เครื่องจักรเชิงน่าจะเป็นสำหรับเรียนรู้ส่วนเพิ่มเพื่อการจำแนกลำดับสัญลักษณ์

Incremental Learning Probabilistic State Machine for Symbolic Sequences Classification

จิตรกร พูลโพธิ์ทอง[1]

บทคัดย่อ

การจำแนกลำดับสัญลักษณ์สามารถนำไปใช้ประโยชน์ได้หลากหลาย เช่น การวิเคราะห์ดีเอ็นเอ การตรวจจับการบุกรุก การวิเคราะห์คลื่นไฟฟ้าหัวใจ ปัจจุบันมีวิธีการมาตรฐานที่ประยุกต์ใช้ได้กับเรื่องนี้ ได้แก่ แบบจำลองภาษาเชิงน่าจะเป็น แบบจำลองโครงข่ายประสาทเทียม เครื่องจักรเวกเตอร์สนับสนุน เป็นต้น อย่างไรก็ตาม ในกรณีที่ข้อมูลอินพุตสำหรับเรียนรู้เป็นสายอักขระที่มีความยาวมากและไม่มีการแบ่งคำ เช่น สายอักขระดีเอ็นเอ เป็นต้น การนำข้อมูลเหล่านี้ไปเรียนรู้ จะต้องแบ่งคำสมมติที่มีความยาวจำกัด ซึ่งทำให้ความน่าจะเป็นและตำแหน่งของคำที่ถูกต้องบิดเบือนไปจากข้อมูลต้นฉบับ ผลที่ตามมาคือ การนำไปประยุกต์ใช้จะได้ผลลัพธ์ไม่ตรงเท่าที่ควร อีกทั้งจะทำให้การเรียนรู้ส่วนเพิ่มด้วยข้อมูลใหม่เพิ่มเติมจะได้ผลลัพธ์ที่คลาดเคลื่อนเช่นกัน งานวิจัยนี้จึงเสนอแบบจำลองการเรียนรู้สายอักขระแบบใหม่ ที่สามารถเรียนรู้ส่วนเพิ่มได้ไม่จำกัด และไม่ต้องแบ่งคำสมมติ แต่ยังคงข้อมูลสถิติของสายอักขระย่อยที่ถูกต้องให้ได้มากที่สุด ในขณะที่มิติข้อมูลอยู่ในขอบเขตที่สามารถจัดการได้ โดยจะใช้วิธีการแบ่งคำอัตโนมัติด้วยสายอักขระเอกลักษณ์และสายอักขระเกิดซ้ำ เพื่อทดสอบประสิทธิภาพของแบบจำลองในงานวิจัย เราได้ทดลองจำแนกสายอักขระดีเอ็นเอของแบคทีเรีย อี.โคไล (E. Coli) 2 กลุ่ม คือ กลุ่มที่เป็นตัวสนับสนุน และกลุ่มที่ไม่เป็น ผลการทดลองพบว่ามีความแม่นยำในการจำแนกกลุ่มถูกต้องร้อยละ 96.23 ซึ่งถือว่ามีความแม่นยำสูงเมื่อเทียบกับวิธีการมาตรฐานอื่น

คำสำคัญ: การจำแนกลำดับสัญลักษณ์ ภาษาเชิงน่าจะเป็น สายอักขระไม่แบ่งส่วน ออโตมาตาเชิงน่าจะเป็น เอ็นแกรม

ABSTRACT

Symbolic sequence classification can be used in a variety of applications such as DNA sequences analysis, intrusion detection, electrocardiography (ECG) analysis. The convention methods which can be applied to this field are for example, probabilistic language model, support vector machine, and artificial neural network. However, sometime the list of words in very long sequences such as DNA sequences are unknown. The fix size of imitation words are used for this case. Consequently, the probability and position of original unknown words are distorted which can lead to incorrect results in long term. Moreover, the update learning may cause difficulty. This research proposes a novel probabilistic language model that can learn infinitely increments and do not have to fix the size of imitation words, but still retain the statistical information of substring accurately as possible while the size of model is tractable. The experiment of classification is

[1] ผู้ช่วยศาสตราจารย์ สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยรามคำแหง

applied to DNA sequences of promoter and non-promoter bacteria E. Coli. The accuracy of classification is 96.23 % which is highly accurate compared to the other standard methods.

Keyword: symbolic sequence classification, probabilistic language, unsegmented string, probabilistic finite automata, n-gram

บทนำ

การจำแนกลำดับสัญลักษณ์ (symbolic sequence classification) เป็นวิธีการจำแนกกลุ่มของข้อมูลประเภทสายอักขระหรือลำดับสัญลักษณ์ ซึ่งสามารถนำไปใช้ประโยชน์ได้หลากหลาย เช่น การวิเคราะห์ดีเอ็นเอ การตรวจจับ การบุกรุก การวิเคราะห์คลื่นไฟฟ้าหัวใจ (Xing et al., 2010) วิธีการมาตรฐานที่สามารถประยุกต์ใช้ได้กับเรื่องนี้ ได้แก่ แบบจำลองภาษาเชิงน่าจะเป็น (Vidal et al., 2005) เช่น เอ็นแกรม (n-gram) (Cavnar and Trenkle 1994) แบบจำลองมาร์คอฟแบบซ่อน (hidden Markov model) (Rabiner, 1989) (Khreich et al., 2012) และเครื่องจักรเวกเตอร์สนับสนุน (support vector machine) (Lodhi et al., 2002) (Leslie et al., 2002) เป็นต้น แต่เนื่องจากข้อมูลสายอักขระใน ปัจจุบัน มีแนวโน้มเพิ่มขึ้นอย่างมาก ข้อมูลบางประเภทเป็นสายอักขระที่มีความยาวมาก และไม่มีการแบ่งคำใน สายอักขระ เช่น สายอักขระดีเอ็นเอ เป็นต้น ทำให้เกิดปัญหาในการวิเคราะห์ เช่น คำที่มีผลต่อการวิเคราะห์ข้อมูล มีอะไรบ้าง และควรมีความยาวเท่าไรจึงจะเหมาะสม ซึ่งหากวิเคราะห์สถิติการปรากฏของสายอักขระย่อยทั้งหมด เพื่อหาตัวแปรสำคัญ จะทำให้มิติของข้อมูลเพิ่มขึ้นจนไม่สามารถจัดการได้ โดยปัจจุบันการวิเคราะห์สายอักขระ มักจะใช้วิธีแบบจำลองภาษา เช่น เคเมอร์ (k-mer) โดยทั่วไปเรียกว่าเอ็นแกรม (n-gram) ซึ่งเป็นวิธีการเลือก สายอักขระย่อยความยาวจำกัด แทนที่จะวิเคราะห์สถิติของสายอักขระย่อยทั้งหมด ถึงกระนั้นยังพบปัญหาคือ สายอักขระดีเอ็นเอหรือข้อมูลลักษณะเดียวกันนี้ อาจจะมีสายอักขระสำคัญ ที่มีความยาวหลากหลาย การใช้สายอักขระความยาวจำกัดบางส่วน จะทำให้บิดเบือนสถิติของการเกิดคำ และตำแหน่งของคำที่ถูกต้อง ซึ่งส่งผลให้ การนำไปใช้งานมีความคลาดเคลื่อน การแก้ปัญหานี้จึงใช้คำมีความยาวเหมาะสมต่อข้อมูล ซึ่งงานวิจัยที่ผ่านมาจะ ใช้วิธีการหาค่า *k* (ค่า *n* สำหรับ *n*-gram) ที่ดีที่สุด เช่น ค่า *k* ที่ทำให้เกิดความแตกต่างของคำมากที่สุด (Rayan and Paul, 2014) หรือ หาค่า *k* ที่แปรผันได้ โดยหาจากสถิติของคำความยาว *k* ที่มีนัยสำคัญ (Ron et al., 1996), (Galata et al., 2001) อย่างไรก็ตามความเห็นของผู้วิจัย การประมาณค่า *k* ที่เหมาะสมตามงานวิจัยก่อนหน้านั้นจะใช้ ความยาวคงที่ ซึ่งยังคงให้ผลสถิติของข้อมูลที่มีความยาวนอกเหนือจากที่กำหนดคลาดเคลื่อนไปได้ นอกจากนี้การนำแบบจำลองไปเรียนรู้ข้อมูลส่วนเพิ่มก็อาจจะเกิดความคลาดเคลื่อน เพราะความยาวเหมาะสมที่เลือกใช้อาจจะ เปลี่ยนไปตามสภาพของข้อมูล ดูรายละเอียดเรื่องการเรียนรู้ส่วนเพิ่มจาก (Ade and Deshmukh, 2013), (Geppert and Hammer, 2016)

