JIST Journal of Information Science and Technology
Volume 14, NO 1 | JAN – JUN 2023 | 9-21
ISSN: 2651-1053 (Online)
ISSN: 1906-9553 (Print)

ขั้นตอนวิธีการบีบอัดข้อความภาษาไทยโดยอาศัยรูปแบบการสร้างคำ

Thai Text Compression Algorithm Employing Word-Formation Creation

*ประหยัด เลวัน และ เชาวลิต ขันคำ**

*Prayat Le-wan and Chouvalit Khancome**

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยรามคำแหง

Department of Computer Science, Faculty of Science, Ramkhamhaeng University

Received: December 8, 2023; Revised: December 31, 2023; Accepted: June 28, 2024; Published: June 30, 2024

ABSTRACT – The compression of text without data loss is a fundamental aspect of computer science, crucial for minimizing the storage space required for large datasets. This principle has been continuously developed and has consistently attracted the interest of researchers. This research article presents a highly efficient design for a new text compression method specifically tailored for compressing Thai language text. The procedural mechanism involves the creation of a new dictionary-like structure termed the "Pre-Processing Section" based on the patterns of word formation in the Thai language. This structure is utilized for referencing terms during compression and decompression processes. The data compression is executed by storing information in a binary file using the newly developed Word-Formation Thai Text Compression Algorithm (WFTTCA). The compression process following this newly developed method can achieve compression rates in theoretical terms, represented by ASCII- TIS620 encoding, ranging from 37.50% to 79.17%, with a maximum average of 63.75%. For Unicode encoding, compression rates range from 68.75% to 89.58%, with a maximum average of 81.88%. In the case of UTF-8 encoding, compression rates range from 79.17% to 93.06%, with a maximum average of 87.92%. These compression rates correspond to a range of 3.51 to 10.50 times the original data size. The experimental results from the development of the program based on the new method, using actual Thai language data randomly sampled from 1Kb-100Kb and imported from news websites, reveal that the program is capable of compressing data encoded with ASCII-TIS620 by percentages ranging from 78.09% to 84.55%. For Unicode encoding, the compression rates range from 81.50% to 86.62%. Similarly, for UTF-8 encoding, the compression rates range from 88.09% to 91.11%. When comparing the compression efficiency achieved with popular current compression software, it is found that the program developed from the new method can achieve significantly higher compression rates, both in terms of percentage compression and compression ratios.

KEYWORDS: Text Compression, Bit-Level Compression, Dictionary Base Compression, Thai Vowel Patterns, Thai Word Formation

บทคัดย่อ -- การบีบอัดข้อความแบบไม่มีการสูญเสียข้อมูลเป็นหลักทางวิทยาการคอมพิวเตอร์ที่จำเป็นอย่างยิ่งต่อการลดขนาดพื้นที่จัดเก็บข้อมูลขนาดใหญ่ให้เหลือน้อยที่สุด หลักการนี้ถูกพัฒนาต่อเนื่องมายาวนาน ทั้งยังอยู่ในความสนใจของนักวิจัยเสมอมา บทความวิจัยนี้นำเสนอการออกแบบขั้นตอนวิธีบีบอัดข้อมูลใหม่ที่มีประสิทธิภาพสูง สำหรับใช้บีบอัดข้อความภาษาไทย กลไกขั้นตอนวิธีคือการสร้างแบบพจนานุกรมแบบใหม่เรียกว่าส่วนเตรียมการประมวลผล (Pre-Processing) โดยอาศัยรูปแบบของสร้างคำศัพท์ของภาษาไทย เพื่อใช้การอ้างอิงคำศัพท์ระหว่างการบีบอัดและคลายบีบอัดข้อมูล ดำเนินการบีบอัดจัดเก็บข้อมูลในแฟ้มแบบบิตด้วยขั้นตอนวิธีที่พัฒนาขึ้น (Word-Formation Thai Text Compression

*Corresponding Author: chouvalit.k@rumail.ru.ac.th

Algorithm (WFTTCA)) กระบวนการบีบอัดตามขั้นตอนวิธีที่พัฒนาขึ้นใหม่นี้ สามารถบีบอัดข้อมูลในเชิงทฤษฎีที่แทนด้วยรหัสแอสกี-ทีไอเอส 620 ได้ถึงร้อยละ 37.50-79.17 ที่ค่าเฉลี่ยมากสุดถึง 63.75 รหัสยูนิโค้ด ร้อยละ 68.75-89.58 ที่ค่าเฉลี่ยมากสุด 81.88 และรหัสยูทีเอฟ-8 ร้อยละ 79.17-93.06 ที่ค่าเฉลี่ยมากสุด 87.92 ด้วยอัตราการบีบอัดอยู่ระหว่าง 3.51-10.5 เท่าของข้อมูลต้นฉบับ ผลทดลองจากการพัฒนาโปรแกรมตามขั้นตอนวิธีที่ออกแบบใหม่ ใช้บีบอัดข้อมูลภาษาไทยขนาด 1 ถึง 100 กิโลไบต์ ที่มาจากการสุ่มและนำเข้าข้อมูลจริงจากข่าวในเว็บไซต์ พบว่า โปรแกรมที่นำเสนอสามารถบีบอัดข้อมูล รหัสแอสกี-ทีไอเอส 620 ได้ร้อยละ 78.09-84.55 รหัสยูนิโค้ดบีบอัดได้ร้อยละ 81.05-86.62 และรหัสยูทีเอฟ-8 บีบอัดได้ร้อยละ 88.09-91.11 และเมื่อเปรียบเทียบประสิทธิภาพการบีบอัดที่ได้กับซอฟต์แวร์บีบอัดข้อมูลที่นิยมใช้ในปัจจุบัน พบว่า โปรแกรมที่พัฒนาขึ้นจากขั้นตอนวิธีใหม่สามารถบีบอัดได้สูงมากอย่างมีนัยสำคัญทั้งร้อยละการบีบอัดและอัตราการบีบอัด

คำสำคัญ: การบีบอัดข้อความ การบีบอัดระดับบิต การบีบอัดด้วยพจนานุกรม รูปแบบสระภาษาไทย รูปแบบการสร้างคำภาษาไทย

1. บทนำ

การบีบอัดข้อความ (Text Compression) เป็นหลักการทางวิทยาการคอมพิวเตอร์ (Computer Science) สำหรับออกแบบกลไกการจัดเก็บข้อมูลขนาดใหญ่โดยใช้เนื้อที่จัดเก็บให้น้อยที่สุด แม้ว่าหลักการนี้จะมีการศึกษาต่อเนื่องมาเป็นเวลานาน แต่ในปัจจุบันก็ยังเป็นปัญหาที่ได้รับความสนใจจากนักวิจัยเสมอ โดยประเพณีแนวคิดการแก้ปัญหาจะแสวงหาขั้นตอนวิธีการบีบอัดข้อความ (Text Compression Algorithm) ที่พยายามบีบอัดข้อความต้นฉบับให้ได้มากที่สุด เพื่อใช้เนื้อที่จัดเก็บน้อยที่สุด กระบวนการหลักของแต่ละขั้นตอนวิธีดังกล่าว จะอ่านข้อมูลแล้วใช้กลวิธีในการบีบอัดก่อนเก็บไว้ในแหล่งจัดเก็บข้อมูล จากนั้นจะออกแบบส่วนการคลายข้อมูลที่บีบอัดไว้ให้กลับคืนสู่ข้อความต้นฉบับอีกครั้ง โดยไม่ยอมให้มีการสูญเสียข้อความต้นฉบับ (Lossless Text Compression) ซึ่งจะได้ข้อความเช่นเดียวกับข้อความต้นฉบับทุกประการ

พิจารณากลไกการบีบอัดของขั้นตอนวิธีต่างๆ พบว่า หลักการบีบอัดที่ได้รับความนิยมคือ การบีบอัดข้อมูลในระดับบิต (Bit Level) และการบีบอัดโดยอาศัยพจนานุกรม (Dictionary) ที่อาศัยพจนานุกรมเป็นสิ่งอ้างอิงสำหรับการบีบและคลายการบีบอัดข้อความ งานวิจัยที่น่าสนใจในรูปแบบนี้แสดงใน [1, 2, 3, 4] เป็นต้น เมื่อพิจารณาถึงประสิทธิภาพการบีบอัดข้อมูลแบบไม่ไม่ยอมให้มีการสูญเสีย พบว่า การบีบอัดโดยพจนานุกรมจะมีประสิทธิภาพสูงบีบอัดข้อมูลในระดับบิต แต่ในขณะเดียวกับการบีบอัดระดับบิตก็ยังเป็นสิ่งที่น่าสนใจและเป็นความท้าทายสำหรับนักวิจัยอยู่เสมอ ยิ่งเป็นงานวิจัยสร้างขั้นตอนวิธีการบีบอัดข้อมูลภาษาไทย พบว่ายังมีการออกแบบและศึกษาค่อนข้างมีน้อย พบเพียงการศึกษาเปรียบเทียบแสดงไว้ใน [5, 6, 7] เท่านั้น

