การหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม

DETERMINATION OF LONGEST COMMON SUBSEQUENCE USING BIGRAM ANALYSIS

บดินทร์ แสงอรุณ
BODIN SANGAROON

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2550

KMITL-2007-SC-M-002-109

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม

DETERMINATION OF LONGEST COMMON SUBSEQUENCE USING BIGRAM ANALYSIS

บดินทร์ แสงอรุณ

BODIN SANGAROON

เลขหมู่.....
เลขทะเบียน 77960
วัน,เดือน,ปี 12 ก.พ. 2551

b 11880813
i.....

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2550

KMITL-2007-SC-M-002-109

DETERMINATION OF LONGEST COMMON SUBSEQUENCE USING BIGRAM ANALYSIS

BODIN SANGAROON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN COMPUTER SCIENCE

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

KMITL-2007-SC-M-002-109

COPYRIGHT 2007

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม

Determination of Longest Common Subsequence Using Bigram Analysis

ชื่อนักศึกษา นายบดินทร์ แสงอรุณ

รหัสประจำตัว 48067507

ปริญญา วิทยาศาสตรมหาบัณฑิต

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.วีระ บุญจริง

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.จีรพร	ศรีสวัสดิ์	
รศ.ดร.วีระ	บุญจริง	
ผศ.ดร.ศรัณย์	อินทโกสุม	
ดร.เฉลิมศักดิ์	เลิศวงศ์เสถียร	

วัน / เดือน/ ปี ที่สอบ 25 กันยายน 2550 เวลา 16.00 น. เป็นต้นไป

สถานที่สอบ ณ อาคารจุฬาภรณวลัยลักษณ์ 1 ห้อง 217

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.จารุวัตร เจริญสุข)

คณบดีบัณฑิตวิทยาลัย

วันที่.....๒๕.....**เดือน**.....กันยายน.....**พ.ศ.**.....๒๕๕๐.....

หัวข้อวิทยานิพนธ์	การหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม
นักศึกษา	นายบดินทร์ แสงอรุณ
รหัสประจำตัว	48067507
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.วีระ บุญจริง

บทคัดย่อ

งานวิจัยนี้เสนอขั้นตอนวิธีใหม่สำหรับหาสตริงย่อยและลำดับย่อยจากเซตของสตริงโดยใช้วิธีการวิเคราะห์ไบแกรม โดยมีวัตถุประสงค์เพื่อให้ได้ขั้นตอนวิธีที่ใช้ในการหาคำตอบจากชุดของสตริงที่มีขนาดใดๆ เพื่อลดจำนวนครั้งในการเปรียบเทียบเมื่อเทียบกับขั้นตอนวิธีแบบที่มีอยู่ งานวิจัยได้ทำการพัฒนาขั้นตอนวิธีที่ใช้การวิเคราะห์ไบแกรมแล้วทำการปรับปรุงขั้นตอนวิธีที่ได้นี้โดยใช้โครงสร้างข้อมูลซัฟฟิกต์อะเรย์ จากนั้นจึงทำการทดลองขั้นตอนวิธีใหม่นี้กับข้อมูลชุดต่างๆ เพื่อนับจำนวนครั้งในการเปรียบเทียบเทียบกับจำนวนครั้งในการเปรียบเทียบที่ได้จากการใช้ขั้นตอนวิธีเดิม ผลการทดลองแสดงว่า จำนวนครั้งในการเปรียบเทียบของขั้นตอนวิธีแบบใหม่น้อยกว่าของวิธีที่มีอยู่เดิมเมื่อสตริงต่างๆ ที่นำเข้าเป็นสตริงที่ไม่คล้ายกัน

Thesis	Determination of Longest Common Subsequence Using Bigram Analysis
Student	Mr. Bodin Sangaroon
Student ID.	48067507
Degree	Master of Science
Program	Computer Science
Year	2007
Thesis Advisor	Assoc.Prof.Dr.Veera Boonjing

