อัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น

นายต่อศักดิ์ เพ็ญภินันท์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

Probabilistic Regular Grammar Inference Algorithm Using Incremental Technique

Mr. Torsak Penpinun

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2017

หัวข้อวิทยานิพนธ์	อัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น
โดย	นายต่อศักดิ์ เพ็ญภินันท์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร. สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ นครทิพย์ พร้อมพูล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์)

.....กรรมการภายนอกมหาวิทยาลัย

(รองศาสตราจารย์ ดร. อานนท์ รุ่งสว่าง)

ต่อศักดิ์ เพ็ญภินันท์ : อัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น (Probabilistic Regular Grammar Inference Algorithm Using Incremental Technique) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร. อรรถสิทธิ์ สุรฤกษ์, 45 หน้า.

การอนุมานไวยากรณ์เป็นเรื่องที่ได้รับการศึกษามาเป็นเวลานาน ซึ่งไวยากรณ์จะถูกแสดงในรูปของกฎการสร้างใหม่ ๆ พร้อมด้วยความน่าจะเป็นที่สนับสนุนกฎการสร้างไวยากรณ์นั้น งานวิจัยนี้สนใจในรูปแบบของไวยากรณ์ทั่วไปที่ได้รับการยอมรับผ่านเครื่องจักรแบบจำกัดสถานะ เทคนิคการอนุมานไวยากรณ์ที่ได้รับความนิยมในปัจจุบันคืออัลกอริทึมอัลเลอเจียร์ (Alergia) ซึ่งวิธีการคือสร้างเครื่องจักรแบบจำกัดสถานะเชิงความน่าจะเป็นจากตัวอย่างเชิงบวกพร้อมกับหาค่าความน่าจะเป็น ซึ่งงานวิจัยนี้นำเสนออัลกอริทึมการอนุมานไวยากรณ์เชิงความน่าจะเป็นจากการพิจารณาตัวอย่างเชิงบวกเริ่มต้นจากความยาวน้อยไปหาความยาวที่มากที่สุดตามลำดับ กำหนดรูปแบบให้กับไวยากรณ์ที่เกิดขึ้น และนำเสนอในรูปแบบของเครื่องจักรแบบจำลองสถานะเชิงความน่าจะเป็น

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2560

5770921521 : MAJOR SOFTWARE ENGINEERING
KEYWORDS: GRAMMAR INFERENCE / PROBABILISTIC FINITE STATE MACHINE / INCREMENTAL TECHNIQUE

TORSAK PENPINUN: Probabilistic Regular Grammar Inference Algorithm Using Incremental Technique. ADVISOR: ASST. PROF. ATHASIT SURARERKS, Ph.D., 45 pp.

Grammatical inference has been studied for a long time where grammar is illustrated by a collection of re-writing rules, together with their probabilities. We are interested in regular language model which can be recognized by a finite state machine. The most popular technique is an alergia algorithm. The objective is to construct a probabilistic finite state machine using only positive examples together with their probabilities (or frequency). In this work, we introduce a probabilistic grammatical inference algorithm in order to construct a finite state machine. The algorithm starts by considering the shortest positive example and generates two patterns of regular grammar rules (productions). Our experimental results show that the probabilities obtained from our probabilistic finite state machine can be more accurate than the one obtained from the alergia algorithm. Our algorithm is an alternative way for constructing a probabilistic finite state machine.

Department: Computer Engineering Student's Signature .....
Field of Study: Software Engineering Advisor's Signature .....
Academic Year: 2017

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สาเร็จลุล่วงได้ด้วยความกรุณาของอาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์ ได้สละเวลาให้ความรู้ คำปรึกษา คำแนะนำและตรวจทานแก้ไขข้อผิดพลาดต่างๆ ตลอดจนการกำกับดูแลและคอยติดตามความก้าวหน้า ทำให้การวิจัยนี้สาเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ผู้วิจัยใคร่ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์นครทิพย์ พร้อมพูล และ รองศาสตราจารย์ ดร.อานนท์ รุ่งสว่าง กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลา ให้คำแนะนา ตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ขอขอบพระคุณบิดา มารดา และญาติพี่น้องที่ให้การสนับสนุนและเป็นกำลังใจที่ดีให้เสมอมาและสนับสนุนด้านทุนทรัพย์ในการศึกษารวมไปถึงทุกท่านที่มีส่วนช่วยเหลือในการทำวิทยานิพนธ์ครั้งนี้ ซึ่งมิได้กล่าวนามในที่นี้

ท้ายที่สุด ผู้วิจัยขอขอบพระคุณเพื่อนร่วมงานทุกคน ที่คอยติดตามและให้กำลังใจ รวมถึงท่านอื่นๆ ที่มิได้กล่าวลงนามไว้ ณ ที่นี้ที่มีส่วนทำให้วิทยานิพนธ์สำเร็จลุล่วงไปได้ ด้วยดี ผู้วิจัยหวังเป็นอย่างยิ่งว่าวิทยานิพนธ์ฉบับนี้จะเป็นประโยชน์บ้างไม่มากก็น้อยสำหรับผู้ที่สนใจจะศึกษารายละเอียดต่อไป

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบันกระบวนการทางวิศวกรรมซอฟต์แวร์มีความสนใจในการนำเอาวิธีการรูปนัย (formal method) มาช่วยในการอธิบายกระบวนการทำงานของแบบจำลองทางด้านซอฟต์แวร์ กล่าวคือ ระบบของซอฟต์แวร์เมื่อทำงานจะมีกระบวนการเกิดขึ้นตามลำดับ (process of sequences) ลำดับขั้นตอนเหล่านี้เองจะเป็นตัวอธิบายระบบว่าทำงานอย่างไร ซึ่งวิธีการรูปนัยจะศึกษาเรียนรู้ลำดับที่เกิดขึ้นนี้ด้วยการบันทึก เก็บรวบรวมข้อมูลของลำดับการทำงานที่เกิดขึ้นจริงจากระบบและนำมาวิเคราะห์หาข้อกำหนดของการทำงานที่สอดคล้องกับพฤติกรรมที่เกิดขึ้น และสกัดหาแบบจำลองที่สามารถอธิบายเหตุการณ์ที่สามารถเกิดขึ้นได้ทั้งหมด ซึ่งวิธีการดังกล่าวนี้เรียกว่า กระบวนการอนุมานไวยากรณ์

กระบวนการพัฒนาซอฟต์แวร์หรือการพัฒนาระบบเรานิยมใช้หลักการทางวิศวกรรมซอฟต์แวร์เข้ามาช่วยให้ได้ผลลัพธ์บรรลุเป้าหมายของการพัฒนาอย่างมีประสิทธิภาพ อาศัยแนวคิดทฤษฎีทางวิทยาศาสตร์และคณิตศาสตร์มาใช้ประโยชน์ทางกระบวนการพิจารณา ซึ่งขั้นตอนในกระบวนการวิศวกรรมซอฟต์แวร์ที่สำคัญกระบวนการหนึ่งคือการออกแบบระบบหรือออกแบบพฤติกรรมการทำงานของซอฟต์แวร์ให้ตรงกับข้อกำหนด (specification) ดังนั้นการสร้างแบบจำลองกระบวนการพัฒนาระบบ (process model) จะเป็นสิ่งที่แสดงให้เห็นถึงการจัดการโครงสร้างลำดับขั้นตอนของกระบวนการทำงาน การอนุมานไวยากรณ์สามารถนำมาใช้ประโยชน์ในด้านวิศวกรรมซอฟต์แวร์เพื่อการวิเคราะห์แบบจำลองการทำงานระบบ ซึ่งการทำงานของระบบจะเป็นไปตามลำดับ เราศึกษาและเก็บรวบรวมกระบวนการที่สามารถเกิดขึ้นทั้งหมดของซอฟต์แวร์มาพิจารณากับข้อกำหนดในอัลกอริทึมและสร้างเป็นแบบจำลองเชิงความน่าจะเป็นเพื่อให้เราทราบโอกาสที่จะเกิดกระบวนการพร้อมค่าความน่าจะเป็น จากแบบจำลองที่สร้างนี้เองเราสามารถนำไปใช้ในการพัฒนาซอฟต์แวร์พร้อมประเมินประสิทธิภาพซอฟต์แวร์จากค่าความน่าจะเป็น

การอนุมานไวยากรณ์ (grammatical inference) จากลำดับกระบวนการ ในที่นี้เราจะนิยามลำดับเหล่านี้ว่าภาษา (language) การศึกษาภาษาที่ไม่รู้จักมาก่อนนั้นมีการศึกษากันอย่างแพร่หลาย (Sakakibara 1997; Honavar & Slutzki 1998) [1],[2] โดยใช้ตัวแบบการเรียนรู้ที่เรียกว่าการระบุภาษาได้ในขอบเขตจำกัด (identification in the limit) แสดงให้เห็นว่าจำเป็นต้องใช้

ตัวอย่างเชิงลบเพื่อเรียนรู้คลาสของภาษาสม่ำเสมอ (regular language) หรือภาษาที่ยอมรับโดยแบบจำลองส่งผ่านสถานะจำกัดเชิงกำหนด (deterministic finite automata, DFA) ซึ่งจะพิจารณาจากกลุ่มตัวอย่างเชิงบวก (positive examples) และตัวอย่างเชิงลบ (negative examples) เป็นขั้นตอนวิธีที่เรียกว่าการอนุมานไวยากรณ์ด้วยตัวอย่างเชิงบวกและตัวอย่างเชิงลบ (regular positive and negative grammatical inference (RPNI)) [3] มีผลทำให้เราสามารถระบุคลาสของภาษารูปนัยภายใต้ขอบเขตที่จำกัดได้ แต่ในเชิงปฏิบัติการนั้นการจะหาตัวอย่างเชิงลบได้ครบทุกกรณีเป็นไปได้ยาก จากตัวอย่างงานวิจัยต่าง ๆ เช่น ภาษาในระดับต่าง ๆ ในเชิงการประมวลผลภาษาธรรมชาติ (natural language processing) [4] หรือสัญญาณรบกวนความถี่ที่หลากหลายของระดับเสียงในเชิงการเรียนรู้จำเสียงพูด (speech recognition) [5] เป็นต้น ซึ่งงานวิจัยที่ผ่านมาได้มีการนำเสนอแบบจำลองส่งผ่านสถานะจำกัดเชิงกำหนดที่สามารถเรียนรู้ได้จากตัวอย่างเชิงบวกเพียงอย่างเดียว [6] ดังนั้นตัวช่วยที่มีประสิทธิภาพก็คือ แบบจำลองส่งผ่านอิงความน่าจะเป็นสถานะจำกัดเชิงกำหนด (deterministic probabilistic finite automata, DPFAs) ซึ่งวิธีนี้จะช่วยหลีกเลี่ยงข้อจำกัดที่กล่าวมาและช่วยเพิ่มประสิทธิภาพให้กับคลาสของแบบจำลองส่งผ่านอิงความน่าจะเป็นสถานะจำกัดเชิงกำหนด

งานวิจัยที่เกี่ยวกับแบบจำลองส่งผ่านอิงความน่าจะเป็นที่ผ่านมา มีเพียงขั้นตอนวิธีอัลเลอเจียร์ [7] ซึ่งพัฒนามาจากขั้นตอนวิธี RPNI เพื่อใช้ในการสกัดสารสนเทศ (information) จากข้อมูลที่สนใจที่มีประสิทธิภาพและมีความแม่นยำ [8] โดยขั้นตอนวิธีอัลเลอเจียร์ จะทำการสร้างต้นไม้อิงคำนำหน้า (prefix tree acceptor, PTA) จากนั้น จะทำการเปรียบเทียบ (compatible) และผสานสถานะ (state merging) ในกรณีที่ความน่าจะเป็นของภาษามีความสัมพันธ์ใกล้เคียงกับสถานะสุดท้ายที่พิจารณา

จากการศึกษาขั้นตอนวิธีอัลเลอเจียร์ พบว่าเป็นวิธีการอนุมานไวยากรณ์ที่ซับซ้อน ไม่สามารถแสดงออกมาในรูปแบบของไวยากรณ์ได้ และต้องใช้เวลาในการคำนวณมาก ดังนั้นงานวิจัยนี้จะนำเสนออัลกอริทึมของการอนุมานไวยากรณ์ที่ออกมาในรูปแบบของไวยากรณ์ ใช้เวลาในการคำนวณน้อยและมีค่าความน่าจะเป็นที่ใกล้เคียงกับค่าความน่าจะเป็นของการเกิดในกระบวนการของแบบจำลอง

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อนำเสนออัลกอริทึมในการการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นโดยการสร้างเครื่องจักรแบบจำกัดสถานะที่มีค่าความน่าจะเป็นที่ใกล้เคียงกับความน่าจะเป็นของการเกิดกระบวนการในแบบจำลอง

1.3 ขอบเขตงานวิจัย

1) งานวิจัยนี้เน้นที่การเรียนรู้จากสายอักขระ เพื่อให้ได้มาซึ่งไวยากรณ์
2) งานวิจัยนี้จะศึกษาเฉพาะสายอักขระที่เป็นตัวอย่างเชิงบวกเท่านั้น เนื่องจากเป็นการนำเสนออัลกอริทึมสำหรับการสร้างแบบจำลองที่อธิบายกระบวนการที่เกิดขึ้นทุกรูปแบบพร้อมด้วยความน่าจะเป็นที่แตกต่างกัน
3) ประสิทธิภาพขั้นตอนวิธีการเรียนรู้ พิจารณาจากค่าความน่าจะเป็นที่ใกล้เคียงกับความน่าจะเป็นของการเกิดกระบวนการในแบบจำลอง

1.4 ขั้นตอนการวิจัย

1) ศึกษาและทำความเข้าใจทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2) หาแนวทางและข้อมูลสนับสนุนงานวิจัย
3) วิเคราะห์และออกแบบอัลกอริทึม
4) ทดสอบและวิเคราะห์ผล
5) สรุปผลงานวิจัย
6) ตีพิมพ์ผลงานทางวิชาการ
7) จัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้มาซึ่งอัลกอริทึมที่ใช้ในการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น
2) สามารถนำแบบจำลองที่เรียนรู้ไปใช้ในงานต่อได้ เช่น การวิเคราะห์ระบบ เป็นต้น