จากงานวิจัยการออกแบบขั้นตอนวิธีบีบอัดภาษาไทยล่าสุด คณะผู้วิจัยนำเสนองานวิจัย [8] ได้ออกแบบขั้นตอนวิธีการบีบอัดข้อมูลภาษาไทย โดยให้ความสนใจกับการบีบอัดทั้งสองกระบวนการ คือ การบีบอัดข้อมูลในระดับบิตและการบีบอัดโดยอาศัยพจนานุกรม โดยใช้รูปแบบของการประสมคำไทยตามรูปแบบสระ (Vowel Patterns) กับอักขระ (Alphabet) ใช้จัดเก็บรูปแบบสระที่จะนำมาประสมคำนั้นไว้ในพจนานุกรมแบบพิเศษที่สร้างขึ้นใหม่ สำหรับใช้อ้างอิงการบีบอัดข้อมูลในรูปแบบพจนานุกรม จัดเก็บในเนื้อที่สำรอง (Temporary Space) โดยแทนแต่ละคำศัพท์ในระดับบิต ซึ่งทำให้อัตราการบีบอัดข้อมูลภาษาไทยทำได้อย่างมีประสิทธิภาพและใช้เนื้อที่จัดเก็บน้อยกว่าขั้นตอนวิธีเดิมที่เคยนำเสนอมาในอดีต เนื้อที่สำหรับจัดเก็บพจนานุกรมน้อยกว่าการจัดเก็บโดยใช้พจนานุกรมแบบปกติ รวมถึงยังรองรับการเป็นพลวัตของพจนานุกรมได้อีกด้วย

เมื่อพิจารณารายละเอียดในงานล่าสุดของคณะผู้วิจัย ยังมีข้อด้อยอยู่หลายประการ เช่น สามารถจัดการบีบอัดคำได้ตามรูปแบบของสระเท่านั้น ซึ่งไม่สามารถตอบสนองต่อการประสมคำแบบซับซ้อนที่ประกอบด้วย รูปแบบ พยัญชนะ สระ วรรณยุกต์ ตัวสะกด ตัวการันต์ รวมถึงผลการทดลองที่ได้นำเสนอไว้ในงานวิจัยดังกล่าวยังมิได้นำเสนอการทดลองด้วย

การเขียนโปรแกรมคอมพิวเตอร์เปรียบเทียบผลการทดลองขั้นตอนวิธีที่นำเสนอกับซอฟต์แวร์บีบอัดที่นิยมใช้งานในปัจจุบัน

ดังนั้น งานวิจัยใหม่นี้ได้ออกแบบขั้นตอนวิธีบีบอัดด้วยพจนานุกรมและจัดเก็บในระดับบิต โดยอาศัยแนวคิดที่มีประสิทธิภาพในงานล่าสุดของคณะผู้วิจัยเป็นฐานการออกแบบเบื้องต้น โดยขั้นตอนวิธีใหม่ที่นำเสนอในบทความวิจัยนี้สามารถบีบอัดข้อมูลภาษาไทยตามรูปแบบการประสมคำไทยจนครบทุกส่วนที่งานวิจัยก่อนหน้าไม่สามารถดำเนินการได้ รวมทั้งได้พัฒนาขั้นตอนวิธีสำหรับเตรียมและคัดแยกข้อความภาษาไทย (Pre-processing) และสร้างพจนานุกรมรูปแบบคำ (Formation Dictionary (FD)) โดยสามารถอ่านข้อความจากแฟ้มหรือป้อนด้วยมือได้อีกด้วย รวมถึงได้นำเสนอผลการทดลองที่มีประสิทธิภาพทั้งทางทฤษฎีและการใช้งานจริงที่เปรียบเทียบกับซอฟต์แวร์บีบอัดข้อมูลที่นิยมใช้งานกันในปัจจุบันอีกด้วย

ลำดับการนำเสนอเนื้อหาดังนี้ หัวข้อที่ 2 แสดงงานวิจัยและทฤษฎีที่เกี่ยวข้อง หัวข้อที่ 3 คือ นิยามพื้นฐานสำหรับการออกแบบขั้นตอนวิธี หัวข้อที่ 4 แสดงแนวคิด ลำดับ ตัวอย่าง และขั้นตอนเตรียมพจนานุกรม และขั้นตอนวิธีการบีบอัดข้อความ หัวข้อที่ 5 นำเสนอขั้นตอนวิธีคลายการบีบอัดข้อความ หัวข้อที่ 6 แสดงผลการทดลองที่ใช้ข้อมูลหลากหลายรูปแบบ หัวข้อที่ 7 อภิปรายผลการทดลอง และส่วนที่ 8 เป็นสรุปผลการวิจัยและข้อเสนอแนะ

2. งานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 ขั้นตอนวิธีการบีบอัดข้อความ

การบีบอัดข้อความ (Text Compression) คือ หลักการพื้นฐานที่สำคัญยิ่งในวิทยาการคอมพิวเตอร์ (Computer Science) นำไปแก้ปัญหาจัดเก็บข้อมูลขนาดใหญ่โดยใช้เนื้อที่จัดเก็บให้น้อยที่สุด ประเภทของการบีบอัดข้อความตามที่ระบุไว้ใน [9] สามารถแบ่งออกเป็น 1) การบีบอัดแบบสูญเสียข้อความต้นฉบับ (Lossy Text Compression) ที่อนุญาตให้สูญเสียเมื่อมีการบีบอัดและคลายการบีบอัด ข้อความต้นฉบับอาจมีการสูญเสียบางส่วนได้ และ 2) การบีบอัดแบบไม่สูญเสียข้อความต้นฉบับ (Lossless Text Compression) การบีบอัดแบบนี้จะไม่อนุญาตให้มีการสูญเสียข้อความต้นฉบับไป จะได้ข้อความต้นฉบับดั้งเดิมเมื่อคลายการบีบอัด ขั้นตอนวิธีบีบอัดข้อมูลมาตรฐานที่มีชื่อเสียง และถือเป็นต้นแบบสำหรับศึกษาวิจัยการบีบอัดคือ Huffman, Ziv-Lempel และ Factor ซึ่งแสดงรายละเอียดอยู่ใน [10] ตัวอย่างขั้นตอนวิธีที่ใช้แนวทางมาตรฐานนี้ แสดงไว้ [11, 12, 13] เป็นต้น

ขั้นตอนวิธีบีบอัดที่รู้จักกันดี (Huffman, Ziv-Lempel และ Factor) ถูกนำมาใช้เป็นขั้นตอนวิธีหลัก ขั้นตอนวิธีบีบอัดทำให้พื้นที่จัดเก็บข้อมูลนั้นน้อยที่สุดถึงร้อยละ 0.09 และสูงสุดร้อยละ 73.74 สำหรับ 30 International Journal of Computer (IJC) (2014) Volume 15, No 1, หน้า 29-41 เป็นขั้นตอนวิธีที่ใช้คำหลักซึ่งเป็นตัวแทนของแหล่งข้อมูล โดยวิเคราะห์คำหลักในแหล่งข้อมูลและแสดงรหัสในโครงสร้างที่เหมาะสม

หลักการพื้นฐานของคำหลักคือ การบีบอัดตามพจนานุกรม คำหลักเหล่านี้ ถูกเก็บไว้ในพจนานุกรมมีการแสดงขั้นตอนวิธีมากมายดังนี้ รูปแบบการบีบอัดข้อความตามพจนานุกรมสองระดับ แสดงสองระดับเพื่อจัดเก็บข้อมูล ทำให้ข้อความต้นฉบับสามารถลดพื้นที่บันทึกได้ประมาณร้อยละ 75 อย่างไรก็ตาม ขั้นตอนวิธีแบบละเอียดใช้ gzip และ bzip สำหรับการบีบอัด ในขณะเดียวกันอัตราส่วนการบีบอัดคือ 2.01 บิตต่ออักขระซึ่งเป็นขั้นตอนวิธีการบีบอัดข้อความแบบไม่สูญเสียข้อมูลที่รวดเร็ว ขั้นตอนวิธีนี้ใช้แผนผังการค้นหาแบบไตรภาคเพื่อเร่งความเร็วการเข้ารหัสทรานส์ฟอร์ม การปรับปรุงประสิทธิภาพการบีบอัดเพิ่มขึ้นมากกว่าร้อยละ 13 เมื่อเทียบกับ bzip2-9, มากกว่าร้อยละ 19 เทียบกับ gzip-9 และมากกว่าร้อยละ 10 เทียบกับ PPMD ในรายละเอียดของขั้นตอนวิธีที่ใช้ PPM, Huffman ในพจนานุกรมข้อมูล การบีบอัดโดยใช้ข้อความที่เข้ารหัส คือ การกำหนดการเข้ารหัสหรือลายเซ็น

ขั้นตอนวิธี [14, 15, 16, 17] และ [5, 6, 7, 18] เป็นขั้นตอนวิธีระดับบิตที่เป็นทางเลือก ส่วนขั้นตอนวิธีที่ประสิทธิภาพต่ำ [5] ใช้ฟังก์ชันบูลีนเพื่อแทนชุดของกลุ่มคำในระดับบิตซึ่งถือว่าเป็นเงื่อนไขขั้นต่ำ ขั้นตอนวิธีนี้สามารถประหยัดพื้นที่ได้มากกว่าร้อยละ 10 มีการแสดงขั้นตอนวิธีที่มีประสิทธิภาพมากกว่าใน [6] ซึ่งสามารถช่วยลดพื้นที่จัดเก็บสูงสุดถึงร้อยละ 20 ขั้นตอนวิธีที่กล่าวมานี้ใช้การบีบอัดแฟ้มมัลติมีเดีย ข้อมูลขนาดใหญ่กว่านี้ สามารถเก็บถาวรได้ใน [15] ขั้นตอนวิธีที่เหนือกว่า เช่น [12] สามารถประหยัดพื้นที่ได้สูงสุดประมาณร้อยละ 80-97 ซึ่งขั้นตอนวิธีดังกล่าวใช้เทคนิคที่เรียกว่า ACW(n) (Adaptive Character Word-length) หลักการนี้ แปลงข้อมูลต้นฉบับลงในลำดับไบนารี โดยใช้รูปแบบม่านอักขระต่อไบนารีซึ่งใช้รหัสแอสกี (ASCII) จากนั้นแบ่งลำดับออกเป็น

