โปรแกรมการตัดคำภาษาไทยโดยหลักสถิติ

Thai Word Segmentation Program

Using Statistical Model

โดย

นายศิวโรจน์ ภูวิภิรมย์

รหัส 42067184

H001821

อาจารย์ที่ปรึกษา

ผศ. ดร. อาริต ธรรมโน

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2544

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	โปรแกรมการตัดคำภาษาไทยโดยใช้หลักสถิติ
นักศึกษา	นายศิวโรจน์ ภูวิภิรมย์
อาจารย์ที่ปรึกษา	ผศ. ดร. อาริต ธรรมโน
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

งานวิจัยนี้มุ่งเพื่อสำรวจและพัฒนาโปรแกรมการตัดคำภาษาไทยโดยใช้แบบจำลองทางสถิติเข้าช่วย วิธีนี้อาศัยค่าสถิติ n-gram จากคลังข้อมูลทางภาษาเพื่อช่วยในการตัดสินใจในกรณีที่การตัดคำด้วยวิธีใช้กฎทางไวยากรณ์อักขระและพจนานุกรมให้ผลลัพธ์มากกว่า 1 แบบ วิธีการตัดคำโดยอาศัยค่าสถิติทางคลังข้อมูลภาษานี้คาดว่าจะตัดคำได้ถูกต้องและแม่นยำมากขึ้นกว่าเดิม

Title	Thai Word Segmentation Program Using Statistical Model
Student	Mr. Sivaroj Phuvipirom
Advisor	Assi. Prof. Dr. Arit Thammano
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2001

Abstract

This research aims at surveying and implementing Thai word segmentation algorithms based on statistical model. The n-gram statistics of Thai word from language corpora are used to support the decision when the conventional rule-based and dictionary-based approaches give more than one possible segmentation. This corpus-based statistic approach is expected to be a better way to segment Thai words accurately and precisely.

กิตติกรรมประกาศ

โครงงานพัฒนาระบบงานนี้สำเร็จลงได้ด้วยดีเพราะความช่วยเหลือจาก ผศ. ดร. อาริต ธรรมโน ที่ได้กรุณาสละเวลา ให้คำแนะนำ ชี้แนะ และให้แนวทางที่เป็นประโยชน์ในการศึกษาและแก้ปัญหาต่างๆ ข้าพเจ้าจึงขอขอบพระคุณเป็นอย่างยิ่ง

ขอบพระคุณ บิดา มารดา และครอบครัว ที่ให้กำลังใจแก่ข้าพเจ้าในการศึกษาต่อจนสำเร็จ ถึงแม้ว่าช่วงเวลาที่ข้าพเจ้าต้องศึกษานี้จะมีเวลาให้ท่านเหล่านี้น้อยลงไปก็ตาม

ขอบคุณ คณะเทคโนโลยีสารสนเทศ ที่ได้ให้โอกาสข้าพเจ้าได้เข้าทำการศึกษาต่อในระดับวิทยาศาสตรมหาบัณฑิต นี้

ขอบคุณ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ที่ได้เผยแพร่ข้อมูลของ Orchid Corpus เพื่อให้เป็นของสาธารณะ สามารถนำไปใช้งานที่เป็นประโยชน์ต่อภาษาไทยได้

ศิวโรจน์ ภูวิภิรมย์

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ความเป็นมาของระบบการตัดคำภาษาไทย

ปัจจุบันมีการนำระบบคอมพิวเตอร์มาช่วยประมวลผลต่างๆมากมาย ในการประมวลผลด้วยคอมพิวเตอร์สำหรับประเทศไทยนั้นจะยังคงมีปัญหาและอุปสรรค์ในด้านภาษาศาสตร์อยู่ เช่นเดียวกับ ประเทศในแถบเอเชีย เช่น ญี่ปุ่น เกาหลี จีน เนื่องจากลักษณะธรรมชาติของภาษาในกลุ่มนี้ไม่มีตัวแบ่งแยกการแบ่งคำเหมือนภาษาอังกฤษที่ใช้อักขระช่องว่าง (space) ในการแบ่งแยกคำที่ชัดเจน ทำให้การพัฒนาระบบด้วยคอมพิวเตอร์ในงานที่ต้องเกี่ยวข้องกับภาษาไทย เช่น งานแปลภาษาด้วยคอมพิวเตอร์ (machine translation) การสังเคราะห์เสียงจากประโยค (text-to-speech synthesis) โปรแกรมประยุกต์สำหรับประมวลผลเอกสาร (word processing) หรือโปรแกรมการสืบค้นข้อมูล (information retrieval system) ระบบต่างๆเหล่านี้จำเป็นจะต้องขจัดปัญหาเกี่ยวกับการแบ่งแยกคำให้หมดไปก่อน ซึ่งการแบ่งแยกคำให้เหมาะกับระบบงานที่จะทำ ยังต้องขึ้นอยู่กับลักษณะของงานที่จะนำระบบคอมพิวเตอร์ไปประยุกต์อีกเช่น

- *งานแปลเอกสารด้วยคอมพิวเตอร์* ต้องการตัดคำเพื่อแยกคำในประโยคให้ได้ใจความสมบูรณ์ไม่ผิดเพี้ยนไปจากเดิม
- *การสังเคราะห์เสียงจากประโยค* โปรแกรมต้องการการแบ่งคำที่ให้ได้เพียงแค่พยางค์สำหรับการออกเสียงได้ถูกต้องเท่านั้นก็เพียงพอ ไม่จำเป็นต้องตรงความหมาย
- *โปรแกรมประมวลผลเอกสาร* ต้องการตัดคำในประโยคให้ออกเป็นคำหรือพยางค์ที่ประกอบขึ้นด้วยอักขระที่ถูกกฎทางไวยากรณ์ของภาษา หรือเป็นคำที่มีอยู่ในพจนานุกรม เพื่อให้เอกสารสามารถจัดรูปแบบได้สวยงาม และมีคำสะกดที่ถูกต้อง

1.2 วัตถุประสงค์ของระบบงาน

ในการพัฒนาระบบงานนี้ได้ติดตามและค้นหาวิธีการตัดคำในประโยคโดยมีจุดประสงค์ที่สามารถตัดคำภาษาไทยได้อย่างถูกต้อง โดยพิจารณาถึงข้อดีของวิธีการตัดคำแบบต่างๆ เพื่อนำมาเป็นแนวทางในการพัฒนาระบบงานดังนี้คือ

1.2.1 ใช้วิธีการตัดคำโดยกฎเกณฑ์ทางภาษา และ วิธีการตัดคำโดยพจนานุกรม เพื่อแยกประโยคที่เป็นไปได้ทั้งหมด และ นำหลักสถิติมาประยุกต์ เพื่อช่วยในการตัดสินใจเลือกประโยคที่มีความน่าจะเป็นสูงสุด ในกรณีที่การตัดคำสามารถทำให้เกิดประโยคได้หลายๆ ประโยค

1.2.2 ใช้หลักสถิติที่เรียกว่า n-gram model โดยเน้นเฉพาะ bigram และ trigram ในการคัดเลือกการแบ่งคำในประโยคที่มีความน่าจะเป็นสูงสุด

1.2.3 โปรแกรมช่วยสำหรับเพิ่มเติมข้อมูลในฐานข้อมูล bigram และ trigram โปรแกรมนี้จะสามารถช่วยให้การบริหารคลังข้อมูลทางภาษาเป็นไปอย่างมีประสิทธิภาพและเป็นประโยชน์ต่อนักภาษาศาสตร์

1.3 ขอบเขตของการพัฒนาระบบงาน

1.3.1 ใช้คลังข้อมูลทางภาษาของ Orchid Corpus ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ซึ่งเมื่อนำมาทำเป็นคลังข้อมูลทางภาษาของระบบงานจะมีขนาดของคำในระดับ 17,000 คำ (คำที่ไม่ซ้ำกัน) จะได้ 82,869 ค่าสำหรับข้อมูลสถิติ bigram และ 176,561 ค่าสำหรับข้อมูลสถิติ trigram

1.3.2 โดยพื้นฐานของ Orchid Corpus นี้ส่วนใหญ่ข้อความจะนำมาจากเอกสารที่เกี่ยวข้องกับเทคโนโลยี, วิทยาศาสตร์ และ คอมพิวเตอร์ เป็นส่วนใหญ่ ดังนั้นโปรแกรมการตัดคำจะให้ค่าความน่าจะเป็นของประโยคที่มีข้อความเกี่ยวข้องกับรูปประโยคดังกล่าวข้างต้น

1.3.3 ประโยคที่ตัดคำนี้จะรับข้อมูลที่เป็นภาษาไทยเท่านั้น

1.3.4 ประโยคที่ประกอบด้วยคำที่ปรากฏในพจนานุกรมสามารถตัดคำได้อย่างถูกต้องรวดเร็ว

1.3.5 ประโยคที่ประกอบด้วยคำที่ไม่ปรากฏในพจนานุกรมประเภทต่างๆ เช่น ชื่อบุคคล สถานที่ คำทับศัพท์ ยังไม่สามารถตัดคำได้อย่างถูกต้องรวดเร็ว

1.3.6 ในการเพิ่มข้อมูลสำหรับค่าสถิติ bigram และ trigram นั้นจำเป็นต้องมีข้อมูลรูปประโยคที่ถูกต้องตามหลักไวยากรณ์ของภาษาศาสตร์ จึงจะทำให้การตัดคำเกิดความถูกต้อง

บทที่ 2

การตัดคำโดยหลักไวยากรณ์

2.1 งานวิจัยที่เกี่ยวข้อง

2.1.1 งานวิจัยของ ยุพิน ไทยรัตนานนท์ เป็นงานวิจัยการตัดพยางค์โดยการใช้กฎที่สร้างขึ้นจากกฎไวยากรณ์ภาษาไทย มีการจัดเก็บพยางค์ต่างๆ ที่เป็นข้อยกเว้นไว้ในแฟ้มข้อมูลเนื่องจากมีบางพยางค์ที่ไม่เป็นไปตามกฎที่สร้างไว้ งานวิจัยนี้ได้จัดแบ่งหมวดหมู่ตัวอักษรตามการนำไปใช้ เป็น 5 หมวดหมู่

- กลุ่มพยัญชนะ (Consonant)
- กลุ่มสระ (Vowel)
- กลุ่มวรรณยุกต์ (Tone mark)
- กลุ่มตัวเลข (Numeral)
- กลุ่มตัวอักษรพิเศษ (Special character)

ในแต่ละหมวดหมู่จะมีกฎที่สร้างขึ้นเพื่อให้โปรแกรมสามารถตัดแบ่งพยางค์ได้

2.1.2 งานวิจัยของ สุรินทร์ จรรยาพรพงศ์ เป็นงานวิจัยที่ได้ทำการสร้างกฎสำหรับพยางค์ไทยซึ่งแบ่งเป็นกฎเป็น 2 ชนิดคือ

- กฎการหาขอบเขตหน้า (Front Boundary recognition rule)
- กฎการหาขอบเขตหลัง (Tail Boundary recognition rule)

ในแต่ละกฎดังกล่าวข้างต้นยังสร้างกฎย่อยอีก 2 กฎคือ กฎการแบ่งโดยคุณสมบัติของตัวอักษร และ กฎการแบ่งโดยใช้สระ

ตัวอย่างกฎที่ได้จากคุณสมบัติของอักษรในการหาขอบเขตหน้าของพยางค์

- สระและวรรณยุกต์ต่อไปนี้ ◌ะ ◌า ◌ิ ◌ี ◌ึ ◌ื ◌ุ ◌ู ◌ั ◌ำ ◌ั ◌ ◌ ◌ ◌ ◌ ◌ จะต้องมีพยัญชนะนำหน้าหนึ่งตัวเสมอ
- เ- แ- ไ- โ- ส่วนใหญ่จะเป็นตัวอักษรแรกในพยางค์เสมอยกเว้นคำบางคำ เช่น ขโมย
- ใ- จะเป็นตัวอักษรแรกในพยางค์เสมอไม่มีข้อยกเว้น

ตัวอย่างกฎที่ได้จากคุณสมบัติของอักษรในการหาขอบเขตหลังของพยางค์

- ศ ณ ญ ษ ฐ ฎ ฏ ฒ พ ฑ จะเป็นตัวสะกดเสมอ
- ◌ั จะต้องตามด้วยตัวสะกดอย่างน้อยหนึ่งตัวเสมอ

2.2 การวิเคราะห์ตัวอักษรในภาษาไทย

ในภาษาไทยมีการจัดแบ่งตัวอักษรได้ดังนี้

พยัญชนะ	มี 44 ตัว คือ ก ข ... ฮ
สระ	มี 17 ตัว คือ ◌ะ ◌ั ◌า ◌ำ ◌ิ ◌ี ◌ึ ◌ื ◌ุ ◌ู เ- แ- โ- ไ- ใ- ฤ ฦ
วรรณยุกต์	มี 4 ตัว คือ ◌่ ◌้ ◌๊ ◌๋
ตัวเลข	มี 10 ตัว คือ ๐๑๒๓๔๕๖๗๘๙
อื่นๆ	ได้แก่ ◌็ ◌์ ฯ ๆ ( ) ! ? : ; , .