ABSTRACT

This research proposes a new algorithm for determining longest common substring and subsequence from given set of strings using bi-gram analysis. Its two purposes are to determine such a substring or subsequence from more than 2 strings at a time and to reduce numbers of comparisons compared with existing algorithms. The research developed two algorithms: the bi-gram algorithm and the bi-gram suffix-array algorithm. The later is the improved version of the former. It then made experiments on the improved one on different set of data to determine number of comparisons compared with existing algorithms. The results show that number of comparisons of the new algorithm is less than of the existing algorithm when strings in comparison are not similar.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้มิอาจจะสำเร็จลุล่วงไปได้ด้วยดี หากมิได้รับคำแนะนำ คำชี้แจง ความรู้ และความเอาใจใส่จาก รศ. ดร. วีระ บุญจริง ผู้เป็นอาจารย์ที่ปรึกษา ซึ่งท่านได้สละเวลาให้กับข้าพเจ้าอย่างเต็มที่ จึงใคร่ขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ ผศ. ดร. จีรพร ศรีสวัสดิ์ ผศ. ดร. ศรัณย์ อินทโกสุม และ ดร. เฉลิมศักดิ์ เลิศวงศ์เสถียร คณะกรรมการสอบหัวข้อและโครงร่างวิทยานิพนธ์ที่กรุณาให้คำแนะนำตลอดจนข้อชี้แนะจนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบพระคุณ ดร.จุพารัตน์ ตันประเสริฐ ที่เปิดโอกาสให้ข้าพเจ้ามาได้มาศึกษาและช่วยสนับสนุนให้การศึกษาเป็นไปได้ด้วยดีโดยตลอด

ขอขอบคุณเพื่อน ๆ พี่ ๆ ที่ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ และ เพื่อน ๆ พี่ ๆ ประจำภาควิชาวิทยาการคอมพิวเตอร์ ทุกท่านที่ค่อยให้กำลังใจและช่วยให้ความช่วยเหลือจนวิทยานิพนธ์ฉบับนี้สำเร็จได้

ขอขอบคุณ คุณ ปันหยี นุ้ยสิน ที่เป็นกำลังใจ คอยสนับสนุน และให้ความช่วยเหลือให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จได้

และสุดท้ายนี้ขอบคุณบิดา และมารดา ที่สนับสนุนให้ได้ศึกษาในระดับที่ตั้งใจ อีกทั้งยังให้การดูแลในเรื่องค่าใช้จ่ายต่างๆ ระหว่างทำการศึกษาเป็นอย่างดี

สำหรับคุณงามความดี และประโยชน์อันใดที่เกิดขึ้นจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดา มารดา อาจารย์ทุกท่านซึ่งเป็นที่เคารพรักยิ่ง ตลอดจนญาติพี่น้อง และเพื่อนๆ ทุกคน

บดินทร์ แสงอรุณ

พฤศจิกายน 2550

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VI

สารบัญรูป VIII

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา 2

1.3 สมมติฐานของการศึกษา 2

1.4 ขอบเขตการศึกษา 2

1.5 ส่วนประกอบของวิทยานิพนธ์ 3

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 4

2.1 การหาอักขระย่อยและลำดับเหมือนยาวที่สุดในชุดอักขระ 4

2.2 วิธีการหาอักขระย่อยและลำดับเหมือนยาวที่สุดในชุดอักขระ 5

2.2.1 การใช้โปรแกรมแบบพลวัต 5

2.2.2 การใช้ซัฟฟิกซ์ทรี 9

2.2.3 การใช้ซัฟฟิกซ์อะเรย์ 13

2.3 แบบจำลองภาษาไบแกรม 17

บทที่ 3 การหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม 19

3.1 วิธีการใช้ไบแกรมในการแบ่งหน่วยของชุดอักขระ 19

3.2 การเปรียบเทียบโดยใช้การพิจารณาจากลำดับ 21

3.3 การเปรียบเทียบโดยใช้โครงสร้างของซัฟฟิกซ์อะเรย์ 26

สารบัญ (ต่อ)