งานวิจัยนี้จึงสนใจการสร้างแบบจำลองภาษาเชิงน่าจะเป็นแบบใหม่ที่สามารถเก็บสถิติของสายอักขระย่อยให้ คงความถูกต้องให้ได้มากที่สุด โดยไม่จำเป็นต้องแบ่งคำหรือจำกัดความยาวของคำที่ใช้เรียนรู้ และสามารถเรียนรู้ ส่วนเพิ่มได้อย่างมีประสิทธิภาพ ในขณะที่มิติของข้อมูลที่เพิ่มขึ้นในขณะเรียนรู้จะยังคงสามารถจัดการได้ โดยมี แนวคิดว่า การวิเคราะห์สถิติของสายอักขระย่อย ควรพิจารณาจากสายเอกลักษณ์และอักขระเกิดซ้ำทุกความยาวที่มีเพื่อเก็บสถิติให้ตรงมากที่สุด และเพื่อรองรับสถิติของข้อมูลใหม่ที่จะเกิดขึ้นได้โดยไม่ต้องทำการเรียนรู้ใหม่ทั้งหมด โดยตัดสายอักขระย่อยที่ไม่เกิดประโยชน์เพื่อลดทอนขนาดของข้อมูลที่ได้จากการเรียนรู้ จากงานวิจัย (Ilie, 2011) พบว่าสายอักขระทุกสายจะมีคุณสมบัติสายอักขระย่อยเอกลักษณ์สั้นสุด (minimum unique substring) และจะเกิดขึ้นคู่กันกับสายอักขระย่อยเกิดซ้ำยาวสุด (maximum repeated substring) เสมอ ซึ่งหลักการนี้จะเป็นแนวทางที่ช่วยในการตัดสายอักขระเอกลักษณให้สั้นลง ในขณะที่ยังคงความน่าจะเป็นที่ถูกต้องได้ งานวิจัยนี้จึงได้ค้นคว้าวิจัยเพื่อสร้างแบบจำลองแบบใหม่โดยใช้ชื่อว่า ออโตมาตาเชิงน่าจะเป็นของ

สายอักขระเอกลักษณ์ที่เหมาะสม (proper unique substring finite automata: PUSFA) ซึ่งเป็นแบบจำลองที่ใช้เก็บสาย อักขระย่อยเอกลักษณ์ที่เหมาะสมและสายอักขระย่อยเกิดซ้ำ รวมถึงความถี่ของสายอักขระเหล่านั้น โดยสามารถ ประยุกต์ใช้คำนวณหาความน่าจะเป็นของสายอักขระใดๆ เพื่อจำแนกข้อมูล หรือเพื่อทำนายข้อมูล รวมถึงเพื่อวิเคราะห์ สายอักขระที่เกิดขึ้นเชิงสถิติ โดยสามารถเรียนรู้จากสายอักขระไม่จำกัดความยาวได้โดยไม่ต้องทำการแบ่งคำ เราได้เสนอขั้นตอนวิธีในการสร้างแบบจำลอง และทำการทดลองนำแบบจำลองไปประยุกต์ใช้จำแนกกลุ่มข้อมูล เพื่อทดสอบว่ามีความแม่นยำมากน้อยเพียงใดเมื่อเทียบกับวิธีการมาตรฐานอื่น และทำการทดสอบว่าขนาดของแบบจำลองจะเติบโตอย่างไร และเวลาที่ใช้ประมวลผลนั้นเติบโตอย่างไร เมื่อเทียบกับความยาวของสายอักขระ ที่นำมาเรียนรู้ เพื่อที่จะรับรองได้ว่า เมื่อนำแบบจำลองไปใช้งานแล้ว แบบจำลองจะยังคงสามารถจัดการได้จริง ในทางปฏิบัติ เนื้อหาในบทความนี้จะประกอบไปด้วย วิธีการดำเนินการวิจัย ซึ่งจะกล่าวถึงทฤษฎีและนิยามที่เกี่ยวข้อง รายละเอียดแบบจำลองที่นำเสนอ วิธีการสร้างแบบจำลอง วิธีการนำแบบจำลองไปใช้งาน การทดลองที่เกี่ยวข้อง ตามด้วยผลการวิจัยและสรุปและวิจารณ์ผล

วิธีดำเนินการวิจัย

การวิเคราะห์สายอักขระ S ใดๆ เพื่อหาสถิติของคำที่เกิดขึ้นให้ได้ความถูกต้องสูงสุดนั้น จะต้องพิจารณาจาก สายอักขระย่อยที่เป็นไปได้ทั้งหมดของ S หากสมมติให้ $S = x_1 x_2 x_3 ... x_l$ สายอักขระย่อยของ S ที่เป็นไปได้ทั้งหมดนั้นได้แก่ สายอักขระย่อยความยาว 1, 2, 3... ไปจนถึงความยาว l ซึ่งจะต้องพิจารณาคำทั้งหมด $1+2+3+...+l$ หรือ ประมาณ $(l^2 + l)/2$ แต่เนื่องจากในกรณีที่ความยาวของ S ยาวมาก จำนวนของคำที่จะต้องพิจารณา จะมีจำนวนมาก เกินกว่าจะจัดการได้ การตัดทอนสายอักขระย่อยบางส่วนออกไปจะทำให้ลดมิติของข้อมูลไปได้ ขณะเดียวกันจะต้องยังคง ความน่าจะเป็นที่ถูกต้องของสายอักขระย่อยเอาไว้ให้ได้ด้วย ซึ่งจากการสังเกตพบว่าสายอักขระย่อยยิ่งยาวมากขึ้นโอกาสจะเป็นสายอักขระเกิดครั้งเดียวก็มากขึ้น ซึ่งมีจำนวนมากที่สุด การลดทอนในงานวิจัยนี้จึงตัดเฉพาะสายอักขระเอกลักษณ์ ให้เหลือเพียงสายอักขระเอกลักษณ์ที่เพียงพอต่อการคำนวณความน่าจะเป็นเท่ากับ 1 ยกตัวอย่าง ให้ S = *CACGTGCGA* เห็นได้ว่า *GC* เกิดขึ้นครั้งเดียว ดังนั้นความน่าจะเป็น $P(G|GC)$ คือ 1 และ $P(G|TGC)$ = $P(G|GTGC)$ = $P(G|CGTGC)$ = 1 ดังนั้นคำที่เพียงพอต่อการหาความน่าจะเป็น จะใช้เพียง *GC* ก็พอ คำที่ยาวกว่านั้นจะตัดทิ้งไป แต่ในกรณี *CG* เป็น สายอักขระที่ปรากฏซ้ำ 2 ครั้ง ดังนั้น $P(T|CG)$ = 1/2 แต่ $P(T|ACG)$ = 1 และ $P(A|CG)$ = 1/2 แต่ $P(A|GCG)$ = 1 การใช้ CG เป็นเงื่อนไขคำนวณความน่าจะเป็นจะไม่เพียงพอ ดังนั้นการหาความน่าจะเป็นตามเงื่อนไขที่ครบจึงต้องมีสถิติของ คำที่มี CG เป็นคำเติมท้ายที่ยาวกว่าเดิม ซึ่งก็คือ *ACG* และ*GCG* ด้วย คำเหล่านี้เราได้นิยามไว้ในด้านล่าง และแบบจำลองเพื่อหาความน่าจะเป็น จะพิจารณาจากคำเหล่านี้ ดูตัวอย่างการสร้างแบบจำลองขึ้นด้วย S = *CACGTGCGA* ตามรูปที่ 1 ซึ่งเราได้เสนอขั้นตอนวิธีการสร้างแบบจำลองนี้ด้วยขั้นตอนวิธีออนไลน์ เพื่อที่จะสามารถรับข้อมูล มาทีละตัวแล้วสร้างเพื่อรองรับการเรียนรู้ส่วนเพิ่มเพื่อรองรับการเรียนรู้แบบไม่จำกัด รายละเอียดอยู่ในเนื้อหา ส่วนถัดไป