ตัวอักษรดังกล่าวสามารถแบ่งเป็นกลุ่มเมื่อพิจารณาจากการประสมรวมกันกับตัวอื่นๆ ได้ดังนี้

กลุ่มที่ 1 Non-spacing character คือกลุ่มอักขระเมื่อนำไปรวมกับพยัญชนะแล้วไม่ทำให้เกิดการเคลื่อนไปทางขวาของตำแหน่งที่จะเขียนต่อไป อักขระในกลุ่มนี้ได้แก่

◌ั ◌ิ ◌ี ◌ึ ◌ื ◌ุ ◌ู ◌็ ◌่ ◌้ ◌๊ ◌๋ ◌์ ◌ ◌ ◌

กลุ่มที่ 2 กลุ่มอักขระที่จำเป็นต้องมีรูปพยัญชนะตามเสมอคือ เ- แ- โ- ไ- ใ-

กลุ่มที่ 3 กลุ่มอักขระที่จำเป็นต้องมีรูปพยัญชนะนำหน้าเสมอคือ ◌ะ ◌า ◌ำ ◌ๆ ซึ่ง อักขระ ๆ นี้จะมีแต่ตัว ฤ ฦ นำหน้าเท่านั้น

กลุ่มที่ 4 กลุ่มอักขระที่ประสมกับการันต์ เช่น ทิพย์ ในที่นี้ ย์ ไม่อ่านออกเสียงและเป็นคำลงท้ายเสมอ

กลุ่มที่ 5 อักขระภาษาไทยที่เหลือทั้งหมด

2.3 โครงสร้างของพยางค์ในภาษาไทย

พยางค์ในภาษไทยหมายถึง หน่วยของคำที่เกิดจากอักขระมารวมกันแล้วได้ความหมาย คำไทยส่วนใหญ่จะเป็นพยางค์เดี่ยว (Monosyllable) แต่จะมีคำบางคำเป็นหลายพยางค์

2.4 ข้อดีและข้อเสียของการตัดคำโดยวิธีไวยากรณ์

ข้อดีของการตัดคำโดยใช้กฎเกณฑ์ทางไวยากรณ์นี้ จะประมวลผลได้รวดเร็ว เนื่องจากเป็นการตรวจสอบจากกฎที่ตั้งเอาไว้ในโปรแกรมแบบตายตัว การทำการเปรียบเทียบเป็นจำนวนน้อยครั้ง

ข้อเสียคือ ความแม่นยำในการแบ่งคำมีน้อย เนื่องจากจะพบคำที่แบ่งผิดแต่ว่าถูกต้องตามกฎที่ตั้งไว้ เช่น “งาง” หรือคำที่นำมาจากภาษาต่างประเทศ เช่น “*บอล*” การเปลี่ยนแปลงหรือเพิ่มกฎเกณฑ์จะทำได้จากเนื่องจากกฎต่างๆ ได้ถูกฝังอยู่ในตัวโปรแกรมแล้ว

บทที่ 3

การตัดคำโดยวิธีพจนานุกรม

เนื่องจากวิธีการตัดคำโดยอาศัยการใช้กฎ ยังไม่สามารถให้ความถูกต้องเพียงพอที่จะยอมรับได้ อีกทั้งยังมีปัญหาของการเพิ่มกฎเกณฑ์ต่างๆ ได้ จึงมีผู้ทำการวิจัยโดยอาศัยพจนานุกรมเป็นส่วนช่วยให้โปรแกรมสำหรับตัดคำมีความถูกต้องมากยิ่งขึ้น

3.1 การตัดคำแบบเลือกคำที่ยาวที่สุด (Longest Matching)

เป็นวิธีการที่อ่านอักขระจากประโยคข้อความ จากซ้ายไปขวานำไปเปรียบเทียบกับคำที่มีอยู่พจนานุกรม เลือกคำที่มีความยาวที่สุดปรากฏอยู่ในพจนานุกรม พร้อมทั้งทำเครื่องหมายย้อนกลับดำเนินการตัดคำจากประโยคที่เหลือต่อไป เมื่อคำที่ตัดได้ต่อไปนั้น ไม่ปรากฏอยู่ในพจนานุกรมจะย้อนกลับไปที่จุดย้อยกลับหลังสุดแล้ว เลือกคำที่มีความยาวน้อยลงเป็นลำดับต่อไป ทำต่อไปจนสิ้นสุดกระบวนการ

ตัวอย่างการตัดคำจากประโยค . "*เขาหากระดาษ*" แสดงในตาราง 3.1

ประโยค	*คำที่ได้*	*คำที่ถูกเลือก*
เขาหากระดาษ	เขา	เขา
หากระดาษ	หา,หาก	หาก
ระดาษ	-	(ย้อนรอย)
หากระดาษ	หา,หาก	หา (คำที่มีความยาวรองลงมา)
กระดาษ	กระดาษ	กระดาษ

ตารางที่ 3.1 ตัวอย่างการตัดคำแบบเลือกคำที่ยาวที่สุด (Longest Matching)

วิธีนี้ยังมีข้อผิดพลาด ดังตัวอย่างประโยค "*ไปหามเหสี*" จะตัดคำได้เป็น *ไป / หาม / เห / สี* เพราะคำว่า *หาม* , *หา* และ *เห* เป็นคำที่ปรากฏอยู่ในพจนานุกรม จึงเลือกคำ *หาม* เพราะเป็นคำที่มีความยาวที่สุดที่ปรากฏอยู่ในพจนานุกรม ดังนั้นวิธีการ Longest matching นี้จึงไม่สามารถเลือกตัดคำจากประโยคเช่นนี้ได้ถูกต้อง

3.2 การตัดคำแบบให้จำนวนน้อยที่สุด (Maximal Matching)

เป็นวิธีการที่ใช้แก้ปัญหาของ Longest matching โดยจุดบกพร่องคือ ขั้นตอนวิธีการตัดคำแบเลือกคำยาวที่สุดจะเลือกคำที่ยาวเกินไปตั้งแต่ครั้งแรก

หลักการของการตัดคำให้จำนวนน้อยที่สุดคือ ขั้นตอนแรกจะตัดคำที่เป็นไปได้ทุกๆ แบบก่อน แล้วหลังจากนั้นจะเลือกประโยคที่มีจำนวนคำน้อยที่สุด ตัวอย่างประโยค "*ไปหามเหสี*" จะได้ทางเลือกที่เป็นไปได้ของประโยคคือ *ไป / หาม / เห / สี*, *ไป / หา / มเหสี* โดยวิธีการนี้จะเลือกประโยคที่ตัดคำน้อยที่สุด คือ ประโยค *ไป / หา / มเหสี* เป็นผลลัพธ์ สำหรับในกรณีที่ได้จำนวนคำเท่ากันจะใช้วิธี Longest Matching เข้ามาช่วยในการเลือกประโยค

3.3 งานวิจัยที่เกี่ยวข้อง

ในงานวิจัยแรกๆ เป็นการตัดพยางค์โดยการใช้กฎเพื่อหาขอบเขตของพยางค์ ต่อมาเป็นงานวิจัยเพื่อหาขอบเขตของคำ โดยการหาขอบเขตของคำไม่สามารถหาได้จากการตัดพยางค์เพียงอย่างเดียวได้ เนื่องจากคำประกอบด้วย 1 พยางค์ หรือหลายพยางค์รวมกัน เช่น *เดิน, เดินทาง* เป็นต้นทำให้มีการคิดค้นหาวิธีการตัดคำโดยใช้พจนานุกรมร่วมกับการใช้กฎในการตัดคำ

3.3.1 งานวิจัยของ ดร.ยืน ภู่วรวรรณ และ วิวรรชน์ อิ่มอารมณ์

เป็นงานวิจัยการแบ่งพยางค์ด้วยพจนานุกรม ซึ่งถือได้ว่าเป็นงานวิจัยงานแรกของการตัดพยางค์ที่มีการนำพจนานุกรมเข้ามาใช้โดยจะจัดเก็บพยางค์ต่างๆ ไว้ในพจนานุกรม และมีการนำกฎไวยากรณ์ต่างๆ เข้ามาช่วยในกรณีที่ไม่พบพยางค์ในพจนานุกรม

หลักการทำงานคือจะทำการตรวจสอบสายอักขระ (String) จากซ้ายไปขวากับพยางค์ที่เก็บไว้ในพจนานุกรม ในกรณีที่ตรวจสอบแล้วพบพยางค์มากกว่า 1 พยางค์ ก็ให้ทำการเลือกแบ่งพยางค์โดย

เลือกพยางค์ที่ยาวที่สุดแล้วทำเครื่องหมายจุดย้อนกลับกับพยางค์ที่เหลือ ทำงานวนต่อไปเรื่อยๆ จนจบสายอักขระ โดยวิธีการตัดคำใช้วิธีของ Longest Matching

3.3.2 งานวิจัยของสรศักดิ์ ไทยแท้

เรื่อง การตัดคำไทยโดยใช้ดิกชันนารีที่มีโครงสร้างข้อมูลแบบแฮชชิ่ง นำเสนอหลักการวิธีการตัดคำโดยใช้หลักการพจนานุกรมที่มีโครงสร้างข้อมูลเป็นแบบทรีและมีวิธีการคำนวณ Address ของข้อมูลพจนานุกรมที่ต้องการค้นหากับคำแบบ Hashing วิธีแบ่งแยกคำจะใช้วิธี Longest Matching คือ หาคำที่ยาวที่สุดที่มีพบในพจนานุกรม

3.3.3 งานวิจัยของ ดร.ดวงแก้ว สวามิภักดิ์

เป็นงานวิจัยการตัดคำภาษาไทยโดยใช้กฎไวยากรณ์ที่สร้างขึ้น และมีการนำพจนานุกรมเข้ามาใช้ประกอบร่วมด้วย เพื่อเป็นการแก้ไขการตัดคำโดยใช้พจนานุกรมเพียงอย่างเดียวซึ่งไม่สามารถตัดคำได้อย่างถูกต้องในกรณีคำนั้นไม่ปรากฏอยู่ในพจนานุกรม

กฎต่างๆ ที่สร้างขึ้นอยู่ในรูปนิพจน์ที่มีกฎเกณฑ์ (Regular Expression) โดยกฎที่สร้างขึ้นประกอบด้วยกฎ 43 กฎดังในตารางที่ 3.2

กฎที่	รูปแบบกฎ	ตัวอย่าง/หมายเหตุ
1	[c][t]?[ะ- ำ]	ตัวอย่าง กะ กา กำ ก้า ก่า
2	[c] [ิ ี ึ ื ุ ู] [t]?	ตัวอย่าง ขี่ คื ดุ
3	[c]็[t]?[s]	ตัวอย่าง คือ หือ
4	[c][t]?[ุู]	ตัวอย่าง สู หู
5	[c]ั[t]?[s]	ตัวอย่าง กับ ขับ คั่น
6	[แโไ][c][t]?	ตัวอย่าง เธ โค่
7	[เ- แ-] [c] ็ [s]	เก็บ แข็ง

กฎที่	รูปแบบกฎ	ตัวอย่าง/หมายเหตุ
8	[เ- แ- โ-] [c] [t] ? ะ	และ โก๊ะ
9	[เ- แ- โ-] [กขคตทบปพฟจซศส] ร [t] ?ะ	แครระ แประ
10	[เ- แ- โ-] [กขคบปผพ] ล [t] ? ะ	เขละ แพละ แกละq
11	เ [กขคตทบปพฟจซศส] ร [-า◌ิ◌ี◌ึ◌ื◌ุ◌ู] [t]?	เครา เสรี
12	เ [กขคบปพฟ] ล [-า◌ิ◌ี◌ึ◌ื◌ุ◌ู] [t] ?	เกลา เขลี่
13	[เ- แ- โ-] [กขค] ว [t] ? ะ	ตัวอย่าง แควะ แขวะ
14	เ [c]◌ี [t] ? ย	ตัวอย่าง เสีย เสี่ย
15	เ [c] [t] ? า ะ	ตัวอย่าง เกาะ เสาะ
16	เ [c] [t] ? [-า-ะ]	ตัวอย่าง เสา
17	เ [c] [t] ?	ตัวอย่าง เก๋
18	[โ- ไ- ใ-] ห [งญนมยรลว] [t] ?	ตัวอย่าง โหง่
19	เ [c]◌ื [t] ? อ	ตัวอย่าง เสือ เสื่อ
20	[c]◌ื [t] ? อ	ตัวอย่าง พือ คือ
21	[c] ◌์	ตัวอย่าง ต์
22	[◌็◌์◌๋◌๊◌๋◌ะ ◌ั -า ◌ำ◌ิ◌ี◌ึ◌ื◌ุ◌ู]	
23	ฯ	
24	เ [จส] ริ ญ	ตัวอย่าง เจริญ เสริญ
25	หร [t] ? [-ะ -า ◌ำ ◌ุ ◌ู]	ตัวอย่าง หรำ
26	หร [◌ิ◌ี◌ึ◌ื] [t] ?	ตัวอย่าง หรี หรึ่
27	เ [กขคบปผพฟ] ลื [t] ? อ	ตัวอย่าง เกลือ
28	เ [กขคตทบปพฟ] รื [t] ? อ	ตัวอย่าง เครื่อ
29	เ [กขคบปผพฟ] ลิ [t] ? [s]	ตัวอย่าง เพลิง
30	เ [กขคตทบปพฟ] ริ [t] ? [s]	ตัวอย่าง เกริ่น
31	แ [กขคตทบปพฟ] ร [t] ? [กงวนบมทตด]	ตัวอย่าง แกรม
32	แ [กขคบปผพฟ] ล [t] ? [กงวนบมทตด]	ตัวอย่าง แกลบ แปลก