1.6 ลำดับการจัดเรียงในวิทยานิพนธ์

เนื้อหาวิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 5 บท โดยเริ่มจากบทที่ 1 กล่าวถึงที่มาและความสำคัญของปัญหา วัตถุประสงค์ของงานวิจัย ขอบเขตงานวิจัย ขั้นตอนการวิจัย และประโยชน์ที่คาดว่าจะได้รับ บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้องที่นำมาใช้ในงานวิจัยนี้ บทที่ 3 การวิเคราะห์และออกแบบอัลกอริทึม รูปแบบอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น ขั้นตอนการอนุมานไวยากรณ์ด้วยอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น บทที่ 4 การทดสอบและวิเคราะห์ผล
วัตถุประสงค์การทดสอบ และผลการทดสอบ และบทที่ 5 สรุปผลการวิจัย ข้อจำกัด และแนวทางสาหรับการทำวิจัยต่อในอนาคต

1.7 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์ฉบับนี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “PROBABILISTIC REGULAR GRAMMAR INFERENCE ALGORITHM USING INCREMENTAL TECHNIQUE” โดย ต่อศักดิ์ เพ็ญภินันท์ และ อรรถสิทธิ์ สุรฤกษ์ ในหนังสือรวมบทความการประชุมวิชาการนานาชาติ 2018 International Workshop on Computer Science and Engineering, (WCSE) ณ กรุงเทพมหานคร ประเทศไทย วันที่ 29 มิถุนายน 2561 หน้า 780-785

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในส่วนนี้จะเริ่มด้วยการนิยามความหมายและสัญลักษณ์ต่าง ๆ ที่จะถูกอ้างถึงในงานวิจัยนี้ โดยแบ่งออกเป็นหัวข้อต่างๆ ดังนี้

2.1 ภาษารูปนัย (Formal Language)

หน่วยย่อยที่สุดของภาษา คือ อักขระหรือสัญลักษณ์ (character or symbol) ซึ่งเมื่อนำตัวอักขระเหล่านี้มารวมกันในรูปแบบของเซต เราเรียกเซตนี้ว่า เซตอักขระ (alphabet) ในกรณีที่ตัวอักขระแต่ละตัวมีอันดับ สามารถแสดงอันดับก่อนหลังของตัวอักขระได้เราเรียกเซตอักขระลักษณะนี้ว่า เซตอักขระมีอันดับ (ordered alphabet) เมื่อนำอักขระต่าง ๆ มาเรียงต่อกัน เราจะเรียกผลของการเรียงต่อกันว่า สายอักขระ (string) และเมื่อนำสายอักขระที่มีความหมายอย่างเดียวกันมารวมกลุ่มกันในรูปแบบเซต เราเรียกเซตของสายอักขระนี้ว่า ภาษา (language)โดยความหมายของสายอักขระในแต่ละภาษาเป็นตัวแสดงความแตกต่างระหว่างภาษาหนึ่งกับอีกภาษาหนึ่ง ซึ่งความหมายเกิดจากการประกอบกันของอักขระอยู่ในรูปแบบกฎที่เรียกว่า ไวยากรณ์ (grammar)ของภาษา สำหรับแบบจำลองที่ใช้ในการรู้จำ (recognize) ภาษาเรียกว่า แบบจำลองส่งผ่าน (automata) ทั้งไวยากรณ์และแบบจำลองส่งผ่านสามารถใช้บอกความซับซ้อนของระดับของภาษาได้ซึ่งมีรูปแบบที่ใช้ในงานวิจัยที่อ้างถึง ดังนั้นคำจำกัดความต่างๆ สามารถอธิบายออกเป็นนิยามได้ ดังนี้

นิยามที่ 2.1.1 ชุดอักขระ (alphabet) จะหมายถึงเซตจำกัดของสัญลักษณ์ที่เป็นหน่วยย่อยที่สุด ไม่สามารถแบ่งแยกได้อีก จะใช้แทนด้วยตัว Σ และเรียกสมาชิกในชุดว่า อักขระ (character) เช่น a , b เป็นต้น

นิยามที่ 2.1.2 สายอักขระ (string) จะหมายถึงลำดับของอักขระ ถ้าลำดับมีจำนวนจำกัดจะเรียกว่า สายอักขระจำกัด (finite string) หากมีค่าเป็นอนันต์ จะเรียกว่า สายอักขระอนันต์ (infinite string) เช่น aa , aaa, bb , aab , aaaabaab เป็นต้น

นิยามที่ 2.1.3 ภาษา (language) จะหมายถึงเซตของสายอักขระที่มีจำนวนจำกัด เช่น { aa, ab, ba , bb, ...} เป็นต้น โดยสมาชิกในภาษาจะเรียกว่า คำ (word)

นิยามที่ 2.1.4 ภาษารูปนัย (formal language) จะหมายถึง ภาษาที่มีคำในภาษามีกฎหรือกติกาที่ชัดเจนในการพิจารณาว่าเป็นสมาชิกในภาษานั้นหรือไม่โดยไม่มีความกำกวม ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.1.4.1

บรรยายแบบแจกแจงสมาชิก

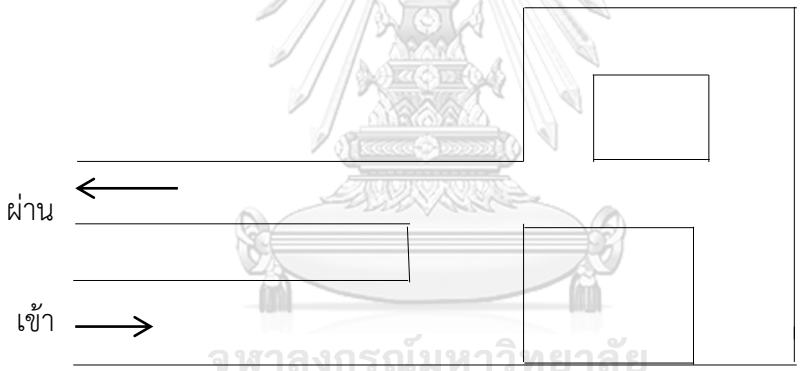
{ 0, 00, 000, 0000, ... }

บรรยายด้วยรูปแบบทั่วไป

$\{ 0^n \mid n = 0, 1, 2, \ldots \}$ ซึ่งยกกำลังนิยมใช้ n ในกรณี การเขียนซ้ำ n ครั้ง

ตัวอย่างที่ 2.1.4.2

ตัวอย่างนี้จะแสดงถึงข้อกำหนดของภาษาที่แทนลำดับกระบวนการบังคับการเดินของหุ่นยนต์ โดยกำหนดให้ 1 คือการเลี้ยวซ้ายของหุ่นยนต์ และ 0 คือการเลี้ยวขวาของหุ่นยนต์เมื่อเจอเส้นทางที่ต้องตัดสินใจ ซึ่งคำตอบของเซตของภาษาคือการที่หุ่นยนต์สามารถเดินออกไปที่ผ่านได้



ภาพที่ 2.1 แสดงข้อกำหนดของภาษาที่แทนลำดับกระบวนการบังคับการเดินของหุ่นยนต์

ดังนั้นจะได้ $\{ 1, 010, 01110, 0111110, \ldots \}$ ซึ่ง $\{ 1 \} \cup \{ 01^m0 \mid m = \text{odd integer} \}$ นั่นเอง

นิยามที่ 2.1.5 สายอักขระที่ไม่มีอักขระจะเรียกว่า สายอักขระว่าง แทนด้วย λ (empty string)

นิยามที่ 2.1.6 ความยาวของสายอักขระ จะหมายถึง จำนวนของอักขระที่อยู่ในสายนั้น โดยที่สายอักขระว่างจะถือว่ามีความยาวเป็น 0

นิยามที่ 2.1.7 Kleenne's star จะหมายถึง ภาษาที่สายอักขระมีทุกรูปแบบที่เป็นไปได้จากอักขระใน Σ จะถูกเรียกว่าเป็น closure ของ Σ โดยใช้สัญลักษณ์เป็น Σ^* เช่น

$\Sigma = \{ 0,1 \}$ จะได้ $\Sigma^* = \{ 0,1 \}^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \}$

นิยามที่ 2.1.8 Kleenne's star บนตัวอักขระ a เขียนแทนด้วย a* คือ เซตของสายอักขระที่เกิดจากอักขระ a เรียงต่อกันกี่ตัวก็ได้ เช่น

a* = { ⋋, a, aa, aaa, aaaa, ... }

นิยามที่ 2.1.9 ตัวอักขระที่เขียนแทนด้วย a + b จะหมายถึง เซตของสายอักขระที่เกิดจากการเลือกตัวอักขระ a หรือ ตัวอักขระ b อย่างใดอย่างหนึ่ง

a + b = { a, b }

2.2 เครื่องจักรแบบจำกัดสถานะ (Finite State Machine)

การตรวจสอบการเป็นสมาชิกในภาษา สามารถทำได้โดยการใช้ตัวแบบทางคณิตศาสตร์ที่เรียกว่า เครื่องจักรแบบจำกัดสถานะ หรือ เรียกอีกอย่างได้ว่า ออโตมาตา (automata) โดยสนใจวิธีในการสร้างตัวแบบที่เหมาะสมกับภาษาที่กำหนดมาแต่แรก ดังนั้นตัวแบบนี้จะสามารถตอบได้เพียง 2 รูปแบบเท่านั้น คือ เป็นสมาชิกหรือไม่เป็นสมาชิก สามารถนิยามได้ ดังนี้

นิยามที่ 2.2.1 เครื่องจักออโตมาตาแบบจำกัด (finite automaton : FA) จะประกอบด้วย 5 ส่วนสำคัญ คือ

$$M = (Q, \Sigma, q_0, A, \delta)$$

โดยที่ Q เป็นเซตของสถานะ (state) ของเครื่องจักร

Σ เป็นเซตของข้อมูลนำเข้า นั่นคือ ตัวอักขระ

q_0 เป็นสถานะเริ่มต้น (initial state) และเป็นสมาชิกใน Q

A เป็นสถานการณ์ยอมรับ (accept state) หรือสถานะจบ (final state)

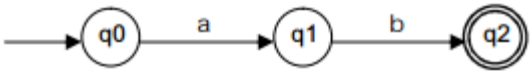
δ เป็นฟังก์ชันการเปลี่ยนสถานะ หรือ เรียกว่า ฟังก์ชันการผ่าน (transition function) โดย

$$\delta = \mathrm{Q} \times \Sigma \qquad \mathrm{Q} \rightarrow$$

การทำงานของออโตมาตา จะทำงานโดยการรับข้อมูลสายอักขระเข้ามาด้วยการอ่านทีละอักขระตามลำดับ เมื่อเริ่มต้น ออโตมาตาจะอยู่ในสภาพเริ่มต้น และเมื่ออ่านอักขระเข้ามาออโตมาตาจะปรับเปลี่ยนสถานะเครื่องจักรไปตามค่าที่รับเข้ามา และจะดำเนินการอ่านไปเรื่อยๆทีละอักขระจนกระทั่งอักขระตัวสุดท้ายถูกอ่านเรียบร้อยแล้ว ออโตมาตาจะหยุดทำงาน สถานะสุดท้ายจะมีผลต่อคำตอบ ตามนิยาม

นิยามที่ 2.2.2 สายอักขระที่ทำให้ออโตมาตาหยุดที่สถานะยอมรับ จะถูกเรียกว่า ถูกยอมรับโดยออโตมาตา ไม่เช่นนั้นจะถูกเรียกว่า ถูกปฏิเสธโดยออโตมาตา

ตัวอย่างที่ 2.2.2.1



แสดงถึงออโตมาตาที่อธิบายตามนิยามได้ ดังนี้

Q คือเซตของสถานะ = { q_0, q_1, q_2 }

Σ เป็นเซตของข้อมูลนำเข้า = { a, b }

q_0 เป็นสถานะเริ่มต้น = q_0

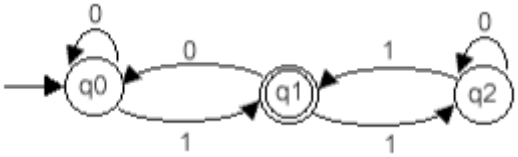
A เป็นสถานการณ์ยอมรับ = q_2

δ เป็นฟังก์ชันการเปลี่ยนสถานะ : $\delta(q_0, a) = q_1$,

$\delta(q_1, b) = q_2$

ดังนั้นการทำงานของออโตมาตาคือ จะรับอักขระจากสายอักขระเข้ามาทีละตัว และเปลี่ยนสถานะตามเงื่อนไขของฟังก์ชันการเปลี่ยนสถานะ กล่าวคือหากสถานะ q_0 รับอักขระ a จะเปลี่ยนไปสถานะ q_1 และ q_1 รับอักขระ b จะเปลี่ยนไปสถานะ q_2 และหากสายอักขระมาจบที่สถานะ q_2 จะถือว่าถูกยอมรับโดยออโตมาตา นั่นคือสายอักขระ ab นั่นเอง

ตัวอย่างที่ 2.2.2.2



แสดงถึงออโตมาตาที่อธิบายตามนิยามได้ ดังนี้

Q คือเซตของสถานะ = { q_0, q_1, q_2 }

Σ เป็นเซตของข้อมูลนำเข้า = { 0, 1 }

q_0 เป็นสถานะเริ่มต้น = q_0

A เป็นสถานการณ์ยอมรับ = q_1

δ เป็นฟังก์ชันการเปลี่ยนสถานะ : $\delta(q_0, 0) = q_0$,

$\delta(q_0, 1) = q_1$,

$\delta(q_1, 1) = q_2$,

$\delta(q_1, 0) = q_0$,

$\delta(q_2, 0) = q_2$,

$\delta(q_2, 1) = q_1$

จากตัวอย่างนี้ ออโตมาตาจะรับอักขระจากสายอักขระเข้ามาทีละตัว และเปลี่ยนสถานะตามเงื่อนไขของฟังก์ชันการเปลี่ยนสถานะ ดังนี้

1) หากสถานะ q_0 รับอักขระ 0 จะไม่เปลี่ยนสถานะ
2) หากสถานะ q_0 รับอักขระ 1 จะเปลี่ยนสถานะเป็น q_1
3) หากสถานะ q_1 รับอักขระ 1 จะเปลี่ยนสถานะเป็น q_2
4) หากสถานะ q_1 รับอักขระ 0 จะไม่เปลี่ยนสถานะ
5) หากสถานะ q_2 รับอักขระ 0 จะไม่เปลี่ยนสถานะ
6) หากสถานะ q_2 รับอักขระ 1 จะเปลี่ยนสถานะเป็น q_1