ความยาว n บิตซึ่งใช้ $d \leq 256$ โดยที่ d คือ ขนาดของตัวอักษร หลังจากนั้น วิธีการนี้จะค้นหาตัวแปรที่เหมาะสมที่สุดของ n (n=9 และ n=10) ในขั้นตอนวิธี [15] เรียกว่า ขั้นตอนวิธีขั้นสูง ของวิธีที่ใช้เทคนิค ACW(n) โซลูชันนี้ เรียกว่า ACW(n,s) ได้เพิ่มเทคนิคใหม่โดยใช้บิตที่ตามมาซึ่งเรียกว่า ค่า s เมื่อใช้ค่า n เท่ากับ 14 พื้นที่ถูกบันทึกจะลดลงประมาณร้อยละ 80-97 แต่บางกรณีมีประสิทธิภาพต่ำ เช่น พื้นที่บันทึกลดลงประมาณร้อยละ 2 หรือร้อยละ 50% เป็นต้น

สำหรับงานวิจัยการบีบอัดข้อความในประเทศไทยที่กล่าวถึงขั้นตอนวิธีบีบอัดข้อมูลที่รู้จักกันดี เศกสิทธิ์ พจมารและจารี ทองคำ [19] ได้เปรียบเทียบขั้นตอนวิธีบีบอัดข้อมูลแบบไม่สูญเสียบนเว็บแอปพลิเคชันจำนวน 5 ตัวได้แก่ Huffman Coding, Deflate, BZip2, LZMA และ LZ4 เพื่อลดขนาดข้อมูลให้เล็กลงก่อนส่งซึ่งจะทำให้การส่งข้อมูลผ่านเว็บแอปพลิเคชันมีความเร็วเพิ่มขึ้นโดยทดสอบกับข้อมูล 3 กลุ่มคือ แฟ้มเอกสาร แฟ้มรูปภาพและแฟ้มมัลติมีเดีย ผลการทดลองพบว่าขั้นตอนวิธี LZ4 มีประสิทธิภาพเชิงเวลาสูงที่สุด ที่อัตราความเร็วเฉลี่ย 7.6865 วินาที ชนาภา ศิลาวงษ์และธนภัทร์ อนุศาสน์อมรกุล [20] เปรียบเทียบวิธีบีบอัดและคลายบีบอัดข้อมูลแบบไม่สูญเสียที่เหมาะสมสำหรับแต่ละประเภทข้อมูล จำนวน 4 วิธีได้แก่ Huffman, LZ77, LZW และ Deflate พบว่า Deflate มีประสิทธิภาพโดยรวมดีที่สุด บรรพต ดลวิทยากุล [21] ศึกษาการบีบอัดเอกสารเอชทีเอ็มแอล (HTML) บนฝั่งเซิร์ฟเวอร์ด้วยการขั้นตอนวิธีแบบฮัฟฟ์แมน (Huffman) และแชนนอน-ฟาโน (Shanon-Fano) เทียบกับขั้นตอน จีซิป (GZIP) พบว่า ฮัฟฟ์แมนมีอัตราส่วนและเวลาบีบอัดเอกสารน้อยกว่า เมื่อใช้กับเอกสารเอชทีเอ็มแอลที่ขนาดแฟ้มแตกต่างกัน

2.2 การสร้างคำภาษาไทย

รูปแบบการสร้างคำภาษาไทยอธิบายว่า คำ คือ พยางค์ที่มีความหมาย โดยเริ่มจากคำ 1 พยางค์ เช่น พ่อ แม่ พี่ น้อง ลุง ป้า น้า อา เป็นต้น โดยแบ่งการสร้างคำภาษาไทยออกดังนี้

พยางค์เปิด คือ คำที่ไม่มีตัวสะกด มี 3 ส่วน คือ พยัญชนะต้น สระ และวรรณยุกต์

พยางค์ปิด คือ คำที่มีตัวสะกด มี 4 ส่วน คือ พยัญชนะต้น สระ วรรณยุกต์ และตัวสะกด

ตัวการันต์ คือ พยางค์ที่ไม่ออกเสียงจะนับเป็นส่วนประสมพิเศษในพยางค์นั้น

ตัวอย่างพยางค์ที่มีส่วนประกอบเสียงแบบต่างๆ ใน [22] แสดงตัวอย่างดังนี้

ส่วนประสม 3 ส่วน (พยัญชนะต้น + สระ + วรรณยุกต์) เช่น งู ม้า ลา ไก่

ส่วนประสม 4 ส่วน (พยัญชนะต้น + สระ + วรรณยุกต์ + ตัวสะกด) เช่น มด ยุง หุง ข้าว

ส่วนประสม 4 ส่วนพิเศษ (พยัญชนะต้น + สระ + วรรณยุกต์ + ตัวการันต์) เช่น เล่ห์ เสน่ห์

ส่วนประสม 5 ส่วน (พยัญชนะต้น + สระ + วรรณยุกต์ + ตัวสะกด + ตัวการันต์) เช่น จันทร์ เทศน์ ทุกข์ เป็นต้น

ดังที่กล่าวไปแล้ว งานวิจัยของคณะวิจัยยังไม่สามารถจัดการบีบอัดรูปแบบภาษาไทยตามรูปแบบใน [22] ได้ครอบคลุมทุกแบบ ดังนั้น งานวิจัยนี้ จึงออกแบบขั้นตอนวิธีการบีบอัดข้อความภาษาไทยได้ครบตามรูปแบบการสร้างคำดังกล่าว ทั้งยังคงประสิทธิภาพที่ดีของการบีบอัดในงานวิจัยเดิมเอาไว้ได้อีกด้วย

3. นิยามพื้นฐานสำหรับการออกแบบขั้นตอนวิธี

เนื้อหาส่วนนี้ นำเสนอนิยาม คำสำคัญ พร้อมทั้งอธิบายตัวอย่างประกอบเพื่อความเข้าใจและใช้ออกแบบขั้นตอน วิธีการบีบอัดและการคลายบีบอัดต่อไป

นิยามที่ 1 กำหนดให้ **D** เป็นเอกสารเดี่ยวบรรจุข้อความภาษาไทยที่ประกอบคำศัพท์ภาษาไทย **w1, w2, w3,..,wn** เขียนแทนด้วย **D = { w1, w2, w3,..,wn }**

ตัวอย่างที่ **1** แสดงตัวอย่างเอกสารเดี่ยว **D** ที่บรรจุคำศัพท์ข้อความภาษาไทย หากข้อความภาษาไทยเป้าหมายที่จะบีบอัดคือ **“เก วาม ทอง โก”** ดังนั้นตามนิยามที่ **1** สามารถเขียนแทนด้วย **D={** เก, วาม, ทอง, โก **}** ซึ่งความหมายตามนิยามหมายถึง **w1=**เก, **w2=**วาม, **w3=**ทอง และ **w4=**โก

นิยามที่ 2 ถ้ามีเอกสารเดี่ยวหลายรายการ **D1, D2, D3, …, Dm** โดยที่ **D1= { w1, w2, w3,..,wn1 }, D2={ w1,w2, w3,...,wn2 }, D3 ={ w1, w2, w3,….,wn3 }, ..., Dm={w1, w2, w3,….,wnm }** เรียกว่า เอกสารหลายฉบับ (**Multi Documents**) เขียนแทนด้วย **MD**

ตัวอย่างที่ **2** แสดงตัวอย่างเอกสารเดี่ยวหลายรายการ ดังที่กล่าวถึงในนิยามที่ **2** จำนวน **5** เอกสาร โดยที่ **MD** ประกอบด้วยเอกสารเดี่ยว **D1, D2, D3, D4** และ **D5** ดังนี้

D1= { เก วาม ทอง โก } **(file 1)**

D2= { ทอง โก เอื้อ อา } **(file 2)**

D3= { อา รี้ เก วาม } **(file 3)**

D4= { เอื้อ โก นคร ทอง } **(file 4)**

D5= { วาม ทอง โก เก } **(file 5)**

D6= { มาตร ศักดิ์ การณ์ กาญจน์ } **(file 6)**

D7= { เล่ห์ เสน่ห์ จันทร์ เทศน์ ทุกข์ }**(file 7)**

ดังนั้น ในความหมายของตัวอย่างที่ **2** เอกสารหลายฉบับดังกล่าว คือ **MD** ตามนิยามที่ **2**

นิยามที่ 3 กำหนดให้ **TD (Temporary Dictionary)** คือ พจนานุกรมชั่วคราว สำหรับใช้จัดเก็บคำภาษาไทยทั้งหมดที่บรรจุใน **MD** ก่อนการวิเคราะห์รูปแบบคำ **(Word Formation)**

