โครงการเผยแพร่ผลงานวิชาการ
คณะอักษรศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ลำดับที่ 212

การประมวลผลภาษาไทย

วิโรจน์ อรุณมานะกุล

โครงการเผยแพร่ผลงานวิชาการ
คณะอักษรศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ลำดับที่ 212

การประมวลผลภาษาไทย

วิโรจน์ อรุณมานะกุล

โครงการเผยแพร่ผลงานวิชาการ
คณะอักษรศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ

วิโรจน์ อรุณมานะกุล

การประมวลผลภาษาไทย.— กรุงเทพฯ : โครงการเผยแพร่ผลงานวิชาการ คณะอักษรศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2567.

243 หน้า

1.ภาษาศาสตร์คอมพิวเตอร์. 2.การประมวลผลคำ. I. ชื่อเรื่อง.

410.285

ISBN 978-616-407-970-0

คำนำ

หนังสือเล่มนี้มีจุดมุ่งหมายเพื่อให้ความรู้ภาษาไทยและการประมวลผลภาษาสำหรับผู้ที่สนใจงานประมวลผลภาษาไทยเป็นหลัก ผู้อ่านควรมีความรู้พื้นฐานการประมวลผลภาษามาบ้างแล้ว จากการอ่านตำราพื้นฐาน เช่น Speech and Language Processing (Jurafsjy & Martin, 2021) หรือ Foundations of Statistical Natural Language Processing (Manning & Schütze, 1999) เพราะจะมีการอ้างถึงศัพท์และมโนทัศน์พื้นฐานรวมถึงวิธีการประมวลผลภาษา เช่น n-gram, decision tree, conditional random field เป็นต้น หนังสือเล่มนี้จะให้ความสำคัญกับการเข้าใจลักษณะภาษาไทยที่มีผลต่อการออกแบบและพัฒนาระบบการประมวลผลต่าง ๆ แม้ว่าหนังสือนี้ไม่ได้เน้นการใช้โครงข่ายประสาทแบบต่าง ๆ ในการประมวลผลซึ่งเป็นแนวทางที่นิยมใช้ในปัจจุบัน แต่ผู้เขียนเชื่อว่าความเข้าใจในปัญหาการประมวลผลภาษาไทยยังเป็นความรู้พื้นฐานที่จำเป็น เพราะไม่ว่าจะเลือกใช้โมเดลแบบไหนประมวลผลภาษา ปัญหาที่ต้องการแก้ไขยังเป็นปัญหาเดิม แนวทางการทำงานแบบดั้งเดิมแม้ว่าจะมีประสิทธิภาพไม่ดีเท่าโมเดลที่ใช้โครงข่ายประสาท แต่จะช่วยให้ผู้อ่านเห็นและเข้าใจปัญหาในงานประมวลผลต่าง ๆ ได้ชัดเจนกว่า ในขณะที่การใช้แบบจำลองโครงข่ายประสาททำการประมวลผลภาษาจะได้ผลลัพธ์ที่ดีกว่าเมื่อมีปริมาณข้อมูลมากพอ แต่สิ่งที่ขาดไปคือความเข้าใจในกระบวนการภายในของภาษาซึ่งเป็นสิ่งที่นักภาษาศาสตร์สนใจศึกษามาโดยตลอด เปรียบเหมือนการเล่นหมากรุกหรือหมากล้อมกับคอมพิวเตอร์ แม้ว่าคอมพิวเตอร์จะเล่นได้เก่งกว่าเสมอ แต่มนุษย์ก็ยังคงเล่นเกมเหล่านี้อยู่เพราะความสนใจในเกมมากกว่าการเอาชนะ นักภาษาศาสตร์เองก็สนใจอยากเข้าใจและหาคำอธิบายกระบวนการทางภาษาที่เกิดขึ้นมากกว่าที่จะสนใจแต่เพียงผลสัมฤทธิ์ว่าคอมพิวเตอร์ทำงานได้ดีมากเพียงใด

เนื้อหาในหนังสือแบ่งเป็นสามตอน ตอนแรกว่าด้วยการประมวลผลระดับคำซึ่งเป็นระดับพื้นฐาน จากนั้นจะกล่าวถึงความรู้เรื่องคำประสมและการระบุหาคำประสมในภาษาไทยเพราะการประสมคำเป็นวิธีพื้นฐานของการสร้างคำภาษาไทย ชื่อเฉพาะและการรู้จำชื่อเฉพาะภาษาไทยก็เป็นเรื่องพื้นฐานอีกเรื่องหนึ่งในการประมวลผลภาษาไทยที่จะต้องจำแนกข้อมูลส่วนนี้ออกมาให้ได้ก่อนจะประมวลผลในระดับที่สูงขึ้น ตอนที่สองเป็นเรื่องต่อจากระดับคำ เริ่มจากการกำหนดชุดหมวดคำและการกำกับหมวดคำซึ่งเป็นพื้นฐานสำหรับการประมวลผลในระดับวากยสัมพันธ์หรือการรวมคำเป็นหน่วยที่ใหญ่ขึ้น แนวทางการวิเคราะห์ทางวากยสัมพันธ์ที่แตกต่างกันก็มีผลให้มีการกำหนดชุดหมวดคำที่ต่างกันได้ ในที่นี้ ผู้เขียนเลือกใช้ไวยากรณ์พึ่งพาตามแนว Universal Dependencies เพราะเชื่อว่าเหมาะกับภาษาไทยและยังทำให้เทียบเคียงกับผลการวิเคราะห์ของภาษาอื่นได้ จากนั้นจะเป็นเรื่องหน่วยปริจเฉทพื้นฐาน เพราะในการประมวลผลข้อความต่อเนื่อง เรามักนิยมแบ่งข้อความต่อเนื่องเป็นหน่วยย่อย ๆ เพื่อใช้ในการประมวลผลมากกว่าจะประมวลผลข้อความทั้งหมดพร้อมกัน นอกจากนี้การประมวลผลข้อความอาจใช้วิธีการที่ไม่จำเป็นต้องวิเคราะห์ตามขั้นตอนการวิเคราะห์ทางภาษาศาสตร์ก็ได้ งานหลายอย่างเลือกใช้วิธีการแปลงหน่วยภาษาเป็นเวกเตอร์เพื่อประมวลผลเลย ในส่วนนี้จึงยกตัวอย่างการสร้างและใช้เวกเตอร์คำด้วย Word2Vec สำหรับตอนสุดท้าย ผู้เขียนจะทดลองใช้ generative AI ในการทำงานประมวลผลภาษาไทย โดยบทแรกจะกล่าวถึงศักยภาพและความสามารถของ generative AI อย่าง ChatGPT เพื่อให้เห็นความสามารถด้านภาษา จากนั้นจึงทดลองใช้ ChatGPT เพื่อประมวลผลภาษาไทยเพื่อดูว่างานใดที่สามารถทำได้และทำได้ดีหรือไม่ ท้ายสุดจึงเสนอแนวทางให้ใช้ generative AI ประมวลผลภาษาไทยเพื่อประโยชน์ในการสร้างข้อมูลสำหรับใช้ศึกษาวิจัยและฝึกสอนโมเดลภาษาในงานต่าง ๆ