ดังนั้นภาษาที่จะถูกยอมรับโดยออโตมาตา ได้แก่ { 1, 01, 001, 111, 1101, 11101, ... } เป็นต้น

2.3 ขอบเขตของโฮฟฟ์ดิ่ง (Hoeffding Bound)

ขอบเขตของโฮฟฟ์ดิ่ง [9],[10] คือ ช่วงของความเชื่อมั่น โดยพิจารณาค่าความน่าจะเป็นและค่าความถี่ของข้อมูล มีเงื่อนไข ดังนี้

$$|p - \frac{f}{n}| < \sqrt{\frac{1}{2n}\log\frac{2}{\propto}}$$

โดยที่

p คือ ค่าความน่าจะเป็น

n คือ ค่าความถี่ทั้งหมดของตัวแปรที่สนใจ

f คือ ค่าความถี่จากการสังเกตตัวแปรที่สนใจ จากความถี่ทั้งหมด

$\propto$ คือ ค่าขีดแบ่ง (threshold) สามารถกำหนดให้มีค่าเท่ากับ 0.05

2.4 พื้นฐานขั้นตอนวิธีอัลเลอเจียร์

วิธีการขั้นตอนอัลเลอเจียร์ นั้นเริ่มต้นด้วยการสร้างแบบจำลองส่งผ่านสถานะจำกัดเชิงกำหนดในรูปแบบของต้นไม้อิงคำนำหน้า (ภาพที่ 2.2-b) จากตัวอย่างที่สนใจ จากนั้นใช้เทคนิคของการผสานสถานะเพื่อลดขนาดของแบบจำลอง (ข้อ 2.4.2) โดยที่แบบจำลองที่ได้หลังจากการผสานสถานะจะได้แบบจำลองส่งผ่านอิงความน่าจะเป็นสถานะจำกัดเชิงกำหนดอย่างย่อ (canonical acceptor) (ภาพที่ 2.4-b) ซึ่งสามารถยอมรับตัวอย่างเชิงบวกได้ทุกตัว (ภาพที่ 2.4) จากขั้นตอนวิธี 1 (Algorithm 1) จะแสดงขั้นตอนวิธี อัลเลอเจียร์ โดยพิจารณาสถานะที่สนใจกับสถานะถัดไปเพื่อทำการผสานสถานะ

Algorithm 1 ALERGIA.

Input: a sample S, $\alpha > 0$
Output: an FFA A
$A \leftarrow$ FPTA(S);
RED $\leftarrow \{q_\lambda\}$;
BLUE $\leftarrow \{q_a : a \in \Sigma \cap$ PREF(S)$\}$;
while CHOOSE q_b from BLUE such that FREQ$(q_b \geq t_0)$ **do**
  if $\exists q_r \in$ RED : ALERGIA-COMPATIBLE(A, q_r, q_b, α) **then**
    $A \leftarrow$ STOCHASTIC-MERGE(A, q_r, q_b)
  else
    RED $\leftarrow$ RED $\cup \{q_b\}$
  end if
  BLUE $\leftarrow \{q_{ua} : ua \in$ PREF(S) $\wedge\ q_u \in$ RED$\}$ \ RED
end while
return A

เทคนิคการผสานสถานะ (state merging technique) จะประกอบด้วยการดำเนินการพื้นฐาน 2 ส่วน คือ การตรวจสอบความสอดคล้อง (compatible) และการผสานสถานะ (state Merging)

2.4.1 การตรวจสอบความสอดคล้อง (Compatible)

เป็นการตรวจสอบความเข้ากันได้ของคู่สถานะ โดยพิจารณาจากหลักการประมาณขอบเขตของโฮฟฟ์ดิ่ง (Hoeffding bound) จากขั้นตอนวิธีที่ 2 (algorithm 2) แสดงให้เห็นขั้นตอนวิธีการตรวจสอบความสอดคล้องของคู่สถานะโดยการเรียกใช้ขั้นตอนวิธีที่ 3 (algorithm 3) ซึ่งอาศัยหลักการประมาณขอบเขตของโฮฟฟ์ดิ่งในการประเมินคู่สถานะ ซึ่งเปรียบเทียบได้ 2 รูปแบบ

1) เปรียบเทียบเชิงปริมาณ

$$\frac{\mathbb{F}_{fr}(q)}{FREQ(q)} \quad \text{และ} \quad \frac{\mathbb{F}_{fr}(q')}{FREQ(q')}$$

2) เปรียบเทียบเชิงอักขระ

$$\frac{\delta_{fr}(q,a)}{FREQ(q)} \quad \text{และ} \quad \frac{\delta_{fr}(q',a)}{FREQ(q')}$$

Algorithm 2 ALERGIA-COMPATIBLE.

Input: an FFA A, two states q_u, q_v, $\alpha > 0$
Output: q_u and q_v compatible ?
Correct $\leftarrow$ true;
if ALERGIA-TEST$(\mathbb{F}_{fr}(q_u), \text{FREQ}_A(q_u), \mathbb{F}_{fr}(q_v), \text{FREQ}_A(q_v), \alpha)$ **then**
  Correct $\leftarrow$ false;
end if
for $a \in \mathcal{A}$ **do**
  if ALERGIA-TEST$(\delta_{fr}(q_u, a), \text{FREQ}_A(q_u), \delta_{fr}(q_v, a), \text{FREQ}_A, \alpha)$ **then**
    Correct $\leftarrow$ false;
  end if
end for
return Correct

Algorithm 3 ALERGIA-TEST.

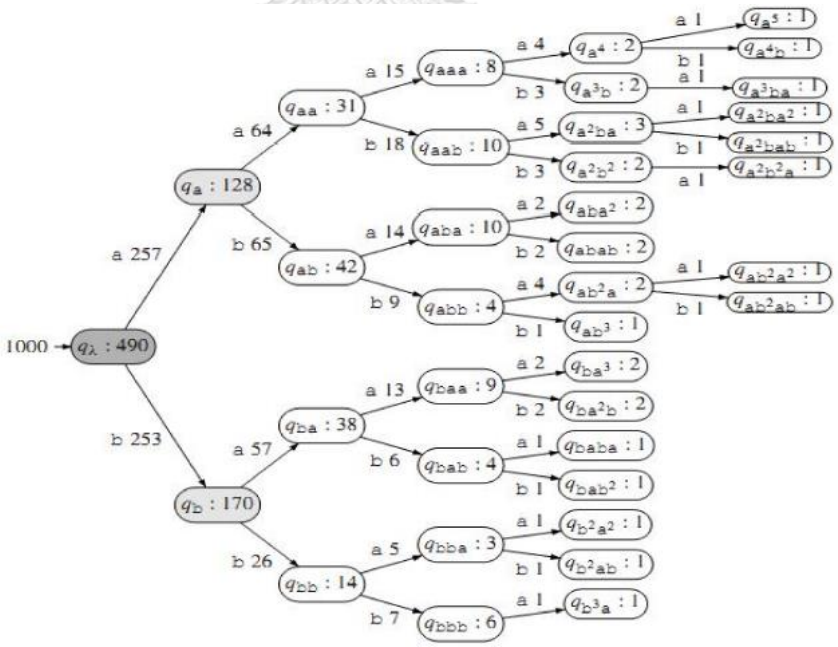
Input: an FFA A, f_1, n_1, f_2, n_2, $\alpha > 0$
Output: a Boolean indication if the frequencies $\frac{f_1}{n_1}$ and $\frac{f_2}{n_2}$ and sufficiently close
$\gamma \leftarrow \left|\frac{f_1}{n_1} - \frac{f_2}{n_2}\right|$;
return $\left(\gamma < \left(\sqrt{\frac{1}{n_1}} + \sqrt{\frac{1}{n_2}} \cdot \sqrt{\frac{1}{2}\ln\frac{2}{\alpha}}\right)\right)$

2.4.2 การผสานสถานะ (State Merging)

เป็นการนำสถานะ 2 สถานะที่ถูกเลือกเข้าเป็นสถานะเดียวจากนั้นทำการกระจายสถานะอื่นๆภายใต้สถานะดังกล่าวไปรวมกับสถานะที่มีความสอดคล้อง ภายใต้ต้นไม้อิงคำนำหน้าที่สร้างขึ้นจากขั้นตอนวิธีที่ 4 (algorithm 4) แสดงให้เห็นขั้นตอนวิธีในการผสานสถานะที่สนใจกับสถานะถัดไป จากนั้นจึงเรียกใช้ขั้นตอนวิธีที่ 5 (algorithm 5) เพื่อกระจายสถานะอื่นๆไปยังต้นไม้อิงคำนำหน้าที่สร้างขึ้น

λ	490	abb	4	abab	2	aaaaa	1
a	128	baa	9	abba	2	aaaab	1
b	170	bab	4	abbb	1	aaaba	1
aa	31	bba	3	baaa	2	aabaa	1
ab	42	bbb	6	baab	2	aabab	1
ba	38	aaaa	2	baba	1	aabba	1
bb	14	aaab	2	babb	1	abbaa	1
aaa	8	aaba	3	bbaa	1	abbab	1
aab	10	aabb	2	bbab	1		
aba	10	abaa	2	bbba	1		

(a) แสดงตารางความถี่



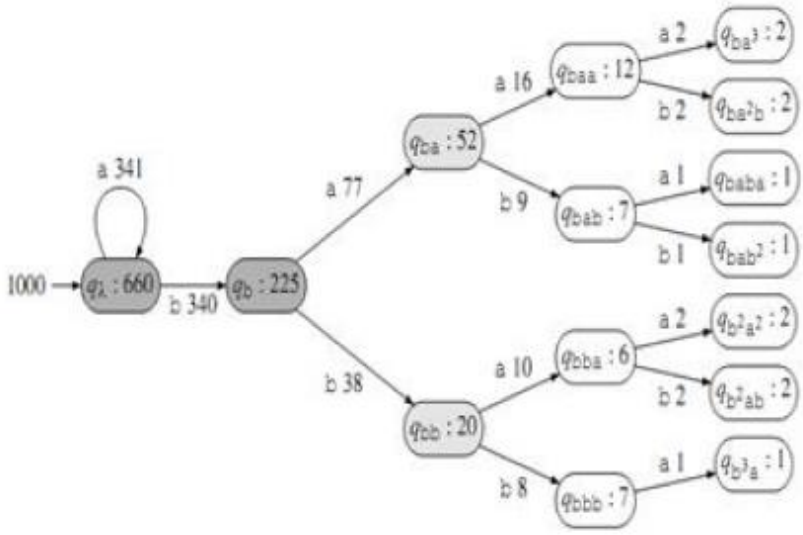
(b) แสดงต้นไม้อิงคำนำหน้า

ภาพที่ 2.2 แสดงการแปลงตารางความถี่เป็นต้นไม้อิงคำนำหน้า

ขั้นตอนที่ 1 พิจารณาที่สถานะ q_λ โดย Alergia จะผสานสถานะ q_λ กับ q_a เนื่องจากการตรวจสอบความเข้ากันได้ของคู่สถานะ โดยพิจารณาจากหลักการประมาณขอบเขตของโฮฟฟ์ดิ่ง (Hoeffding bound) กล่าวคือ

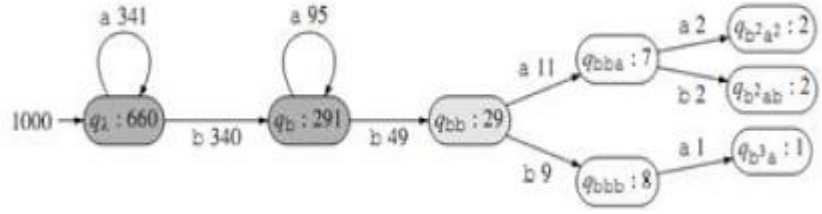
$$\left|\frac{\delta_{fr}(q_\lambda,a)}{FREQ(q_\lambda)} - \frac{\delta_{fr}(q_\lambda,a)}{FREQ(q_\lambda)}\right| = \left|\frac{257}{1000} - \frac{64}{257}\right| < \left(\sqrt{\frac{1}{1000}} + \sqrt{\frac{1}{257}}\right)\sqrt{\ln\frac{2}{\propto}}$$

ด้วยค่า $\propto$ = 0.05



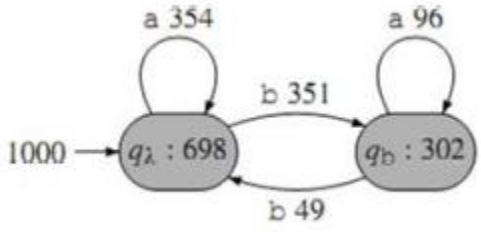
(a) แสดงการผสานสถานะ รอบที่ 1

ขั้นตอนที่ 2 ตรวจสอบความเข้ากันได้ของคู่สถานะ q_λ กับ q_b ไม่สามารถผสานสถานะได้จากการเปรียบ $\frac{660}{1341}$ และ $\frac{225}{340}$ และเมื่อตรวจสอบความเข้ากันของคู่สถานะ q_λ กับ q_{ba} ก็ไม่สามารถผสานสถานะได้ จากการเปรียบเทียบ เปรียบ $\frac{660}{1341}$ และ $\frac{52}{77}$ จากนั้นพิจารณาที่สถานะ q_b จากการตรวจสอบความเข้ากันได้ของคู่สถานะ จะสามารถผสานสถานะ q_b กับ q_{ba}



(b) แสดงการผสานสถานะ รอบที่ 2

ขั้นตอนที่ 3 ตรวจสอบความเข้ากันได้ของคู่สถานะ q_λ กับ q_{bb} พบว่า สามารถผสานสถานะได้ ดังนั้นจะได้แบบจำลองส่งผ่านอิงความน่าจะเป็นสถานะจำกัดเชิงกำหนด ดังรูป



(c) แสดงการผสานสถานะ รอบที่ 3

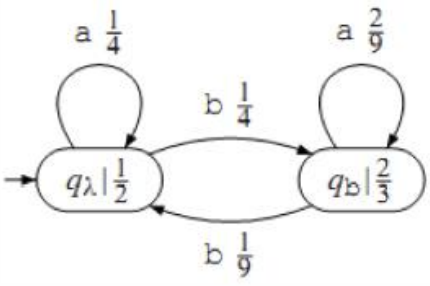
ภาพที่ 2.3 แสดงขั้นตอนการผสานสถานะจากต้นไม้อิงคำนำหน้า

Algorithm 4 STOCHASTIC-MERGE.

