1(31)$	รัก, ร่ม, รั่ว, ร่วง
ฤ	196	$I_1(32)$	ฤดี, ฤาษี, ฤทธิ์
ล	197	$I_1(33)$	ลวก, ละคร, ลูก, ลวด
ว	199	$I_1(34)$	ว่าง, วาง, วิ่ง
ศ	200	$I_1(35)$	ศาลา, ศาล, ศอก
ส	202	$I_1(36)$	สวย, สาว, สวน, สี่
ห	203	$I_1(37)$	ห้าม, หาย, ห่วง
อ	205	$I_1(38)$	อ่าง, อาย, อยู่
ฮ	206	$I_1(39)$	ฮิปโป, ฮอลแลนด์
เ	224	$I_1(40)$	เกิด, เชา, เคย, เงิน
แ	225	$I_1(41)$	แก่, แยก, แปลก, แตก

ตารางที่ 2.1 (ต่อ)

อักขระตัวแรกของคำศัพท์	แอสกีนัมเบอร์ของอักขระตัวแรก	กลุ่มดรรชนีตัวที่ 1	ตัวอย่าง
โ	226	$I_1(42)$	โรงเรียน, โรงแรม
ใ	227	$I_1(43)$	ใคร, ใจ, ให้, ใส่
ไ	228	$I_1(44)$	ไป, ไหน, ไกล, ไส้

สำหรับการกำหนดค่าของ $I_{j,2}(k)$ หรือดรรชนีตัวที่สอง ตามสมการที่ (3) เพื่อแบ่งคำศัพท์ชั้นที่ 1 ให้เป็นกลุ่มย่อยตามค่าแอสกีนัมเบอร์ของอักขระตัวที่สองของคำศัพท์นั้นจะต้องกำหนดค่า k ให้สัมพันธ์กับการปรากฏ ของอักขระตัวที่สองในคำศัพท์ภาษาไทย นั่นคือจำนวนกลุ่มย่อยของคำศัพท์ภายในกลุ่มคำศัพท์ชั้นที่ 1 แต่ละกลุ่มจะมีจำนวนเท่าใดนั้นขึ้นอยู่กับความเป็นไปได้ในหลักภาษาไทยของการผสมอักขระตัวที่สองกับอักขระตัวที่หนึ่งของกลุ่มคำศัพท์ชั้นที่หนึ่งดังกล่าว

ในที่นี้กำหนดให้ K_j เป็นจำนวนอักขระตัวที่สองที่สามารถผสมกับอักขระตัวแรกของคำศัพท์ชั้นที่หนึ่งในกลุ่ม $I_1(j)$ ดังนั้นจะได้ $k = 1, 2, \ldots, K_j$ โดยที่ $K_j > 0$ เราจะแสดงความสัมพันธ์ของ $I_1(j)$ และ $I_{j,2}(k)$ ในรูปของเซ็ทได้ดังสมการที่ 5

$$I_1(j) = \{ I_{j,2}(k) \} \qquad j = 1, 2, \ldots, 44$$
$$k = 1, 2, \ldots, K_j \qquad \ldots(5)$$

ค่าของ K_j ได้แสดงไว้ในตารางที่ 2.2 และตัวอย่างของการแบ่งกลุ่มคำศัพท์ในชั้นนี้ได้แสดงในตารางที่ 2.3 เราจะเรียกกลุ่มคำศัพท์ย่อยที่ถูกแบ่งตามค่าของแอสกีนัมเบอร์ของอักขระตัวที่สองของคำศัพท์ ซึ่งถูกชี้บอกด้วยดรรชนี $I_{j,2}(k)$ ว่า **"กลุ่มคำศัพท์ชั้นที่ 2 (Second Level**

Vocabulary)"

ตารางที่ 2.2 แสดงจำนวนกลุ่มคำศัพท์ชั้นที่ 2 ในแต่ละกลุ่มคำศัพท์ชั้นที่ 1

กลุ่มคำศัพท์ชั้นที่หนึ่งแสดงด้วยดรรชนีตัวที่ 1	จำนวนกลุ่มคำศัพท์ชั้นที่สอง
$I_1(1)$	$K_1 = 20$
$I_1(2)$	$K_2 = 15$
$I_1(3)$	$K_3 = 14$
$I_1(4)$	$K_4 = 3$
$I_1(5)$	$K_5 = 11$
$I_1(6)$	$K_6 = 13$
$I_1(7)$	$K_7 = 9$
$I_1(8)$	$K_8 = 15$
$I_1(9)$	$K_9 = 10$
$I_1(10)$	$K_{10} = 2$
$I_1(11)$	$K_{11} = 3$
$I_1(12)$	$K_{12} = 2$
$I_1(13)$	$K_{13} = 1$
$I_1(14)$	$K_{14} = 2$
$I_1(15)$	$K_{15} = 1$
$I_1(16)$	$K_{16} = 14$
$I_1(17)$	$K_{17} = 14$
$I_1(18)$	$K_{18} = 13$

ตารางที่ 2.2 (ต่อ)

กลุ่มคำศัพท์ชั้นที่หนึ่งแสดงด้วยดรรชนีตัวที่ 1	จำนวนกลุ่มคำศัพท์ชั้นที่สอง
$I_1(19)$	$K_{19} = 13$
$I_1(20)$	$K_{20} = 9$
$I_1(21)$	$K_{21} = 15$
$I_1(22)$	$K_{22} = 17$
$I_1(23)$	$K_{23} = 17$
$I_1(24)$	$K_{24} = 17$
$I_1(25)$	$K_{25} = 18$
$I_1(26)$	$K_{26} = 17$
$I_1(27)$	$K_{27} = 11$
$I_1(28)$	$K_{28} = 15$
$I_1(29)$	$K_{29} = 15$
$I_1(30)$	$K_{30} = 17$
$I_1(31)$	$K_{31} = 18$
$I_1(32)$	$K_{32} = 9$
$I_1(33)$	$K_{33} = 16$
$I_1(34)$	$K_{34} = 11$
$I_1(35)$	$K_{35} = 13$
$I_1(36)$	$K_{36} = 20$
$I_1(37)$	$K_{37} = 18$
$I_1(38)$	$K_{38} = 18$
$I_1(39)$	$K_{39} = 6$

ตารางที่ 2.2 (ต่อ)

กลุ่มคำศัพท์ชั้นที่หนึ่งแสดงด้วยดรรชนีตัวที่ 1	จำนวนกลุ่มคำศัพท์ชั้นที่สอง
$I_1(40)$	$K_{40} = 18$
$I_1(41)$	$K_{41} = 14$
$I_1(42)$	$K_{42} = 17$
$I_1(43)$	$K_{43} = 14$
$I_1(44)$	$K_{44} = 18$

ตารางที่ 2.3 แสดงการกำหนดค่าดรรชนีตัวที่ 2 เพื่อชี้บอกกลุ่มคำศัพท์ชั้นที่ 2 ของแต่ละกลุ่มคำศัพท์ชั้นที่ 1

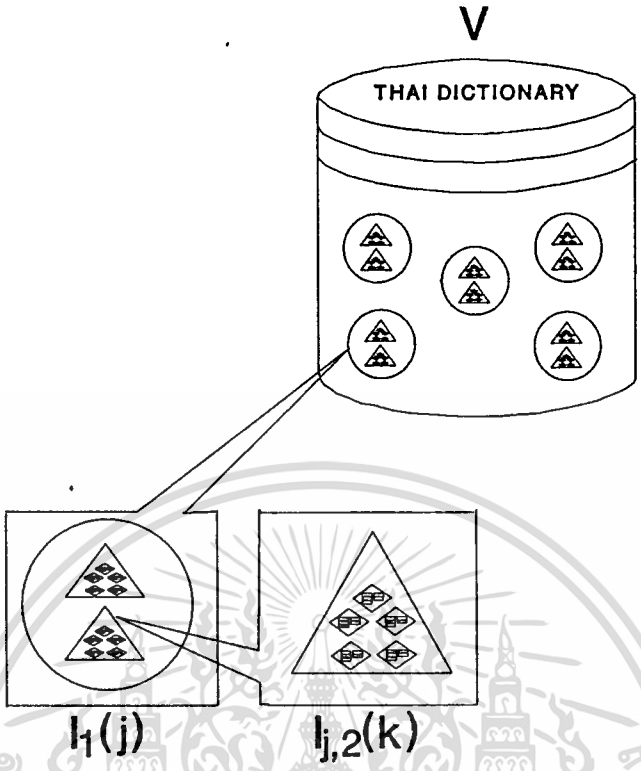
คำศัพท์ชั้นที่ 1		คำศัพท์ชั้นที่ 2			ตัวอย่าง
อักขระตัวที่หนึ่งของคำศัพท์	ดรรชนีตัวที่ 1	อักขระตัวที่สองของคำศัพท์	แอสกีนัมเบอร์ของอักขระตัวที่ 2	ดรรชนีตัวที่สอง	
ก	$I_1(1)$	ก	226	$I_{1,2}(1)$	กก
		ข	225	$I_{1,2}(2)$	ก้น
		ค	224	$I_{1,2}(3)$	ก่น
		ง	223	$I_{1,2}(4)$	กี

ตารางที่ 2.3 (ต่อ)

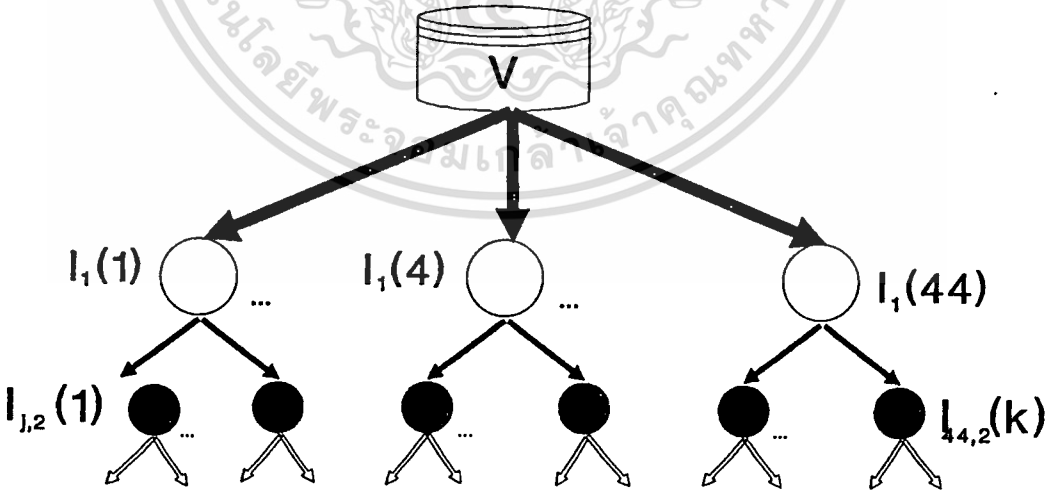
คำศัพท์ชั้นที่ 1		คำศัพท์ชั้นที่ 2			
อักขระตัวที่หนึ่งของคำศัพท์	ดรรชนีตัวที่ 1	อักขระตัวที่สองของคำศัพท์	แอสกีนัมเบอร์ของอักขระตัวที่ 2	ดรรชนีตัวที่สอง	ตัวอย่าง
		ั	221	$I_{1,2}(5)$	กัด
		ี	218	$I_{1,2}(6)$	กีด
		ิ	217	$I_{1,2}(7)$	กิน
		ู	216	$I_{1,2}(8)$	กู
		ุ	215	$I_{1,2}(9)$	กุด
		ำ	207	$I_{1,2}(10)$	กำลัง
		า	206	$I_{1,2}(11)$	กาว
		ะ	204	$I_{1,2}(12)$	กะ
		อ	202	$I_{1,2}(13)$	กอง
		ล	195	$I_{1,2}(14)$	กลม
		ร	193	$I_{1,2}(15)$	กรรไกร
		บ	185	$I_{1,2}(16)$	กบ
		ด	178	$I_{1,2}(17)$	กด
		ฏ	173	$I_{1,2}(18)$	กฏ
		ง	167	$I_{1,2}(19)$	กง
		ก	161	$I_{1,2}(20)$	กก

รูปที่ 2.4 แสดงให้เห็นถึงโครงสร้างของการแบ่งคำศัพท์ชั้นที่ 2 ภายในกลุ่มคำศัพท์ชั้นที่ 1

ของโครงสร้างพจนานุกรมไทยของระบบ Fast Word Matching ที่กล่าวมาทั้งหมด



รูปที่ 2.4ก แสดง Chart ความสัมพันธ์ของการแบ่งคำศัพท์ชั้นที่ 2 ภายในคำศัพท์ชั้นที่ 1



รูปที่ 2.4ข แสดง Tree ความสัมพันธ์ของการแบ่งคำศัพท์ชั้นที่ 2 ภายในคำศัพท์ชั้นที่ 1

ท้ายสุดนี้จะกล่าวถึงการกำหนดดรรชนีตัวที่ 3 ได้แก่ $I_{j,k,3}(N)$ ตามสมการที่ 4 ดังที่กล่าวมาแล้วข้างต้นว่า เรากำหนดค่าดรรชนีตัวที่ 3 จากจำนวนอักขระทั้งหมด นั่นคือค่า n ของคำศัพท์นั้นๆ เพื่อจัดแบ่งและเรียงคำศัพท์เป็นกลุ่มย่อยๆภายในกลุ่มคำศัพท์ชั้นที่ 2 แต่ละกลุ่มตามค่าของ n เราเรียกการแบ่งคำศัพท์เป็นกลุ่มย่อยๆตามค่าของจำนวนอักขระนี้ว่าการแบ่ง "**กลุ่มคำศัพท์ชั้นที่ 3 (Third Level Vocabulary)**" จากการศึกษาคุณสมบัติของคำในภาษาไทยพบว่า คำที่สั้นที่สุดในภาษาไทยจะต้องประกอบด้วยจำนวนอักขระอย่างน้อย 2 ตัว ดังนั้น จำนวนอักขระของคำใดๆจะเป็น

$$n \geq 2 \qquad \ldots(6)$$

ในทางตรงกันข้าม เราไม่สามารถกำหนดจำนวนอักขระสูงสุดของคำศัพท์ชั้นที่ 3 แต่ละกลุ่มได้ อย่างไรก็ตามเราทราบว่าค่า n ของคำศัพท์ใดๆ จะเริ่มจากเลขจำนวนเต็ม 2 และตามด้วย 3, 4, 5, ... ตามลำดับ ถ้ากำหนดให้ $M_{j,k}$ เป็นลำดับที่มีจำนวนอักขระสูงสุดของกลุ่มคำศัพท์ชั้นที่ 3 (ที่แสดงด้วยดรรชนีตัวที่ 3 ได้แก่ $I_{j,k,3}(M_{j,k})$) ภายในกลุ่มคำศัพท์ชั้นที่ 2 กลุ่มที่ k และคำศัพท์ชั้นที่ 1 กลุ่มที่ j ตามลำดับ ดังนั้นจำนวนอักขระของคำศัพท์ใดๆจะเป็น

$$M_{j,k} \geq n \geq 2 \qquad \ldots(7)$$

เมื่อ $j = 1, 2, \ldots, 44$

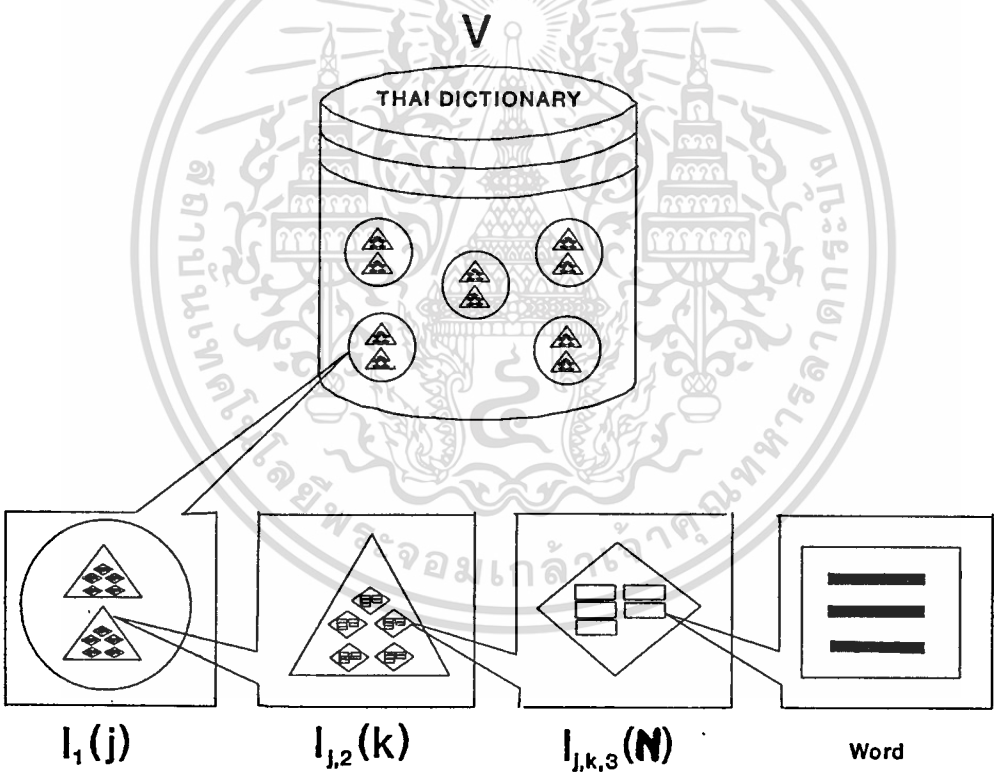
$k = 1, 2, \ldots, K_j$

ในระบบ Fast Word Matching นี้ เราได้จัดเรียงคำศัพท์ชั้นที่ 3 ภายในพจนานุกรมไทยไปตามลำดับค่าของ n โดยให้ค่ามากอยู่ตำแหน่งต้นๆของการเก็บบันทึกลงในพจนานุกรมไทย กล่าวคือสำหรับ j และ k ใดๆ การเรียงคำศัพท์ชั้นที่ 3 ภายในพจนานุกรมจะเป็นดังนี้

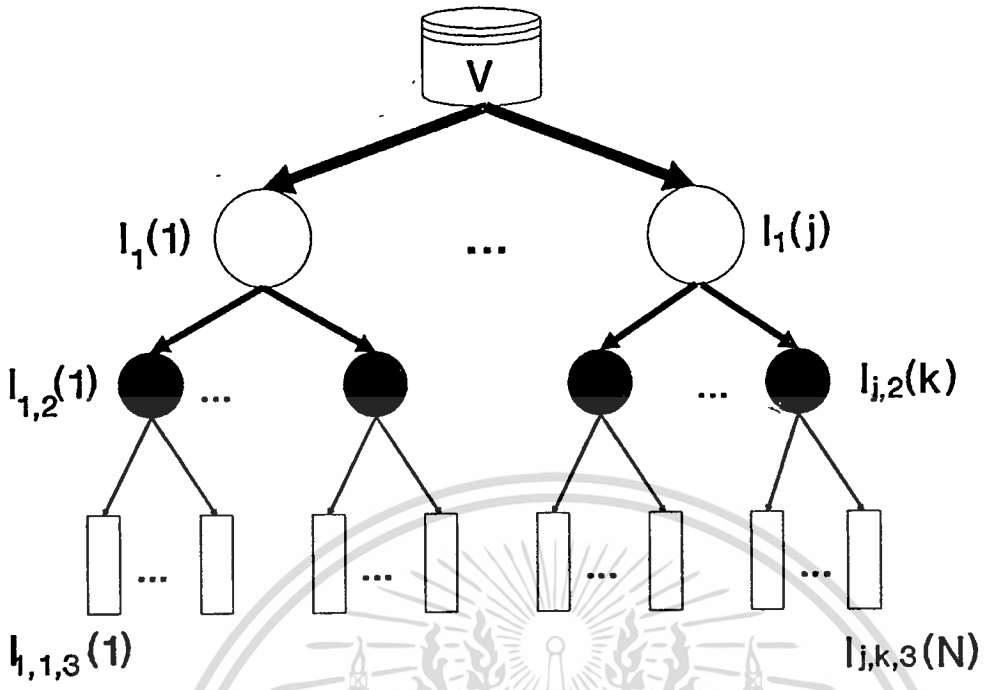
$$I_{j,k,3}(M_{j,k}) > I_{j,k,3}(M_{j,k}-1) > \ldots > I_{j,k,3}(3) > I_{j,k,3(2)} \ldots (8)$$

หมายเหตุ เครื่องหมาย A > B หมายความว่า A มีลำดับมาก่อน B

รูปที่ 2.5 แสดงให้เห็นถึงความสัมพันธ์ของคำศัพท์ชั้นที่ 3 ชั้นที่ 2 และชั้นที่ 1



รูปที่ 2.5ก แสดง Chart ความสัมพันธ์ของการจัดแบ่งคำศัพท์ทั้ง 3 ชั้น



รูปที่ 2.5ข แสดง Tree ความสัมพันธ์ของการจัดแบ่งคำศัพท์ทั้ง 3 ชั้น

2.2.2 ความสัมพันธ์ของดรรชนีในการจัดแบ่งกลุ่มคำศัพท์

จากการกำหนดดรรชนีของกลุ่มคำศัพท์ที่กล่าวในข้อ 2.2.1 เป็นการจัดแบ่งคำศัพท์ภายในพจนานุกรมไทยของระบบ และจัดเรียงให้เป็นกลุ่มเป็นชั้นต่างๆ โดยใช้ค่าของดรรชนีเป็นตัวชี้บอกกลุ่มและชั้นของคำศัพท์นั้นๆ ซึ่งกลุ่มคำศัพท์ชั้นที่ 3 ที่ถูกแสดงลำดับตำแหน่งโดยดรรชนีตัวที่ 3 ได้แก่ $I_{j,k,3}(N)$ จะเป็นคำศัพท์ชั้นในสุดและเป็นคำศัพท์ที่บันทึกในพจนานุกรม (หน่วยความจำของคอมพิวเตอร์) นั่นเอง ทั้งนี้คำศัพท์ชั้นที่ 3 จะถูกล้อมกรอบให้เป็นกลุ่มคำศัพท์ชั้นที่ 2 โดยค่าดรรชนีตัวที่ 2 ของคำศัพท์ได้แก่ $I_{j,2}(k)$ และท้ายที่สุด กลุ่มคำศัพท์ชั้นที่ 2 ก็จะถูกล้อมกรอบให้เป็นกลุ่มใหญ่ขึ้น นั่นคือ เป็นกลุ่มคำศัพท์ชั้นที่ 1 โดยค่าดรรชนีตัวที่ 1 ได้แก่ $I_1(j)$

ดังนั้นความสัมพันธ์ของกลุ่มคำศัพท์แต่ละชั้นขึ้นอยู่กับค่าดรรชนีที่ชี้บอกกลุ่มคำศัพท์เหล่านั้นความสัมพันธ์ของดรรชนีจึงเป็นความสัมพันธ์ของกลุ่มคำศัพท์นั่นเอง ความสัมพันธ์ของดรรชนีขึ้นอยู่กับค่า j, k

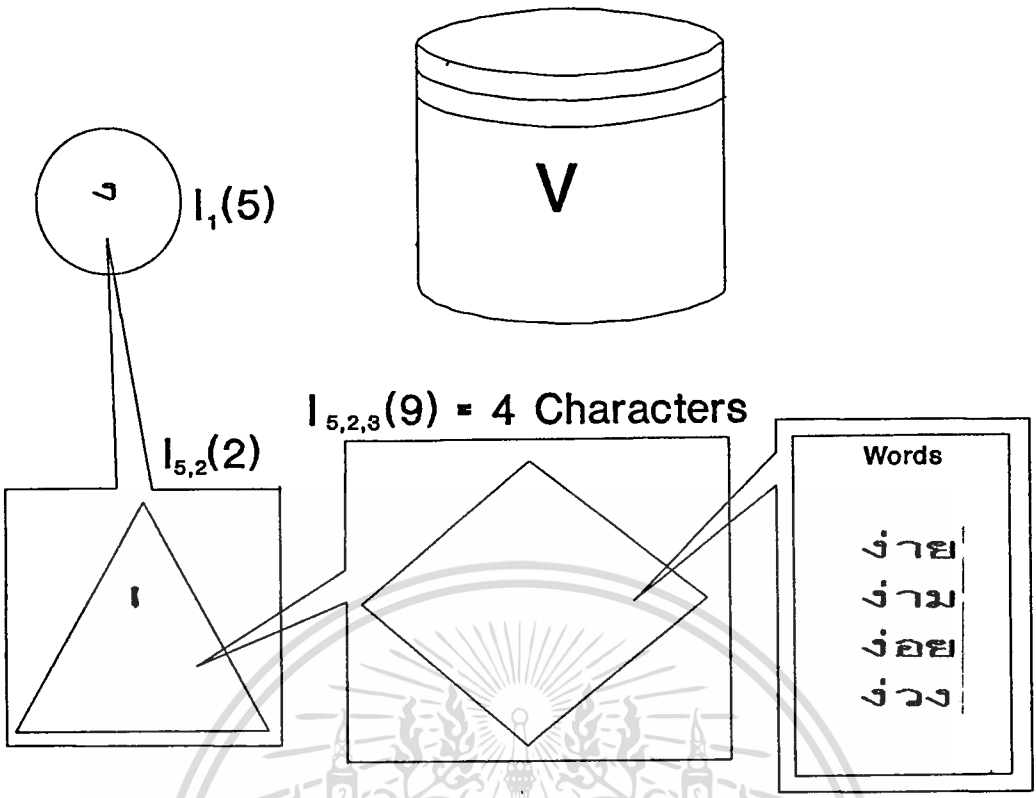
และ N และความสัมพันธ์นี้จะแสดงให้เห็นว่าคำศัพท์แต่ละคำจะอยู่ในกลุ่มคำศัพท์ชั้นที่ 3 ภายในกลุ่มคำศัพท์ชั้นที่ 2 กลุ่มที่เท่าใด และภายในกลุ่มคำศัพท์ชั้นที่ 1 กลุ่มที่เท่าใด ซึ่งเราได้ใช้หลักการย้อนกลับจากข้อความที่กล่าวข้างต้น ในการค้นหาคำศัพท์ในพจนานุกรมโดยการเปรียบเทียบกับส่วนของประโยคที่จะทำการแยกแยะหน่วยคำ นั่นคือการเริ่มจาก ดรรชนีตัวที่ 1 ไปสู่ดรรชนีตัวที่ 2 และเชื่อมไปหาดรรชนีตัวที่ 3 ตามลำดับ ความสัมพันธ์ต่อเนื่องในการค้นหาคำศัพท์จากค่า $I_1(j)$, $I_{j,2}(k)$, และ $I_{j,k,3}(N)$ แสดงได้ดังนี้

$$I_1(j) \Longrightarrow I_{j,2}(k) \Longrightarrow I_{j,k,3}(N) \qquad \ldots(9)$$

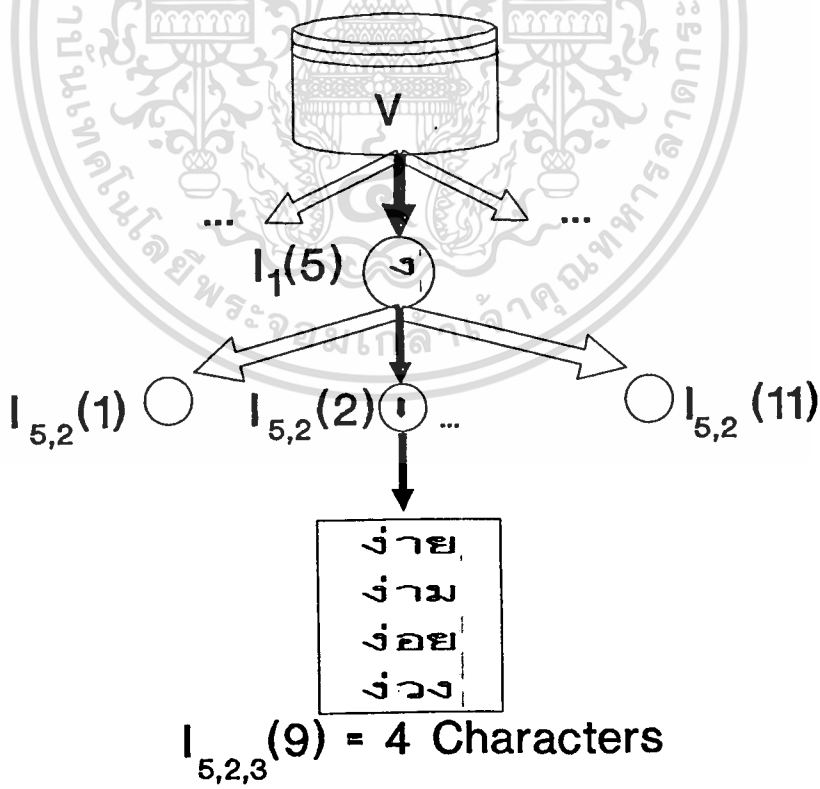
จากความสัมพันธ์ในสมการที่ (9) ทำให้เราพัฒนาระบบซอฟท์แวร์ในการจัดเรียงคำศัพท์ภายในโครงสร้างพจนานุกรมไทยของระบบได้เป็นอย่างดี และในทำนองเดียวกันทำให้เราพัฒนาระบบซอฟท์แวร์ในการค้นหาคำศัพท์เพื่อการแยกแยะคำจากประโยคได้สะดวกและรวดเร็วขึ้น

ในที่นี้จะขอยกตัวอย่างการจัดแบ่งกลุ่มคำศัพท์ที่มีอักขระตัวแรกเป็น "ง" ดังแสดงในรูปที่ 2.6 เป็นตัวอย่างของกลุ่มคำศัพท์ชั้นที่ 3 คือ $I_{ง,2,3}(9)$ ได้แก่ ง่วง ง่วน ... ง่ายซึ่งมีอักขระตัวที่ 1 เป็น "ง" และอักขระตัวที่ 2 เป็น " " (ไม้เอก) นั่นคือมีดรรชนีตัวที่ 1 เป็น $I_1(ง) = 167$ และ ดรรชนีตัวที่ 2 เป็น $I_{ง,2}(2) = 224$ ตามลำดับ

ที่กล่าวมาทั้งหมดในหัวข้อ 2.2 นี้ เป็นโครงสร้างพจนานุกรมไทยของระบบ Fast Word Matching ที่สร้างขึ้น เพื่อใช้เป็นฐานข้อมูลในการแยกแยะหน่วยคำจากส่วนของประโยค ปัจจุบันมีคำศัพท์บรรจุไว้ทั้งสิ้น 14,000 คำ ซึ่งเพียงพอที่จะทำการแยกแยะคำในประโยคภาษาไทย รายละเอียดของกระบวนการแยกแยะคำในประโยคภาษาไทยจะได้กล่าวในหัวข้อที่ 2.3 ต่อไป



รูปที่ 2.6ก แสดงตัวอย่างการจัดแบ่งคำศัพท์ที่เริ่มต้นด้วยอักขระ " ง " ในรูปของ Chart



รูปที่ 2.6ข แสดงการจัดแบ่งคำศัพท์ที่เริ่มต้นด้วยอักษร " ง " ในรูปแบบของTree

2.3 กระบวนการแยกแยะหน่วยคำจากประโยคของระบบ Fast Word Matching

ในการแยกแยะหน่วยคำที่เขียนเรียงติดต่อกันอยู่ในประโยคภาษาไทยด้วยระบบคอมพิวส์เตอร์ ของระบบ Fast Word Matching นั้น จะต้องมีการกำหนดกระบวนการ หรืออัลกอริทึม (Algorithem) ของการค้นหาหน่วยคำในประโยคดังกล่าว แล้วสร้างเป็นระบบซอฟท์แวร์ ซอฟท์แวร์ที่สร้างขึ้นบนพื้นฐานของอัลกอริทึม จะทำหน้าที่ในการค้นหาและตัดสินใจแยกแยะหน่วยคำออกจากประโยค โดยอาศัยฐานข้อมูลคำศัพท์ในพจนานุกรมไทยที่จัดเรียงเป็นหมวดหมู่ตามค่าดรรชนีที่กล่าวมาแล้วในหัวข้อที่ 2.2 เป็นข้ออ้างอิง

ในหัวข้อนี้จะได้กล่าวถึง อัลกอริทึมของการแยกหน่วยคำไทยออกจากประโยคในระบบ Fast Word Matching ที่งานวิจัยนี้ได้พัฒนาขึ้น อัลกอริทึมดังกล่าวจะมีหลักการทำงาน 3 ขั้นตอนใหญ่ๆที่มีความสัมพันธ์กับโครงสร้างพจนานุกรมไทยที่สร้างขึ้นดังนี้คือ

ก) การหาจุดเริ่มต้นของส่วนประโยคเพื่อกำหนดดรรชนีตัวที่ 1 และตัวที่ 2

ข) การคำนวณความยาวของส่วนประโยคเพื่อกำหนดดรรชนีตัวที่ 3

ค) การค้นหาคำในพจนานุกรมไทย เพื่อแยกแยะคำจากประโยค

ในที่นี้ ถ้ากำหนดให้ประโยค (Sentence) ที่จะทำการแยกแยะหน่วยคำประกอบด้วยอักขระ $C_1\ C_2\ C_3\ \ldots\ C_n$ โดยที่ n เป็นจำนวนอักขระทั้งหมดของประโยค และกำหนดให้

Sentence = $C_1\ C_2\ C_3\ \ldots\ C_n$

Word From Dictionary = $D_1\ D_2\ \ldots\ D_t$ = คำศัพท์ที่มีอักขระ t ตัว

$D_1\ D_2\ \ldots\ D_t$ = อักขระของคำศัพท์ตัวที่ 1 ถึง t

Lenght = จำนวนอักขระที่จะใช้ในการเปรียบเทียบ

$I_1(j)$ = ดรรชนีคำศัพท์ชั้นที่ 1 กลุ่มที่ j โดย j = 1, 2, ...

$I_{j.2}(k)$ = ดรรชนีคำศัพท์ชั้นที่ 2 กลุ่มที่ k ซึ่งเป็นสมาชิกของ $I_1(j)$ โดยที่ $k = 1, 2, \ldots, K_j$

$I_{j.k.3}(N)$ = ดรรชนีคำศัพท์ชั้นที่ 3 กลุ่มที่ N ซึ่งเป็นสมาชิกของ $I_{j.2}(k)$ โดยที่ N = 1, 2, ...

I_1 = $I_1(j)$ = ตัวแปรที่เก็บค่าดรรชนีคำศัพท์ชั้นที่ 1 เพื่อใช้ในการเปรียบเทียบ

I_2 = $I_{j,2}(k)$ = ตัวแปรที่เก็บค่าดรรชนีคำศัพท์ชั้นที่ 2 เพื่อใช้ในการเปรียบเทียบ

$I_3(m)$ = $I_{j,k,3}(N)$= ตัวแปรที่เก็บค่าดรรชนีคำศัพท์ชั้นที่ 3 ซึ่งแยกหน่วยคำออกจากประโยคครั้งที่ m ในแต่ละครั้งที่มีดรรชนีชั้นที่ 1 และ 2 เดียวกัน โดย m = 1, 2, 3, ...

C_p = อักขระของประโยคตัวที่ p

C_{p+1} = อักขระของประโยคตัวที่ p+1

$I_{j,k,3}(M_{j,k})$ = ค่าดรรชนีคำศัพท์ชั้นที่ 3 ที่มีลำดับสูงสุด และมีลำดับของดรรชนีคำศัพท์ชั้นที่ 1 เป็น j และ ลำดับของดรรชนีคำศัพท์ชั้นที่ 2 เป็น k

$I_{j,k,3}(MIN)$ = ค่าดรรชนีคำศัพท์ชั้นที่ 3 ที่มีลำดับน้อยที่สุด และมีลำดับของดรรชนีคำศัพท์ชั้นที่ 1 เป็น j และลำดับของดรรชนีคำศัพท์ชั้นที่ 2 เป็น k

Mark1 (Word1) = ตำแหน่งของคำ ที่สามารถจะแยกหน่วยคำออกจากประโยคได้เป็นคำที่ 1 ที่มีดรรชนีชั้นที่ 1 และ 2 เดียวกัน

Mark2 (Word2) = ตำแหน่งของคำ ที่สามารถจะแยกหน่วยคำออกจากประโยคได้เป็นคำที่ 2 ที่มีดรรชนีชั้นที่ 1 และ 2 เดียวกัน

จากคำจำกัดความของสัญลักษณ์ทั้งหมด เราสามารถอธิบายอัลกอริทึมสำหรับการเปรียบเทียบคำศัพท์จากพจนานุกรมไทยเพื่อการแยกแยะหน่วยคำออกจากประโยคของระบบ Fast Word Matching ได้ตามลำดับดังนี้

<Step 1>

หาจำนวนของอักขระทั้งหมดของประโยค (Sentence) ที่จะทำการแยกแยะหน่วยคำ โดยการนับจำนวนอักขระทั้งหมด ซึ่งจำนวนอักขระทั้งหมดของประโยคต้องไม่มากกว่า 2 ดังนี้

Lenght = n ตัว

If (Lenght \< 0) THEN not a sentence

<Step 2>

หาดรรชนีคำศัพท์ชั้นที่ 1 ได้แก่ $I_1(j)$ จากค่าแอสกีนัมเบอร์ของอักขระตัวที่ 1 (C_1) ของประโยคเมื่อเริ่มเปรียบเทียบครั้งแรกดังนี้

$$I_1 = I_1(j) = \text{ASCII of } C_1$$

<Step 3>

หาดรรชนีคำศัพท์ชั้นที่ 2 ได้แก่ $I_{j,2}(k)$ จากค่าแอสกีนัมเบอร์ของอักขระตัวที่ 2 (C_2) ของประโยคเมื่อเริ่มเปรียบเทียบครั้งแรกดังนี้

$$I_2 = I_{j,2}(k) = \text{ASCII of } C_2$$

<Step 4>

หากลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ($I_{j,k,3}(N)$) ได้จากจำนวนอักขระของประโยคทั้งหมดมาเปรียบเทียบหากกลุ่มดรรชนีคำศัพท์ชั้นที่ 3 ในพจนานุกรมดังต่อไปนี้

<Sub Step 4.1>

ถ้าจำนวนอักขระของประโยค(Lenght) มากกว่าหรือเท่ากับจำนวนอักขระของกลุ่มดรรชนีคำศัพท์ชั้นที่ 3 ที่มีจำนวนอักขระมากที่สุด ($I_{j,k,3}(M_{j,k})$) ให้การเปรียบเทียบเริ่มที่กลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 นี้ ไปหากลุ่มดรรชนีคำศัพท์ชั้นที่ 3 ที่มีจำนวนอักขระน้อยที่สุดภายในกลุ่มของคำศัพท์ที่มีดรรชนีคำศัพท์ชั้นที่ 1 และ 2 เดียวกัน ดังนี้

IF Lenght >= $I_{j,k,3}(M_{j,k})$ THEN

Lenght = $I_{j,k,3}(M_{j,k})$

<Sub Step 4.2>

ถ้าจำนวนอักขระของประโยค(Lenght) น้อยกว่าจำนวนอักขระของคำศัพท์ที่มีจำนวนอักขระมากที่สุด($I_{j,k,3}(M_{j,k})$)ในกลุ่มตรรชนีคำศัพท์ชั้นที่ 3 นั้นๆแล้ว

<Inter Sub Step 4.2.1>

ถ้าจำนวนอักขระของประโยคเท่ากับจำนวนอักขระของคำศัพท์ในตรรชนีคำศัพท์ชั้นที่ 3 นั้นๆ ให้เริ่มการเปรียบเทียบคำศัพท์กับประโยค ในกลุ่มของคำศัพท์นั้นๆ ไปหากลุ่มของคำศัพท์ที่มีจำนวนอักขระน้อยที่สุด ในกลุ่มของตรรชนีคำศัพท์ชั้นที่ 3 นั้นๆ ดังนี้

IF Lenght < $I_{j,k,3}(M_{j,k})$ THEN

IF Lenght = $I_{j,k,3}(M_{j,k}-1)$ THEN $I_3(m) = I_{j,k,3}(M_{j,k}-1)$

IF Lenght = $I_{j,k,3}(M_{j,k}-2)$ THEN $I_3(m) = I_{j,k,3}(M_{j,k}-2)$

IF Lenght = $I_{j,k,3}(M_{j,k}-3)$ THEN $I_3(m) = I_{j,k,3}(M_{j,k}-3)$

. .

. .

. .

IF Lenght = $I_{j,k,3}(MIN)$ THEN $I_3(m) = I_{j,k,3}(MIN)$

<Inter Sub Step 4.2.2>

ถ้าจำนวนอักขระของประโยค(Lenght) ไม่เท่ากับจำนวนอักขระของคำศัพท์ในดรรชนีคำศัพท์ชั้นที่ 3 ใดๆเลย แต่จำนวนอักขระของประโยคอยู่ระหว่างกลุ่มของคำศัพท์ที่มีจำนวนอักขระมากที่สุด กับกลุ่มของคำศัพท์ที่มีจำนวนอักขระน้อยที่สุดในดรรชนีคำศัพท์ชั้นที่ 3 นั้นๆ ให้เริ่มเปรียบเทียบคำศัพท์กับประโยค กับกลุ่มที่มีจำนวนอักขระน้อยกว่าจำนวนอักขระของประโยค ที่มีจำนวนอักขระใกล้เคียงกับอักขระของประโยคมากที่สุดในกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 นั้นๆ ไปหากลุ่มดรรชนีคำศัพท์ที่น้อยที่สุดในกลุ่มดรรชนีคำศัพท์ชั้นที่ 3 นั้นๆ ดังนี้

IF Lenght < $I_{j,k,3}(M_{j,k})$ AND Lenght > $I_{j,k,3}(M_{j,k}-1)$
THEN $I_3(m) = I_{j,k,3}(M_{j,k}-1)$

IF Lenght < $I_{j,k,3}(M_{j,k}-1)$ AND Lenght > $I_{j,k,3}(M_{j,k}-2)$
THEN Lenght = $I_{j,k,3}(M_{j,k}-2)$

IF Lenght < $I_{j,k,3}(M_{j,k}-2)$ AND Lenght > $I_{j,k,3}(M_{j,k}-3)$
THEN Lenght = $I_{j,k,3}(M_{j,k}-3)$

. .
. .
. .