หนังสือเล่มนี้ไม่ได้กล่าวถึงเรื่องภาษาในระดับอักขรวิธี สำหรับผู้ที่สนใจเรื่องอักขรวิธีภาษาไทยสามารถหาความรู้ได้จากหนังสือ “อักขรวิธีไทยและการถอดอักษร

ระหว่างภาษาไทยและภาษาอังกฤษ” ซึ่งเคยตีพิมพ์เผยแพร่แล้ว ปัจจุบันสามารถโหลดจากเว็บผู้เขียนได้ เนื้อหาหลายส่วนในหนังสือนี้ได้เคยเผยแพร่เป็นบทความบนบล็อก Medium ให้คนทั่วไปอ่าน หรือเป็นเอกสารเผยแพร่บนหน้าเว็บของผู้เขียนเอง ผู้เขียนตัดสินใจรวบรวมงานเขียนเหล่านั้นมาเรียบเรียงใหม่และเขียนเพิ่มเติมส่วนที่ยังไม่ได้เขียนเพื่อให้เนื้อหาทั้งเล่มได้ความต่อเนื่องกัน ในหนังสือเล่มนี้ยังได้ผนวกการสาธิตโมดูลภาษา Python คือโมดูล TLTK (Thai Language Toolkit) ที่ผู้เขียนพัฒนาขึ้นสำหรับใช้ประกอบการสอนเรื่องการประมวลผลภาษาไทย ผู้ใช้สามารถเข้าไปดูใน Google collab ที่ทำไว้และทดลองใช้คำสั่งต่าง ๆ ตามที่แสดงได้

เนื่องจากหนังสือเล่มนี้มีหลายบทที่นำผลงานจากหลากหลายแหล่งของผู้เขียน บ้างเขียนเป็นรายงานวิจัย บ้างเขียนเป็นบทความเผยแพร่ผ่านบล็อก จึงทำให้ภาษาที่ใช้ไม่ต่อเนื่อง บางบทมีลักษณะเป็นการบอกเล่าด้วยภาษาพูดมากกว่า แม้ผู้เขียนจะพยายามปรับคำและการใช้ภาษาให้เป็นทางการขึ้นในการเรียบเรียงใหม่ครั้งนี้ แต่ลักษณะการเล่าเรื่องแบบชวนคิดชวนพิจารณาก็ยังคงอยู่ จึงอาจทำให้ผู้อ่านรู้สึกถึงความไม่สม่ำเสมอของภาษาหากอ่านต่อเนื่องระหว่างบทได้

ผู้เขียนหวังว่าผู้อ่านที่สนใจงานประมวลผลภาษาไทยจะได้เข้าใจลักษณะและธรรมชาติของภาษาไทยที่เป็นโจทย์หรือปัญหาสำหรับการประมวลผลภาษาไทยมากขึ้น เพราะแม้ว่าเทคโนโลยีการประมวลผลภาษาจะพัฒนาไปในทิศทางที่คอมพิวเตอร์สามารถทำงานและวิเคราะห์งานต่าง ๆ เองได้ง่ายมากขึ้น แต่เทคโนโลยีเหล่านั้นก็เป็นเพียงเครื่องมือหรือวิธีการที่ถูกนำมาใช้เพื่อประมวลผลภาษา ลักษณะเฉพาะของภาษาไทยเองก็ยังคงเป็นโจทย์หรือปัญหาที่ต้องประมวลผล ความเข้าใจในตัวปัญหาหรือลักษณะของภาษาไทยที่ส่งผลให้ระบบทำงานได้ดีหรือไม่ได้ดีจะช่วยให้เราสามารถปรับแต่งหรือเลือกใช้เครื่องมือวิธีการต่าง ๆ ได้อย่างมีหลักการหรือมีคำอธิบายมากขึ้น และที่สำคัญความเข้าใจในการประมวลผลภาษาคือความเข้าใจในความรู้เรื่องภาษาของมนุษย์เองที่จะตอบคำถามพื้นฐานว่าทำไมเราจึงใช้ภาษาสื่อสารและเข้าใจกันได้

สุดท้ายนี้ ผู้เขียนขอขอบคุณผู้ทรงคุณวุฒิที่ได้ให้ความเห็นและข้อเสนอแนะในการปรับปรุงแก้ไขหนังสือเล่มนี้ให้เป็นระบบ ถูกต้องและสมบูรณ์มากขึ้น ขอบคุณโครงการเผยแพร่ผลงานวิชาการ คณะอักษรศาสตร์และผู้มีส่วนเกี่ยวข้องที่ให้ความช่วยเหลือในการจัดทำเผยแพร่หนังสือเล่มนี้

๘ สิงหาคม ๒๕๖๗
วิโรจน์ อรุณมานะกุล
ผู้อำนวยการสถาบันภาษาไทยสิรินธร และ
อาจารย์ประจำภาควิชาภาษาศาสตร์ คณะอักษรศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