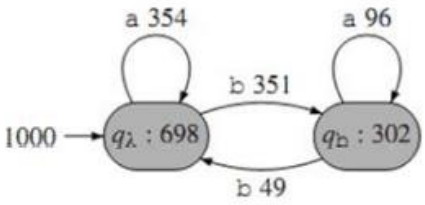
Input: an FFA A, 2 states $q \in$ RED, $q' \in$ BLUE
Output: A updated
Let(q_f, a) be such that $\delta_A(q_f, a) = q'$;
$n \leftarrow \delta_{fr}(q_f, a, q')$;
$\delta_A(q_f, a) \leftarrow q$;
$\delta_{fr}(q_f, a, q) \leftarrow n$;
$\delta_{fr}(q_f, a, q') \leftarrow 0$;
return STOCHASTIC-FOLD(A, q, q')

Algorithm 5 STOCHASTIC-FOLD.

Input: an DFFA A, 2 states $q \in$ RED, $q' \in$ BLUE
Output: A updated, where subtree in q' is folded into q
$\mathbb{F}_{fr}(q) \leftarrow \mathbb{F}_{fr}(q) + \mathbb{F}_{fr}(q')$;
for $a \in \mathcal{A}$ such that $\delta_A(q', a)$ is defined **do**
  if $\delta_A(q, a)$ is defined **then**
    $\delta_{fr}(q, a, \delta_A(q, a)) \leftarrow \delta_{fr}(q, a, \delta_A(q, a)) + \delta_{fr}(q', a, \delta_A(q', a))$
    $A \leftarrow$ STOCHASTIC-FOLD$(A, \delta_A(q, a), \delta_A(q', a))$
  else
    $\delta_A(q, a), \leftarrow \delta_A(q', a)$;
    $\delta_{fr}(q, a, \delta_A(q, a)) \leftarrow \delta_{fr}(q', a, \delta_A(q', a))$
  end if
end for
return A



(a) แบบจำลองต้นแบบ



(b) แบบจำลองส่งผ่านอิงความน่าจะเป็นอย่างย่อ

ภาพที่ 2.4 แสดงการเปรียบเทียบแบบจำลองต้นแบบกับแบบจำลองส่งผ่านอิงความน่าจะเป็นอย่างย่อ

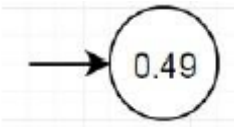
2.5 Probabilistic Deterministic Finite Automaton Learning Model

มีงานวิจัยของ Kasha Pinprasert [11] ที่นำเสนออัลกอริทึมชื่อ “MPD” (minimum probabilistic difference) ซึ่งเป็นอัลกอริทึมสำหรับการสร้าง PDFA (probabilistic deterministic finite automaton) จากตัวอย่างเชิงบวกของสายอักขระ โดยใช้การคำนวณหาความน่าจะเป็นของการเกิด และนำไปเปรียบเทียบกับอัลกอริทึมวิธีอัลเลอเจียร์ ซึ่งรายละเอียดอัลกอริทึม MPD มีขั้นตอน ดังนี้

ตารางที่ 2.1 แสดงตารางความถี่ของสายอักขระ

λ	490	abb	4	abab	2	aaaaa	1
a	128	baa	9	abba	2	aaaab	1
b	170	bab	4	abbb	1	aaaba	1
aa	31	bba	3	baaa	2	aabaa	1
ab	42	bbb	6	baab	2	aabab	1
ba	38	aaaa	2	baba	1	aabba	1
bb	14	aaab	2	babb	1	abbaa	1
aaa	8	aaba	3	bbaa	1	abbab	1
aab	10	aabb	2	bbab	1		
aba	10	abaa	2	bbba	1		

ขั้นตอนที่ 1 สร้างสถานะจากสายอักขระที่มีความยาวสั้นที่สุดนั่นคือ λ จากนั้นหาความน่าจะเป็นที่สายอักขระจะหยุดที่สถานะนี้ นั่นคือ ผลรวมของความถี่สายอักขระทั้งหมดมี 1,000 และมีสายอักขระที่หยุดที่สถานะนี้ 490 ดังนั้นมีความน่าจะเป็น 490/100 = 0.49

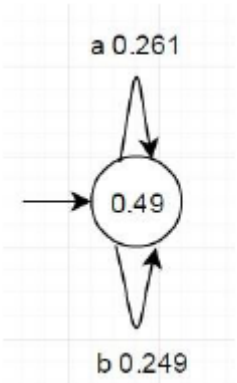


(a) แสดงอัลกอริทึม MPD ขั้นตอนที่ 1

ขั้นตอนที่ 2 จากนั้นจะต้องผสานสถานะถัดไป ซึ่งจะคำนวณค่าความเป็นไปได้ โดยพิจารณาสถานะที่มีความยาวสั้นที่สุดลำดับต่อไป คือ a ดังนี้

(Pa)(P⋀) = 128/1000 โดย P⋀ = 0.49

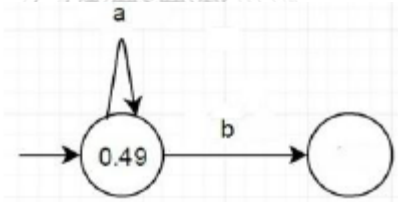
จะได้ Pa = 0.261 และ Pb = 1-Pa- P⋀ = 0.249



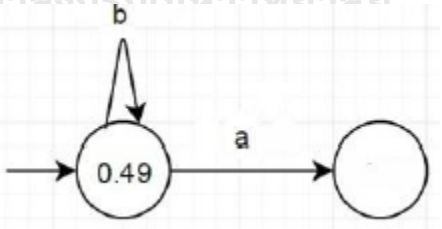
(b) แสดงอัลกอริทึม MPD ขั้นตอนที่ 2

ภาพที่ 2.5 แสดงอัลกอริทึม MPD

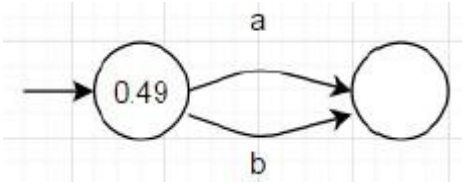
ขั้นตอนที่ 3 ตรวจสอบความสอดคล้องของการผสานสถานะ ซึ่งการผสานสถานะจะสามารถทำได้ 3 รูปแบบ ดังรูป



(a) แสดงรูปแบบการผสานสถานะ รูปแบบที่ 1



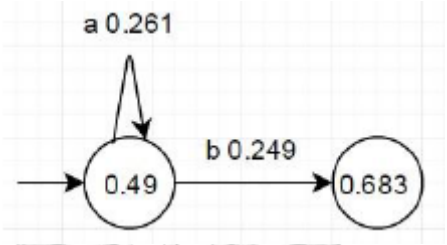
(b) แสดงรูปแบบการผสานสถานะ รูปแบบที่ 2



(c) แสดงรูปแบบการผสานสถานะ รูปแบบที่ 3

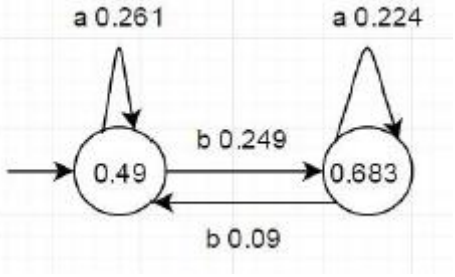
ภาพที่ 2.6 แสดงแสดงรูปแบบการผสานสถานะ

จากการคำนวณค่าความสอดคล้องพบว่า อักขระ “b” ไม่สอดคล้อง เนื่องจาก ค่าความน่าจะเป็นจากตารางมีค่า 170/1000 = 0.17 นำมาเปรียบเทียบกับค่าความน่าจะเป็นจากภาพที่ 5-b คือ (0.249)(0.49) = 0.122 จะได้ค่าผลต่างคือ |0.017 – 0.122| = 0.048 พบว่ามีค่ามากกว่าค่าของ $\propto$ (พิจารณาค่า $\propto$ = 0.005) ซึ่งมีค่าผิดพลาดมากเกินนั่นเอง ดังนั้นหลังจากผสานสถานะจะได้ดังรูปแบบที่ 1



(a) แสดงอัลกอริทึม MPD หลังผสานสถานะครั้งที่ 1

ขั้นตอนที่ 4 พิจารณาต่อไป สุดท้ายจะได้ PDFA ดังรูป



(b) แสดงอัลกอริทึม MPD หลังผสานสถานะครั้งที่ 2

ภาพที่ 2.7 แสดงอัลกอริทึม MPD หลังผสานสถานะ

งานวิจัยนี้ได้ใช้ตารางข้อมูลของสายอักขระ 2 ตัวอย่างเพื่อเปรียบเทียบอัลกอริทึมอัลเลอเจียร์ และ MPD

ตารางที่ 2.2 แสดงตารางความถี่ของสายอักขระทั้ง 2 ตัวอย่าง

Ex1		Ex2		Ex2 (con.)
λ	490.0	λ	312.0	abaab 3.0
a	128.0	a	245.0	ababa 3.0
b	170.0	b	113.0	ababb 2.0
aa	31.0	aa	17.0	abbab 1.0
ab	42.0	ab	119.0	bbaaa 1.0
ba	38.0	ba	8.0	bbaab 1.0
bb	14.0	bb	43.0	bbaba 2.0
aaa	8.0	aaa	8.0	bbbaa 1.0
aab	10.0	aab	8.0	aaabab 1.0
aba	10.0	aba	18.0	aabbab 1.0
abb	4.0	abb	20.0	ababaa 1.0
baa	9.0	bab	6.0	abbabb 2.0
bab	4.0	bba	7.0	bbabab 1.0
bba	3.0	bbb	17.0	aaababb1.0
bbb	6.0	aaab	1.0	abbaaba1.0
aaaa	2.0	aaba	1.0	abbbaab1.0
aaab	2.0	aabb	2.0	bbababb1.0
aaba	3.0	abaa	3.0	abaabaaa1.0
aabb	2.0	abab	4.0	baaababb1.0
abaa	2.0	abba	3.0	bbaaaaaa1.0
abab	2.0	abbb	1.0	bbaaaabb1.0
abba	2.0	baab	1.0	
abbb	1.0	bbaa	1.0	
baaa	2.0	bbab	1.0	
baab	2.0	bbbb	4.0	
baba	1.0	aaaaa	1.0	
babb	1.0	aaaab	2.0	
bbaa	1.0	aaabb	1.0	
bbab	1.0	aabab	1.0	
bbba	1.0	abaaa	2.0	

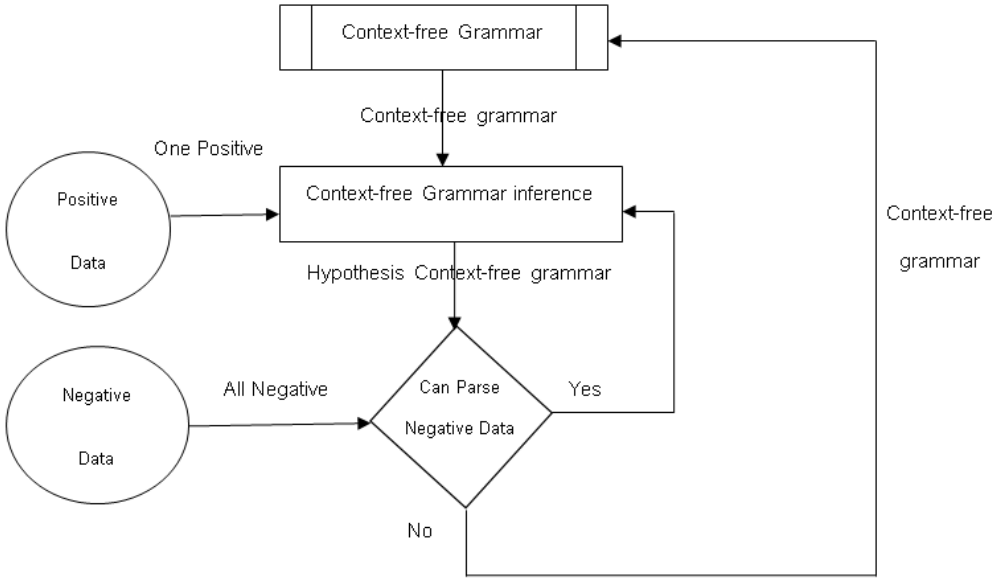
ตารางที่ 2.3 แสดงผลลัพธ์ของการใช้อัลกอริทึมอัลเลอเจียร์ และ MPD

ALERGIA Ex1	MPD Ex1
Alpha : #State : Avr Diff	Alpha :#State :Avr Diff
0.5000 : 2 : 1.2445349625416635	0.0050 : 2 : 0.6084270474347239
0.0600 : 2 : 1.397587635257587	0.0026 : 4 : 0.45387093289663377
	0.0024 : 4 : 0.42954445712431755
	0.0017 : 4 : 0.40452935800303916
	0.0015 : 5 : 0.30538560417130567
	0.0013 : 7 : 0.26457534592784504
ALERGIA Ex2	MPD Ex2
Alpha : #State : Avr Diff	Alpha :#State :Avr Diff
0.8000 : 5 : 1.7014027230232944	0.0300 : 2 : 2.4721284273432553
0.7500 : 5 : 1.7275936607549605	0.0299 : 2 : 2.5160447999496994
0.7000 : 4 : 1.6222162851760451	0.0215 : 3 : 2.6334020811729242
0.4200 : 4 : 1.8873555980710068	0.0214 : 3 : 2.0639791154167404
0.2600 : 4 : 2.168731887133727	0.0183 : 4 : 1.6121304379835014
0.2500 : 3 : 3.497274375144539 4	0.0129 : 4 : 1.5362777263756633
0.1400 : 3 : 2.7397306791728355	0.0083 : 4 : 1.4927543975588519
0.0900 : 3 : 2.3494165240616513	0.0079 : 4 : 1.3401735690907999
0.0300 : 3 : 2.5195792907613064	0.0062 : 4 : 1.2456490306979591
	0.0054 : 5 : 1.0923973579720057
	0.0041 : 5 : 1.0405845204369981
	0.0040 : 5 : 1.0752374179700062
	0.0038 : 6 : 1.0066554278114137
	0.0031 : 6 : 0.9811745758641687
	0.0028 : 7 : 0.8789557852455739
	0.0026 : 7 : 0.8232041746434137