กฎที่	รูปแบบกฎ	ตัวอย่าง/หมายเหตุ
33	เห[งญนมยรลว][t]?า	ตัวอย่าง เหมา เหงา
34	[A-Za-z เครื่องหมายพิเศษ]*	อักษรภาษาอังกฤษ A-Z a-z เครื่องหมายพิเศษต่างๆเรียงติดต่อกันเป็นจำนวนกี่ตัวก็ได้
35	ฯลฯ	
36	ๆ	
37	เ[กขค] วี [t] ? ย	ตัวอย่าง เกวีย เขวีย
38	เ[กขคฅฆบปพฟ] รี [t] ? ย	ตัวอย่าง เกวีย เทรีย
39	เ[กขคงบผพฟ] ลี [t] ? ย	ตัวอย่าง เกลีย
40	" "	ช่องว่าง
41	[0-9]*	เลข 0123456789 เรียงติดกันเป็นจำนวนกี่ตัวก็ได้
42	"\n"	อักขระขึ้นบรรทัดใหม่
43	.	อักษรทุกตัวที่ไม่อยู่ในกฎข้างต้น

ตารางที่ 3.2 กฎการตัดพยางค์

เมื่อได้กลุ่มของพยางค์ (Token) จากการตัดคำด้วยกฎแล้วต่อไปจะเป็นการรวมกลุ่มของพยางค์โดยทำการตรวจสอบจากพจนานุกรมอีกครั้ง

3.4 ข้อดีและข้อเสียของวิธีการตัดคำโดยวิธีพจนานุกรม

ข้อดีของการตัดคำโดยใช้พจนานุกรม คือ มีความแม่นยำสูงกว่า จึงแน่ใจได้ว่าคำที่ตรวจสอบกับพจนานุกรมจะเป็นคำที่มีความหมาย

ข้อเสียคือ ยังไม่สามารถแก้ปัญหาสำหรับคำกำกวม เช่น “ *เรือ โคลงเพราะ โคลงเรือ* ” จะตัดคำได้เป็น “ *เรือ / โคลง / เพราะ / โคลง / เรือ* ” ซึ่งถูกต้องตามกฎไวยากรณ์แต่ไม่ถูกต้องตรงความหมาย

บทที่ 4

การตัดคำโดยวิธีหลักสถิติ

การตัดคำภาษาไทยในยุคก่อนๆอาศัยความรู้ที่ตายตัวแบบ rule-based หรือ heuristic approach กฎทางไวยากรณ์และคำในพจนานุกรมมีข้อจำกัดหลายอย่าง อาทิ ไม่สามารถจัดการกับความกำกวมทางความหมาย หรือ ไม่ยืดหยุ่นพอที่จะจัดการกับคำที่ไม่พบในพจนานุกรมได้

การตัดคำภาษาไทยโดยวิธีการทางสถิติเพื่อแก้ปัญหาดังกล่าวข้างต้น จำเป็นต้องอาศัยคลังข้อมูลทางภาษาเพื่อนำค่าของข้อมูลดังกล่าวมาคำนวณค่าทางสถิติตามแต่ละโมเดลของการวิเคราะห์คำทางสถิติต่างๆ โดยพื้นฐานแล้ว จะอาศัยคลังข้อมูลจำนวนที่มากพอของข้อมูลตัวอย่าง อีกทั้งยังมีไว้สำหรับตรวจสอบความถูกต้องของผลของการวิเคราะห์ คำตามโมเดลนั้นๆอีกด้วย

4.1 คลังข้อมูลภาษา (Corpus)

คลังข้อมูลทางภาษาเป็นสิ่งสำคัญอย่างมากในการพัฒนาระบบประมวลผลภาษาธรรมชาติ คลังข้อมูลทางภาษาคือ ภาษาเขียนหรือภาษาพูดที่เป็นภาษาจริงที่ใช้ในชีวิตประจำวันซึ่งรวบรวมมาจากหนังสือ บทความวารสาร บทความทางวิทยุ เอกสารต่างๆ จำนวนมาก แล้วนำมาเก็บไว้ในคอมพิวเตอร์ โดยทั่วไปคลังข้อมูลทางภาษาจะใช้สำหรับศึกษาปรากฏการณ์ทางภาษา รวมทั้งใช้หาค่าสถิติที่น่าสนใจเพื่อหาหรือสร้างเป็น language model คลังข้อมูลทางภาษาอาจมีการเพิ่มเติมข้อมูลทางภาษาลงไปด้วย เช่น การแบ่งคำที่ถูกต้อง การกำกับชนิดของคำ เป็นต้น

ตัวอย่างของคลังข้อมูลสำหรับภาษาไทยคือ Orchid Curpus ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ ซึ่งมีข้อมูลของคำในภาษาไทย และหน้าที่ของคำกำกับอยู่ด้วย

4.2 แบบจำลองทางสถิติ n-gram

n-gram คือ ลำดับย่อยของสัญลักษณ์ n ตัวใดๆ ที่อยู่ติดกันในเอกสารหรือสายอักขระหนึ่งๆ ทั้งหมดที่เป็นไปได้ คำว่าสัญลักษณ์ในที่นี้อาจจะเป็นอักขระ (character) คำ (word) หรือ เป็นหน้าที่ของคำ ของคำ (Part Of Speach) ก็ได้ ขึ้นอยู่กับโมเดลที่ใช้งาน ตัวอย่างเช่น ในสตริง internationalization จะได้ลำดับย่อย trigram ของอักขระ ต่างๆ ที่เป็นไปได้ดังนี้

int, nte, ter, ern, rna, nat, ati, tio, ion, ona, ali, liz, iza, zat, ati, tio, ion

หรือถ้าเป็น bigram ของคำในประโยคที่ว่า I liketo eat Thai food จะได้ลำดับย่อยต่างๆ ดังนี้

I like, like to, to eat, eat Thai, Thai food

เมื่อได้ลำดับย่อยแล้ว จะทำการนับความถี่ของการเกิดลำดับย่อยที่ไม่ซ้ำกัน และคำนวณหาความน่าจะเป็นของการเกิดคำนั้น ดังนั้น trigram ระดับอักขระจากตัวอย่างคำว่า internationalization จะได้ดัง ตารางที่ 4.1

ali (1/17)	ati (2/17)	ern (1/17)	int (1/17)	ion (2/17)
iza (1/17)	liz (1/17)	nat (1/17)	nte (1/17)	ona (1/17)
rna (1/17)	ter (1/17)	tio (2/17)	zat (1/17)	

ตารางที่ 4.1 ตัวอย่างค่าความถี่ของ trigram ในคำว่า internationalization

ค่าความน่าจะเป็นของ n-gram นั้นเป็นค่าทางสถิติของภาษา แนวทางการศึกษาภาษาศาสตร์ยุคใหม่จะใช้คลังข้อมูลทางภาษา ซึ่งใช้อยู่จริงและมีปริมาณมาก เพื่อนำมาหาค่าทางสถิติแบบต่างๆ ของภาษา ซึ่งเรียกว่า Langauge Model นอกจาก n-gram แล้ว ยังมีค่าทางสถิติอื่นๆ เช่น entropy, perplexity, mutual information, ฯลฯ

การประยุกต์ใช้งานของ n-gram มีหลายอย่าง อาทิ การวัดความใกล้เคียงหรือเหมือนกันของเอกสาร (similarity measure) การทำดัชนีอัตโนมัติ (automatic indexing) การทายข้อความหรือคำที่ขาดหายไป การหาคำที่ไม่รู้จัก การตรวจสอบหาคำสะกดผิด (spelling checker) การตัดคำ (word segmentation) และกำกับชนิดของคำทางไวยากรณ์ (part of speech tagging)

4.3 ทฤษฎีความน่าจะเป็นพื้นฐาน

ทฤษฎีที่เกี่ยวข้องส่วนมากแล้วจะเป็นทฤษฎีของ Bayes' Theorem ซึ่งมีการนำไปประยุกต์ใช้งานด้านต่างๆ มากมาย

4.3.1 Prior Probability

เป็นการคำนวณความน่าจะเป็นโดยไม่สนใจส่วนอื่นๆที่อยู่รอบข้างคำนวณได้จาก

$$P(w_i) = \frac{C(w_i)}{\sum_i C(w_i)} = \frac{C(w_i)}{N} \quad \text{.....(4.1)}$$

เมื่อ C(w) คือฟังชันการนับจำนวนตัวอักษร

4.3.2 Joint Probability

เป็นความน่าจะเป็นของเหตุการณ์ที่คำสองคำเกิดร่วมกัน คำนวณได้จาก

$$P(w_{i-1}w_i) = \frac{C(w_{i-1}w_i)}{N} \quad \text{.....(4.2)}$$

4.3.3 Condition Probability

เป็นการหาความน่าจะเป็นของคำใดๆ เมื่อมีคำก่อนหน้าปรากฏอยู่ คำนวณได้จาก

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} \quad \text{.....(4.3)}$$

4.3.4 Bayes' Theorem

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \quad \text{.....(4.4)}$$

4.4 ความน่าจะเป็นของการเกิดประโยค

กำหนดประโยค $W = w_1 w_2 \ldots w_n$ โดยที่ w_i เป็น อักษร หรือ กลุ่มของตัวอักษร (token) ก็ได้

ความน่าจะเป็นของการเกิดประโยค W คำนวณได้จาก

$$P(W) = P(w_1)P(w_2|w_1) * P(w_3|w_2 w_1) * \ldots * P(w_n|w_{n-1} w_{n-2} \ldots w_1) \quad \text{.....(4.5)}$$

หรือเขียนได้อีกรูปแบบหนึ่งคือ

$$P(W) = \prod_{t=1}^{N} P(w_t | w_1^{t-1}) \quad \text{.....(4.6)}$$

จากสมการที่ (4.6) คำนวณได้ยาก และต้องการคลังข้อมูลขนาดใหญ่มาก จึงมีวิธีประมาณค่าในการคำนวณแบบอื่นๆ แทน

4.5 การประมาณค่าโดยวิธี Bigram

Bigram เป็นแบบจำลอง n-gram ที่ใช้ค่า n = 2 นั่นคือ เมื่อพิจารณาคำใดๆ จะสนใจคำที่อยู่ก่อนหน้าจำนวน 1 คำ และคำนวณหาความน่าจะเป็นแบบ Condition Probability ดังนั้นจากสมการที่ (4.6) จึงประมาณค่าใหม่ได้เป็น

$$P(W) = P(w_1)P(w_1|<space>) * P(w_2|w_1) * \ldots * P(w_n|w_{n-1}) \quad \text{.....(4.7)}$$

หรือ

$$P(W) = \prod_{t=1}^{N} P(w_t | w_{t-1}) \quad \text{.....(4.8)}$$

จากสมการที่ (4.8) เป็นการประมาณค่าความน่าจะเป็นของการเกิดประโยค โดยการคำนวณผลคูณรวมของความน่าจะเป็นแบบ Bigram ของทุกๆ คำที่อยู่ในประโยคนั้นๆ

4.6 การประมาณค่าโดยวิธี Trigram

Trigram เป็นแบบจำลอง n-gram ที่ใช้ค่า n = 3 นั่นคือ เมื่อพิจารณาคำใดๆจะสนใจคำที่อยู่ก่อนหน้าจำนวน 2 คำ และคำนวณหาความน่าจะเป็นแบบ Condition Probability ดังนั้นจากสมการที่ (4.6) จึงประมาณค่าใหม่ได้เป็น

$$P(W) = P(w_1)P(w_1 | < s >, < s >) * P(w_2 | w_1, < s >) * ... * P(w_n | w_{n-1} w_{n-2}) \quad \text{.....(4.9)}$$

หรือ

$$P(W) = \prod_{t=1}^{N} P(w_t | w_{t-1} w_{t-2}) \quad \text{.....(4.10)}$$

จากสมการที่ (4.10) เป็นการประมาณค่าความน่าจะเป็นของการเกิดประโยค โดยการคำนวณผลคูณรวมของความน่าจะเป็นแบบ Trigram ของทุกๆ คำที่อยู่ในประโยคนั้นๆ

บทที่ 5

โปรแกรมการตัดคำโดยวิธีหลักสถิติ

5.1 การทำงานของโปรแกรมตัดคำ

5.1.1 โปรแกรมจะรับสายตัวอักษร (string)

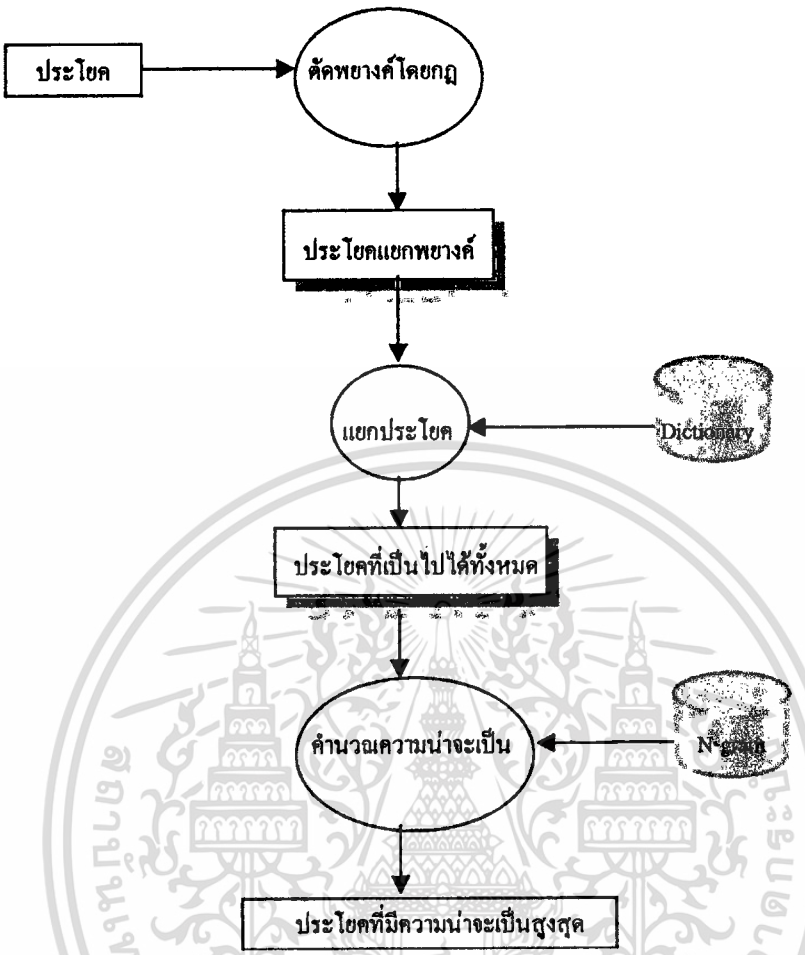
5.1.2 ทำการแยกพยางค์เบื้องต้นโดยกฎไวยากรณ์