IF Lenght < $I_{j,k,3}(MIN+1)$ AND Lenght > $I_{j,k,3}(MIN)$
THEN $I_3(m) = I_{j,k,3}(MIN)$

<Step 5>

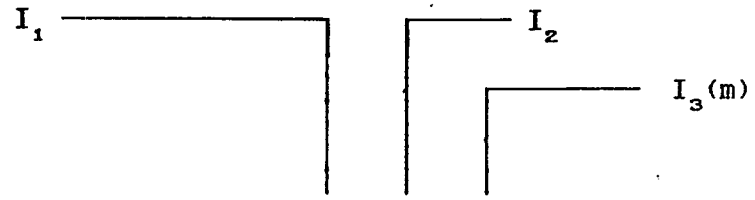
นำคำศัพท์(Word) ที่ได้จากพจนานุกรมมาเปรียบเทียบกับประโยค(Sentence) โดยให้อักขระตัวแรกของคำศัพท์ (D_1) ตรงกับอักขระตัวแรกของประโยคที่จะเปรียบเทียบและให้เปรียบเทียบคำศัพท์กับประโยคเท่ากับจำนวนอักขระของคำศัพท์ที่ได้จากค่าดรรชนีชั้นที่ 3 (I_3(m)) แล้วทำการเปรียบเทียบในลักษณะอักขระต่ออักขระที่ตรงกัน ดังนี้

POSITION	1	2	3	4	5	...	n
SENTENCE	C_1	C_2	C_3	C_4	C_5	...	C_n

<Sub Step 5.1>

หากคำศัพท์เปรียบเทียบกับประโยค แล้วไม่มีผลต่าง ก็ให้ทำเครื่องหมายสามารถแยกคำออกจากประโยคได้ จากนั้นก็นำคำศัพท์คำอื่นที่อยู่ในกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ที่มีดรรชนีชั้นที่ 1 และ 2 เดียวกัน มาทำการเปรียบเทียบต่อ จนกระทั่งถึงคำศัพท์คำสุดท้ายที่อยู่ในชั้นของดรรชนีคำศัพท์ชั้นที่ 3 ที่มีอักขระน้อยที่สุด ที่มีดรรชนีชั้นที่ 1 และ 2 เดียวกัน จึงหยุดการเปรียบเทียบ แล้วแยกหน่วยคำออกจากประโยคตามจุดที่ทำเครื่องหมายสามารถตัดคำได้ ดังนี้

First Comparing :



POSITION	1	2	3	...	t	t+1	...	n
SENTENCE	C_1	C_2	C_3	...	C_t	C_{t+1}	...	C_n
Word From Dic	D_1	D_2	D_3	...	D_t			
RESULT	0	0	0	...	0			

IF ALL POSITION ON RESULT = 0 THEN

POSITION	1	2	3	...	t	t+1	...	n
SENTENCE	C_1	C_2	C_3	...	C_t	C_{t+1}	...	C_n
Word From Dic	D_1	D_2	D_3	...	D_t			
RESULT	0	0	0	...	0			

MARK1 (WORD1)

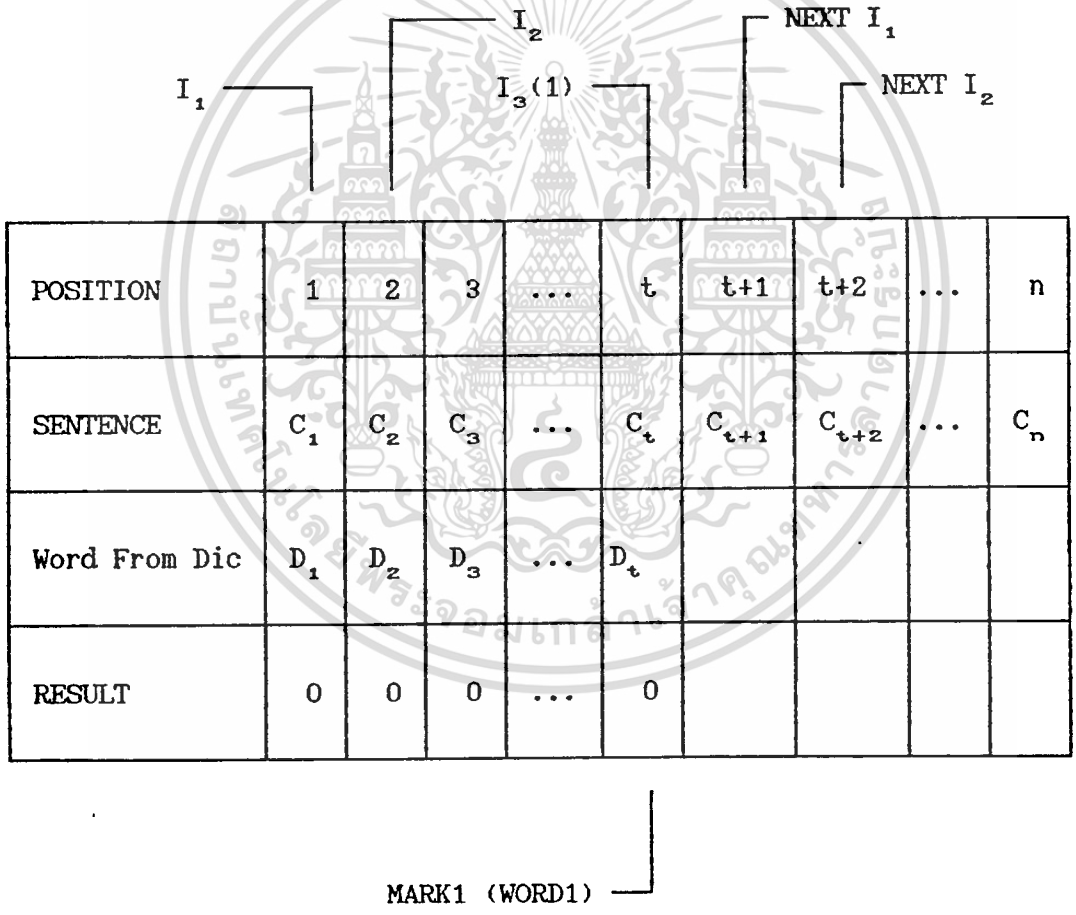
ELSE

NEXT WORD

<Inter Sub Step 5.1.1>

หากที่จุดเริ่มต้นการเปรียบเทียบเดียวกัน(ตรรชนีชั้นที่ 1 และ 2 เดียวกัน) สามารถแยกหน่วยคำออกจากประโยคได้เพียง 1 คำก็ให้อักขระตัวที่อยู่ถัดมาของคำที่ตัดออกจากประโยค เป็นอักขระเริ่มต้นการเปรียบเทียบครั้งต่อไป ดังนี้

First Comparing :

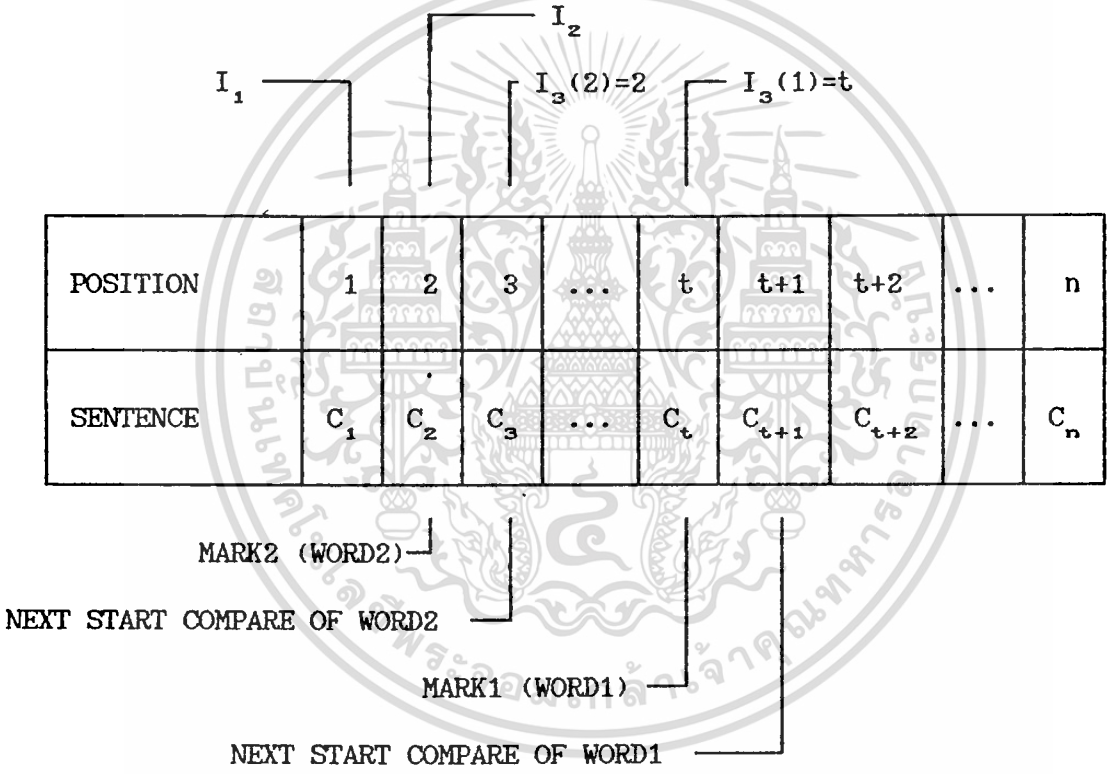


POSITION	1	2	3	...	t	t+1	t+2	...	n
SENTENCE	C_1	C_2	C_3	...	C_t	C_{t+1}	C_{t+2}	...	C_n
Word From Dic	D_1	D_2	D_3	...	D_t				
RESULT	0	0	0	...	0				

<Inter Sub Step 5.1.2>

หากที่จุดเริ่มต้นการเปรียบเทียบเดียวกัน (มีดรรชนีชั้นที่ 1 และ 2 เดียวกัน) สามารถแยกหน่วยคำออกจากประโยคได้มากกว่า 1 คำ ให้อักขระตัวที่อยู่ถัดมาของหน่วยคำ ที่แยกออกจากประโยคในแต่ละครั้งเป็นดรรชนีชั้นที่ 1 เพื่อเป็นจุดเริ่มต้นในการเปรียบเทียบครั้งต่อไป ดังนี้

First Comparing :



<Sub Step 5.2>

หากคำศัพท์เปรียบเทียบกับประโยคแล้วมีผลต่าง ให้นำคำศัพท์คำอื่นที่อยู่ในชั้นนั้นๆ มาทำการเปรียบเทียบต่อ จนกระทั่งถึงคำศัพท์คำสุดท้ายในกลุ่มที่มีอักขระน้อยที่สุด ในชั้นของดรรชนีคำศัพท์ชั้นที่ 3 นั้นๆ จึงหยุดการเปรียบเทียบหรือ เมื่อจบประโยค ดังนี้

First Comparing :

I_2

I_1 $I_3(m)=t$

POSITION	1	2	3	...	t	t+1	t+2	...	n
SENTENCE	C_1	C_2	C_3	...	C_t	C_{t+1}	C_{t+2}	...	C_n
Word From Dic	D_1	D_2	D_3	...	D_t				
RESULT	0	0	1	...	1				

Second Comparing :

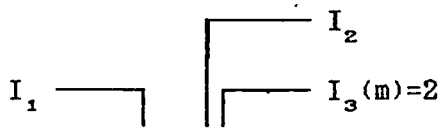
I_2

I_1 $I_3(m)=1$

POSITION	1	2	3	...	t	t+1	t+2	...	n
SENTENCE	C_1	C_2	C_3	...	C_t	C_{t+1}	C_{t+2}	...	C_n
Word From Dic	D_1	D_2	D_3	...	D_t				
RESULT	0	0	0	...	0				

MARK1 (WORD1)

Last Comparing :



POSITION	1	2	3	...	t	t+1	t+2	...	n
SENTENCE	C_1	C_2	C_3	...	C_t	C_{t+1}	C_{t+2}	...	C_n
Word From Dic	D_1	D_2							
RESULT	0	0							

MARK2 (WORD2)

<Inter Sub Step 5.2.1>

ในกลุ่มของดรรชนีคำศัพท์ต่างๆ เมื่อนำคำศัพท์เหล่านั้นมาเปรียบเทียบกับส่วนประโยคที่จะแยกแยะคำแล้ว ไม่มีคำศัพท์ในพจนานุกรมคำใดที่เหมือนกับส่วนประโยค นั่นคือไม่มีการเปรียบเทียบทุกตัวอักษร กรณีใดที่ให้ผลต่างเป็น "0" ทั้งหมด ในกรณีเช่นนี้ให้ถือว่าส่วนประโยคที่นำมาวิเคราะห์นั้นไม่ถูกต้อง หรือมิได้เป็นหน่วยคำที่มีอยู่ในพจนานุกรมภาษาไทย ระบบจะยกเลิกการเปรียบเทียบทั้งหมดในส่วนนั้นๆนั่นคือ

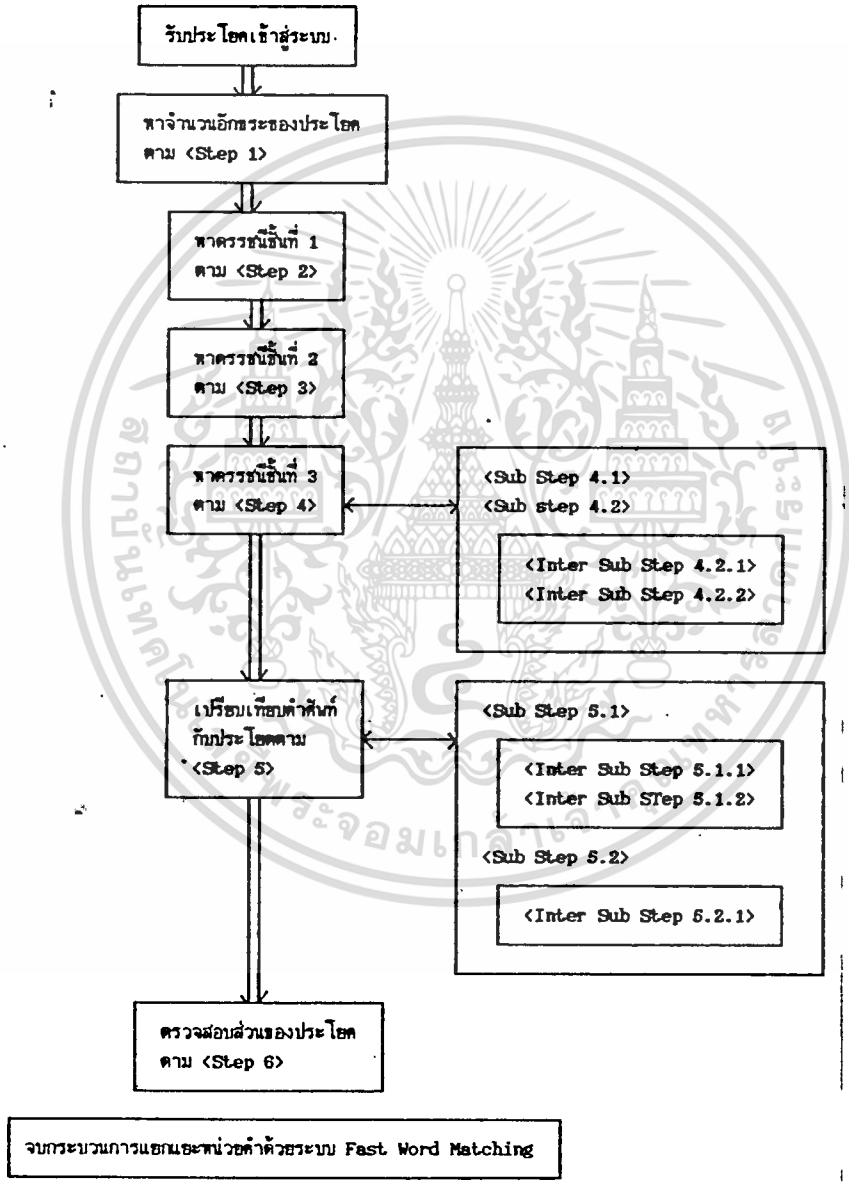
IF all results of comparing are not equal to "0",

THEN no successful segmentation and terminate the searching algorithm.

<Step 6>

หากยังไม่จบส่วนของประโยค ให้กระทำซ้ำตามข้างต้นจาก Step 1 ถึง Step 5

จากขั้นตอนการทำงานข้างต้น สามารถแสดงแผนผังการทำงานในระบบ Fast Word Matching เป็นโฟล์ชาร์ทได้ดังรูปที่ 2.7



รูปที่ 2.7 แผนผังการทำงานของระบบ Fast Word Matching

2.4 ตัวอย่างการทำงานของระบบ Fast Word Matching

ในหัวข้อนี้จะได้ยกตัวอย่างการแยกแยะคำด้วยกระบวนการของระบบ Fast Word Matching ตัวอย่างประโยคภาษาไทยที่ใช้ในตัวอย่างนี้คือ **"ฉันกินข้าว"** ส่วนลำดับขั้นตอนการแยกหน่วยคำ จะเป็นไปตามกระบวนการที่กล่าวมาในบทที่ 2.3 ดังนี้

< ประโยคที่ต้องการแยกหน่วยคำ > = " ฉันกินข้าว "

< การทำงานของระบบ > เมื่อมีอินพุทที่เป็นประโยคต้นแบบถูกป้อนเข้าสู่ระบบ แล้วกระบวนการค้นหา (Searching) และแยกแยะ (Parsing) หน่วยคำในประโยคโดยใช้ฐานข้อมูลคำศัพท์จากพจนานุกรมไทยเป็นฐานเปรียบเทียบก็จะเริ่มทำงานตามลำดับขั้นตอนของอัลกอริทึมระบบดังต่อไปนี้

<Step 1>

หาจำนวนอักขระทั้งหมดของประโยค(Lenght) คือ 10 อักขระดังนี้

POSITION	1	2	3	4	5	6	7	8	9	10
SENTENCE	ฉ	ั	น	ก	ิ	น	ข	้	า	ว

<Step 2>

หาดรรชนีคำศัพท์ชั้นที่ 1 ($I_1(j)$) จากอักขระตัวแรกของจุดที่จะเริ่มเปรียบเทียบของประโยคคือ อักขระ ฉ มีค่าแอสกีเป็น 169 ($I_1 = I_1(7) = 169$)

<Step 3>

หาดรรชนีคำศัพท์ชั้นที่ 2 ($I_{j,2}(k)$) จากอักขระตัวที่สองของจุดที่จะเริ่มเปรียบเทียบของประโยค คือ อักขระ ั มีค่าแอสกีเป็น 209 ($I_2 = I_{7,2}(3) = 209$)

<Step 4 and Step 5>

เข้าหากลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ตามค่าจำนวนอักขระของประโยค (Lenght) ที่นับได้จากประโยค โดยพิจารณาการเลือกกลุ่มต่างๆตามข้างต้นในกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 มีกลุ่มที่มีค่าเท่ากับ 10 ($I_{7,3,3}(2) = 10$) จึงเลือกกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ที่เท่ากับ 10 ตามวิธีการเลือกกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ข้างต้น แล้วนำคำศัพท์ที่ได้จากกลุ่มของดรรชนีชั้นที่ 3 ที่ได้มาเปรียบเทียบกับประโยคที่จะแยกแยะหน่วยคำจนกระทั่งถึงกลุ่มของคำศัพท์ที่มีจำนวนอักขระน้อยที่สุด ซึ่งมีค่าเท่ากับ 2 ($I_{7,3,3}(9) = 2$) ซึ่งผลการเปรียบเทียบจะได้คำศัพท์ ฉัน พอดีกับอักขระในประโยค (RESULT = 0) ทำเครื่องหมายสามารถที่จะแยกแยะหน่วยคำได้ แล้วเปรียบเทียบคำศัพท์คำต่อไปจนกระทั่งถึงคำศัพท์คำสุดท้ายในดรรชนีคำศัพท์ชั้นที่ 3 ที่มีจำนวนอักขระน้อยที่สุดในดรรชนีคำศัพท์ชั้น 3 นั้นๆ (ซึ่งสามารถที่ทำเครื่องหมายสิ้นสุดของหน่วยคำได้ 1 แห่ง (Mark1 (Word1)) คือ ฉัน_กิน ข้าว จากนั้นก็แยกหน่วยคำออกจากประโยคตามเครื่องหมายที่ทำไว้ โดยการทำงานของระบบจะย้อนกลับไปเริ่มที่ Step 1 ตามลำดับดังนี้

<Step 1>

หาจำนวนอักขระที่เหลือทั้งหมดของประโยคเป็นค่าของดรรชนีชั้นที่ 3 มีอยู่ 7 อักขระ ($I_3(1) = I_{j,k,3}(N) = 7$)

POSITION	1	2	3	4	5	6	7
SENTENCE	ก	ิ	น	ข	้	า	ว

<Step 2 and Step 3>

การแยกแยะหน่วยคำออกจากประโยคยังไม่หมด ให้หาค่าดรรชนีชั้นที่ 1 (I_1) และ 2 (I_2) จากตำแหน่งของอักขระที่ต่อจากจุดสิ้นสุดของหน่วยคำที่สามารถแยกแยะหน่วยคำได้ ซึ่งตัวอย่างนี้ดรรชนีคำศัพท์ชั้นที่ 1 คือ ก ($I_1 = I_1(1) = 161$)และดรรชนีคำศัพท์ชั้นที่ 2 คือ ้ ($I_2 = I_{1,7}(4) = 212$)

<Step 4 and Step 5>

เข้าหากลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ตามค่าจำนวนอักขระของประโยค (Lenght) ที่นับได้จากประโยค โดยพิจารณาการเลือกกลุ่มต่างๆตามข้างต้น ในกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 มีกลุ่มที่มีค่าเท่ากับ 7 ($I_{1,7,3}(12) = 7$) จึงเลือกกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ที่เท่ากับ 7 ตามวิธีการเลือกกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ข้างต้น แล้วนำคำศัพท์ที่ได้จากกลุ่มของดรรชนีชั้นที่ 3 ที่ได้มาเปรียบเทียบกับประโยคที่จะแยกแยะหน่วยคำจนกระทั่งถึงกลุ่มของคำศัพท์ที่มีจำนวนอักขระน้อยที่สุด ซึ่งมีค่าเท่ากับ 2 ($I_{1,7,3}(17) = 2$) ซึ่งผลการเปรียบเทียบจะได้คำศัพท์ กิน พอดีกับอักขระในประโยค (RESULT=0) ทำเครื่องหมายสามารถที่จะแยกแยะหน่วยคำได้ แล้วเปรียบเทียบคำศัพท์คำต่อไป จนกระทั่งถึงคำศัพท์คำสุดท้ายในดรรชนีคำศัพท์ชั้นที่ 3 ที่มีจำนวนอักขระน้อยที่สุดในดรรชนีคำศัพท์ชั้น 3 นั้นๆ (ซึ่งสามารถที่ทำเครื่องหมายสิ้นสุดของหน่วยคำได้ 1 แห่ง (Mark1 (Word1)) คือ กิน_ข้าว จากนั้นก็แยกหน่วยคำออกจากประโยคตามเครื่องหมายที่ทำไว้ต่อไป โดยการทำงานของระบบจะย้อนกลับไปที่ Step 1 และ Step ต่อๆไปตามลำดับดังนี้

<Step 1>

ให้จุดเริ่มต้นการเปรียบเทียบครั้งต่อไป (หากประโยคยังไม่สิ้นสุด) อยู่ที่ตำแหน่งต่อจากจุดสิ้นสุดของหน่วยคำที่สามารถแยกแยะหน่วยคำได้ ซึ่งในตัวอย่างคือ ข

<Step 2 and Step 3>

หาจำนวนอักขระที่เหลือทั้งหมดของประโยคเป็นค่าของดรรชนีชั้นที่ 3 มีอยู่ 4 อักขระ

$(I_3(2) = I_{j,k,3}(N) = 4)$

POSITION	1	2	3	4
SENTENCE	ข	้	า	ว

<Step 2 and Step 3>

การแยกแยะหน่วยคำออกจากประโยคยังไม่หมด ให้หาค่าดรรชนีชั้นที่ 1(I_1) และ 2 (I_2) จากตำแหน่งของอักขระที่ต่อจากจุดสิ้นสุดของหน่วยคำที่สามารถแยกแยะหน่วยคำได้ ซึ่งในตัวอย่างนี้ ดรรชนีคำศัพท์ชั้นที่ 1 คือ ข ($I_1(2) = 162$)และดรรชนีคำศัพท์ชั้นที่ 2 คือ ้ ($I_{2,2}(1) = 225$)

<Step 4 and Step 5>

เข้าหากลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ตามค่าจำนวนอักขระของประโยค (Lenght) ที่นับได้จากประโยค โดยพิจารณาการเลือกกลุ่มต่างๆตามข้างต้น ในกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 มีกลุ่มที่มีค่าเท่ากับ 4 ($I_{2,2,3}(13) = 4$) จึงเลือกกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ที่เท่ากับ 4 ตามวิธีการเลือกกลุ่มของดรรชนีคำศัพท์ชั้นที่ 3 ข้างต้น แล้วนำคำศัพท์ที่ได้จากกลุ่มของดรรชนีชั้นที่ 3 ที่ได้มาเปรียบเทียบกับประโยคที่จะแยกแยะหน่วยคำจนกระทั่งถึงกลุ่มของคำศัพท์ที่มีจำนวนอักขระน้อยที่สุด ซึ่งมีค่าเท่ากับ 2 ($I_{2,2,3}(15)=2$)ซึ่งผลการเปรียบเทียบจะได้คำศัพท์ ข้าว พอดีกับอักขระในประโยค (RESULT = 0) ทำเครื่องหมายสามารถที่จะแยกแยะหน่วยคำได้ (Mark1 (Word1)) แล้วเปรียบเทียบคำศัพท์คำต่อไป ได้คำว่า ข้า อีกหนึ่งคำหนึ่ง (RESULT = 0) ทำเครื่องหมายสามารถที่จะแยกแยะหน่วยคำได้ (Mark2 (Word2)) แล้วเปรียบเทียบคำศัพท์คำต่อไป จนกระทั่งถึงคำศัพท์คำสุดท้ายในดรรชนีคำศัพท์ชั้นที่ 3 ที่มีจำนวนอักขระน้อยที่สุดในดรรชนีคำศัพท์ชั้น 3 นั้นๆ ซึ่งสามารถที่ทำเครื่องหมายสิ้นสุดของหน่วยคำได้ 2 แห่ง ตำแหน่งแรก (Mark1 (Word1)) คือ ข้าว กับ ตำแหน่งที่ 2 (Mark2 (Word2)) คือ ข้า จากนั้นก็แยกหน่วยคำออกจาก

ประโยคตามเครื่องหมายที่ทำไว้ต่อไป โดยการทำงานของระบบจะย้อนกลับไปที่ Step 1 ดังนี้

<Step 1>

หาจำนวนอักขระที่เหลือทั้งหมดของประโยค หลังเครื่องหมายตัดคำตำแหน่งแรก (Mark1 (Word1)) ไม่มีส่วนของประโยคเหลือ (Lenght = 0) ส่วนหลังเครื่องหมายตัดคำตำแหน่งที่ 2 (Mark2 (Word1)) เหลือส่วนของประโยคอยู่ จึงทำการหาจำนวนอักขระของส่วนของประโยค จำนวนอักขระที่เหลือทั้งหมดของประโยคเป็นค่าของดรรชนีชั้นที่ 3 มีอยู่ 1 อักขระ ($I_3(3) = I_{j.k.3}(N) = 1$)

POSITION 1

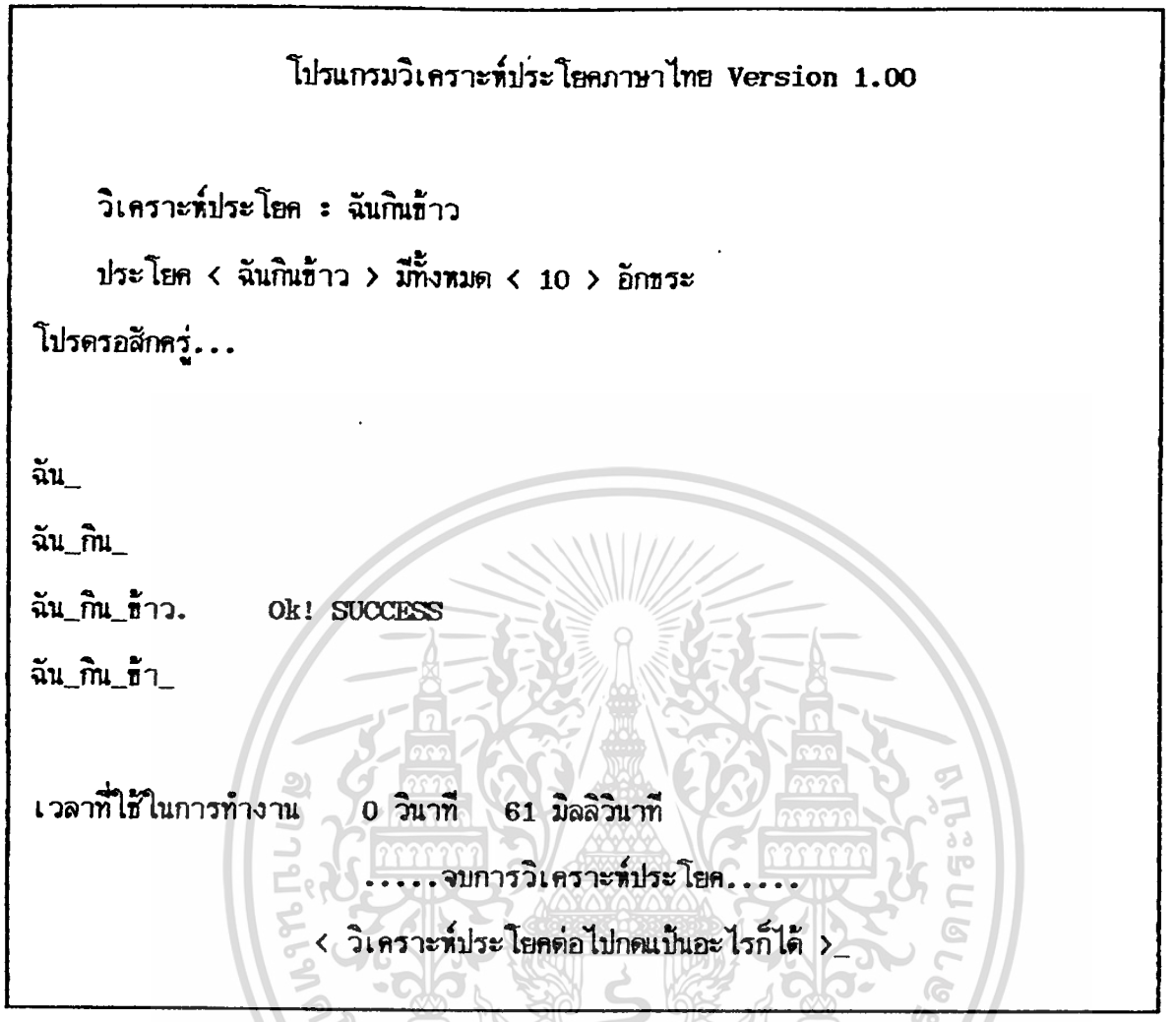
SENTENCE ว

<Step 1>

การแยกแยะหน่วยคำออกจากประโยคยังไม่หมดให้หาค่าดรรชนีชั้นที่ 1(I_1) และ 2(I_2) จากตำแหน่งของอักขระที่ต่อจากจุดสิ้นสุดของหน่วยคำที่สามารถแยกแยะหน่วยคำได้ ซึ่งในตัวอย่างนี้ ดรรชนีคำศัพท์ชั้นที่ 1 คือ ว ($I_1(34) = 199$) แต่ดรรชนีคำศัพท์ชั้นที่ 2 ไม่สามารถจะหาได้ เนื่องจากไม่มีอักขระ หยุดการแยกแยะหน่วยคำออกจากประโยค ในส่วนของตำแหน่งเริ่มต้นการเปรียบเทียบ Mark2 (Word2)

ผลลัพธ์ที่ได้จากการแยกแยะคือ **ฉัน กิน ข้าว**

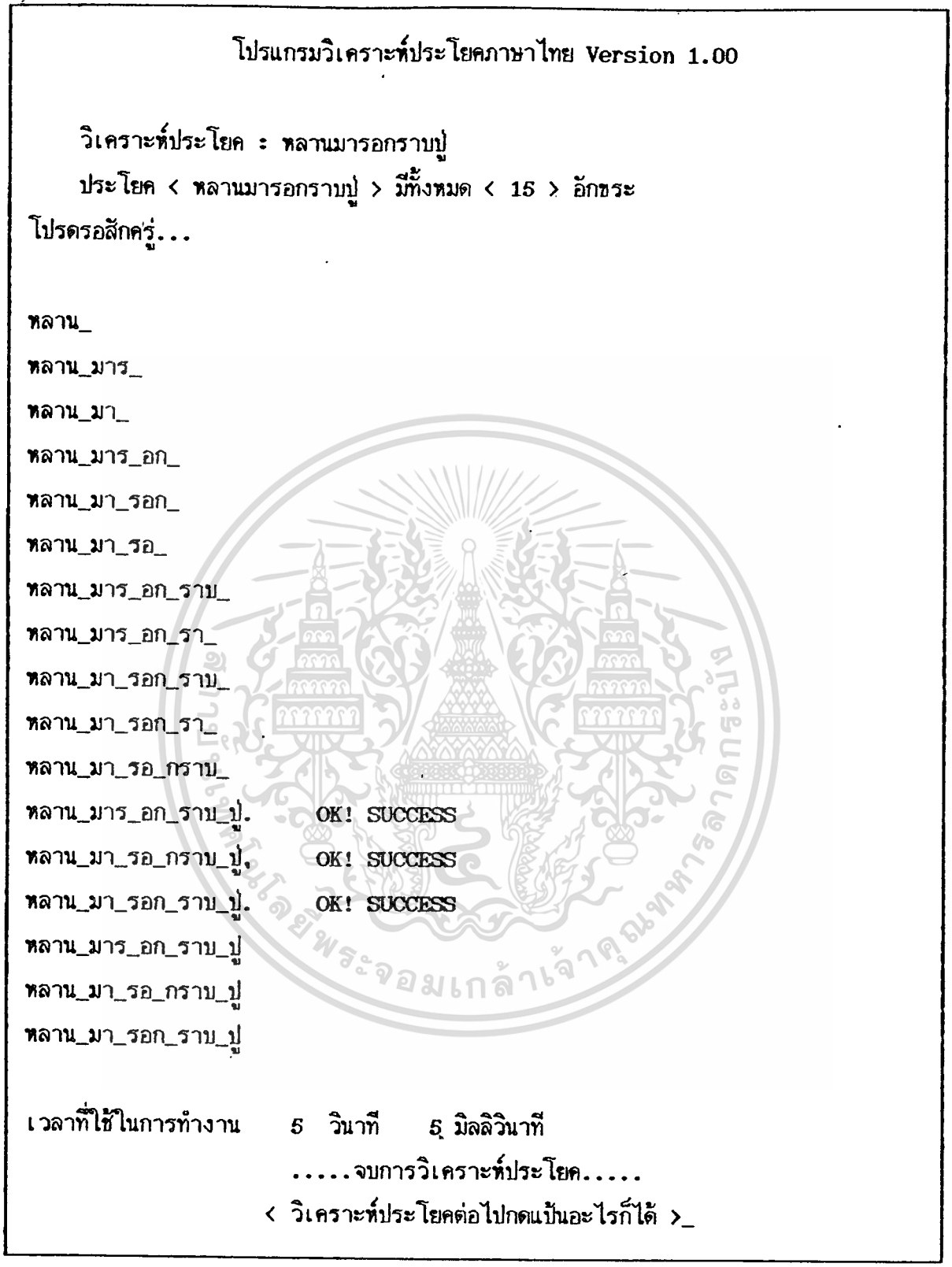
จากวิธีการข้างต้น ได้เขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซีแยกแยะคำออกจากประโยค ผลลัพธ์จากการแยกแยะคำออกจากประโยคโดยซอฟท์แวร์ที่สร้างขึ้นแสดงได้ดังรูปที่ 2.8



รูปที่ 2.8 ผลลัพธ์ของการแยกแยะหน่วยคำจากประโยค"ฉันกินข้าว"ด้วยซอฟท์แวร์ของระบบ Fast Word Matching

สำหรับการแยกแยะหน่วยคำของประโยค "ฉันกินข้าว" ที่ใช้เป็นตัวอย่างนี้ จะให้ผลลัพธ์ของการแยกแยะเพียงกรณีเดียว ถ้าเป็นเช่นนี้ก็จะไม่มีปัญหาทางด้านความกำกวมของหน่วยคำที่ได้จากการแยกแยะ และสามารถนำผลลัพธ์นี้ไปวิเคราะห์ทางโครงสร้างได้ทันที

อย่างไรก็ตาม ดังที่ได้กล่าวมาแล้วว่า ในบางครั้ง สำหรับการแยกแยะหน่วยคำของบางประโยค อาจจะให้ผลลัพธ์ของการแยกแยะหน่วยคำได้มากกว่า 1 กรณี ยกตัวอย่างเช่น ประโยค "หลานมารอกราบปู่" เมื่อผ่านระบบแยกแยะหน่วยคำ Fast Word Matching จะให้ผลลัพธ์ 3 กรณี ดังแสดงในรูปที่ 2.9 ดังนี้



รูปที่ 2.9 แสดงผลลัพธ์ที่ได้จากระบบ Fast Word Matching สำหรับประโยค "หลานมารอกราบปู่"

ผลลัพธ์ที่ 1 หลาน มาร อก ราบ ปู

ผลลัพธ์ที่ 2 หลาน มา รอ กราบ ปู

ผลลัพธ์ที่ 3 หลาน มา รอก ราบ ปู

ถ้าพิจารณาทั้ง 3 ผลลัพธ์ดังกล่าวข้างต้น จะพบว่าเฉพาะด้านของการแยกหน่วยคำแล้วนับได้ว่าเป็นผลลัพธ์ที่ถูกต้องทั้งหมด เพราะทุกๆหน่วยคำที่แยกได้ในแต่ละผลลัพธ์มีการบัญญัติเป็นคำศัพท์ในพจนานุกรมไทยทั้งสิ้น แต่เมื่อพิจารณาทางด้านโครงสร้าง และด้านความหมายแล้วจะมีผลลัพธ์ที่ถูกต้องเพียง กรณีเดียว สำหรับตัวอย่างประโยคนี้ก็คือ ผลลัพธ์ที่ 2 ดังนั้นในการที่จะให้ระบบคอมพิวเตอร์ตัดสินเลือกการแยกแยะหน่วยคำที่มีโครงสร้างไวยากรณ์ถูกต้องนั้น จะต้องใช้กฎไวยากรณ์ทางภาษามาตัดสิน ซึ่งงานวิจัยนี้ได้เสนอระบบการวิเคราะห์โครงสร้างประโยคภาษาไทย M-ATN ดังที่จะได้อธิบายโดยละเอียดในบทที่ 4 ด้วยระบบซอฟท์แวร์ของกระบวนการ M-ATN นั้นเราสามารถตัดสินเลือกผลลัพธ์ที่ 2 ได้อย่างถูกต้อง

2.5 สรุปผลการทดลอง

จากการทดลองแยกแยะคำไทยออกจากประโยคด้วยวิธี Fast Word Matching จากประโยคกว่า 1,000 ประโยค ด้วยพจนานุกรมคำศัพท์ประมาณ 14,000 คำ ผลลัพธ์ที่ได้จากการแยกแยะคำในประโยค ยังไม่ปรากฏข้อผิดพลาดในการแยกประโยคเลย แต่ในบางกรณีอาจจะมีผลลัพธ์ได้มากกว่า 1 ประโยค ซึ่งการที่จะเลือกว่าประโยคใดถูกต้องนั้น จะต้องอาศัยกฎทางไวยากรณ์ (Syntax) และความหมาย (Semantic) เป็นตัวช่วยตัดสินใจ แต่เนื่องจากวิธี Fast Word Matching นี้ไม่ได้ใช้กฎเกณฑ์ทางไวยากรณ์และความหมายมาช่วยในการแยกแยะคำในประโยค ทำให้ไม่สามารถเลือกได้ว่าประโยคใดถูกต้องที่สุด ซึ่งในบทต่อไปจะกล่าวถึงงานวิจัยของเราในด้านวิธีการนำกฎเกณฑ์ทางไวยากรณ์และความหมายทางด้านภาษาศาสตร์มาช่วยในการแยกแยะคำในประโยคให้ถูกต้องมากยิ่งขึ้น ในส่วนของซอฟท์แวร์ระบบ Fast Word Matching ที่ได้พัฒนาขึ้นตลอดจน ผลของการปฏิบัติงานจริงนั้นจะได้อธิบายโดยละเอียดในบทที่ 5