งานวิจัยใช้อัลกอริทึมอัลเลอเจียร์ และ MPD ด้วยค่า $\propto$ แตกต่างกันหลายค่าจากตารางที่ 3 จะพบว่าเมื่อจำนวนสถานะเท่ากัน MPD มีค่าเฉลี่ยความน่าจะเป็นของผลต่างที่น้อยกว่าวิธีของอัลเลอเจียร์ ดังนั้นจากงานวิจัยนี้จะพบว่าอัลกอริทึม MPD จะสามารถสร้าง PDFA ที่มีการประเมินที่ดีกว่าอัลกอริทึมอัลเลอเจียร์ แต่อัลกอริทึม MPD ก็ยังเป็นอัลกอริทึมสำหรับการสร้าง Automaton ที่มีความน่าจะเป็นเท่านั้น ยังไม่ใช่อัลกอริทึมที่ช่วยสร้างไวยากรณ์ที่มีความน่าจะเป็นของการเกิดสายอักขระ

2.6 อัลกอริทึมการอนุมานไวยากรณ์ไม่พึ่งบริบท

ได้มีการนำเสนองานวิจัยที่นำเสนอไวยากรณ์ของภาษารูปนัยที่เรียนรู้จากตัวอย่างของสมาชิกในภาษา เพื่อให้ได้ไวยากรณ์ที่นำไปสร้างภาษาที่ใกล้เคียงความเป็นจริงมากที่สุด โดยงานวิจัยนี้จะนำตัวอย่างลบมาประกอบการพิจารณาด้วย งานวิจัยนี้ถึงแม้จะนำเสนออัลกอริทึมสำหรับการอนุมานไวยากรณ์ แต่ขอบเขตของงานวิจัยจะมุ่งเน้นที่ภาษาไม่พึ่งบริบท(context free)เท่านั้น ดังนั้นอัลกอริทึมนี้จะเป็นอัลกอริทึมของการอนุมานไวยากรณ์ไม่พึ่งบริบท วิธีการของอัลกอริทึมนั้นจะนำตัวอย่างเชิงบวกมาเรียนรู้รูปแบบที่เกิดขึ้นภายในตัวอย่างแต่ละตัวอย่าง และสกัดเอารูปแบบที่เกิดขึ้นโดยพิจาณาการเกิดซ้ำของรูปแบบไปพร้อมกัน จากนั้นนำมาพิจารณาประกอบกับตัวอย่างลบ ซึ่งข้อกำหนดคือต้องไม่สามารถแจงส่วนตัวอย่างลบได้แม้แต่ตัวอย่างเดียว ซึ่งขั้นตอนการทำงาน ดังนี้



ภาพที่ 2.8 แสดงการทำงานของอัลกอริทึมการอนุมานไวยากรณ์ไม่พึ่งบริบท

<u>ขั้นตอนการทำงานของอัลกอริทึมการอนุมานไวยากรณ์ไม่พึ่งบริบท</u>

การทำงานของอัลกอริทึมสำหรับอนุมานไวยากรณ์ไม่พึ่งบริบท เริ่มต้นจากการพิจาณาตัวอย่างข้อมูลนำเข้าที่เป็นตัวอย่างเชิงบวก โดยพิจารณาทีละตัวอย่าง สำหรับข้อกำหนดเบื้องต้นข้อมูลจะต้องจัดเรียงตามความยาวของตัวอย่าง จากนั้นแต่ละครั้งที่พิจารณาจะมีการอนุมานไวยากรณ์ที่สามารถแจงส่วนตัวอย่างเชิงบวกที่พิจารณาอยู่ และรวมถึงตัวอย่างเชิงบวกทั้งหมดที่ได้พิจารณาในรอบก่อนหน้าด้วย เมื่อได้กฎไวยากรณ์ใหม่จะต้องนำไปพิจารณากับตัวอย่างลบว่า ต้องไม่

สามารถถูกแจงส่วนได้จากไวยากรณ์ที่เพิ่มขึ้นมานี้ การอนุมานไวยากรณ์ที่กล่าวมาทั้งหมดนี้ผลลัพธ์ที่ได้จากอัลกอริทึมคือ กฎไวยากรณ์สำหรับการแจงส่วนภาษาที่ครอบคลุมตัวอย่างเชิงบวกทั้งหมด

Algorithm

Input : an ordered set S_P of positive samples strings.

an ordered set S_N of negative sample strings.

Output : A set P of rules such that all the strings in S_P are derived from P but no

String in S_N is derived from P.

Begin

for i = 1 to $|S_P|$ do

if can_not_parse $(S_P[i])$ with TG then

Replace_Terminal_with_Nonterminal$(S_P[i], TG)$

Create_Rule$(S_P[i], TG)$

Merge_Rule(TG)

endif

endfor

$P \leftarrow TG$

Return P

end

บทที่ 3
วิเคราะห์และออกแบบอัลกอริทึม

จากการศึกษาวิธีการอนุมานไวยากรณ์ด้วยอัลกอริทึมอัลเลอเจียร์ พบว่า เป็นเทคนิคที่ได้รับความนิยมเป็นอย่างสูง [8] เนื่องจากมีประสิทธิภาพและมีความแม่นยำ แต่มีขั้นตอนการทำงานที่ซับซ้อนและมีขนาดสถานะของเครื่องจักรแบบสถานะอิงความน่าจะเป็นที่มีขนาดใหญ่ จึงทำให้ใช้เวลาในการคำนวณค่อนข้างสูง

งานวิจัยนี้นำเสนออัลกอริทึมในการอนุมานไวยากรณ์สม่ำเสมอด้วยเทคนิคการเพิ่มขึ้น โดยการสร้างเครื่องจักรแบบสถานะอิงความน่าจะเป็น ซึ่งค่าความน่าจะเป็นจากผลลัพธ์ที่ได้จะใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริงจากตัวอย่างเชิงบวกที่ศึกษาเมื่อเปรียบเทียบกับขั้นตอนวิธีอัลเลอเจียร์

แนวคิดของอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น เริ่มต้นด้วยการกำหนดรูปแบบของไวยากรณ์พื้นฐาน และคำนวณค่าความน่าจะเป็นของไวยากรณ์พื้นฐานทั้งหมดที่สามารถนำไปสร้างสายอักขระได้ทุกรูปแบบ โดยเริ่มต้นจากการกำหนดตัวแปร S สำหรับไวยากรณ์พื้นฐานที่ใช้สร้างสายอักขระความยาว n ตัวอักษร (เมื่อ n เริ่มต้นด้วย 0 และเพิ่มขึ้นทีละ 1 ตัวอักษร) สำหรับสายอักขระที่มีความยาวมากกว่า n จะเพิ่มตัวแปรใหม่เพื่อรองรับและปรับค่าความน่าจะเป็นให้สอดคล้อง ผลลัพธ์จากจากอัลกอริทึมจะได้ไวยากรณ์พื้นฐานที่สามารถนำไปสร้างสายอักขระได้ทุกรูปแบบพร้อมค่าความน่าจะเป็น จากนั้นสร้างเครื่องจักรแบบสถานะอิงความน่าจะเป็นจากไวยารกรณ์พื้นฐานเพื่อเปรียบเทียบกับขั้นตอนวิธีอัลเลอเจียร์

3.1 แนวคิดของงานวิจัย

งานวิจัยนี้สนใจการอนุมานไวยากรณ์สำหรับภาษาสม่ำเสมอ ซึ่งแตกต่างจากงานวิจัยที่ใช้แนวทางของกลุ่มภาษาไวยากรณ์ไม่พึงบริบท ทั้งนี้ผู้วิจัยมีแนวคิดจากการพิจารณาว่า แต่ละภาษาสามารถเขียนไวยากรณ์ได้มากกว่าหนึ่งรูปแบบ และด้วยความหลากหลายของรูปแบบของไวยากรณ์เราสามารถนำมาเป็นช่องทางในการสร้างไวยากรณ์ที่มีความน่าจะเป็นในการสร้างสมาชิกในภาษาที่ค่าความน่าจะเป็นมีความอิสระจากกันมากขึ้น โดยแนวคิดนี้จะนำมาใช้ในการอธิบายความน่าจะเป็นของกลุ่มของสมาชิกในภาษาที่มีสายอักขระเริ่มต้น (prefix) เหมือนกัน ผู้วิจัยขอยกตัวอย่างแบบง่ายเพื่อความเข้าใจในแนวคิดของงานวิจัยดังนี้

หากพิจารณาสมาชิกสองคำในภาษา เช่น a และ ab ซึ่งมีค่าความน่าจะเป็นของการเกิดคำเป็น x และ y ตามลำดับ จะเห็นว่าทั้งสองสมาชิกมีส่วนหน้าของคำที่เหมือนกัน คือ a ดังนั้นสมาชิกทั้งสองสามารถถูกสร้างได้ด้วยไวยากรณ์

ไวยากรณ์ที่ 1 $S \rightarrow aT$ ด้วยค่าความน่าจะเป็น m

ไวยากรณ์ที่ 2 $T \rightarrow \lambda$ ด้วยค่าความน่าจะเป็น n

ไวยากรณ์ที่ 3 $T \rightarrow bU$ ด้วยค่าความน่าจะเป็น o

ไวยากรณ์ที่ 4 $U \rightarrow \lambda$ ด้วยค่าความน่าจะเป็น p

โดยที่ a เกิดจากการใช้ไวยากรณ์ที่ 1 และ 2 ตามลำดับ ด้วยความน่าจะเป็น mn และ ab เกิดจากการใช้ไวยากรณ์ที่ 1 3 และ 4 ตามลำดับ ด้วยความน่าจะเป็น mop และเนื่องจากค่าความน่าจะเป็นของคำนั้นเกิดจากผลคูณของค่าความน่าจะเป็นของไวยากรณ์ทั้งหมดที่ถูกใช้ในการสร้างคำนั้น และหากเราทราบค่าความน่าจะเป็นของทั้งสองคำ เราสามารถที่จะคำนวณค่าความน่าจะเป็นให้กับทั้ง 4 ไวยากรณ์เพื่อให้ผลคูณของไวยากรณ์ที่ 1 และ 2 มีค่าเท่ากับ x ได้ และ ผลคูณของไวยากรณ์ที่ 1 3 และ 4 นั้นมีค่าเท่ากับ y ได้เช่นกัน

เนื่องจากแนวทางในการเขียนไวยากรณ์สม่ำเสมอมีความหลากหลาย และมีวิธีในการเขียนมากขึ้นเมื่อคำมีความยาวมากขึ้น ในงานวิจัยนี้สนใจการเขียนอนุมานไวยากรณ์ที่มีรูปแบบที่มีความซับซ้อนน้อยที่สุดเท่าที่จะทำได้ กล่าวคือ มีรูปแบบของไวยากรณ์เพียง 2 รูปแบบ ทั้งนี้เพราะเป็นรูปแบบที่สามารถใช้อธิบายโครงสร้างต้นไม้ได้ครบทุกรูปแบบ นั่นคือ

ไวยากรณ์แบบที่ 1 คือ $A \rightarrow aB$ และ

ไวยากรณ์แบบที่ 2 คือ $A \rightarrow \lambda$ เท่านั้น

เมื่อ A และ B หมายถึงตัวแปร

a แทนตัวอักขระ และ

λ แทนอักขระว่าง

จากแนวคิดดังกล่าว ผู้วิจัยจึงเสนอแนวทางในการอนุมานไวยากรณ์โดยพิจารณาจากตัวอย่างเชิงบวกของสมาชิกในภาษา ซึ่งเริ่มจากการพิจารณาสมาชิกที่มีความยาวน้อยสุดก่อน จากนั้นจะพิจารณาสมาชิกที่มีความยาวเพิ่มขึ้นตามลำดับ โดยคาดหวังว่า สมาชิกใหม่ที่ถูกนำมาพิจารณานั้น จะสามารถอธิบายได้จากไวยากรณ์ที่อนุมานมาก่อนหรืออาจมาจากการอนุมานไวยากรณ์เพิ่มเติมให้น้อยที่สุดเท่าที่จะทำได้ โดยอาศัยหลักการเพิ่มขึ้นของไวยากรณ์ที่สอดคล้องกับตัวอย่างเชิงบวกที่ได้เรียนรู้

ซึ่งต่างจากงานวิจัยที่ผ่านมาที่นำไวยากรณ์ทั้งหมดมาสร้างเป็นโครงสร้างต้นไม้รวมและทำการเปรียบเทียบค่าเพื่อลดขนาดของต้นไม้ให้อยู่ในรูปแบบที่เล็กที่สุด โดยคาดหวังว่า แนวคิดในการอนุมานแบบของผู้วิจัยเสนอจะให้คำตอบที่ไม่แตกต่างไปจากเดิมและสามารถลดเวลาและความยุ่งยากในการอนุมานไวยากรณ์ได้

3.2 รูปแบบอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น

ในส่วนนี้จะนำเสนออัลกอริทึมสำหรับการสร้างไวยากรณ์พื้นฐาน โดยใช้ตัวอย่างเชิงบวกจากตารางความถี่ของสายอักขระเชิงบวก (ตารางที่ 2.1)

Algorithm: Probabilistic grammar inference using incremental technique

Let $\sum$ be the finite set of alphabet.

Let V be the set of variables.

Input: A finite set of all positive examples in canonically ordering

Output: A finite set of all probabilistic productions

Step 1: Initialize grammar patterns

Two types of productions can be generated by the algorithm are $A \rightarrow \lambda$ and $A \rightarrow aB$ Where A and B are variables and *a* is a character in the alphabet $\sum$ and λ be an empty string.

Step 2: Initialize the probability of the λ-production

Compute the probability of λ-production (string with length = 0) from the examples.

Step 3: Determine the probability of string with length = n – 1 (start from n = 1)

Construct production for generating strings with length = n

Consistency checking

Adjust the probability of string with length = n – 1.