หน้า

บทที่ 4 การประเมินผล 30

4.1 ข้อมูลที่ใช้ในการทดลอง 30

4.2 เครื่องมือที่ใช้ในการทดลอง 32

4.3 การทดลอง 32

4.4 ผลทดลอง 33

4.4.1 การเปรียบเทียบหาอักขระย่อยเหมือนยาวที่สุด 33

4.4.2 การเปรียบเทียบหาลำดับเหมือนย่อยที่สุด 38

บทที่ 5 สรุปและข้อเสนอแนะ 43

5.1 สรุป 43

5.2 ข้อเสนอแนะ 44

เอกสารอ้างอิง 45

ประวัติผู้เขียน 46

สารบัญตาราง

ตารางที่ หน้า

2.1 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.3 6
2.2 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.4 7
2.3 ผลลัพธ์ชุดที่สองในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.4 7
2.4 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.5 8
2.5 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.6 9
2.6 ผลลัพธ์ชุดที่สองในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.6 9
2.7 ผลลัพธ์การแจงซัฟฟิกซ์จากตัวอย่างที่ 2.7 10
2.8 ผลลัพธ์การแจงซัฟฟิกซ์จากตัวอย่างที่ 2.8 11
2.9 ผลลัพธ์การแจงซัฟฟิกซ์จากตัวอย่างที่ 2.9 13
2.10 เก็บค่าชุดอักขระ I ลงซัฟฟิกซ์อะเรย์ 14
2.11 เก็บค่าชุดอักขระ J ลงซัฟฟิกซ์อะเรย์ 14
2.12 การเรียงลำดับตัวอักษระของชุดอักขระ I 15
2.13 การเรียงลำดับตัวอักษระของชุดอักขระ J 15
2.14 การเรียงลำดับตัวอักษรของชุดอักขระ I และ J รวมกัน 16
3.1 ตารางผลลัพธ์การแบ่งไบแกรมจากตัวอย่างที่ 3.1 20
3.2 ผลลัพธ์การแบ่งหน่วยของชุดอักขระที่ 1 20
3.3 ผลลัพธ์การแบ่งหน่วยของชุดอักขระที่ 2 21
3.4 ตารางการเปรียบเทียบชุดอักขระลำดับที่ 1 และ 2 22
3.5 เลือกชุดอักขระที่เหมือนกัน 23
3.6 อักขระชุดที่ 1 24
3.7 อักขระชุดที่ 2 24
3.8 ตารางการเปรียบเทียบชุดอักขระลำดับที่ 1 และ 2 25
3.9 ตัวอย่างการจัดซัฟฟิกซ์อะเรย์ของชุดอักขระที่ 1 27
3.10 ตัวอย่างการจัดซัฟฟิกซ์อะเรย์ของชุดอักขระที่ 2 28
3.11 การตัดชุดอักษรที่มีดัชนีซ้ำกันของชุดอักขระที่ 1 28
3.12 การตัดชุดอักษรที่มีดัชนีซ้ำกันของชุดอักขระที่ 2 28
3.13 ผลลัพธ์จากการเปรียบเทียบซัฟฟิกซ์อะเรย์ทั้งหมด 29

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

4.1 ผลลัพธ์จำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัตและแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาอักขระย่อยเหมือนยาวที่สุด 34

4.2 ตารางอัตราส่วนร้อยละจากจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัตและแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาอักขระย่อยเหมือนยาวที่สุด 35

4.3 ผลลัพธ์จำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัตและแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด 36

4.4 ตารางอัตราส่วนร้อยละจากจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัตและแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด..... 37

4.5 ผลลัพธ์จำนวนครั้งในการเปรียบเทียบแบบโครงสร้างซัฟฟิกซ์อะเรย์กับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด 38

4.6 ตารางอัตราส่วนร้อยละจากจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัตและแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด..... 39

4.7 ตารางจำนวนครั้งในการเปรียบเทียบของชุดอักขระดีเอ็นเอของทั้ง 2 วิธี 40

4.8 สรุปข้อดีข้อเสียของการวิเคราะห์ทั้ง 3 วิธี..... 42

สารบัญรูป

รูปที่		หน้า
2.1	ซัฟฟิกซ์ทรี ของชุดอักขระทั้งสองชุด	11
2.2	ซัฟฟิกซ์ทรี ของชุดอักขระทั้งสามชุด	12
4.1	กราฟจำนวนครั้งในการเปรียบเทียบการหาอักขระย่อยเหมือนยาวที่สุด	35
4.2	กราฟจำนวนครั้งในการเปรียบเทียบการหาลำดับเหมือนยาวที่สุด	37
4.3	กราฟจำนวนครั้งในการเปรียบเทียบการหาลำดับเหมือนยาวที่สุด	38
4.4	กราฟจำนวนครั้งในการเปรียบเทียบของวิธีคิดทั้ง 2 วิธี	41