ฐานความรู้เกี่ยวกับไวยากรณ์ภาษาไทย

3.1 บทนำ

ในการแปลภาษาไทยไปสู่ภาษาอื่นด้วยระบบคอมพิวเตอร์นั้น ในขั้นแรกจะต้องมีการแยกหน่วยคำในประโยค เนื่องจากภาษาไทยมีลักษณะการเขียนคำที่เรียงติดกัน และการแยกหน่วยคำนั้นจะต้องแยกเฉพาะคำที่เป็นคำศัพท์ในภาษาไทย นั่นคือเป็นคำที่มีความหมายและบรรจุอยู่ในพจนานุกรมไทย นอกจากนั้นการแยกหน่วยคำด้วยระบบซอฟท์แวร์คอมพิวเตอร์ จะต้องได้ผลลัพธ์รวดเร็วและถูกต้องที่สุด ความถูกต้องในที่นี้หมายความว่า จะต้องแยกหน่วยคำได้ทุกกรณี การแยกหน่วยคำด้วยระบบซอฟท์แวร์คอมพิวเตอร์จะช่วยลดเวลาของการตัด หรือแยกหน่วยคำด้วยผู้แปล(บุคคล) ในการทำ Pre-editing ซึ่งในวิทยานิพนธ์นี้ได้เสนอผลงานวิจัยเกี่ยวกับระบบแยกแยะหน่วยคำไทย Fast Word Matching ดังที่กล่าวมาแล้วในบทที่ 2

เมื่อทำการแยกแยะหน่วยคำไทยจากประโยคต้นแบบ (Source Language) นั้นในบางครั้งสำหรับบางประโยค อาจจะให้ผลลัพธ์ของการแยกหน่วยคำได้หลายแบบ ซึ่งหน่วยคำในแต่ละแบบที่แยกได้ถูกต้องในด้านความหมายและโครงสร้างของประโยค ดังนั้นจึงจำเป็นต้องมีกระบวนการตัดสินผลของการแยกหน่วยคำที่ถูกต้องของความสัมพันธ์ของหน่วยคำที่แยกได้ กระบวนการนี้เป็นศาสตร์หนึ่งของการวิเคราะห์โครงสร้างภาษาไทย นั่นเอง

ในการวิเคราะห์โครงสร้างของประโยคภาษาไทยนั้น สิ่งที่ต้องใช้เป็นฐานความรู้ก็คือ หลักไวยากรณ์ของการใช้หน่วยคำในภาษาไทย เกี่ยวกับการจัดแบ่งประเภทของกลุ่มคำความสัมพันธ์ทางโครงสร้างของคำ ตลอดจนการเรียงลำดับของคำในประโยคนั้น สิ่งเหล่านี้เป็นความรู้พื้นฐานทางภาษาศาสตร์ ซึ่งงานวิจัยนี้ได้รวบรวมเป็นฐานความรู้ (Grammatical Knowledge Base) ของระบบการวิเคราะห์โครงสร้างประโยคภาษาไทย ดังนั้นในบทที่ 3 นี้จะขออธิบายรายละเอียดทางด้านหลักไวยากรณ์ภาษาไทย[4] ที่ใช้เป็นฐานความรู้ของระบบการวิเคราะห์โครงสร้างประโยคภาษาไทยซึ่งจะกล่าวโดยละเอียดในบทที่ 4

3.2 ชนิดของคำในภาษาไทย

คำ หมายถึง อักษรที่ประสมกันแล้วมีความหมาย ซึ่งแบ่งออกเป็น 7 ชนิดได้แก่

1) คำนาม
2) คำสรรพนาม
3) คำกริยา
4) คำวิเศษณ์
5) คำบุพบท
6) คำสันธาน
7) คำอุทาน

3.2.1 คำนาม

คำนาม คือ คำที่เป็นชื่อของคน สัตว์ สถานที่ สิ่งของ และกริยาอาการต่างๆ แบ่งย่อยออกเป็น 5 ชนิด คือ

ก) **สามานยนาม** คือคำที่เป็นชื่อของคน สัตว์ สถานที่ สิ่งของ และกริยาอาการทั่วไป เช่น

- เด็ก ไป โรงเรียน
- กวาง ชอบกิน หญ้า
- ดินสอ อยู่ใน กระเป๋า

ข) **วิสามานยนาม** คือคำนามที่เป็นชื่อเฉพาะของ คน สัตว์ สถานที่ และสิ่งของ ที่กำหนดขึ้นสำหรับใช้เรียก และชี้เฉพาะเจาะจงลงไปว่าเป็นใคร หรืออะไร เช่น

- สมศรี เป็นคนสวย

- นาย แดง และ นาย ดำ เป็นพี่น้องกัน
- เขาไปสมัครงานที่ กระทรวงศึกษาธิการ

ค) ลักษณนาม คือคำนามที่ทำหน้าที่ประกอบนามอื่น เพื่อแสดงรูปลักษณะ ขนาด หรือประมาณของนามนั้นให้ชัดเจนยิ่งขึ้น เช่น

- รถ 1 คัน
- เรือ ลำ นี้มีพาย 2 เล่ม
- นก ฝูง หนึ่งลงกินข้าวในนา แปลง หนึ่ง

ง) สมุหนาม คือ

1) คำนามที่ทำหน้าที่แสดงหมวดหมู่ของสามานยนามและวิสามานยนาม เช่น

- ฝูง นก บิน ไป ใน อากาศ
- คณะ ครู ประชุม ในห้องสมุด
- กอง ทหาร ตั้งอยู่ที่ เชิงเขา

2) คำนามที่เป็นชื่อสถานที่ หรือองค์การต่างๆ แต่สมมติให้เป็นบุคคลชั้นตามความนิยมของภาษาคือ แทนที่จะให้มีความหมายถึงสถานที่ หรือองค์การต่างๆ ตามรูปศัพท์ แต่เปลี่ยนความหมายให้เป็นกลุ่มคนผู้มีหน้าที่รับผิดชอบในสถานที่ หรือองค์การนั้นๆ เช่น

- ประเทศไทย ยินดีต้อนรับชาวต่างชาติ
- โรงเรียน ต้องการนักศึกษาที่ดี
- กระทรวงกลาโหม ประกาศรับทหารอาสา

จ) อาการนาม คือคำนามซึ่งเกิดจากคำกริยาหรือคำวิเศษณ์ที่มีคำว่า "การ" หรือ"ความ" นำหน้า คำนามชนิดนี้มีลักษณะผิดกับคำนามชนิดอื่น คือใช้คำประสมทั้งสิ้น เช่น การยืน การเดิน การนอน ความรู้ ความชั่ว ความมักง่าย ฯลฯ

3.2.2 คำสรรพนาม

คำสรรพนาม คือคำที่ใช้แทนนาม หรือข้อความที่กล่าวมาแล้ว เพื่อไม่ต้องกล่าวนามหรือข้อความนั้นซ้ำอีกเพราะภาษาต้องการความไพเราะและความหมดจดเกลี้ยงเกลา ถ้าต้องกล่าวคำหรือข้อความซ้ำกันอยู่บ่อยๆ ก็ขาดความสละสลวยสรรพนามแบ่งย่อยออกเป็น 6 ชนิด คือ

ก) บุรุษสรรพนาม คือสรรพนามที่ใช้แทนชื่อผู้พูด ผู้ที่พูดด้วย และผู้ที่พูดถึง แบ่งออกเป็น 3 ชนิด คือ

1) ถ้าใช้แทนชื่อผู้พูด เช่นคำว่า ฉัน ผม ข้าพเจ้า ดิฉัน อาตมา เป็นต้น เรียกว่า **บุรุษที่ 1**

2) ถ้าใช้แทนชื่อผู้ที่พูดด้วย เช่นคำว่า ท่าน เธอ ใต้เท้า ฝ่าพระบาท เป็นต้น เรียกว่า **บุรุษที่ 2**

3) ถ้าใช้แทนชื่อผู้ที่พูดถึง เช่นคำว่า เขา มัน ใคร ผู้ใด เป็นต้น เรียกว่า **บุรุษที่ 3**

ข) ประพันธสรรพนาม คือสรรพนามที่ใช้แทนนาม หรือแทนสรรพนามที่อยู่ติดต่อกันข้างหน้า ได้แก่คำว่า ผู้ ที่ ซึ่ง อัน ดัง ผู้ที่ ผู้ซึ่ง ดังตัวอย่าง ต่อไปนี้

- คน <u>ที่</u> เป็นครูต้องมีความอดทน
- เขาบูชาความรัก <u>ซึ่ง</u> ทำให้เขาตาบอด
- ไม้ <u>อัน</u> อยู่ในห้องคือไม้ตะพด

ประพันธสรรพนาม ตามตัวอย่างนั้น ทำหน้าที่ในคราวเดียวกัน 2 ประการคือ

1) ใช้แทนคำนามหรือสรรพนามที่อยู่ข้างหน้า

2) เป็นตัวขยายนามหรือสรรพนามที่อยู่ข้างหน้า

คำประพันธสรรพนามบางคำ คือ **ที่ ซึ่ง อัน** ใช้ซ้ำกับคำประพันธวิเศษณ์ แต่มีวิธีใช้ต่างกันคือ ถ้าเป็นประพันธสรรพนามต้องเรียงติดต่อกับนามหรือสรรพนาม ถ้าเป็นประพันธวิเศษณ์ต้องเรียงติดต่อกับคำกริยาหรือคำวิเศษณ์ ดังตัวอย่างต่อไปนี้

- คน ที่ ยืนอยู่นั้นเป็นทหาร (ที่ เป็นประพันธสรรพนาม)
- เขาเป็นคนดี ที่ ฉันพึงปรารถนา (ที่ เป็นประพันธวิเศษณ์)
- ของ ซึ่ง วางอยู่ในห้องหายไปไหน (ซึ่ง เป็นประพันธสรรพนาม)
- มันเป็นของดี ซึ่ง ควรแก่การรักษา (ซึ่ง เป็นประพันธวิเศษณ์)

ค) **วิภาคสรรพนาม** คือสรรพนามที่ใช้แทนนามหรือสรรพนามที่แยกออกเป็น แต่ละคน แต่ละสิ่ง หรือแต่ละพวก ได้แก่คำว่า ต่าง บ้าง กัน เช่น

- นักกีฬา ต่าง ทำหน้าที่ของตน
- กรรมกร ต่าง ก็ขนของขึ้นจากเรือ
- พี่น้องสูงเท่า กัน
- เขามองดูซึ่ง กัน และ กัน

คำ **ต่าง บ้าง กัน ที่** เป็นวิภาคสรรพนามนั้น จะต้องใช้แทนนามหรือสรรพนาม ดังตัวอย่างที่แสดงไว้นั้น ถ้าทำหน้าที่ประกอบนาม สรรพนาม กริยา หรือ วิเศษณ์ ต้องนับว่าเป็นคำวิเศษณ์ เช่น

1) **ประกอบนาม** ต่าง คน ต่าง ใจ, ต่าง จิต ต่าง ใจ, คน บ้าง สัตว์ บ้าง

2) **ประกอบสรรพนาม** ต่าง เขา ต่าง เรา, เขา บ้าง เรา บ้าง

3) **ประกอบกริยา** เขาพูด ต่าง กับฉัน, ทำ บ้าง หยุด บ้าง, เขาพยายาม กัน มาก

4) ประกอบวิเศษณ์ เขาเป็นคนดี ต่าง กัน, ทำมาก บ้าง ทำน้อย บ้าง

คำ ต่าง และ กัน ยังใช้เป็นคำชนิดอื่นได้อีก เช่น

- เขา ต่าง กับฉัน	(ต่าง เป็นกริยา)
- ต่าง อยู่บนหลังคา	(ต่าง เป็นนาม)
- กัน มีเพื่อนสามคน	(กัน เป็นบุรุษสรรพนาม)
- ฉัน กัน เงินไว้จำนวนหนึ่ง	(กัน เป็นกริยา)

ง) นิยมสรรพนาม คือสรรพนามที่ใช้แทนนาม หรือข้อความที่กล่าวมาแล้วหรือที่ปรากฏอยู่เฉพาะหน้า เป็นสรรพนามชี้เฉพาะเพื่อบ่งบอกความให้ชัดเจน ได้แก่ คำต่อไปนี้คือ นี่ นั่น โน่น นี้ นั้น โน้น ทั้งนี้ ทั้งนั้น เช่นนี้ เช่นนั้น อย่างนี้ อย่างนั้น อย่างโน้น เช่น

- นี้ คือหนังสือที่ฉันชอบ
- นั่น เป็นมติของกรรมการ
- ฉันอยู่ นี่ สบายกว่าอยู่ โน่น
- นี้ เป็นความเห็นของฉัน
- โน้น คือเขาภูกระดึง

จ) อนิยมสรรพนาม คือสรรพนามที่ใช้แทนนามทั่วๆไป ไม่ชี้เฉพาะเจาะจงเหมือนนิยมสรรพนามได้แก่คำต่อไปนี้คือ ใคร อะไร ไหน ผู้ใด อื่น ผู้อื่น ผู้ใดผู้หนึ่ง ผู้หนึ่งผู้ใด เช่น

- ใคร จะมากับฉันก็ได้
- ฉันไม่ทราบว่า อะไร เป็น อะไร แล้ว
- ผู้ใด เป็นคนดีเราควรคบผู้นั้น

ฉ) ปฤจฉาสรรพนาม คือสรรพนามที่ใช้แทนนามแต่มีเนื้อความเป็นคำถาม ได้แก่ ใคร อะไร ไหน ผู้ใด ฯลฯ คำเหล่านี้มีใช้อยู่แล้วในอนิยมสรรพนาม แต่ต่างกับอนิยมสรรพนาม เพราะใช้เป็นคำถาม ส่วนอนิยมสรรพนามใช้เป็นคำแทนชื่อที่ไม่ชี้เฉพาะหาได้ใช้เป็นคำถามไม่ เช่น

- เมื่อเช้านี้ ใคร มาหาฉัน (เป็นปฤจฉาสรรพนาม)
- ใคร จะมาหาฉันก็ได้ (เป็นอนิยมสรรพนาม)
- อะไร อยู่ในกระเป๋า (เป็นปฤจฉาสรรพนาม)
- ฉันจะรับประทาน อะไร ก็ได้ (เป็นอนิยมสรรพนาม)
- ไหน คือบ้านของท่าน (เป็นปฤจฉาสรรพนาม)
- ฉันจะอยู่ ไหน ก็ได้ (เป็นอนิยมสรรพนาม)

3.2.3 คำกริยา

คำกริยา คือคำที่แสดงอาการของนามและสรรพนาม แบ่งย่อยออกเป็น 5 ชนิด

ก) สกรรมกริยา คือคำกริยาที่ต้องมีกรรมมารับจึงจะให้ใจความสมบูรณ์ครบถ้วนตามกระแสความ เช่น

- ทหาร ถือ ปืน
- คนครัว หุง ข้าว
- ชาวนา ตัด ต้นไม้
- พ่อค้า ขาย ของ

จะเห็นได้ว่า คำกริยาทั้ง 5 ประโยคนั้น ถ้าไม่มีคำนามมาเป็นกรรมรับข้างหลัง จะไม่ได้ความครบสมบูรณ์

ข) อกรรมกริยา คือกริยาที่มีความหมายครบถ้วนในตัวเอง โดยไม่ต้องมีกรรมมารับก็ได้ความหมายชัดเจน เช่น

- นักเรียน <u>เดิน</u> ที่ถนน
- นก <u>บิน</u> ในอากาศ
- เด็ก <u>นั่ง</u> บนเตียง
- ครู <u>ยืน</u> ในห้อง
- ต้นไม้ <u>โค่น</u> ข้างถนน

ค) วิตรรถกริยา คือกริยาที่ไม่สำเร็จความหมายในตัวเอง และจะใช้เป็นกริยาของประธานตามลำพังตัวเองก็ไม่ได้ จะต้องมีคำนาม คำสรรพนาม หรือคำวิเศษณ์มาขยาย จึงจะได้ความ กริยาพวกนี้ได้แก่คำว่า **เป็น เหมือน** เท่า คล้าย คือ เสมือน ดุจ ประดุจ ประหนึ่ง ราวกับ เปรียบเสมือน เช่น

1) ใช้คำนามขยาย
   - นายแดง <u>เป็น</u> ครู
   - งู <u>คือ</u> สัตว์เลื้อยคลานชนิดหนึ่ง

2) ใช้คำสรรพนามขยาย
   - ถ้าเธอ <u>เป็น</u> ฉัน เธอจะรู้สึกอย่างไร
   - เขา <u>คล้าย</u> เธอมาก

3) ใช้คำวิเศษณ์ขยาย
   - เขาพูดอะไร <u>เป็น</u> เท็จไปหมด

ง) กริยานุเคราะห์ คือคำกริยาที่ทำหน้าที่ช่วยกริยาชนิดอื่นให้แสดงความหมายออกมา เป็น กาล มาลา หรือวาจก ต่างๆ เพราะคำกริยาในภาษาไทยมีรูปคงที่ ไม่เปลี่ยนแปลงไปตามกาล มาลาหรือวาจก เหมือนภาษาที่มีวิภัติปัจจัย เช่น ภาษาบาลี สันสกฤต ภาษาอังกฤษ เป็นต้น การที่จะรู้ว่าเป็น กาล มาลา หรือวาจกนั้น นอกจากสังเกตความหมายของรูปประโยค แล้ว จำเป็นจะต้องอาศัยกริยานุเคราะห์เป็นเครื่องช่วยด้วย คำกริยา-นุเคราะห์ที่มีอยู่มากชนิดด้วยกัน จะจำแนกออกเป็นชนิดตามที่ทำหน้าที่ บอกกาล มาลา และวาจก ดังต่อไปนี้

1) ชนิดบอกกาล

1.1) บอกอนุตกาล ได้แก่คำ ย่อม

1.2) บอกปัจจุบันกาล ได้แก่คำ อยู่ ยัง ยัง-อยู่ กำลัง กำลัง-อยู่

1.3) บอกอดีตกาล ได้แก่คำ ได้ เคย

1.4) บอกอนาคตกาล ได้แก่คำ จะ จัก

1.5) บอกกาลสมบูรณ์ ได้แก่ แล้ว เสร็จ

2) ชนิดบอกมาลา

2.1) บอกปริกัลปมาลา ได้แก่คำ ชะรอย รอย เห็นจะ ทีจะ ท่าจะ

2.2) บอกศักดิมาลา ได้แก่คำ คง คงจะ ต้อง

2.3) บอกอานัติมาลา ได้แก่คำ จง อย่า จงอย่า ขอจง ขออย่า อย่าเพิ่ง อย่าเพ่อ ชิ นะ เถอะ เถิด เทอญ

3) ชนิดบอกวาจก

3.1) บอกกรรตุวาจา ได้แก่คำ ให้

3.2) บอกกรรมวาจก ได้แก่คำ ถูก ถูก-ให้ ถูกให้

3.3) บอกการิตวาจก ได้แก่คำ ถูก ถูก-ให้ ถูกให้

จ) กริยาสภาวมาลา คือคำกริยาที่ทำหน้าที่คล้ายกับนาม อาจจะเป็นประธาน เป็นกรรม หรือเป็นบทขยายส่วนใดส่วนหนึ่งของประโยคก็ได้ เช่น

- นอน มีประโยชน์กว่าอริยาบทอื่น
- ฉันชอบ ชี่ จักรยาน

3.2.4 คำวิเศษณ์

คำวิเศษณ์ คือคำที่ทำหน้าที่ประกอบคำนาม สรรพนาม คำกริยา และคำวิเศษณ์ด้วยกันให้ได้ใจความชัดเจนยิ่งขึ้น ตัวอย่าง เช่น

1) ประกอบคำนาม

- คน อ้วน เป็นเพื่อนกับคน ผอม
- โต๊ะ กลม ทาสี เหลือง
- คน ดี มีวาจา ไพเราะ

2) ประกอบคำสรรพนาม

- ท่าน ทั้งหลาย อยากดูใคร บ้าง
- เขา ทั้งหมด จะมาหาเรา ทั้งสอง วันนี้

3) ประกอบคำกริยา

- เขาพูด เพราะ
- ม้าวิ่ง เร็ว
- ผมมีปากกาด้ามเดียว ขอรับ

4) ประกอบคำวิเศษณ์

- เขากินอาหาร จุ มาก (ประกอบ จุ)

- คนอ้วน ตุ๊ตะ วิ่งช้า (ประกอบ อ้วน)

- เขารักน้องมาก เหลือเกิน (ประกอบ มาก)

คำวิเศษณ์แบ่งย่อยออกเป็น 10 ชนิด คือ

ก) ลักษณวิเศษณ์ คือคำวิเศษณ์ที่ประกอบบอกลักษณะต่างๆ เช่น บอกชนิด สี สัณฐาน ขนาด เสียง อาการ กลิ่น รส ความรู้สึก เป็นต้น เช่น ดี ชั่ว ขาว ดำ กลม แบน ใหญ่ เล็ก โครม เปรี้ยง เร็ว ช้า หอม เหม็น เปรี้ยว หวาน เย็น ร้อน ฯลฯ คำนามและคำกริยาที่นำมาใช้เป็นคำวิเศษณ์ โดยมากเป็นลักษณวิเศษณ์ และคำวิเศษณ์ใดๆที่จัดเข้าในคำวิเศษณ์ชนิดอื่นๆไม่ได้ ต้องนับว่าเป็นลักษณะวิเศษณ์ทั้งสิ้น

ข) กาลวิเศษณ์ คือคำวิเศษณ์ที่ประกอบบอกเวลา เช่น เช้า สาย บ่าย เที่ยง เย็น ค่ำ อดีต อนาคต ปัจจุบัน ฯลฯ

ค) สถานวิเศษณ์ คือคำวิเศษณ์ที่ประกอบบอกสถานที่ เช่น บน ล่าง เหนือ ใต้ ไกล ใกล้ บก น้ำ บ้าน ป่า ฯลฯ และคำวิเศษณ์ชนิดนี้ ถ้ามีคำนามหรือสรรพนามมารับข้างหลังจะกลายเป็นคำบุพบท

ง) ประมาณวิเศษณ์ คือคำวิเศษณ์ที่ประกอบบอกจำนวน แบ่งออกเป็น 2 ชนิดคือ

1) บอกจำนวนนับ เช่น หนึ่ง สอง สาม ที่หนึ่ง ที่สอง ฯลฯ จะใช้เป็นตัวเลข หรือ ตัวหนังสือก็ได้

2) บอกจำนวนปริมาณ คือมิได้บ่งบอกชัดลงไปว่าเท่านั้นเท่านี้ เป็นแต่กำหนดว่ามากหรือน้อยซึ่งพอจะรู้ความหมายได้โดยปริมาณ เช่น มาก น้อย หลาย ทั้งหลาย จุ ทั้งปวง ทั้งหมด บรรดา คนละ สิ่งละ ต่าง บาง กัน ฯลฯ

จ) **นิยมวิเศษณ์** คือคำวิเศษณ์ที่ประกอบบอกความชี้เฉพาะหรือจำกัดลงไปว่าเป็นเช่นนั้นเช่นนี้ หรือสิ่งนั้นสิ่งนี้ เช่น นี่ นั่น โน่น นี้ นั้น โน้น ทั้งนี้ ทั้งนั้น อย่างนี้ อย่างนั้น ดังนี้ ดังนั้น แท้ จริง เฉพาะ เอง ดอก แน่นอน ทีเดียว เจียว

ฉ) **อนิยมวิเศษณ์** คือคำวิเศษณ์ที่ประกอบบอกบอกความไม่ชี้เฉพาะ หรือไม่จำกัดลงไปว่าเป็นเช่นนั้นเป็นเช่นนี้ หรือเป็นสิ่งนั้นเป็นสิ่งนี้ เช่น ใด ไร ไหน กี่ อะไร ทำไม ฉันใด เช่นไร ฯลฯ แต่คำพวกนี้จะต้องไม่ใช้ในข้อความที่เป็นคำถาม หรือสงสัยจึงจะเป็นอนิยมวิเศษณ์ ถ้าใช้ในข้อความที่เป็นคำถามหรือสงสัย เรียกว่า ปฤจฉาวิเศษณ์

ช) **ปฤจฉาวิเศษณ์** คือคำวิเศษณ์ที่ประกอบบอกเนื้อความเป็นคำถาม หรือความสงสัยเช่น ใด ไร ไหน กี่ อะไร ทำไม ฉันใด เช่นไร ไหม อันใด อย่างไร เท่าไร ไย หรือ หนอ ฯลฯ

ซ) **ประติชญาวิเศษณ์** คือคำวิเศษณ์ที่ประกอบบอกเสียงเรียกร้องและเสียงขานรับ เพื่อแสดงความสละสลวยของภาษา และแสดงความเป็นกันเองในระหว่างผู้พูดเช่น จ๋า ขอรับ เว้ย โวย โว้ยและบรรดาคำรับต่างๆ

ฌ) **ประติเษธวิเศษณ์** คือคำวิเศษณ์ที่ประกอบบอกความปฏิเสธ หรือไม่ยอมรับ เช่น ไม่ มิ ไม่ใช่ มิใช่ บ บ่ บมิ ฤา มิได้ ไม่ได้ หาไม่ หามิได้ ฯลฯ

ญ) **ประพันธวิเศษณ์** คือคำวิเศษณ์ที่ประกอบบอกคำกริยาหรือคำวิเศษณ์ เพื่อเชื่อมประโยคให้มีข้อความเกี่ยวข้องกัน เช่น ที่ ซึ่ง อัน อย่างที่ ชนิดที่ ให้ ว่า ที่ว่า คือ เพื่อ ให้ เพื่อว่า ซึ่งมีคำบางคำเหมือนกันกับประพันธสรรพนาม เช่น ที่ ซึ่ง อัน ซึ่งข้อแตกต่างระหว่างประพันธสรรพนามกับประพันธวิเศษณ์ดังตารางที่ 3.1

ตารางที่ 3.1 เป็นการเปรียบเทียบระหว่าง ประพันธสรรพนาม กับ ประพันธวิเศษณ์

ประพันธสรรพนาม	ประพันธวิเศษณ์
1. ใช้แทนนามหรือสรรพนามที่อยู่ข้างหน้า 2. ต้องเรียงอยู่ติดกับนาม หรือสรรพนามที่มันแทน	1. ประกอบคำกริยา หรือคำวิเศษณ์ 2. ต้องเรียงอยู่ข้างหลังคำอื่น นอกจากคำนามและสรรพนาม

223.2.5 คำบุพบท

คำบุพบท คือคำที่แสดงความสัมพันธ์ระหว่างคำหรือประโยค เพื่อให้รู้ว่าคำหรือประโยคที่อยู่หลังคำบุพบทนั้นมีหน้าที่เกี่ยวข้องกับคำ หรือประโยคที่อยู่ข้างหน้าอย่างไร ซึ่งหน้าที่ของคำบุพบทมีดังนี้คือ

ก) นำหน้านาม เช่น

- หนังสือ <u>ของ</u> บิดาหาย
- เขาไป <u>กับ</u> เพื่อน
- นักเรียนอยู่ <u>ใน</u> ห้อง

ข) นำหน้าสรรพนาม เช่น

- ปากกา <u>ของ</u> ฉันอยู่ <u>ที่</u> เขา
- ฉันคิด <u>ถึง</u> ท่านมาก
- ฉันจะอยู่ <u>ใกล้</u> เธอ

ค) นำหน้ากริยา เช่น

- เขากิน เพื่อ อยู่
- เขาเก็บอาหารไว้ สำหรับ รับประทาน
- เขาทำงาน กระทั่ง ตาย

ง) นำหน้าคำวิเศษณ์ เช่น

- เขาทำ สิ้น ดี
- ถ้าพูดกัน ตาม จริงแล้ว เขาต้องมาหาฉัน โดย เร็ว

จ) นำหน้าประโยค เช่น

- เขายืนใกล้ กับ นายดำทำงาน
- เขามา ตั้งแต่ ฉันตื่นนอน
- เขาให้รางวัล เฉพาะ คนแต่งโคลงได้ที่หนึ่ง

จากหน้าที่ของคำบุพบท เราจะแบ่งคำบุพบทได้เป็น 5 ชนิดดังต่อไปนี้

ก) บุพบทที่แสดงความสัมพันธ์ระหว่าง นามกับนาม นามกับสรรพนาม หรือนามกับคำกริยา เช่น

- ฉันชอบนาฬิกาของนายแดง
- โต๊ะในห้องเรียนมีสามสิบห้าตัว
- มือของเขาสวยมาก

ข) บุพบทที่แสดงความสัมพันธ์ระหว่างสรรพนามกับนาม สรรพนาม กับสรรพนาม หรือ สรรพนามกับกริยา เช่น

- ท่านจะต้องการอะไรที่นายแดง
- เธอพบใครในห้องนั้น

ค) บุพบทที่แสดงความสัมพันธ์ระหว่างกริยากับนาม กริยากับสรรพนาม กริยากับกริยา หรือกริยาวิเศษณ์ บุพบทชนิดนี้อาจนำหน้าบทกรรม บทที่เป็นเจ้าของหรือบทขยายก็ได้ เช่น

- เขาเห็นแก่หน้ากัน
- เขาอยู่ในบ้าน
- ฉันไปกับเธอ
- เขาไปโดยเร็ว

ง) บุพบทที่ไม่แสดงความสัมพันธ์กับบทอื่น ได้แก่คำ ดูกร ดูก่อน ดูแน่ะ ดูรา ข้าแต่ ยัง

จ) บุพบทที่แสดงความสัมพันธ์ระหว่างประโยคกับประโยค เช่น

- เขานอนตั้งแต่ฝนตกครั้งแรก
- เขาเก็บอาหารไว้สำหรับคนดายหญ้าที่สนาม

3.2.6 คำสันธาน

คำสันธาน คือคำที่ทำหน้าที่เชื่อมคำกับคำ ประโยคกับประโยค ข้อความกับข้อความ หรือเชื่อมความให้สละสลวย ชนิดของคำสันธานแบ่งย่อยออกเป็น 11 ชนิด

ก) เชื่อมความที่คล้อยตามกัน เช่น และ พอ ถ้า ก็ กับ จึง เช่น ว่า ให้ คือ ทั้ง ก็คือ ก็ดี ก็ได้ เท่ากับ ฯลฯ เช่น

- เขาพบครู และ นักเรียน
- บิดามารดา และบุตรไปเที่ยว
- นายดำ กับ นายแดงเป็นเพื่อนกัน

ช) เชื่อมความที่ขัดแย้งกัน เช่น แต่ แต่ว่า แต่ทว่า กว่า...ก็ ถึง...ก็ ฯลฯ เช่น

- เขาชอบแกงเผ็ด แต่ ฉันชอบแกงจืด

ค) เชื่อมความที่เป็นเหตุเป็นผลกัน เช่น เพราะ ด้วย จึง ฉะนั้น ฉะนี้ ค่าที่ ด้วยว่า เหตุเพราะ เหตุว่า เพราะว่า ฉะนั้น...จึง เพราะฉะนั้น เหตุฉะนี้ ฯลฯ เช่น

- เขาเรียนหนังสือเก่ง เพราะ เขาอ่านหนังสือมาก

ง) เชื่อมความที่เลือกเอา เช่น หรือ มิฉะนั้น ไม่ก็ ไม่เช่นนั้น มิฉะนั้น หรือมิฉะนั้น

- เธอจะไป หรือ ไม่ไป

จ) เชื่อมความแสดงลักษณะอาการ เช่น ให้ ว่า ทั้ง

- เขาพูด ว่า นายแดงเป็นทหาร

ฉ) เชื่อมความแสดงประมาณ เช่น ตลอดจน จน

- ฉันรอ จน คนเหล่านั้นกลับหมด

ช) เชื่อมความแสดงเวลา เช่น จน เมื่อ

- เก็บอาหารไว้ จน ปีหนึ่งผ่านไป
- เขานอน เมื่อ นาฬิกาตี 11 ที

ช) เชื่อมความแสดงเหตุ เช่น เพราะ เพราะว่า

- น้ำท่วม <u>เพราะ</u> ฝนตก
- ต้นไม้โค่น <u>เพราะว่า</u> โดนคนตัด

ฌ) เชื่อมความแสดงผล เช่น จน

- เขา ดุ จนลูกร้องไห้
- ลมพัด <u>จน</u> หน้าต่างเปิด

ญ) เชื่อมความแสดงเปรียบเทียบ เช่น เหมือน ราวกับ กว่า อย่าง

- เขาเดินเร็ว <u>เหมือน</u> ม้าวิ่ง
- เขาทำได้ดี <u>กว่า</u> ผมทำ

ฎ) เชื่อมความให้สละสลวย เช่น อัน อันว่า อย่างไรก็ตาม อย่างไรก็ดี ก็ ถึงกระนั้นก็ดี

คำสันธาน จะเรียงไว้หน้าคำ ข้างหน้าประโยค ข้างหลังคำ ข้างหลังประโยค หรือในระหว่างประโยคก็ได้ดังตารางที่ 3.2

ตารางที่ 3.2 ตารางการใช้คำสันธาน

ตำแหน่ง	ตัวอย่าง
หน้าคำ	ฉันเห็นนายดำ <u>และ</u> นายแดง

ตารางที่ 3.2 (ต่อ)

ตำแหน่ง	ตัวอย่าง
	ท่านชอบข้าวสวย หรือ ข้าวต้ม
หน้าประโยค	อย่างไรก็ตาม ผมจะมาก่อน 8.00 น. เสมอ อัน ความดีนั้น ใครๆก็ต้องการ
หลังคำ	คน ก็ตาม สัตว์ ก็ตาม ต้องการอาหารทั้งนั้น เสื้อ ก็ดี กางเกง ก็ดี เราควรเก็บให้เป็นที่
หลังประโยค	เขาจะเป็นใคร ก็ตาม ฉันยินดีรับรองเขาทั้งนั้น
ระหว่างประโยค	เขาเดินมา แต่ ฉันเดินไป เขามาลำบาก เพราะว่า ฝนตกหนัก

3.2.7 คำอุทาน

คำอุทาน คือคำที่แสดงถึงเสียงที่เปล่งออกมาในเวลาดีใจ เสียใจ ตกใจ ประหลาดใจ หรือกริ่งใจ เป็นต้น หรือเป็นคำที่ใช้ต่อถ้อยเสริมบทให้บริบูรณ์ยิ่งขึ้น คำอุทานแบ่งออกเป็น 2 ชนิดคือ

ก) อุทานบอกอาการ สำหรับ

1) ใช้แสดงความรู้สึกต่างๆในการพูดเช่นคำ ชิ ชิๆ ชิชะ โธ่ บ๊ะ วะ วา

อื้อหือ เหม่ แหม อนิจจัง อ๊ะ อือ อนิจา อุบ๊ะ เอ เอ๊ะ เอ๊ว เอ้อ เฮอ โอ โอย โอ๊ย ฮะ ฮ้า ฮิ เฮ้ เฮ้ย เฮ้ว เฮ้อ ไฮ้ เป็นต้น

2) **ใช้เป็นคำขึ้นต้นประโยคในคำประพันธ์** เพื่อแสดงความรำพึง วิงวอน หรือปลอบโยน เป็นต้นได้แก่คำ อ้า โอ้ โอ้ว่า

ข) **อุทานเสริมบท** คือคำอุทานที่ใช้เป็นคำสร้อยหรือคำเสริมบทต่างๆ เพื่อให้มีคำครบถ้วนตามที่ต้องการ หรือให้มีความกระชับ หรือให้สละสลวยขึ้น มี 3 ชนิดคือ

1) **อุทานเสริมบทที่ใช้เป็นคำสร้อย** คือคำอุทานใช้เป็นสร้อยของโคลงและร่าย หรือใช้เป็นคำลงท้ายในคำประพันธ์ แสดงว่าจบข้อความบริบูรณ์แล้ว เช่น อ้า เฮย แฮ แล นารา เอย ฯลฯ

2) **อุทานเสริมบทที่ใช้เป็นคำแทรก** คือคำอุทานที่ใช้แทรกในระหว่างคำหรือข้อความ มี 2 ชนิดคือ

2.1) ใช้เป็นบทบูรณ์ คือเป็นคำที่ทำให้บทประพันธ์มีพยางค์ครบถ้วนตามฉันทลักษณ์ได้แก่คำ นุ ชิ สิ นิ เป็นต้น

2.2) ใช้ประกอบข้างท้ายให้มีความกระชับหรือสละสลวยขึ้นได้แก่คำ นา เอย เอ่ย เอ๋ย โวย ฯลฯ เช่น

3) **อุทานเสริมบทที่ใช้เป็นคำเสริม** คือคำอุทานที่ใช้ต่อถ้อยเสริมคำให้ยาวเยินออกไป แต่ไม่ต้องการความหมายที่เสริมนั้น เช่น นาง วา สารา รี กระเดี้ยว ฯลฯ

เมื่อทราบถึงชนิดของหน่วยคำที่มีอยู่ในภาษาไทยว่าเป็นอย่างไรบ้างแล้ว สิ่งที่ต้องรู้ต่อไปก็คือความสัมพันธ์ของคำต่างๆ เมื่อประกอบกันเป็นประโยค โดยคำชนิดใดบ้างควรจะอยู่ตำแหน่งใดได้บ้างในประโยค ซึ่งจะกล่าวโดยละเอียดในหัวข้อต่อไป

3.3 สัญลักษณ์ของกลุ่มคำภาษาไทย

ประโยคภาษาไทยมีการกำหนดตำแหน่งของคำที่ปรากฏในประโยคพื้นฐาน (Simple Sentence) ทั่วๆไป ซึ่งมักประกอบด้วยส่วนของนามวลี(Noun Phrase : NP) กับกริยาวลี(Verb Phrase : VP) ซึ่งบางส่วนอาจถูกละไว้เช่น วิเศษณ์วลี(Adverbial Phrase : ADVP) เป็นต้น ที่มักจะปรากฏในประโยคที่มีโครงสร้างซับซ้อน ซึ่งเกิดจากการนำประโยคธรรมดามาเรียงต่อกัน ทำให้ประโยคในภาษาไทย มีได้หลายรูปแบบของประโยค โดยอักษรย่อของคำแต่ละคำในบทนี้จะมีความหมายตามตารางที่ 3.3

ตารางที่ 3.3 แสดงสัญลักษณ์ย่อชนิดของคำ

คำย่อ	ชนิด	ความหมาย
ADJ	Adjective	คำคุณศัพท์
ADVP	Adverb Phrase	คำวิเศษณ์วลี
AUX	Auxiliary Verb	คำกริยาช่วย
CL	Classifier	คำลักษณะนาม
CONJ	Conjunction	คำสันธาน
CPD	Compound Sentence	ประโยคประสม

ตารางที่ 3.3 (ต่อ)

คำย่อ	ชนิด	ความหมาย
CPX	Complex Sentence	ประโยคซับซ้อน
DET	Determinative	คำกำหนด
G.M.	Genitive Marker	"ของ"
LINK	Linker	คำเชื่อม
MARK	Marker	คำหมาย
N	Noun	คำนาม
NP	Noun Phrase	นามวลี
NUM	Numeral	คำนับ
NEG	Negator	คำปฏิเสธ
OPT	Optional Transformation	คำให้เลือก
PCL	Particle	คำลงท้าย
PREP	Preposition	คำบุพบท
PROH.NEG	Prohibitive Negator	คำปฏิเสธห้าม
REL.LINK	Relative Linker	คำเชื่อมประโยค
S	Sentence	ประโยค
V	Verb	คำกริยา
VP	Verb Phrase	กริยาวลี

3.4 โครงสร้างของวลีในภาษาไทย

ในประโยคธรรมดาทั่วๆไปอาจจะประกอบด้วยนามวลีเพียงอย่างเดียว หรือกริยาวลีเพียงอย่างเดียวก็ได้ ซึ่งเราก็จะเรียกเป็นประโยคเหมือนกัน แต่โดยทั่วไปแล้วประโยคภาษาไทยจะมีโครงสร้างพื้นฐานที่ประกอบเรียงตามลำดับได้ดังนี้

Sentence ——> (ADVP) NP (ADVP) + VP (ADVP)

แสดงโครงสร้างพื้นฐานของประโยคของภาษาไทย ซึ่งสามารถที่จะยกตัวอย่างตามโครงสร้างพื้นฐานนี้ได้เช่น

(ADVP)	NP	(ADVP)	VP	(ADVP)
วานนี้	ฝน	ที่เชียงใหม่	ตก	หนัก
วานนี้	ฝน		ตก	หนักที่เชียงใหม่
ที่	เชียงใหม่		ฝนตก	หนักวานนี้

แต่ประโยคที่ง่ายที่สุดของภาษาไทยนั้น ในส่วนของนามวลีจะประกอบด้วยคำนาม (Noun : N) เพียงคำเดียว และในส่วนของกริยาวลีจะประกอบไปปด้วยคำกริยา(Verb : V) เพียงคำเดียวเท่านั้นไม่มีส่วนขยายอื่นมาเพิ่มเติมก็ได้ดังนี้

NP ——————> N

VP ——————> V

สำหรับวิเศษณ์วลีนั้น ส่วนมากมักจะได้แก่ เวลาและสถานที่ ซึ่งจะอยู่ในตำแหน่งต่างๆของประโยคได้ โดยสามารถที่จะอยู่หน้าหรือหลังนามวลี ก็มีความหมายเหมือนกับวางอยู่หลังกริยาวลีดังตัวอย่างต่อไปนี้