ตอน ๑ : เริ่มจากคำ

คำเป็นหน่วยพื้นฐานที่สำคัญของภาษาไทย และเป็นปัญหาแรก ๆ ที่ถูกกล่าวถึงในการประมวลผลภาษาไทย เหตุเพราะในภาษาไทยไม่ได้เขียนแยกเป็นคำ ๆ แบบภาษาอื่น ข้อมูลที่คอมพิวเตอร์เห็นหรือรับเข้าจึงเป็นสายอักขระที่ประกอบด้วยคำหลายคำเขียนต่อเนื่องกันไป ในตอนแรกนี้จึงได้กล่าวถึงปัญหาเรื่องคำและการตัดคำ ซึ่งต้องอาศัยความรู้ว่าคำคืออะไร การกำหนดขอบเขตคำเองก็ไม่ใช่เรื่องง่ายสำหรับมนุษย์โดยเฉพาะในกรณีของคำประสมที่มีโครงสร้างและความซับซ้อนได้คล้ายกับวลี นอกจากคำแล้ว ชื่อเฉพาะก็เป็นหน่วยภาษาประเภทหนึ่งที่ต้องระบุขอบเขตและชนิดของชื่อเฉพาะให้ได้ การประมวลผลคำและชื่อเฉพาะจึงเป็นงานพื้นฐานส่วนแรกสำหรับการประมวลผลภาษาไทย

การตัดคำในภาษาไทย[1]

ความเป็นมา

การตัดคำภาษาไทยเป็นงานพื้นฐานของการประมวลผลภาษาไทยที่ทำกันมาตั้งแต่แรกเริ่มของการใช้ภาษาไทยบนคอมพิวเตอร์ ภาษาไทยจำเป็นต้องมีการตัดคำด้วยเหตุว่าในการเขียนภาษาไทย เราไม่มีการวรรคระหว่างคำ คอมพิวเตอร์เห็นเป็นสายอักขระต่อเนื่องไปไม่รู้ว่าขอบเขตคำอยู่ที่ใด การตัดคำเมื่อแรกเริ่มเป็นการทำเพื่อช่วยงานพิมพ์ให้คอมพิวเตอร์ปัดข้อความขึ้นบรรทัดใหม่ให้ถูกต้อง ไม่ตัดแยกส่วนคำระหว่างบรรทัด งานช่วงแรกออกมาในรูปของการใช้กฎเพื่อระบุขอบเขตว่าควรจะตัดที่ตำแหน่งไหนได้ กฎต่าง ๆ ที่เสนอเป็นเรื่องของพยางค์มากกว่าตัดคำ เช่น ถ้าพบ "เ" จะตัดหลังสระนี้ไม่ได้ เพราะต้องมีพยัญชนะตามมาเสมอ (Thairatananond, 1981)

จากนั้นจึงมีแนวคิดที่ใช้พจนานุกรมช่วยตัด โดยคิดว่าจะต้องมีรายการคำเพื่อให้คอมพิวเตอร์รู้ว่ารูปใดเป็นคำและขอบเขตคำอยู่ที่ไหน แนวคิดพื้นฐานคือเอาคำในพจนานุกรมไปเทียบกับข้อมูลหากพบรูปตรงก็จะตัดได้ ซึ่งก็ทำแบบ longest matching ได้ (สมปรารถนา รัทยานนท์, 2535; รัตติกร วรากูลศิริพันธ์ และคณะ, 2538) คือตัดหาคำให้ยาวที่สุดก่อน แต่ก็จะเห็นว่ามีปัญหา เช่น *"ไปหามเหสี"* ตัด

---

[1] เรียบเรียงใหม่จากบล็อกผู้เขียน https://awirote.medium.com “การตัดคำภาษาไทย : ความเป็นมา” (23 มิถุนายน 2561), “การตัดคำภาษาไทย : ตัดคำอย่างไร” (23 มิถุนายน 2561), และ “การตัดคำภาษาไทย : ประเมินผลอย่างไร” (1 พฤศจิกายน 2561)

ออกมาเป็น *ไป|หาม|เห|สี* เลยมีการเสนออีกแนวคิดให้ตัดแบบ maximum matching (วิรัช ศรเลิศล้ำวาณิช, 2536) คือตัดคำให้ได้จำนวนคำน้อยที่สุด ตัวอย่างนี้จึงได้ *"ไป|หา|มเหสี"* เพราะมีจำนวนคำที่ได้น้อยกว่า

คำถามที่น่าสนใจ คือ ทำไมจึงเลือกการตัดคำที่ได้จำนวนคำน้อยสุด ทำไมไม่เลือกการตัดคำที่ได้จำนวนคำมากสุด คาดว่าเป็นเพราะภาษาไทยมีการสร้างคำจากการประสมคำจำนวนมาก เมื่อเจอคำอย่างเช่น *"หน้าต่าง"* โอกาสที่สายอักขระนี้จะเป็นคำประสมคำเดียวจะมากกว่าที่จะเป็นคำเดี่ยวสองคำ การเลือกผลที่มีจำนวนคำน้อยสุดจึงมีโอกาสถูกต้องมากกว่า

ข้อเสียของการตัดคำด้วยพจนานุกรม คือ ถ้าพบรูปคำที่ไม่รู้จักคือยังไม่ได้บรรจุลงในพจนานุกรมที่ใช้ก็จะตัดคำไม่ได้ ทำให้เกิดปัญหาที่เรียกว่า unknown word ซึ่งปัญหานี้อาจมาจากพจนานุกรมมีรายการคำไม่ครอบคลุมพอ หรือเป็นคำที่เกิดขึ้นใหม่ เป็นชื่อเฉพาะ เป็นคำทับศัพท์ หรือเป็นคำที่สะกดผิด เป็นต้น นอกจากปัญหา unknown word ปัญหาพื้นฐานเรื่องความกำกวมก็ยังคงอยู่ โดยเฉพาะเมื่อตัดแล้วได้จำนวนคำเท่ากัน ตัวอย่างที่มักยกมาใช้ เช่น *ตาก|ลม* กับ *ตา|กลม* ตัดมาแล้วมีสองคำเท่ากัน