ตัวอย่างที่ **3** เมื่อวิเคราะห์คำที่มีใน **MD** ตัวอย่างที่ **2** จะได้ **TD** ดังรูปที่ **1** ที่มีรูปแบบคำที่ประกอบด้วยคำครบตาม พยัญชนะ+สระ+วรรณยุกต์+ตัวสะกด+การันต์

เอื้อ อา รี้ เก วาม ทอง โก นคร
มาตร ศักดิ์ การณ์ กาญจน์
เล่ห์ เสน่ห์ จันทร์ เทศน์ ทุกข์

*รูปที่ 1. แสดงตัวอย่าง **TD** จาก **MD** ในตัวอย่างที่ 2*

นิยามที่ 4 กำหนดให้รูปแบบคำภาษาไทยเป็น **F0, F1, F2, …., Fn** โดยที่แต่ละ **Fx** คือ รูปแบบของการประสมคำในภาษาไทยที่ไม่ซ้ำแบบกัน เรียกรูปแบบ ดังกล่าวนี้ว่า **FD** ซึ่งก็คือ พจนานุกรมรูปแบบคำ **(Formation Dictionary)**

ตัวอย่างที่ **4** จาก **TD** ในตัวอย่างที่ **2** สามารถวิเคราะห์และแทนรูปแบบคำแต่ละคำ **Fx** แล้วจัดเก็บใน **FD** ได้ดังรูปที่ **2**

F0:เ_ื้อ, F1:_า, F2:_ี้, F3:เ_ , F4:_าม, F5:_อง, F6:โ_, F7:_คร, F8:_าตร, F9:_ักดิ์
F10:_ารณ์, F11:_าญจน์, F12:เ_่ห์, F13:เ_น่ห์, F14:_ันทร์, F15:เ_ศน์, F16:_ุกข์

*รูปที่ 2. แสดงตัวอย่าง **FD** ของตัวอย่างที่ 2*

นิยามที่ 5 รายการอักขระ **(Alphabet Lists)** ของภาษาไทย จาก ก, ข, ฃ, ค, ...ห, อ, ฮ เขียนแทนด้วย **AL**

ตัวอย่างที่ **5** แสดงตาราง **AL** ของอักขระภาษาไทยดังรูปที่ **3**

1	2	3	4	5	...	43	44
ก	ข	ฃ	ค	ฅ	...	อ	ฮ

*รูปที่ 3. แสดงตัวอย่าง **AL** ตารางอักขระภาษาไทย*

นิยามที่ 6 เมื่อนำรูปแบบคำ **Fx** ในนิยาม **4** มาเขียนรวมกับหมายเลขของอักขระในตาราง **AL** ดังในนิยาม **5** ให้เป็นรูปแบบ **Fx+AL** แล้วเรียกรูปดังกล่าวนี้ว่า **RN (Representing Number)** การแทนคำศัพท์ด้วยชุดตัวเลข

ตัวอย่างที่ **6** พิจารณา **RN** ของคำว่า “เสน่ห์” เมื่อแปลงให้อยู่ในรูป **Fx+AL** จะได้ **F13+AL** จะได้ ซึ่ง **F13** นำมาจากพจนานุกรมรูปแบบคำ **FD** นำมารวมกับหมายเลขตัวอักขระจากตารางรายการอักขระ **AL** ทำให้ได้ผลลัพธ์ของ **RN** เพื่อจัดเตรียมสู่การบีบอัดหรือคลายบีบอัดข้อความจะได้ **RN= 13+40** สำหรับคำ ว่า “เสน่ห์” เป็นต้น

นิยามที่ 7 เมื่อมีรูปแบบ **RN** ดังนิยามที่ **6** แล้วสามารถแปลงให้อยู่ในรูปแบบบิต **(Bit Form)** ได้ เรียกรูปแบบของบิตดังกล่าวว่า **DBF (Data Bit Form)** แทนข้อความในรูปแบบบิต

ตัวอย่างที่ **7** แสดงคำในรูปแบบของ **DBF** ต่อจาก **RN** ของคำว่า “เสน่ห์” ดังรูปที่ 4

*รูปที่ 4. แสดงตัวอย่างการแปลงจาก **RN** สู่ **DBF** ดังนิยามที่ 7*

นิยามที่ 8 เรียกแฟ้มที่ถูกบีบอัด **(Compressed File)** ว่า **CF**

นิยามที่ 9 เรียกพื้นที่ชั่วคราวระหว่างบีบอัดหรือคลายการบีบอัดข้อความว่า **TM (Temporary Memory)**

นิยามที่ 10 เรียกแฟ้มข้อมูลต้นฉบับ **(Original File)** ก่อนเรียกใช้ขั้นตอนวิธีการบีบอัดและหลังการเรียกใช้ขั้นตอนคลายบีบอัดว่า **OF**

นิยามที่ 11 เรียกแฟ้มข้อมูลภาษาไทยสำหรับเรียนรู้รูปแบบคำว่า ข้อความภาษาไทยเพื่อการเรียนรู้รูปแบบการสร้างคำ แทนด้วยสัญลักษณ์ **TTL (Thai Text for Learning)**

4. ขั้นตอนวิธีการบีบอัด

4.1 ขั้นตอนวิธีการสร้างพจนานุกรมรูปแบบการสร้างคำ

เบื้องต้นก่อนการบีบอัดข้อมูลต้องสร้างพจนานุกรมรูปแบบคำตามนิยาม 4 (FD---Formation Dictionary) ซึ่งขั้นตอนนี้จะต้องดำเนินการก่อน (Pre-Processing) เพื่อใช้อ้างอิงและนำเข้าสู่ขั้นตอนวิธีการบีบอัดข้อมูลต่อไป

กำหนดให้ TTL คือ แฟ้มข้อความภาษาไทยสำหรับการเรียนรู้เพื่อจัดเก็บรูปแบบโดยมีคำไทยปรากฏอยู่ตั้งแต่ 1 ถึง q คำ กำหนดให้ Single-Text คือ ตัวแปรสำหรับการป้อนข้อมูลด้วยมือหรืออ่านจากแฟ้มข้อมูล ขั้นตอนวิธีการประมวลผลสำหรับสร้างรูปแบบเพื่อการอ้างอิงสำหรับการบีบอัด แสดงดัง Algorithm 1:Pre-Processing ต่อไปนี้

Algorithm 1: Pre-Processing

Input: Empty FD, TTL or Single-Text (Manual Input)

Output: FD

```
1. Input Thai Alphabets to AL       //นำเข้าพยัญชนะไทย
2. Input Sing Thai Vowels to FD     //นำเข้าสระภาษาไทย
3. Input Thai Tone Marks to FD      //นำเข้าวรรณยุกต์ไทย
4. If input is Single-Text  Then    //ป้อนรูปแบบด้วยมือ
5.     If Single-Text not exists in FD Then
6.         Create a new Fx and Single-Text to Fx
7.         Add Fx to FD
8.     End If
9.  Else
10.    For i=1 to q Do  //ป้อนด้วยไฟล์เรียนรู้รูปแบบ TTL
11.      Scan the format of word in TTL at the word of i
12.      If the format of word i not exists in FD Then
13.        Create a new Fx and add for the format of word i
             to Fx
14.        Add Fx to FD
15.      End If
16.   End For
17. Return FD
```

4.2 แนวคิดภาพรวมของขั้นตอนวิธีการบีบอัด

แนวคิดภาพรวมของขั้นตอนวิธี แสดงเป็นขั้นตอนดังนี้

ขั้นตอนที่ 1: อ่านแฟ้มข้อความภาษาไทย ตัดคำ วิเคราะห์คำหลักและสร้างพจนานุกรมรูปแบบคำ FD

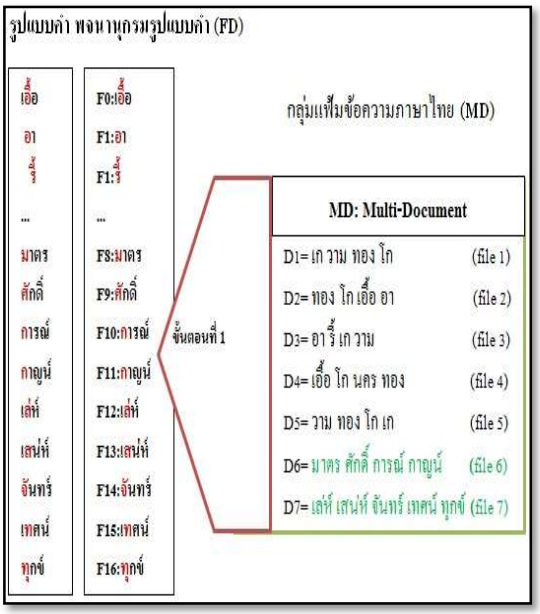
ขั้นตอนที่ 2: แทนคำศัพท์ภาษาไทยแต่ละคำตามรูปแบบสระและรูปแบบคำด้วยตัวเลขในรูปแบบ RN

ขั้นตอนที่ 3: แปลง RN ไปเป็นรูปแบบบิตแทนข้อความ DBF

ขั้นตอนที่ 4: ใช้ขั้นตอนวิธีการบีบอัดข้อความภาษาไทยเพื่อบีบอัดด้วยพจนานุกรมที่มีอยู่แล้วเขียนแทนแต่ละคำในรูปแบบบิตข้อมูลให้กับแฟ้มที่ถูกบีบอัด