สถานะ (Q)	สถานะเอกลักษณ์ยาวสุดก่อนหน้า (ρ)	สถานะปลายทางและความถี่ของการเกิดตัวอักษรต่อจากสถานะ (δ, τ)				ประเภทสายอักขระ (U, R)
		อักขระ A	อักขระ C	อักขระ G	อักขระ T	
ε	-	(A, 2)	(C, 3)	(G, 3)	(T, 1)	เกิดซ้ำ
A	-	(—, 0)	(AC, 1)	(—, 0)	(—, 0)	เกิดซ้ำ
C	-	(CA, 1)	(—, 0)	(CG, 2)	(—, 0)	เกิดซ้ำ
G	-	(GA, 1)	(GC, 1)	(—, 0)	(GT, 1)	เกิดซ้ำ
T	-	(—, 0)	(—, 0)	(TG, 1)	(—, 0)	เกิดซ้ำ
AC	CA	(—, 0)	(—, 0)	(ACG, 1)	(—, 0)	เอกลักษณ์สั้นสุด
CA	-	(—, 0)	(AC, 1)	(—, 0)	(—, 0)	เอกลักษณ์สั้นสุด
CG	-	(GA, 1)	(—, 0)	(—, 0)	(GT, 1)	เกิดซ้ำ
GA	GCG	(—, 0)	(—, 0)	(—, 0)	(—, 0)	เอกลักษณ์สั้นสุด
GC	TG	(—, 0)	(—, 0)	(GCG, 1)	(—, 0)	เอกลักษณ์สั้นสุด
GT	ACG	(—, 0)	(—, 0)	(TG, 1)	(—, 0)	เอกลักษณ์สั้นสุด
TG	GT	(—, 0)	(GC, 1)	(—, 0)	(—, 0)	เอกลักษณ์สั้นสุด
ACG	-	(—, 0)	(—, 0)	(—, 0)	(GT, 1)	เอกลักษณ์เหมาะสม
GCG	-	(GA, 1)	(—, 0)	(—, 0)	(—, 0)	เอกลักษณ์เหมาะสม

รูปที่ 1 ตัวอย่างแบบจำลองเก็บสายอักขระเกิดซ้ำและสายอักขระเอกลักษณ์

นิยามที่เกี่ยวข้อง

สายอักขระ S สามารถแทนได้ด้วย $S = x_1 x_2 x_3 ... x_l$ ความยาวของ S เขียนแทนได้ด้วย $|S|$ มีค่าเท่ากับ l สายอักขระย่อยของ S แทนด้วย $S_{i..j} = x_i x_{i+1} x_{i+2} ... x_j$ เมื่อ $i, j \in l^+$ ความยาวของสายอักขระย่อยแทนด้วย $|S_{i..j}|$ สามารถคำนวณได้จาก $(j - i) + 1$ เช่น $|S_{i..j}|$ มีค่าเท่ากับ 3 *คำเติมท้าย* (suffix) ของ S แทนได้ด้วย $S_{i..l}$ เป็นสายอักขระย่อยของ S ที่เริ่มจากตำแหน่ง i ไปจนถึง l เช่น ถ้า $S = TGCGA$ แล้ว คำเติมท้ายของ S ที่เป็นไปได้ทั้งหมดได้แก่ {ε, *A*, *GA*, *CGA*, *GCGA*, *TGCGA*} และ เซตของคำเติมท้ายแท้ (proper suffix) ของ S หมายถึงคำเติมท้ายทุกตัวยกเว้น S ได้แก่ {ε, *A*, *GA*, *CGA*, *GCGA* } *คำนำหน้า* (prefix) ของ S แทนได้ด้วย $S_{1..j}$ เป็นสายอักขระย่อยของ S ที่เริ่มจากตำแหน่ง 1 ไปจนถึง j เช่น ถ้า $S = TGCGA$ แล้ว คำนำหน้าของ S ที่เป็นไปได้ทั้งหมดได้แก่ {ε, *T*, *TG*, *TGC*, *TGCG*, *TGCGA*} และเซตของคำนำหน้าแท้ (proper prefix) ของ S หมายถึงคำนำหน้าทุกตัวยกเว้น S ได้แก่ {ε, *T*, *TG*, *TGC*, *TGCG*} *สายอักขระเกิดซ้ำ* (repeating substring) หมายถึง สายอักขระย่อย $S_{i..j}$ ใดๆ ที่ปรากฏขึ้นมากกกว่า 1 ครั้งบนสายอักขระ S ตัวอย่างเช่น $S = CACGTGCGA$ แล้วสายอักขระเกิดซ้ำได้แก่ {ε, *A*, *C*, *G*, *T*, *CG*} เพื่อความสะดวกในการพิจารณา กำหนดให้สายอักขระที่มีความยาวน้อยกว่า 2 ตัวอักษรเป็นสายอักขระเกิดซ้ำทั้งหมด ไม่ว่าจะปรากฏกี่ครั้งก็ตาม*สายอักขระเอกลักษณ์* (unique substring) หมายถึง สายอักขระย่อย $S_{i..j}$ ใดๆ ที่ปรากฏขึ้นเพียง 1 ครั้งบนสายอักขระ S ตัวอย่างเช่น $S = CACGTGCGA$ แล้วสายอักขระเอกลักษณ์ได้แก่ *AC*, *CA*, *GA*, *GC*, *GT*, *ACG*, *GCG*, *CGT*, *GTG*, *TGC*,...เป็นต้น สายอักขระเอกลักษณ์สั้นสุด (minimum unique substring) หมายถึง สายอักขระเอกลักษณ์ที่สั้นที่สุดเท่าที่ เป็นไปได้ ซึ่งหากพิจารณาสายอักขระย่อยที่สั้นกว่านั้นจะกลายเป็นสายอักขระเกิดซ้ำ กล่าวได้ว่าหากมีสายอักขระ เอกลักษณ์สั้นสุด $S_{i..j}$ แล้วจะมี $S_{i+1..j}$ และ $S_{i..j-1}$ เป็นสายอักขระเกิดซ้ำ ยกตัวอย่างเช่น $S = CACGTGCGA$

สายอักขระเอกลักษณ์สั้นสุดที่เป็นไปได้ทั้งหมด คือ *AC*, *CA*, *GA*, *GC*, *GT*, *TG* สังเกตได้ว่า สายอักขระย่อย ของคำเหล่านี้ที่สั้นกว่า 1 ตัวอักษรจะเป็นสายอักขระเกิดซ้ำ*สายอักขระเอกลักษณ์ที่เหมาะสม* (proper unique substring) หมายถึง สายอักขระเอกลักษณ์ที่มีคำ นำหน้าเป็นสายอักขระเอกลักษณ์สั้นสุดและไม่มีคำเติมท้ายแท้เป็นสายอักขระเอกลักษณ์สั้นสุดตัวอื่นอีก ยกตัวอย่าง S = *CACGTGCGA* สายอักขระเอกลักษณ์ที่เหมาะสมที่เป็นไปได้ทั้งหมดได้แก่ *AC*, *CA*, *GA*, *GC*, *GT*, *TG*, *ACG*, *GCG* แต่ *GTG* ไม่เป็นเพราะ มี TG เป็นคำเติมท้าย ข้อสังเกตคือ สายอักขระสั้นสุดทุกตัวเป็นสายอักขระเอกลักษณ์ ที่เหมาะสม