Step 4: Increase n by 1

Repeat Step 3 again until the end of examples.

3.3 ขั้นตอนการอนุมานไวยากรณ์ด้วยอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น

ขั้นตอนการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้นจะเป็นการอนุมานไวยากรณ์ด้วยการสร้างไวยากรณ์พื้นฐานจากการเรียนรู้ตัวอย่างเชิงบวกและพิจารณาค่าความน่าจะเป็นจากจำนวนสายอักขระที่เกิดขึ้น ดังนั้นจากตัวอย่างเชิงบวก (ตารางที่ 2.1) จะพิจารณาค่าความน่าจะเป็นที่เกิดขึ้น ประกอบด้วย 2 ขั้นตอน ดังนี้

ขั้นตอนที่ 1 พิจารณาตัวอย่างเชิงบวกจากนั้นคำนวณหาความน่าจะเป็นของการเกิดของทุกกรณี

ตารางที่ 3.1 ตารางแสดงความน่าจะเป็นการเกิดของแต่ละตัวอย่างเชิงบวกจากตารางที่ 2.1

สายอักขระ	ความถี่	ความน่าจะเป็นการเกิด	สายอักขระ	ความถี่	ความน่าจะเป็นการเกิด
λ	490	0.4900	abaa	2	0.0020
a	128	0.1280	abab	2	0.0020
b	170	0.1700	abba	2	0.0020
aa	31	0.0310	abbb	1	0.0010
ab	42	0.0420	baaa	2	0.0020
ba	38	0.0380	baab	2	0.0020
bb	14	0.0140	baba	1	0.0010
aaa	8	0.0080	babb	1	0.0010
aab	10	0.0100	bbaa	1	0.0010
aba	10	0.0100	bbab	1	0.0010
abb	4	0.0040	bbba	1	0.0010
baa	9	0.0090	aaaaa	1	0.0010
bab	4	0.0040	aaaab	1	0.0010

ตารางที่ 3.2 ตารางแสดงความน่าจะเป็นการเกิดของแต่ละตัวอย่างเชิงบวกจากตารางที่ 2.1 (ต่อ)

สายอักขระ	ความถี่	ความน่าจะเป็นการเกิด	สายอักขระ	ความถี่	ความน่าจะเป็นการเกิด
bba	3	0.0030	aaaba	1	0.0010
bbb	6	0.0060	aabaa	1	0.0010
aaaa	2	0.002	aabab	1	0.001
aaab	2	0.002	aabba	1	0.001
aaba	3	0.003	abbaa	1	0.001
aabb	2	0.002	abbab	1	0.001

จากนั้นสร้างเครื่องจักรแบบสถานะอิงความน่าจะเป็นจากไวยากรณ์พื้นฐานด้วยอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น ตามขั้นตอนดังนี้

ขั้นตอนที่ 2 อนุมานไวยากรณ์ด้วยอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น

Step 1: Initialize grammar patterns

ขั้นตอนการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้นจะเป็นการอนุมานไวยากรณ์ด้วยการสร้างไวยากรณ์พื้นฐาน (Productions) ที่มีรูปแบบตามข้อกำหนด ดังนี้

$$A \rightarrow \lambda$$

$$A \rightarrow aB$$

โดยที่ A, B คือ ค่าตัวแปร

a คือ อักขระภายในสายอักขระ

ซึ่งการสร้างไวยากรณ์พื้นฐานนั้นจะสร้างจากการเรียนรู้ตัวอย่างเชิงบวกที่เกิดขึ้น โดยเริ่มต้นพิจารณาตัวอย่างเชิงบวกจากสายอักขระที่มีความยาว λ จะได้รูปแบบไวยากรณ์พื้นฐานที่สามารถสร้างสายอักขระความยาว λ ดังนี้

$$S \rightarrow \lambda$$

Step 2: Initialize the probability of the λ-production

จากตารางแสดงความน่าจะเป็นการเกิดของแต่ละตัวอย่างเชิงบวก (ตารางที่ 4) มีความถี่ทั้งหมดจากตัวอย่างเชิงบวกที่เกิดขึ้นจำนวน 1,000 และเป็นความถี่จากสายอักขระ λ ตัวอักษรจำนวน 490 ดังนั้นสามารถคำนวณค่าความน่าจะเป็น ดังนี้

$$prob(\lambda) = 490/1,000 = 0.49$$

Step 3: Determine the probability of string with length = 0

สายอักขระ λ เกิดจากไวยากรณ์พื้นฐาน ($S \rightarrow \lambda$) ดังนั้นค่าความน่าจะเป็นสามารถกำหนดได้ ดังนี้

$$prob(S \rightarrow \lambda) = 0.49$$

Step 4: Increase n by 1 and repeat Step 3 again until the end of examples.

ขั้นตอนนี้จะดำเนินการ Step 3 อีกครั้งโดยพิจารณาสายอักขระที่เพิ่มมากขึ้น 1 ตัวอักษร และคำนวนณค่าความน่าจะเป็นรวมของไวยากรณ์พื้นฐานที่เกิดขึ้น พร้อมกับปรับปรุงค่าความน่าจะเป็นรวมให้สอดคล้อง ซึ่งจะดำเนินการจนกว่าจะครบความยาวเชิงบวก

1) พิจารณาสายอักขระที่มีความยาว 1 ตัวอักษร คือ a และ b ดังนั้นสร้างไวยากรณ์พื้นฐานเพื่อรองรับ ดังนี้

$$prob(a) = prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$$

$$(0.128) = prob(S \rightarrow aS) \times (0.49)$$

$$prob(b) = prob(S \rightarrow bS) \times prob(S \rightarrow \lambda)$$

$$(0.17) = prob(S \rightarrow bS) \times (0.49)$$

พบว่าเราจะได้ไวยากรณ์พื้นฐานพร้อมค่าความน่าจะเป็น ดังนี้

$$prob(S \rightarrow \lambda) = 0.49$$

$$prob(S \rightarrow aS) = 0.2612$$

$$prob(S \rightarrow bS) = 0.3469$$

ซึ่งไวยากรณ์พื้นฐานทั้งหมดนี้สามารถนำไปใช้สร้างสายอักขระได้ทุกรูปแบบจากตัวอย่างเชิงบวก พร้อมค่าความน่าจะเป็นที่สามารถเกิดขึ้นได้ โดยทั่วไปค่าความน่าจะเป็นของไวยากรณ์พื้นฐานที่มีตัวแปรเดียวกันรวมกันต้องมีค่าเท่ากับ 1 แต่จากค่าที่ได้จะเห็นว่าค่าความน่าจะเป็นของไวยากรณ์พื้นฐานจากตัวแปร S รวมกันมีค่าเท่ากับ 1.0981 จึงต้องปรับค่าความน่าจะเป็นของไวยากรณ์พื้นฐานใหม่ โดยพิจารณาการปรับค่าความน่าจะเป็นของ $prob(S \rightarrow aS)$ หรือ $prob(S \rightarrow bS)$ เพื่อให้ค่าความน่าจะเป็นรวมมีค่าเป็น 1 ($prob(S \rightarrow \lambda)$ มีค่าตรงกับค่า จริง จึงไม่ต้องพิจารณา)

2) พิจารณาสายอักขระที่มีความยาว 2 ตัวอักษร คือ aa, ab, ba และ bb โดยนำไวยากรณ์พื้นฐานที่มีไปใช้

$$prob(aa) = prob(S \rightarrow aS) \times prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$$

$$(0.031) = (0.0334) \text{ มีค่าความผิดพลาด } 0.0024$$

$$prob(bb) = prob(S \rightarrow bS) \times prob(S \rightarrow bS) \times prob(S \rightarrow \lambda)$$

$$(0.014) = (0.059) \text{ มีค่าความผิดพลาด } 0.045$$

จะพบว่าการใช้ $prob(S \rightarrow bS)$ จะเกิดค่าความผิดพลาดมากกว่า $prob(S \rightarrow aS)$ ดังนั้นเราจะเพิ่มตัวแปร T และปรับค่าความน่าจะเป็นให้สอดคล้องกับค่าความน่าจะเป็นรวม ดังนี้

$$prob(S \rightarrow \lambda) = 0.49$$

$$prob(S \rightarrow aS) = 0.2612$$

$$prob(S \rightarrow bT) = 1 - (0.49 + 0.2612) = 0.2488$$

$$prob(T \rightarrow aS) = \text{ยังไม่ทราบค่า}$$

$$prob(T \rightarrow bS) = \text{ยังไม่ทราบค่า}$$

$$prob(T \rightarrow \lambda) = \text{ยังไม่ทราบค่า}$$

หาค่าความน่าจะเป็นของไวยากรณ์พื้นฐานที่ยังไม่ทราบค่า ดังนี้

$$prob(ab) = prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow \lambda)$$

$$(0.042) = (0.2612)(0.2488) \times prob(T \rightarrow \lambda)$$

$$prob(ba) = prob(S \rightarrow bT) \times prob(T \rightarrow aS) \times prob(S \rightarrow \lambda)$$

$$(0.038) = (0.2488)(0.49) \times prob(T \rightarrow aS)$$

$$prob(bb) = prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow \lambda)$$

$$(0.042) = (0.2612)(0.2488) \times prob(T \rightarrow bS)$$

พบว่าเราจะได้ไวยากรณ์พื้นฐานพร้อมค่าความน่าจะเป็น ดังนี้

$$prob(S \rightarrow \lambda) = 0.49$$

$$prob(S \rightarrow aS) = 0.2612$$

$$prob(S \rightarrow bT) = 0.2488$$

$$prob(T \rightarrow aS) = 0.3117$$

$$prob(T \rightarrow bS) = 0.1148$$

$$prob(T \rightarrow \lambda) = 0.6462$$

ค่าความน่าจะเป็นรวมของตัวแปร T มีค่ารวมกันเท่ากับ 1.0727 ซึ่งยังไม่ถูกต้อง จึงต้องปรับค่าความน่าจะเป็นของ$prob(T \rightarrow aS)$ หรือ $prob(T \rightarrow bS)$ เพื่อให้ค่าความน่าจะเป็นรวมมีค่าเป็น 1

3) พิจารณาสายอักขระที่มีความยาว 3 ตัวอักษร คือ aaa, aab, aba, abb, baa และ bbb โดยนำไวยากรณ์พื้นฐานที่มีไปใช้

$$prob(aba) = prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow aS) \times prob(S \rightarrow \lambda)$$

$$(0.01) = (0.0099) \quad \text{มีค่าความผิดพลาด } 0.0001$$

$$prob(abb) = prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow \lambda)$$

(0.004) = (0.0037) มีค่าความผิดพลาด 0.0003

$$prob(baa) = prob(S \rightarrow bT) \times prob(T \rightarrow aS) \times prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$$

(0.009) = (0.0099) มีค่าความผิดพลาด 0.0009

$$prob(bab) = prob(S \rightarrow bT) \times prob(T \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow \lambda)$$

(0.004) = (0.0125) มีค่าความผิดพลาด 0.0085

$$prob(bba) = prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$$

(0.003) = (0.037) มีค่าความผิดพลาด 0.0007

$$prob(bbb) = prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow bT) \times prob(T \rightarrow \lambda)$$

(0.006) = (0.035) มีค่าความผิดพลาด 0.0025

จะพบว่าการใช้ $prob(T \rightarrow aS)$ ทำให้เกิดค่าความผิดพลาดที่สูงในการสร้างสายอักขระ bab ดังนั้นเราจะเปลี่ยนไวยากรณ์พื้นฐาน $(T \rightarrow aS)$ เป็น $(T \rightarrow aT)$ และปรับค่าความน่าจะเป็นให้สอดคล้องกับค่าความน่าจะเป็นรวมของตัวแปร T ดังนี้

$prob(S \rightarrow \lambda) = 0.49$

$prob(S \rightarrow aS) = 0.2612$

$prob(S \rightarrow bT) = 0.2488$

$prob(T \rightarrow aT) = 1 - (0.1148 + 0.6462) = 0.239$

$prob(T \rightarrow bS) = 0.1148$

$prob(T \rightarrow \lambda) = 0.6462$

ดังนั้นเมื่อจบขั้นตอนนี้เราจะได้ไวยากรณ์พื้นฐานที่สามารถนำไปสร้างสายอักขระจากตัวอย่างเชิงบวกได้ทุกรูปแบบพร้อมด้วยค่าความน่าจะเป็น ดังนี้

ตารางที่ 3.3 ตารางแสดงการสร้างสายอักขระจากไวยากรณ์พื้นฐานพร้อมคำนวณค่าความน่าจะเป็น

สายอักขระ	ไวยากรณ์	เกิดจากไวยากรณ์พื้นฐาน	ค่าความน่าจะเป็น
λ	$(S \rightarrow \lambda)$	$prob(S \rightarrow \lambda)$	0.4900
a	$(S \rightarrow a)$	$prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$	0.1280
b	$(S \rightarrow b)$	$prob(S \rightarrow bT) \times prob(T \rightarrow \lambda)$	0.1608
aa	$(S \rightarrow aa)$	$prob(S \rightarrow aS) \times prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$	0.0334
ab	$(S \rightarrow ab)$	$prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow \lambda)$	0.0420
ba	$(S \rightarrow ba)$	$prob(S \rightarrow bT) \times prob(T \rightarrow aT) \times prob(T \rightarrow \lambda)$	0.0384
bb	$(S \rightarrow bb)$	$prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow \lambda)$	0.0140
aaa	$(S \rightarrow aaa)$	$prob(S \rightarrow aS) \times prob(S \rightarrow aS) \times prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$	0.0087
aab	$(S \rightarrow aab)$	$prob(S \rightarrow aS) \times prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow \lambda)$	0.0110
aba	$(S \rightarrow aba)$	$prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow aT) \times prob(T \rightarrow \lambda)$	0.0100
abb	$(S \rightarrow abb)$	$prob(S \rightarrow aS) \times prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow \lambda)$	0.0037
baa	$(S \rightarrow baa)$	$prob(S \rightarrow bT) \times prob(T \rightarrow aT) \times prob(T \rightarrow aT) \times prob(T \rightarrow \lambda)$	0.0092
bab	$(S \rightarrow bab)$	$prob(S \rightarrow bT) \times prob(T \rightarrow aT) \times prob(T \rightarrow bS) \times prob(S \rightarrow \lambda)$	0.0033
bba	$(S \rightarrow bba)$	$prob(S \rightarrow bT) \times prob(T \rightarrow bS) \times prob(S \rightarrow aS) \times prob(S \rightarrow \lambda)$	0.0037