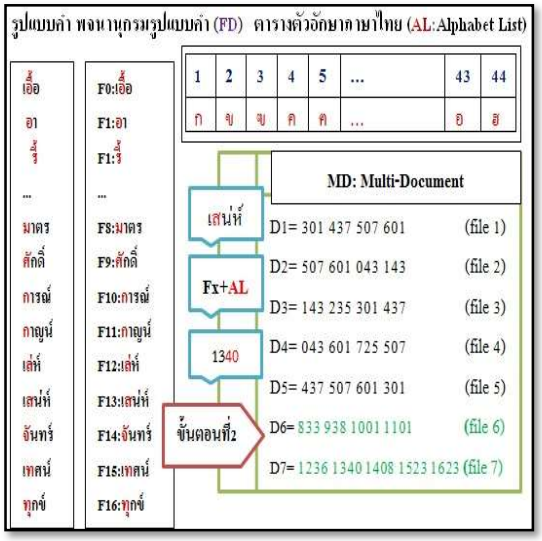
ลำดับแนวคิดของขั้นตอนวิธีบีบอัดข้อความ (Thai Text Compression Algorithm) ภาษาไทยทีละขั้น ดังภาพต่อไปนี้

ขั้นตอนที่ 1: อ่านแฟ้มข้อความภาษาไทย ตัดคำ วิเคราะห์คำหลักและสร้างพจนานุกรมรูปแบบสระและคำ ดังรูปที่ 5



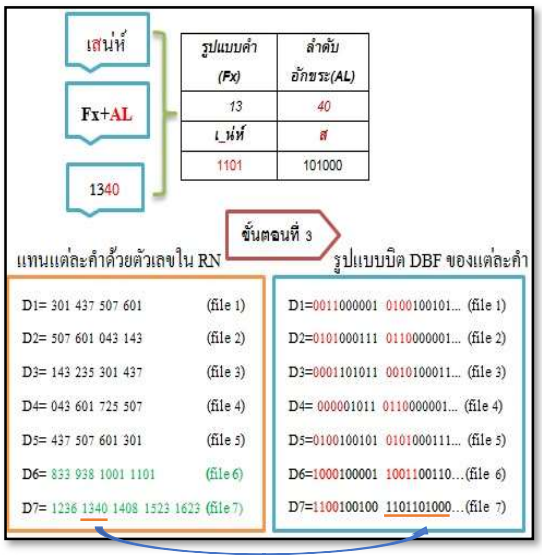
รูปที่ 5. แสดงขั้นตอนที่ 1 ตัดคำ วิเคราะห์คำหลักและสร้างพจนานุกรมรูปแบบคำ FD

ขั้นตอนที่ 2: แทนคำศัพท์ภาษาไทยแต่ละคำตามรูปแบบสระและคำด้วยตัวเลขลงใน RN เช่นรูปที่ 2 เช่น คำว่า “เสน่ห์” แปลง $F_{13} + AL_{40}$ โดยที่ F13 นำมาจากพจนานุกรมรูปแบบคำ (FD) ส่วน AL ลำดับของตัวอักษรจากตารางลำดับอักษรภาษาไทย แสดงดังตัวอย่างในรูปที่ 6



รูปที่ 6. ขั้นตอนที่ 2 แทนคำภาษาไทยตามรูปแบบคำด้วยลำดับตัวเลขแล้วเก็บลงใน RN

ขั้นตอนที่ 3: แปลง RN เป็นบิตรูปแบบ DBF เช่นรูปที่ 7



รูปที่ 7. ขั้นตอนที่ 3 แปลง RN ไปเป็นบิตรูปแบบ DBF

4.3 ขั้นตอนวิธีการบีบอัดข้อความภาษาไทย

กำหนดให้ n_m เป็นจำนวนของแฟ้มข้อความส่วน s_m เป็นจำนวนคำศัพท์ภาษาไทยทั้งหมดจากทุกแฟ้ม โดยที่ s_m ขึ้นอยู่กับขนาดของเอกสารแต่ละแฟ้ม ขั้นตอนวิธีการบีบอัดแสดงดัง Algorithm 2: Word-Formation Thai Text Compression Algorithm (WFTTCA) ต่อไปนี้

Algorithm 2: WFTTCA
Input: MD, FD, RN[n_m]
Output: CF
1. Initiate empty CF //CF: Compression File
2. For i=1 to n_m Do
3. For k=1 to s_m Do //FD: Formation Dictionary
4. If w_k not exists in FD Then
5. Add current keyword w_k to FD
6. End If
7. Add number of related w_k in FD into RN[i][s_k]
8. End For
9. End For
10. Convert the RN to DBF
11. Convert DBF to CF
12. Write CF to storage

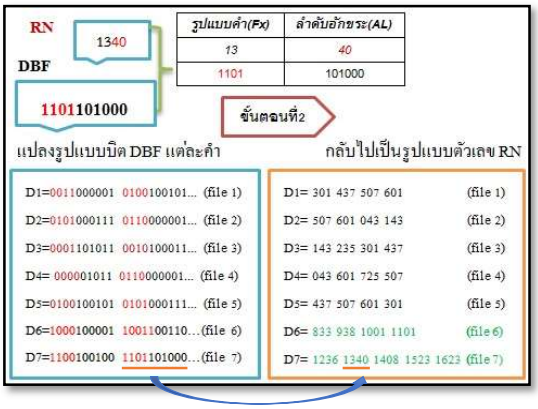
5. ขั้นตอนวิธีการคลายการบีบอัด

5.1 ภาพรวมของขั้นตอนวิธี

ขั้นตอนคลายการบีบอัดข้อความภาษาไทยนำเสนอ มี 4 ขั้นตอน ดังนี้

ขั้นตอนที่ 1: อ่านแฟ้มข้อความภาษาไทยที่ถูกบีบอัดคือ CF

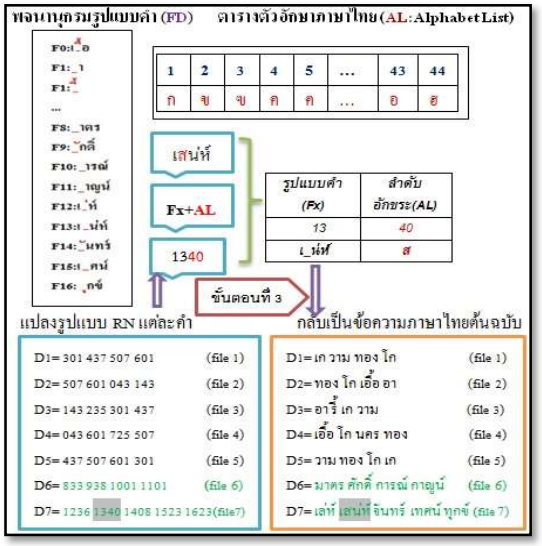
ขั้นตอนที่ 2: แปลงรูปแบบบิตข้อมูล DBF กลับเป็น RN ทีละบล็อกในรูปแบบจำนวนเต็ม แสดงตัวอย่าง ดังรูปที่ 8



รูปที่ 8. แสดงการแปลงรูปแบบบิตข้อมูล DBF กลับไปเป็น RN

ขั้นตอนที่ 3: ทำการแปลง RN กลับไปเป็นคำต้นฉบับ โดยใช้พจนานุกรมรูปแบบคำ (FD) พร้อมตารางตัวอักษร (AL) ที่มี

แสดงการแปลง RN จากเลขจำนวนเต็ม “1340” กลับเป็นคำศัพท์ต้นฉบับของคำว่า “เสน่ห์” โดยใช้พจนานุกรมรูปแบบคำ (FD) และตารางตัวอักษรภาษาไทย (AL) แสดงดังรูปที่ 9



รูปที่ 9. แสดงการคลายบีบอัดกลับมาเป็นแฟ้มต้นฉบับ OF

ขั้นตอนที่ 4 : เขียนคำศัพท์ภาษาไทยแต่ละคำ w_i ลงในพื้นที่สำหรับคลายบีบอัดชั่วคราว (TM) เพื่อให้ได้แฟ้มต้นฉบับแล้วเขียนลง OF ต่อไป

5.2 ขั้นตอนวิธีคลายบีบอัดข้อความภาษาไทย

ลำดับการทำงานของขั้นตอนวิธี คือ เมื่อใช้ขั้นตอนวิธีการคลายบีบอัด พื้นที่ชั่วคราวจะถูกจอง เริ่มต้นด้วย TM จากนั้น ขั้นตอนวิธีจะอ่านบล็อกที่เกี่ยวข้องของ DBF ทีละบล็อกและแปลงรูปแบบบิตกลับมาเป็นคำ ซึ่งจะถูกเก็บไว้ใน TM ถ้าบิตทั้งหมดถูกแปลง จากนั้นคำจะถูกจัดเก็บไว้ในที่เก็บข้อมูล (OF) นอกจากนี้ ให้ m_{max} เป็นความยาวสูงสุดในหน่วย s_m ขั้นตอนวิธีหลักแสดงดัง Algorithm 3: Word-Formation Thai Text Decompression Algorithm (WFTTDA)