นอกจากการตัดคำแบบเลือกให้มีคำน้อยสุดแล้ว ก็มีการใช้วิธีการทางสถิติคือใช้ n-gram ของคำมาช่วยในการแก้ปัญหาความกำกวมในการตัด (Kawtrakul et al., 1995) เช่น หากพบว่ามีข้อมูลจริง *นั่ง|ตาก|ลม|อยู่* หรือ *ทำ|ตา|กลม|แป๋ว* ก็จะอาศัยบริบทเหล่านี้มาช่วยตัดสินว่าจะเลือกตัดแบบไหนดีกว่า ซึ่งจะทำเช่นนี้ได้ ก็จะต้องมีการตัดคำด้วยมือก่อนให้มีจำนวนตัวอย่างของ n-gram ที่มากพอ และเมื่อมีคลังข้อมูลที่ตัดคำเป็นตัวอย่างแล้ว ก็เริ่มมีการใช้ข้อมูลอื่นนอกเหนือจากสถิติจาก n-gram เช่น Meknavin et al. (1997) ใช้ feature-based ตัดคำโดยเรียนรู้จากลักษณ์อื่น ๆ ในบริบทเพื่อเลือกการตัดคำที่ดีสุดจากทุกรูปแบบที่ได้จากการใช้ maximum matching มาก่อน

นอกจากการตัดคำแบบอาศัยพจนานุกรม บางคนก็ใช้วิธีการอื่นเพื่อเลี่ยงปัญหา unknown word โดยไม่อิงพจนานุกรมเลย เช่น ใช้ TCC (Thai character cluster) เป็นตัวแยกหน่วยย่อยก่อนจะตัดสินใจว่าขอบเขตคำอยู่ที่ไหน TCC อาศัยแนวคิดว่ามีตัวอักษรบางตัวที่รู้ว่าไม่ควรไปตัดคำ ณ ตำแหน่งนั้น เช่น Cา จะไม่ตัดหน้าสระอา Cา จึงเป็น cluster หนึ่ง ส่วนการตัดสินว่า TCC ไหนรวมเป็นคำได้หรือจะตัดคำที่ TCC ไหน ก็อาจใช้วิธีเรียนรู้จากคลังข้อมูลที่ตัดคำไว้ให้ เช่น ใช้ decision tree (Theeramunkong & Usanavasin, 2001)

การแยกหน่วยย่อยก่อนการตัดคำ นอกจากการใช้ TCC ก็มีการใช้รูปพยางค์เป็นหน่วยย่อยด้วย เช่น Aroonmanakun (2002) เลือกตัดพยางค์ก่อนด้วยกฎโครงสร้างพยางค์ ซึ่งแม้แต่การตัดพยางค์ก็ยังมีความเป็นไปได้หลายแบบ จึงต้องใช้ n-gram ของคลังข้อมูลที่ตัดพยางค์แล้วช่วยเลือกการตัดพยางค์ที่ดีสุดก่อนจะไปตัดสินใจรวบพยางค์เป็นคำในภายหลัง การใช้คลังข้อมูลที่ตัดพยางค์ด้วยมือมีข้อดีกว่าการใช้คลังข้อมูลที่ตัดคำด้วยมือ เพราะการตัดพยางค์ไม่มีปัญหาเรื่องการตัดแล้วไม่สม่ำเสมอ และจำนวนข้อมูลภาษาที่เท่ากันก็จะได้จำนวน token ของพยางค์ที่มากกว่าจำนวน token ของคำ

งานตัดคำช่วงต่อมาได้อาศัยคลังข้อมูลภาษาที่มีการตัดคำแล้วและใช้การเรียนรู้ด้วยเครื่องแบบต่าง ๆ เช่น ใช้ CRF (conditional random field), SVM (support vecyor machine) ล่าสุดก็นิยมใช้ deep learning เช่น KutCum, deepcut, Attacut แต่ทั้งหมดก็ต้องอาศัยคลังข้อมูลที่มีการตัดคำเป็นตัวอย่างให้เรียนก่อน วิธีการเหล่านี้จึงไม่จำเป็นต้องใช้พจนานุกรมได้ เพราะสามารถเริ่มจากหน่วยเล็กสุดคือ ตัวอักษร หรือกลุ่มตัวอักษรก็ได้ หรือจะใช้รูปพยางค์ก็เป็นไปได้

ปัญหาของการใช้คลังข้อมูลที่มีการตัดคำ นอกจากต้องลงแรงในการตัดคำเองจำนวนมากแล้ว ยังมีปัญหาจากการที่แต่ละคนตัดคำไม่เหมือนกัน หรือแม้คน ๆ เดียวกัน ก็อาจตัดคำออกมาไม่เสมอกัน ทำให้ข้อมูลไม่คงที่แบบที่คาดหวัง ปัญหาตัดคำ

ไม่เสมอกันนี้ ทำให้ต้องมาตอบคำถามว่าคำคืออะไร ทำไมแต่ละคนจึงกำหนดขอบเขตคำไม่เหมือนกัน และการตัดคำทำไปเพื่ออะไร ทำอย่างไรจึงจะสร้างคลังข้อมูลขนาดใหญ่ที่มีการตัดคำที่สม่ำเสมอได้

คำคืออะไร

คำถามพื้นฐานของการตัดคำภาษาไทย คือ อะไรคือคำ กำหนดขอบเขตอย่างไร เรามักคิดว่าปัญหาเรื่องตัดคำทำให้ประมวลผลภาษาไทยยาก ไม่เหมือนภาษาอังกฤษที่เขียนแยกเป็นคำ ๆ ให้แล้ว แต่ความจริง ภาษาอังกฤษก็ยังมีเรื่องของการหาคำที่ประกอบสร้างขึ้นจากคำย่อย ๆ เช่น *ice cream, web address, distance learning, on show, by and larage, kick the bucket* คำจึงเป็นหน่วยที่แม้ในภาษาที่มีการเขียนแบบเว้นวรรคคำก็ไม่ใช่ว่าจะปราศจากปัญหา ตำราไวยากรณ์ไทยมักพูดถึงคำว่ามีแยกเป็นคำมูล คำประสม คำซ้ำ คำซ้อน โดยให้คำมูลเป็นคำที่มีพยางค์เดียวหรือหลายพยางค์ก็ได้แต่เมื่อแยกส่วนแล้วจะไม่มีความหมายหรือมีความหมายก็ไม่มีเค้าความหมายของคำนั้น เช่น *“สามี”* แม้แยกมาแล้ว *“สา”* จะมีความหมาย *“มี”* จะมีความหมาย แต่ความหมายเหล่านั้นไม่มีเค้าสัมพันธ์กับคำ*“สามี”* แต่ในกลุ่มคำที่สร้างจากคำมูล เช่น *“แม่น้ำ”* แยกมาแล้วเป็นสองคำ *“แม่”* และ *“น้ำ”* ยังเห็นเค้าความเกี่ยวข้องกับคำนั้นได้