ตารางที่ 3.4 ตารางแสดงการสร้างสายอักขระจากไวยากรณ์พื้นฐานพร้อมคำนวณค่าความน่าจะเป็น (ต่อ)

สายอักขระ	ไวยากรณ์	เกิดจากไวยากรณ์พื้นฐาน	ค่าความน่าจะเป็น
bbb	(S → bbb)	prob(S → bT) x prob(T → bS) x prob(S → bT) x prob(T → λ)	0.0046
aaaa	(S → aaaa)	prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → λ)	0.0023
aaab	(S → aaab)	prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → λ)	0.0029
aaba	(S → aaba)	prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → aT) x prob(T → λ)	0.0026
aabb	(S → aabb)	prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → bS) x prob(S → λ)	0.0010
abaa	(S → abaa)	prob(S → aS) x prob(S → bT) x prob(T → aT) x prob(T → aT) x prob(T → λ)	0.0024
abab	(S → abaa)	prob(S → aS) x prob(S → bT) x prob(T → aT) x prob(T → bS) x prob(S → λ)	0.0009
abba	(S → abba)	prob(S → aS) x prob(S → bT) x prob(T → bS) x prob(S → aT) x prob(T → λ)	0.0010
abbb	(S → abbb)	prob(S → aS) x prob(S → bT) x prob(T → bS) x prob(S → bT) x prob(T → λ)	0.0012
baaa	(S → baaa)	prob(S → bT) x prob(T → aT) x prob(T → aT) x prob(T → aT) x prob(T → λ)	0.0022

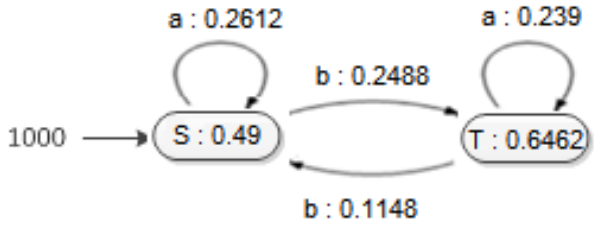
ตารางที่ 3.5 ตารางแสดงการสร้างสายอักขระจากไวยากรณ์พื้นฐานพร้อมคำนวณค่าความน่าจะเป็น (ต่อ)

สายอักขระ	ไวยากรณ์	เกิดจากไวยากรณ์พื้นฐาน	ค่าความน่าจะเป็น
baab	(S → baab)	prob(S → bT) x prob(T → aT) x prob(T → aT) x prob(T → bS) x prob(S → λ)	0.0008
baba	(S → baba)	prob(S → bT) x prob(T → aT) x prob(T → bS) x prob(S → aS) x prob(S → λ)	0.0009
babb	(S → babb)	prob(S → bT) x prob(T → aT) x prob(T → bS) x prob(S → bT) x prob(T → λ)	0.0011
bbaa	(S → abba)	prob(S → bT) x prob(T → bS) x prob(S → aS) x prob(S → aS) x prob(S → λ)	0.0010
bbab	(S → bbab)	prob(S → bT) x prob(T → bS) x prob(S → aS) x prob(S → bT) x prob(T → λ)	0.0012
bbba	(S → bbba)	prob(S → bT) x prob(T → bS) x prob(S → bT) x prob(T → aT) x prob(T → λ)	0.0011
aaaaa	(S → aaaaa)	prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → λ)	0.0006
aaaab	(S → aaaab)	prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → λ)	0.0007
aaaba	(S → aaaba)	prob(S → aS) x prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → aT) x prob(T → λ)	0.0007
aabaa	(S → aabaa)	prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → aT) x prob(T → aT) x prob(T → λ)	0.0006

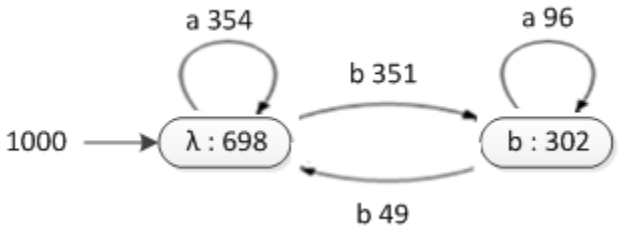
ตารางที่ 3.6 ตารางแสดงการสร้างสายอักขระจากไวยากรณ์พื้นฐานพร้อมคำนวณค่าความน่าจะเป็น (ต่อ)

สายอักขระ	ไวยากรณ์	เกิดจากไวยากรณ์พื้นฐาน	ค่าความน่าจะเป็น
aabab	(S → aabab)	prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → aT) x prob(T → bS) x prob(S → λ)	0.0002
aabba	(S → aabba)	prob(S → aS) x prob(S → aS) x prob(S → bT) x prob(T → bS) x prob(S → aS) x prob(S → λ)	0.0002
abbaa	(S → abbaa)	prob(S → aS) x prob(S → bT) x prob(T → bS) x prob(S → aS) x prob(S → aS) x prob(S → λ)	0.0002
abbab	(S → abbab)	prob(S → aS) x prob(S → bT) x prob(T → bS) x prob(S → aS) x prob(S → bT) x prob(T → λ)	0.0003

เมื่อได้ไวยากรณ์พื้นฐานพร้อมความน่าจะเป็นที่สามารถสร้างสายอักขระจากตัวอย่างเชิงบวกได้ครบทุกรูปแบบแล้ว จะนำไปสร้างเครื่องจักรแบบสถานะเชิงความน่าจะเป็นเพื่อนำไปเปรียบเทียบกับวิธีอัลเลอเจียร์



ภาพที่ 3.1 เครื่องจักรแบบสถานะเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้น



ภาพที่ 3.2 เครื่องจักรแบบสถานะเชิงความน่าจะเป็นด้วยขั้นตอนวิธีอัลเลอเจียร์

บทที่ 4
การทดสอบและวิเคราะห์ผล

4.1 วัตถุประสงค์ของการทดสอบ

เพื่อสนับสนุนการทดสอบอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้นที่ได้พัฒนาในบทที่ 4 มีผลการทดสอบที่มีประสิทธิภาพ โดยพิจารณาความแม่นยำด้วยการอาศัยค่าความน่าจะเป็นของการนำไวยากรณ์พื้นฐานไปใช้ในการสร้างสายอักขระในแต่ละรูปแบบเปรียบเทียบกับอัลกอริทึมอัลเลอเจียร์

4.2 สรุปผลการทดสอบ

4.2.1 การเปรียบเทียบเชิงความน่าจะเป็น

ตารางที่ 4.1 ตารางแสดงค่าความน่าจะเป็นที่เกิดขึ้นจริงจากตัวอย่างเชิงบวก, อัลกอริทึมอัลเลอเจียร์และอัลกอริทึมเทคนิคการเพิ่มขึ้น

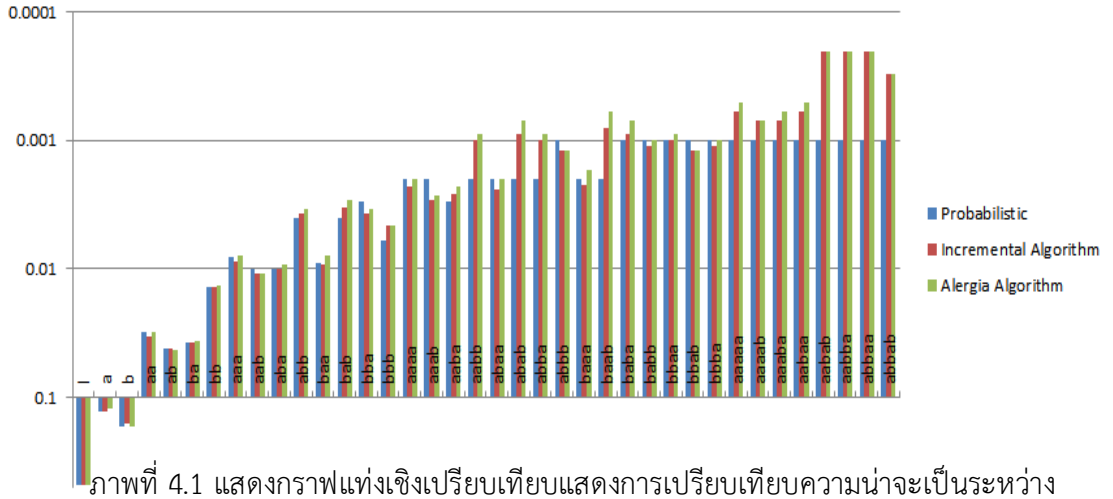
สายอักขระจากตัวอย่างเชิงบวก		ค่าความน่าจะเป็นด้วย Alergia Algorithm	ค่าความน่าจะเป็นด้วย Incremental Algorithm	ผลการเปรียบเทียบ
สายอักขระ	ค่าความน่าจะเป็น			
λ	0.4900	0.4905	0.4900	เทคนิคการเพิ่มขึ้น
a	0.1280	0.1238	0.1280	เทคนิคการเพิ่มขึ้น
b	0.1700	0.1690	0.1608	อัลเลอเจียร์
aa	0.0310	0.0312	0.0334	อัลเลอเจียร์
ab	0.0420	0.0426	0.0420	เทคนิคการเพิ่มขึ้น

ตารางที่ 4.2 ตารางแสดงค่าความน่าจะเป็นที่เกิดขึ้นจริงจากตัวอย่างเชิงบวก, อัลกอริทึมอัลเลอเจียร์และอัลกอริทึมเทคนิคการเพิ่มขึ้น (ต่อ)

สายอักขระจากตัวอย่างเชิงบวก		ค่าความน่าจะเป็นด้วย Alergia Algorithm	ค่าความน่าจะเป็นด้วย Incremental Algorithm	ผลการเปรียบเทียบ
bb	0.0140	0.0135	0.0140	เทคนิคการเพิ่มขึ้น
aaa	0.0080	0.0079	0.0087	อัลเลอเจียร์
aab	0.0100	0.0108	0.0110	อัลเลอเจียร์
aba	0.0100	0.0092	0.0100	เทคนิคการเพิ่มขึ้น
abb	0.0040	0.0034	0.0037	เทคนิคการเพิ่มขึ้น
baa	0.0090	0.0078	0.0092	เทคนิคการเพิ่มขึ้น
bab	0.0040	0.0029	0.0033	เทคนิคการเพิ่มขึ้น
bba	0.0030	0.0034	0.0037	อัลเลอเจียร์
bbb	0.0060	0.0046	0.0046	เท่ากัน
aaaa	0.0020	0.0020	0.0023	อัลเลอเจียร์
aaab	0.0020	0.0027	0.0029	อัลเลอเจียร์
aaba	0.0030	0.0023	0.0026	เทคนิคการเพิ่มขึ้น
aabb	0.0020	0.0009	0.0010	เทคนิคการเพิ่มขึ้น
abaa	0.0020	0.0020	0.0024	อัลเลอเจียร์
abab	0.0020	0.0007	0.0009	เทคนิคการเพิ่มขึ้น
abba	0.0020	0.0009	0.0010	เทคนิคการเพิ่มขึ้น
abbb	0.0010	0.0012	0.0012	เท่ากัน
baaa	0.0020	0.0017	0.0022	เทคนิคการเพิ่มขึ้น

ตารางที่ 4.3 ตารางแสดงค่าความน่าจะเป็นที่เกิดขึ้นจริงจากตัวอย่างเชิงบวก, อัลกอริทึมอัลเลอเจียร์และอัลกอริทึมเทคนิคการเพิ่มขึ้น (ต่อ)

baab	0.0020	0.0006	0.0008	เทคนิคการเพิ่มขึ้น
baba	0.0010	0.0070	0.0009	เทคนิคการเพิ่มขึ้น
babb	0.0010	0.0010	0.0011	อัลเลอเจียร์
bbaa	0.0010	0.0090	0.0010	เทคนิคการเพิ่มขึ้น
bbab	0.0010	0.0012	0.0012	เท่ากัน
bbba	0.0010	0.0010	0.0011	อัลเลอเจียร์
aaaaa	0.0010	0.0050	0.0006	เทคนิคการเพิ่มขึ้น
aaaab	0.0010	0.0007	0.0007	เท่ากัน
aaaba	0.0010	0.0006	0.0007	เทคนิคการเพิ่มขึ้น
aabaa	0.0010	0.0005	0.0006	เทคนิคการเพิ่มขึ้น
aabab	0.0010	0.0002	0.0002	เท่ากัน
aabba	0.0010	0.0002	0.0002	เท่ากัน
abbaa	0.0010	0.0002	0.0002	เท่ากัน
abbab	0.0010	0.0003	0.0003	เท่ากัน



ภาพที่ 4.1 แสดงกราฟแท่งเชิงเปรียบเทียบแสดงการเปรียบเทียบความน่าจะเป็นระหว่างอัลกอริทึมอัลเลอเจียร์และอัลกอรึทึมเทคนิคการเพิ่มขึ้น

จากการทดลองในบทที่ 3 จะสามารถเปรียบเทียบค่าความน่าจะเป็นระหว่างอัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้นและอัลกอริทึมอัลเลอเจียร์ ซึ่งมี 20 รูปแบบในตัวอย่างเชิงบวกที่อัลกอริทึมการอนุมานไวยากรณ์สม่ำเสมอเชิงความน่าจะเป็นด้วยเทคนิคการเพิ่มขึ้นมีค่าความน่าจะเป็นใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริงมากกว่าอัลกอริทึมอัลเลอเจียร์ มีเพียง 10 รูปแบบในตัวอย่างเชิงบวกที่อัลกอริทึมอัลเลอเจียร์ มีค่าความน่าจะเป็นใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริงมากกว่า และมี 8 รูปแบบที่มีค่าความน่าจะเป็นใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริงเท่ากัน