แบบจำลองที่นำเสนอ

แบบจำลองที่นำเสนอในงานวิจัยนี้ เรียกว่า *ออโตมาตาจำกัดเชิงน่าจะเป็นของสายอักขระเอกลักษณ์เหมาะสม* (proper unique substrings finite automata: PUSFA) ซึ่งเรียกโดยย่อว่า พัสฟา มีองค์ประกอบ 7 องค์ประกอบ ได้แก่ M = (Q, R, U, Σ, δ, τ, ρ) โดยมีรายละเอียดดังนี้ Q คือ เซตของสถานะ (state) โดยแต่ละสถานะจะมีฉลากชื่อกำกับไว้ ซึ่งจะใช้แทนสายอักขระข้อมูลที่ได้ จากการเรียนรู้ ประกอบไปด้วย สายอักขระเกิดซ้ำ สายอักขระเอกลักษณ์เหมาะสม R คือ เซตของสถานะที่เป็นสายอักขระเกิดซ้ำ (repeated substrings) และ $R \subset Q$ U คือ เซตของสถานะที่เป็นสายอักขระเอกลักษณ์ที่เหมาะสม (proper unique substrings) และ $U \subset Q$ Σ คือ เซตจำกัดของสัญลักษณ์อินพุต (input alphabet) δ คือ ฟังก์ชันเปลี่ยนสถานะ (transition function) ซึ่งโดยเปลี่ยนสถานะและอินพุตไปเป็นสถานะปลายทาง แสดงได้ด้วยความสัมพันธ์ δ: (Q, Σ) $\Rightarrow$ Q τ คือ ฟังก์ชันความถี่ (frequency function) จะเป็นฟังก์ชันที่เปลี่ยนจากสถานะและอินพุตไปเป็นจำนวนเต็ม แสดงได้ด้วยความสัมพันธ์ τ: (Q, Σ) $\Rightarrow$ I^+ ρ คือ ฟังก์ชันสายอักขระเอกลักษณ์เหมาะสมก่อนหน้า (previous proper unique substring function) ทำหน้าที่คืนสถานะเอกลักษณ์เหมาะสมที่มีความยาวมากที่สุด ที่มีเส้นทางมาถึงสถานะเอกลักษณ์สั้นสุดที่สนใจ แสดงด้วยความสัมพันธ์ ρ(U) $\Rightarrow$ $U \cup \{\varepsilon\}$ จากรูปที่ 1 สถานะ *GT* จะมีสถานะเอกลักษณ์เหมาะสมก่อนหน้า ที่ยาวที่สุดที่มีเส้นทางมาถึง คือ *ACG* ตัวอย่างอื่นเช่น ρ(*GC*) $\Rightarrow$ *TG* แบบจำลองนี้สามารถแทนได้ด้วยตารางตามตัวอย่างในรูปที่ 1

ขั้นตอนวิธีสร้างแบบจำลอง

กำหนดให้ S คือข้อมูลที่นำมาเรียนรู้เพื่อสร้างแบบจำลอง และสมมติให้ $S = x_1 x_2 x_3...x_l$ แล้วหลักการสร้างแบบจำลองอธิบายโดยคร่าว คือ อ่านสายอักขระอินพุตแต่ละตัวจาก x_1 จนถึง x_l ในขณะที่อ่าน x_i แต่ละตัวจะมีสถานะ Q_j หลายตัวที่ต้องพิจารณาอยู่ในคิวซึ่งเป็นเซตของคำเติมท้ายของตัวที่ยาวที่สุดเรียงคิวตามลำดับน้อยไปมาก เริ่มต้นให้ Q_j มีตัวเดียวคือ ε หากมีเส้นทางจาก Q_j ไปด้วย x_i อยู่แล้วให้เพิ่มความถี่ แต่หากยังไม่มีเส้นทาง ให้สร้างสถานะใหม่ โดยใช้ชื่อใหม่ด้วย Q_j ต่อกันกับ x_i แล้วตั้งเป็นสถานะปลายทางร่วมเพื่อสร้างเส้นทางของ Q_j ตัวที่ยาวกว่าที่เหลือในคิวชี้เส้นทางมายังสถานะปลายทางร่วมนี้ ซึ่งทุกครั้งที่สร้างสถานะใหม่ นั้นหมายความว่ามีสายอักขระเอกลักษณ์ปรากฏขึ้น เนื่องจากยังไม่มีเส้นทางมาก่อน สายอักขระเอกลักษณ์นั้นจะเป็นสายอักขระเอกลักษณ์สั้นสุด สายอักขระ เอกลักษณ์สั้นสุดจะมีข้อมูลสายอักขระเอกลักษณ์ก่อนหน้าเก็บไว้ด้วยฟังก์ชัน ρ เพื่อใช้ปรับเปลี่ยน ในกรณีที่ สายอักขระเอกลักษณ์สั้นสุดนั้นเปลี่ยนเป็นสายอักขระเกิดซ้ำ ดูรายละเอียดในขั้นตอนวิธีที่ 1 ในกรณีที่มีเส้นทางปรากฏอยู่แล้วและปลายทางเป็นสถานะเอกลักษณ์มาก่อน หากเพิ่มความถี่แล้วจะต้อง พิจารณาปรับเปลี่ยนปลายทางให้เป็นสถานะเกิดซ้ำด้วย และเมื่อปรับสถานะบางตัวในคิวให้เป็นสถานะเกิดซ้ำแล้ว เส้นทางของสถานะอื่นที่เคยมีเส้นทางมาที่สถานะตัวนี้ร่วมกันจะใช้เป็นปลายทางร่วมไม่ได้อีกต่อไป เพราะถูกเปลี่ยน เป็นสายอักขระเกิดซ้ำแล้ว จึงต้องขยายสถานะอื่นๆเหล่านั้นยาวออกไป โดยจะต้องเปลี่ยนเส้นทางใหม่ด้วยการสร้าง สถานะขึ้นมาใหม่ แล้วให้ตัวใหม่นี้จะถูกพิจารณาให้เป็นสถานะปลายทางร่วมสำหรับตัวอื่นๆที่ยาวกว่าต่อไป ซึ่งมีรายละเอียดค่อนข้างมากและซับซ้อน ดูรายละเอียดขั้นตอนวิธีในขั้นตอนวิธีที่ 1 และฟังก์ชันแตกสถานะเพิ่มใน ขั้นตอนวิธีที่ 2

ขั้นตอนวิธี 1 การสร้าง PUSFA

Input: input string S **Output:** Probabilistic Unique Substring Finite Automata M

for x_i **in** S

{ **for** Q_j **in** *activeQueue*

{ **if** (τ (Q_j, x_i) == 0)

{ **if** *target* == *null*

{ $newQ = Q_j + x_i$, τ (Q_j, x_i) = 1, δ(Q_j, x_i) = $newQ$

if (|$newQ$| >= 2)

target = $newQ$, **add** $newQ$ **to** U

add $newQ$ **to** *nextActiveQueue* }

else

{ **if** ρ(*target*) **is a suffix of** Q_j **then**

{ τ (Q_j, x_i) = 1, δ(Q_j, x_i) = *target*, ρ(*target*) = Q_j}

else

{ $newQ = Q_j + x_i$, **add** $newQ$ **to** U, τ (Q_j, x_i) = 1, δ(Q_j, x_i) = $newQ$

splitNode (*target*, Q_j, x_i)

target = $newQ$ }

add *target* **to** *nextActiveQueue* } }

else

{ $nextState$ = δ(Q_j, x_i), τ ($nextState$, x_i) +=1

if $nextState \in U$ **and** ρ($nextState$) $\neq$ *NULL*

target = $nextState$

if τ ($nextState$, x_i) > 1

$nextState \in R$

add $nextState$ **to** *nextActiveQueue* }}

copy *nextActiveQueue* to *activeQueue* }}

ขั้นตอนวิธี 2 ฟังก์ชันแตกสถานะเพิ่ม (Split Node)

```
function splitNode(splitState, activeState, x_i )
{   newTarget = NULL, splitState ∈ R, add splitState to nextActiveQueue
    previousState = ρ(splitState)
    for each Q_k where Q_k is a suffix of previousState
    {   if |Q_k| >= | splitState |
        {   if newTarget == NULL
            {   newNode = Q_k + x_i, δ(Q_k, x_i) = newNode, newNode ∈ U
                if Q_k is a suffix of activeState
                {   add newNode to nextActiveQueue, newNode ∈ R }
                if |Q_k| == |activeState|
                {   newTarget = newNode }
                for c in Σ
                {   if τ (splitState, x_i) > 0
                    {   nextOfSplitState = δ(splitState, c), τ (newNode, c) = τ (splitState, c),
                            ρ(nextSplitState) = newNode
                            δ(newNode, c) = nextOfSplitState } }
            else
            {   δ(Q_k, x_i) = newTarget, newTarget ∈ U
                ρ (newTarget) = Q_k } } } }
```