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเปรียบเทียบตัวอักขระ (String Matching) เป็นวิธีการที่ใช้แก้ปัญหาเกี่ยวกับการหาความแตกต่างของตัวอักขระที่สนใจ โดยการพิจารณาเพื่อหาความแตกต่างของตัวอักขระที่สนใจนั้นอาจทำโดยการเปรียบเทียบความคล้ายคลึงกันของชุดอักขระเพื่อหาชุดอักขระที่มีความคล้ายคลึงกันมากที่สุด หรือทำโดยการเปรียบเทียบเพื่อหาชุดอักขระที่มีความแตกต่างกันมากที่สุด ก็ได้

วิธีการเปรียบเทียบตัวอักขระนั้นมีด้วยกันหลายวิธี สำหรับในวิทยานิพนธ์ฉบับนี้จะนำเสนอวิธีการเปรียบเทียบตัวอักขระเพื่อให้ได้ตัวอักขระที่มีความเหมือนกันและมีความยาวมากที่สุดโดยใช้วิธีการเปรียบเทียบทั้งหมด 2 วิธี ได้แก่ วิธีการหาอักขระย่อยเหมือนยาวที่สุด (Longest Common Substring) และวิธีการหาลำดับเหมือนยาวที่สุด (Longest Common Subsequence) ผลลัพธ์ที่ได้จากการวิเคราะห์ของทั้ง 2 วิธี สามารถนำไปใช้ประโยชน์ในการเปรียบเทียบอักขระเพื่อจัดกลุ่มอักขระ หรือหาอักขระที่มีความคล้ายคลึงกันมากที่สุด (Similarity) ผลลัพธ์ที่ได้เป็นค่าความแตกต่างระหว่างแต่ละชุดอักขระ (Distance) และค่าความแตกต่างนี้สามารถนำไปประยุกต์ใช้งานได้หลากหลายประเภท เช่น การจัดกลุ่มข้อมูล (Data Clustering) ซึ่งเป็นวิธีการหนึ่งในการวิเคราะห์ข้อมูลที่มีขนาดมหาศาล (Data Mining) เป็นต้น สำหรับวิทยานิพนธ์ฉบับนี้มุ่งเน้นการนำขั้นตอนวิธีใหม่เข้ามาใช้แก้ปัญหาการหาอักขระย่อยเหมือนยาวที่สุด และวิธีการหาลำดับเหมือนยาวที่สุด ซึ่งแต่เดิมขั้นตอนวิธีที่ใช้ในการวิเคราะห์นั้นมีด้วยกัน 2 วิธีคือ การโปรแกรมแบบพลวัต (Dynamic Programming) และการใช้โครงสร้างต้นไม้ (Suffix Array)

วิทยานิพนธ์นี้นำวิธีการที่สามารถหาได้ทั้งวิธีการหาอักขระย่อยเหมือนยาวที่สุด และวิธีการหาลำดับเหมือนยาวที่สุดมาใช้ กล่าวคือ ได้ทำการวิเคราะห์เพื่อหาข้อจำกัดของวิธีการโปรแกรมแบบพลวัต ซึ่งข้อจำกัดที่พบคือ 1) เนื่องจากการเปรียบเทียบชุดอักขระจะอยู่ในรูปแบบของตาราง 2 มิติ ทำให้กลุ่มของชุดอักขระที่ต้องการเปรียบเทียบสามารถทำได้ครั้งละ 2 ชุดเท่านั้น ซึ่งเป็นข้อจำกัดหลักของการใช้งานตาราง 2 มิติ และ 2) ข้อจำกัดทางด้านความเร็วซึ่งพิจารณาจากจำนวนครั้งในการนำอักขระไปเปรียบเทียบกันในแต่ละครั้งที่มีการทำงาน พบว่ามีจำนวนครั้งในการทำงานสูงมาก วิทยานิพนธ์ฉบับนี้จึงเสนอแนวคิดการนำแบบจำลองภาษา (Language Model) มาแก้ปัญหาการวิเคราะห์หาอักขระย่อยเหมือนยาวที่สุด และหาลำดับเหมือนยาวที่สุดในชุดอักขระโดยใช้แบบจำลองภาษาแบบจำลองมาร์คอฟ หรือ แบบจำลองเอ็นแกรม ซึ่งเป็นแบบจำลองที่ใช้ในการคำนวณค่าความน่าจะเป็นของสายอักขระ