4.2.2 การเปรียบเทียบความซับซ้อนเชิงเวลา

ในหัวข้อนี้จะขอพิจารณาความซับซ้อนเชิงเวลาของอัลกอริทึมที่นำเสนอในงานวิจัยนี้ เทียบกับอัลกอริทึมอัลเลอเจียร์ โดยพิจารณาดังนี้

กำหนดให้ n แทนจำนวนของตัวอย่างบวกที่ใช้ในการอนุมานไวยากรณ์

m แทนความยาวสูงสุดของตัวอย่างบากที่ใช้ในการอนุมานไวยากรณ์

P แทนจำนวนตัวอ้กขระที่แตกต่างกันทั้งหมด

พิจารณาอัลกอริทึมอัลเลอเจียร์ จะได้ว่า

1. ต้นไม้ที่ถูกสร้างในตอนเริ่มต้นจะมีความลึกสูงสุด เท่ากับ m และแต่ละโหนดในต้นไม้จะแตกกิ่งได้ P กิ่ง (เท่ากันจำนวนตัวอักขระ)

2. การพิจารณาลดรูปต้นไม้ จะเริ่มจากแต่ละโหนดของต้นไม้ (เริ่มจากราก) จะพิจารณาเทียบกับต้นไม้ย่อยทุกต้นที่เป็นไปได้ ในกรณีของการพิจารณาราก ต้นไม้ย่อยจะมีขนาดเท่ากับ $P^0 + P^1 + P^2 + \ldots + P^{m-1}$
3. ในการผสานสถานะของโหนดจะพิจารณาการยุบรวมทีละคู่ ดังนั้นหากมี P รูปแบบจะต้องพิจารณาโดย $\binom{P}{2} = \frac{P!}{2!(P-2)!} = \frac{P(P-1)}{2} \sim P^2$
4. ดังนั้นจะต้องพิจารณาการผสานสถานะทั้งหมด $(P^2)(P^m - 1) \sim P^{m+2} - 1$
5. การพิจารณาจะทำทีละระดับ จากรากไปสิ้นสุดที่ใบของต้นไม้ทั้งหมด ดังนั้นจำนวนต้นไม้ย่อยมากที่สุดที่ต้องพิจารณา คือ $P^{m+1} + P^m + \ldots + P^2 + P^1 + 1 = P^{m+2} - 1$ ซึ่งสามารถหาค่าได้ $P^{m+2} - P - 2$

ดังนั้น ความซับซ้อนเชิงเวลาของการทำงานจะเป็น $O(P^{m+2} - P - 2)$ หรือ $O(P^{m+2})$

พิจารณาอัลกอริทึมที่นำเสนอในงานวิจัยนี้

1. พิจารณาสร้างกฎไวยากรณ์จากสมาชิกตัวที่สั้นที่สุด เริ่มที่ความยาว 0
2. จำนวนตัวอย่างเชิงบวกทุกจำนวนต้องถูกพิจารณา
3. เมื่อมีการเพิ่มตัวแปร เป็นการเพิ่มโหนด และ ต้องคำนวณค่าความน่าจะเป็นเดิม ดังนั้นแต่ละระดับ จะเป็นการพิจารณา 2 รอบ ดังนั้น จะต้องพิจารณา 2n

ดังนั้น ความซับซ้อนเชิงเวลาของการทำงานจะเป็น $O(2n)$

เมื่อพิจาณาความซับซ้อนเชิงเวลาของอัลกอริทึมของ อัลเลอเจียร์จะใช้ความยาวสูงสุดของตัวอย่างเชิงบวก (m) แต่ อัลกอริทึมที่นำเสนอในงานวิจัยใช้ จำนวนของตัวอย่างเชิงบวก (n) จึงต้องหาความพิจารณาความสัมพันธ์ ระหว่าง m และ n จะพบว่า $(P^{m+2}) > 2n$ เสมอ เช่น เมื่อพิจารณาสายอักขระที่ยาว 2 ตัวอักษร จะได้ค่า m = 2 และ n = 7 (λ, a, b, aa, ab, ba, bb) เมื่อแทนค่าจะได้ (P^{2+2}) = 16 และ 2(7) = 14 ซึ่ง 16 > 14 นั่นเอง จึงแสดงให้เห็นว่า2n มีอัตราการเติบโตน้อยกว่า ดังนั้นจากตัวอย่างที่นำมาศึกษาอัลกอริทึมที่นำเสนอมีความซับซ้อนเชิงเวลาน้อยกว่าอัลกอริทึมอัลเลอเจียร์

บทที่ 5
สรุปผลการวิจัย

บทที่ 5 จะกล่าวถึงการสรุปผลวิจัยรวมถึงข้อจากัดในงานวิจัยชิ้นนี้ และแนวทางสาหรับการวิจัยต่อในอนาคต โดยจะกล่าวดังต่อไปนี้

5.1 สรุปผลการวิจัย

การอนุมานไวยากรณ์เป็นวิธีการที่ได้รับความสนใจเป็นอย่างมากที่นำมาช่วยในด้านการศึกษาเรียนรู้และอธิบายกระบวนการทำงานทั้งหมดของแบบจำลองทางด้านซอฟต์แวร์ การอนุมานไวยากรณ์จะเป็นการนำเอาภาษารูปนัยมาช่วยในการศึกษากระบวนการที่เกิดขึ้นตามลำดับและสร้างข้อกำหนดขั้นตอนการทำงาน นำมาอธิบายในรูปของแบบจำลอง ซึ่งวิธีการอนุมานไวยากรณ์ที่ได้รับความนิยมและน่าเชื่อถือคือวิธีการอัลเลอเจียร์ เนื่องจากวิธีการนี้มีความแม่นยำและมีค่าความน่าจะเป็นจากแบบจำลองที่ใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริง ขั้นตอนวิธีอัลเลอเจียร์ นั้นเริ่มต้นด้วยการนำตัวอย่างเชิงบวกมาแสดงในรูปของต้นไม้อิงคำนำหน้า จากนั้นใช้ขอบเขตของโฮฟฟ์ดิ่งเพื่อคำนวณหาความสอดคล้องกันของสถานะ เมื่อได้สถานะที่มีความสอดคล้องกันจะทำการผสานสถานะจนสุดท้ายแสดงออกมาในรูปของเครื่องจักรแบบสถานะอิงความน่าจะเป็น จากการศึกษาขั้นตอนวิธีอัลเลอเจียร์ พบว่าเป็นวิธีการที่มีความซับซ้อนและใช้เวลาในการคำนวณค่อนข้างสูง ดังนั้นงานวิจัยนี้จึงนำเสนออัลกอริทึมที่ช่วยในการอนุมานไวยากรณ์ที่มีความแม่นยำและใช้เวลาที่รวดเร็วกว่าขั้นตอนวิธีอัลเลอเจียร์ โดยกำหนดรูปแบบเป็นไวยากรณ์สม่ำเสมอ ซึ่งการออกแบบอัลกอริทึมนั้นเริ่มต้นจากการพิจารณาตัวอย่างเชิงบวกที่มีสายอักขระสั้นที่สุดและเพิ่มขนาดความยาวขึ้นครั้งละ 1 ตัวอักษร โดยการพิจารณาแต่ละรอบจะมีการสร้างตัวแปรเพื่อนำไปสร้างไวยากรณ์พื้นฐานที่สามารถอธิบายการเกิดตัวอย่างเชิงบวกได้ทุกรูปแบบ พร้อมทั้งคำนวณค่าความน่าจะเป็นที่เกิดขึ้นให้สอดคล้องกัน และผลสุดท้ายนำเสนอออกมาในรูปของเครื่องจักรแบบสถานะอิงความน่าจะเป็น ดังที่แสดงในบทที่ 3

จากการออกแบบและนำไปใช้งานในบทที่ 3 สามารถนำผลการทดสอบมาวิเคราะห์ได้ในบทที่ 4 ซึ่งจะวิเคราะห์ประสิทธิภาพ 2 ด้าน คือ ความแม่นยำ ซึ่งความแม่นยำจะพิสูจน์ได้จากค่าความน่าจะเป็นที่ได้คำนวณได้จากอัลกอริทึมมีค่าใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริงมากกว่าขั้นตอนวิธีอัลเลอเจียร์ และวิเคราะห์ความซับซ้อนเชิงเวลา ซึ่งใช้เวลาน้อยกว่าขั้นตอนวิธีอัลเลอเจียร์เช่นกัน

จากผลการทดสอบและนำมาวิเคราะห์ประสิทธิภาพที่ได้จากตัวอย่างเชิงบวกที่ทดลองพบว่า อัลกอริทึมอัลเลอเจียร์ จะมีความซับซ้อนเชิงเวลามาก ซึ่งจากการศึกษาพบว่าเวลาที่สูญเสียเกิดจากการที่นำตัวอย่างเชิงบวกมาสร้างเป็นต้นไม้อิงคำนำหน้า (prefix tree acceptance) หากตัวอย่างเชิงบวกมีความยาวของสายอักขระที่มาก ขนาดของต้นไม้ก็จะใหญ่ตามไปด้วย และเมื่อขั้นตอนการตรวจสอบความสอดคล้องด้วยขอบเขตของโฮฟฟ์ดิ่งพร้อมกับผสานสถานะจะต้องทำหลายรอบตามไปด้วย แต่อัลกอริทึมเทคนิคการเพิ่มขึ้นที่ได้นำเสนอจะเริ่มต้นพิจารณาสายอักขระที่มีความยาวสั้นที่สุดสร้างไวยากรณ์พื้นฐานที่สามารถรองรับการสร้างสายอักขระได้ทุกรูปแบบตั้งแต่แรก รอบการพิจารณาจะขึ้นอยู่กับการปรับค่าความน่าจะเป็นรวมของแต่ละตัวแปรให้มีค่าเท่ากับ 1 ดังนั้นเมื่อตัวอย่างเชิงบวกที่นำมาใช้มีความยาวมาก อัลกอริทึมเทคนิคการเพิ่มขึ้นจะเป็นทางเลือกที่ดีกว่าในการพิจารณาใช้งาน

ขั้นตอนเรียนรู้และอนุมานไวยากรณ์นั้นสามารถนำไปใช้ประโยชน์ได้ในด้านของวิศวกรรมซอฟต์แวร์ กล่าวคือ การวิเคราะห์ระบบ เนื่องจากกระบวนการพัฒนาโปรแกรมนั้นจะต้องมีขั้นตอนการออกแบบระบบ ดังนั้นหากเราสามารถวิเคราะห์ได้ว่าระบบควรทำงานอย่างไร จะสามารถพัฒนาโปรแกรมได้อย่างมีประสิทธิภาพ เราสามารถนำกระบวนการที่เกิดขึ้นของระบบที่ต้องการมาเรียนรู้และนำอัลกอริทึมที่นำเสนอมาใช้ในการสร้างแบบจำลองการทำงานของโปรแกรมพร้อมค่าความน่าจะเป็นของลำดับกระบวนการที่สามารถเกิดขึ้นได้ ซึ่งวิธีการทั้งหมดเราจะสามารถออกแบบระบบการทำงานของโปรแกรมที่จะพัฒนาได้ตามที่ต้องการ

5.2 ข้อจำกัดในงานวิจัย

1) ตัวอย่างที่นำมาใช้งานวิจัยเป็นตัวอย่างเชิงบวกเท่านั้น
2) ไวยากรณ์ที่ใช้จะอยู่ในรูปแบบของไวยากรณ์สม่ำเสมอ
3) อัลกอริทึมที่นำเสนอจะพิจารณาค่าความน่าจะเป็นจากสายอักขระที่มีความยาวสั้นที่สุดก่อน ดังนั้นหากรูปแบบของตัวอย่างเชิงบวกมีไม่ครบทุกรูปแบบจะไม่สามารถคำนวณหาค่าความน่าจะเป็นของสายอักขระที่ยาวขึ้นได้

5.3 แนวทางสำหรับการวิจัยต่อในอนาคต

1) เปรียบเทียบผลการอนุมานไวยากรณ์กับอัลกอริทึมอื่นนอกเหนือจากอัลกอริทึมอัลเลอเจียร์

2) พิจารณาตัวอย่างเชิงบวกที่วิธีการของอัลกอริทึมอัลเลอเจียร์มีค่าความน่าจะเป็นใกล้เคียงกับค่าความน่าจะเป็นที่เกิดขึ้นจริงมากกว่าและหาข้อแก้ไข

รายการอ้างอิง

1. Sakakibara, Y., Recent advances of grammatical inference, Theoretical Computer Science 1997: p. 15-45.
2. Honavar, V. and G. Slutzki, Grammatical inference, No.1443 in Lecture Notes in Artificial Intelligence, Ames, Iowa, Springer-Verlag. 1998.
3. Onica, J. and P. Garcia, Identifying regular languages in polynomial times. Advances in Structural and Syntactic Pattern Recognition. 1992: p. 99-108.
4. Lawrence, S., S. Fong, and C.L. Giles, Natural language grammatical inference: a comparison of recurrent neural networks and machine learning methods. Connectionist, Statistical, and Symbolic Approaches to learning for natural language processing, S. Wermter, E. Riloff, and G. Scheler, eds., . 1996.
5. Sakakibara, Y., Grammatical inference in bioinformatics. IEEE Transaction on Pattern Analysis and Machine Intelligence 2005.
6. Denis, F., Learning regular languages from simple positive examples in Machine Learning 2000.
7. Carrasco, R. and J. Oncina, Learning stochastic regular grammars by means of a state merging method. Lecture Notes in Artificial Intelligence 1994: p. 139-150.
8. Sutapa, D. and Subhasis, A Grammar Inference Approach for Predicting Kinase Specific Phosphorylation Sites. DOI:10.1371/journal.pone.0122294. 2015.
9. Hoeffding, W., Probability inequalitties for sums of bounded random variables. American Statistical Association Journal. 1963: p. 13-30.
10. Higuera, C.D., Grammatical Inference, Cambridge University Press, New York. 2010.
11. Pinprasert, K., Probabilistic Deterministic Finite Automaton Learning Model.

ภาคผนวก

ประวัติผู้เขียนวิทยานิพนธ์

นาย ต่อศักดิ์ เพ็ญภินันท์ เกิดเมื่อวันที่ 30 กรกฎาคม พ.ศ. 2529 ที่จังหวัดสุรินทร์ สาเร็จการศึกษาปริญญาตรีหลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิชาการพัฒนาซอฟต์แวร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2557