5.1.3 หาประโยคที่เป็นไปได้จากการจับกันของพยางค์ในรูปแบบต่างๆ

5.1.4 คัดเลือกประโยคที่มีความน่าจะเป็นของประโยคสูงที่สุด จากโมเดลทางสถิติ bigram หรือ trigram พร้อมทั้งแสดงค่าทางสถิติที่คำนวณได้ในแต่ละประโยค

5.2 การออกแบบโปรแกรม

แนวทางการออกแบบโปรแกรมแบ่งเป็นขั้นตอนดังแสดงในรูปที่ 5.1

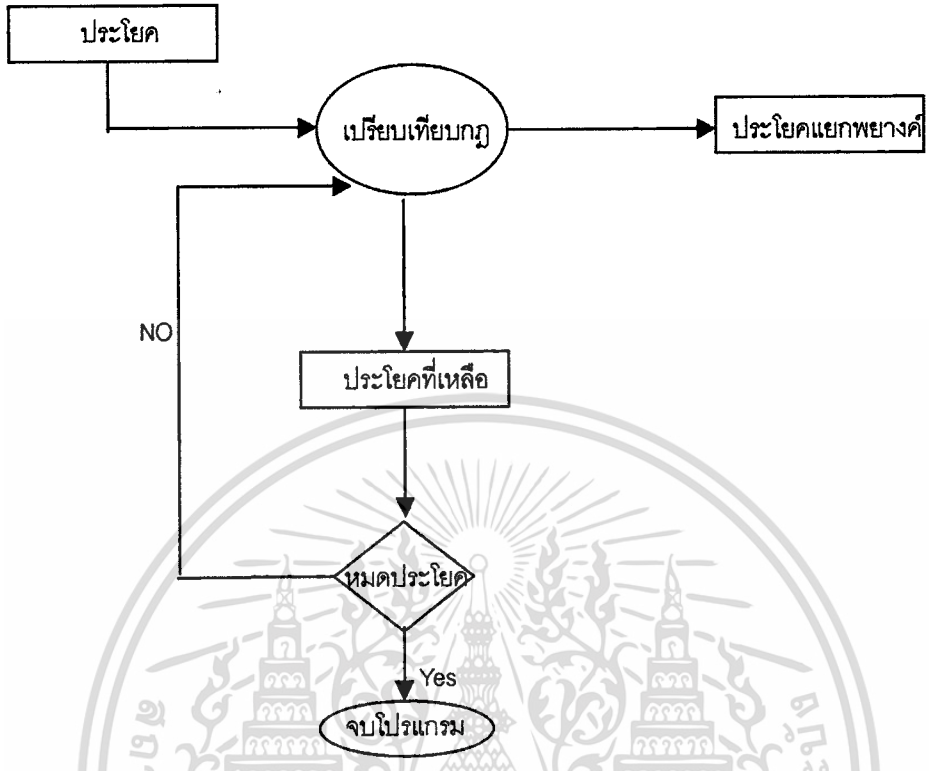


รูปที่ 5.1 แสดงขั้นตอนการตัดคำ

5.2.1 การตัดพยางค์โดยวิธีกฎไวยากรณ์

การใช้กฎไวยากรณ์นี้เพื่อเป็นการตัดพยางค์เบื้องต้นเพื่อให้ได้พยางค์ที่เล็กที่สุด สามารถลดรูปสายตัวอักษรให้มีความยาวน้อยลงในการเก็บข้อมูลแบบอาเรย์ ก่อนนำไปประมวลผลเพื่อหาคำต่อไป ทำให้สามารถลดจำนวนวนรอบของการอ่านสายตัวอักษรได้

กฎไวยากรณ์ที่นำมาประยุกต์ใช้อาศัย กฎไวยากรณ์ตามแนวทางของ ดร.ดวงแก้ว สวามิภักดิ์ แต่ได้มีการตัดกฎที่ทำให้เกิดคำควบกล้ำบางกฎทิ้งไปเพราะอาจทำให้เกิดความสับสนในการประมวลผลคำโดยวิธีพจนานุกรมได้ แสดงการทำงานของโปรแกรมดังรูปที่ 5.2



รูปที่ 5.2 ขั้นตอนการตัดพยางค์โดยกฎไวยากรณ์

แสดงตัวอย่างจากประโยคดังนี้ *"การออกกำลังกาย"* เมื่อผ่านกระบวนการตัดพยางค์โดยกฎไวยากรณ์จะได้ พยางค์ที่เล็กที่สุดคือ "กา", "ร", "อ", "อ", "ก", "กำ", "ลัง", "กา", "ย", กฎสำหรับการตัดพยางค์เบื้องต้นแสดงไว้ใน ภาคผนวก ก

5.2.2 การหาประโยคทั้งหมดโดยใช้พจนานุกรม

เป็นการนำพยางค์ที่ได้จากขั้นตอนแรกเพื่อหาคำที่เป็นไปได้จากการจับกันของพยางค์เหล่านั้น วิธีการคือนำพยางค์มาเรียงกันเรียงกันทีละพยางค์เปรียบเทียบกับพจนานุกรม

ขั้นตอนการหาพยางค์ในประโยคเพื่อตรวจสอบคำในพจนานุกรม

จากประโยคตัวอย่าง *"การออกกำลังกาย"* พยางค์ที่ได้คือ กา / ร / อ / อ / ก / กำ / ลัง / กา / ย

- พยางค์แรกที่ได้คือ คือ กา
- เก็บค่าที่ตัวชี้ปัจจุบันในตัวแปร หาความยาวมากที่สุดที่คำขึ้นต้นด้วยตัวแปรนี้
- วนรอบจนตัวแปรมีความยาวมากที่สุดที่ขึ้นต้นด้วยพยางค์ปัจจุบันหรือสิ้นสุดประโยคก่อน
- นำพยางค์ต่อมาประกอบกับพยางค์หน้าในตัวแปร ได้เป็น "การ" และตรวจสอบกับพจนานุกรม จะได้คำที่เป็นไปได้สองทางคือ "กา" , "การ" บันทึกลงในโครงสร้างข้อมูล
- พยางค์ที่ 2 คือ ร
- หาความยาวมากที่สุดของคำที่ขึ้นต้นด้วย ร ทั้งหมดจากพจนานุกรม (ได้ค่า 9)
- วนรอบจนตัวแปรมีความยาว >= 9 หรือ สิ้นสุดประโยคก่อน
- นำพยางค์ต่อมาประกอบกับพยางค์หน้า ได้เป็น "รอ" และตรวจสอบกับพจนานุกรม

โปรแกรมจะสร้าง Object สำหรับเก็บคำที่ถูกประกอบด้วยพยางค์ดังกล่าวข้างต้น โดยที่มีโครงสร้างแบบ Link List คือมีตัวชี้ (Pointer) ไปยัง Object ต่อไป

ลักษณะของ Object ที่ใช้มี 2 ลักษณะคือ

- WordClass เป็น Object ที่มีโครงสร้างข้อมูลของคำ , ตัวชี้ตำแหน่ง และฟังชันการทำงานสำหรับอ่านค่าและการท่องไปบน Object แบบ Link List มีรูปแบบแสดงดังในตารางที่ 5.1
- PathClass เป็น Object สำหรับเก็บข้อมูลเพื่อบอกว่าคำที่ถูกชี้ต่อไปนี้จะสามารถแยกคำได้มากกว่า 1 คำเป็นต้นไป PathClass เป็น Object ที่มีโครงสร้างข้อมูลแบบ Link List คือมีตัวชี้ที่เก็บค่า Address ของ Node ต่อไป มีโครงสร้างของ Object ดังตารางที่ 5.2

ชื่อตัวแปร/ฟังชัน	*ความหมาย*	*ชนิด*
Word	คำที่ได้จากการรวมกันของพยางค์	String
TokenStart	ตำแหน่งเริ่มต้นของพยางค์	Int

ชื่อตัวแปร/ฟังชั่น	ความหมาย	ชนิด
TokenEnd	ตำแหน่งสิ้นสุดของพยางค์ที่มารวมกัน	Int
NextWordClass	Pointer ชี้ไป Node ต่อไป ในกรณีสิ้นสุดประโยคมีค่าเป็น null	Pointer
Dictionary	ตัวแปรสำหรับบอกว่าคำมีอยู่ในพจนานุกรมหรือไม่	Boolean
WordNode()	Constructor สำหรับ Object เป็นการกำหนดค่าเริ่มต้น	Constructor
SetWord()	ฟังชันการให้กำหนดค่าให้ WORD	Function
ReadWord()	ฟังชันสำหรับการอ่านค่าคำ (Word) ของ Object	Function
SetTokenStart	ฟังชันกำหนดค่าให้ตัวแปร TokenStart	Function
SetTokenEnd	ฟังชันกำหนดค่าให้ตัวแปร TokenEnd	Function
SetNextWordClass()	ฟังชันการกำหนดตัวชี้สำหรับชี้ Node ต่อไป	Function
ScanWords()	ฟังชันการท่องบน Object แบบ Link List โดยสนใจเฉพาะ Node ที่คำปรากฏในพจนานุกรมเท่านั้น (Dictionary = true) เป็นการทำงานแบบเรียกตัวเอง (Recursive)	Function
ScanWords2()	ฟังชันการท่องบน Object แบบ Link List โดยจะท่องไปบนทุกๆ Node ที่ตัวชี้อ้างถึง	Function

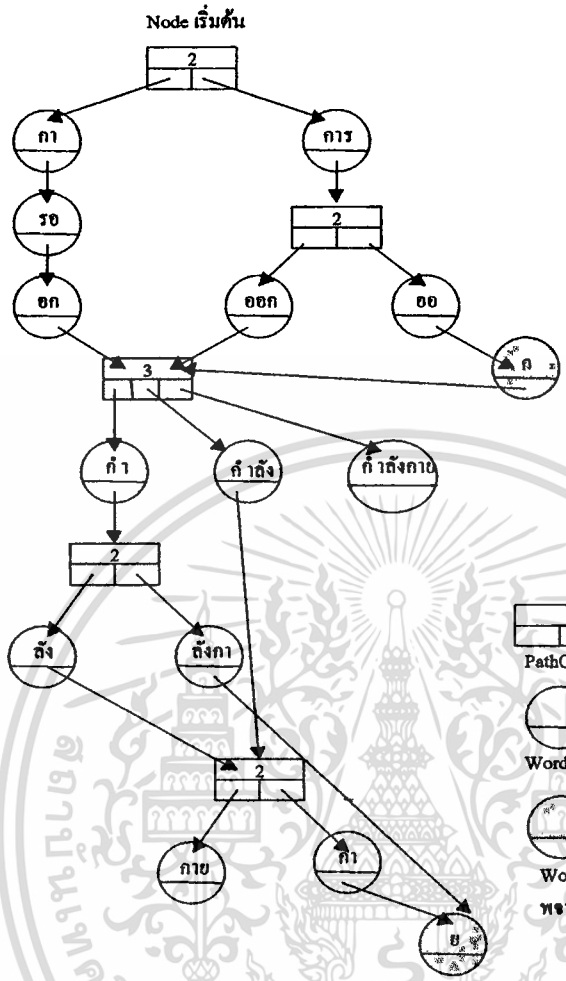
ตารางที่ 5.1 โครงสร้างข้อมูลสำหรับ Object WordClass

ชื่อตัวแปร/ฟังชั่น	ความหมาย	ชนิด
Position	ตำแหน่งของพยางค์ที่ประกอบได้คำมากกว่า 1 คำ	Int
MaxPath	จำนวนคำที่เกิดได้สูงสุดจากตำแหน่งพยางค์ (Position)	Int
NextPoint	เป็น array ที่เก็บ Address ของ Node ต่อไป	Array Pointer

ชื่อตัวแปร/ฟังชัน	ความหมาย	ชนิด
NextWordClass	Pointer ชี้ไป Node ต่อไป ในกรณีสิ้นสุดประโยคมีค่าเป็น null	WordClass/ PathClass
TokenStart	ตำแหน่งของพยางค์แรก	Int
TokenEnd	ตำแหน่งสุดท้ายของพยางค์ที่มารวมกัน	Int
PathNode()	ฟังชัน Constructor สำหรับ Object ในการ initial ค่า	Constructor
SetMaxPath()	ฟังชันการกำหนดค่าให้ MaxPath	Function
SetPosition()	ฟังชันการกำหนดค่าให้ Position	Function
SetNextPoint()	ฟังชันการกำหนดตัวชี้สำหรับชี้ Node ต่อไป	Function
ScanWords()	ฟังชันการท่องบน Object แบบ Link List โดยสนใจเฉพาะ Node ที่บรรจุคำปรากฏในพจนานุกรมเท่านั้น (Dictionary = true) เป็นการทำงานแบบเรียกตัวเอง (Recursive)	Function
ScanWord2()	ฟังชันการท่องบน Object แบบ Link List	Function