การประยุกต์ใช้จำแนกข้อมูล

การจำแนกข้อมูลในที่นี้หมายถึง ปัญหาที่ว่า มีกลุ่มข้อมูล 2 กลุ่ม สมมติเป็น กลุ่ม *A* และ กลุ่ม *B* แล้วสายอักขระอินพุต *S* ที่สนใจนั้นน่าจะเป็นสมาชิกของกลุ่มข้อมูลใดมากกว่ากัน เราเลือกใช้วิธีตัวจำแนกประเภท แบบเบย์อย่างง่าย (Naive Bayes Classifier) ซึ่งใช้กันอย่างแพร่หลายและมีประสิทธิภาพดีในการจำแนกภาษา (Rish, 2001) (Ting et al., 2011) ตัวจำแนกจะตัดสินใจว่าเมื่อใช้แบบจำลอง *M* เรียนรู้ข้อมูล *A* และ *B* แล้ว *S* จะอยู่ในกลุ่มใด โดยคำนวณ *S* จากแบบจำลอง เราประยุกต์วิธีการดังกล่าวกับแบบจำลองจากงานวิจัยนี้ด้วยการสร้างแบบจำลองพัสฟาขึ้นมา 2 ตัว ตัวแรกสร้างขึ้นจากข้อมูลกลุ่ม *A* สมมติแทนได้ด้วย M_A และอีกตัวหนึ่งสร้างขึ้นจากข้อมูลกลุ่ม *B* แทนด้วย M_B จากนั้นจะใช้ M_A อ่านอินพุต *S* เพื่อคำนวณหาความน่าจะเป็นของการเกิดสายอักขระ *S* แทนด้วย $P(S \mid M_A)$ และใช้ M_B อ่านอินพุต *S* เพื่อคำนวณความน่าจะเป็นของการเกิดสายอักขระ *S* แทนด้วย $P(S \mid M_B)$ ตามหลักการในการ จำแนกสายอักขระใดๆเราจะพิจารณาว่า *S* อยู่ในกลุ่ม *A* หรือ *B* โดยพิจารณาจากสมมติฐานที่มีความน่าจะเป็น มากที่สุด (Maximum a posteriori hypothesis) ซึ่งได้สมการในการจำแนกดังนี้

$$l_{max} = \underset{i \in \{A,B\}}{argmax} P(M_i) \cdot P(S|M_i) \quad (1)$$

จากสมการที่ (1) l_{max} หมายถึงกลุ่มข้อมูลที่เป็นคำตอบอาจจะเป็น *A* หรือ *B* ซึ่งจะได้มาจากการเลือกค่า M_A หรือ M_B ที่ทำให้ได้ค่าความน่าจะเป็นตามสมการที่มีค่ามากที่สุด ก็จะเป็นคำตอบว่า *S* ควรอยู่ในกลุ่มใด ปัญหาต่อมา คือ จะมีวิธีการคำนวณ $P(S \mid M)$ ได้อย่างไร

วิธีการคำนวณความน่าจะเป็น

กำหนดให้สายอักขระย่อยของ S ที่ต้องการหาความน่าจะเป็นคือ $x_{1...i} = x_1 x_2 x_3...x_i$ แล้ว อักขระตำแหน่งที่ i แทนด้วย x_i และความน่าจะเป็นของการเกิด x_i คือ $P(x_i | x_1, x_2, x_3,...,x_{i-2}, x_{i-1})$ สามารถคำนวณได้ตามหลักการของ ลูกโซ่มาร์คอฟ แต่เนื่องจากการพิจารณาความน่าจะเป็น ด้วยเงื่อนไขจาก x_{i-1} ย้อนกลับไปถึง x_1 สำหรับข้อมูลที่มี ขนาดยาวมากนั้นจะทำให้ไม่สามารถจัดการได้ การคำนวณในงานวิจัยนี้จึงพิจารณาด้วยเงื่อนไขเพียง x_{i-1} ไปถึง x_{i-k} เมื่อ $x_{i-1...i-k}$ เป็นสายอักขระเอกลักษณ์ที่ปรากฏบนสถานะของแบบจำลอง ซึ่งเพียงพอต่อการคำนวณความน่าจะเป็น ที่ถูกต้องดังที่ได้อธิบายเหตุผลก่อนหน้า ดังนั้นความน่าจะเป็นในการเกิด x_i คือ $P(x_i | Y)$ เมื่อ Y คือป้ายชื่อของ สถานะปลายทางที่ได้จากฟังก์ชันการใช้ δ เพื่อหาสถานะปลายทางจากอินพุต $x_1, x_2, x_3,...,x_{i-2}, x_{i-1}$ ไปตามลำดับ และ $P(x_i | Y)$ คำนวณได้จาก

$$P(x_i|Y) = \frac{\tau(Y,x_i)}{\sum_{c\in\Sigma}^{n}\tau(Y,c)} \quad (2)$$

จากสมการที่ (2) $\tau(Y, c)$ คือ ฟังก์ชันความถี่ของการเปลี่ยนสถานะด้วยอักขระ c ที่สถานะ Y เช่น จากรูปที่ 1 $\tau(CG, T)$ จะมีค่าเท่ากับ 1 เป็นต้น และผลรวมของความน่าจะเป็นทุกตัวอักษรคือ $\tau(CG, T)$+ $\tau(CG, A) = 2$ ดังนั้น $P(A|CG) = 1/2$. และความน่าจะเป็นของการเกิดสายอักขระ $x_{1...i} = x_1 x_2 x_3...x_i$ สามารถหาได้จากกฎลูกโซ่

$$P(S|M) = P(x_1|\varepsilon) \cdot P(x_2|x_1) \cdot \ldots \cdot P(x_{i-1}|x_1 x_2 \ldots x_{i-2}) \cdot P(x_i|Y) \quad (3)$$

ตัวอย่างการคำนวณความน่าจะเป็นของ $S = CGTGC$ จากเครื่องจักรในรูปที่ 1 จะได้ว่า $P(CGTGC | M) = P(C | \boldsymbol{\varepsilon})\cdot P(G | C) \cdot P(T | CG)\cdot P(G | GT)\cdot P(C | TG) = 3/9\cdot 2/3\cdot 1/2\cdot 1\cdot 1 = 1/9$ อย่างไรก็ดี ในการคำนวณความน่าจะเป็นบางครั้ง $\tau(Y,x_i) = 0$ ยกตัวอย่าง $\tau(TG, T)$ ในรูปที่ 1 ซึ่งจะ ทำให้การคำนวณ $P(S | M)$ มีค่าเป็น 0 วิธีการแก้ปัญหานี้วิธีหนึ่งที่ง่าย เหมาะสำหรับแบบจำลองภาษา คือ การปรับเรียบแบบแบคออฟ (back off smoothing) (Chen, 1999) โดยมีหลักการว่า ถ้ามองเหตุย้อนกลับไปไกลเกิน ก็มองกลับไปให้สั้นลง ดังนั้นเมื่อมองเงื่อนไขสั้นลงในเส้นทางเดิมก็จะทำให้ได้ความน่าจะเป็นที่ไม่เป็น 0 เราประยุกต์ใช้ เทคนิคเดียวกับการแบคออฟ (back off) โดยใช้คำเติมท้ายที่สั้นลงของคำที่มีปัญหา ซึ่งสามารถให้ ความน่าจะเป็นที่ดีที่สุด เป็นเงื่อนไขแทนคำเดิม ดังนั้นในกรณีที่เป็น $\tau(Y, x_i) = 0$ จะสามารถคำนวณความน่าจะเป็นใหม่ได้จาก สถานะ Z ซึ่งเป็นคำเติมท้ายที่ยาวที่สุดของ Y ที่ทำให้ $\tau(Y, x_i) \neq 0$ ซึ่งจะใช้สมการที่ (4) แทนสมการที่ (2) ได้ดังนี้

$$P(x_i|Y) = \underset{Z\in suffix(Y)}{argmax} \frac{\tau(Z,x_i)}{\sum_{c\in\Sigma}^{n}\tau(Z,c)} \quad (4)$$