N	V
รถ	หยุด
ฝน	ตก
เขา	นอน
เขา	เดิน

เมื่อเราทราบถึงโครงสร้างของประโยคอย่างคร่าวๆแล้ว ต่อไปจะกล่าวถึงส่วนประกอบโดยละเอียดของ นามวลี กริยาวลี และ วิเศษณ์วลี

3.4.1 นามวลีและส่วนขยาย

ในส่วนของนามวลีที่มีโครงสร้างง่ายที่สุดจะประกอบด้วยคำนามเพียงตัวเดียว จะไม่มีส่วนขยายใดๆจนกระทั่งโครงสร้างที่ยากๆในประโยคผสม(Compound or Complex Sentence) นั้นสามารถที่จะสรุปออกมาเป็นรูปแบบต่างๆ ได้ดังนี้

ก) เป็นนามวลีที่ประกอบไปด้วยคำนาม และตามด้วยคำลักษณะนาม (Classifier) หรือคำคุณศัพท์(Adjective) หรือคำลักษณะนาม หรือคำกำหนด (Determinative : DET)

(หมายเหตุ คำว่า "และ" หมายความว่า จะต้องมีคำชนิดนั้นๆในประโยคไม่มีไม่ได้
"หรือ" หมายความว่า จะมีหรือไม่มีคำชนิดนั้นๆในประโยคก็ได้)

NP ----------> N + (CL) (ADJ) (CL) (DET)

(หมายเหตุ ชนิดของคำในเครื่องหมายวงเล็บจะมีหรือไม่มีก็ได้)

จากรูปแบบของประโยคนี้สามารถที่จะยกตัวอย่างประโยคได้ดังนี้

N	(CL)	(ADJ)	(CL)	(DET)
หนังสือ				
หนังสือ	เล่ม			นี้
ไก่		ดำ		
ไก่	ตัว	ดำ		
ไก่	ตัว	ดำ		นี้
นก	ตัว	ใหญ่	ตัว	นี้

ข) นามวลีที่ประกอบด้วย นามวลี และหรือ "ของ" และ นามวลี และหรือ "ของ" และ นามวลี และหรือ คำลักษณะนาม หรือ คำกำหนด

NP ————————> NP + (G.M.) NP + (G.M.) NP + (CL) (DET)

ตัวอย่างเช่น

NP	(G.M.)	NP	(G.M.)	NP	+ (CL)	(DET)
บ้าน		เพื่อน				
บ้าน			ของ	เพื่อน		
บ้าน	ของ	เพื่อน	ของ	ผม		
แมว	ของ	ป้า	ของ	ผม	ตัว	นี้

ค) นามวลีที่ประกอบด้วย คำนาม และหรือ คำลักษณะนาม และคำคุณศัพท์ และหรือ "ของ" และนามวลี

NP ----------> N (CL) + ADJ (G.M.) + NP

ตัวอย่างเช่น

N	(CL)	ADJ	(G.M.)	NP
เสื้อ	ตัว	ใหม่		
เสื้อ		สีดำ	ของ	แม่
เสื้อ	ตัว	ใหม่	ของ	แม่
รองเท้า	คู่	เก่า	ของ	พ่อ
นาฬิกา	เรือน	ใหม่	ของ	ผม

ง) นามวลีที่ประกอบด้วย คำนาม และหรือ คำคุณศัพท์ และ คำนับ และ คำลักษณะนาม และหรือ คำกำหนด

NP ----------> N (ADJ) + NUM + CL (DET)

ตัวอย่างเช่น

N	(ADJ)	NUM	CL	(DET)
คน		สอง	คน	
คน	แก่	สอง	คน	
บ้าน	ใหญ่	สอง	หลัง	นั้น
คน	หลาย	คน		
คน	บาง	คน		

จ) นามวลีที่ประกอบด้วย คำนาม และหรือ คำคุณศัพท์ และ คำลักษณะนาม และ คำนับ และหรือ คำกำหนด

NP ----------> N (ADJ) + CL + NUM (DET)

ตัวอย่างเช่น

N	(ADJ)	CL	NUM	(DET)
บ้าน		หลัง	เดียว	
คน		คน	เดียว	
บ้าน	ใหญ่	หลัง	เดียว	
บ้าน	ใหญ่	หลัง	หนึ่ง	
บ้าน	ใหญ่	หลัง	ที่หนึ่ง	
บ้าน	ใหญ่	หลัง	ที่สอง	นั้น

ฉ) นามวลีที่ประกอบด้วย คำนาม และ คำเชื่อม และ [คำนาม / คำนับ และหรือ คำลักษณะนาม / คำหมาย]

หรือ คำกำหนด

NP ----------> N + LINK + [N / NUM (CL) / MARK] (DET)

ตัวอย่างเช่น

N	LINK	N / NUM / MARK	(CL)	(DET)
คน	ทั้ง	บ้าน		

		N		
N	LINK	NUM	(CL)	(DET)
		MARK		
นก	ทั้ง		ฝูง	
แมว	ทั้ง	สอง	ตัว	นั้น
คน	ทั้ง	สอง		
ห้อง	ทั้ง	สาม	ห้อง	

ช) นามวลีที่ประกอบด้วย คำหมาย และ คำนาม หรือ คำสรรพนาม และหรือ คำหมาย

NP ----------> MARK + N / PRO (MARK)

ตัวอย่างเช่น

MARK	N / PRO	(MARK)
พวก	ครู	
พวก	เรา	
พวก	เรา	ทั้งหลาย
พวก	ตำรวจ	ทั้งหลาย

ซ) นามวลีที่ประกอบด้วย คำนาม และ คำสรรพนาม

NP ----------> N + PRO

ตัวอย่างเช่น

N	PRO
พ่อ	ท่าน
แมว	มัน

ฌ) นามวลีที่ประกอบด้วย คำนาม และ คำเชื่อม และหรือ คำสันธาน และ คำนาม

NP ----------> N + LINK (CONJ) + N

ตัวอย่างเช่น

N	LINK	(CONJ)	N
พ่อ	กะ		แม่
พ่อ	กับ		แม่
โต๊ะ		หรือ	เก้าอี้
สมุด		และ	หนังสือ

ญ) นามวลีที่ประกอบด้วย คำนาม และหรือ คำลักษณะนาม และ คำคุณศัพท์ และ คำลักษณะนาม และหรือ [คำกำหนด / คำนับ]

NP ----------> N (CL) + ADJ + CL [DET / NUM]

ตัวอย่างเช่น

N	(CL)	ADJ	CL	DET	/	NUM
บ้าน		ใหญ่	หลัง	นั้น		
บ้าน	หลัง	ใหญ่	หลัง	นั้น		
บ้าน	หลัง	ใหญ่	หลัง			ที่สอง

ฎ) นามวลีที่ประกอบด้วย คำเชื่อม และ คำนาม และ คำเชื่อม และ คำนาม

NP ----------> LINK + N + LINK + N

ตัวอย่างเช่น

LINK	N	LINK	N
ทั้ง	พ่อ	ทั้ง	แม่
ทั้ง	นก	และ	หนู
ทั้ง	พี่	และ	น้อง

ฎ) นามวลีที่ประกอบด้วย คำนาม และ คำเชื่อม และ คำนาม และหรือ "ของ" และ คำนาม

NP ----------> N + LINK + N + (G.M.) N

ตัวอย่างเช่น

N	LINK	N	(G.M.)	N
สมุด	กับ	ปากกา	ของ	ผม

3.4.2 กริยาวลีและส่วนขยาย

ส่วนประกอบของกริยาวลีที่น้อยที่สุด จะประกอบด้วยคำกริยาเพียงตัวเดียว แต่เมื่อรวมส่วนขยายต่างๆ ทำให้กริยาวลีมีโครงสร้างที่แตกต่างกันออกไป ซึ่งแสดงได้ดังนี้

ก) กริยาวลีที่ประกอบด้วย คำกริยา และหรือ คำนาม/คำสรรพนาม และหรือ คำหมาย และหรือ คำลงท้าย

VP ----------> V + N/PRON (MARK) (PCL)

ตัวอย่างเช่น

V	N / PRON	(MARK)	(PCL)
ทาน	ข้าว		
ให้		แล้ว	
ทั้ง	จดหมาย	แล้ว	ครับ

ข) กริยาวลีที่ประกอบด้วย คำคุณศัพท์/คำกริยา และหรือ "ดี" และ "กว่า" และ นามวลี/กริยาวลี

VP ----------> ADJ/V (ดี) + (กว่า) + NP/VP

ตัวอย่างเช่น

ADJ / V	(ดี)	(กว่า)	NP / VP
	ดี	กว่า	คนนั้น
น้อย		กว่า	ของผม

ADJ / V	(ดี)	(กว่า)	NP / VP
ไป	ดี	กว่า	อยู่

ค) กริยาวลีที่ประกอบด้วย คำกริยา และหรือ คำนาม/คำสรรพนาม และ คำหมาย และนามวลี

VP ---------- > V (N/PRO) + MARK + NP

ตัวอย่างเช่น

V	(N / PRO)	MARK	NP
ทำงาน		เพื่อ	เงิน
กิน	ข้าว	ด้วย	มือ
เห็น		กับ	ตา

ง) กริยาวลีที่ประกอบด้วย คำกริยา และ นามวลี และหรือ คำกริยา และ คำนับ และคำลักษณะนาม

VP ---------- > V + NP (V) + NUM + CL

ตัวอย่างเช่น

V	NP	(V)	NUM	CL
ส่ง	จดหมาย	ไป	สอง	ฉบับ
มี	เงิน	อยู่	ร้อย	บาท
ให้	หนังสือ	ไป	สอง	เล่ม

จ) กริยาวลีที่ประกอบด้วย คำกริยาช่วย และ คำกริยา และหรือ คำนาม/คำสรรพนาม และหรือ คำหมาย และหรือ คำลงท้าย

VP ---------- > AUX + V (N/PRO) (MARK) (PCL)

ตัวอย่างเช่น

AUX	V	(N / PRO)	(MARK)	(PCL)
กำลัง	อ่าน	จดหมาย		
จะ	ไป		แล้ว	นะ

ฉ) กริยาวลีที่ประกอบด้วย คำกริยาช่วย และคำกริยา และหรือ คำบุรพบท และ นามวลี และหรือ คำหมาย และ คำกริยา และหรือ คำนาม/คำสรรพนาม

VP ----------> (AUX) V + (PREP) NP + (MARK) V (N/PRO)

ตัวอย่างเช่น

(AUX)	V	(PREP)	NP	(MARK)	V	(N/PRO)
จะ	ไป	ที่	ตลาด	เพื่อ	ซื้อ	ปลา
จะ	ไป	ที่	บ้าน	เพื่อ	ทาน	ข้าว

ช) กริยาวลีที่ประกอบด้วย คำกริยาช่วย และ คำกริยา และหรือ คำนาม/คำสรรพนาม และ คำกริยา และหรือ คำบุรพบท และ คำนาม

VP ----------> (AUX) V (N/PRO) + V + (PREP) N/PRO

ตัวอย่างเช่น

(AUX)	V	(N / PRO)	V	(PREP)	N / PRO
จะ	เอา	เงิน	ให้		ผม
	ซื้อ	ปากกา	ให้		น้อง

ซ) กริยาวลีที่ประกอบด้วย คำกริยาช่วย และ คำกริยา และหรือ นามวลี และ คำเชื่อม และ นามวลี

VP ----------> (AUX) V (NP) + LINK + NP

ตัวอย่างเช่น

(AUX)	V	(NP)	LINK	NP
จะ	ไป	วัด	กับ	แม่
	ซื้อ	ผัก	กับ	หมู
จะ	ฟัง	วิทยุ	กับ	น้อง

ฌ) กริยาวลีที่ประกอบด้วย คำกริยาช่วย และ คำกริยา และหรือ นามวลี และ คำเชื่อม และ ประโยค

VP ----------> (AUX) V (NP) + LINK + S

ตัวอย่างเช่น

(AUX)	V	(NP)	LINK	S
จะ	ทาน	ข้าว	แล้ว	ไปดูหนัง

ญ) กริยาวลีที่ประกอบด้วย กริยาช่วย และ คำกริยา และหรือ นามวลี และ คำกริยา และหรือ บุพบทวลี และหรือ นามวลี และหรือ คำหมาย

VP ----------> (AUX) V (NP) + V (PREP P) (NP) (MARK)

ตัวอย่างเช่น

(AUX)	V	(NP)	V	(PREP)	(NP)	(MARK)
เคย	เขียน	จดหมาย	ไป	ที่	เขา	แล้ว
จะ	พา	เขา	ไป	ที่	บ้านพรุ่งนี้	

ฎ) กริยาวลีที่ประกอบด้วย คำกริยา และหรือ คำบุพบท และ คำนาม และหรือ "ของ" และ นามวลี

VP ----------> V + (PREP) + (NP) + (G.M.) (NP)

ตัวอย่างเช่น

V	(PREP)	(NP)	(G.M.)	(NP)
สนใจ	ใน	การพูด	ของ	เขา
เข้าใจ	ใน	การทำงาน	ของ	กรรมกร

3.4.3 วิเศษณ์วลีและส่วนขยาย

ในส่วนของวิเศษณ์วลีนี้ จะเป็นกลุ่มคำที่แสดงความสัมพันธ์ระหว่างคำ หรือประโยค โดยจะบ่งบอกเกี่ยวกับ สถานที่ ลักษณะ กาลเวลา ซึ่งคำเหล่านี้สามารถที่จะอยู่ต้น หรือท้ายประโยคก็ได้ และ

ในบางครั้งก็อาจจะปรากฏอยู่กลางประโยคก็ได้ โดยไม่ทำให้ความหมายโดยรวมเปลี่ยนไป ในที่นี้กำหนดให้คำเหล่านี้เป็นคำวิเศษณ์ในภาษาไทยทั้งหมด ซึ่งมีรูปแบบที่ปรากฏในภาษาได้ดังนี้

ก) วิเศษณ์วลีที่ประกอบด้วย คำวิเศษณ์ และ คำกำหนด

ADVP ---------- > ADV + DET

ตัวอย่างเช่น

ADV	DET
เร็ว	นี้

ข) วิเศษณ์วลีที่ประกอบด้วย คำบุพบท และ นามวลี และหรือ คำกำหนด

ADVP ---------- > PREP + NP (DET)

ตัวอย่างเช่น

PREP	NP	(DET)
ใน	บ้าน	
ที่	โรงเรียน	
ใน	วันนี้	
โดย	เครื่องบิน	
ใน	เดือน	นี้

ค) วิเศษณ์วลีที่ประกอบด้วย คำปฏิเสธ และ คำคุณศัพท์ หรือ คำวิเศษณ์ หรือ คำกริยา

ADVP ---------- > NEG + ADJ / ADV / V

ตัวอย่างเช่น

NEG	ADJ	/	ADV	/	V
ไม่	ดี				
ไม่	สวย				
ไม่	สะดวก				
ไม่	เลว				

ง) วิเศษณ์วลีที่ประกอบด้วย คำวิเศษณ์ และ คำวิเศษณ์ และหรือ คำวิเศษณ์

ADVP ---------- > ADV + ADV (ADV)

ตัวอย่างเช่น

ADV	ADV	(ADV)
โดยปรกติ	ตอนนี้	คนแยะ
ตอนนี้	คนแยะ	โดยปรกติ

3.5 บทสรุป

ที่กล่าวมาแล้วทั้งหมด เป็นความรู้พื้นฐานทางภาษาศาสตร์ของไวยากรณ์เกี่ยวกับโครงสร้างภาษาไทยที่งานวิจัยนี้ได้รวบรวมขึ้นในการสร้างเป็นฐานความรู้ทางด้านไวยากรณ์เพื่อใช้ในการวิเคราะห์โครงสร้างประโยคภาษาไทย โดยเราได้สร้างฐานความรู้ทางไวยากรณ์ดังกล่าวในรูปของโครงข่าย

ความสัมพันธ์ของหน่วยคำ หรือกลุ่มคำไทย เราเรียกวิธีการนี้ว่า Modified-Augmented Transition Network (M-ATN) ซึ่งจะกล่าวโดยละเอียดในบทที่ 4

บทที่ 4

การประมวลผลโครงสร้างประโยคภาษาไทยด้วยระบบ M-ATN

4.1 บทนำ

ในการวิเคราะห์โครงสร้างประโยคภาษาไทยด้วยระบบคอมพิวเตอร์ของงานวิจัยนี้ ในขั้นแรกเราจะต้องทำการประมวลผลเพื่อแยกแยะหน่วยคำ (Word Segmentation) จากประโยคออกเป็นหน่วยคำ (Morpheme) ที่มีความหมายตามพจนานุกรมไทยด้วยระบบ Fast Word Matching เสียก่อน เหตุผลที่ต้องมีการแยกแยะหน่วยคำ เพราะประโยคภาษาไทยจะไม่มีการเว้นช่องว่างระหว่างคำเช่นเดียวกับภาษาอังกฤษดังที่กล่าวมาแล้วในบทก่อนๆ ดังนั้นจึงจำเป็นต้องมีการแยกคำเพื่อที่จะนำไปวิเคราะห์โครงสร้างของประโยคทางด้านไวยากรณ์ (Syntax Analysis) ต่อไป

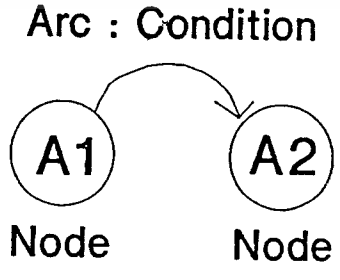
สำหรับการประมวลผลโครงสร้างประโยคของภาษาต่างๆทางด้านการประมวลผลภาษาธรรมชาตินั้นมีหลายวิธี[5] ในวิทยานิพนธ์ฉบับนี้ขอเสนอวิธีการใหม่ที่ได้พัฒนาขึ้นโดยให้ชื่อว่า M-ATN (Modified Augmented Transition Network)[6] ซึ่งเราได้วิจัยและพัฒนาจากระบบ ATN (Augmented Transition Network) ด้วยวิธีการของ M-ATN นี้ เราได้แสดงโครงสร้างทางไวยากรณ์ของประโยคภาษาไทยในรูปของโครงข่ายความสัมพันธ์ (Syntactical Relation Network) ของหน่วยคำไทยด้านนามวลี (Noun Phrase) กริยาวลี (Verb Phrase) และวลีขยาย (Auxiliary Phrase) โดยที่ในแต่ละโครงข่ายความสัมพันธ์จะมีการสร้างโครงข่ายความสัมพันธ์ของหน่วยคำย่อยอย่างละเอียดในรูปของกลุ่มคำต่างๆ โครงข่ายความสัมพันธ์ของหน่วยคำไทยทั้งหมดที่พัฒนาขึ้นในระบบ M-ATN นี้ เมื่อนำมาใช้รวมกันจะครอบคลุมทุกโครงสร้างของประโยคภาษาไทย ทั้งนี้เพราะเราได้ใช้ฐานข้อมูลทางด้านภาษาศาสตร์เกี่ยวกับโครงสร้างประโยคภาษาไทย ที่ได้รับการศึกษาวิจัยจากนักภาษาศาสตร์[7] แล้ว ดังที่ได้กล่าวมาทั้งหมดในบทที่ 3 เราได้ใช้โครงข่ายความสัมพันธ์เหล่านี้เป็นฐานความรู้สำหรับโครงสร้างประโยคภาษาไทยทางด้านไวยากรณ์ (Thai Sentence Structure Knowledge Base) ประกอบกับกระบวนการประมวลผลตรวจสอบโครงสร้างประโยคเพื่อการคัดเลือก

ผลลัพธ์จากการแยกแยะหน่วยคำด้วยระบบ Fast Word Matching แล้วจากผลจาการทดลอง เราสามารถตัดสินเลือกผลลัพธ์ที่ถูกต้องได้อย่างแม่นยำ และสำหรับผลลัพธ์ที่มีโครงสร้างไม่ถูกต้องจะถูกตัดทิ้งด้วยระบบการประมวลผลตรวจสอบโครงสร้างประโยคของ M-ATN นี้สามารถที่จะนำไป พัฒนาเป็นระบบการหาความสัมพันธ์ของหน่วยคำไทยในระดับแก่น (Conceptual Relation Structure) หรือระดับลึก (Deep Relation Structure) ซึ่งมีประโยชน์ต่อการแปลภาษาด้วย ระบบคอมพิวเตอร์ที่ใช้ภาษากลางเป็นตัวกลางของการแปลภาษา[8]

ก่อนที่จะเข้าสู่รายละเอียดของระบบ M-ATN ของงานวิจัยนี้ จะขออธิบายถึงระบบการประมวลผลโครงสร้างปะโยคอื่นๆโดยย่อในบทต่อไป ซึ่งระบบเหล่านี้เป็นต้นแบบที่เรานำมาวิจัยพัฒนาทำให้ได้ระบบ M-ATN ขึ้นมาได้แก่ Transition Network (TN), Recursive Transition Network (RTN) และ Augmented Transition Network (ATN) ตามลำดับ

4.2 ส่วนประกอบของโครงข่ายในระบบต่างๆ

โครงข่าย (Network) ที่ใช้ในการตรวจสอบโครงสร้างประโยคของระบบต่างๆ จะประกอบด้วย (Node)และอาร์ค(Arc) โดยโนดแต่ละโนดจะมีชื่อกำกับและจะต้องเป็นชื่อที่ไม่ซ้ำกันโนดเหล่านี้จะถูกเชื่อมต่อกันเป็นโครงข่ายด้วยอาร์ค(หรือกลุ่มของอาร์ค) โดยอาร์คจะมีเงื่อนไข (Condition) ของการเคลื่อนย้ายที่แสดงความสัมพันธ์ทางไวยากรณ์ของโนดทั้งสองกำกับอยู่ดังแสดงในรูปที่ 4.1 ในรูปนี้ A1 และ A2 จะเป็นชื่อของโนด



รูปที่ 4.1 แสดงส่วนประกอบพื้นฐานของโครงข่ายในระบบต่างๆ

ในวิทยานิพนธ์ฉบับนี้จะใช้วงกลมแทนโนด และลูกศรแทนอาร์ค ส่วนการเคลื่อนย้ายจากโนดหนึ่งไปยังอีกโนดหนึ่ง จะใช้เส้นทางเดินของอาร์คเป็นตัวบ่งบอกทิศทางการเคลื่อนที่จากโนดที่อยู่ต้นลูกศรไปยังโนดที่อยู่ปลายลูกศร โดยกำหนดการเคลื่อนย้ายไปตามเงื่อนไขทางไวยากรณ์ที่กำกับไว้

นอกจากนั้นจะกำหนดให้ S เป็นสัญลักษณ์แทนประโยคที่ประกอบด้วยคำที่ถูกแยกแยะได้ทั้งหมด n คำ โดยมีการเรียงลำดับของคำเป็น W_1 W_2 W_3 W_4 ... W_{n-1} W_n นั่นคือ

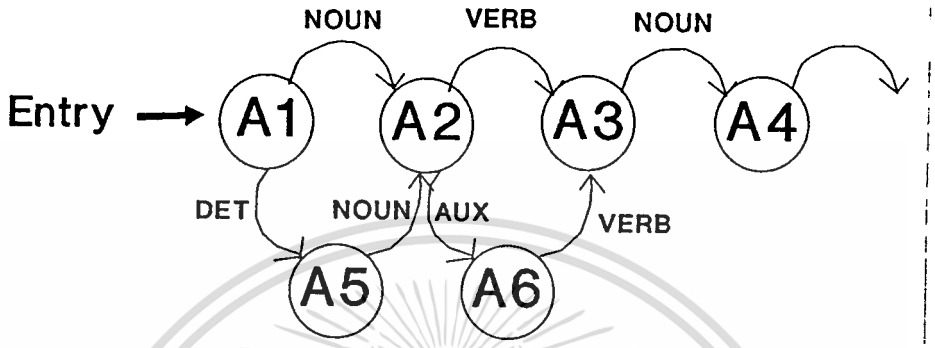
$$S = W_1 \; W_2 \; W_3 \; W_4 \; \ldots \; W_{n-1} \; W_n$$

เมื่อ W_i ($i = 1, 2, 3, \ldots n$: โดย $n > 1$) เป็นคำที่แยกแยะได้จากประโยคด้วยระบบ Fast Word Matching ซึ่งจะมีชนิดของคำกำกับอยู่ ในที่นี้ W_1 จะเป็นคำแรกสุดของประโยค และ W_n จะเป็นคำสุดท้ายของประโยค

4.3 การทำงานของระบบประมวลผลโครงสร้างประโยค : TN, RTN และ ATN[9]

4.3.1 หลักการทำงานพื้นฐานของระบบ Transition Network : TN

ระบบ TN จะประกอบด้วยกลุ่มของโนดที่ถูกเชื่อมด้วยกลุ่มของอาร์ค ซึ่งการเชื่อมต่อเป็นไปตามเงื่อนไขความสัมพันธ์ทางไวยากรณ์ เช่น คำนาม (Noun) คำคุณศัพท์ (Adjective) คำกริยา (Verb) คำบุพบท (Preposition) คำช่วย (Auxiliary) เป็นต้น โดยจะมีการจัดเรียงอาร์คตามหลักไวยากรณ์ภาษา ซึ่งระบบ TN จะสามารถแสดงโครงสร้างของภาษาดังตัวอย่างโครงสร้างง่ายๆในภาษาอังกฤษที่แสดงในรูปที่ 4.2



รูปที่ 4.2 ตัวอย่างแสดงความสัมพันธ์ทางโครงสร้างบางส่วนของภาษาอังกฤษในระบบ TN

เมื่อนำตัวอย่างโครงข่ายรูปที่ 4.2 ของระบบ TN ไปหาชนิดของคำในประโยค คำแรกของประโยคจะถูกเริ่มประมวลผลที่โนดเริ่มต้นทุกครั้ง (ในรูปคือ Entrty node ได้แก่โนด A1 นั่นเอง) จากนั้นระบบก็จะทำการตรวจสอบว่า เงื่อนไขของอาร์คใดตรงกับชนิดของคำตามหลักไวยากรณ์ที่จะประมวลผลตามหลักการทำงานดังตารางที่ 4.1

ตารางที่ 4.1 แสดงกระบวนการทำงานของระบบ TN

ลำดับที่ 1 : เริ่มต้นการประมวลผลที่ Entry โนด โดยเริ่มจากคำแรกสุดของประโยคได้แก่ W_1 (โดย i=1) ซึ่งจะกำหนดให้เป็นคำที่จะประมวลผลคำแรก แล้วเริ่มลำดับที่ 2
ลำดับที่ 2 : ตรวจสอบเงื่อนไขของอาร์คทั้งหมดที่ออกจากโนดนี้ว่ามีเงื่อนไขใดที่ตรงกับชนิดของคำ

ที่จะประมวลผล (W_i) หรือไม่

2.1) ถ้าเงื่อนไขของอาร์คใดตรงตามชนิดของคำที่จะประมวลผลให้เลือกอาร์คนั้นไว้ให้กับคำที่ W_i เลื่อนการประมวลผลไปคำที่ W_{i+1}ของประโยค พร้อมกับกำหนดให้ W_{i+1} เป็นคำที่จะประมวลผลคำใหม่ และเลื่อนการทำงานไปลำดับที่ 3

2.2) ถ้าไม่มีเงื่อนไขของอาร์คใดตรงกับชนิดของคำที่ประมวลผล (W_i)แสดงว่าโครงสร้างทางไวยากรณ์ของประโยคนั้นผิด ให้หยุดการประมวลผลที่ตำแหน่งนั้น

ลำดับที่ 3 : เริ่มต้นการประมวลผลใหม่กับคำต่อมา (W_{i+1}) ที่โนดซึ่งอยู่ต่อจากอาร์คที่เลือกไว้ โดยย้อนการทำงานไปที่ลำดับที่ 2 จนกระทั่งหมดคำที่ประมวลผล(เมื่อ i=n)หรือโนดที่จะเริ่มการประมวลผลจึงจะเลื่อนการทำงานไปลำดับที่ 4

ลำดับที่ 4 : ให้นำเงื่อนไขอาร์คที่เลือกไว้ทั้งหมด มาเรียงกันตามลำดับเพื่อแสดงโครงสร้างทางไวยากรณ์ที่ถูกต้องของประโยคนั้น

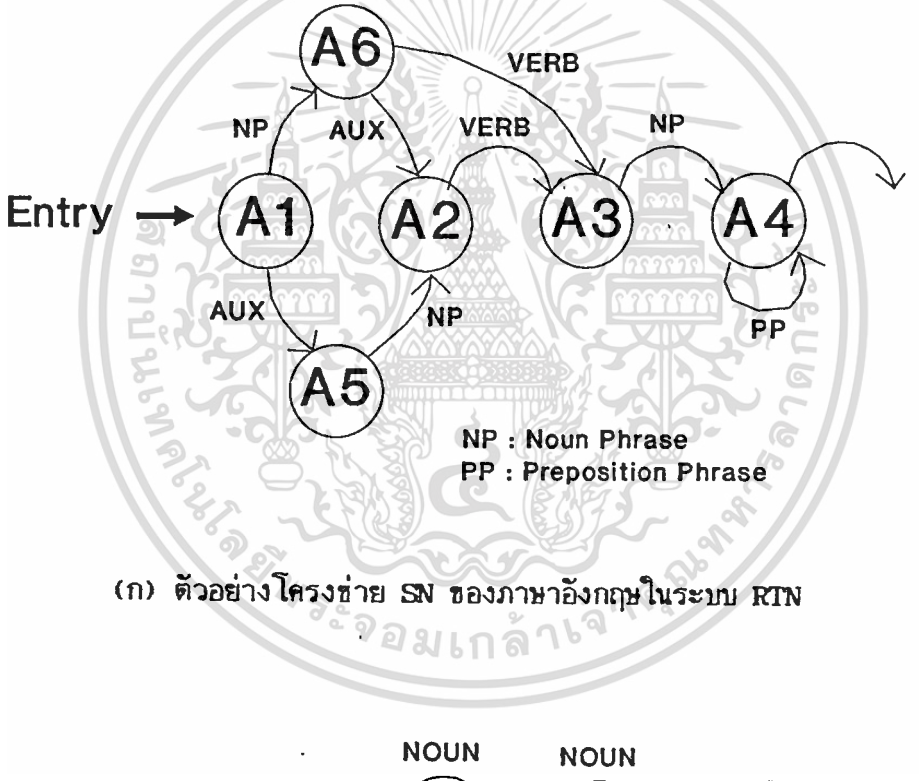
จากตัวอย่างการตรวจสอบโครงสร้างประโยคด้วยระบบ TN จะพบว่ามีความจำเป็นที่จะต้องกระจายโครงข่ายความสัมพันธ์ทางไวยากรณ์ให้ครอบคลุมทุกๆโครงสร้างของประโยคในภาษานั้นๆ นั่นคือ เงื่อนไขความสัมพันธ์ทางไวยากรณ์จะเป็นตัวเพิ่มจำนวนอาร์คให้มากขึ้นตามไปด้วย ทำให้โครงข่ายมีขนาดใหญ่และมีความซับซ้อนมากขึ้น ในการแก้ไขปรับปรุงโครงข่ายจะกระทำได้ลำบาก และถ้านำไปประยุกต์ใช้กับระบบคอมพิวเตอร์ ซอฟท์แวร์ที่สร้างขึ้นจะมีขนาดใหญ่และมีขั้นตอนการตรวจสอบที่ล่าช้าได้ นี่เป็นสาเหตุทำให้มีการพัฒนาระบบ RTN ขึ้นมา

4.3.2 หลักการทำงานพื้นฐานของระบบ Recursive Transition Network : RTN

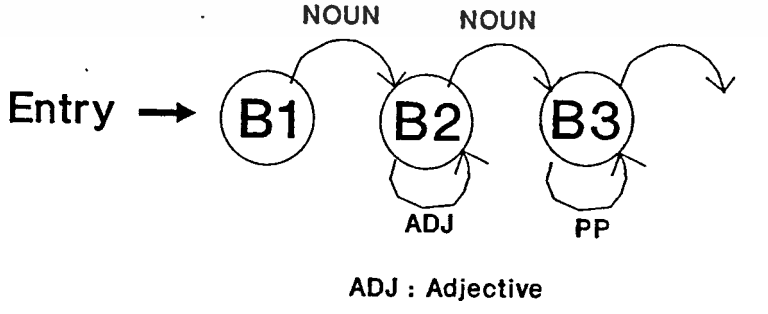
ระบบ RTN มีลักษณะโครงสร้างและหน้าที่การทำงานคล้ายๆกับระบบ TN คือ มีจุดเริ่มต้นการประมวลผลคำเพียงโนดเดียว (Entry Node) และจะมีกี่โนดสิ้นสุดก็ได้ โดยแต่ละโนดจะต้องมีชื่อกำกับและชื่อนั้นจะต้องไม่ซ้ำกัน ส่วนอาร์คจะมีเงื่อนไขที่แสดงชนิดของคำตามไวยากรณ์ในภาษา ส่วนที่แตกต่างจากระบบ TN ได้แก่การแบ่งโครงข่ายทั้งหมดที่มีในระบบออกเป็น 3 ส่วนใหญ่ๆ ซึ่งในระบบ TN

จะมีรวมไว้เพียงโครงข่ายเดียว โครงข่ายทั้ง 3 ส่วนของระบบ RTN ได้แก่

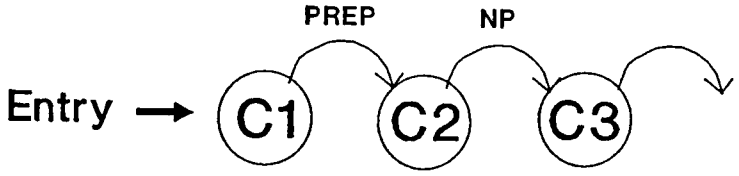
1. Sentence Network [SN] โดยจะทำหน้าที่เป็นโครงข่ายของประโยค ซึ่งเป็นส่วนหลักของโครงข่ายทั้งหมด
2. Noun Phrase Network [NPN] เป็นโครงข่ายที่แสดงหน้าที่เกี่ยวกับคำนามและส่วนขยายนาม
3. Preposition Phrase Network [PPN] เป็นโครงข่ายที่แสดงหน้าที่ของคำบุพบทและส่วนขยายประโยค



(ก) ตัวอย่างโครงข่าย SN ของภาษาอังกฤษในระบบ RTN



(ข) ตัวอย่างโครงข่าย NPN ของภาษาอังกฤษในระบบ RTN



PREP : Preposition

(ค) ตัวอย่างโครงข่าย PPN ของภาษาอังกฤษในระบบ RTN

รูปที่ 4.3 ตัวอย่างการแสดงความสัมพันธ์ทางโครงสร้างบางส่วนของประโยคภาษาอังกฤษในระบบ RTN

การทำงานของระบบ RTN ก็คล้ายกับระบบ TN คือมีการตรวจสอบว่าคำในประโยคที่ทำการประมวลผลนั้นตรงกับเงื่อนไขของอาร์คหรือไม่ โดยจะเริ่มจากโครงข่าย SN ดังตารางที่ 4.2

ตารางที่ 4.2 แสดงกระบวนการทำงานของระบบ RTN

ลำดับที่ 1 : เริ่มต้นการประมวลผลที่ Entry โนดของ SN โดยเริ่มจากคำแรก (W_1 : i=1) ของประโยค ซึ่งกำหนดให้เป็นคำที่จะประมวลผลคำแรก แล้วเริ่มลำดับที่ 2
ลำดับที่ 2 : ตรวจสอบเงื่อนไขของอาร์คทั้งหมดที่ออกจากโนดนี้ว่ามีเงื่อนไขใดที่ตรงกับชนิดของคำ (W_1) ที่ประมวลผลหรือไม่
2.1) ถ้าเงื่อนไขของอาร์คใดตรงตามชนิดของคำที่ประมวลผล W_1 ให้เลือกอาร์คนั้นไว้ให้กับ

คำนั้น แล้วเลื่อนการประมวลผลไปที่คำต่อไปของประโยค (W_{i+1}) พร้อมกับกำหนดให้เป็นคำที่จะประมวลผลคำใหม่ และเลื่อนการทำงานไปลำดับที่ 3

2.2) ถ้าไม่มีเงื่อนไขของอาร์คใดตรงกับชนิดของคำที่ประมวลผล (W_i) ให้ย้อนกลับไปยังโนดก่อนหน้านั้นตามเงื่อนไขของอาร์คที่ข้ามมา และให้นำคำศัพท์ก่อนหน้าคำศัพท์ที่ไม่สามารถประมวลผลได้ (คำที่ W_{i-1}) มาประมวลผลเงื่อนไขของอาร์คใหม่ ที่ตำแหน่งของอาร์คใหม่ แต่หากประมวลผลไม่ได้แล้วย้อนกลับจนกระทั่งไปถึง Entry โนดของ SN (ที่ตำแหน่งคำศัพท์คำแรกของประโยค W_1) แล้วไม่สามารถประมวลผลได้อีก ให้ถือว่าประโยคนั้นผิดไวยากรณ์ให้หยุดการทำงาน

2.3) หากระบบทำการประมวลผลเงื่อนไขของอาร์คที่จะต้องอ้างถึงโครงข่ายอื่น (NPN, PPN) ให้ข้ามไปประมวลผลตามโครงข่ายนั้นๆและประมวลผลเงื่อนไขตามลำดับที่ 2.1 และ 2.2 เมื่อประมวลผลเสร็จสิ้นให้ย้อนกลับไปยังโครงข่ายที่ออกมา

ลำดับที่ 3 : เริ่มต้นการประมวลผลใหม่กับคำต่อไป (W_{i+1}) ที่โนดซึ่งอยู่ต่อจากอาร์คที่เลือกไว้ โดยย้อนการทำงานไปที่ลำดับที่ 2 จนกว่าจะหมดคำที่จะประมวลผล (เมื่อ i=n) หรือโนดที่จะเริ่มการประมวลผลจึงจะเลื่อนการทำงานไปลำดับที่ 4

ลำดับที่ 4 : ให้นำเงื่อนไขของอาร์ค(ชนิดของคำ)ที่เลือกไว้ทั้งหมด มาเรียงกันตามลำดับเพื่อแสดงโครงสร้างทางไวยากรณ์ที่ถูกต้องของประโยคนั้น

ข้อที่แตกต่างของการทำงานในระบบ RTN เมื่อเปรียบเทียบกับระบบ TN ก็คือ เมื่อการตรวจสอบคำมาถึงเงื่อนไขของอาร์คที่แสดงเป็น NP หรือ PP การทำงานของระบบก็จะออกจากโครงข่าย SN ชั่วคราวเพื่อข้ามไปทำการตรวจสอบในส่วนของ NP หรือ PP ในโครงข่าย NPN และ PPN ตามลำดับ เมื่อทำการตรวจสอบเสร็จก็จะกลับคืนสู่โครงข่าย SN ในตำแหน่งอีกข้างหนึ่งของอาร์คแล้วเริ่มการตรวจสอบต่อไป แม้ระบบ RTN จะมีความสามารถในการเพิ่มเติม หรือสร้างโครงข่ายง่าย แต่ กระบวนการทำงานยังมีปัญหา คือเมื่อไปถึงโนดใดๆแล้วไม่มีเงื่อนไขของอาร์คที่ต่ออยู่ในโนดนั้นๆ ตรงกับคำที่นำมาประมวลผล ระบบจะทำการย้อนกลับ (Backtrack) แล้วทำการตรวจสอบเงื่อนไขของอาร์คใหม่ โดยระบบไม่มีความสามารถที่จะตรวจสอบว่าอาร์คใดที่เคยตรวจสอบแล้ว ดังนั้นจึงทำการ

ตรวจสอบใหม่ อีก ทำให้เสียเวลาโดยเปล่าประโยชน์ และระบบ RTN ยังไม่สามารถที่จะใช้กับประโยคที่มีความซับซ้อนมากๆได้ จึงได้มีการพัฒนาระบบ Augmented Transition Network ขึ้นมา

4.3.3 หลักการทำงานพื้นฐานของระบบ Augmented Transition Network : ATN

ATN มีพื้นฐานมาจาก TN(Transition Network)[7] ซึ่งเป็นการประยุกต์ทางคณิตศาสตร์ด้านทฤษฎีกราฟ (Graph Theory) และ Finite State Machine โดยจะแสดงโครงสร้างประโยคของภาษาใดๆในรูปของโครงข่ายได้โดยการจัดกลุ่มของอาร์ค และโนดมาประกอบกันเป็นรูปของโครงข่าย ทั้งนี้ โนดจะแสดงหน่วยคำ และอาร์คจะแสดงความสัมพันธ์ระหว่างหน่วยคำที่เชื่อมติดกัน ความสัมพันธ์ระหว่างหน่วยคำจะเป็นไปตามไวยากรณ์ทางด้านชนิดของคำเมื่อปรากฏในประโยค ข้อมูลทางภาษานี้จะเป็นตัวกำหนดการเชื่อมโยงโนดด้วยอาร์ค และการกำหนดความสัมพันธ์ หรือเงื่อนไขของอาร์คนั้นๆ นอกจากนั้นเมื่อนำโครงข่าย TN ไปประมวลผลโครงสร้างประโยคในภาษาใดๆ จะมีการเพิ่มเงื่อนไขสำหรับการเคลื่อนย้ายโนดและการจดจำ (Register)ว่าขั้นตอนของการประมวลผลอยู่ส่วนใดของประโยค และมีการตรวจสอบเงื่อนไขของอาร์ค ว่าเคยตรวจสอบมาแล้วหรือยัง ซึ่งในระบบ ATN มีอาร์คพื้นฐาน 4 แบบดังนี้