ตารางที่ 5.2 แสดงโครงสร้างข้อมูลสำหรับ Object PathClass

ตัวอย่างเมื่อมี พยางค์พยางค์ที่เล็กที่สุดคือ “กา” , “ร” , “อ” , “อ” , “ก” , “กำ” , “ลัง” , “กา” , “ย” เมื่อผ่านกระบวนการวิธีพจนานุกรม จะมีโครงสร้างดังรูปที่ 5.3



รูปที่ 5.3 โครงสร้างของ Object สำหรับการตัดคำโดยพจนานุกรม

จากรูปที่ 5.3 นั้นจะสามารถหาประโยคที่เป็นไปได้ทั้งหมดจากการท่องไปบน Object ต่างๆ จนครบทุกเส้นทางโดยโปรแกรมจะทำงานแบบ **Recusive** จากประโยคตัวอย่างจะได้ประโยคดังนี้

1.	กา / รอ / อก / กำ ลัง / กา / ย
2.	กา / รอ / อก / กำ / ลัง / กาย
3.	กา / รอ / อก / กำ / ลังกา / ย
4.	กา / รอ / อก / กำลัง / กา / ย

5.	กา / รอ / อก / กำลัง / กาย
6.	กา / รอ / อก / กำลังกาย
7.	การ / ออ / ก / กำ / ลัง / กา / ย
8.	การ / ออ / ก / กำ / ลัง / กาย

9.	การ / ออ / ก / กำ / ลังกา / ย
10.	การ / ออ / ก / กำลัง / กา / ย
11.	การ / ออ / ก / กำลัง / กาย
12.	การ / ออ / ก / กำลังกาย
13.	การ / ออก / กำ / ลัง / กา / ย
14.	การ / ออก / กำ / ลัง / กาย

15.	การ / ออก / กำ / ลังกา / ย
16.	การ / ออก / กำลัง / กา / ย
17.	การ / ออก / กำลัง / กาย
18.	การ / ออก / กำลังกาย

ตารางที่ 5.3 การตัดคำที่ได้ทั้งหมดจากประโยค "การออกกำลังกาย"

จากตารางที่ 5.3 แสดงถึงการได้รูปประโยคจากคำที่เก็บไว้ในโครงสร้างของ Object แบบ Link List แต่จะมีบางประโยค ที่อาจประกอบด้วยคำที่ไม่ปรากฏในพจนานุกรม เช่น คำ "ออ" , "ก" , "ย" ดังนั้นเมื่อต้องการตัดคำโดยไม่ให้เกิด คำที่ไม่ปรากฏในพจนานุกรม จะตรวจสอบคำและจะข้ามคำที่ไม่ปรากฏในพจนานุกรมก่อนการคัดเลือก ซึ่งจากประโยคตัวอย่างจะได้ประโยคที่ 2, 5, 6, 14, 17, 18 ทำให้เหลือประโยคที่จะนำไปคำนวณน้อยลงเป็นอย่างมาก

5.2.3 อัลกอริทึมการเลือกประโยคที่มีความน่าจะเป็นสูงสุด

จากประโยคที่เกิดขึ้นได้ทั้งหมดจะนำไปคำนวณหาความน่าจะเป็นโดยอาศัยโมเดลทางสถิติ 2 แบบ คือ

5.2.3.1 Bigram Model

ในแต่ละคำที่ปรากฏอยู่ในประโยคจะคำนวณความน่าจะเป็นได้จาก

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$ เมื่อ C() คือฟังชันการนับจำนวน

และการคำนวณความน่าจะเป็นของทั้งประโยคจะใช้ สมการที่ (4.8) นั่นคือรวมผลคูณของความน่าจะเป็นที่เกิดคำทุกๆ คำในประโยคนั้นมีอัลกอริทึมสำหรับการคำนวณคือ

```
//  คำนวณค่าความน่าจะเป็นสำหรับประโยคใดๆ หนึ่งประโยค
for($i=0;$i= จำนวนคำในประโยค;$i++){
  $w2=SENTENCES[$i];            //$w2 ตำแหน่งคำจาก คำที่ 1 ถึงสิ้นสุดประโยค
  if($i-1<0)
   $w1 = "";
  else
   $w1=SENTENCES[$i-1]);        // $w2 = คำที่สนใจ  $w1 = คำที่เกิดก่อนหน้า 1 ตำแหน่ง
  $keyspair = "$w2|$w1";
  $query  = "select b.count as prior_wordcount , w.count as wordcount
          from bigram as  b,wordlist_bigram as w
          where ( b.previous_word = '$w1' and b.word = '$w2' )
          and   (b.word = w.word)";
  execute $query;
  if ($query does not exixts)
    $ prob_bigram *= 0.000006;          // ในกรณีไม่พบคำในฐานข้อมูลให้ค่าน้อยที่สุด
  else
  $pro_bigram  *= (prior_wordcount/wordcount);      // ผลคูณของความน่าจะเป็น
}
```

เมื่อได้ความน่าจะเป็นของการเกิดประโยคจากทุกๆ ประโยค จะเลือกเอาประโยคที่เป็นประโยคที่ตัดคำดีที่สุดจากค่าความน่าจะเป็นที่คำนวณได้สูงที่สุด

5.2.3.2 Tri-gram Model

ในแต่ละคำที่ปรากฏอยู่ในประโยคจะคำนวณความน่าจะเป็นได้จาก

$$P(w_i \mid w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_i)}$$ เมื่อ C() คือฟังชันการนับจำนวน

และการคำนวณความน่าจะเป็นของทั้งประโยคจะใช้ สมการที่ (4.10) นั่นคือรวมผลคูณของความน่าจะเป็นที่เกิดคำทุกๆ คำในประโยคนั้นมีอัลกอริทึมสำหรับการคำนวณคือ

```
for($i=0;$i= จำนวนคำในประโยค;$i++){
  $w3=SENTENCES[$i];                    //$w2 ตำแหน่งคำจาก คำที่ 1 ถึงสิ้นสุดประโยค
  if($i-1<0)
   $w2 = "";
  else
   $w2=SENTENCES[$i-1];
  if($i-2<0)
   $w1 = "";
  else
   $w1=SENTENCES[$i-2];
  else
  $keyspair = "$w3|$w2,$w1";
  $query = "select b.count as join_wordcount , w.count as wordcount from trigram as
           b,wordlist_trigram as w
           where ( b.word = '$w3' and b.joinword1 = '$w1' and b.joinword2 = '$w2' )
           and   (b.word = w.word)";
  execute $query;
  if ($query does not exixts)
    $ prob_bigram *= 0.000006;          // ในกรณีไม่พบคำในฐานข้อมูลให้ค่าน้อยที่สุด
  else
  $pro_bigram  *= (prior_wordcount/wordcount);      // ผลคูณของความน่าจะเป็น
}
```

เมื่อได้ความน่าจะเป็นของการเกิดประโยคจากทุกๆ ประโยค จะเลือกเอาประโยคที่เป็นประโยคที่ตัดคำดีที่สุดจากค่าความน่าจะเป็นที่คำนวณได้สูงที่สุด

5.2.4 การออกแบบฐานข้อมูล

5.2.4.1 TABLE bigram สำหรับเก็บค่าการเกิดของคำในโมเดล bigram

Fields	*Type*	*Description*
Previous_word	varchar(100) binary	คำที่อยู่ก่อนหน้า
Word	varchar(100) binary	คำที่ต้องการหาค่าความน่าจะเป็น
Count	Int	ความถี่ของ word ที่เกิด

ตารางที่ 5.4 โครงสร้าง TABLE bigram

5.2.4.2 TABLE dic ตารางสำหรับเก็บคำในพจนานุกรม

Fields	*Type*	*Description*
Word	Varchar(100) binary	คำในพจนานุกรม

ตารางที่ 5.5 โครงสร้าง TABLE dic

5.2.4.3 TABLE trigram สำหรับเก็บค่าการเกิดของคำในโมเดล trigram

Fields	*Type*	*Description*
joinword1	varchar(100) binary	คำแรกสุดของของการจับคู่คำ
joinword2	varchar(100) binary	คำที่สองของการจับคู่คำ
Word	varchar(100) binary	คำที่ต้องการหาความน่าจะเป็น
Count	int	ความถี่ของ word ที่เกิด

ตารางที่ 5.6 โครงสร้าง TABLE trigram

5.2.4.4 TABLE wordlist_bigram สำหรับเก็บค่าความถี่ของคำ word ที่เกิดขึ้นทั้งหมด ในโมเดล Bi-gram

Fields	*Type*	*Description*
Word	varchar(100) binary	คำที่ต้องการหาค่าความน่าจะเป็น
Count	Int	ความถี่ของ word ที่เกิด

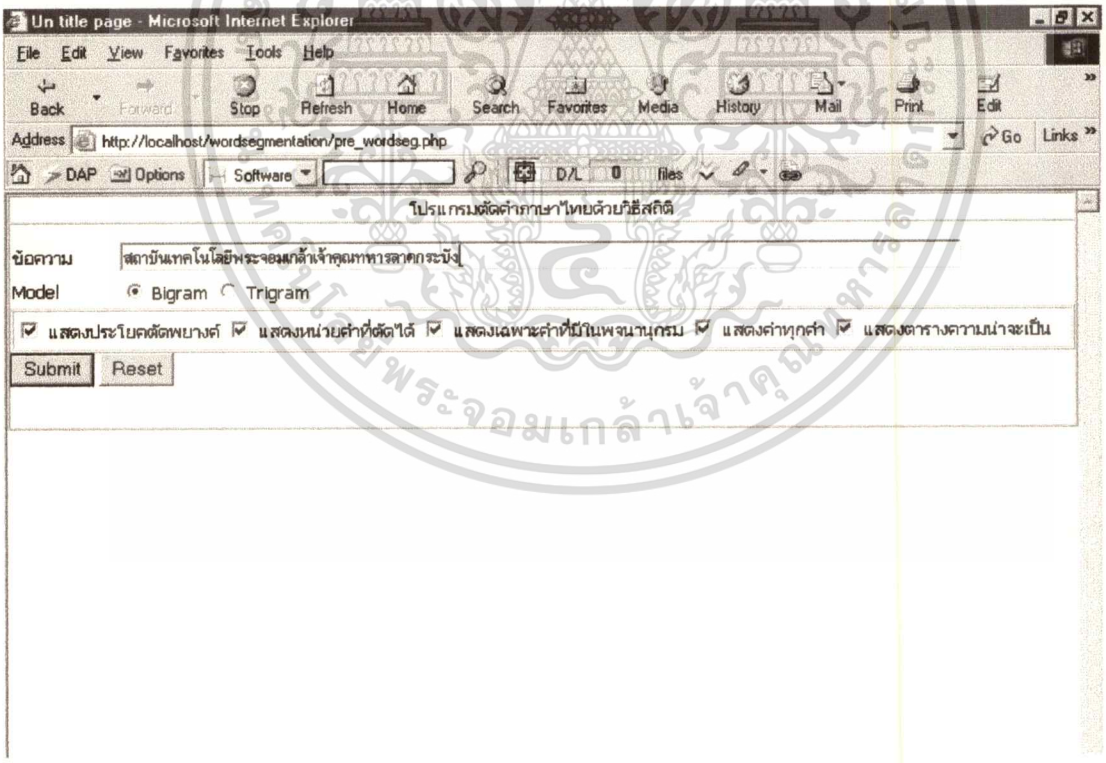
ตารางที่ 5.7 โครงสร้าง TABLE wordlist_bigram

5.2.4.5 TABLE wordlist_trigram สำหรับเก็บค่าความถี่ของคำ word ที่เกิดขึ้นทั้งหมด ในโมเดล Tri-gram

Fields	Type	Description
Word	varchar(100) binary	คำที่ต้องการหาค่าความน่าจะเป็น
Count	Int	ความถี่ของ word ที่เกิด