ตัวอย่างการคำนวณความน่าจะเป็นของสายอักขระ S ด้วยสมการที่ 4 เช่น ให้ $S = ACGCTG$ แล้วจะได้ $P(ACGCTG | M) = P(A | \boldsymbol{\varepsilon})\cdot P(C | A) \cdot P(G | AC) \cdot P(C | G) \cdot P(T | \boldsymbol{\varepsilon})\cdot P(G | T) = 2/9\cdot 1\ 1\cdot 1/3\cdot 1/9\cdot 1 = 2/243$ โดยพจน์ที่ขีดเส้นใต้คือ การคำนวณโดยใช้คำเติมท้ายด้วยสมการที่ (4)

การทดลอง

ในงานวิจัยนี้สนใจว่าแบบจำลองจะสามารถนำเอาไปใช้งานเพื่อจำแนกข้อมูลได้แม่นยำขึ้นเมื่อเปรียบเทียบกับวิธีการมาตรฐานที่ตัดคำสมมติด้วยความยาวคงที่หรือไม่ และในการนำแบบจำลองไปเรียนรู้ข้อมูลนั้นแบบจำลอง จะเติบโตอย่างไรและใช้เวลาในการเรียนรู้เป็นอย่างไรเมื่อเทียบกับความยาวข้อมูล เพื่อทำให้เชื่อมั่นได้ว่าจะสามารถนำแบบจำลองนี้ไปเรียนรู้กับข้อมูลที่มีขนาดใหญ่ได้ เราจึงได้ทำการทดลอง 2 ส่วนคือ 1. ทดสอบความแม่นยำในการจำแนกข้อมูล และ 2. ทดสอบการเติบโตและเวลาที่ใช้ในการเรียนรู้ของแบบจำลองเมื่อเทียบกับความยาวข้อมูล การทดลองเพื่อทดสอบความแม่นยำในการจำแนกข้อมูลนั้น จำเป็นจะต้องเปรียบเทียบประสิทธิภาพกับงานวิจัยอื่นๆ ภายใต้สถานการณ์และข้อมูลชุดเดียวกัน โดยงานวิจัยนี้ได้เลือกใช้ข้อมูลดีเอ็นเอของ แบคทีเรีย อี.โคไล ซึ่งเป็นข้อมูลมาตรฐานในการทดสอบประสิทธิภาพแบบจำลอง (Dua and Graff, 2019) โดยแบ่งกลุ่มข้อมูลดีเอ็นเอจากชุดข้อมูลตัวอย่าง 2 กลุ่ม กลุ่มแรกเป็นข้อมูลดีเอ็นเอที่เป็นตัวสนับสนุน (promoters) และอีกกลุ่มหนึ่ง เป็นดีเอ็นเอที่ไม่เป็น ตัวสนับสนุน (non-promoters) โดยตัวสนับสนุนจะเป็นส่วนหนึ่งของดีเอ็นเอ ที่อยู่ส่วนต้นของยีน มีรูปแบบการจัดเรียง นิวคลีโอไทด์เฉพาะตัว ซึ่งถูกนำไปใช้ประโยชน์ทางการวิเคราะห์ดีเอ็นเอ การทดลอง

นี้จะนำข้อมูลดีเอ็นเอทั้งสองกลุ่ม มาสร้างแบบจำลองเพื่อรู้จำทั้งส่วนที่เป็นตัวสนับสนุนและส่วนที่ไม่เป็น แล้วนำแบบจำลองที่ได้นั้นไปทดสอบ จำแนกส่วนของดีเอ็นเอที่ไม่เคยพบมาก่อนว่าจะเป็นตัวสนับสนุนหรือไม่ ข้อมูลดีเอ็นเอของแบคทีเรียจะมี 2 กลุ่มคือกลุ่มตัว 53 สาย มีความยาวสายละ 57 ตัวอักษร รวม 2 กลุ่มมีจำนวน 106 สาย การแบ่งข้อมูลเพื่อใช้สำหรับทดสอบนั้น จะใช้วิธีการแบ่งข้อมูลเช่นเดียวกับวิธีการมาตรฐานอื่นที่ได้ใช้ข้อมูลเดียวกันนี้ในการทดสอบ เพื่อเปรียบเทียบกัน (Towell et al., 1990) (Xing et al., 2008) โดยใช้หลักการเอาออกหนึ่งสาย (leave one out) เพื่อนำข้อมูลออกจากกลุ่ม ที่ใช้สร้างแบบจำลอง ให้เป็นตัวทดสอบที่แบบจำลองไม่เคยพบมาก่อน เช่น ข้อมูลตัวสนับสนุน 53 สาย เอาออก 1 สายจะเหลือข้อมูลที่นำไปสร้างแบบจำลอง 52 สาย โดยทำซ้ำจำนวน 53 ครั้ง และอีกกลุ่มหนึ่งคือชุดที่ไม่เป็น ตัวสนับสนุนก็ทำการสร้างแบบจำลองด้วยข้อมูล 53 สาย และทำในวิธีการเช่นเดียวกันนี้ แต่เปลี่ยนชุดที่ดึงออก ทำเช่นนี้จนครบ 53 สาย ข้อมูลแต่ละสายที่ดึงออกมาเพียงสายเดียวในแต่ละครั้งนั้น จะถูกนำมาทดสอบด้วย แบบจำลองว่า มีความน่าจะเป็นเท่าไร และมีความน่าจะเป็น ในกลุ่มใดมากกว่ากัน โดยได้ผลการทดลองดังนี้

ผลการวิจัย

การทดลองได้ผลลัพธ์จากการดึงข้อมูลออกจำนวน 106 ตัวแบ่งเป็นกลุ่มตัวสนับสนุน 53 สาย และกลุ่มไม่เป็นตัวสนับสนุน 53 สาย พบว่าแบบจำลองใน

งานวิจัยนี้ทำนายกลุ่มตัวสนับสนุนผิดพลาดเพียง 4 สายจาก 53 สาย โดยแบบจำลองได้ทำนายสายดีเอ็นเอที่เป็นตัวสนับสนุนว่าไม่เป็น ซึ่งถือเป็นผลลบลวง (false negative) และกลุ่มที่ไม่เป็นตัวสนับสนุนนั้นแบบจำลองสามารถจำแนกกลุ่มได้ถูกต้องครบทุกตัว คิดเป็นความถูกต้องได้ถึง 102 สายจาก 106 สาย ผิดพลาดเพียง 4 สายจาก 106 สาย ซึ่งถูกต้องแม่นยำถึงร้อยละ 96.23 หรือผิดพลาดเพียงร้อยละ 3.77