1) CATEGORY Arc(Cat Arc) หมายถึง อาร์คที่สั่งให้มีการประมวลผลตามลำดับที่กำหนด

2) SEEK Arc หมายถึง การให้ข้ามไปทำงานในตำแหน่งของโนดใน Network ที่กำหนดชื่อไว้เมื่อเสร็จ แล้วให้นำผลลัพธ์กลับมาด้วย เพื่อใช้ในการประมวลผลโนดต่อไป

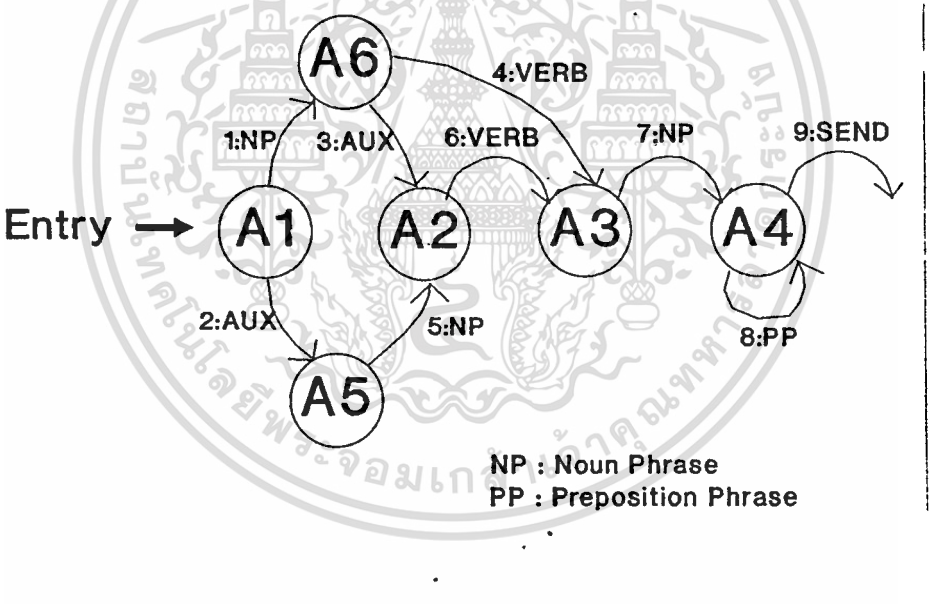
3) JUMP Arc หมายถึง การทำงานของระบบเมื่อมาถึงอาร์คนี้ จะไม่ทำการตรวจสอบเงื่อนไขใดๆทั้งสิ้น ให้ข้ามไปยังโนดต่อไป หรือโนดที่กำหนด

4) SEND Arc หมายถึง อาร์คที่บอกถึงการสิ้นสุดของกระบวนการในโครงข่ายนั้นๆ แต่อาจจะไม่ใช่โนดสุดท้ายก็ได้

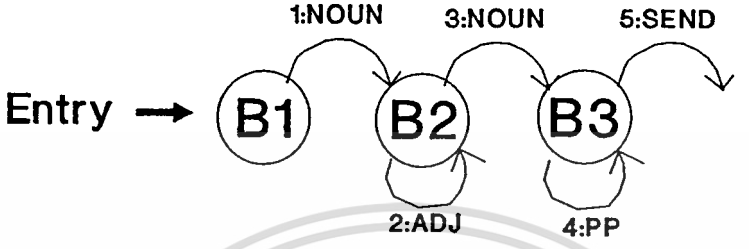
ระบบ ATN ได้แก่การแบ่งโครงข่ายทั้งหมดที่มีในระบบออกเป็น 3 ส่วนใหญ่ๆ ซึ่งโครงข่าย

ทั้ง 3 ส่วนของระบบ ATN ได้แก่

1. Sentence Network [SN] โดยจะทำหน้าที่เป็นโครงข่ายของประโยค ซึ่งเป็นส่วนหลักของโครงข่ายทั้งหมด
2. Noun Phrase Network [NPN] เป็นโครงข่ายที่แสดงหน้าที่เกี่ยวกับคำนามและส่วนขยายนาม
3. Preposition Phrase Network [PPN] เป็นโครงข่ายที่แสดงหน้าที่ของคำบุพบทและส่วนขยายประโยค

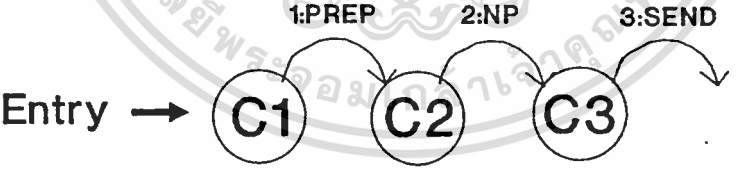


(ก) ตัวอย่างโครงข่าย SN ของภาษาอังกฤษในระบบ ATN



ADJ : Adjective

(ข) ตัวอย่างโครงข่าย NPN ของภาษาอังกฤษในระบบ ATN



PREP : Preposition

(ค) ตัวอย่างโครงข่าย PPN ของภาษาอังกฤษในระบบ ATN

รูปที่ 4.4 ตัวอย่างการแสดงความสัมพันธ์ทางโครงสร้างบางส่วนของประโยคภาษาอังกฤษในระบบ ATN

การทำงานของระบบ ATN ก็คล้ายกับระบบ RTN คือมีการตรวจสอบว่าคำในประโยคที่ทำการประมวลผลนั้นตรงกับเงื่อนไขของอาร์คหรือไม่ โดยจะเริ่มจากโครงข่าย SN ดังนี้

ตารางที่ 4.3 แสดงกระบวนการทำงานของระบบ ATN

ลำดับที่ 1 : เริ่มต้นการประมวลผลที่ Entry โนดของ SN โดยเริ่มจากคำแรก (W_1 : i=1) ของประโยค ซึ่งกำหนดให้เป็นคำที่จะประมวลผลคำแรก แล้วเริ่มลำดับที่ 2

ลำดับที่ 2 : ตรวจสอบเงื่อนไขของอาร์คทั้งหมดที่ออกจากโนดนี้ว่ามีเงื่อนไขใดที่ตรงกับชนิดของคำที่ประมวลผล (W_1) หรือไม่

2.1) ถ้าเงื่อนไขของอาร์คใดตรงตามชนิดของคำที่ประมวลผลให้เลือกอาร์คนั้นไว้ให้กับคำนั้น (W_1) ไว้พร้อมกับเก็บบันทึกลำดับของโนดและลำดับของอาร์คที่เลือกไว้ แล้วเลื่อนการประมวลผลไปที่คำต่อไป (W_{1+1}) ของประโยค แล้วเลื่อนการทำงานไปลำดับที่ 3

2.2) ถ้าไม่มีเงื่อนไขของอาร์คใดตรงกับชนิดของคำที่ประมวลผล (W_1) ให้ย้อนกลับไปยังโนดก่อนหน้านั้นตามเงื่อนไขของอาร์คที่ข้ามมา และให้นำคำศัพท์ที่ W_{1-1} ซึ่งเป็นคำที่อยู่ก่อนคำที่ไม่สามารถประมวลผลได้มาประมวลผลเงื่อนไขของอาร์คใหม่ที่ตำแหน่งของอาร์คใหม่ แต่หากประมวลผลไม่ได้แล้วย้อนกลับจนกระทั่งถึง Entry โนดของ SN (ที่ตำแหน่งคำศัพท์คำแรก W_1 ของประโยค) แล้วไม่สามารถประมวลผลได้อีกให้ถือว่าประโยคที่นำมาประมวลผลนั้นผิดไวยากรณ์ ให้หยุดการทำงาน (เก็บบันทึกลำดับของโนดและลำดับของอาร์คไว้)

2.3) ถ้าประมวลผลเงื่อนไขของอาร์คตรงกับชนิดของคำที่ประมวลผล (W_1) ก่อนที่จะเลือกอาร์คนี้ไว้ให้ตรวจสอบว่าเส้นทางของอาร์คที่เลือกไว้นั้นชี้ไปยังโนดที่เคยเก็บบันทึกไว้ว่า ไม่สามารถประมวลผลคำศัพท์คำต่อไป (W_{1+1}) กับเงื่อนไขของอาร์คที่ต่อออกจากโนดนั้นได้ ให้ประมวลผลเงื่อนไขของอาร์คอื่น (ถ้ามี) และเก็บบันทึกลำดับของโนดและลำดับของอาร์คไว้

2.4) หากระบบทำการประมวลผลเงื่อนไขของอาร์คที่จะต้องอ้างถึงโครงข่ายอื่น (NPN,

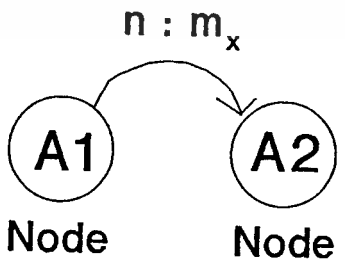
PPN) ให้ข้ามไปประมวลผลตามโครงข่ายนั้นๆ และประมวลผลเงื่อนไขตามลำดับที่ 2 เมื่อประมวลผลเสร็จสิ้นให้ย้อนกลับไปยังโครงข่ายที่ออกมา

ลำดับที่ 3 : เริ่มต้นการประมวลผลใหม่กับคำต่อไป (W_{i+1}) ที่โนดซึ่งอยู่ต่อจากอาร์คที่เลือกไว้ โดยย้อนการทำงานไปที่ลำดับที่ 2 จนกว่าจะหมดคำที่จะประมวลผล (เมื่อ i=n) หรือโนดที่จะเริ่มการประมวลผลจึงจะเลื่อนการทำงานไปลำดับที่ 4

ลำดับที่ 4 : ให้นำเงื่อนไขของอาร์ค(ชนิดของคำ)ที่เลือกไว้ทั้งหมด มาเรียงกันตามลำดับเพื่อแสดงโครงสร้างทางไวยากรณ์ที่ถูกต้องของประโยคนั้น

4.4 หลักการทำงานของระบบ M-ATN ในการประมวลผลโครงสร้างประโยคภาษาไทย[3,4]

ระบบ M-ATN ของงานวิจัยนี้ได้พัฒนาขึ้นมาจากระบบ ATN โดยอาศัยพื้นฐานมาจากทฤษฎีกราฟที่ว่าโครงข่าย จะต้องประกอบด้วยโนด และอาร์คโดยที่โนดแต่ละโนดจะต้องมีชื่อไม่ซ้ำกันและจะถูกเชื่อมต่อกันด้วยอาร์ค หรือกลุ่มของอาร์คกลายเป็นโครงข่าย โดยที่อาร์คแต่ละอาร์คจะมีเงื่อนไขปรากฏอยู่ดังแสดงในรูปที่ 4.5



รูปที่ 4.5 แสดงส่วนประกอบต่าง ๆ ของโครงข่ายในระบบ M-ATN

จากรูป สัญลักษณ์ $n{:}m_x$ มีความหมายดังนี้

n: เป็นตัวเลขบอกลำดับของการตรวจสอบในแต่ละอาร์ค

m_x คือเงื่อนไขทางไวยากรณ์ที่ใช้ในการตรวจสอบ

อาร์คจะมีทิศทางตามลูกศร ดังนั้นการเคลื่อนย้ายจากโนด A ไปยังโนด B จึงประกอบด้วยเงื่อนไขตามที่กำหนด และทิศทางการเคลื่อนย้ายจะเป็นไปตามทิศลูกศรของอาร์ค การประมวลผลโครงสร้างประโยคด้วยวิธี M-ATN นี้เป็นการประมวลผลที่ได้พัฒนาให้มีประสิทธิภาพดียิ่งขึ้นกว่าระบบก่อนๆ และให้เหมาะสมกับภาษาไทย เพราะทำให้สามารถหาชนิดของคำในประโยคได้อย่างถูกต้อง โดยเฉพาะการค้นหากริยาหลัก (Main Predicate) ในประโยคซึ่งเป็นสิ่งจำเป็นอย่างมาก ในการหาโครงสร้างลึกของประโยค เพื่อนำไปสู่การแปลภาษาด้วยเครื่องคอมพิวเตอร์ต่อไป

ในวิทยานิพนธ์นี้เราได้กำหนดหลักการทำงานและโครงสร้างพื้นฐานของระบบ M-ATN ไว้ว่า การประมวลผลคำในประโยคจะตรวจสอบจากซ้ายไปขวา กระบวนการตรวจสอบประกอบด้วย การตรวจสอบเงื่อนไข การย้อนกลับและการจดจำผลของการตรวจสอบ ในระบบ M-ATN จะมีอาร์คทั้งหมด 3 ประเภทคือ

1) Fixed Arc สัญลักษณ์ $n{:}m_1$

ความหมาย m_1 เป็นเงื่อนไขที่ใช้ในการตรวจสอบประกอบด้วย 2 อาร์คคือ JUMP Arc และ SEND Arc

หน้าที่ JUMP Arc ใช้ในการข้ามการตรวจสอบไปยังโนดต่อไปภายหลังจากเสร็จสิ้นการตรวจสอบเงื่อนไขของอาร์คที่มีลำดับมาก่อนหน้านี้แล้ว

SEND Arc เป็นอาร์คที่บอกถึงการเสร็จสิ้นกระบวนการตรวจสอบในแต่ละโครงข่ายย่อยและการตรวจสอบจะย้อนกลับไปยังโครงข่ายก่อนหน้านี้

2) Network Arc สัญลักษณ์ $n{:}[m_2]$

ความหมาย m_2 หมายถึงชื่อโครงข่ายซึ่งในวิทยานิพนธ์ฉบับนี้ได้รวบรวมไว้ทั้งหมด

4 โครงข่าย สำหรับประโยคในภาษาไทยดังจะได้กล่าวรายละเอียดในหัวข้อที่ 4.4.1

หน้าที่ ทำการข้ามการตรวจสอบไปยังโครงข่ายที่ระบุด้วยค่า m_2 และเมื่อเสร็จสิ้นการตรวจสอบในโครงข่าย m_2 แล้วให้ย้อนกลับมาที่โครงข่ายเดิม

3. Category Arc สัญลักษณ์ $n:m_3$

ความหมาย m_3 หมายถึงค่าที่บอกชนิดของคำ(Category)ซึ่งจะเป็นผลลัพธ์สุดท้ายของการตรวจสอบ

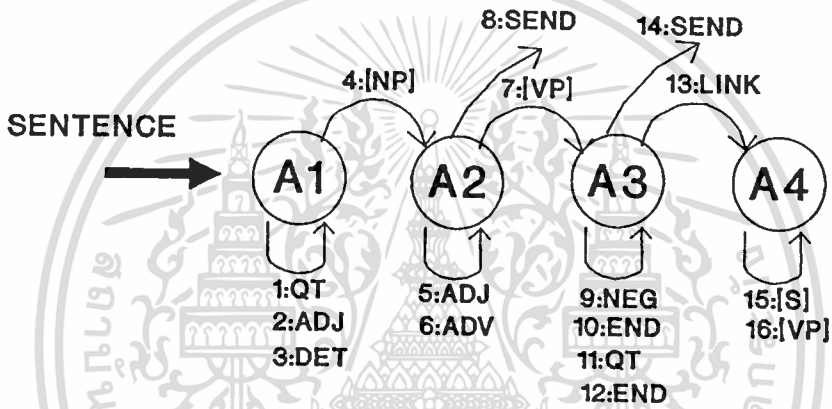
หน้าที่ ทำการตรวจสอบหาชนิดของคำตามหลักไวยากรณ์ภาษาไทยซึ่งได้กำหนดไว้ในหัวข้อ 4.4.1

4.4.1 การกำหนดรูปแบบโครงข่ายของประโยคภาษาไทยในระบบ M-ATN และกระบวนการประมวลผล

ในวิทยานิพนธ์ฉบับนี้ได้รวบรวมรูปแบบโครงสร้างประโยคภาษาไทยเพื่อใช้ในระบบ M-ATN โดยอาศัยฐานข้อมูลทางภาษาศาสตร์[2] ที่ได้รับการวิจัยมาแล้ว และได้ทำการดัดแปลงปรับปรุงให้กระทัดรัดและเหมาะสมกับการแสดงด้วยโครงข่าย และง่ายต่อการพัฒนาระบบซอฟแวร์คอมพิวเตอร์ ซึ่งผลสุดท้ายของการวิจัยสามารถกำหนดได้ทั้งหมด 4 โครงข่าย โดยแต่ละโครงข่ายจะแสดงโครงสร้างของประโยคภาษาไทย ความหมายและชื่อของโครงข่ายทั้ง 4 มีดังต่อไปนี้

1) **โครงข่ายของประโยค (Sentence Network)** [SN] หมายถึง โครงสร้างของประโยคที่ถูกสร้างตามหลักไวยากรณ์ของภาษาไทย ซึ่งเรียกว่า "โครงสร้างหลักของประโยค" มีส่วนประกอบที่สำคัญ 3 ส่วนคือ นามวลี (Noun Phrase) กริยาวลี (Verb Phrase) และ

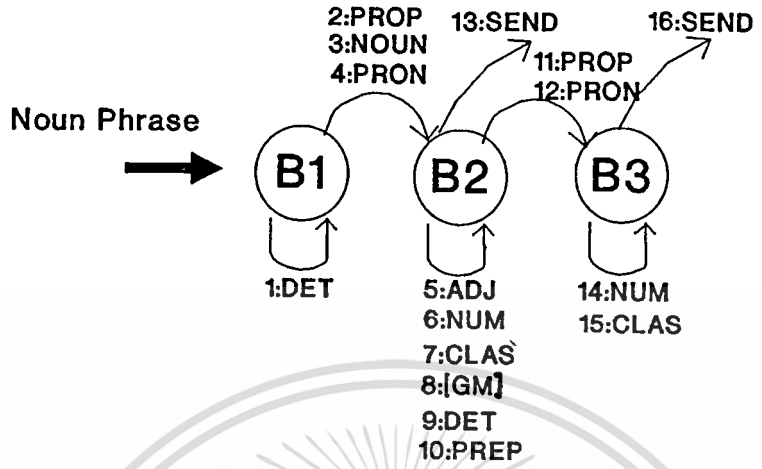
กลุ่มของ(Genitive Marker) นอกจากนี้ยังประกอบด้วยคำคุณศัพท์ (Adjective) คำกำหนด (Determinative) คำปฏิเสธ (Negator) คำแสดงคำถาม (Quationator) คำลงท้าย (Particle) และคำเชื่อม (Linker) ดังแสดงเป็นโครงข่ายรูปที่ 4.6



รูปที่ 4.6 แสดงโครงข่ายของประโยค (Sentence Network : [SN])

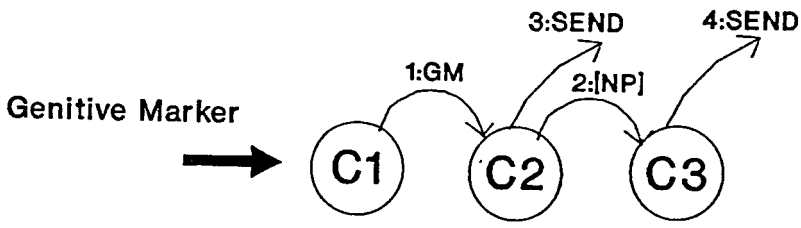
2) โครงข่ายของนามวลี (Noun Phrase) หมายถึง กลุ่มของคำนามและ คำสรรพนาม โดยมีส่วนขยายร่วมอยู่ด้วย ซึ่งเรียกว่า **"นามวลี"** จะประกอบไปด้วยโครงข่ายต่างๆ ได้แก่ คำนาม (Noun) คำสรรพนาม (Pronoun) คำนามเฉพาะ (Propernoun) คำคุณศัพท์ คำบุพบท (Preposition) คำลักษณะนาม (Classifier) คำนับ (Number) กลุ่ม [NP]

ของ และคำกำหนด ดังแสดงในรูปที่ 4.7



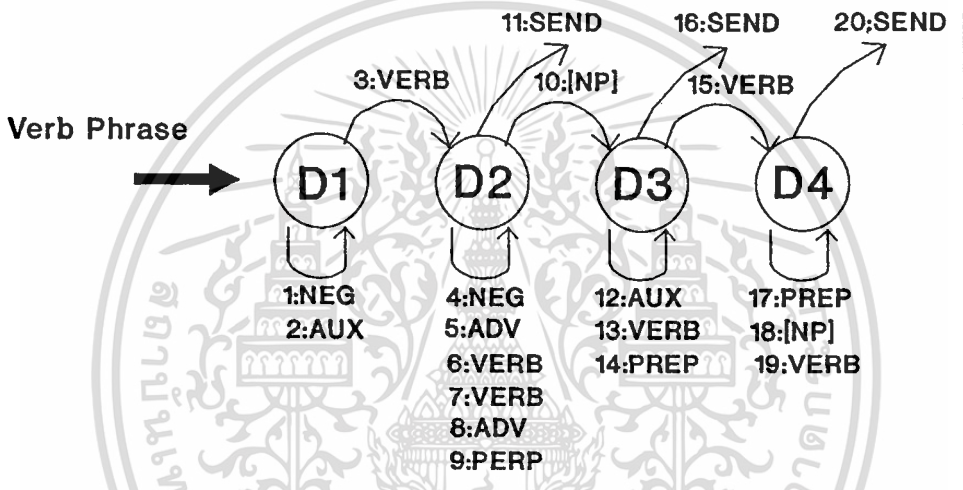
รูปที่ 4.7 แสดงโครงข่ายนามวลี (Noun Phrase : [NP])

3) โครงข่ายของกลุ่มของ (Genitive Marker) เรียกว่า "กลุ่มของ" หมายถึง กลุ่มของคำ [GM] ว่า "ของ" และส่วนขยายคือ นามวลี ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 แสดงโครงข่ายกลุ่มของ (Genitive Marker : [GM])

4) โครงข่ายของกริยาวลี (Verb Phrase) เรียกว่า "กริยาวลี" หมายถึงกลุ่มของคำกริยาและส่วนขยายคำกริยาด้วยซึ่งได้แก่ คำปฏิเสธ คำกริยานุเคราะห์ (Auxilary Verb) คำวิเศษณ์ (Adverb) คำกริยา (Verb) คำบุพบท และ นามวลี ดังแสดงในรูปที่ 4.9 [VP]



รูปที่ 4.9 แสดงโครงข่ายของกริยาวลี (Verb Phrase : [VP])

สำหรับชนิดของคำในภาษาไทยที่จะกำหนดในเงื่อนไขชนิดของคำในอาร์คนั้นได้กำหนดตามหลักไวยากรณ์ภาษาไทย[4] โดยได้กล่าวมาแล้วในบทที่ 3 ซึ่งบทความวิจัยนี้ได้กำหนดให้เหมาะสมกับระบบ M-ATN ไว้ทั้งสิ้น 16 ลักษณะดังนี้

1) คำนาม (NOUN) ได้แก่กลุ่มคำ สามานยนาม เช่น บ้าน ต้นไม้ ไก่ หนังสือ ฯลฯ

2) คำสรรพนาม (PRON) ได้แก่กลุ่มคำ บุรุษสรรพนาม เช่น ฉัน เขา เธอ หล่อน เรา ท่าน ฯลฯ

3) คำนามเฉพาะ (PROP) ได้แก่กลุ่มคำ วิสามานยนาม สมุหนาม เช่น สมศักดิ์ วัดโพธิ์ ฯลฯ

4) คำวิเศษณ์ (ADV) หมายถึงคำซึ่งมีหน้าที่ประกอบคำกริยา คำนาม ได้แก่กลุ่มคำ ลักษณวิเศษณ์ กาลวิเศษณ์ ประพันธวิเศษณ์ นิยมวิเศษณ์ เช่น อย่างนี้ อย่างไร อย่างนั้น อย่างมาก อย่างรวดเร็ว อย่างช้า ดี ชั่ว เดี๋ยวนี้ ดังนั้น ฯลฯ

5) คำกริยา (VERB) หมายถึงคำที่ทำหน้าที่แสดงกริยาอาการได้แก่กลุ่มคำวิกตรรถกริยา สกรรมกริยา อกรรมกริยา กริยาสภาวมาลา เช่น กิน เดิน นอน นั่ง ยืน เท่า คล้าย เสมือน เป็น คือ ฯลฯ

6) คำกริยาช่วย (AUX) หมายถึงคำที่มาช่วยให้กริยามีความความสมบูรณ์ หรือให้ความหมายดียิ่งขึ้นได้แก่กลุ่มคำกริยานุเคราะห์ เช่น กำลัง จะ ควร กำลังจะ ต้อง ฯลฯ

7) คำคุณศัพท์ (ADJ) หมายถึงคำขยายคำนาม สรรพนาม และ สิ่งของ ได้แก่กลุ่มคำ ลักษณวิเศษณ์ สถานวิเศษณ์ กาลวิเศษณ์ นิยมวิเศษณ์ เช่น บน ล่าง ดำ อ้วน ผอม สูง เตี้ย

8) คำลักษณะนาม (CLASS) หมายถึงการบ่งบอกลักษณะของ คำนาม สรรพนาม คน สัตว์ และ สิ่งของ ได้แก่กลุ่มคำลักษณนาม สมุหนาม เช่น อัน หลัง ตัว ด้าม คน เครื่อง เล่ม ใบ

9) คำกำหนด (DET) หมายถึงคำที่ใช้แทน คน สัตว์ สิ่งของ เช่น นั่น โน้น

10) คำหมายของ (GM) หมายถึงคำเชื่อม "ของ"

11) คำเชื่อม (LINKER) หมายถึงคำที่ใช้เชื่อม คำนามกับคำนามได้แก่กลุ่มคำสันธาน เช่น กะ กับ และ หรือ ทั้ง ตั้ง ฯลฯ

12) คำนับ (NUM) หมายถึงคำที่แสดงถึงจำนวนของสิ่งของได้แก่กลุ่มคำประมาณวิเศษณ์ เช่น หนึ่ง สอง แต่ละ บาง หลาย ทุก กี่ ฯลฯ

13) คำปฏิเสธ (NEG) หมายถึงคำที่มีความหมายในทางปฏิเสธได้แก่กลุ่มคำประติเษธวิเศษณ์ เช่น ไม่ มิได้ ไม่ได้ ฯลฯ

14) คำลงท้าย (END) หมายถึงคำที่อยู่ท้ายประโยคเพื่อทำให้ประโยคมีความหมายดียิ่งขึ้น เช่น ครับ ค่ะ นะ ฯลฯ

15) คำบุพบท (PREP) หมายถึงคำที่แสดงความสัมพันธ์ระหว่างคำ หรือประโยค ได้แก่คำ สำหรับ กับ ตาม โดย เพื่อ ที่ กระทั่ง ดังนั้น

16) คำแสดงคำถาม (QT) หมายถึงคำที่แสดงการถามได้แก่กลุ่มคำปฤจฉาวิเศษณ์เช่น หรือ ไหม อะไร

ซึ่งโครงข่ายที่กำหนดขึ้นนี้จะมีข้อกำหนดในการทำงานเป็นข้อๆดังตารางที่ 4.4

ตารางที่ 4.4 แสดงข้อกำหนดของระบบ M-ATN

ลำดับที่ 1. ต้องเริ่มต้นการตรวจสอบทั้งหมดที่โครงข่าย SN และจบสิ้นกระบวนการสุดท้ายที่โครงข่าย SN เช่นกัน

ลำดับที่ 2. มีลำดับการตรวจที่โนดใดๆตามลำดับตัวเลขที่กำกับไว้ในเงื่อนไขของอาร์คนั้นๆ

ลำดับที่ 3. หากมีการเรียกใช้โครงข่ายย่อยแล้วเงื่อนไขของอาร์คในโครงข่ายย่อยไม่ตรงตามชนิดของคำที่ประมวลผลเลย ให้ถือว่าไม่ได้มีการเรียกใช้โครงข่ายย่อยนั้นๆ

ลำดับที่ 4. การจะข้ามโนดจากโนดหนึ่งไปยังอีกโนดหนึ่งซึ่งถูกเชื่อมต่อด้วยอาร์คใดๆ จะต้องมีเงื่อนไขของอาร์คตรงกับชนิดของคำที่ประมวลผล

จากข้อกำหนดข้างต้นสามารถที่จะเขียนเป็นขั้นตอนการทำงานได้ในตารางที่ 4.5

ตารางที่ 4.5 แสดงกระบวนการทำงานของระบบ M-ATN

ลำดับที่ 1 : เริ่มต้นการประมวลผลที่ Entry โนดของ SN โดยเริ่มจากคำแรก (W_1 : i=1) ของประโยค ซึ่งกำหนดให้เป็นคำที่จะประมวลผลคำแรกกับเงื่อนไขของอาร์คอันดับที่ 1 แล้วเริ่มลำดับที่ 2

ลำดับที่ 2 : ตรวจสอบเงื่อนไขของอาร์คทั้งหมดที่ออกจากโนดนี้ว่ามีเงื่อนไขใดที่ตรงกับชนิดของคำ (W_1) ที่จะประมวลผลหรือไม่ ตามลำดับหมายเลขที่กำหนดไว้จากน้อยไปหามาก

2.1) ถ้าเงื่อนไขของอาร์คใดตรงตามชนิดของคำที่ประมวลผล (W_1) ให้เลือกอาร์คนั้นไว้ให้กับคำนั้นไว้พร้อมกับเก็บบันทึกลำดับของโนดและลำดับของอาร์คที่เลือกไว้ แล้วเลื่อนการประมวลผลไปที่คำต่อไปของประโยค แล้วเลื่อนการทำงานไปลำดับที่ 3

2.2) ถ้าไม่มีเงื่อนไขของอาร์คใดตรงกับชนิดของคำที่ประมวลผล (W_1) ให้ย้อนกลับไปยังโนดก่อนหน้านั้นตามเงื่อนไขของอาร์คที่ข้ามมา และให้นำคำศัพท์ก่อนหน้าคำศัพท์ที่ไม่สามารถประมวลผลได้ (W_{1-1}) มาประมวลผลเงื่อนไขของอาร์คใหม่ ที่ตำแหน่งของ

อาร์คใหม่ แต่หากประมวลผลไม่ได้จนกระทั่งย้อนกลับไปถึง Entry โนดของ SN (ที่ตำแหน่งคำศัพท์คำแรก W_1 ของประโยค)แล้วไม่สามารถประมวลผลได้อีก ให้ถือว่าประโยคนั้นผิดไวยากรณ์ ให้หยุดการทำงาน(เก็บบันทึกลำดับของโนดและลำดับของอาร์คไว้)

2.3) ถ้าประมวลผลเงื่อนไขของอาร์คตรงกับชนิดของคำที่ประมวลผล (W_i) ก่อนที่จะเลือกอาร์คนี้ไว้ให้ตรวจสอบว่าเส้นทางของอาร์คที่เลือกไว้นั้นชี้ไปยังโนดที่เคยเก็บบันทึกไว้ว่า ไม่สามารถประมวลผลคำศัพท์คำต่อไป (W_{i+1}) กับเงื่อนไขของอาร์คที่ต่อออกจากโนดนั้นได้ให้ประมวลผลเงื่อนไขของอาร์คอื่น (ถ้ามี)และเก็บบันทึกลำดับของโนดและลำดับของอาร์คไว้

2.4) หากระบบทำการประมวลผลเงื่อนไขของอาร์คที่จะต้องอ้างถึงโครงข่ายอื่น (NPN, PPN) ให้ข้ามไปประมวลผลตามโครงข่ายนั้นๆ และประมวลผลเงื่อนไขตามลำดับที่ 2 เมื่อประมวลผลเสร็จสิ้นให้ย้อนกลับไปยังโครงข่ายที่ออกมา

ลำดับที่ 3 : เริ่มต้นการประมวลผลใหม่กับคำต่อไป (W_{i+1}) ที่โนดซึ่งอยู่ต่อจากอาร์คที่เลือกไว้ โดยย้อนการทำงานไปที่ลำดับที่ 2 จนกว่าจะหมดคำที่จะประมวลผล (เมื่อ i=n)หรือโนดที่จะเริ่มการประมวลผลจึงจะเลื่อนการทำงานไปลำดับที่ 4

ลำดับที่ 4 : ให้นำเงื่อนไขของอาร์ค(ชนิดของคำ)ที่เลือกไว้ทั้งหมด มาเรียงกันตามลำดับเพื่อแสดงโครงสร้างทางไวยากรณ์ที่ถูกต้องของประโยคนั้น

ด้วยระบบ M-ATN ทำให้สามารถลดขั้นตอนการประมวลผลให้น้อยลงและรัดกุมขึ้นเมื่อเปรียบเทียบกับระบบ ATN และที่แน่นอนคือจะได้ความถูกต้องและความรวดเร็วในการคัดเลือกโครงสร้างของประโยค เพราะเราได้ลดความซับซ้อนให้น้อยลง เมื่อเปรียบเทียบกับระบบอื่นๆที่กล่าวมา การออกแบบและการสร้าง ระบบซอฟท์แวร์ให้กับกระบวนการ M-ATN ก็ทำได้สะดวก ซึ่งในงานวิจัยนี้ได้ใช้ภาษาซี (Turbo C) ในการเขียนซอฟท์แวร์ทั้งหมด โดยการแบ่งเป็นส่วนต่างๆไปตามขั้นตอนการประมวลผลของระบบ M-ATN รายละเอียดในส่วนนี้จะกล่าวในบทที่ 5

4.5 ตัวอย่างการประมวลผลประโยคภาษาไทยด้วยวิธี M-ATN

ในระบบการแปลภาษาด้วยเครื่องคอมพิวเตอร์ โดยเฉพาะภาษาไทยปัญหาที่สำคัญของการแปลภาษาไทย ก็คือในขั้นตอนของการประมวลผลประโยค เพราะว่าภาษาไทยในแต่ละประโยค ไม่มีการแบ่งคำกันอย่างเด่นชัดเหมือนกับภาษาอื่น เช่นภาษาอังกฤษ ตัวอย่างประโยคภาษาไทยที่สามารถแยกผลลัพธ์ได้มากกว่า 1 ผลลัพธ์ เช่น

ประโยคภาษาไทย ---> หลานมารอกราบปู่

ดังนั้นในระบบการแปลภาษาไทยจึงต้องมีการกระจายคำออกจากประโยคเสียก่อน โดย[6] ตรวจสอบคำจากพจนานุกรม ที่ได้เตรียมไว้แล้ว (หรืออาจกล่าวได้ว่าอยู่ในขั้นตอนของการประมวลผลคำนั้นเอง) แต่ถึงกระนั้นก็ตามปัญหาที่ตามมาก็คือ ผลลัพธ์ของการแยกแยะหน่วยคำ จากระบบ Fast Word Matching อาจจะมีได้มากกว่า 1 ผลลัพธ์ เช่น ประโยคภาษาไทยข้างต้น เมื่อทำการแยกแยะหน่วยคำจะได้ผลลัพธ์เกิดขึ้นมา 3 ประโยคดังนี้คือ

ผลลัพธ์ที่ 1 หลาน มา รอ กราบ ปู่
ผลลัพธ์ที่ 2 หลาน มา รอก ราบ ปู่
ผลลัพธ์ที่ 3 หลาน มาร อก ราบ ปู่

จะเห็นได้ว่าในการแยกแยะหน่วยคำเพียง 1 ประโยค อาจจะเกิดประโยคตามมามากมาย ทำให้ไม่สามารถทราบประโยคแท้จริงที่ถูกต้อง ดังนั้นในระบบ M-ATN จึงได้มีการกำหนดรูปแบบภาษาไทย (Thai Syntax) หรือรูปแบบโครงสร้างภาษาไทยขึ้นมาเพื่อใช้ในการตรวจสอบกับประโยคที่เกิดขึ้นจากการแยกแยะหน่วยคำทั้งหมดดังกล่าว ซึ่งจะทำการตัดผลลัพธ์ที่มีโครงสร้างไม่ถูกต้องตามรูปแบบ ไวยากรณ์ทั่วไป ทำให้ได้ประโยคที่ถูกต้องเป็นผลลัพธ์สุดท้าย จากตัวอย่างประโยคข้างต้นสามารถแยกแยะหน่วยคำและชนิดของคำออกตามผลลัพธ์ทั้ง 3 ประโยคได้ดังนี้

ผลลัพธ์ที่ 1	หลาน	มา	รอ	กราบ	ปู
ชนิดของคำ	คำนาม	คำกริยา	คำกริยา	คำกริยา	คำนาม

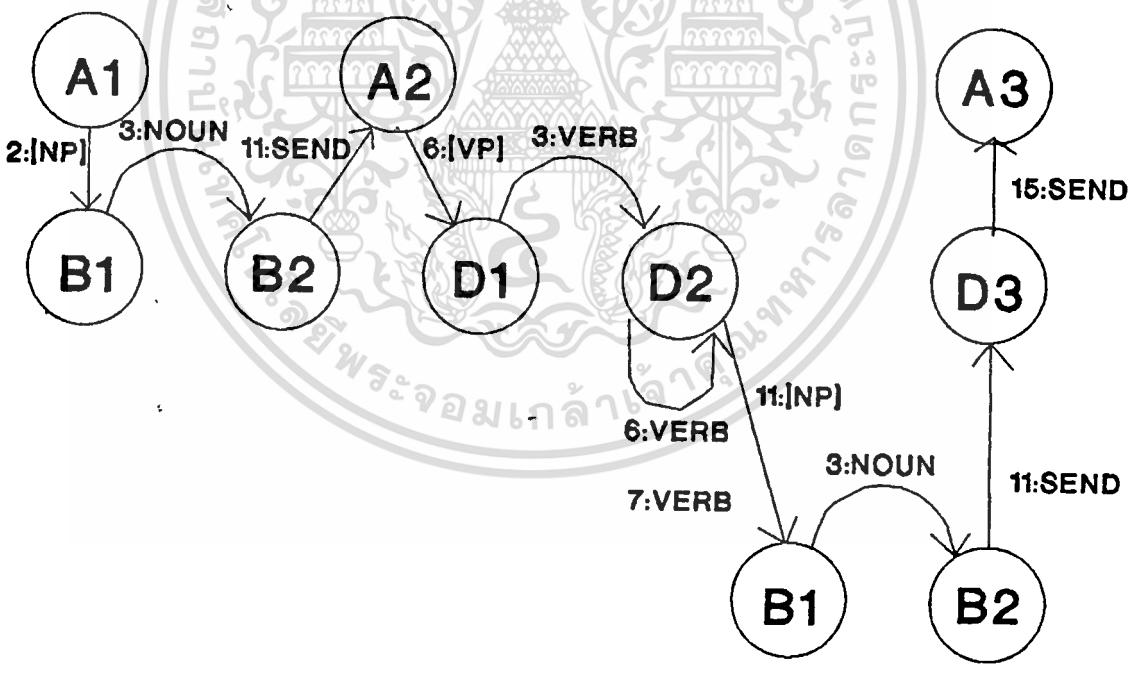
ผลลัพธ์ที่ 2	หลาน	มา	รอก	ราบ	ปู
ชนิดของคำ	คำนาม	คำกริยา	คำนาม	คำคุณศัพท์	คำนาม

ผลลัพธ์ที่ 3	หลาน	มาร	อก	ราบ	ปู
ชนิดของคำ	คำนาม	คำนาม	คำนาม	คำคุณศัพท์	คำนาม

จากผลลัพธ์ที่ 1 เราสามารถนำมาประมวลผลลัพธ์โดยเริ่มต้นจากใช้โครงข่าย **"โครงสร้างหลักของประโยค" (รูปที่ 4.6)** จะได้ผลลัพธ์ของการตรวจสอบที่ถูกต้องตามลำดับดังแสดงในรูปที่ 4.10

ผลลัพธ์ที่ 1	หลาน	มา	รอ	กราบ	ปู่
ชนิดของคำ	คำนาม	คำกริยา	คำกริยา	คำกริยา	คำนาม
คำย่อในโครงข่าย	NOUN	VERB	VERB	VERB	NOUN

(ก) แสดงชนิดของคำ และคำย่อในโครงข่าย



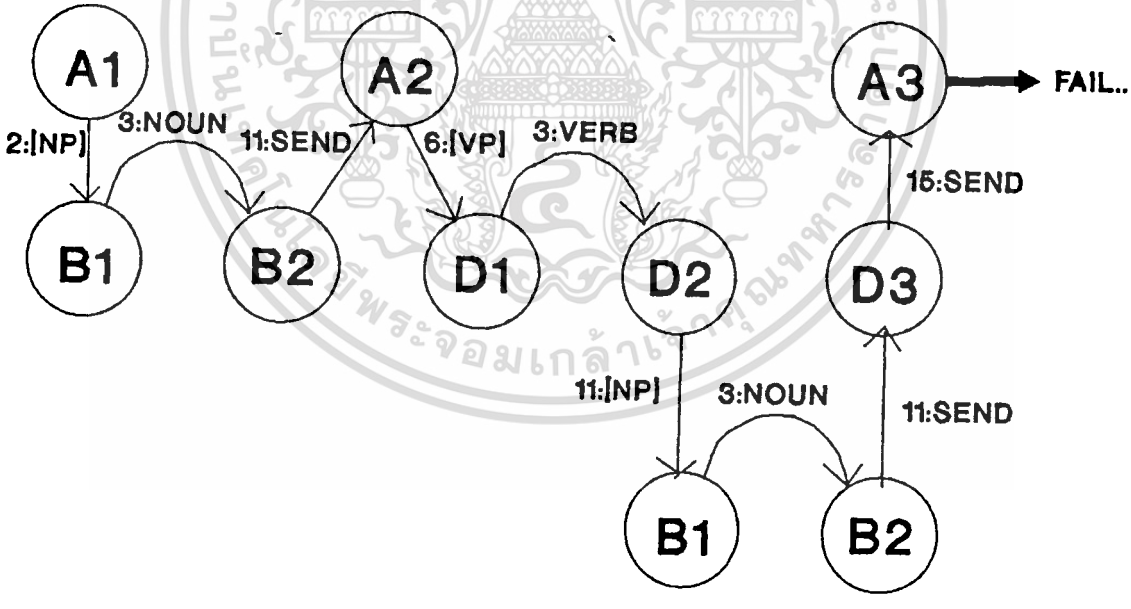
(ข) แสดงการตรวจสอบชนิดของคำในโครงข่าย

รูปที่ 4.10 แสดงการตรวจสอบของประโยคที่ถูกต้องตามหลักไวยากรณ์ที่กำหนด

สำหรับผลลัพธ์ที่ 2 เมื่อนำมาประมวลผลลัพธ์โดยเริ่มต้นจากโครงข่าย "โครงสร้างหลักของประโยค" รูปที่ 4.6 เราจะได้ผลลัพธ์ว่าผลลัพธ์ที่ 2 นี้มีรูปแบบไม่ถูกต้องตามหลักไวยากรณ์ ดังแสดงได้ในรูปที่ 4.11