ตารางที่ 5.8 โครงสร้าง TABLE wordlist_trigram

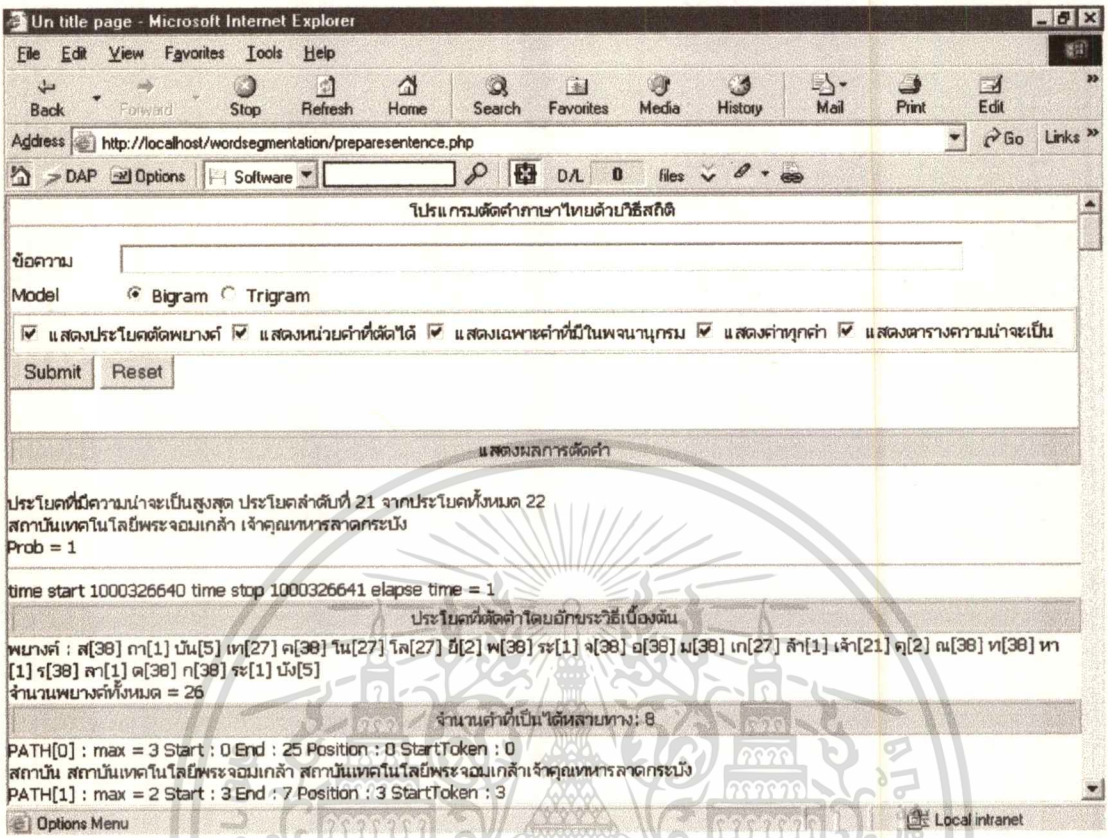
5.3 โปรแกรมการตัดคำ



รูปที่ 5.4 แสดงโปรแกรมสำหรับการตัดคำ

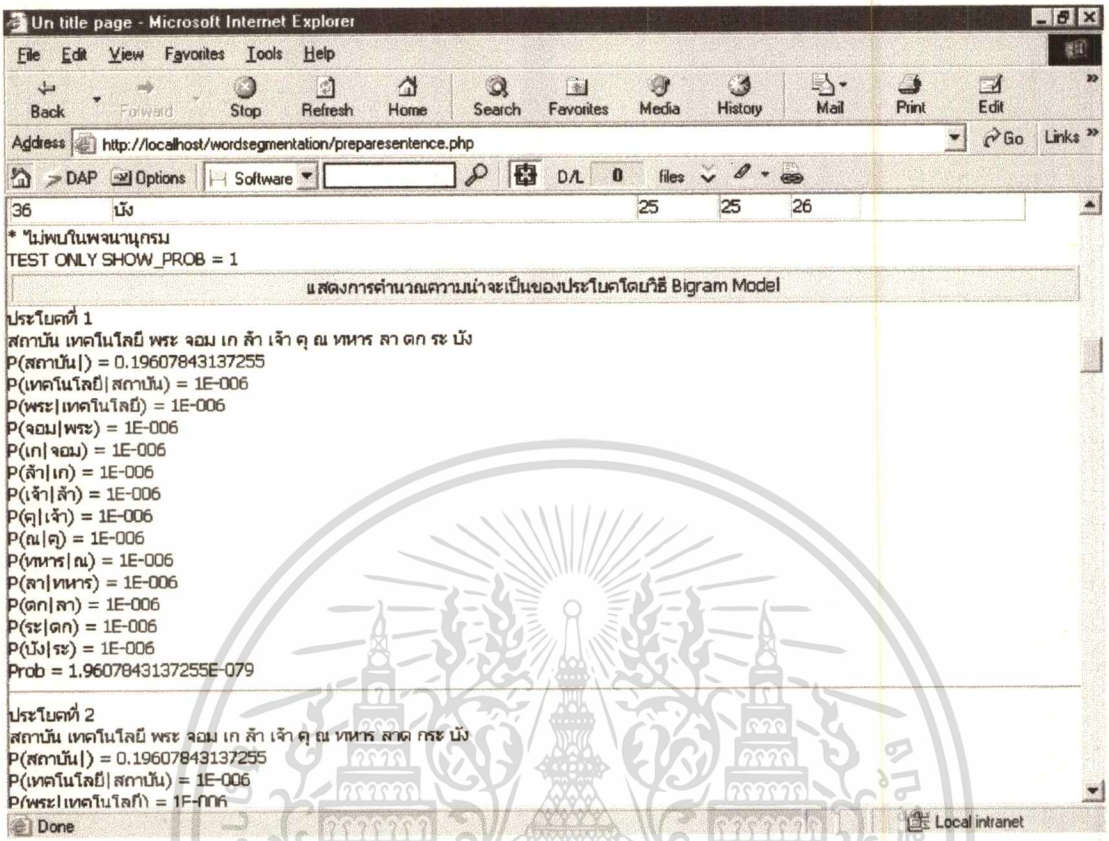
รูปที่ 5.4 เป็นตัวอย่างอินเตอร์เฟสของโปรแกรมสำหรับการตัดคำภาษาไทย โดยโปรแกรมนี้จะรับประโยคภาษาไทยที่ช่องประโยค และมีการปรับค่าพารามิเตอร์ดังนี้

- **Model** สำหรับเลือกการตัดคำโดยพิจารณาจากค่าสถิติ Bi-gram หรือ Tri-gram
- **แสดงประโยคตัดพยางค์** ถ้าต้องการแสดงผลการตัดพยางค์เบื้องต้น
- **แสดงหน่วยคำที่ตัดได้** ถ้าต้องการแสดงผลคำที่เกิดจากการจับกันของพยางค์ทั้งหมด
- **แสดงเฉพาะคำที่มีในพจนานุกรม** จะแสดงประโยคที่เป็นไปได้ทุกๆ ประโยคจะประกอบด้วยคำที่มีอยู่ในพจนานุกรม และประโยคที่มีพยางค์ที่ไม่มีอยู่ในพจนานุกรมจะถูกละทิ้ง
- **แสดงค่าทุกค่า** โปรแกรมจะแสดงประโยคที่เป็นไปได้ทั้งหมดและแสดงผลการคำนวณความน่าจะเป็นที่เกิดประโยค
- **แสดงตารางความน่าจะเป็น** ในการคำนวณความน่าจะเป็นของการเกิดประโยคนั้นจะคำนวณความน่าจะเป็นของคำแต่ละคำในประโยค เพื่อลดเวลาประมวลผลจะเก็บค่าความน่าจะเป็นเหล่านี้ไว้ในหน่วยความจำเพื่อค้นหาในกรณีที่ต้องคำนวณคำนั้นซ้ำ จะได้หาค่าอย่างรวดเร็ว
- **Process** เริ่มต้นกระบวนการตัดคำ ผลลัพธ์ของการตัดคำแสดงได้ในรูปที่ 5.5



รูปที่ 5.5 ผลลัพธ์การทำงานของโปรแกรม

- **แสดงผลการตัดคำ** เป็นส่วนที่แสดงประโยคที่ตัดคำได้ดีที่สุด และ ผลการคำนวณค่าความน่าจะเป็นของการเกิดประโยค พร้อมทั้งแสดงค่าความน่าจะเป็นของการเกิดประโยคนี้
- **ประโยคที่ตัดคำโดยอักขระวิธีเบื้องต้น** แสดงการตัดคำเบื้องต้นโดยกฎไวยากรณ์ โดยจะแสดงกฎที่ใช้นำมาตัดพยางค์ด้วย

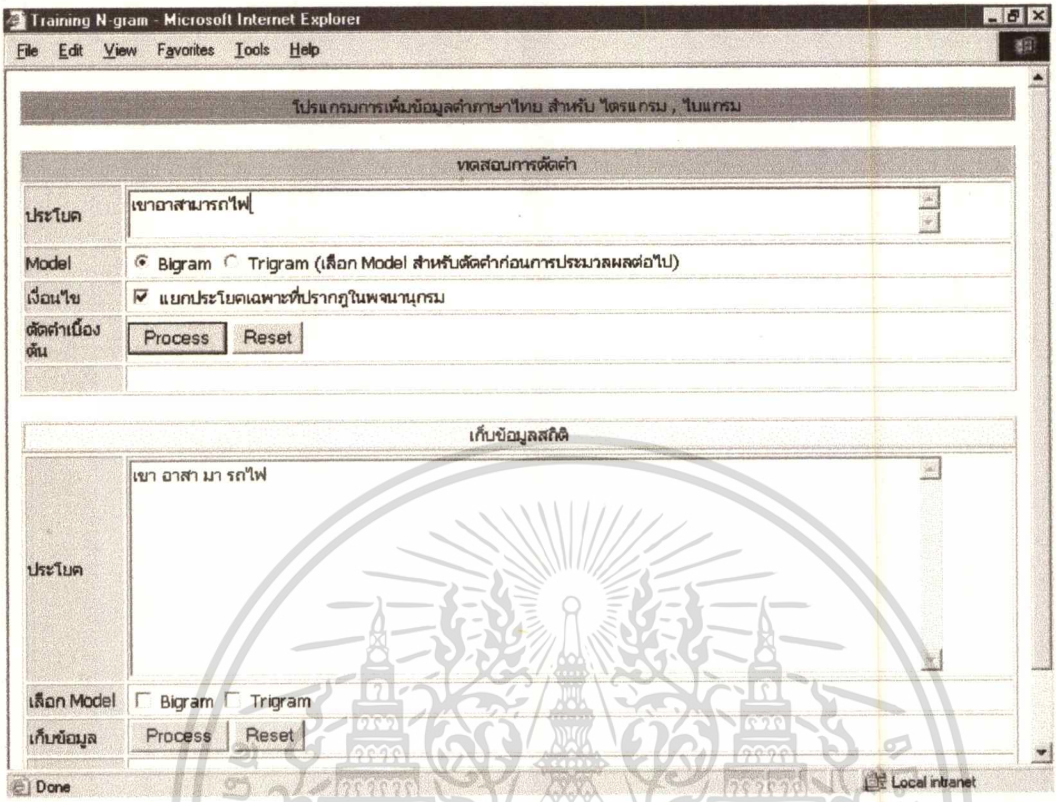


รูปที่ 5.6 แสดงผลการทำงานของโปรแกรม (ต่อ)

แสดงผลประโยคที่เป็นไปได้ทั้งหมด พร้อมทั้งแสดงการคำนวณค่าความน่าจะเป็นของแต่ละประโยค

5.4 โปรแกรมสำหรับการเพิ่มฐานข้อมูล Bigram , Trigram

เป็นโปรแกรมสำหรับเพิ่มการเก็บข้อมูลสถิติให้กับฐานข้อมูล bigram , trigram ดังในรูปที่ 5.7



รูปที่ 5.7 โปรแกรมเพิ่มข้อมูลสถิติให้กับฐานข้อมูล

ทดสอบการตัดคำ จะทำการทดสอบการตัดคำภาษาไทยและส่งข้อมูลที่ตัดคำได้ส่งให้ส่วนของ การเก็บข้อมูลสถิติ มีการทำงานคือ

- รับข้อมูลภาษาไทย
- เลือก Model สำหรับการคำนวณการตัด Bi-gram , Tri-gram
- เลือกเฉพาะประโยคเฉพาะที่ทุกคำมีอยู่ในพจนานุกรมหรือไม่

เมื่อ click ปุ่ม Process โปรแกรมจะ สร้าง Object สำหรับการตัดคำและ initial ค่า พร้อมส่งคำสั่งการตัดคำตามแบบโมเดลที่เลือก ทำการตัดคำพร้อมทั้งส่งประโยคผลลัพธ์ที่ตัดคำแล้วให้ส่วนของการเพิ่มข้อมูลสถิติเพื่อทำการปรับแต่งประโยคให้ถูกต้องต่อไป

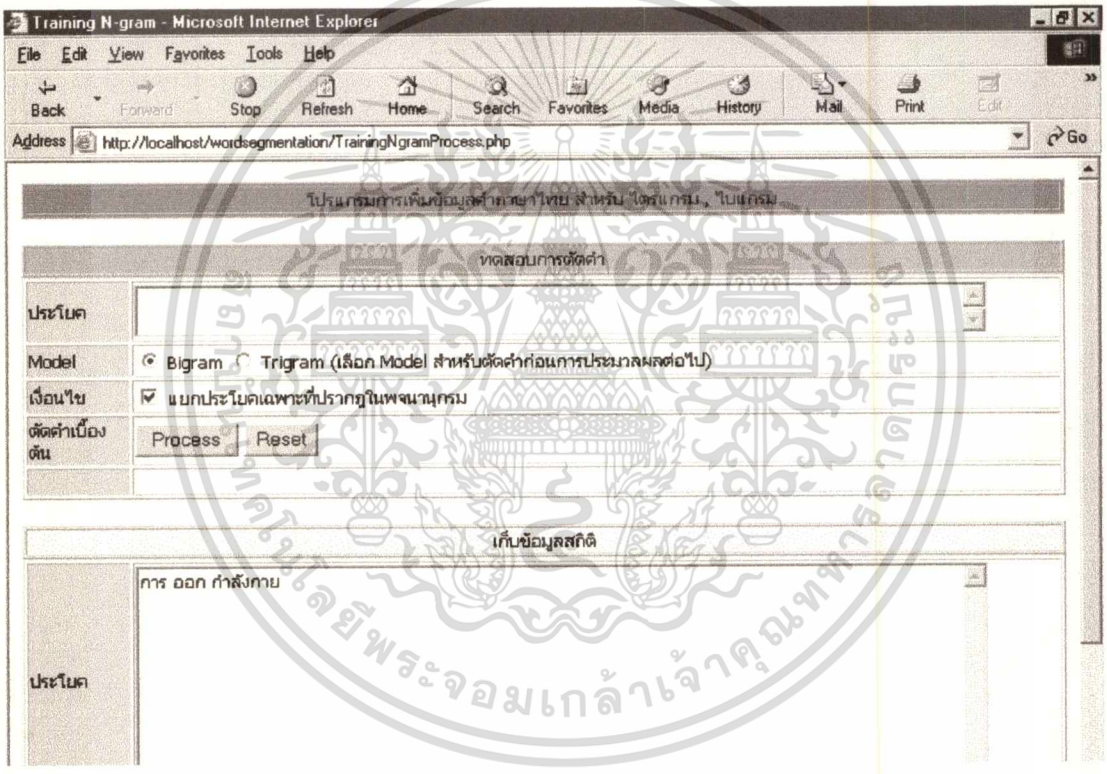
เก็บข้อมูลสถิติ ในส่วนนี้จะเป็นการนำประโยคที่ถูกตัดคำแล้ว โดยวิธีการจาก **ทดสอบการตัดคำ** ที่ส่งค่ามาให้ หรือ ป้อนข้อมูลที่ตัดคำแล้วข้อมูลด้วยตนเองก็ได้ นำข้อมูลไปปรับปรุงฐานข้อมูลสถิติต่างๆ มีการทำงานคือ