แม้คำอธิบายเรื่องคำจะฟังดูเข้าใจได้ไม่ยาก แต่การระบุขอบเขตคำในข้อมูลจริงกลับเป็นเรื่องยากที่คนอาจเห็นต่างกันได้โดยเฉพาะกรณีคำประสมที่มาจากการประสมคำหลายคำหลายระดับ ทำให้เกิดความสับสนระหว่างการเป็นคำประสมหรือวลี (ประเด็นนี้ผู้เขียนจะกล่าวถึงโดยละเอียดอีกครั้งในบทที่ว่าด้วยเรื่องคำประสมภาษาไทย)

การสร้างคำ

การสร้างคำในภาษานั้นทำได้หลากหลายวิธีแล้วแต่ภาษา หลักพื้นฐานคำมาจากการนำหน่วยคำ (morpheme) มา process เป็นคำ (word) หน่วยคำคือหน่วยที่เล็กสุดในภาษาที่มีความหมาย เช่น *un-* เป็น prefix เติมหน้าหน่วยคำอิสระ *happy* เป็นคำ *unhappy* กระบวนการสร้างคำด้วยการเติม affix นี้อาจเป็นกระบวนการที่เรียกว่า inflection คือสร้างรูปคำต่าง ๆ จากหน่วยศัพท์เดียวกัน เช่น สร้าง *lives, living* จาก *live* หรือเป็นกระบวนการ derivation คือการสร้างหน่วยศัพท์ใหม่จากเดิม เช่น สร้าง *reader, readable* จาก *read* ทั้งสอง process นี้พบมากในภาษาที่เป็น inflectional language ภาษาไทยไม่ได้มีกระบวนการนี้ชัดเจน อาจมองว่ามีบางคำ เช่น *การ ความ ผู้ นัก น่า* ที่ทำหน้าที่เป็นเหมือน prefix ได้

การซ้ำคำหรือ reduplication ก็เป็นอีกกระบวนการหนึ่งในการสร้างคำ ในภาษาไทยมีการสร้างคำลักษณะนี้มาก ถ้าเป็นการซ้ำรูปเราเรียกคำซ้ำ เช่น *เก่า ๆ, แดง แดง* แต่ถ้าเป็นการซ้ำความหมายเราจะเรียกคำซ้อน เช่น *ชั่วร้าย, โหดเหี้ยม* บางครั้งก็ซ้ำบางส่วน เช่น *หนักอกหนักใจ, ออกดอกออกผล, กระดาษเงินกระดาษทอง* คำซ้ำนี้บางตำราก็จัดว่าเป็นประเภทย่อยหนึ่งของการประสมคำ

การประสมคำ

กระบวนการสร้างคำที่พบมากอีกอย่างในภาษาไทยคือ การประสมคำ หรือ compounding คือการนำคำอิสระตั้งแต่สองคำขึ้นไปมาประสมกันแล้วเกิดเป็นคำที่มีความหมายใหม่ที่ต่างจากเดิมหรืออาจมีเค้าความเดิมบ้าง เช่น *แม่น้ำ ปากกา ดินปืน กระจกเงา ไม้กวาด* ในภาษาอังกฤษก็มีการสร้างคำจากการประสมคำ แต่ส่วนใหญ่จะเขียนติดกันเป็นรูปคำเดียว เช่น *firewood, grandfather, boyfriend,* แต่บางคำก็ยังเห็นรอยต่อจากการเติม hyphen เช่น *dry-clean, washer-dryer* บางคำก็ยังเขียนแยกเป็นสองรูป เช่น *back seat* แต่ถ้าเทียบกับภาษาไทย ซึ่งอาศัยการประสมคำเป็น

กลไกหลักในการสร้างคำ คำประสมในภาษาไทยจึงมีใช้มากและซับซ้อนกว่า และเป็นสาเหตุหลักที่ทำให้การตัดคำด้วยคนนั้นมีความไม่คงที่หรือเห็นไม่ตรงกันได้

หนังสือไวยากรณ์ไทยมักยกตัวอย่างคำประสมที่เมื่อรวมกันแล้วมีความหมายใหม่เลย เช่น *ลูกเสือ พ่อตา* หรือประสมกันแล้วมีความหมายใหม่แต่ยังเห็นเค้าเดิมบ้าง เช่น *น้ำแข็ง แม่บ้าน* หรือบางคำก็เหมือนจะมีความหมายของคำเดิมทั้งหมด เช่น *น้ำหวาน รถบรรทุก* ปัญหาที่พบมักเกิดจากการที่คำประสมสามารถประสมกันหลายชั้นได้ เช่น *คนขับรถ* เป็นคำประสม *รถบรรทุก* เป็นคำประสม *คนขับรถบรรทุก* มาจากการประสมที่เอาคำประสม *รถบรรทุก* มาประสมซ้อน *คนขับรถ* อีกครั้ง

เมื่อจำนวนคำเริ่มมากขึ้น ก็จะเริ่มลังเลว่าควรวิเคราะห์ให้เป็นคำหรือวลีดีกว่า เช่น *คนขับรถแท็กซี่ คนขับรถเมล์* ในเชิงความหมายก็มีลักษณะเดียวกับ *คนขับรถ* เพราะมีความหมายใหม่ที่เป็นอาชีพหนึ่ง จึงน่าจะให้เป็นคำประสมได้ แต่เมื่อประสมยาวขึ้น เป็น *คนขับรถบรรทุกสิบล้อ คนขับรถโดยสารประจำทางปรับอากาศ* ก็จะเริ่มไม่แน่ใจแล้วว่าควรตัดเป็นคำเดียวหรือไม่ คำควรจะยาวได้มากเพียงใด