ผลลัพธ์ที่ 2	หลาน	มา	รอก	ราบ	ปู่
ชนิดของคำ	คำนาม	คำกริยา	คำนาม	คำคุณศัพท์	คำนาม
คำย่อในโครงข่าย	NOUN	VERB	NOUN	ADJ	NOUN

(ก) แสดงชนิดของคำ และคำย่อในโครงข่าย



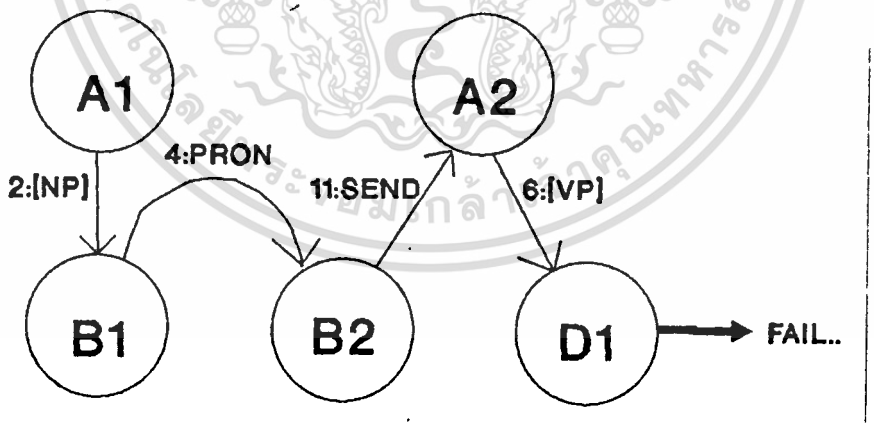
(ข) แสดงการตรวจสอบชนิดของคำในโครงข่าย

รูปที่ 4.11 แสดงการตรวจสอบชนิดของคำในประโยคที่ 2 แล้วไม่ถูกต้องตามหลักไวยากรณ์

ประมวลผลได้ตาม"โครงสร้างหลักของประโยค" ตามรูปที่ 4.6 ในทำนองเดียวกันนี้เราสามารถแสดงได้ว่าผลลัพธ์ที่ 3 ก็ไม่เป็นไปตามรูปแบบของโครงสร้างภาษาไทยคือไม่มีภาคแสดงปรากฏในผลลัพธ์ดังกล่าว ซึ่งสามารถแสดงได้ดังรูปที่ 4.12

ผลลัพธ์ที่ 3	หลาน	มาร	อก	ราบ	ปู่
ชนิดของคำ	คำนาม	คำนาม	คำนาม	คำคุณศัพท์	คำนาม
คำย่อในโครงข่าย	NOUN	NOUN	NOUN	ADJ	NOUN

(ก) แสดงชนิดของคำ และคำย่อในโครงข่าย



(ข) แสดงการตรวจสอบชนิดของคำในโครงข่าย

รูปที่ 4.12 แสดงการตรวจสอบชนิดของคำในประโยคที่ 2 แล้วไม่ถูกต้องตามหลักไวยากรณ์

จากตัวอย่างดังกล่าวข้างต้นจะเห็นได้ว่าผลลัพธ์ที่ 1 นั้นถูกต้องตามโครงสร้างหลักของภาษาไทย แต่สำหรับผลลัพธ์ที่ 2 และ 3 ไม่สามารถนำมาประมวลผลประโยคได้ เนื่องจากคำแต่ละคำในผลลัพธ์ไม่เป็นไปตามรูปแบบของโครงสร้างภาษาไทยที่ได้กำหนดขึ้น ทำให้สามารถตัดผลลัพธ์ที่ 2 และ 3 ที่เกิดจากการกระจายคำไม่ถูกต้องออกไปได้และเหลือผลลัพธ์ที่ถูกต้องเพียงผลลัพธ์เดียว ผลลัพธ์ที่ได้จากการประมวลผลประโยคในครั้งนี้ สามารถนำไปใช้ในขั้นตอนต่อไปของระบบการแปลภาษาด้วยเครื่องคอมพิวเตอร์

4.6 บทสรุป

จากตัวอย่างที่ผ่านมาจะเห็นได้ว่า การประมวลผลประโยคด้วยวิธี M-ATN ทำให้สามารถที่จะประมวลผลว่าผลลัพธ์ไหนถูกต้องตามหลักไวยากรณ์ของโครงสร้างภาษาไทย และผลลัพธ์ไหนที่ไม่ถูกต้องตามหลักไวยากรณ์ดังกล่าว เพื่อเป็นเครื่องมือช่วยในการ ประมวลผลประโยค เฉพาะโครงสร้างทางด้านไวยากรณ์ (Syntactical Analysis) ในกรณีที่ต้องการตัดผลลัพธ์ที่ผ่านมาจากขั้นตอนของการประมวลผลคำและการกระจายคำ ให้เหลือเฉพาะผลลัพธ์ที่ถูกต้องที่สุด เพื่อนำไปใช้ในขั้นตอนของการหาโครงสร้างลึกของประโยคต่อไป

บทที่ 5

ซอฟท์แวร์ของระบบการวิเคราะห์โครงสร้างภาษาไทย

5.1 บทนำ

ในการสร้างหรือเขียนซอฟท์แวร์ในแต่ละครั้งสิ่งที่ต้องคำนึงถึงอันดับแรกก็คือ อัลกอริทึมของซอฟท์แวร์ที่จะสร้าง ว่าจะมีขั้นตอนการทำงานอย่างไรบ้าง หรือมีลักษณะของข้อมูลเข้า (Data Input) หรือลักษณะของข้อมูลออก (Data Output) เป็นอย่างไรบ้าง เป็นต้น ซึ่งสิ่งเหล่านี้มีความสำคัญต่อการสร้างซอฟท์แวร์มาก จากอัลกอริทึมการทำงานของระบบแยกแยะหน่วยคำ(Fast Word Matching Mechanism) ในบทที่ 2 ฐานความรู้ทางไวยากรณ์ภาษาไทย(Knowledge Base of Thai Syntax) ที่กล่าวโดยละเอียดในบทที่ 3 และอัลกอริทึมการทำงานของระบบประมวลผลโครงข่ายของประโยคภาษาไทย (M-ATN Mechanism) ในบทที่ 4 เราได้นำอัลกอริทึมและฐานความรู้เหล่านี้มาออกแบบและสร้างเป็นซอฟท์แวร์คอมพิวเตอร์ โดยจะมีลำดับการทำงานเริ่มจากการรับข้อมูลเข้าทางแป้นพิมพ์ในลักษณะของอักขระภาษาไทย แล้วผ่านระบบ Fast Word Matching จะได้ข้อมูลออกเป็นคำศัพท์ที่แยกออกจากประโยคเป็นคำๆ ต่อจากนั้นก็หาชนิดของคำแล้วส่งไปยังระบบ M-ATN ก็จะได้หน่วยคำที่แยกแยะออกจากประโยคซึ่งประมวลผลโครงสร้างภาษาไทยอย่างถูกต้องแล้ว ในบทนี้จะได้กล่าวถึงการออกแบบ ซอฟท์แวร์ของส่วนต่าง ๆ ของระบบทั้งหมด ตลอดจนการสร้างโปรแกรมเหล่านั้นและการใช้งานจริงของโปรแกรมทั้งหมด

5.2 การออกแบบและการสร้างซอฟท์แวร์ของระบบวิเคราะห์โครงสร้างภาษาไทย

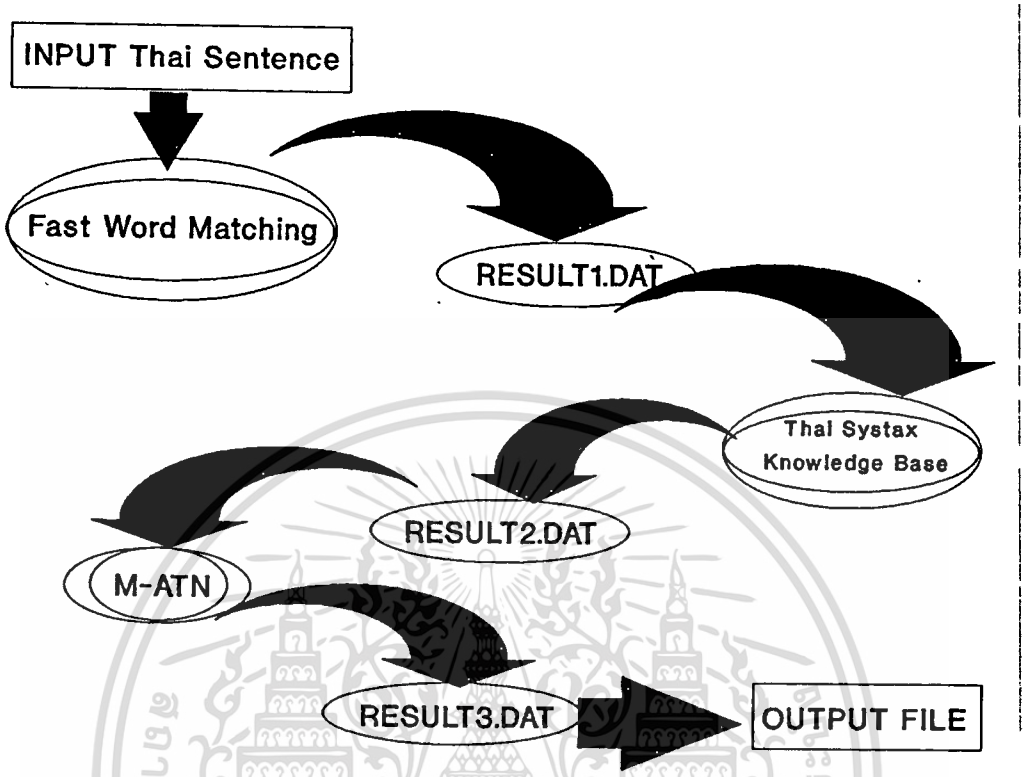
จากที่กล่าวมาแล้วว่าระบบวิเคราะห์โครงสร้างประโยคภาษาไทยของงานวิจัยนี้แบ่งออกเป็น 3 ส่วนใหญ่ๆ ซึ่งมีการทำงานร่วมกันตามลำดับดังนี้

ลำดับที่ 1 : ส่วนการแยกแยะหน่วยคำออกจากประโยคด้วยขบวนการ Fast Word Matching

ลำดับที่ 2 : ส่วนฐานความรู้ทางไวยากรณ์ไทย(Knowledge Base of Thai Syntax)

ลำดับที่ 3 : ส่วนการประมวลผลโครงสร้างที่ถูกต้องของประโยคภาษาไทยด้วย

ขบวนการของ Modified Augmented Transition Network :M-ATN แต่ละส่วนจะมีหน้าที่เฉพาะอย่างที่ต่อเนื่องกันดังนี้โดยระบบ Fast Word Matching จะแยกหน่วยคำของประโยคภาษาไทยและระบบฐานความรู้ทางไวยากรณ์ จะหาชนิดของคำที่จะทำการประมวลผล ต่อจากนั้นก็จะส่งผลลัพธ์ต่อไปยังระบบ M-ATN เพื่อทำการประมวลผลทางโครงสร้าง ซึ่งผลลัพธ์ที่ได้จะเป็นประโยคซึ่งถูกแยกแยะคำที่ถูกต้องตามกฎไวยากรณ์ภาษาไทย จากอัลกอริทึมของระบบ Fast Word Matching ระบบฐานความรู้ทางไวยากรณ์ และระบบ M-ATN เราได้นำมาสร้างเป็นซอฟท์แวร์คอมพิวเตอร์โดยมีขั้นตอนการทำงานดังรูปที่ 5.1



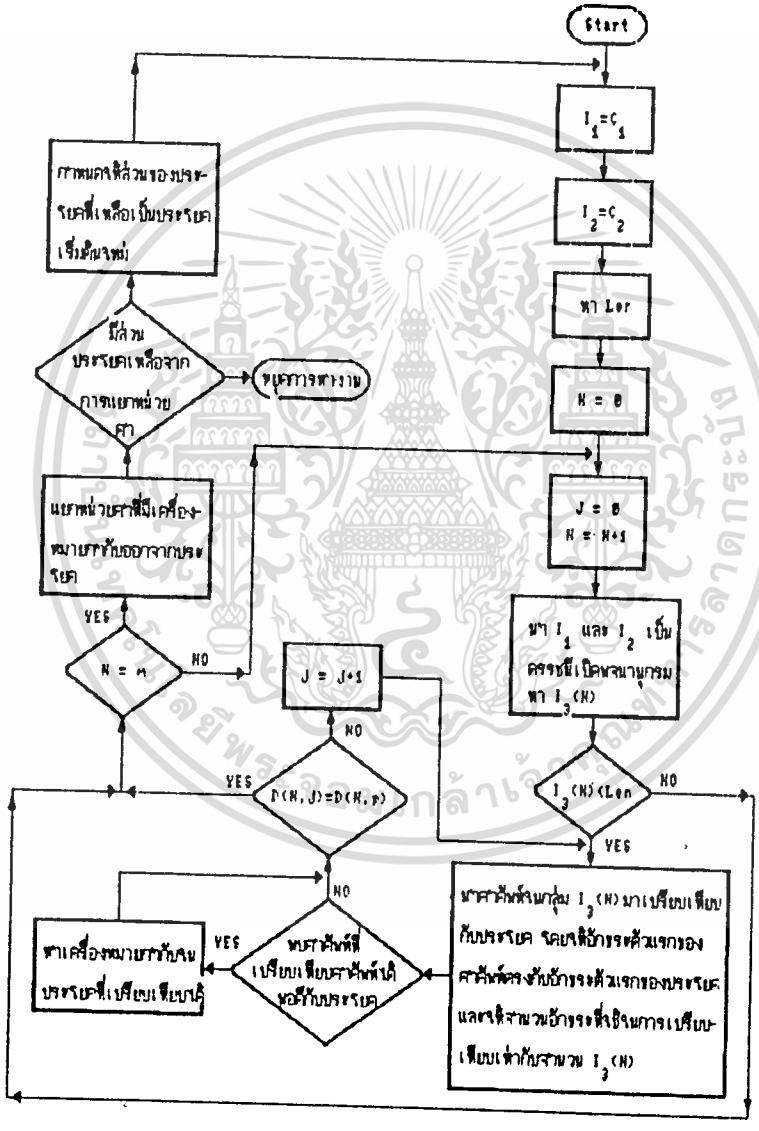
รูปที่ 5.1 แสดงขั้นตอนการทำงานของซอฟท์แวร์ในระบบวิเคราะห์โครงสร้างภาษาไทย

ในรูปที่ 5.1 จะมีการทำงานกล่าวคือ เมื่อเราป้อนข้อมูลเป็นประโยคภาษาไทย (Thai Sentence) ซึ่งเป็น อักขระที่ได้จากแป้นพิมพ์ ประโยคที่ป้อนเข้ามาก็จะเข้าสู่ระบบ Fast Word Matching จะได้ข้อมูลที่เป็นผลลัพธ์ในลักษณะของไฟล์ข้อมูลชื่อ Result1.DAT แล้วผ่าน กระบวนการฐานความรู้ทางไวยากรณ์ไทย จะได้ข้อมูลที่เป็นผลลัพธ์ในลักษณะของไฟล์ข้อมูล (Text File) ชื่อ RESULT2.DAT จากนั้นระบบ M-ATN ก็จะรับแม้มข้อมูลนี้ไปปฏิบัติตามอัลกอริทึมที่ได้กล่าวมาแล้วในบทที่ 4 ซึ่งเมื่อเสร็จสิ้นกระบวนการก็จะได้ผลลัพธ์ออกมาในรูปของไฟล์ข้อมูลอีกไฟล์หนึ่งชื่อ RESULT3.DAT ซึ่งผลลัพธ์ที่ได้จากระบบ M-ATN นี้ จะเป็นผลลัพธ์ของระบบวิเคราะห์โครงสร้างภาษาไทย

จากที่กล่าวมาข้างต้น งานวิจัยนี้ได้ทำการออกแบบและสร้างซอฟท์แวร์ของแต่ละส่วนซึ่งจะกล่าวโดยละเอียดดังต่อไปนี้

5.2.1 ซอฟท์แวร์ของระบบแยกแยะหน่วยคำ Fast Word Matching

จากขั้นตอนการทำงานของระบบแยกแยะหน่วยคำ Fast Word Matching ที่กล่าวมาแล้วในบทที่ 2 ได้นำมาออกแบบซอฟท์แวร์โดยเขียนเป็นโฟล์ชาร์ตดังรูปที่ 5.2



รูปที่ 5.2 แสดงโฟล์ชาร์ตของซอฟท์แวร์ระบบแยกแยะหน่วยคำ Fast Word Matching

เมื่อ

I_1 = ดรรชนีคำศัพท์ชั้นที่ 1

I_2 = ดรรชนีคำศัพท์ชั้นที่ 2

$I_3(N)$ = ดรรชนีคำศัพท์ชั้นที่ 3 กลุ่มที่ N โดย N = 1...m

m = กลุ่มสุดท้ายของดรรชนีคำศัพท์ชั้นที่ 3

C_1 = รหัสแอสกีของอักขระตัวที่ 1 ของประโยค

C_2 = รหัสแอสกีของอักขระตัวที่ 2 ของประโยค

D(N,j) = คำศัพท์จากพจนานุกรมคำที่ j ในดรรชนีคำศัพท์ชั้นที่ 3 กลุ่มที่ N
โดย j = 1...p

p = ลำดับคำศัพท์คำสุดท้ายจากพจนานุกรมคำศัพท์ ในดรรชนีคำศัพท์ชั้นที่ 3 กลุ่มที่ N

D(N,p) = คำศัพท์คำสุดท้ายจากพจนานุกรมคำศัพท์ ในดรรชนีคำศัพท์ชั้นที่ 3 กลุ่มที่ N

Len = จำนวนอักขระของประโยค

จากโฟล์ชาร์ตในรูปที่ 5.2 นำมาสร้างซอฟท์แวร์ ซึ่งซอฟท์แวร์นี้สามารถที่จะเริ่มทำงานได้นั้น จะต้องมีการป้อนข้อมูลที่เป็นประโยคภาษาไทย โดยมีรูปแบบของข้อมูลจะต้องเป็นอักขระภาษาไทยที่พิมพ์ต่อเนื่องกันเมื่อประโยคป้อนเข้าสู่ระบบแยกแยะหน่วยคำ จะถูกเก็บไว้ในหน่วยเก็บความจำชั่วคราว (Buffer) ซึ่งมีลักษณะเรียงลำดับของอักขระดังรูปที่ 5.3

1	2	3	4	5	6	7	8	9	10	11
ฉ	ั	น	ก	ิ	น	ข	้	า	ว	<Enter>

รูปที่ 5.3 แสดงการเรียงลำดับการเก็บตัวอักขระในหน่วยเก็บความจำชั่วคราว

เมื่อผ่านระบบ Fast Word Matching แล้วจะได้ข้อมูลที่เป็นผลลัพธ์ออกมาซึ่งเก็บในรูปของไฟล์ข้อมูล 2 ไฟล์คือ ไฟล์ชื่อ RESULT.DAT กับไฟล์ชื่อ RESULT1.DAT

5.2.1.1 ไฟล์ RESULT.DAT ในส่วนนี้จะเป็นไฟล์ข้อมูลซึ่งเก็บผลลัพธ์ทั้งหมดของการแยกแยะหน่วยคำของระบบแยกแยะหน่วยคำ Fast Word Matching ตั้งแต่หน่วยคำแรกที่เริ่มแยกแยะออกจากประโยคจนกระทั่งถึงคำสุดท้ายของประโยคที่แยกแยะออกมาได้ ซึ่งหน่วยคำที่แยกแยะออกมาได้แต่ละคำจะมีสัญลักษณ์ขีดล่าง (_) คั่นระหว่างหน่วยคำที่แยกได้ถ้าแยกแยะหน่วยคำได้จนจบประโยคแล้วจะลงท้ายด้วยจุด(.) แต่ถ้าหากยังไม่ถึงหน่วยคำสุดท้ายของประโยคหรือไม่สามารถแยกแยะหน่วยคำได้อีกจะจบลงด้วยสัญลักษณ์ขีดล่าง (_) เมื่อจบสิ้นกระบวนการแยกแยะหน่วยคำจะจบลงด้วยอักขระคิว (q) 1 ตัว (โดยจะเก็บไว้บรรทัดสุดท้ายของไฟล์)

ตัวอย่าง

ฉัน_

ฉัน_กิน_

ฉัน_กิน_ข้าว.

ฉัน_กิน_ข้า_

q

รูปที่ 5.4 แสดงลักษณะของข้อมูลในไฟล์ RESULT.DAT

5.2.1.2ไฟล์ RESULT1.DAT ในส่วนนี้จะเป็นไฟล์ข้อมูลที่เก็บเฉพาะผลลัพธ์ของการแยกแยะหน่วยคำออกจากประโยคที่สามารถแยกแยะหน่วยคำจนกระทั่งจบประโยคได้ไว้ โดยหน่วยคำที่แยกแยะออกมาได้แต่ละคำจะมีสัญลักษณ์ขีดหลัง(/) คั่นระหว่างหน่วยคำที่แยกได้และจบท้ายประโยคด้วยจุด(.) และเมื่อสิ้นสุดไฟล์จะจบลงด้วยอักษรคิว(q)

ตัวอย่าง

ฉัน/กิน/ข้าว.

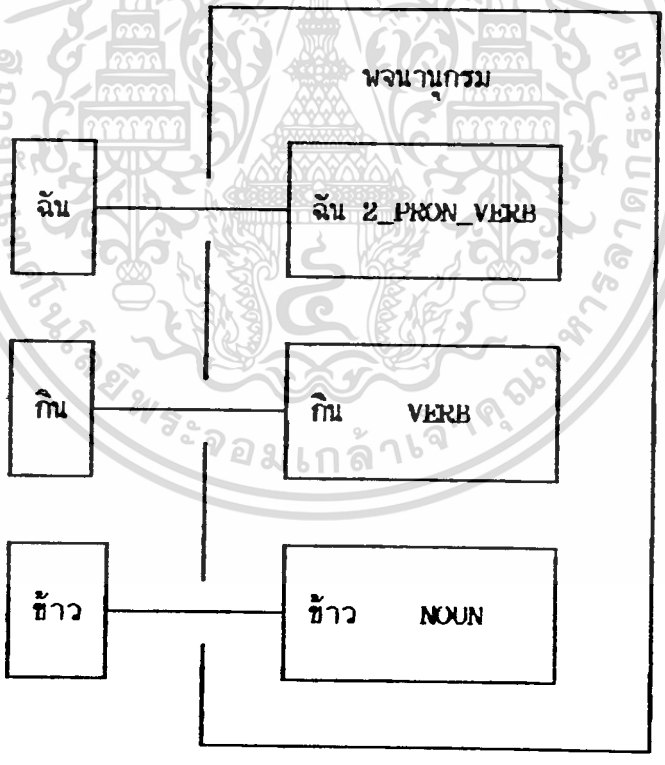
q

รูปที่ 5.5 แสดงลักษณะของข้อมูลในไฟล์ RESULT1.DAT

5.2.2 ซอฟท์แวร์ของระบบฐานข้อมูลความรู้ทางไวยากรณ์ภาษาไทย(Knowledge Base of Thai Syntax)

อัลกอริทึมนี้จะรับข้อมูลต่อจากระบบ Fast Word Matching แล้วนำมาหาชนิดของคำในภาษาไทยซึ่งเป็นส่วนของฐานความรู้ โดยซอฟท์แวร์ในส่วนนี้จะแบ่งเป็น 2 ส่วน คือ

5.2.2.1 ส่วนของการหาชนิดของคำ จะรับไปไฟล์ข้อมูล ชื่อ RESULT1.DAT ซึ่งเป็นผลลัพธ์มาจากระบบ Fast Word Matching โดยจะนำหน่วยคำที่แยกได้ไปเป็นดรรชนีเข้าสู่พจนานุกรมคำศัพท์ เพื่อเรียกใช้ฐานความรู้ทางไวยากรณ์ภาษาไทยของหน่วยคำ โดยจะมีลักษณะของหน่วยคำดังรูปที่ 5.6



หมายเหตุ 2_PRON_VERB หมายถึง คำศัพท์ที่มีชนิดของคำ 2 อย่างคือคำสรรพนามกับคำกริยา

รูปที่ 5.6 แสดงการเข้าหาชนิดของคำศัพท์

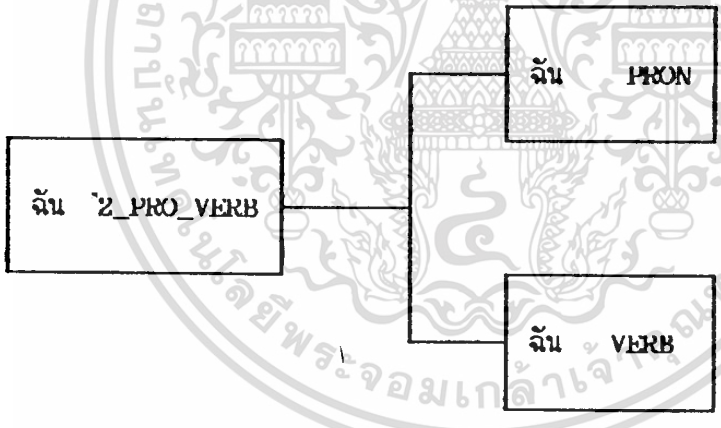
5.2.2.2 ส่วนของการแยกชนิดของคำที่มีมากกว่า 1 ชนิด ภายหลังจากการหาชนิดของหน่วยคำแล้ว ซอฟท์แวร์จะทำการตรวจสอบอีกว่ามีชนิดของหน่วยคำมากกว่า 1 ชนิด ก็จะทำการแยกชนิดของคำออก โดยมีลักษณะการแยกดังนี้

PRON หมายถึง คำคำนั้นมีชนิดของหน่วยคำได้ 1 ชนิดคือเป็นคำสรรพนาม

2_PRON_VERB หมายถึง คำคำนั้นมีชนิดของหน่วยคำได้ 2 ชนิด คือเป็นคำสรรพนาม (PRON) และคำกริยา(VERB)

3_NOUN_PRON_VERB หมายถึง คำคำนั้นมีชนิดของหน่วยคำ 3 ชนิด คือเป็นคำนาม (NOUN), คำสรรพนาม(PRON) และ คำกริยา (VERB)

ดังแสดงตัวอย่างได้ดังรูปที่ 5.7



รูปที่ 5.7 แสดงการแยกคำศัพท์ที่มีชนิดของคำมากกว่า 1 ชนิด

ซึ่งผลลัพธ์ที่ได้จากระบบ Knowledge Base จะเก็บในรูปของไฟล์ข้อมูลชื่อ RESULT2.DAT โดยจะมีลักษณะการเรียงลำดับของข้อมูลเริ่มต้นด้วย หน่วยคำและตามด้วยชนิดของคำ เรียงลำดับจากซ้ายไปขวา เรียงลำดับกันอย่างนี้จนกระทั่งถึงหน่วยคำสุดท้ายและชนิดของคำของหน่วยคำสุดท้าย จะจบ

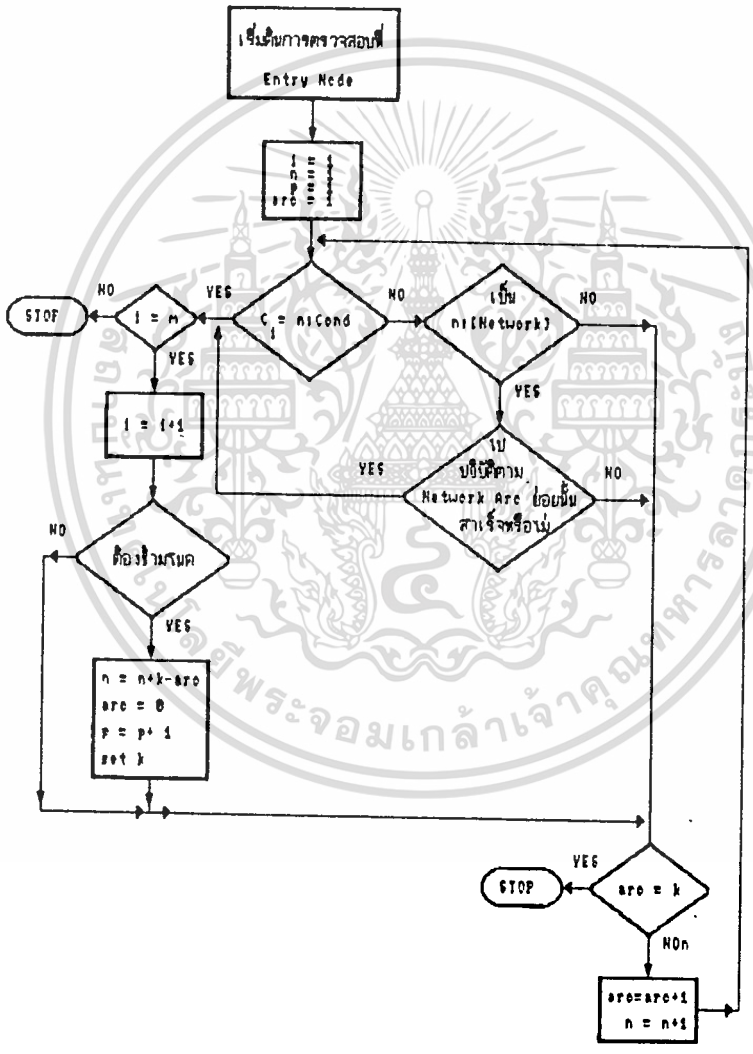
ประโยคด้วยสัญลักษณ์จุด(.) เมื่อจบสิ้นกระบวนการแล้วจะจบไฟล์ข้อมูลด้วยอักขระคิว(q) 2 ตัว จะมีรูปแบบดังรูปที่ 5.8

```
ฉัน PRON กิน VERB ข้าว NOUN .
ฉัน VERB กิน VERB ข้าว NOUN .
q   q
```

รูปที่ 5.8 แสดงรูปแบบของข้อมูลในไฟล์ข้อมูลออกจากระบบฐานความรู้ทางไวยากรณ์ภาษาไทย

5.2.3 ซอฟท์แวร์ของระบบ M-ATN

จากขั้นตอนการทำงานของระบบประมวลผลโครงสร้างภาษาไทยที่กล่าวมาแล้วในบทที่ 4 ได้นำมาออกแบบซอฟท์แวร์โดยเขียนเป็นโฟล์ชาร์ตดังรูปที่ 5.9



รูปที่ 5.9 แสดงโฟล์ชาร์ตของซอฟท์แวร์ระบบ M-ATN

เมื่อ

C_i = ชนิดของคำตำแหน่งที่ i โดยที่ i = 1...m

m = ตำแหน่งสุดท้ายของชนิดของคำที่ตรวจสอบ

n = ลำดับการตรวจสอบ โดยที่ n = 1,2,...

n:Cond = เป็นเงื่อนไขของอาร์คที่เป็น Category Arc หรือ Fixed Arc ลำดับการตรวจสอบที่ n

n:LNetworkJ = เป็นเงื่อนไขของอาร์คที่เป็น Network Arc ลำดับการตรวจสอบที่ n

k = จำนวนของอาร์คทั้งหมดที่ต่ออยู่กับโนดใดๆ

arc = จำนวนของอาร์คที่ตรวจสอบมาแล้วในโนดใดๆ

อัลกอริทึมในส่วนนี้จะรับข้อมูลต่อจากระบบฐานความรู้ทางไวยากรณ์ ซึ่งจะมีการนำชนิดของคำมาประมวลผลทางไวยากรณ์ของประโยคภาษาไทย ดังจะมีการทำงานของซอฟท์แวร์ตามอัลกอริทึมที่กำหนดในบทที่ 4 ระบบนี้จะเริ่มทำงานโดยจะรับไฟล์ข้อมูลจากไฟล์ชื่อ RESULT2.DAT โดยจะเก็บข้อมูลที่ได้ลงในหน่วยความจำของคอมพิวเตอร์ครั้งละประโยค (การจะรู้ว่าข้อมูลจบประโยคนั้นสามารถตรวจสอบด้วยสัญลักษณ์จุด(.)) จากนั้นก็นำประโยคที่เก็บไว้ในหน่วยความจำมาครั้งละ 1 ประโยค มาตรวจสอบชนิดของคำศัพท์ตามโครงสร้างที่กำหนดไว้ทีละหน่วยคำศัพท์ โดยจะตรวจสอบชนิดของคำศัพท์เรียงลำดับจากซ้ายไปขวา ตามวิธีการที่กำหนดไว้ในบทที่ 4 ซึ่งผลลัพธ์ที่ถูกต้องตามไวยากรณ์ที่กำหนดจะถูกเก็บไว้ในไฟล์ข้อมูล RESULT3.DAT ดังแสดงในรูปที่ 5.9 โดยจะมีการเรียงลำดับของข้อมูลเริ่มต้นด้วยหน่วยคำและตามด้วยชนิดของคำเรียงลำดับจากซ้ายไปขวาเรียงลำดับอย่างนี้จนกระทั่งจบประโยค ซึ่งเมื่อจบประโยคจะแสดงด้วยสัญลักษณ์จุด(.) และจบไฟล์ข้อมูลด้วย สัญลักษณ์คิว(q)

```
ฉัน PRON กิน VERB ข้าว NOUN .
q
```

รูปที่ 5.10 แสดงรูปแบบของผลลัพธ์ในไฟล์ข้อมูลที่ได้จากการประมวลผลด้วยระบบ M-AIN

5.3 การใช้งานซอฟท์แวร์ของระบบการวิเคราะห์โครงสร้างภาษาไทย

ในการวิเคราะห์ประโยคภาษาไทยจะต้องประกอบไปด้วยไฟล์ข้อมูลดังนี้

1. *.DIC จะเป็นไฟล์ข้อมูลสำหรับเก็บพจนานุกรมคำศัพท์และดัชนีของไฟล์
2. TAS.EXE เป็นไฟล์ข้อมูลที่ใช้สำหรับ RUN ซอฟท์แวร์วิเคราะห์โครงสร้างภาษาไทยทั้งระบบ
3. FWM.EXE เป็นซอฟท์แวร์สำหรับแยกแยะหน่วยคำ
4. KLB.EXE เป็นซอฟท์แวร์ฐานความรู้ทางไวยากรณ์ภาษาไทย
5. M-ATN.EXE เป็นซอฟท์แวร์ประมวลผลทางไวยากรณ์ภาษาไทย
6. *.DAT เป็นไฟล์ข้อมูลสำหรับเก็บผลลัพธ์ที่ได้จากส่วนต่างๆของระบบ

เมื่อจะเริ่มต้นวิเคราะห์ประโยคภาษาไทยจะมีขั้นตอนดังนี้

1. พิมพ์คำว่า TAS แล้วตามด้วยการกดแป้น ENTER หรือ RETURN ดังนี้

```
C > TAS <ENTER>
```

2. จะปรากฏข้อความว่าดังรูปที่ 5.11

ระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์

อาจารย์ที่ปรึกษา ผศ. ดร. รัตติกร วรากูลศิริพันธุ์

โดย นายสมศักดิ์ จันวัน

[1] Fast Word Matching

[2] Knownledge Base

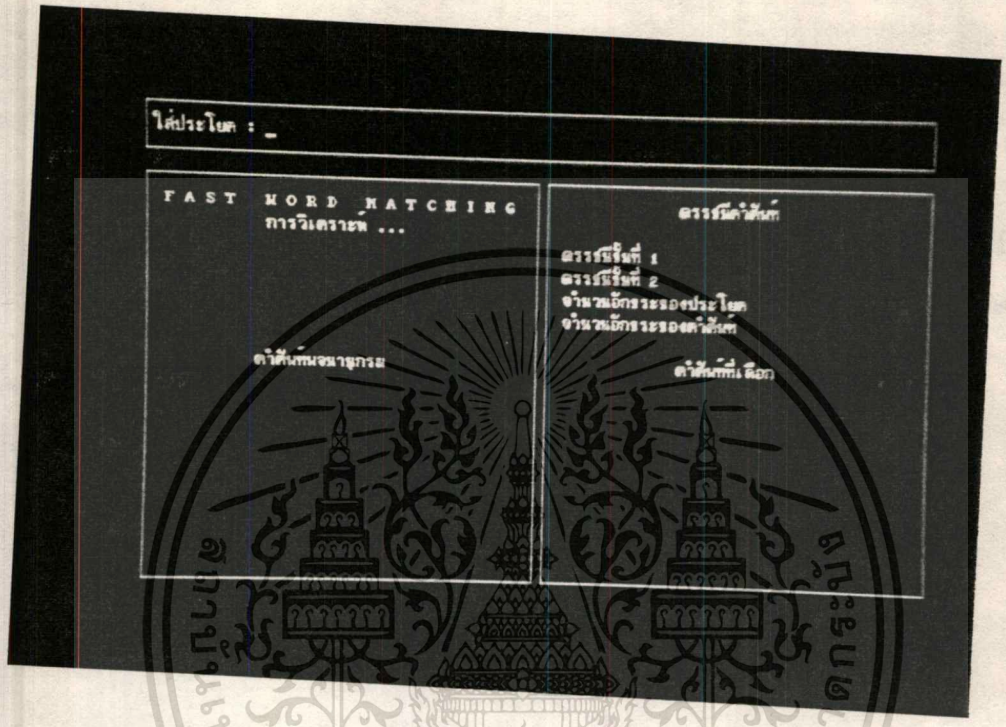
[3] M-ATN

[4] EXIT

Input Number 1, 2, 3 or 4 :

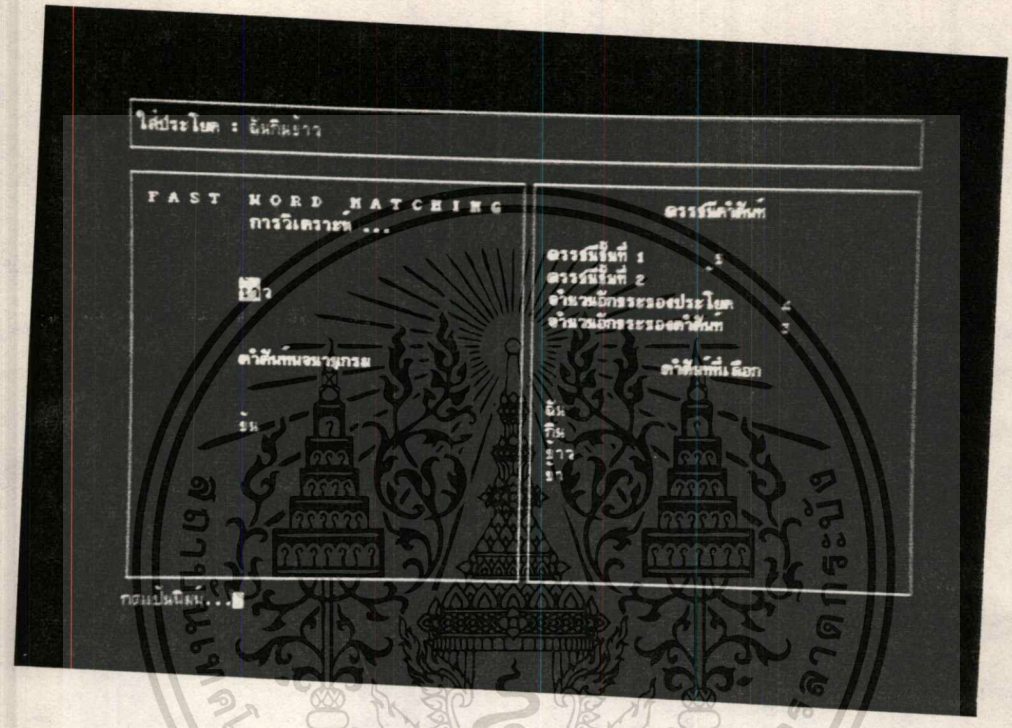
รูปที่ 5.11 แสดงภาพที่ปรากฏบนจอมอนิเตอร์เมื่อ RUN ซอฟร์แวร์ TAS.EXE

2.1 เมื่อเลือกหมายเลข 1 (Fast Word Matching) โดยการกดแป้นพิมพ์หมายเลข 1 แล้วกดแป้นพิมพ์ Return หรือ Enter เพื่อที่จะแยกแยะหน่วยคำในประโยค ซึ่งจะปรากฏภาพบนจอมอนิเตอร์ดังรูปที่ 5.12 (การเปลี่ยนแป้นพิมพ์จากภาษาอังกฤษเป็นภาษาไทย หรือภาษาไทยเป็นภาษาอังกฤษ โดยการกดแป้นพิมพ์นิเสธ (~))