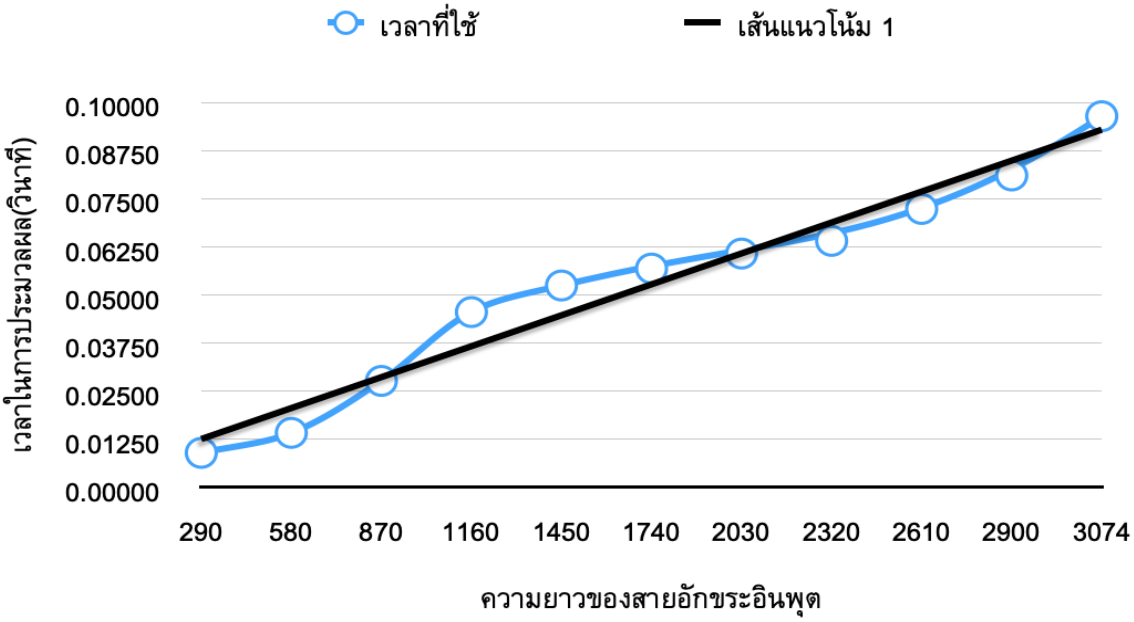
นอกจากนี้งานวิจัยนี้ได้นำไปเปรียบเทียบกับผลลัพธ์จากวิจัยอื่นๆ ที่เป็นมาตรฐานซึ่งใช้ข้อมูลชุดเดียวกันและ วิธีการทดสอบแบบเดียวกันได้แก่ 1. วิธีการโครงข่ายประสาทเทียมแบบฐานความรู้ (knowledge based neural network) หรือ KBANN (Towell et al., 1990) 2. วิธีการโครงข่ายประสาทเทียมมาตรฐาน (standard neural network) แบบส่งค่าย้อนกลับ (back propagation) หรือ BP (Rumelhart et al., 1986) 3. วิธีการกฎการจำแนกลำดับ (sequential classification rule:SCR) และต้นไม้ตัดสินใจสำหรับลำดับแบบทั่วไป (generalized sequential decision tree: GSDT) (Xing et al., 2008) 4. วิธีการเฉพาะทางในทางชีวสารสนเทศของโอนีล (O'neil, 1989) 5. วิธีการต้นไม้ตัดสินใจ (decision tree) เรียกโดยย่อ ID3 (Mingers, 1989) และวิธีการเอ็นแกรมขนาด 2, 3, 4, 8 แกรม ซึ่งได้ผลดังตารางที่ 1 แสดงให้เห็นว่า เมื่อใช้การแบ่งคำโดยอัตโนมัติเพื่อคำนวณความน่าจะเป็นตามงานวิจัยนี้จะมีความแม่นยำสูงกว่าวิธีการที่ใช้การแบ่งคำสมมติด้วยความยาวคงที่

ตารางที่ 1 จำนวนครั้งของความผิดพลาดในการจำแนกข้อมูล

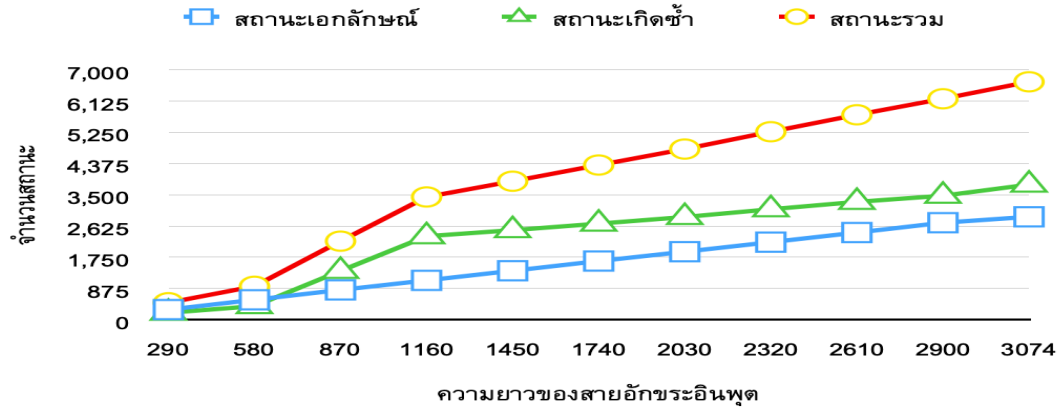
วิธีการที่ใช้จำแนกกลุ่มข้อมูล	จำนวนข้อมูลที่จำแนกผิดพลาด n/106
KBANN	4
BP	8
O'neil	12
ID3	21
SCR	7
GSDT	10
2-gram	18
3-gram	13
4-gram	19
8-gram	8
PUSFA	4

ความซับซ้อนเชิงเวลาและหน่วยความจำในการทำงาน

การทดลองในส่วนที่ 2 เพื่อดูว่า ขนาดของแบบจำลองและเวลาที่ใช้ในการสร้างแบบจำลองนั้นจะเติบโตอย่างไร เพื่อยืนยันว่าเมื่อนำวิธีการสร้างแบบจำลองนี้ไปใช้งานกับข้อมูลขนาดใหญ่จะยังคงจัดการได้ในทางปฏิบัติ โดยเราทำการวิเคราะห์ในทางทฤษฎีได้ว่า การพัฒนาโปรแกรมเพื่อสร้างแบบจำลองตามขั้นตอนวิธี ในงานวิจัยนี้ มีการวนซ้ำเพื่ออ่านสายอักขระเรียนรู้สมมติให้มีความยาว n และในขณะที่อ่านอักขระแต่ละตัว จะต้องพิจารณา แถวคอยขนาดเท่ากับสายเอกลักษณ์เหมาะสมแต่ละคำสมมติให้เป็น m ซึ่งมีจำนวนตัวแปรผันได้ขึ้นอยู่กับสภาพข้อมูล อย่างไรก็ตามค่า m จะมีค่าไม่เกิน n ดังนั้นความซับซ้อนของขั้นตอนวิธีสามารถคำนวณได้ด้วย m n ซึ่งจะได้ $O(n^2)$ อย่างไรก็ดีพบว่าในกรณีที่ดีที่สุดในกรณี m มีค่าเท่ากับ 2 และหากค่า m มีค่าไม่เกิน k แล้วจะทำให้กรณีที่ดีที่สุดเป็น $O(n)$ ในทางปฏิบัติเนื่องจาก ข้อมูลที่นำมาเรียนรู้นั้นจะมีค่า m ไม่เกินความยาว 57 ตัวอักษรตามรูปแบบข้อมูล ดังนั้นในทางปฏิบัติแล้ว การเติบโตของเวลาก็ควรเป็น $O(n)$ สำหรับขนาดของแบบจำลองก็สามารถประมาณได้โดย หากว่าสายอักขระเอกลักษณ์มีความยาวรวม n จะมีสายอักขระย่อยที่เป็นไปได้ทั้งหมด เท่ากับ $\sum_{i=1}^{n} i$ ซึ่งประมาณได้เป็น $O(n^2)$ แต่หากข้อมูลสายอักขระเกิดซ้ำ มีความยาวไม่เกิน k จะมีสายอักขระย่อยประมาณ k^2 ตัว โดยจะมีจำนวนทั้งหมดแบ่งเป็น n/k จำนวน ดังนั้นจะมีจำนวนสายอักขระย่อยทั้งหมด $k^2 \cdot n/k = kn$ ซึ่งมีขอบเขตเป็น $O(n)$ ซึ่งค่า k ในทางปฏิบัติสำหรับการทดลองนี้ จะมีค่าเท่ากับ 57 ตัวอักษร งานวิจัยนี้ได้ทำการทดลอง เพื่อดูว่าในทางปฏิบัติแล้วจะใช้เวลา และ หน่วยความจำที่ใช้สอดคล้องกับทางทฤษฎีหรือไม่ เราใช้เครื่องคอมพิวเตอร์ระบบปฏิบัติการ macOS v.10.x หน่วยประมวลผล 1.3 GHz Dual-Core Intel Core i5 ทำการสร้างแบบจำลองด้วยข้อมูลเดียวกันกับการทดลองแรก แล้ววัดเวลาและจำนวนสถานะที่ใช้ในแต่ละช่วงที่อ่านข้อมูลแต่ละตัว พบว่ามีการใช้เวลาในการทำงานตามรูปที่ 2 ซึ่งมีแนวโน้มการเติบโตของฟังก์ชันเวลาเป็นเชิงเส้น และหน่วยความจำที่ใช้วัดจากจำนวนสถานะที่สร้างขึ้นตามรูปที่ 3 ซึ่งมีแนวโน้มการเติบโตของฟังก์ชันพื้นที่เป็นเชิงเส้นเช่นกันซึ่งตรงกับที่วิเคราะห์ทางทฤษฎี ที่มีการเติบโตของเวลาและขนาดเป็น $O(n)$