ปัญหาลักษณะนี้ทำให้การตัดคำด้วยมืออาจเกิดปัญหา เพราะแต่ละคนอาจมองขอบเขตคำต่างกันไป บางคนก็ยอมให้คำประสมยาวมากได้ บางคนก็คิดว่าไม่ควรจะยาวมากไป การสร้าง BEST corpus (Booriboon et al., 2009) ที่เป็นข้อมูลตัดคำไทยด้วยมือ ผู้สร้างจึงยึดเกณฑ์ให้ตัดคำให้สั้นเป็นหลัก อะไรที่ตัดแล้วไม่เสียความหมายไปก็ให้ตัดไว้ก่อน วิธีนี้จะช่วยให้การทำงานของคนหลายคนมีความคงที่ได้มากกว่า ด้วยหลักการนี้คำว่า *รถโดยสาร* ก็จะตัด *รถ|โดยสาร คนขับรถ* ก็จะเป็น *คน|ขับ|รถ* เมื่อยึดหลักนี้ คำประสมที่ไม่ได้มีความหมายต่างจากความหมายรวมของคำย่อยก็จะถูกตัดเป็นหลายคำ *ไม้ถูพื้น โต๊ะกินข้าว* เหล่านี้ก็ควรตัดเป็นสามคำไปด้วย เพราะสามารถแยกคำได้แบบเดียวกันและความหมายไม่ได้เปลี่ยนไปมาก ยังคงเป็นไม้สำหรับถูพื้นเป็นโต๊ะสำหรับกินข้าว

ความคิดให้ตัดคำเป็นหน่วยที่เล็กสุดที่ไม่เสียความหมายไปก่อนได้นำเสนอใน Aroonmanakun (2007) เช่นกัน โดยมองเหมือนเป็นกระบวนการสร้างคำย่อยให้ได้ก่อน หลังจากนั้นจึงให้มีกระบวนการหา multi-word unit หรือ lexeme ภายหลังในลักษณะเดียวกับที่ภาษาอังกฤษเองก็ยังต้องมีการหา multi-word expression ข้อดีของวิธีนี้คือช่วยให้การตัดคำด้วยมือมีความสม่ำเสมอมากกว่า และการตัดคำย่อยก็ไม่ได้มีผลกระทบนักกับงานประมวลผลภาษาอย่าง information retrieval, text classification เพราะคำประสมแม้จะถูกตัดย่อย ก็ยังคงอยู่ในเอกสารนั้นและอยู่ติดกันเสมอ แต่บางงานที่ต้องการใช้ lexeme เช่น machine translation, information extraction, topic modeling ก็อาจต้องมีกระบวนการหา multi-word expression เพื่อรวมคำย่อย ๆ ของคำประสมเป็นหน่วยใหญ่ขึ้นมาก่อนจะใช้

ควรตัดคำย่อยเสมอหรือไม่

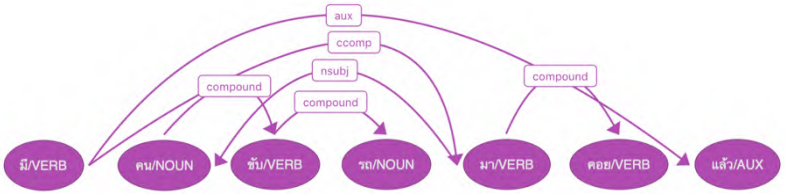
คำตอบขึ้นกับหลายปัจจัย ถ้าหากต้องการสร้างคลังข้อมูลตัดคำด้วยมือในเวลาสั้น ๆ โดยใช้คนจำนวนมากมาช่วยตัดคำ วิธีตัดคำย่อยนี้ช่วยให้คลังข้อมูลที่ได้มีความสม่ำเสมอมากกว่า และถ้างานที่ต้องการนำข้อมูลไปใช้ไม่มีความจำเป็นต้องระบุคำใหญ่ก็สามารถใช้คลังข้อมูลที่สร้างด้วยวิธีนี้ได้เลย ข้อดีอีกประการคือ จะไม่มีปัญหากับการประมวลผลคำประสมที่ยังไม่รู้จักหรือถูกระบุมาก่อน เพราะการระบุหาคำประสมที่ไม่ใช่คำประสมแท้จะไม่ถูกประมวลผลในขั้นตอนนี้

คำถามสำคัญอีกคำถาม คือ หากตัดคำย่อยแล้ว เราจะแยกความแตกต่างของคำประสมกับวลีหรือประโยคอย่างไร เพราะการตัดสินว่ารูปคำที่เห็นเป็นคำประสม เป็นวลี หรือเป็นประโยค เป็นเรื่องที่ต้องอาศัยบริบทในการบอก เช่น *คนขับรถ* อาจเป็นคำประสม หรือเป็นประโยค ก็ได้

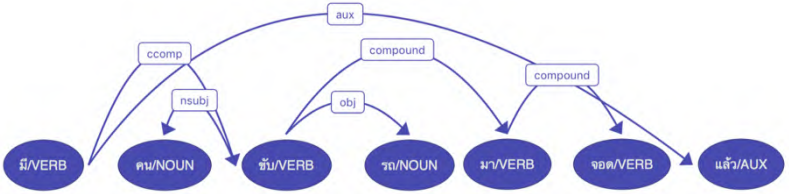
มีคนขับรถมาคอยแล้ว คนขับรถ เป็นคำประสมหมายถึง คนคนหนึ่งที่ทำมีอาชีพขับรถ