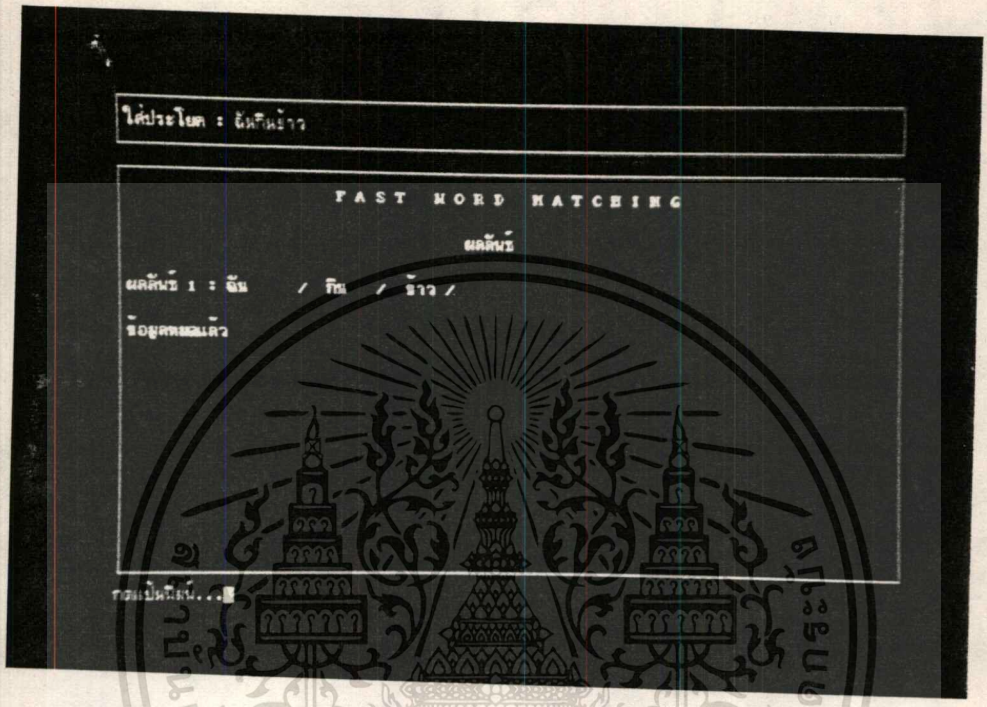
รูปที่ 5.12 แสดงภาพที่ปรากฏบนจอมอนิเตอร์เมื่อเลือกหมายเลข 1

โดยระบบจะรอรับการพิมพ์ประโยคภาษาไทยที่ช่อง **ใส่ประโยค** ซึ่งเมื่อป้อนประโยคเสร็จแล้วให้กดแป้นพิมพ์ Return หรือ Enter ระบบจะทำการแยกแยะหน่วยคำออกได้ดังรูปที่ 5.13



รูปที่ 5.13 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ขณะที่ระบบ Fast Word Matching ทำงาน

เมื่อระบบ Fast Word Matching ทำงานเสร็จสิ้นจะให้ผลลัพธ์ที่แสดงได้ดังรูปที่ 5.14



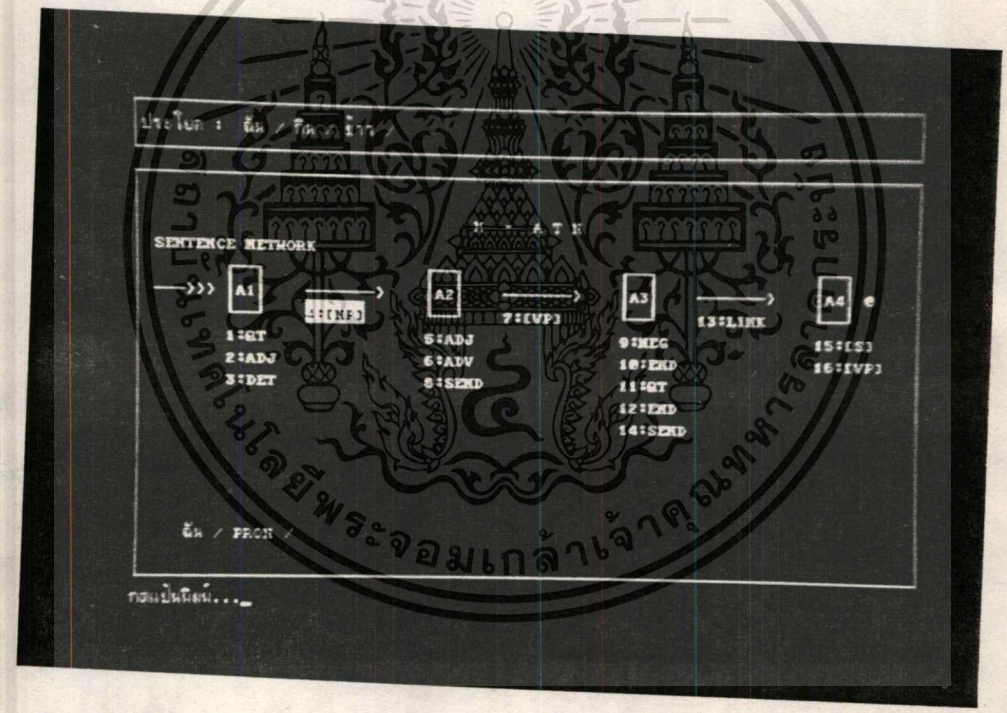
รูปที่ 5.14 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ซึ่งเป็นผลลัพธ์ของระบบ Fast Word Matching

2.2 เมื่อเลือกหมายเลข 2 (Knowledge Base) โดยการกดแป้นพิมพ์หมายเลข 2 แล้วกดแป้นพิมพ์ Return หรือ Enter เพื่อที่จะหาชนิดของคำจากฐานข้อมูลทางไวยากรณ์ภาษาไทย จะปรากฏหน้าจอมอนิเตอร์ดังรูปที่ 5.15 ซึ่งบนจอมอนิเตอร์จะแสดงหน่วยคำ และชนิดของคำของประโยคที่แสดงอยู่ในช่องผลลัพธ์ของระบบฐานข้อมูลไวยากรณ์ภาษาไทย

รูปที่ 5.15 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ซึ่งเป็นผลลัพธ์ของระบบ Knowledge Base

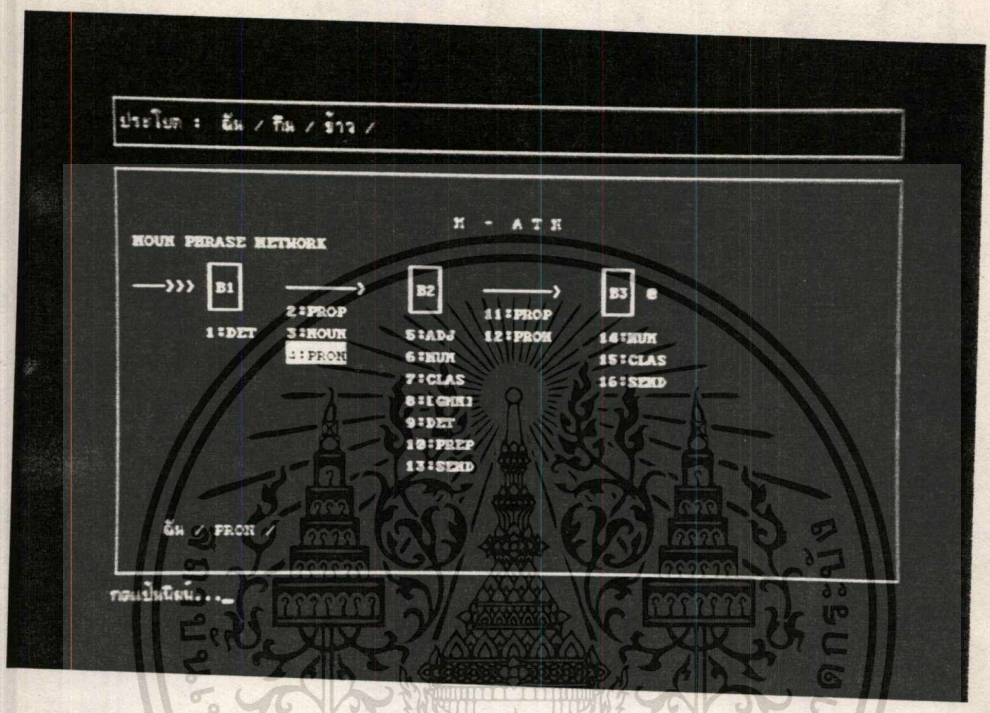
2.3 เมื่อเลือกหมายเลข 3 (M-ATN) โดยการกดแป้นพิมพ์หมายเลข 3 แล้วกดแป้นพิมพ์ Return หรือ Enter เพื่อต้องการจะประมวลผลทางไวยากรณ์ตามวิธีการ M-ATN ซึ่งจะเริ่มทำการประมวลผลเป็นส่วน ๆ ตามโครงข่ายที่กำหนดโดยจะทำการตรวจสอบโครงข่ายเป็นส่วนๆ โดยแสดงบนจอมอนิเตอร์ได้ดังต่อไปนี้

2.3.1 โครงข่ายประโยค (Sentence Network) จะแสดงได้ดังรูปที่ 5.16



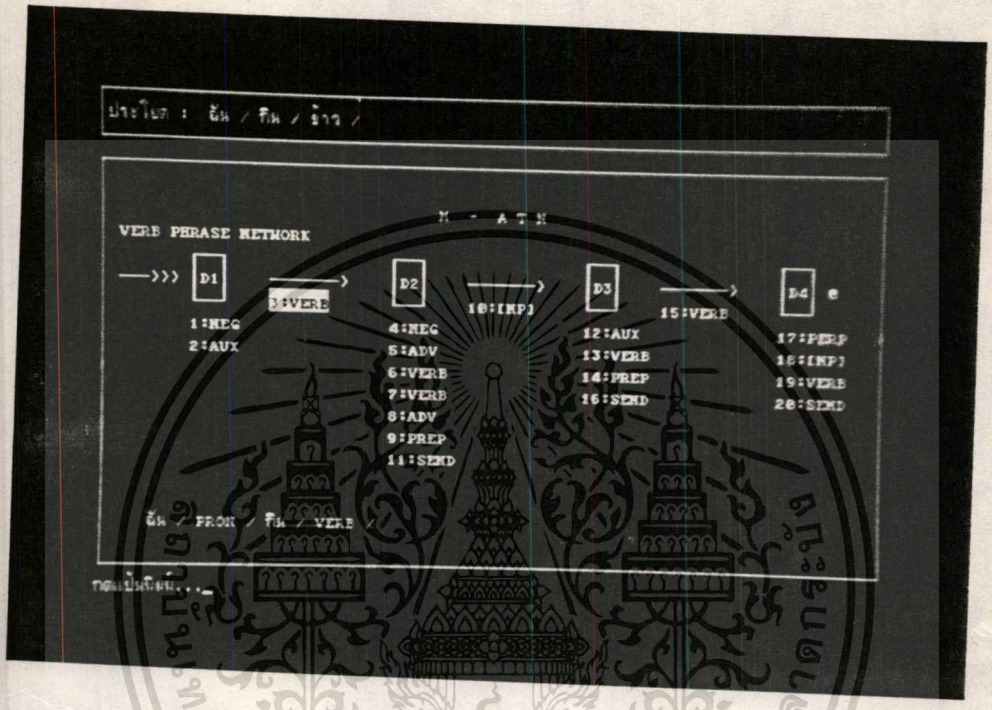
รูปที่ 5.16 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ของระบบ M-ATN ในส่วนของโครงข่ายประโยค

2.3.2 โครงข่ายนามวลี (Noun Phrase) จะแสดงได้ดังรูปที่ 5.17



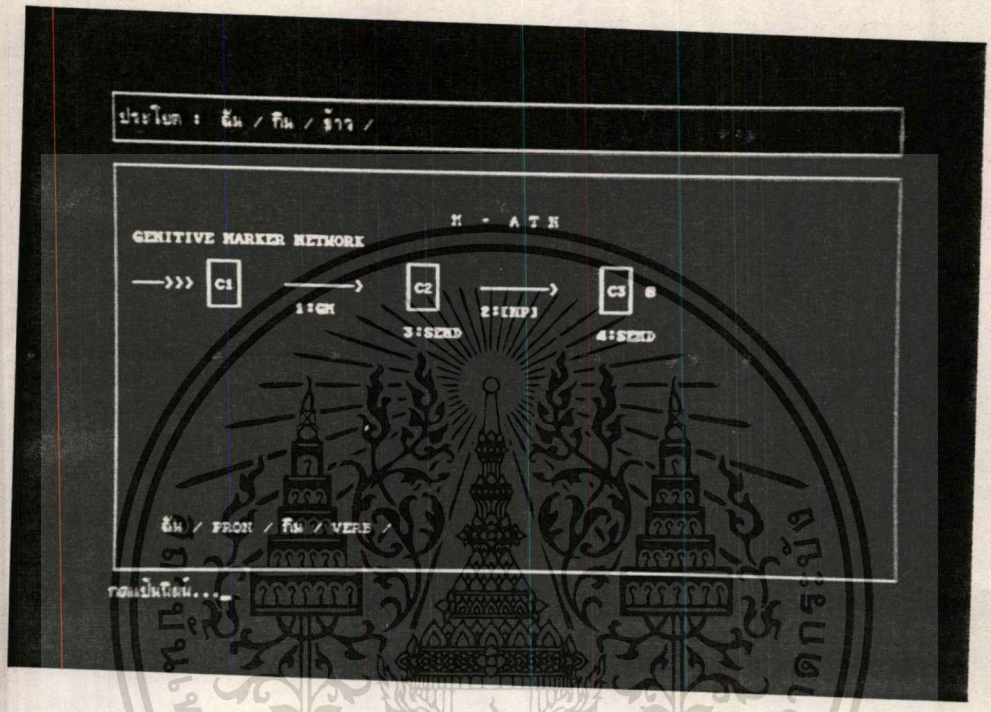
รูปที่ 5.17 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ของระบบ M-ATN ในส่วนของโครงข่ายนามวลี

2.3.3 โครงข่ายกริยา (Verb Phrase) จะแสดงได้ดังรูปที่ 5.18



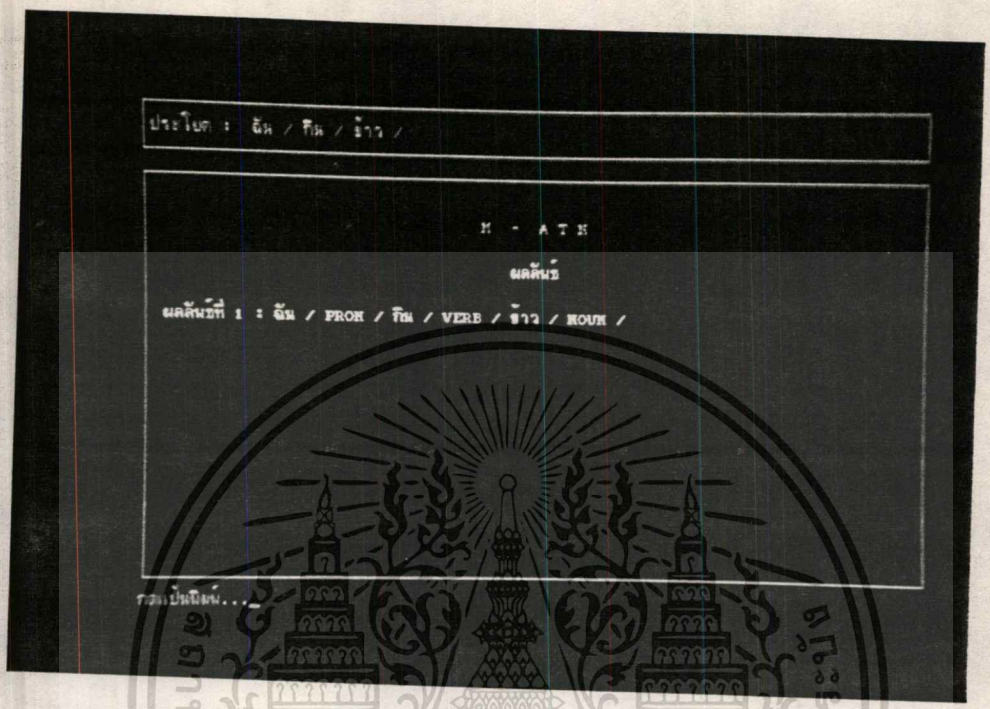
รูปที่ 5.18 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ของระบบ M-ATN ในส่วนของโครงข่ายกริยาวลี

2.3.4 โครงข่าย"ของ" (Genitive Marker) จะแสดงได้ดังรูปที่ 5.19



รูปที่ 5.19 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ของระบบ M-ATN ในส่วนของโครงข่าย "ของ"

และจะแสดงผลลัพธ์ว่าประโยคที่ได้รับการแยกแยะจากหน่วยคำนั้นถูกต้องตามโครงข่าย ดังรูปที่ 5.20 ซึ่งจะแสดงทั้งหน่วยคำและชนิดของหน่วยคำ และนอกจากนี้ยังเป็นผลลัพธ์ของระบบวิเคราะห์โครงสร้างภาษาไทยด้วย



รูปที่ 5.20 แสดงภาพที่ปรากฏบนจอมอนิเตอร์ผลลัพธ์ของระบบ M-ATN

2.4 เมื่อเลือกหมายเลข 4 EXIT โดยการกดแป้นพิมพ์ หมายเลข 4 และกดแป้นพิมพ์ Return หรือ Enter เพื่อต้องการยกเลิกระบบวิเคราะห์โครงสร้างภาษาไทย

5.4 สรุป

ซอฟท์แวร์ที่สร้างขึ้นนั้นจะเห็นว่ามีความอ่อนตัวในแง่ของการที่จะพัฒนา ปรับปรุง เปลี่ยนแปลงอัลกอริทึมในส่วนต่างๆของระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์ของการแยกแยะหน่วยคำ การสร้างฐานความรู้ทางไวยากรณ์ หรืออัลกอริทึมในการประมวลผลทางไวยากรณ์ของภาษาไทย ทำได้ง่ายเนื่องจากส่วนต่างๆของระบบวิเคราะห์โครงสร้างภาษาไทย จะให้ผลลัพธ์ออกมาในรูปของไฟล์ข้อมูล ฉนั้นหากมีการเปลี่ยนแปลงวิธีการหรืออัลกอริทึมใหม่ในส่วนหนึ่งส่วนใดของระบบ (ระบบ Fast Word Matching ระบบ Knowledge Base หรือ ระบบ M-AIN) แล้วให้อยู่ในรูปแบบของข้อมูลเข้าหรือข้อมูลออกเป็นรูปแบบที่กำหนดไว้นั้น ระบบก็ยังคงสามารถทำงานต่อไปได้

การประเมินประสิทธิภาพของระบบ

6.1 บทนำ

จากที่กล่าวมาแล้วทั้งหมดของระบบวิเคราะห์ประโยคด้วยคอมพิวเตอร์ในวิทยานิพนธ์นี้ ซึ่งแบ่งออกเป็น 3 ส่วนใหญ่ ๆ นั้คือ

1. ส่วนแยกแยะหน่วยคำ (Fast Word Matching)
2. ส่วนฐานความรู้ทางไวยากรณ์ภาษาไทย (Knowledge Base of Thai Syntax)
3. ส่วนของการประมวลผลทางไวยากรณ์ (M-ATN)

ทั้ง 3 ส่วนนี้ สามารถที่จะแยกกันทำงานได้อย่างอิสระและสามารถที่จะแก้ไขส่วนของตัวโปรแกรมได้โดยไม่กระทบกระเทือนส่วนอื่นของระบบ เพียงแต่จะต้องมีรูปแบบของข้อมูลเข้าและข้อมูลออกตามที่กำหนดไว้ โดยข้อมูลเข้าเริ่มต้นจะเป็นประโยคภาษาไทย ซึ่งในการออกแบบขั้นต้นนั้นได้มีแนวความคิดว่าเราจะสามารถหาข้อมูลอะไรได้บ้างจากประโยคที่เป็นข้อมูลเข้า เพื่อใช้ในการแยกแยะหน่วยคำออกจากประโยคได้อย่างถูกต้อง และได้ข้อมูลหรือข่าวสารมากที่สุด ซึ่งก็เริ่มต้นทดลองแยกแยะหน่วยคำจากการใช้สูตรในการแยกแยะหน่วยคำ กล่าวคือ วิธีการหาว่าอักษรตัวใดสามารถที่จะรวมกับสระ หรือวรรณยุกต์ตัวใดได้บ้าง แล้วนำมาเปรียบเทียบกับประโยค ถ้าอักขระของประโยครวมกันได้ถูกต้องตามสูตรที่กำหนดไว้ ก็จะแยกแยะหน่วยคำตามสูตรนั้นๆ ซึ่งวิธีการนี้ไม่ใช้พจนานุกรมมาเป็นตัวเปรียบเทียบทำให้ วิธีนี้มีข้อยกเว้นมากมาย มักจะแยกแยะได้เฉพาะคำโดด ยากแก่การพัฒนา และผลลัพธ์ที่ได้จากการแยกแยะหน่วยคำไม่ถูกต้องนัก ต่อมาก็ได้ทดลองใช้วิธี Longest Mapping คือ วิธีการใช้คำศัพท์ในพจนานุกรมไปเปรียบเทียบกับประโยคว่าตรงกันหรือไม่ ถ้าตรงกันจะเลือกเฉพาะคำศัพท์ที่ยาวที่สุด (มีจำนวนอักขระมากที่สุด) เท่านั้น ซึ่งวิธีนี้จะดีกว่าวิธีของการใช้สูตรกล่าวคือคำที่แยกแยะออกจากประโยค

นั้นจะมีความหมายทุกคำ เพราะใช้พจนานุกรมเป็นตัวเปรียบเทียบ แต่ลักษณะของประโยค ภาษาไทยนั้นมีการนำหน่วยคำมาผสมกันเพื่อให้เกิดความหมายใหม่ แต่เมื่อแยกหน่วยคำที่ผสมเหล่านี้ออกจากกันจะมีอีกความหมายหนึ่ง ทำให้การแยกแยะหน่วยคำของวิธี Longest Mapping นี้ ไม่สามารถที่จะแยกหน่วยคำได้ถูกต้อง หรือผิดความหมายไป ดังนั้นระบบการแยกแยะหน่วยคำด้วยวิธี Fast Word Matching จึงได้ถูกพัฒนาขึ้นมาเพื่อการแยกแยะหน่วยคำ โดยพยายามแยกแยะหน่วยคำที่เป็นทั้งคำผสมหรือเป็นคำโดดออกจากประโยค และจะต้องได้ข้อมูลและข่าวสารมากที่สุด ซึ่งการแยกแยะหน่วยคำโดย วิธีนี้จะใช้ข้อมูลจากพจนานุกรมเป็นตัวอ้างอิง ทำให้หน่วยคำที่แยกออกมานั้นมีความหมายทุกคำ แต่ในการที่ต้องการจะแยกแยะหน่วยคำให้ได้ข้อมูลหรือข่าวสารมากที่สุดนั้น ทำให้ในบางครั้งการแยกแยะหน่วยคำออกจากประโยคอาจจะมีผลลัพธ์ของการแยกแยะหน่วยคำได้มากกว่า 1 ผลลัพธ์ ซึ่งการตัดสินใจในการเลือกผลลัพธ์ใดถูกต้องนั้นจะใช้วิธีการประมวลผลทางไวยากรณ์ ซึ่งวิธีนี้จะต้องมีข้อมูลเพิ่มเติมบางส่วนเข้าไปในพจนานุกรมของระบบคือ ชนิดของคำว่าเป็นคำชนิดใด ในส่วนนี้จะเป็นฐานความรู้จากการแบ่งชนิดของคำตามหลักภาษาไทย ซึ่งได้รับการวิเคราะห์จากนักภาษาศาสตร์จำนวนมากแล้ว จากนั้นก็นำฐานความรู้ในส่วนนี้ไปประมวลผลทางโครงสร้างตามระบบ M-ATN ซึ่งเป็นระบบประมวลผลโครงสร้างประโยคภาษาไทย โดยมีการแบ่งออกโครงสร้างออกเป็นส่วนย่อยต่างๆ ตามลักษณะไวยากรณ์ของประโยคภาษาไทย จากการทำงานร่วมกันของระบบ Fast Word Matching ฐานความรู้ทางไวยากรณ์ภาษาไทย และระบบ M-ATN เราจะได้ผลลัพธ์ของการวิเคราะห์โครงสร้างประโยคภาษาไทยได้อย่างถูกต้อง โดยในแต่ละส่วนของระบบวิเคราะห์โครงสร้างประโยคภาษาไทยที่ยังมีข้อจำกัดอยู่บางส่วนคือ

1. ประสิทธิภาพของระบบ Fast Word Matching
   - สามารถรับประโยคที่จะทำการแยกแยะหน่วยคำได้คราวละ 1 ประโยคนั้นจะต้องมีจำนวนอักขระไม่เกิน 80 อักขระ
   - สามารถป้อนข้อมูลทางแป้นพิมพ์
   - สามารถดูผลลัพธ์ของการทำงานได้ทั้งในรูปที่แสดงผลทาง Monitor
   - สามารถเก็บผลลัพธ์ลง Floppy Disk ได้ในลักษณะของ Text File
   - สามารถเรียกผลลัพธ์ออกมาแก้ไขได้โดยแก้ไขบน Word Processor ได้
   - ขณะทำงานสามารถแสดงผลของการเปรียบเทียบคำศัพท์กับประโยคที่เลือกได้
   - ขณะทำงานจะแสดงจำนวนอักขระของประโยคและดรรชนีต่าง ๆ

2. ระบบฐานความรู้ทางไวยากรณ์ไทย

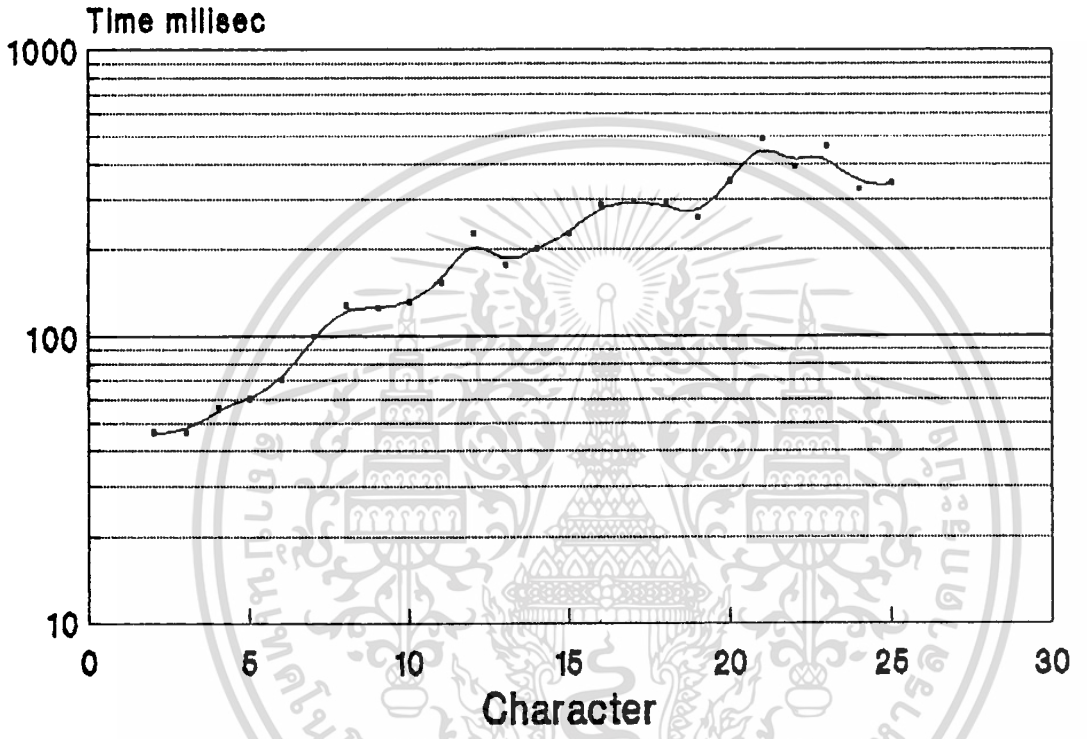
- พจนานุกรมเก็บในรูปของ Text File สามารถแก้ไข ลบ และเพิ่มคำศัพท์ ชนิดของคำ ใน Word Processor ได้
- มีคำศัพท์ประมาณ 10,000 คำ

3. ประสิทธิภาพของระบบ M-ATN

- สามารถวิเคราะห์โครงสร้างประโยคภาษาไทยได้
- สามารถแก้ไขหรือดัดแปลงโครงสร้างได้

6.2 การวิเคราะห์ทางสถิติของการแยกแยะหน่วยคำด้วยระบบ Fast Word Matching

จากการทดลองสุ่มแยกแยะหน่วยคำศัพท์ของประโยคภาษาไทยด้วยระบบ Fast Word Matching จากประโยคที่มีจำนวนอักขระตั้งแต่ 2 อักขระถึง 25 อักขระ โดยกำหนดให้แกน X (แนวนอน) เป็นแกนของจำนวนอักขระ ส่วนแกน Y (แนวตั้ง) เป็นแกนของเวลาดังแสดงได้ดังรูปที่ 6.1 ซึ่งเป็นรูปที่แสดงกราฟการแยกแยะหน่วยคำที่มีอักขระจำนวนต่างๆกัน ด้วยระบบการแยกแยะหน่วยคำ Fast Word Matching



รูปที่ 6.1 แสดงกราฟการแยกแยะหน่วยคำของระบบ Fast Word Matching ต่อหน่วยของเวลา

ข้อมูลกราฟในรูปที่ 6.1 นั้นได้ทดลองบนเครื่อง IBM Compatible AT 286 ที่ความถี่ 8 Mhz จะเห็นได้ว่า เมื่อจำนวนอักขระมากขึ้น เวลาของการแยกกยะหน่วยคำก็จะมากขึ้นตามไปด้วย แต่มีบางช่วงของจำนวนอักขระ ที่เวลาในการแยกแยะหน่วยคำที่มีจำนวนอักขระมากใช้เวลาน้อยกว่าเวลาที่ใช้ในการแยกแยะหน่วยคำที่มีจำนวนอักขระน้อยกว่า เนื่องจากคำศัพท์ของภาษาไทยในช่วงนั้นๆมีจำนวนน้อย

จึงทำให้การค้นหาได้รวดเร็วกว่าช่วงของคำศัพท์อื่นๆ เวลาจึงน้อยลงไปด้วย

6.3 การประยุกต์ใช้งานของระบบ

เป็นที่ทราบกันอยู่แล้วว่าระบบวิเคราะห์โครงสร้างภาษาไทยนี้ ประกอบด้วยส่วนย่อยๆ 3 ส่วน ซึ่งสามารถที่จะแยกแต่ละส่วนออกมาทำงานอย่างอิสระได้แล้วแต่ชนิดของงาน ซึ่งระบบงานที่สามารถนำไปใช้ได้ ได้แก่

6.3.1 การแปลภาษาด้วยคอมพิวเตอร์ (Machine Translation)

ในการแปลภาษาจากภาษาไทยไปเป็นภาษาอื่นๆ นั้น สิ่งสำคัญคือการแยกแยะหน่วยคำได้ถูกต้อง แล้วก็นำมาประมวลผลทางโครงสร้าง และประมวลผลทางด้านความหมายแล้วแปลสู่ภาษาปลายทาง ซึ่งระบบวิเคราะห์โครงสร้างภาษาไทยสามารถนำไปใช้ได้ ในส่วนของการแยกแยะหน่วยคำของประโยคภาษาไทยด้วยระบบ Fast Word Matching และประมวลผลทางโครงสร้างด้วยระบบ M-ATN

6.3.2 การจัดเรียงพิมพ์ของหนังสือพิมพ์ (Letters Setting)

ในการจัดเรียงพิมพ์บนหน้าหนังสือพิมพ์จะมีการแบ่งคำออกเป็นหลายๆ แถว ดังนั้นการจัดเรียงคำให้พอดีในแต่ละแถวนั้น ยังคงใช้คนเป็นผู้จัดเรียงคำให้อยู่พอดีกับแถวที่กำหนด เมื่อมีการจัดเรียงเป็นจำนวนมากๆทำให้ผิดพลาดได้ง่ายและเป็นการล่าช้า หากนำระบบวิเคราะห์โครงสร้างภาษาไทยมาประยุกต์ใช้งานจะทำให้การเรียงพิมพ์รวดเร็วและถูกต้องมาก

6.3.3 งานตรวจสอบคำผิดในงานจดจำตัวอักษร (Pattern Recognition) และการตรวจสอบคำผิดใน Word Processor

ในงานจดจำตัวอักษรจะใช้การเก็บอักษรจากตัวแบบมาอยู่ในรูปของกราฟฟิค แล้วแปลงจากกราฟฟิคมาเป็นตัวอักษรในรูปแบบของไฟล์ข้อความ (Text File) ซึ่งการแปลงจากกราฟฟิคมาเป็นตัวอักษรในรูปแบบของไฟล์ข้อความนั้น หากภาพต้นแบบไม่ชัดเจน หรือขาดหายไปบางส่วนทำให้การแปลงเป็นตัวอักษรในรูปแบบของไฟล์ข้อความผิดพลาดไป ซึ่งการจะหาว่าอักขระที่แปลงมานั้นถูกต้องรึเปล่าสามารถกระทำได้โดยการนำระบบวิเคราะห์โครงสร้างภาษาไทยของงานวิจัยนี้มาตรวจสอบ ซึ่งหากคำใดไม่สามารถที่จะแยกแยะหน่วยคำได้ หรือไม่ถูกต้องตามโครงสร้างที่กำหนด ก็ถือว่าจุดนั้นเป็นคำผิด เหตุผลที่ตัดสินเช่นนั้นได้เพราะระบบวิเคราะห์โครงสร้างภาษาไทยนี้ได้ใช้คำศัพท์ในพจนานุกรมไทยเป็นตัวเปรียบเทียบ

6.4 แนวทางในการพัฒนาต่อของระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์

ระบบวิเคราะห์โครงสร้างภาษาไทยของงานวิจัยนี้ยังสามารถที่จะพัฒนาต่อไปให้มีความสามารถเพิ่มขึ้นได้อีกดังนี้

6.4.1 การวิเคราะห์คราวละหลายๆประโยค เนื่องจากระบบวิเคราะห์โครงสร้างประโยคภาษาไทยนี้ สามารถรับประโยคที่จะวิเคราะห์ได้จากแป้นพิมพ์ทางเดียว และวิเคราะห์ประโยคได้คราวละหนึ่งประโยคเท่านั้น ระบบวิเคราะห์สามารถที่จะพัฒนาให้รับข้อมูลเข้าจาก Floopy Disk, Hard Disk หรือ Megnetic Tape เป็นต้น และรับประโยคที่จะวิเคราะห์ได้มากกว่าหนึ่งประโยค โดยการอ่านประโยคคราวละหลายๆประโยคแล้วเก็บไว้ในหน่วยเก็บความจำชั่วคราว จากนั้นก็นำประโยคที่เก็บไว้ในหน่วยเก็บความจำชั่วคราวมาวิเคราะห์ทีละประโยค

6.4.2 การแยกแยะประโยคออกจากย่อหน้า ในการแยกประโยคออกจากย่อหน้ามีความจำเป็นมากในการศึกษาลักษณะของประโยคที่มีความต่อเนื่องกัน หรือมีความเกี่ยวข้องกันระหว่างประโยคที่กล่าวถึงก่อน กับประโยคที่กล่าวถึงที่หลังทั้งทางด้านโครงสร้างและทางด้านความหมาย เราสามารถที่จะแยกแยะประโยคออกมาจากย่อหน้าได้ โดยวิธีการแยกแยะหน่วยคำ Fast Word Matching แล้วนำหน่วยคำที่แยกออกมาไปตรวจสอบกับโครงข่าย M-ATN เมื่อตรวจสอบถูกต้องแล้วก็ให้ระบบแยกแยะหน่วยคำ Fast Word Matching แยกแยะหน่วยคำคำต่อไป แล้วนำหน่วยคำที่แยกได้ไปตรวจสอบกับโครงข่าย M-ATN อีกจนกระทั่งจบโครงข่ายประโยค ซึ่งการกระทำเช่นนี้ต้องแก้ไขส่วนของระบบ Fast Word Matching ให้สามารถย้อนกลับได้ในกรณีที่ไม่สามารถแยกแยะหน่วยคำ หรือเมื่อตรวจสอบด้วยโครงข่าย M-ATN แล้วได้ผลลัพธ์ไม่ถูกต้อง และในส่วนของโครงข่าย M-ATN นั้นต้องมีการเพิ่มเติมบางส่วนในโครงข่าย เนื่องจากประโยคบางประโยคจะมีการละความหมาย หรือข้อความบางส่วนของประโยคซึ่งกล่าวมาข้างต้น ทำให้ประโยคที่ตรวจสอบกับโครงข่าย M-ATN ผิดพลาดได้

6.5 สรุป

จากอัลกอริทึมที่ได้คิดค้นขึ้นมานี้ ผู้วิจัยคิดว่าเป็นอัลกอริทึมที่ดีที่สุดในขณะนี้ เนื่องจากได้ศึกษาค้นคว้า วิเคราะห์อัลกอริทึมอื่นๆและขอคำแนะนำจากผู้รู้แจ้งทางด้านภาษาศาสตร์โดยตรง นอกจากนั้นผู้วิจัยได้ร่วมวิจัยกับทางประเทศญี่ปุ่นในเรื่องระบบแปลภาษาด้วยคอมพิวเตอร์ ทำให้ได้ทราบปัญหาและแนวทางในการแก้ปัญหาต่างๆเกี่ยวกับภาษาไทย ถึงแม้ว่าแนวทางในวิทยานิพนธ์นี้จะไม่ได้แก้ปัญหาทั้งหมดเกี่ยวกับการประมวลผลภาษาไทยด้วยคอมพิวเตอร์ แต่ก็สามารถที่จะแก้ปัญหาได้บางส่วน ซึ่งผู้วิจัยจะได้พัฒนาแนวทางในการแก้ปัญหาต่างๆต่อไปอีก และปัญหาอีกส่วนหนึ่งก็คือการสร้างโปรแกรมในการสนับสนุนอัลกอริทึมต่างๆที่คิดค้นขึ้น ขณะนี้ใช้ภาษาซีในการเขียนโปรแกรมดังกล่าวบนเครื่องไมโครคอมพิวเตอร์ 16 บิท ซึ่งยังมีความล่าช้าในการทำงานของระบบฮาร์ดแวร์อยู่ ถ้าได้มีการประยุกต์ใช้กับเครื่องคอมพิวเตอร์ที่มีความเร็วสูงขึ้น และใช้เทคนิคโปรแกรมของภาษาสูงขึ้นไปอีก ก็จะได้ซอฟท์แวร์ของระบบ

วิเคราะห์ภาษาไทยที่มีประสิทธิภาพมากยิ่งขึ้นทั้งในด้านความเร็วและความถูกต้อง ผู้วิจัยคาดหวังไว้ว่าแม้งานวิจัยชิ้นนี้จะไม่สามารถครอบคลุมการประมวลผลภาษาไทยด้วยคอมพิวเตอร์ทั้งหมด แต่ก็น่าจะเป็นจุดเริ่มต้นหรือแนวทางที่ดีในการประมวลผลภาษาไทยด้วยคอมพิวเตอร์สำหรับนักวิจัยรุ่นต่อๆไป ในการคิดค้นอัลกอริทึมใหม่ๆ ตลอดจนวิธีการและสร้างโปรแกรมให้ดียิ่งขึ้นในอนาคตอันใกล้นี้

กิตติกรรมประกาศ

ในการทำวิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยดีนั้น ได้รับความช่วยเหลือจากผู้มีพระคุณหลายท่าน ขอกราบขอบพระคุณ คุณพ่อและคุณแม่ของข้าพเจ้าที่ให้การสนับสนุนให้เรียนระดับปริญญาโท และขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.รัตติกร วรากูลศิริพันธุ์ อาจารย์ที่ปรึกษา ผู้ซึ่งให้ความอนุเคราะห์ทั้งด้านความรู้ แนวทาง อุปกรณ์ในการทำงาน เสียสละทั้งแรงกายและแรงใจ และเป็นผู้ให้กำลังใจเมื่อยามท้อแท้ ขอกราบขอบพระคุณอาจารย์ควบคุมการสอบวิทยานิพนธ์ทุกท่านที่สละเวลาที่มีค่ามาควบคุมการสอบ นอกจากนี้ยังขอขอบคุณน้องๆที่ช่วยเหลือทุกคนมาโดยตลอด

หนังสืออ้างอิง

[1] การตรวจสอบตัวสะกดด้วยคอมพิวเตอร์ (Spell Check by Computer); ยืน ภู่วรวรรณ, วิวรรธน์ อึ่มอารมณ์, การประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 10, คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 24-25 พฤศจิกายน 2530

[2] การตัดคำออกจากประโยคด้วยวิธี LONGEST WORD MAPPING; ดร. รัตติกร วรากูลศิริพันธุ์, ดร. จงกล งามวิวิทย์, สมศักดิ์ จันวัน, สุธาทิพย์ จิวัธยากูล, ศักดิ์ชัย ทิพย์จักษุรัตน์, เอกสารประกอบ การประชุมใหญ่ทางวิชาการประจำปี 2532 วิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถัมภ์, กรุงเทพฯ, 26 - 28 ตุลาคม 2532