Algorithm 3: WFTTDA
Input: CF, TD, m_{max}, n
Output: OF // OF: Output File
1. Initiate the space DBF with size m_{max}*n (TM)
2. While CF not EOF Do
3. Read data from CF
4. Convert each data to RN form
5. Read record which corresponds RN of TD and get word from TD
6. Write that word to TM
7. End While
8. Write TM to OF

6. ผลการทดลอง

6.1 ผลการทดลองเชิงทฤษฎี

ผลการทดลองเชิงทฤษฎีด้วยแฟ้มข้อมูลที่ถูกกำหนดขึ้นตัดคำมาแล้วพร้อมสุ่มคำที่มีรูปแบบสระหลากหลายเท่าเทียมกันเป็นคลังสำหรับทดสอบที่มุ่งเน้นไปที่เนื้อหาที่บันทึกในระบบคอมพิวเตอร์ไว้เป็นคำและรหัสแทนข้อมูลทั้ง ASCII-TIS620, Unicode และ UTF-8 โดยใช้สูตรการคำนวณ 1) ร้อยละพื้นที่ลดลงของแฟ้มที่ถูกบีบอัด และ 2) อัตราส่วนบีบอัด ()

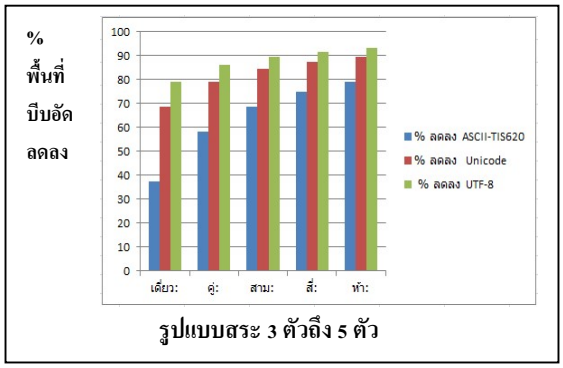
- % of Saved Space (per word) = ((Original Bits) - Required Bits)/Original Bits) *100 (1)
- Compression Ratio = Original Bits / Required Bits (2)

ดังแสดงในตารางที่ 1 แสดงขนาดพื้นที่ใช้จัดเก็บข้อความภาษาไทยต้นฉบับ แต่ละรูปแบบสระ ตั้งแต่สระเดี่ยว สระคู่ สระ 3 ตัว สระ 4 ตัว และสระ 5 ตัว ต่อข้อความภาษาไทย 1 คำ โดยเปรียบเทียบเนื้อที่ต้นฉบับเมื่อจัดเก็บข้อความด้วยระบบรหัส ASCII-TIS620, Unicode และ UTF-8 โดยที่ พื้นที่จัดเก็บข้อมูลต้นฉบับของรหัส ASCII-TIS620 : 1) สระเดี่ยว ขนาด 2 ไบต์ 2) สระคู่ ขนาด 3 ไบต์ 3) สระสาม ขนาด 3 ไบต์ 4) สระสี่ ขนาด 4 ไบต์ และสระห้า ขนาด 5 ไบต์

ตารางที่ 1. เปรียบเทียบขนาดพื้นที่ลดลงหลังบีบอัดต่อคำรูปแบบสระเดี่ยว สระคู่ สระ 3 ตัว สระ4 ตัว และสระ 5 ตัว

รูปแบบสระ	พื้นที่หลังบีบอัด	พื้นที่จัดเก็บก่อนบีบอัด | % พื้นที่ลดลงหลังบีบอัด					
		ASCII (บิต)	% บีบอัด	Unicode (บิต)	% บีบอัด	UTF-8 (บิต)	% บีบอัด
เดี่ยว:	10	16	37.5	32	68.75	48	79.17
สอง:	10	24	58.33	48	79.17	72	86.11
สาม:	10	32	68.75	64	84.38	96	89.58
สี่:	10	40	75.00	80	87.50	120	91.67
ห้า:	**10**	**48**	**79.17**	**96**	**89.58**	**144**	**93.06**
เฉลี่ย	10	32	63.75	64	81.876	96	87.918

ส่วนรูปที่ 10 แสดงแผนภูมิรูปภาพเปรียบเทียบพื้นจัดเก็บที่ลดลงของแฟ้มบีบอัด (CF) จากรหัสแทนข้อความภาษาไทยที่ต่างกัน ประกอบด้วย ASCII-TIS620 (1ไบต์) Unicode (2ไบต์) และ UTF-8 (3ไบต์) กับรูปแบบสระเดี่ยว ถึงสระจำนวนห้าตัว

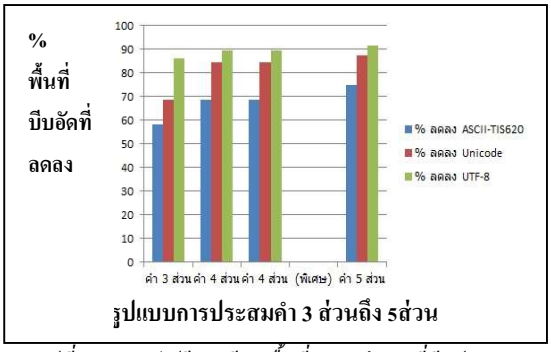


รูปที่ 10. เปรียบเทียบพื้นที่ลดลงต่อคำ สระขนาด 1-5 อักขระ โดยรูปแบบคำที่มีสระขนาด 5 ตัว มีร้อยละการบีบอัดสูงที่สุด

ตารางที่ 2 แสดงขนาดพื้นก่อนและหลังบีบอัดต่อคำ สำหรับการประสมคำแบบ 3 ส่วน 4 ส่วน 4 ส่วน(พิเศษ) และ 5 ส่วน

รูปแบบประสมคำ	พื้นที่หลังบีบอัด	พื้นที่จัดเก็บก่อนบีบอัด | % พื้นที่ลดลงหลังบีบอัด					
		ASCII (บิต)	% บีบอัด	Unicode (บิต)	% บีบอัด	UTF-8 (บิต)	% บีบอัด
คำ 3 ส่วน	10	24	58.33	48	68.75	72	86.11
คำ 4 ส่วน	10	32	68.75	64	84.38	96	89.58
คำ 4 ส่วน (พิเศษ)	10	32	68.75	64	84.38	96	89.58
คำ 5 ส่วน	**10**	**40**	**75.00**	**80**	**87.50**	**120**	**91.67**
เฉลี่ย	10	32	67.78	64.00	81.25	96	89.24

ส่วนรูปที่ 11 แสดงแผนภูมิเปรียบเทียบพื้นจัดเก็บที่ลดลงของแฟ้มบีบอัดจากรหัสแทนข้อความภาษาไทยที่ต่างกัน ประกอบด้วย ASCII-TIS620 (1ไบต์) Unicode (2ไบต์) และ UTF-8 (3ไบต์) ต่อคำ ทั้งตัวสะกดการันต์ ประสมคำ 3 ส่วน 4 ส่วน 4 ส่วน(พิเศษ)และ 5 ส่วน(พื้นที่จัดเก็บลดลงหน่วยบิต)



รูปที่ 11. กราฟเปรียบเทียบพื้นที่ลดลงต่อคำ ที่มีรูปแบบตัวสะกดการันต์ ตามรูปแบบประสมคำ 3, 4, 4(พิเศษ), 5 ส่วน

ตารางที่ 3. เปรียบประสิทธิภาพการบีบอัดแฟ้มข้อความภาษาไทยต้นฉบับขนาด 1 K-words (ทุกรูปแบบคำ)แจงตามรูปแบบรหัสแทนข้อความ (เฉพาะขั้นตอนวิธีที่นำเสนอ)

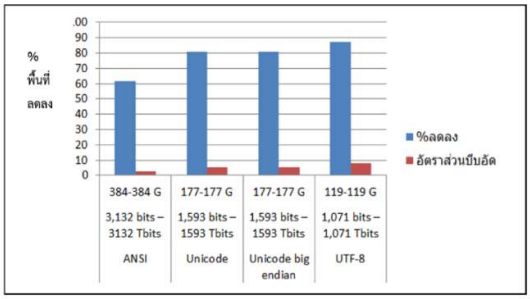
ระบบรหัส	ต้นฉบับ (ไบต์)	บีบอัด (ไบต์)	%บีบอัด	อัตราส่วนบีบอัด
ASCII-TIS620	4,099	1,280	71.55	3.51
Unicode	9,000	1,280	85.78	7.03
big endian	9,000	1,280	85.78	7.03
UTF-8	**13,402**	**1,280**	**90.45**	**10.50**

ตารางที่ 4 แสดงขนาดแฟ้มข้อความภาษาไทยต้นฉบับตั้งแต่ 1 KB – 1TB ของแต่ละรหัสแทนข้อมูลต่างกันทั้ง ASCII-TIS620 (1ไบต์) Unicode (2ไบต์) Unicode big endian (2ไบต์) และ UTF-8 (3ไบต์)

ตารางที่ 4. แสดงประสิทธิภาพการบีบอัดแฟ้มข้อความภาษาไทยที่มีขนาดตั้งแต่ 1 KB – 1TB

ระบบรหัส	ขนาดแฟ้มบีบอัด (บิต) (Tbits)	จำนวนคำ Giga-Word	%บีบอัด	อัตราส่วนบีบอัด
ASCII-TIS620	3,132– 3132	384-384	61.77	2.62
Unicode	1,593 – 1593	177-177	80.55	5.14
big endian	1,593 – 1593	177-177	80.55	5.14
UTF-8	**1,071 – 1,071**	**119-119**	**86.93**	**7.65**