รูปที่ 2 กราฟแสดงเวลาที่ใช้ในการดำเนินงานเมื่อเทียบกับความยาวของอินพุต



รูปที่ 3 กราฟแสดงจำนวนสถานะที่ใช้ในการดำเนินงานเมื่อเทียบกับความยาวของอินพุต

สรุปและวิจารณ์ผล

งานวิจัยนี้ได้นำเสนอแบบจำลองภาษาเพื่อเรียนรู้สายอักขระหรือลำดับสัญลักษณ์ โดยไม่ต้องแบ่งคำสมมติ ด้วยความยาวคงที่ในกรณีที่สายอักขระมีความยาวมาก โดยสามารถเรียนรู้ส่วนเพิ่มได้ไม่จำกัดโดยไม่ต้องปรับขนาดคำสมมติ โดยในงานวิจัยนี้จะใช้การตัดคำด้วยสายอักขระเอกลักษณ์ที่เหมาะสม และเก็บสายอักขระเกิดซ้ำ ทุกตัวที่เป็นไปได้ เพื่อให้ได้ค่าทางสถิติที่ใช้คำนวณความน่าจะเป็นได้แม่นยำยิ่งขึ้น งานวิจัยนี้ได้เสนอวิธีการนำแบบจำลองนี้ ไปประยุกต์ใช้ในการจำแนกข้อมูลที่มีลักษณะความยาวมากและไม่มี เห็นว่าแบบจำลองนี้สามารถประยุกต์ใช้กับข้อมูลขนาดใหญ่ได้ แบบจำลองในงานวิจัยนี้จึงเหมาะสมกับข้อมูลที่มีลักษณะดังกล่าวมา อย่างไรก็ดี แบบจำลองนี้ยังการแบ่งคำ เช่น ข้อมูลสายอักขระดีเอ็นเอ เป็นต้น ผลการทดลองแสดงให้เห็นว่า การจำแนกกลุ่มข้อมูลแบคทีเรีย อี.โคไล ด้วยแบบจำลองของงานวิจัยนี้ ทำได้แม่นยำกว่า แบบจำลองที่แบ่งคำด้วยความยาวคงที่อื่นๆ แต่เท่ากันกับวิธีการ KBANN ซึ่งมีการใช้ความรู้เกี่ยวกับกฎเกณฑ์ ความเป็นโปรโมเตอร์มาเสริมทำให้มีความแม่นยำมากกว่าวิธีการอื่นๆ ในขณะที่งานวิจัยนี้สามารถทำได้ โดยอัตโนมัติ ปราศจากการใช้ความรู้อื่นมาเสริม การทดลองเพื่อดูอัตราการเติบโตของเวลาและขนาดของแบบจำลองในขณะเรียนรู้เมื่อเทียบกับความยาวข้อมูลที่ใช้เรียนรู้ พบว่าเติบโตแบบเชิงเส้น แสดงให้ต้องการทฤษฎีเกี่ยวกับสายอักขระเอกลักษณ์เหมาะสมมาอธิบายสนับสนุน เช่น ดีกว่าแบบจำลอง เอ็นแกรม ทางทฤษฎีในขอบเขตใดบ้าง อีกทั้งยังต้องการขั้นตอน

วิธีสร้างแบบจำลองที่กระชับและง่ายกว่าเดิม เพื่อให้สามารถนำไปใช้งาน ได้ง่ายขึ้น รวมถึงการนำแบบจำลองไปประยุกต์ใช้กับการทำนายข้อมูลและจำแนกข้อมูลอื่นๆ ให้มีความแม่นยำยิ่งขึ้นต่อไป

กิติกรรมประกาศ

งานวิจัยนี้ได้รับทุนวิจัยจากมหาวิทยาลัยรามคำแหง ประจำปีงบประมาณ 2559 และได้ใช้ข้อมูลชุดสำหรับ ทดลองจากคลังข้อมูลการเรียนรู้เครื่องจักรของมหาวิทยาลัยแคลิฟอเนียเออร์ไวน์ (UCI machine learning repository) (Dua and Graff, 2019)

เอกสารอ้างอิง

Ade, R. R. and Deshmukh, P.R. 2013. Methods for Incremental Learning: A Survey. International Journal of Data Mining and Knowledge Management Process, 119-125.

Cavnar, W. B. and Trenkle, J. M. 1994. N-Gram-Based Text Categorization. In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, 161–175.

Chen, S.F. and Goodman, J., 1999. An empirical study of smoothing techniques for language modeling. Computer Speech and Language, 13(4), 359-394.

Dua, D. and Graff, C. 2019 UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Galata, A. Johnson, N. and Hogg, D. 2001. Learning Variable Length Markov Models of Behavior. Computer Vision and Image Understanding. 81(3): 398-413

Geppert, E. and Hammer, B. 2016. Incremental learning algorithms and applications. *ESANN*, 357-368

Ilie, L., Smyth, W. F. 2011. Minimum Unique Substrings and Maximum Repeats. Fundam. Inf, 110 (1-4),183 - 195.

Leslie, C., Eskin, E., and Noble, W. S. 2002. The spectrum kernel: a string kernel for SVM protein classification. PAC Symp Biocomput, 564-575.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. 2002. Text classification using string kernels. Journal of Machine Learning Research, 2, 419-444.

Mingers, J. 1989. An Empirical Comparison of Pruning Methods for Decision Tree Induction, Machine Learning 4 227–243.

O'Neill, Michael. 1989. Escherichia coli promoters. I. Consensus as it relates to spacing class, specificity, repeat substructure, and three-dimensional organization. The Journal of biological chemistry. 264. 5522-30.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77 (2), 257-286.

Rayan Chikhi, and Paul Medvedev. 2014. Informed and automated k-mer size selection for genome assembly, Bioinformatics, 30(1), 31-37.

Rish, I. 2001. An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence 3(22) 41-46.

Ron, D., Singer, Y., and Tishby, N. 1996. The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length. Machine learning, 117–149.

Rumelhart, D. E., G. Hinton, G. E., and Williams, R. J.1986 Learning Internal Representations by Error Propagation, in: Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations, D. E. Rumelhart and J. L. McClelland (Eds.), 318–363.

Ting, S. L., Ip, W. H., and Tsang, A. H. 2011. Is Naive Bayes a good classifier for document classification. International Journal of Software Engineering and Its Applications, 5(3), 37-46.

Towell, G., Shavlik, J., and Noordewier, M. 1990. Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks. In Proceedings of the Eighth National Conference on Artificial Intelligence, 861-866.

Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., and Carrasco, R. C. 2005. Probabilistic Finite-State Machines-Part I. IEEE Trans. Pattern Anal. Mach. Intell., 27 (7), 1013--1025.

Xing, Z., Pei, J., Dong, G., and Yu, P. S. 2008. Mining Sequence Classifiers for Early Prediction. Proceedings of the SIAM International Conference on Data Mining, 644--655.