มีคนขับรถมาจอดแล้ว คนขับรถ เป็นประโยคซ้อนภายใน หมายถึง คนคนหนึ่งขับรถมา

กรณีเช่นนี้ หากเตรียมข้อมูลโดยตัดคำย่อย ข้อมูลที่สร้างจะปรากฏเป็นคำสามคำเหมือนกัน คือ *คน|ขับ|รถ* วิธีการหนึ่งที่จะช่วยแยกความต่างนี้คือการวิเคราะห์ข้อมูลลงลึกต่อ เช่น ถ้าใช้กรอบการวิเคราะห์ Universal Dependencies วิเคราะห์ประโยคสองอันนี้มีโครงสร้างและความสัมพันธ์ระหว่างคำที่ต่างกัน ดังนี้



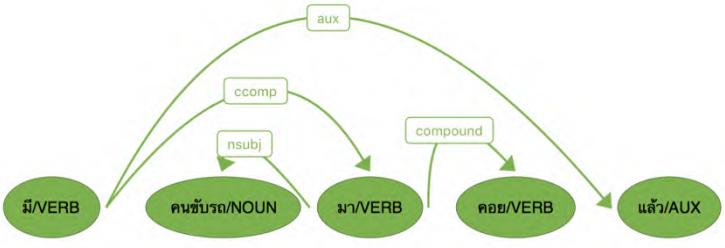
รูปที่ 1: การวิเคราะห์ที่ คนขับรถ เป็นคำประสมคำเดียว



รูปที่ 2: การวิเคราะห์ที่ คนขับรถ เป็นคำสามคำ

การวิเคราะห์แบบนี้เป็นการผลักภาระงานไปในระดับต่อไป ทำให้การตัดคำเป็นงานที่ง่ายขึ้น โอกาสผิดก็น้อยลง เพราะไม่ต้องมาตัดสินคำประสมเหล่านี้แล้วว่าจะเป็นคำเดียวหรือหลายคำเพราะได้ตัดเป็นหลายคำไปแล้ว แต่ภาระก็จะไปอยู่กับกระบวนการต่อไป เช่น ให้ parser แยกความสัมพันธ์แบบที่แสดงความต่างตามรูปออกมา หากทำแบบนี้ งาน parser จะซับซ้อนขึ้น เพราะนอกจากจะต้องวิเคราะห์ทาง syntax ที่หาโครงสร้างประโยคว่าคำต่างๆ มาประกอบกันอย่างไร ยังต้องวิเคราะห์ส่วน morphology คือหาโครงสร้างคำประสมเพิ่มมาด้วย ซึ่งทำให้งานยากขึ้นเพราะคำประสมชนิดหนึ่งที่เรียกกว่า synthetic compound จะมีลำดับคำเหมือนประโยคได้ คือ noun-verb-noun *ไม้-ถู-พื้น* หรือ noun-verb *คน-ใช้*

ถ้าในแผนงานมีความคิดที่จะพัฒนาโปรแกรม parser ลักษณะนี้ต่อ ก็สามารถใช้คลังข้อมูลที่ตัดคำย่อยแบบนี้ได้ โดยมองว่าความต่างของการวิเคราะห์จะถูกผลักภาระไปอยู่ที่ในระดับการหา dependency parserd tree แต่ก็ต้องยอมรับว่างานจะซับซ้อนขึ้น แต่หากไม่แน่ใจว่ามีเวลาและมีนักภาษาศาสตร์ที่จะช่วยทำงานส่วนนี้ได้ และต้องการใช้คลังข้อมูลตัดคำเพื่องานต่าง ๆ เลย ในกรณีนี้ก็อาจจะต้องตัดคำให้ยาวขึ้นบ้าง มองเหมือนว่าการระบุคำประสมเป็นงานลักษณะเดียวกับการระบุ named entity ที่ต้องมีการบอกขอบเขตว่าเริ่มและจบที่ไหน กรณีคำประสม การไม่ตัดคำย่อยเกินไปก็เหมือนเป็นการระบุขอบเขตคำประสมให้กับเครื่อง การตัดคำจึงต้องรวมเอาการคำประสมเหล่านี้ไว้เป็นหนึ่งคำด้วย ซึ่งความจริงก็เป็นแนวคิดดั้งเดิมของการตัดคำที่ต้องการได้คำที่เป็นหน่วยศัพท์หรือ lexeme



รูปที่ 3: การวิเคราะห์โครงสร้าง Dependency Tree จากข้อมูลที่ คนขับรถ เป็นหนึ่งคำ

นอกจากนี้ การตัดคำที่แตกต่างกัน ก็ส่งผลต่อการนำข้อมูลไปใช้งานต่อด้วย เช่น หากต้องการสร้าง word2vec จากคลังข้อมูลที่ตัดเป็นคำย่อย เช่น ตัด *กลางคืน* เป็นสองคำ เมื่อแปลงเป็น word2vec จะไม่มีเวกเตอร์ของ *กลางคืน* แต่มีของคำ *กลาง* กับ *คืน* ดังนี้ (ข้อมูลจากการใช้ BEST corpus)

most_similar("กลาง") => ['เหนือ', 'เที่ยง', 'อีสาน', 'ล่าง', 'บ่าย', 'ใหญ่', 'สี่', 'สาม', 'นี้', 'แต่ละ']

most_similar("คืน") => [('ดึก', 'วัน', 'วาน', 'เช้า', 'เดือน', 'บ่าย', 'มื้อ', 'อาทิตย์', 'พรุ่ง', 'ค่ำ']

ผู้ที่ไม่ทราบมาก่อนว่าข้อมูลที่ใช้มีการตัดคำแบบตัดย่อยนี้ ก็อาจสงสัยว่า ทำไม *กลาง* จึงคล้ายกับ *เที่ยง* หรือ *บ่าย* ได้ ผลที่ออกมาแบบนี้ เพราะ *กลาง* ในคลังข้อมูลนี้

มีทั้ง *กลาง* ที่สัมพันธ์กับทิศ ตำแหน่ง ขนาด และ *กลาง* ที่เป็นคำเกิดหน้า *วัน* กับ *คืน* จึงสัมพันธ์กับเวลา *เที่ยง บ่าย*