- รับประโยคที่ตัดคำแล้ว ยอมรับการแก้ไขปรับปรุงจากผู้ใช้งาน
- เลือกปรับปรุงฐานข้อมูล Bigram , Trigram หรือ ทั้งปรับปรุงทั้งสองฐานข้อมูล

5.5 ผลการทดสอบโปรแกรม

5.5.1 การตัดคำจากประโยคทั่วไป

โปรแกรมสามารถตัดคำจากประโยคทั่วๆไปได้อย่างถูกต้อง ดังประโยคตัวอย่าง



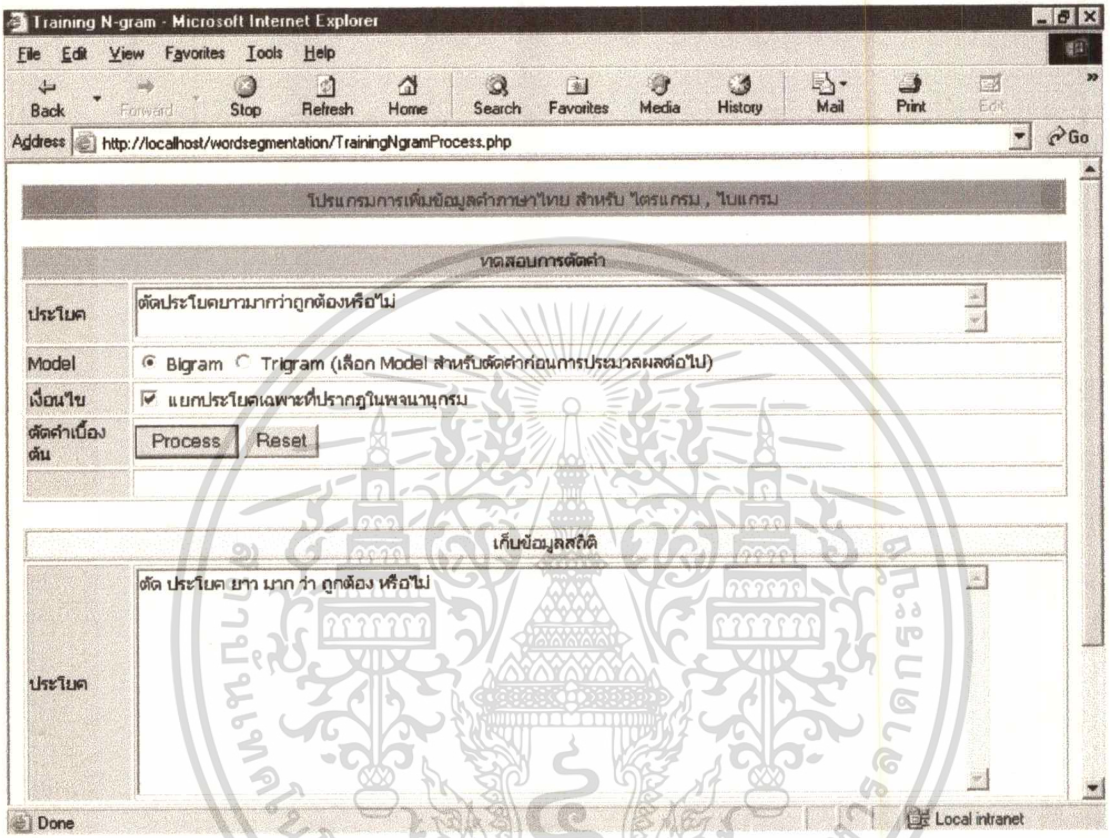
รูปที่ 5.8 การตัดคำของโปรแกรม

รูปที่ 5.8 จะเห็นได้ว่าโปรแกรมสามารถตัดคำประโยคทั่วๆ ไปได้อย่างถูกต้อง

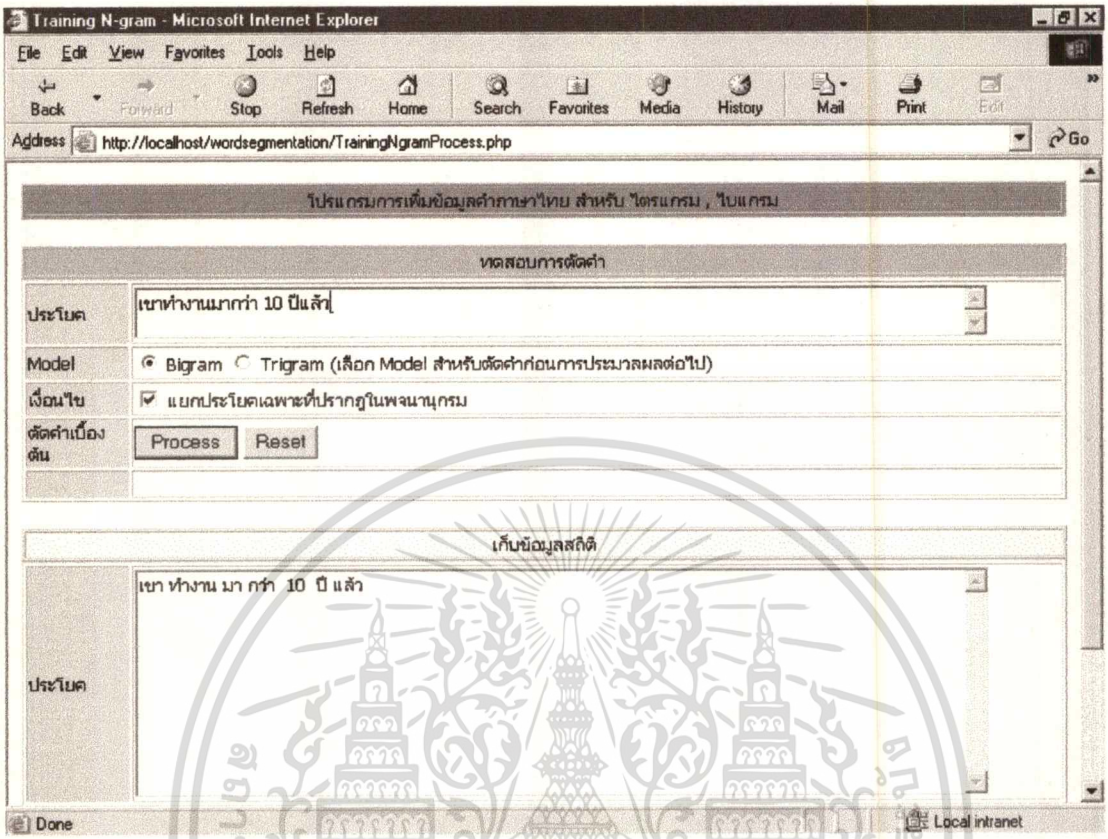
5.5.2 การตัดคำจากประโยคที่กำกวม

ในประโยคที่มีความกำกวมสามารถตัดคำได้ได้หลายแบบ เช่น คำว่า "มากว่า" อาจแบ่งแยกคำได้เป็น *มา / กว่า* และ *มาก / ว่า*

โปรแกรมสามารถตัดคำได้ถูกต้องก็ต่อเมื่อ ในฐานข้อมูลมีตัวอย่างการเกิดของคำนั้นๆ อยู่และสามารถให้ค่าสถิติการเกิดของเกิดของคำ ดังตัวอย่างในรูปที่ 5.9 และ 5.10



รูปที่ 5.9 การทดสอบการตัดคำที่เป็นไปได้หลายแบบ



รูปที่ 5.10 การทดสอบการตัดคำที่เป็นไปได้หลายแบบ (ต่อ)

จากรูปที่ 5.9 จะเห็นได้ว่าโปรแกรมสามารถตัดคำได้ถูกต้องโดยเลือกประโยค "ตัด /ประโยค/ยาว/มาก /ว่า /ถูกต้อง /หรือไม่"

จากรูปที่ 5.10 จะเห็นได้ว่าโปรแกรมสามารถตัดคำได้ถูกต้องโดยเลือกประโยค "เขา / ทำงาน / มา /กว่า / 10 / ปี / แล้ว"

ทั้งนี้เพราะฐานข้อมูลสำหรับ Bigram และ Trigram นั้นมีข้อมูลสำหรับการเกิดคำ "มา / กว่า", "มาก /ว่า" อยู่แล้ว

บทที่ 6

บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

ในงานทางด้านภาษาศาสตร์มีนักวิจัยหลายท่านให้ความสนใจและพัฒนาต่อเนื่อง ในการตัดคำภาษาไทยนี้เป็นส่วนหนึ่งของงานทางด้านภาษาศาสตร์ ซึ่งในโครงงานพัฒนาระบบงานนี้ได้นำเสนอวิธีการตัดคำ โดยคัดเลือกประโยคที่มีความน่าจะเป็นสูงสุดจากประโยคที่เป็นไปได้ทั้งหมด โดยอาศัยหลักการโมเดลทางสถิติแบบ n-gram โดยเฉพาะ bigram และ trigram ซึ่งคาดว่าจะให้ความถูกต้องในการตัดคำมากกว่าวิธีการตัดคำ โดยหลักไวยากรณ์ หรือ วิธีใช้พจนานุกรมเพียงอย่างเดียว อีกทั้งยังได้พัฒนาโปรแกรมสำหรับเพิ่มค่าสถิติการเกิดของคำในฐานข้อมูล bigram , trigram ทำให้เมื่อสามารถขยายความถูกต้องในการคำนวณได้

สำหรับอัลกอริทึมของโปรแกรมนี้ พยายามทำให้เกิดความเร็วมากที่สุด โดยพยามใช้วิธีควบคู่กับหลายอย่างคือ

1. ใช้กฎไวยากรณ์เพื่อตัดพยางค์ที่เล็กที่สุดเพื่อลดความซับซ้อนของการจับกันของอักขระประโยค
2. ใช้พจนานุกรมสำหรับตรวจสอบคำที่เกิดจากการจับกับของพยางค์ต่างๆ เพื่อให้มั่นใจได้ว่าความถูกต้องของการตัดคำในอัตราสูง

6.2 ข้อเสนอแนะ

คลังข้อมูลทางภาษาที่นำมาใช้ในโครงงานพัฒนาระบบงานนี้ มีความสำคัญยิ่งในการให้ความถูกต้องแม่นยำในการตัดคำ ซึ่งคลังข้อมูล Orchid Corpus นี้ส่วนใหญ่ทำมาจากเอกสารที่เกี่ยวข้องกับเทคโนโลยี , วิทยาศาสตร์ , คอมพิวเตอร์ ดังนั้นในการทดสอบการตัดประโยคของโปรแกรมนี้ ถ้าประโยคอยู่ในแนวทางเดียวกับคลังข้อมูล Orchid Corpus จะพบว่าสามารถให้ความถูกต้องแม่นยำ

ในอัตราที่สูง ทำนองเดียวกัน เมื่อนำไปตัดคำจากประโยคที่ไม่เกี่ยวข้องความถูกต้องแม่นยำจะลดลงไป ฉะนั้นแล้วการทำให้โปรแกรมสามารถตัดคำได้ถูกต้อง คลังข้อมูลควรมีขนาดใหญ่เพียงพอ และประโยคในการสร้างค่าสถิติในฐานข้อมูลควรมาจากแหล่งข้อมูลที่หลากหลาย เช่น แหล่งข้อมูลนิตยสารประเภทต่างๆ หนังสือทั่วไป หนังสือพิมพ์ เป็นต้น

การบริหารคลังข้อมูลทางภาษานี้จำเป็นต้องอาศัยนักภาษาศาสตร์ในการตรวจสอบข้อมูลก่อนการนำเข้าในฐานข้อมูลเพราะจะทำให้ข้อมูลนำเข้ามีความถูกต้องตรงตามหลักภาษา

สิ่งที่ต้องเพิ่มเติมคือ การจัดการกับคำที่ไม่ปรากฏในพจนานุกรม (Unknown Word) แบบต่างๆ เช่น ชื่อคน สัตว์ สิ่งของ สถานที่ คำเฉพาะ คำทับศัพท์ ตัวเลข ที่แทรกอยู่ในประโยค ในการค้นหา unknown word เหล่านี้ได้มีผู้คิด อัลกอริทึม ที่น่าจะนำมาประยุกต์ใช้งานได้

บรรณานุกรม

Kanlayanawat Witoon, Prasitjutrakul Somchai. 1997. "Automatic Indexing for Thai Text with Unknown Words using Trie Structure." *NLPRS*, 115-120.

Kawtrakul Asnee, Poovorawan Yuen. 1996. "Corpus Based Thai Lexical Analysis." Research No. ค-02.38, Kasetsart University.