รูปที่ 12 แสดงแผนภูมิรูปภาพเปรียบเทียบร้อยละของเนื้อจัดเก็บลดลงและอัตราบีบอัด (Compression Ratio) โดยรหัสแทนข้อความภาษาไทยต่างกันทั้ง ASCII-TIS620 (1ไบต์) Unicode (2ไบต์) Unicode big endian (2ไบต์) และ UTF-8 (3ไบต์) กับรูปแบบสระที่คละกัน (ตั้งแต่สระเดี่ยว – สระห้าตัว)



รูปที่ 12. เปรียบเทียบร้อยละเนื้อที่จัดเก็บลดลงและอัตราบีบอัดเมื่อคละรูปแบบสระตั้งแต่ 1-5 ตัว

6.2 เปรียบเทียบประสิทธิภาพกับโปรแกรมบีบอัดที่นิยม

คณะวิจัยได้พัฒนาโปรแกรมตามขั้นตอนวิธีที่นำเสนอด้วยภาษาไพธอน เพื่อเทียบประสิทธิภาพกับโปรแกรมบีบอัดข้อมูลที่นิยมใช้กันทั่วไป คือ *7-Zip, WinRAR และ WinZIP* โดยเปรียบเทียบทั้งในมุมขนาดแฟ้มและรหัสแทนข้อความ

1) โดยสุ่มข่าวภาษาไทยอินเทอร์เน็ต [23] เข้าสู่พจนานุกรมขนาด 1 กิโลไบต์ ถึง 100 กิโลไบต์ จากนั้นบีบอัดข้อมูลจากขั้นตอนวิธีที่พัฒนาขึ้นเพื่อเปรียบเทียบ ผลการทดลองดังนี้

ตารางที่ 5. แสดงประสิทธิภาพการบีบอัดแฟ้มข่าวภาษาไทย **ด้วยขั้นตอนวิธี WFTTC** *ขนาดแฟ้มตั้งแต่ 1KB ถึง 100KB แจงตามการเข้ารหัส*

ระบบรหัส	ANSI-TIS620		Unicode		UTF-8	
แฟ้มข่าวต้นฉบับ	แฟ้มบีบอัด (ไบต์)	%บีบอัด	แฟ้มบีบอัด (ไบต์)	%บีบอัด	แฟ้มบีบอัด (ไบต์)	%บีบอัด
1 KB	199	80.57	141	86.62	**121**	**88.18**

2) เปรียบเทียบประสิทธิภาพการบีบอัดข่าวของวิธีการที่นำเสนอเทียบกับซอฟต์แวร์บีบอัดข้อมูลที่นิยมใช้ทั่วไป ด้วยการบีบอัดแฟ้มข่าวภาษาไทยขนาด 1 กิโลไบต์ ถึง 100 กิโลไบต์ ได้ผลการทดลองดังตารางต่อไปนี้

ตารางที่ 6. เปรียบเทียบประสิทธิภาพวิธีที่นำเสนอกับซอฟต์แวร์บีบอัดข้อมูลบีบอัดแฟ้มข่าวภาษาไทยขนาดแฟ้ม 1KB แยกตามรหัส

ระบบรหัส	ANSI-TIS620		Unicode		แบบ UTF-8	
วิธีบีบอัด	%บีบอัด	อัตราส่วนบีบอัด	%บีบอัด	อัตราส่วนบีบอัด	%บีบอัด	อัตราส่วนบีบอัด
WFTTC	**80.57**	**5.15**	**86.62**	**7.47**	**91.11**	**11.3**
7-ZIP	76.56	4.27	77.25	4.39	77.25	4.53
WinRar	**81.64**	**5.45**	83.01	5.89	82.42	5.69
WinZIP	73.83	3.82	74.32	3.89	74.02	3.85

ตารางที่ 7. เปรียบเทียบประสิทธิภาพวิธีที่นำเสนอกับซอฟต์แวร์บีบอัดข้อมูล บีบอัดแฟ้มข่าวภาษาไทยขนาดแฟ้ม 10KB แยกตามรหัส

ระบบรหัส	ANSI-TIS620		Unicode		UTF-8	
วิธีบีบอัด	%บีบอัด	อัตราส่วนบีบอัด	%บีบอัด	อัตราส่วนบีบอัด	%บีบอัด	อัตราส่วนบีบอัด
WFTTC	**78.09**	**4.56**	**81.50**	**5.41**	**89.08**	**9.16**
7-ZIP	63.68	2.75	75.80	4.13	78.81	4.72
WinRar	63.79	2.76	74.14	3.86	78.08	4.56
WinZIP	63.38	2.73	72.88	3.68	77.61	4.46

ตารางที่ 8. เปรียบเทียบประสิทธิภาพวิธีที่นำเสนอกับซอฟต์แวร์บีบอัดข้อมูล บีบอัดแฟ้มข่าวภาษาไทยขนาดแฟ้ม 100KB แยกตามรหัส

ระบบรหัส	ANSI-TIS620		Unicode		UTF-8	
วิธีบีบอัด	%บีบอัด	อัตราส่วนบีบอัด	%บีบอัด	อัตราส่วนบีบอัด	%บีบอัด	อัตราส่วนบีบอัด
WFTTC	**84.55**	**6.47**	**84.76**	**6.56**	**88.09**	**8.59**
7-ZIP	71.66	3.53	83.92	6.22	85.97	7.13
WinRar	70.39	3.38	82.29	5.64	84.96	6.65
WinZIP	63.38	2.73	81.30	5.35	84.59	6.48

7. อภิปรายผล

พิจารณาผลลัพธ์ทางทฤษฎี เมื่อกำหนดรูปแบบสระ 1-5 อักขระ เก็บด้วย ASCII-TIS620 บีบอัดได้ร้อยละ 37.50, 58.33, 68.75, 75.0 และ 79.17 ตามลำดับ Unicode บีบอัดได้ร้อยละ 68.75, 79.17, 84.37, 88.75 และ 89.58 ตามลำดับ สำหรับ UTF-8 บีบอัดได้ร้อยละ 79.17, 86.11, 89.58, 91.67 และ 93.06 อธิบายได้ว่า ขั้นตอนวิธีใหม่นี้สามารถทำงานได้ดี บีบอัดได้มากยิ่งขึ้นเมื่อขนาดรูปแบบสระมีจำนวนอักขระมากขึ้น

เมื่อพิจารณารูปแบบการประสมคำแบบ 3 ส่วน 4 ส่วน 4 ส่วน(พิเศษ) และ 5 ส่วน พบว่า เมื่อเก็บด้วย ASCII-TIS620 บีบอัดได้ร้อยละ 58.33, 68.75, 68.75, และ 75.0 ตามลำดับ Unicode บีบอัดได้ร้อยละ 68.75, 84.38, 84.38, และ 87.50 ตามลำดับ สำหรับ UTF-8 บีบอัดได้ร้อยละ 86.11, 89.58, 89.58 และ 91.67 อธิบายได้ว่า ขั้นตอนวิธีใหม่นี้สามารถทำงานได้ดี บีบอัดได้มากยิ่งขึ้นเมื่อรูปแบบการประสมคำมากขึ้น แต่น้อยกว่าการกำหนดรูปแบบสระเล็กน้อย

พิจารณาเชิงทฤษฎีของข้อความภาษาไทยขนาด 1 กิโลคำ (รูปแบบคำคละกันและขนาดแฟ้มไม่เท่ากัน) จัดเก็บด้วยระบบรหัส ASCII-TIS620, Unicode, Unicode big endian, และ UTF-8 พบว่า ร้อยละของการบีบอัดสามารถทำได้ ตามลำดับดังนี้ 71.55, 85.78, 85.78, และ 90.45 ในขณะที่อัตราการบีบอัดสามารถทำได้ที่ 3.51, 7.03, 7.03, และ 10.50 ตามลำดับ อธิบายได้ว่า เมื่อกำหนดขนาดปริมาณของคำแทนรูปแบบสระและรูปแบบการประสมคำ ขั้นตอนวิธีใหม่ก็ยังสามารถทำงานได้ในอัตราการบีบอัดและร้อยละของการบีบอัดสามารถทำได้มากเช่นการกำหนดตามรูปแบบสระและการประสมคำอย่างมีนัยสำคัญเช่นกัน

เมื่อทดลองเขียนโปรแกรมเพื่อเทียบประสิทธิภาพกับโปรแกรมบีบอัดข้อมูลที่นิยมใช้กันทั่วไป โดยสุ่มแฟ้มข่าวขนาด 1 กิโลไบต์ ถึง 100 กิโลไบต์ แทนข้อมูลด้วย ASCII-TIS620 พบว่า วิธีการที่นำเสนอมีประสิทธิภาพสูง สามารถบีบอัดข่าวได้ตั้งแต่ร้อยละ 78.09 – 84.55 อัตราส่วนการบีบอัด 4.56 – 6.47 เท่า รหัส Unicode พบว่า วิธีการที่นำเสนอมีประสิทธิภาพสูง สามารถบีบอัดข่าวได้ตั้งแต่ร้อยละ 81.50 – 86.62 อัตราส่วนการบีบอัดได้ตั้งแต่ 5.41 - 7.47 เท่า และรหัส UTF-8 พบว่า วิธีการที่นำเสนอมีประสิทธิภาพสูง สามารถบีบอัดข่าวได้ตั้งแต่ร้อยละ 88.09 – 91.11 อัตราส่วนการบีบอัดสูงตั้งแต่ 8.59 - 11.30 เท่า