แต่หากในคลังข้อมูลไม่ได้ตัด *กลางคืน* เป็นสองคำ ก็จะได้เวกเตอร์ของทั้งสามคำ ซึ่งแสดงชุดคำที่สัมพันธ์กันต่างออกไป ดังนี้ (ข้อมูลจากการใช้ Thai National Corpus)

most_similar("กลางคืน") => ['กลางวัน', 'หัวค่ำ', 'บ่าย', 'ดึก', 'เช้ามืด', 'เช้าตรู่', 'ค่ำ', 'ตอนเช้า', 'เที่ยง', 'ตอนเย็น']

most_similar("กลาง") => ['ใต้', 'ล่าง', 'ปลาย', 'บน', 'ตรงกลาง', 'ใหญ่', 'ใกล้', 'กึ่งกลาง', 'ริม', 'นอก']

most_similar("คืน") => ['วัน', 'เช้า', 'ปี', 'คราว', 'เดือน', 'งวด', 'กลับบ้าน', 'สัปดาห์', 'คืนชีพ', 'ต้นเงิน'][2]

การนำคลังข้อมูลตัดคำไปสร้าง word2vec ใช้งานต่อ จึงควรเข้าใจว่าข้อมูลที่ใช้นั้นมีลักษณะแบบไหน และเวกเตอร์ที่ได้จะเป็นแบบใด หรือในทางตรงข้าม ความต้องการที่จะใช้ word2vec ที่แสดงความสัมพันธ์ของคำแบบไหน ก็จะเป็นปัจจัยให้เลือกว่าควรตัดคำแบบใดก่อนจะสร้างข้อมูล word vector

ทำอย่างไรจึงจะตัดคำประสมได้คงที่

หากเลือกแนวทางตัดคำใหญ่ขึ้น คำถามเดิมก็จะกลับมาว่าทำอย่างไรจึงจะให้มีการตัดคำด้วยมือแบบสม่ำเสมอได้ ข้อเสนอคือให้เดินสายกลาง ปัญหาความไม่สม่ำเสมอมักจะมาจากการตัดคำที่ยาวเกินไป ดังนั้น ถ้าระบุขอบเขตว่า คำประสมที่ตัดนั้นไม่ควรยาวมากไป คำประสมโดยทั่วไปมักประกอบด้วยคำ 2–3 คำ ยกเว้นคำซ้ำคำซ้อนที่อาจยาวกว่านั้นได้ คำประสมทั่วไป เช่น *ไม้ถูพื้น โต๊ะกินข้าว อาหารเย็น สมุดพก*

[2] ตัวอย่างข้อมูล word2vec มาจากการทดลองใช้คลังข้อมูล BEST และ TNC

ฯลฯ คำเหล่านี้โอกาสที่จะพบใช้แบบไม่ใช่คำประสมนั้นแม้จะนึกตัวอย่างได้ เช่น *เขานั่งโต๊ะกินข้าวกับพ่อแม่ โต๊ะกินข้าว* จะไม่ใช่คำประสมแล้ว แต่โอกาสแบบนี้ไม่น่าจะเกิดขึ้นบ่อยนัก เพราะผู้พูดมักจะเลี่ยงพูดไปแบบอื่นหรือไม่ เช่น *เขานั่งกินข้าวกับพ่อแม่* เสียมากกว่า ดังนั้น การไม่รวบคำประสมไว้ในข้อมูลที่เห็น ก็เป็นการปล่อยโอกาสที่จะกำหนดขอบเขตคำประสมทั่วไปที่พบใช้มากในภาษาไทย แล้วต้องมาแก้ปัญหาภายหลัง เช่น อาจต้องมาระบุคำประสมแบบเดียวกับที่ระบุชื่อเฉพาะ

ส่วนการคุมการทำข้อมูลให้ตรงกัน ควรสร้าง online compound glossary ให้คนทำข้อมูลเห็นร่วมกัน เพื่อให้มีการตัดคำได้ตรงกัน อาจใช้ webboard เพื่อเปิดประเด็นคำถามและแลกเปลี่ยนความเห็นจนได้ข้อสรุป วิธีการเหล่านี้ก็จะช่วยให้การตัดคำประสมมีความสม่ำเสมอได้ เหมือนคนที่ทำงานแปลที่ต้องแบ่งข้อมูลช่วยกันแปล ก็จะใช้เทคโนโลยีของ machine aided translation ที่ช่วยให้สามารถแชร์ glossary ร่วมกัน มีโปรแกรมช่วยตรวจความสม่ำเสมอของการแปลจากผู้แปลหลาย ๆ คนได้

ประเมินผลอย่างไร

การประเมินผลการตัดคำแบบที่นิยมทำกันคือ การประเมินโดยเทียบกับคำตอบหรือเฉลยที่เตรียมไว้ ในรายละเอียดก็อาจมีวิธีการวัดที่แตกต่างกัน เช่น วัดจากระดับอักขระคือดูว่าแต่ละตัวอักษรมีคำตอบที่เป็นไปได้สองอย่าง คือ ตัดหรือไม่ตัด Y/N ก็เทียบผลที่ตัดออกมาได้กับเฉลยที่มีว่าแตกต่างกันไหม ค่าที่ได้คือค่า accuracy ซึ่งโดยทั่วไปจะสูง เพราะคิดรวมตัวอักษรที่ไม่ใช่ word boundary เข้าไปด้วย กรณีสมมติที่โปรแกรมไม่ตัดคำที่ใดเลย คือให้เป็น N หมดทุกตัวอักษร

accuracy = (no.char - no.word) / no.char

ถ้าสมมติว่าความยาวของคำโดยเฉลี่ยคือ 10 ตัวอักษร accuracy จะอยู่ที่ 90% โดยไม่ต้องทำอะไรเลย ตัวอย่างเช่น ในตัวอย่างข้างล่าง หากต้องการผลตัดคำแบบนี้

สารกึ่งตัวนำ|ที่|มี|คุณสมบัติ|ทาง|ไฟฟ้า|อยู่|ระหว่าง|ตัวนำ|ไฟฟ้า| |และ|ฉนวน|ไฟฟ้า| |จึง|เป็น|สาร|ที่|เรา|สามารถ|ควบคุม|คุณสมบัติ|นำ|ไฟฟ้า|ของ|มัน|ได้|