Kawtrakul Asanee, Thumkanon Chalatip, Poovorawn Yuen. 1997. "Automatic Thai Unknown Word Recognition." *NLPRS*, 341-346.

Kooptiwoot Chompunuch. 1999. "Segmentation of Ambiguous Thai Words by Inductive Logic Programming." Master Thesis, Chulalongkorn University.

Meknavin Surapant, Charoenpornsawat Paisarn, Kijsirikul Boonserm. 1997. "Feature-based Thai Word Segmentation." *NLPRS*, 41-46.

Potipti Tanapong, Sornlertlamvanich Virach, Thatsanee Charoenporn. 2000. "Automatic Corpus-based Thai Word Extraction." *SNLP*, Chieng Mai, Thailand.

ThaiTae Sorsak. 1998. "Thai Word Separator using Hashing Dictionary.", *ISBN 974-622-303-8*, King Mongkut's Institute of Technology Ladkrabang.

Thatsanee Charoenporn, Sornlertlamvanich Virach, Hitoshi Isahara. 1997. "Building A Large Thai Text Corpus Part Of Speech Tagged Corpus: ORCHID." *TR-NECTEC-1997-001*, URL: http://www.links.nectec.or.th/virach/publication.html.

ภาคผนวก ก

กฎไวยากรณ์ที่ใช้ในการตัดพยางค์ในประโยค

กำหนดให้

c = พยัญชนะในภาษาไทย

t = ◌ั ◌ิ ◌ี ◌ึ ◌ื ◌ุ ◌ู ◌็

s = ตัวสะกดในภาษาไทย

กฎที่	รูปแบบกฎ	ตัวอย่าง / หมายเหตุ
1	/^([c]◌ื[t]?[s]?)/	มือ ถือ
2	/^([c][◌ิ◌ี◌ึ◌ื◌ุ◌ู◌็][t]?)/	สุ ที่
3	/^([c][t][◌ะ◌า◌ำ]?)/	ค่ะ ค่ำ ก่ะ
4	/^([c] [◌ะ◌า◌ำ])/	กา ทำ
5	/^([c]◌ั[t]?[s])/	ตั่น กั้น
6	/^(เ[c][t]?าะ)/	เกาะ เราะ
7	/^([แโ][c][t]?[าะ])/	เรา เขา โก๊ะ
8	/^([เ แ][c]◌็[s])/	เข็น แข็ง เก็บ
9	/^(เ[c]◌ี[t]?ย)/	เสี่ย เสีย
10	/^(เ[c]◌ื[t]?อ)/	เสื่อ
11	/^(เ[กขค]วี[t]?ย)/	เกวีย
12	/^([ไใ]ห[งญนมยรลว][t]?)/	ใหญ่ ใหม่
13	/^([แเไใ][c][t]?)/	โห่ แห่
14	/^([c]์)/	ด์ ห์
15	/^(ฯลฯ)/	

กฎที่	รูปแบบกฎ	ตัวอย่าง / หมายเหตุ
16	/^[ะ า ำ เ แ โ ใ ไ]/	
17	/^[a-zA-Z0-9๐-๙]+/	ตัวอักษรภาษาอังกฤษ ,เลขอารบิค , เลขไทย
18	/^[ ]+/	ช่องว่างเกินตั้งแต่ 1 ช่องว่างขึ้นไป
19	/^[\S]/	ตัวอักษรที่ไม่เข้าเกณฑ์ข้างบนทั้งหมด

หมายเหตุ การเรียงลำดับก่อนหลังของกฎมีผลต่อการตัดพยางค์ในประโยค

อธิบายสัญลักษณ์การเปรียบเทียบคำแบบ Regular Expression

เครื่องหมาย	ความหมาย
^	เปรียบเทียบกฎจากตำแหน่งเริ่มต้นของประโยค
[ ]	อักขระที่อยู่ในเครื่องหมาย [ ] ให้ค่าเป็นจริงอย่างหนึ่งอย่างใด เช่น [กข] อักขระ ก หรือ ข อย่างหนึ่งอย่างใดและเพียงตัวเดียวเท่านั้น
?	อักขระหรือกลุ่มของอักขระจะมีหรือไม่ก็ได้
+	จำนวนที่ปรากฏ >= 1

โปรแกรม php สำหรับตัดพยางค์

```
<?php
function word_seg($txt,$start_sep_char,$end_sep_char){
 $c ="ก-ฮ";  /consonant
 $s  ="งมยวกรขคฆดตถฏฐฒศษสจชทบปพฟภนฌลญฬฑ"; //speller
 $v="-ะ-าิีึืุูแโ-ำไใฤๅ"; // vowel
 $tonal    = "่้๊๋";
 $numeral   ="0123456789๐๑๒๓๔๕๖๗๘๙";
 $special = "ฯ!@#$%^&*()-_+=ๆ";
 $txt_segment = "";
 $txt=preg_replace("/[\\r\\n\\t\\v\\f]/","",$txt);
 $txt_rule = "rules :  ";
 $loop = true;
 while($loop)  {
 if(    preg_match("/^([$c][ัิีึืุู็่้๊๋์][$t]?)/",$txt,$args)       )
 {    $txt_rule .= "1 ";
      $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
      $txt = substr($txt,strlen($args[0]));
      if( (strlen($args[0])==0) or (strlen($txt)==0) )
        $loop=false;
 }
 else if( preg_match( "/^([$c][ัิีึืุู็่้๊๋์][$t]?)/",$txt,$args) )  {
   $txt_rule .= "2 ";
   $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
   $txt = substr($txt,strlen($args[0]));
   if( (strlen($args[0])==0) or (strlen($txt)==0) )
     $loop=false;
 }
 else if( preg_match( "/^([$c][$t][_ะ _า ำ]?)/",$txt,$args) )  {
   $txt_rule .= "3 ";
```

```
 if( (strlen($args[0])==0) or (strlen($txt)==0) )
    $loop=false;
}
else if( preg_match( "/^([แเ][กขคตทบปพฟจชศส]ร[$tonal]?ะ)/",$txt,$args) ) {
 $txt_rule .= "8 ";
 $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
 $txt = substr($txt,strlen($args[0]));
 if( (strlen($args[0])==0) or (strlen($txt)==0) )
    $loop=false;
}
else if( preg_match( "/^(เ[$c]็[$t]?ย)/",$txt,$args) ) {
 $txt_rule .= "9 ";
 $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
 $txt = substr($txt,strlen($args[0]));
 if( (strlen($args[0])==0) or (strlen($txt)==0) )
    $loop=false;
}
else if( preg_match( "/^(เ[$c]็[$t]?ย)/",$txt,$args) ) {
 $txt_rule .= "10 ";
 $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
 $txt = substr($txt,strlen($args[0]));
 if( (strlen($args[0])==0) or (strlen($txt)==0) )
        $loop=false;
}
else if( preg_match( "/^(เ[กขค]ี[$t]?ย)/",$txt,$args) ) {
 $txt_rule .= "11 ";
 $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
 $txt = substr($txt,strlen($args[0]));
 if( (strlen($args[0])==0) or (strlen($txt)==0) )
        $loop=false;
```

```
  $txt_rule .= "16 ";
  $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
  $txt = substr($txt,strlen($args[0]));
  if( (strlen($args[0])==0) or (strlen($txt)==0) )
     $loop=false;
 }
 else if( preg_match( "/^[a-zA-Z0-9๐-๙]+/",$txt,$args) )  {
  $txt_rule .= "17 ";
  $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
  $txt = substr($txt,strlen($args[0]));
  if( (strlen($args[0])==0) or (strlen($txt)==0) )
     $loop=false;
 }
 else if( preg_match( "/^[ ]+/",$txt,$args) )  {
  $txt_rule .= "18 ";
        $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
        $txt = substr($txt,strlen($args[0]));
    if( (strlen($args[0])==0) or (strlen($txt)==0) )
     $loop=false;
 }
 else if( preg_match( "/^[\S]/",$txt,$args) )
 {
  $txt_rule .= "19 ";
  $txt_segment .= $start_sep_char.$args[0].$end_sep_char;
  $txt = substr($txt,strlen($args[0]));
  if( (strlen($args[0])==0) or (strlen($txt)==0) )
     $loop=false;
 }
 } // while
$rtn[0] = $txt_segment;
```

ภาคผนวก ข

โครงสร้าง Orchid Corpus

โครงงานพัฒนาระบบงานนี้ใช้คลังข้อมูลทางภาษา Orchid Corpus ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) มีรูปแบบดังนี้

Mark-up for text information line

Mark-up	*Description*
%TTitle:	Title of the document written in Thai.
%ETitle:	Title of the document written in English.
%TAuthor:	Authors name written in Thai.
%EAuthor:	Authors name written in English.
%TInbook:	Title of the book where the document exists, written in Thai.
%EInbook:	Title of the book where the document exists, written in English.
%TPublisher:	Publisher of the book, written in Thai.
%EPublisher:	Publisher of the book, written in English.
%Page:	Page number or the range of pages of the document.
%Year:	Published year (A.D.).
%File:	File number of the document. A long document may be separated into a number of files.

Mark-up for Numbering Line

Mark-up	*Description*
#P[number]	Paragraph number of a text. The number in the bracket is shown in a sequence within a text.
#[number]	Sentence number of a paragraph. The number in the bracket is shown in a sequence within a paragraph.

Special Characters for Mark-up

Mark-up	*Description*
\\	Line break symbol.
//	Sentence break symbol.
/[POS]	Tag marker for the appropriate POS of a word.

Defined Strings for Special Characters

Special Character	*Defined String*
space	<space>
!	<exclamation>
"	<quotation>
#	<number>
$	<dollar>
%	<percent>
&	<ampersand>
'	<apostrophe>
(	<left_parenthesis>
)	<right_parenthesis>
*	<asterisk>
+	<plus>
,	<comma>
-	<minus>
.	<full_stop>

Special Character	*Defined String*
/	<slash>
:	<colon>
;	<semi_colon>
<	<less_than>
=	<equal>
>	<greater_than>
?	<question_mark>
@	<at_mark>
[	<left_square_bracket>
]	<right_square_bracket>
^	<circumflex_accent>
_	<low_line>
{	<left_curly_bracket>
}	<right_curly_bracket>
~	<tilde>

ตัวอย่างข้อมูลใน Orchid Corpus

%TTitle: การประชุมทางวิชาการ ครั้งที่ 1

%ETitle: [1st Annual Conference]

%TAuthor:

%EAuthor:

%TInbook: การประชุมทางวิชาการ ครั้งที่ 1, โครงการวิจัยและพัฒนาอิเล็กทรอนิกส์และคอมพิวเตอร์, ปีงบประมาณ 2531, เล่ม 1

%EInbook: The 1st Annual Conference, Electronics and Computer Research and Development Project, Fiscal Year 1988, Book 1

%TPublisher: ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ, กระทรวงวิทยาศาสตร์เทคโนโลยีและการพลังงาน

%EPublisher: National Electronics and Computer Technology Center, Ministry of Science, Technology and Energy

%Page:

%Year: 1989

%File:

#P1

#1

การประชุมทางวิชาการ ครั้งที่ 1//

การ/FIXN

ประชุม/VACT

ทาง/NCMN

วิชาการ/NCMN

<space>/PUNC

ครั้ง/CFQC

ที่ 1/DONM

//

#2

โครงการวิจัยและพัฒนาอิเล็กทรอนิกส์และคอมพิวเตอร์//

โครงการวิจัยและพัฒนา/NCMN

อิเล็กทรอนิกส์/NCMN

และ/JCRG

คอมพิวเตอร์/NCMN

//

#3

ปีงบประมาณ 2531//

ปีงบประมาณ/NCMN

<space>/PUNC

2531/NCNM

//

ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ//

ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ/NPRP

//

#2

กระทรวงวิทยาศาสตร์ เทคโนโลยีและการพลังงาน//

กระทรวงวิทยาศาสตร์ เทคโนโลยีและการพลังงาน/NPRP

//

ฯพณฯ รัฐมนตรีว่าการกระทรวงวิทยาศาสตร์\\

เทคโนโลยีและการพลังงาน//

ฯพณฯ/NTTL

<space>/PUNC

รัฐมนตรีว่าการ/NCMN

กระทรวงวิทยาศาสตร์ เทคโนโลยีและการพลังงาน/NPRP

//

#P3

#1

ประเทศไทยได้มีการปรับเปลี่ยนโครงสร้างในการพัฒนาเศรษฐกิจของประเทศ \\

จากประเทศเกษตรกรรมไปสู่ความเป็นประเทศอุตสาหกรรมมากยิ่งขึ้น//

ประเทศไทย/NPRP

ได้/XVAM

มี/VSTA

การ/FIXN

ปรับเปลี่ยน/VACT

โครงสร้าง/NCMN

ใน/RPRE

การ/FIXN

พัฒนา/VACT

เศรษฐกิจ/NCMN

ของ/RPRE

ประเทศ/NCMN

<space>/PUNC

จาก/RPRE

ประเทศ/NCMN

เกษตรกรรม/NCMN

ไปสู่/RPRE

ความ/FIXN

เป็น/VSTA

ประเทศอุตสาหกรรม/NCMN

มาก/ADVN

ยิ่งขึ้น/ADVN

//

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย ศิวโรจน์ ภูวิภิรมย์
วัน เดือน ปีเกิด	27 พฤษภาคม 2506
สถานที่เกิด	กรุงเทพมหานครฯ
วุฒิการศึกษาระดับปริญญาตรี	ว.ท.บ.(สถิติและคอมพิวเตอร์)
สถาบันที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ มหาวิทยาลัยรามคำแหง
ปีที่สำเร็จการศึกษา	2530