อธิบายได้ว่า การใช้งานขั้นตอนวิธีใหม่เมื่อนำไปใช้งานในสถานการณ์จริงสามารถบีบอัดได้สูงทั้งในเชิงทฤษฎีและโปรแกรมที่ใช้งานจริงอยู่อย่างมีนัยสำคัญทั้งร้อยละและอัตราการบีบอัด

8. สรุปผลการวิจัยและข้อเสนอแนะ

บทความวิจัยนี้นำเสนอขั้นตอนวิธีใหม่สำหรับการบีบอัดข้อความภาษาไทยด้วยรูปแบบการประสมคำที่ครบทั้งพยัญชนะ สระ วรรณยุกต์ ตัวสะกด การันต์ โดยไม่อนุญาตให้มีการสูญเสียข้อมูลระหว่างการบีบอัดและคลายบีบอัด กลไกของขั้นตอนวิธีได้ออกแบบขั้นตอนวิธีส่วนเตรียมการประมวลผลเพื่อนำรูปแบบการสร้างคำของภาษาไทยเก็บไว้ในพจนานุกรมแบบใหม่โดยอาศัยรูปแบบการประสมคำครบถ้วนสมบูรณ์ตามรูปแบบคำที่มีในภาษาไทย จากนั้นใช้ขั้นตอนวิธีที่พัฒนาขึ้นมา (WFTTCA) เพื่ออ่านข้อมูลที่จะบีบอัดโดยอ้างอิงรูปแบบคำจากพจนานุกรม แล้วแทนรูปแบบการบีบอัดแต่ละเป็นบิตก่อนเขียนลงสู่แฟ้มที่ถูกบีบอัด และยังสามารถคลายข้อมูลด้วยขั้นตอนวิธีการคลายข้อมูล (WFTTDA) ที่พัฒนาขึ้นตามแนวทางนี้ได้แบบไม่มีการสูญเสียข้อมูลระหว่างการบีบอัด ผลการทดลองทางทฤษฎีและในเชิงเปรียบเทียบชี้ให้เห็นว่า ขั้นตอนวิธีที่พัฒนาขึ้นใหม่สามารถบีบอัดข้อมูลได้ด้วยประสิทธิภาพสูงมากอย่างมีนัยสำคัญ ตั้งแต่ร้อยละ 78.09 ถึง 91.11 ด้วยอัตราการบีบอัดมากที่สุดถึง 4.56 ถึง 11.30 เท่า แม้ทดสอบบีบอัดแฟ้มหลายขนาดตั้งแต่ 1 กิโลไบต์ถึง 100 กิโลไบต์ก็ยังคงมีประสิทธิภาพสูงทั้งในมุมมองของร้อยละการบีบอัดและอัตราส่วนการบีบอัด

ข้อเสนอแนะสำหรับพัฒนางานวิจัยต่อไป มีข้อควรพิจารณาคือเมื่อปริมาณข้อความมีขนาดมากยิ่งขึ้นแนวโน้มของร้อยละการบีบอัดจะลดลงบ้าง ดังนั้นหากจะพัฒนาให้มีประสิทธิภาพมากยิ่งขึ้น ควรศึกษาลักษณะของคำภาษาไทยที่มีรูปแบบแตกต่างกันในการบีบอัด เพื่อจะได้ทำให้ทราบถึงความเหมาะสมในการนำไปพัฒนาขั้นตอนวิธีให้มีประสิทธิภาพมากยิ่งขึ้น อีกประการคือ น่าจะขยายแนวคิดและขีดความสามารถของขั้นตอนวิธีไปสู่การบีบอัดข้อมูลภาษาอื่นๆ ต่อไปได้

เอกสารอ้างอิง

[1] Z. Karim Zia, D. Fayzur Rahman, and C. Mofizur Rahman. Two-Level Dictionary-Based Text Compression Scheme . Proceedings of 11th International

Conference on Computer and Information Technology (ICCIT 2008) 25-27 December, 2008, Khulna, Bangladesh, 13-18.

[2] W. Wen-Yen and J. W. Mao-Jiun, "Two-dimensional object recognition through two-stage string matching," Image Processing, IEEE Transactions on, vol. 8, 978-981, 1999.

[3] F. Amar Mukherjee. Data Compression Using Encrypted Text Robert. Proceedings of ADL '96 ,1996, 130-138.

[4] G. Hwee Ong and S. Ying Huang. A Data Compression Scheme for Chinese Text Files Using Huffman Coding and a Two-Level Dictionary. INFORMATION SCIENCES 84, 85 99 (1995) 85-99.

[5] A. A. Sharieh. An enhancement of Huffman coding for the compression of multimedia file. Transactions of Engineering Computing and Technology, Vol. 3, No. 1, 2004, 303-305.

[6] C. Khancome. Bit-level Text Compression Algorithm Using Position of Characters. 2010 2nd International Conference on Information and Multimedia Technology (ICIMT 2010). Vol. 1-242, 2010, 242-245.

[7] C. Khancome. New Full Text Compression Algorithm Based on Position of Character. 2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010). IEEE Conference, Vol. 5, 2010, 631-634.

[8] ประหยัด เลวัน เชาวลิต ขันคำ, "ขั้นตอนวิธีการบีบอัดข้อความภาษาไทยด้วยรูปแบบสระ" The 15th National Conference on Information Technology (NCIT2023), เชียงราย, ประเทศไทย, 2566, หน้า 50-55.

[9] สัญฉกร วุฒิสิทธิกุลกิจ, สุวิทย์ นาคพีระยุทธ, ปิติฉัตร สุทธาโรจน์ และ สมภพ โชคชัยธรรม. เทคโนโลยีการบีบอัดข้อมูลเบื้องต้น, สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย: กรุงเพพฯ, 2549.

[10] M. Crochemore, and W. Rytter, (2023, March, 18). Text Algorithms. Available: http://monge.univ-mlv.fr/~mac/REC/ text-algorithms.pdf.

[11] A. Mofat, and R.Y.K. Isal. Word-based text compression using the burrows-wheeler transform. Information Processing and Management, Vol. 41, No. 5, 2005, 1175-1192.

[12] J. Adiego, and P. de. la Feunte, On the use of words as source alphabet symbols in PPM. In Proceedings of Data Compression Conference, IEEE, 2006, 435.

[13] J. Lánský and M. Žemlička. Text compression: Syllables. In Proceedings of the Dateso Workshop on Database, Texts, Specifications and Objects, 2005, 32-45.

[14] H. Al-Bahadili and S. M. Hussain. An adaptive character wordlength algorithm for data compression. Computers & Mathematics with Applications, Vol. 55, No. 6, 2008, 1250-1256.

[15] S. Nofal. Bit-level text compression. In Proceedings of the 1st International Conference on Digital Communications and Computer Applications, Irbid, Jordan, 2007, 486-488.

[16] A. Rababáa. An Adaptive Bit-Level Text Compression Scheme Based on the HCDC Algorithm. M.Sc., dissertation, Amman Arab University for Graduate Studies, Amman, Jordan, 2008.

[17] H. Al-Bahadili and S. M. Hussain. A Bit-level Text Compression Scheme Based on the ACW Algorithm. International Journal of Automation and Computing, Vol. 7 No. 1, 2010, 123-131.

[18] C. Khancome. Text Compression Algorithm Using Bits for Character Representation. International Journal of Advanced Computer Science. Vol. 1, No. 6, 2010, 215-219.

[19] เศกสิทธิ์ พจมารและจารี ทองคำ "การเปรียบเทียบขั้นตอนวิธีการบีบอัดข้อมูลแบบไม่สูญเสียข้อมูลบนเว็บแอปพลิเคชัน" RMUTT JOURNAL Science and Technology Vol.13 No. 3, pp 120-133, Sep-Dec. 2020.

[20] ชนาภา ศิลาวงษ์และธนภัทร์ อนุศาสน์อมรกุล "การศึกษาเปรียบเทียบวิธีบีบอัดข้อมูลที่เหมาะสมสำหรับแต่ละประเภทข้อมูล" วารสารวิศวกรรม มก.. ฉบับที่ 91 ปีที่ 28 หน้า 83-92 มกราคม-มีนาคม 2558.

[21] บรรพต ดลวิทยา "ศึกษาการบีบอัดเอกสารเอชทีเอ็มแอลบนฝั่งเซิร์ฟเวอร์ด้วยการขั้นตอนวิธีแบบ Huffman"

วิทยานิพนธ์วิทยาศาสตร์มหาบัณฑิต สาขาวิทยการคอมพิวเตอร์ มหาวิทยาลัยศิลปากร 2550.

[22] ทีมงานทรูปลูกปัญญา, หลักการสร้างคำในภาษาไทย, (Access 4 ธ.ค. 66), [Online] Available: https://www.trueplookpanya.com/learning/detail/34513.

[23] รัฐบาลไทย-ข่าวทำเนียบรัฐบาล. (Access 4 ธ.ค. 66), [Online] Available: https://www.thaigov.go.th/news/contents/details/31431.