[3] หลักภาษาไทย; กำชัยทองหล่อม, สำนักพิมพ์ บำรุงสาร์น, 2530

[4] พจนานุกรมไทย (ฉบับภาคอังกฤษ); ดร. วิทย์ เที่ยงบูรณธรรม, สำนักพิมพ์ แพร่วิทยา

[5] พจนานุกรม ฉบับราชบัณฑิตยสถาน; 2525, สำนักพิมพ์ อักษรเจริญทัศน์

[6] Thai Structure; P. Ruengdet,Southeast Asian Language Center Faculty of Graduate Studies Mahidol University, 1979

[7] Panupong, Vichin; Inter - Sentence Relations in Modern Conversational Thai The Siam Society, Bangkok 1970

[8] Warotama Sikkhadit, Udom; Thai Syntax: An Outline Mouton The Hague Paris 1972

[9] โครงสร้างของภาษาไทย; ดร.วิจินตน์ ภาณุพงศ์ ภาควิชาภาษาไทยและภาษาตะวันออก คณะมนุษยศาสตร์ มหาวิทยาลัยรามคำแหง, 2530

[10] โครงสร้างภาษาอังกฤษ; ดร.กรรณิการ์ ช่อไม้ทอง, ภาควิชาภาษาอังกฤษและภาษาศาสตร์ คณะมนุษยศาสตร์ มหาวิทยาลัยรามคำแหง, 2529

[11] Information Structure; B.J. Lings Department of Computer Science University of Excter Chapman and Hall, 1986

[12] Programs and Data Structure in C; Leenderr Ammeraal Hogeschool Utrecht

The Netherlands, JOHN WILEY & SONS, 1987

[13] AN INTRODUCTION TO DATA STRUCTURES WITH APPLICATIONS (second Edition) JEAN - PAUL Tremday, Paul G. Sorenson , McGRAW - HILL INTERNATIONAL BOOK COMPANY, 1984

[15] การวิเคราะห์โครงสร้างประโยคด้วยวิธี M - ATN; ดร. รัตติกร วรากูลศิริพันธุ์, สมศักดิ์ จันวัน, การประชุมทางวิชาการ วิศวกรรมไฟฟ้า 9 สถาบัน ครั้งที่ 11,คณะวิศวกรรมเทคโนโลยี สถาบันเทคโนโลยีราชมงคล 16 - 17 ธันวาคม 2531

[16] การวิเคราะห์โครงสร้างประโยคภาษาไทยด้วยวิธี M - ATN; ดร. รัตติกร วรากูลศิริพันธุ์, สมศักดิ์ จันวัน, ณรงค์ มณีเนตร, การประชุมทางวิชาการวิศวกรรมไฟฟ้า 9 สถาบัน ครั้งที่ 12 ณ.ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์, 16 - 17 พฤศจิกายน 2532

[17] การวิเคราะห์เลือกปรโยคที่ถูกต้องจากความถี่ของการใช้คำ; ดร. รัตติกร วรากูลศิริพันธุ์, สมศักดิ์ จันวัน, วราภรณ์ สุชัยชิต, ศักดิ์ชัย ทิพย์จักษุรัตน์, การประชุมทางวิชาการ วิศวกรรมไฟฟ้า 9 สถาบัน ครั้งที่ 12, ณ.ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์, 16 - 17 พฤศจิกายน 2532

[18] LANGUAGE AS A COGNITIVE PROCESS, Volume I: Syntax Terry Winograd, Stanford University; Adilison Wealey Publishing Company. 1983

[19] Introduction to Natural Language Processing; Marry Dee Harris,Loyola University, New Orleans, Reston Publishing Company, Inc,1985

ตารางแอสกรี

ก 161	ช 162	ค 163	ฆ 164	ง 165	จ 166	ฉ 167
ช 168	ซ 169	ฌ 170	ญ 171	ฎ 172	ฏ 173	ฐ 174
ฑ 175	ฒ 176	ณ 177	ด 178	ต 179	ถ 180	ท 181
ธ 182	น 183	บ 184	ป 185	ผ 186	ฝ 187	พ 188
ฟ 189	ภ 190	ม 191	ย 192	ร 193	ฤ 194	ล 195
ว 196	ศ 197	ษ 198	ส 199	ห 200	ฬ 201	อ 202
ฮ 203	ะ 204	า 205	ำ 206	เ 207	แ 208	โ 209
ใ 211	ไ 212	ๆ 213	ๆ 214	◌ุ 215	◌ู 216	◌ึ 217
◌ี 218	◌ื 219	◌ี 220	◌ั 221	◌่ 222	◌้ 223	◌๊ 224
◌๋ 225	◌์ 226	◌ๆ 227	◌ๅ 228			

ภาคผนวก ข Listing Program TAS.EXE

```
#include <stdio.h>
#include <stdlib.h>

main()
{
     int     t;

     for(;;)
        {
        clrscr();
        gotoxy( 17, 2);  printf("ระบบวิเคราะห์โครงสร้างภาษาไทยด้วยคอมพิวเตอร์");
        gotoxy( 33, 6);  printf("โดย สมศักดิ์ จันวัน");
        gotoxy( 33, 5);  printf("อาจารย์ที่ปรึกษา ผศ.ดร. รัตติกร วรากูลศิริพันธุ์");
        gotoxy( 13, 11); printf( "[1]  Fast Word Matching");
        gotoxy( 13, 12); printf( "[2]  Knowledge Base");
        gotoxy( 13, 13); printf( "[3]  M-ATN");
        gotoxy( 13, 14); printf( "[4]  Exit");
        gotoxy( 10, 20); printf("Input Number 1, 2 , 3 or 4 : ");
        scanf("%d", &t);

        switch(t)
             {
             case(1) : system("fwm_4.exe");
                       break;
             case(2) : system("klb_4.exe");
                       break;
             case(3) : system("square_1.exe");
                       break;
             case(4) : clrscr();
                       exit(0);
             default : printf("Input Error");
                        break;
             }
        }
}
```

```
            a, aa, b, c2, m, mm, mmm, mmtt, start;
    int     start, flag, can_cut, fail, ever, succ;
    char    *t_pos = 0;
    char    *cat_pos = 0;
    char    token[80], buff_cat[80], cata[5], state[5], search_dic[25];
    char    word1[50], word2[5];
    char    sentence[30][50], sentence1[20][50];
    char    sub_sentence1[20][50], sub_sentence2[20][50],
            sub_sentence3[20][50];
    char    temp[MID_STR], test_1[MAX_STR], test_2[MAX_STR],
            test_3[MAX_STR];
    FILE    *hp, *ip, *kp;
    int     char_pos  (char *str, int ch);
    int     remove_wd (char *str, char *temp, int where);
    int     empty_str (char *str);
    char    *str_index;
    void    delete    (char *str, int start, int how_many);
    void    pause(void);
    void    head(int x, int y , int delay_time, char str_demo[80]);
    void    find_index( char str_index[80]);
    void    story(char sent_show[80], char dic_show[80]);
    void    number_of_word( char word_dic[80]);
    void    number_of_sentence( char word[80]);
    void    show_hightlight( char word_sen[80], char word_dic[80]);
    void    choose_dic( char dic_chose[80], int k);
    void    show_table_result(void);
    void    show_result( char sentence[80]);
    void    show_result_t( char sentence[80]);

main()
{
    struct  time   time;
    div_t   total_used_time;
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>

#define  MAX_STR      100
#define  MID_STR      100
#define  SHORT_STR     20
#define  NULL_CHAR      0
#define  BLANK_CHAR    32
#define  TRUE           1
#define  FALSE          0
#define  MASSAGE_0      "กดแป้นพิมพ์..."
#define  MASSAGE_1      "F A S T  W O R D  M A T C H I N G"
#define  MASSAGE_2      "การวิเคราะห์ ... "
#define  MASSAGE_3      "ผลลัพธ์"
#define  MASSAGE_4      "คำศัพท์พจนานุกรม"
#define  MASSAGE_5      "คำศัพท์ที่เลือก"
#define  MASSAGE_6      "ดรรชนีชั้นที่ 1"
#define  MASSAGE_7      "ดรรชนีชั้นที่ 2"
#define  MASSAGE_8      "จำนวนอักขระของประโยค"
#define  MASSAGE_9      "จำนวนอักขระของคำศัพท์"
#define  MASSAGE_10     "ดรรชนีคำศัพท์"
#define  MASSAGE_11     "ข้อมูลหมดแล้ว"
#define  CLEAR           "                                                "

     int       kk, kkk, idx;
     int       a1, a2, a3, a4, a5, a7, a8,
               b1, b2, b3, b4, b5, b6, b7, y1, z1, z2,
```

```
    int     a1, a2, a3, a4, a5, a7, a8, b1, b2, b3, b4, b5, b6, b7;
    int     start, flag, can_cut;
    int     c, d, success, delay_time_h;
    int     sstart_time, sstop_time, mstart_time, mstop_time;
    char    a_buff[20][100], b_buff[20][100], c_buff[20][100],
            e_buff[20][100], f_buff[20][100], x_buff[20][100],
            y_buff[20][100], z_buff[20][100];
    char    str[100];
    FILE    *p1, *p2, *p3, *p4, *p5, *p6, *p7;

state_start:
    p2 = fopen("result_1.dat","w");
    p3 = fopen("result_2.dat","w");
    p4 = fopen("result_3.dat","w");
    p5 = fopen("result_4.dat","w");
/* state_start: */
   clrscr();
   textattr( BLACK);
   textbackground( WHITE);
   textattr( WHITE);
   textbackground( BLACK);
   h_line_input();
   v_line_input();

   delay_time_h = 0;
   a1       = -1;
   a5       = 0;
   flag     = 1;
   start    = 0;

   gotoxy( 2, 2);
   cprintf("ใส่ประโยค :");
```

```
head(  3,  5, delay_time_h, MASSAGE_1);
head( 55,  5, delay_time_h, MASSAGE_10);
head( 13,  6, delay_time_h, MASSAGE_2);
head( 12, 12, delay_time_h, MASSAGE_4);
head( 55, 12, delay_time_h, MASSAGE_5);
head( 43,  7, delay_time_h, MASSAGE_6);
head( 43,  8, delay_time_h, MASSAGE_7);
head( 43,  9, delay_time_h, MASSAGE_8);
head( 43, 10, delay_time_h, MASSAGE_9);
kk      = 0;
kkk     = 0;
idx     = 0;

 for( d=0; d <= 20; d++)
    {
     strcpy(a_buff[d],"");
     strcpy(b_buff[d],"");
     strcpy(c_buff[d],"");
     strcpy(e_buff[d],"");
     strcpy(f_buff[d],"");
     strcpy(x_buff[d],"");
     strcpy(y_buff[d],"");
     strcpy(z_buff[d],"");
    }

  gotoxy( 14, 2);
  gets(c_buff[0]);
  if( strlen(c_buff[0]) <= 1)
    {
     cprintf("ประโยคไม่ถูกต้อง");
     fclose(p2);
     clrscr();
     exit(0);
```

```
        }

state_1:

    a2 = -1;
    a3 = -1;
    a4 = 0;
    a8 = 0;

state_2:

    a2++;
    if( a2 > a5)
      {
      if( flag != 0)
        {
        while( a4 <= a1)
             {
        if( strlen(f_buff[a4]) > 0)
          {
          strcpy(a_buff[a8], e_buff[a4]);
          strcpy(x_buff[a8], y_buff[a4]);
          strcat(a_buff[a8], "  /  ");
          strcat(x_buff[a8], "_");
          strcpy(c_buff[a8], f_buff[a4]);
          fprintf( p3, "%s\n", a_buff[a8]);
          a8++;
          }

        if( strlen(f_buff[a4]) == 0)
          {
          strcpy( a_buff[a4], e_buff[a4]);
          strcpy( x_buff[a4], y_buff[a4]);
```

```
        if( strlen( a_buff[a4]) != 0)
          {
          fprintf( p2, "%s /\n", a_buff[a4]);
          fprintf( p3, "%s /\n", a_buff[a4]);
          fprintf( p4, "%s .\n", a_buff[a4]);
          fprintf( p5, "%s.\n", x_buff[a4]);
          }
        }
       a4++;
/*       b6 = a4-1;*/
       }
   for( d = 0; d <= a1; d++)
      {
      if (strlen(f_buff[d]) != 0)     success = 1;
      }
       if (success == 0)   goto state_end;
       a5 = a8-1;
       flag = 0;
       success = 0;
       a1 = -1;
       goto state_1;
       }
   }
   else
   {
   strcpy(test_1,c_buff[a2]);
   strcpy(test_2,test_1);
   strcpy(test_3,test_1);
   a3++;
   b1 = strlen(test_1);
   can_cut = 0;
   if( b1 > 0)
     {flag = 1;}
```

```
else
  {
  flag = 0;
  goto state_2;
  }

switch(test_1[0])
      {
   case 'ก':       p1 = fopen("a1.dic","r");
         break;
   case 'ข':       p1 = fopen("a2.dic","r");
         break;
   case 'ค':       p1 = fopen("a3.dic","r");
         break;
   case 'ง':       p1 = fopen("a4.dic","r");
         break;
   case 'จ':       p1 = fopen("a5.dic","r");
         break;
   case 'ฉ':       p1 = fopen("a6.dic","r");
         break;
   case 'ช':       p1 = fopen("a7.dic","r");
         break;
   case 'ซ':       p1 = fopen("a8.dic","r");
         break;
   case 'ญ':       p1 = fopen("a33.dic","r");
         break;
   case 'ฐ':       p1 = fopen("a35.dic","r");
         break;
   case 'ด':       p1 = fopen("a9.dic","r");
         break;
   case 'ต':       p1 = fopen("a10.dic","r");
         break;
   case 'ถ':       p1 = fopen("a11.dic","r");
```

```
        break;
case 'ท':       p1 = fopen("a12.dic","r");
        break;
case 'ธ':       p1 = fopen("a13.dic","r");
        break;
case 'น':   p1 = fopen("a14.dic","r");
        break;
case 'บ':       p1 = fopen("a15.dic","r");
        break;
case 'ป':       p1 = fopen("a16.dic","r");
        break;
case 'ผ':       p1 = fopen("a17.dic","r");
        break;
case 'ฝ':       p1 = fopen("ax.dic","r");
        break;
case 'พ':       p1 = fopen("a18.dic","r");
        break;
case 'ฟ':       p1 = fopen("a19.dic","r");
        break;
case 'ภ':       p1 = fopen("a20.dic","r");
        break;
case 'ม':       p1 = fopen("a21.dic","r");
        break;
case 'ย':       p1 = fopen("a22.dic","r");
        break;
case 'ร':     p1 = fopen("a23.dic","r");
        break;
case 'ฤ':       p1 = fopen("a24.dic","r");
        break;
case 'ล':   p1 = fopen("a25.dic","r");
        break;
case 'ว':   p1 = fopen("a26.dic","r");
        break;
```

```
    case 'ศ':   p1 = fopen("a27.dic","r");
          break;
    case 'ส':   p1 = fopen("a28.dic","r");
          break;
    case 'ห':       p1 = fopen("a29.dic","r");
          break;
    case 'อ':       p1 = fopen("a30.dic","r");
          break;
    case 'ฮ':       p1 = fopen("a31.dic","r");
          break;
    case 'ะ':       p1 = fopen("asara_a.dic","r");
          break;
    case 'เ':       p1 = fopen("asara_e.dic","r");
          break;
    case 'โ':       p1 = fopen("asara_i.dic","r");
          break;
    case 'ๅ':       p1 = fopen("asara_o.dic","r");
          break;
    case 'ไ':       p1 = fopen("asara_u.dic","r");
          break;
      default  :
              can_cut = 0;
                goto state_3;
    }
    }

/* ***************************  Compare Word  ************************
    fscanf(p1,"%s",str);
    while( str[0] != 'q')
    {
       if( str[1] < test_1[1])   goto state_3;
       if( str[1] > test_1[1])   goto ABC;
       b2 = strlen(str);
```

```
    b3 = strlen( test_1);
    if( b2 > b3)    goto ABC;
if( str[0] == test_1[0] && str[1] == test_1[1])
  {
    story( test_1, str);
    if( strncmp( test_1, str, b2) == 0)
      {
       b4 = remove_wd(test_1,temp,b2-1);
       choose_dic( str, kk);
       kk++;
       a1++;
       can_cut = 1;
       if( start == 0)
         {
         strcpy( b_buff[a1], temp);
         strcpy( f_buff[a1], test_1);
         strcpy( e_buff[a1], b_buff[a1]);
         strcpy( y_buff[a1], b_buff[a1]);
         strcpy( test_1, test_2);
         start = 1;
         }
       else
         {
         strcpy( b_buff[a1], a_buff[a3]);
         strcpy( z_buff[a1], x_buff[a3]);
         strcat( b_buff[a1], temp);
         strcat( z_buff[a1], temp);
         strcpy( f_buff[a1], test_1);
         strcpy( test_3, test_1);
         strcpy( e_buff[a1], b_buff[a1]);
         strcpy( y_buff[a1], z_buff[a1]);
         strcpy( test_1, test_2);
         }
```

```
        }
    }

ABC:    fscanf( p1, "%s", str);
  }

state_3:

  if (can_cut == 0)
    {
      strcpy(f_buff[a2],"");          /* f_buff ประโยคที่เหลือ */
      strcpy(e_buff[a2],"");          /* e_buff ประโยคที่ตัดได้ */
    }

  fclose(p1);
  can_cut = 0;
  goto state_2;

state_end:

   fprintf( p2, "q");
   fprintf( p3, "q");
   fprintf( p4, "q");
   fprintf( p5, "q");
   fclose(p2);
   fclose(p3);
   fclose(p4);
   fclose(p5);

   pause();
   for( b4 = 0; b4<8; b4++)      choose_dic( "              ", b4);
   show_table_result();
```

```
    textattr( WHITE);
    textbackground( BLACK);

     p2 = fopen("result_1.dat","r");
     for(;;)
        {
        fgets( str, 80, p2);
        if( strcmp( str, "q") == 0)       goto AZ;
        else                  show_result( str);
        }
AZ:
     fclose(p2);
     show_result_t( CLEAR);
     show_result_t( MASSAGE_11);
     pause();
/*     goto state_start;  */
}
/* ++++++++++++++++++++  remove word   ++++++++++++++++++++ */
int   remove_wd (char *str, char *temp, int where)
{
         int    index;
   int   char_pos(char *str,int);
   int    i;

   while ( !char_pos(str,BLANK_CHAR))
      delete(str,1,1);

   if ( where >= 0)
      {
        for (index = 0; index <= where; index = index + 1)
      {
        temp[index] = str[index];
      }
```

```
        temp[index] = NULL_CHAR;
        delete(str, 1, where + 1);
       }
     else
       {
        strcpy (temp, str);
        str[0] = NULL_CHAR;
           }
}

/* ++++++++++++++++++  find position of word  ++++++++++++++++++++ */
int   char_pos (char *str, int ch)
{
   int    index;

   for ( index = 0; (str[index] != ch) && (str[index] != NULL_CHAR)
        && (str[index] != '.');
        index = index + 1);
   if (str[index] == NULL_CHAR)
   {
      return(-1);
   }
    else
    {
               return(index);
    }
}

/* ++++++++++++  delete word from sentence  +++++++++++++++++ */
void delete (char *str, int start, int how_many)
{
   int   index1, index2;
```

```
    if (start < 1)  start = 1;
    if (start + how_many -1 > strlen (str))
        how_many = strlen(str) + 1 - start;

    for ( index1 = start - 1,
                index2 = start - 1 + how_many;
          str[index2] != NULL_CHAR;
          index1++, index2++)
          {
       str[index1] = str[index2];
           }
          str[index1] = NULL_CHAR;
}

void   choose_dic( char dic_chose[80], int k)
{
   gotoxy( 43+kkk, 14+k);
   printf( "%s     ", dic_chose);
   if( k == 7)
     {
     kk   =   -1;
     kkk  =  kkk + 15;
     }
}

void   story(char sent_show[80], char dic_show[80])
{
   find_index( sent_show);
   gotoxy( 12, 15);
   printf("                              ");
   delay( 0);
   gotoxy( 12, 15);
```

```
    printf("%s", dic_show);

    number_of_word( dic_show);

    textattr(WHITE);
    textbackground(BLACK);
/*  gotoxy( 12, 9); */
    gotoxy( 4, 9);
    cprintf("                              ");

    show_hightlight( sent_show, dic_show);
    delay(0);
}

void   show_hightlight( char word_sen[80], char word_dic[80])
{
    int   i, j;

    j = strlen( word_dic);
    for( i=0; i<j; i++)
       {
       textattr(BLACK);
       textbackground(WHITE);
       gotoxy( 12+i, 9);
       cprintf("%c", word_sen[i]);
       }
/* ,, */
       textattr( WHITE);
       textbackground( BLACK);

    number_of_sentence( word_sen);
    gotoxy( 12, 9);
    printf("%s", word_sen);
```

```
    delay(0);
  }

void   number_of_word( char word_dic[80])
{
    int   i;

    i = strlen( word_dic);
    gotoxy( 67, 10);
    printf("%d   ", i);
}

void   number_of_sentence( char word[80])
{
    int   i;

    i = strlen( word);
    gotoxy( 67, 9);
    printf("%d  ", i);
}

void   find_index(char str_index[80])
{
    int   i;

    gotoxy( 60, 7);
    printf("%c", str_index[0]);
    gotoxy( 60, 8);
    printf("%c", str_index[1]);
}

void      pause(void)
{
```

```
    int   c;

    gotoxy( 1, 23);
    printf("%s", MASSAGE_0);
    getch();
    gotoxy( 1, 23);
    printf("                    ");
}

void    head(int x, int y , int delay_time, char str_demo[80])
{
    int   i, j;

    j = strlen(str_demo);

    for( i=0; i<j; i++)
    {
    gotoxy( x+i, y);
    cprintf("%c", str_demo[i]);
    delay(delay_time);
    }
}

h_line_input()
{
    int   i;

    for( i = 2; i<80; i++)
       {
       gotoxy(i,1);
       printf("-");
       gotoxy(i,3);
```

```
        printf("-");
        }

    for( i = 2; i<40; i++)
        {
        gotoxy(i,4);
        printf("-");
        gotoxy(i,22);
        printf("-");
        }

    for( i = 41; i<80; i++)
        {
        gotoxy(i,4);
        printf("-");
        gotoxy(i,22);
        printf("-");
        }
}

v_line_input()
{
    int   i;

    gotoxy(1,1);        printf("┌");
    gotoxy(80,1);         printf("┐");
    gotoxy(1,3);          printf("└");
    gotoxy(80,3);         printf("┘");
    gotoxy(1,2);          printf("│");
    gotoxy(80,2);         printf("│");

    for( i=5; i<23; i++)
         {
```

```
        gotoxy(1,i);        printf("│");
        gotoxy(40,i);       printf("│");
         }

    gotoxy(1,4);        printf("┌");
    gotoxy(40,4);         printf("┐");
    gotoxy(1,22);         printf("└");
    gotoxy(40,22);        printf("┘");

    for( i=5; i<23; i++)
         {
        gotoxy(41,i);         printf("│");
        gotoxy(80,i);         printf("│");
         }

    gotoxy(41,4);       printf("┌");
    gotoxy(80,4);         printf("┐");
    gotoxy(41,22);        printf("└");
    gotoxy(80,22);        printf("┘");

}

void   show_table_result(void)
{
   int   i;

    for( i=9; i<23; i++)
       {
       gotoxy(1,i);        printf("│");
       gotoxy(80,i);        printf("│");
       }
    gotoxy(1,4);        printf("┌");
    gotoxy(80,4);               printf("┐");
```

```
    gotoxy(1,22);       printf("└");
    gotoxy(80,22);              printf("┘");

    for( i = 2; i<80; i++)
       {
       gotoxy(i,4);
       printf("-");
       gotoxy(i,22);
       printf("-");
       }

    for( i=5; i<22; i++)
       {
       gotoxy( 2, i);
       printf("                                                ");
       gotoxy( 35, i);
       printf("                                                ");
       }

    textattr(WHITE);
    textbackground(BLACK);
    gotoxy( 12, 9);
    printf("                    ");

    head( 23,  5, 0, MASSAGE_1);
    head( 36,  7, 0, MASSAGE_3);
}

void    show_result( char sentence[80])
{
    gotoxy( 2, 9+idx);
    cprintf("%s %d : ", MASSAGE_3, idx+1);
    cprintf("%s", sentence);
```

```
    idx++;
}

void    show_result_t( char sentence[80])
{
    textattr( WHITE);
    textbackground( BLACK);
    gotoxy( 2, 9+idx);
    cprintf("                                        ");
    gotoxy( 2, 9+idx);
    printf("%s", sentence);
    idx++;
}
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include "line.h"

#define MAX_STR     100
#define MID_STR      50
#define NULL_CHAR     0
#define BLANK_CHAR   32
#define TRUE          1
#define FALSE         0
#define MASSAGE1     "ผลลัพธ์ที่ "
#define MASSAGE2     "K N O W L E D G E   B A S E"

char     super_sentence[10][10][80];
char     temp[50], token[80], str1[80], str2[80];
char     sentence[20][50], sentence1[20][50], sentence_s[20][50];
char    sub_sentence1[20][50], sub_sentence2[20][50], sub_sentence3[20][50];
char     *t_pos = 0;
char    *cat_pos = 0;
char    buff_cat[80];
char    cata[5];
char    state[5];
char    search_dic[25];
int     ever, count, succ, fall;
FILE    *hp, *ip, *kp, *lp;
int     a, aa, ax, b, c2, m, mm, mmm, mmtt, start;
int     ever_sentence, number_sent = 1, num_sentence_buff;
int     t;

main()
```

```
{
     char     str[100];
     int           i, j;

    ip = fopen("result_4.dat","r");
    kp = fopen("result_3.dat","w");
    lp = fopen("result_1.dat","r");
    clrscr();
    h_line_input();
    v_line_input();
    show_table_result();
    textattr( WHITE);
    textbackground( BLACK);

    start = 0;
    c2    = 13;
    a     = 0;
    ax    = 0;

    strcpy( str, "STRUCTURE");
    strcpy( str, "SHOW_STRUCTURE");

    do
      {
     fscanf( ip, "%s", str);
     if( strcmp( str[0], 'q') == 0)      goto AAA;
     if ( strlen(str) != 1)       strcpy( sentence[a], str);
     a++;
      } while( strlen(str) > 1);

AAA:
     count = a;
    do
```

```
 {
fgets( str, 80, lp);
if( strcmp( str[0], 'q') == 0)      goto BBB;
if (strlen(str) != 1)       strcpy( sentence_s[ax], str);
ax++;
 } while( strlen(str) > 1);


BBB:
     for( aa = 0; aa < count; aa++)
       {
         head( 2, 2, 0, MASSAGE1);
         gotoxy( 13, 2);
/*       strcat( sentence_s[aa], "                                    ");   */
         cprintf("%d :  %s", aa+1, sentence_s[aa]);
         t_pos = sentence[aa];
         strcpy( token, "_");
         ever_sentence = 0;
         while( strlen(t_pos) > 1)
             {
              get_token();
              find_cat();
              cpy_sent_to_buff0();
            if( strcmp( t_pos[0], '.') == 0)
              {
               for( t=0; t< number_sent; t++)
                  {
                  show_result( super_sentence[1][t]);
                /* cprintf( "\n%s", super_sentence[1][t]); */
                  fprintf( kp,  "%s\n", super_sentence[1][t]);
                  strcpy( super_sentence[0][t], "");
                  strcpy( super_sentence[1][t], "");
                  }
                  number_sent = 1;
```

```
                }
            }
        }
        fprintf(kp,"q");
        fclose(ip);
        fclose(kp);
        fclose(lp);
        show_result_t( MASSAGE_11);
        pause();
        exit(0);
}

get_token()
{
      char      *p;

      p = token;
      if( *t_pos == '_')  *t_pos++;
      if( *t_pos == '.')
        {
          *p++ = '.';
          *p = '\0';
          return 1;
        }
 while( *t_pos != ' ' && *t_pos != '.' && *t_pos != '_' && *t_pos != 'q')
           {
           *p = *t_pos++;
           p++;
           }
           *p = '\0';
}

s_cat()
```

```
{
      char      *r;
      r = buff_cat;
      if( *cat_pos == '_')  *cat_pos++;
      if( *cat_pos == '2')  *cat_pos++;
      if( *cat_pos == '.')
        {
          *r++ = '.';
          *r = '\0';
          return 1;
         }
      while( *cat_pos != ' ' && *cat_pos != '.' && *cat_pos != '_'
          && *cat_pos != 'q' && *cat_pos != '')
            {
            *r = *cat_pos++;
            r++;
            }
            *cat_pos++;
            *r = '\0';
}

find_cat()
{
      int  b2, flag, b3, t, tt, t3;

      switch( token[0])
      {
         case 'ก':        hp = fopen("1.dic","r");
                      break;
         case 'ข':        hp = fopen("2.dic","r");
                      break;
         case 'ค':        hp = fopen("3.dic","r");
                      break;
```

```
case 'ง':       hp = fopen("4.dic","r");
                break;
case 'จ':       hp = fopen("5.dic","r");
                break;
case 'ฉ':       hp = fopen("6.dic","r");
                break;
case 'ช':       hp = fopen("7.dic","r");
                break;
case 'ซ':       hp = fopen("8.dic","r");
                break;
case 'ญ':       hp = fopen("33.dic","r");
                break;
case 'ฐ':       hp = fopen("35.dic","r");
                break;
case 'ด':       hp = fopen("9.dic","r");
                break;
case 'ต':       hp = fopen("10.dic","r");
                break;
case 'ถ':       hp = fopen("11.dic","r");
                break;
case 'ท':       hp = fopen("12.dic","r");
                break;
case 'ธ':       hp = fopen("13.dic","r");
                break;
case 'น':       hp = fopen("14.dic","r");
                break;
case 'บ':       hp = fopen("15.dic","r");
                break;
case 'ป':       hp = fopen("16.dic","r");
                break;
case 'ผ':       hp = fopen("17.dic","r");
                break;
case 'ฝ':       hp = fopen("x.dic","r");
```

```
            break;
case 'พ':       hp = fopen("18.dic","r");
            break;
case 'ฟ':       hp = fopen("19.dic","r");
            break;
case 'ภ':       hp = fopen("20.dic","r");
            break;
case 'ม':       hp = fopen("21.dic","r");
            break;
case 'ย':       hp = fopen("22.dic","r");
            break;
case 'ร':       hp = fopen("23.dic","r");
            break;
case 'ฤ':       hp = fopen("24.dic","r");
            break;
case 'ล':       hp = fopen("25.dic","r");
            break;
case 'ว':       hp = fopen("26.dic","r");
            break;
case 'ศ':       hp = fopen("27.dic","r");
            break;
case 'ส':       hp = fopen("28.dic","r");
            break;
case 'ห':       hp = fopen("29.dic","r");
            break;
case 'อ':       hp = fopen("30.dic","r");
            break;
case 'ฮ':       hp = fopen("31.dic","r");
            break;
case 'เ':       hp = fopen("sara_a1.dic","r");
            break;
case 'แ':       hp = fopen("sara_e.dic","r");
            break;
```

```
    case 'ิ':       hp = fopen("sara_i.dic","r");
                    break;
    case 'ี':       hp = fopen("sara_o.dic","r");
                    break;
    case 'ึ':       hp = fopen("sara_u.dic","r");
                    break;
    default  :  goto state_0;
}

fscanf( hp, "%s %s", str1, str2);
        flag = 0;
        b = 0;
  while( str1[0] != 'q')
       {
     if( str1[0] < token[0])   goto state_0;
     if( strcmp( token, str1) == 0)
       {
/*       IF FIRST WORD ever_sentence IS 0 ELSE IS 1          */
/*       if ( ever_sentence == 0 )     ever_sentence = 1;    */

     switch( str2[0])
          {
           case '2' : b = 0;
                      cat_pos = str2;
                     for( t= 0; t<=1; t++)
                        {
                         s_cat();
                         strcpy( sentence1[t], str1);
                         strcat( sentence1[t], "   ");
                         strcat( sentence1[t], buff_cat);
                         b++;
                        }
                    break;
```

```
                default  : strcpy( sentence1[0], str1);
                           strcat( sentence1[0], "   ");
                           strcat( sentence1[0], str2);
                           b = 1;
                           break;
            }
            flag = 1;
            goto state_0;
          }
      fscanf(hp,"%s %s", str1, str2);
    }

state_0:
        fclose(hp);
        if( flag != 1)   strcpy( cata, " FAIL..");
        flag = 0;
        return;
    }

cpy_sent_to_buff0()
{
      int     i, j, t;

      t = 0;

      for( i=0; i< number_sent; i++)
         {
         for( j=0; j<b; j++)
             {
             if( ever_sentence == 0)
                {
                strcpy( super_sentence[0][j], sentence1[j]);
                }
```

```
            else
               {
               strcat( super_sentence[0][i], "  ");
               strcat( super_sentence[0][i], sentence1[j]);
               }
            }
         }
      for( i=1; i<= b*number_sent; i++)
         {
         strcpy( super_sentence[1][i-1], super_sentence[0][i-1]);
/*         strcpy( sentence1[i-1], super_sentence[1][i-1]);  */
         num_sentence_buff = i;
         ever_sentence = 1;
         }
         number_sent = num_sentence_buff;
/*      for( i=0; i< number_sent; i++)
         {
         for( j=0; j<b; j++)
            {
            strcpy( super_sentence[0][t], super_sentence[1][i]);
            t++;
            }
         } */
}
```

```
#include <stdio.h>
#include "line.h"

main()
{
    int    t, y;

    clrscr();
    textattr( WHITE);
    textbackground( BLACK);
    h_line_input();                /* ตีกรอบในแนวนอน */
    v_line_output();               /* ตีกรอบในแนวตั้ง */
    head( 2, 2, 0, "ประโยค :  ฉัน / กิน / ข้าว /");
    head( 35, 6, 0, "M  -  A T N");
    sentence_network();
    textattr( BLACK);
    textbackground( WHITE);
    gotoxy( 18, 10);    cprintf( "%s", sent_cat_a1a2[0]);
    head( 6, 20, 0, "ฉัน / PRON /");
    pause();

    textattr( WHITE);
    textbackground( BLACK);
    np_network();
    textattr( BLACK);
    textbackground( WHITE);
    gotoxy( 18, 12);    cprintf( "%s", np_cat_b1b2[2]);
    pause();

    textattr( WHITE);
    textbackground( BLACK);
```

```
gm_network();
head( 6, 20, 0, "ฉัน / PRON / กิน / VERB /");
pause();

textattr( WHITE);
textbackground( BLACK);
vp_network();
textattr( BLACK);
textbackground( WHITE);
gotoxy( 18, 10);    cprintf( "%s", vp_net_d1d2[0]);
head( 6, 20, 0, "ฉัน / PRON / กิน / VERB /");
pause();

textattr( WHITE);
textbackground( BLACK);
vp_network();
textattr( BLACK);
textbackground( WHITE);
gotoxy( 38, 10);    cprintf( "%s", vp_net_d2d3[0]);
head( 6, 20, 0, "ฉัน / PRON / กิน / VERB / ข้าว / NOUN /");
pause();

textattr( WHITE);
textbackground( BLACK);
np_network();
textattr( BLACK);
textbackground( WHITE);
gotoxy( 18, 11);    cprintf( "%s", np_cat_b1b2[1]);
pause();

textattr( WHITE);
textbackground( BLACK);
clear_square();
```

```
    gotoxy( 6, 20);  printf("                                                    ");
    head( 38, 8, 0, "ผลลัพธ์");
    head( 3, 10, 0, "ผลลัพธ์ที่ 1 : ฉัน / PRON / กิน / VERB / ข้าว / NOUN /");
    pause();
}

sentence_network()
{
    int    t, y;

    t=5;
    position_square( t);
    position_line( t);
    position_label( sentence_net, t);

/************************** NODE A1 ****************************/
    gotoxy( 10, 11);    cprintf( "%s", sent_cat_a1[0]);
    gotoxy( 10, 12);    cprintf( "%s", sent_cat_a1[1]);
    gotoxy( 10, 13);    cprintf( "%s", sent_cat_a1[2]);

/************************** NODE A1A2 ****************************/
    gotoxy( 18, 10);    cprintf( "%s", sent_cat_a1a2[0]);
    for( y=0; y<=2; y++)
       {
       textattr( BLACK);
       textbackground( WHITE);
       gotoxy( 10, 11+y);    cprintf( "%s", sent_cat_a1[y]);
       textattr( WHITE);
       textbackground( BLACK);
       gotoxy( 10, 11+y);    cprintf( "%s", sent_cat_a1[y]);
       }
       gotoxy( 18, 10);    cprintf( "%s", sent_cat_a1a2[0]);
```

```
    for( y=0; y<=2; y++)
       {
       gotoxy( 30, 11+y);    cprintf( "%s", sent_cat_a2[y]);
       }
       gotoxy( 38, 10);    cprintf( "%s", sent_cat_a2a3[0]);

    for( y=0; y<5; y++)
       {
       gotoxy( 50, 11+y);    cprintf( "%s", sent_cat_a3[y]);
       }
       gotoxy( 58, 10);    cprintf( "%s", sent_cat_a3a4[0]);

    for( y=0; y<=1; y++)
       {
       gotoxy( 70, 11+y);    cprintf( "%s", sent_cat_a4[y]);
       }
}

np_network()
{
    int    t, y;

    t=4;
    clear_square();
    position_label( np_net, t);
    position_square( t);
    position_line( t);
    gotoxy( 10, 11);    cprintf( "%s", np_cat_b1[0]);

    for( y=0; y<=2; y++)
       {
       gotoxy( 18, 10+y);    cprintf( "%s", np_cat_b1b2[y]);
       }
```

```
    for( y=0; y<=6; y++)
       {
       gotoxy( 30, 11+y);     cprintf( "%s", np_cat_b2[y]);
       }

    for( y=0; y<=1; y++)
       {
       gotoxy( 38, 10+y);     cprintf( "%s", np_cat_b2b3[y]);
       }

    for( y=0; y<3; y++)
       {
       gotoxy( 50, 11+y);     cprintf( "%s", np_cat_b3[y]);
       }
}

gm_network()
{
    int    t, y;

    t=4;
    clear_square();
    position_label( gm_net, t);
    position_square( t);
    position_line( t);
    gotoxy( 19, 10);     cprintf( "%s", gm_net_c1[0]);
    gotoxy( 38, 10);     cprintf( "%s", gm_net_c1c2[0]);
    gotoxy( 30, 11);     cprintf( "%s", gm_net_c2[0]);
    gotoxy( 50, 11);     cprintf( "%s", gm_net_c2c3[0]);
}

vp_network()
{
```

```
    int    t, y;

    t=5;
    clear_square();
    position_label( vp_net, t);
    position_square( t);
    position_line( t);

    for( y=0; y<=1; y++)
       {
       gotoxy( 10, 11+y);    cprintf( "%s", vp_net_d1[y]);
       }
       gotoxy( 18, 10);    cprintf( "%s", vp_net_d1d2[0]);

    for( y=0; y<7; y++)
       {
       gotoxy( 30, 11+y);    cprintf( "%s", vp_net_d2[y]);
       }
       gotoxy( 38, 10);    cprintf( "%s", vp_net_d2d3[0]);


    for( y=0; y<4; y++)
       {
       gotoxy( 50, 11+y);    cprintf( "%s", vp_net_d3[y]);
       }
       gotoxy( 58, 10);    cprintf( "%s", vp_net_d3d4[0]);


    for( y=0; y<4; y++)
       {
       gotoxy( 70, 11+y);    cprintf( "%s", vp_net_d4[y]);
       }
}
```

```
position_line( int line)        /* เขียนลูกศรบอกทิศทางมีจำนวนเท่ากับ line */
{
    int    t;

    for( t=3; t<=(line-1)*20; t=t+20)
       {
       if( t== 3)    { gotoxy( t, 9);    cprintf( "——>>>");   }
       else          { gotoxy( t-5, 9);   cprintf( "————>"); }
       }
                     gotoxy( t-8, 9);    cprintf( "@");
}

position_square( int node)      /* จำนวน node ที่สร้างสร้างขึ้น */
{
    int    t;

    for( t=10; t<(node*20)-10; t=t+20)
       {
       square( t, 8);
       }
}

square( int x, int y)                  /* สร้าง node รูปสี่เหลี่ยม */
{
    gotoxy( x, y);        cprintf( "——");
    gotoxy( x, y+2);      cprintf( "——");
    gotoxy( x, y);        cprintf( "┌");
    gotoxy( x, y+1);      cprintf( "│");
    gotoxy( x, y+2);      cprintf( "└");
    gotoxy( x+3, y);      cprintf( "┐");
    gotoxy( x+3, y+1);    cprintf( "│");
    gotoxy( x+3, y+2);    cprintf( "┘");
}
```

```
position_label( char *label_net[20], int node)
{
    int    t, y;

    y = 0;
    gotoxy( 3, 7);        cprintf("%s", label_net[y]);

    for( t=10; t<(node*20)-20; t=t+20)
       {
       y++;
       gotoxy( t+1, 9);    cprintf("%s", label_net[y]);
       }
}

clear_square()
{
    int    t;

    for( t=7; t<=18; t++)
       {
       gotoxy( 3, t);
       cprintf( "                                      ");
       gotoxy( 41, t);
       cprintf( "                                      ");
       }
}
```