วิทยานิพนธ์ฉบับนี้เลือกการแบ่งหน่วย ซึ่งเรียกว่า แกรม (Gram) เป็นหน่วยที่ใช้ในการสร้างแบบจำลองและเลือกใช้หน่วยแบบจำลองที่เรียกว่า ไบแกรม (Bi-gram) มาทำการเปรียบเทียบและวิเคราะห์ การนำไบแกรมมาช่วยในการวิเคราะห์ทำให้การเปรียบเทียบชุดอักขระมีความตรงไปตรงมา และมีวิธีการที่ไม่ซับซ้อน ซึ่งผลที่ได้จากการหาอักขระย่อยเหมือนยาวที่สุดโดยใช้ไบแกรมมาช่วยในการวิเคราะห์นั้น สามารถลดข้อจำกัดของจำนวนชุดอักขระที่ต้องการเปรียบเทียบ และจำนวนครั้งในการเปรียบเทียบ สำหรับการหาลำดับเหมือนที่ยาวที่สุดนั้น สามารถลดข้อจำกัดในส่วนของชุดอักขระที่เปรียบเทียบเท่านั้นส่วนข้อจำกัดเรื่องจำนวนครั้งในการเปรียบเทียบจำเป็นต้องนำโครงสร้างซัฟฟิกซ์อะเรย์ ที่มีการพัฒนามาจากโครงสร้างต้นไม้ที่ได้กล่าวถึงข้างต้นมาช่วยจึงจะสามารถลดจำนวนครั้งในการเปรียบเทียบอักขระได้ รวมทั้งสามารถลดจำนวนครั้งในการเปรียบเทียบได้อย่างมีประสิทธิภาพในชุดอักขระที่มีความแตกต่างกันมาก

1.2 ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้จัดทำขึ้นโดยมีวัตถุประสงค์ เพื่อพัฒนาขั้นตอนวิธีใหม่ ในการวิเคราะห์หาอักขระย่อยเหมือนยาวที่สุด โดยขั้นตอนวิธีใหม่นี้สามารถลดข้อจำกัดของวิธีการทำงานแบบเดิม กล่าวคือ สามารถเปรียบเทียบชุดอักขระได้ครั้งละหลายชุด และลดจำนวนครั้งในการเปรียบเทียบอักขระลง ซึ่งจะทำให้การเปรียบเทียบมีประสิทธิภาพ และทำงานได้รวดเร็วขึ้น

1.3 สมมุติฐานของการศึกษา

การพัฒนาขั้นตอนวิธีใหม่ โดยใช้ไบแกรมเข้ามาช่วยในการแบ่งหน่วยอักขระเพื่อนำไปวิเคราะห์ สามารถทำการวิเคราะห์หาอักขระย่อยเหมือนยาวที่สุด และการหาลำดับเหมือนยาวที่สุดโดยลดข้อจำกัดของขั้นตอนวิธีเดิม ในด้านการวิเคราะห์ข้อมูลครั้งละหลายชุด และลดจำนวนการเปรียบเทียบอักขระ ทำให้ระบบมีประสิทธิภาพมากขึ้น และทำงานได้อย่างรวดเร็ว

1.4 ขอบเขตการศึกษา

วิทยานิพนธ์ฉบับนี้เป็นการพัฒนาขั้นตอนวิธีของการวิเคราะห์หาอักขระย่อยเหมือนยาวที่สุด และหาลำดับเหมือนยาวที่สุดในชุดอักขระโดยนำไบแกรมมาใช้เพื่อลดข้อจำกัดของวิธีการทำงานโดยการโปรแกรมแบบพลวัต และวัดประสิทธิภาพของขั้นตอนวิธีใหม่โดย

1.4.1 พิจารณาจากความสามารถในการรองรับจำนวนชุดอักขระที่ใช้ในการวิเคราะห์ข้อมูลแต่ละครั้ง

1.4.2 นำจำนวนครั้งในการเปรียบเทียบชุดอักขระโดยการโปรแกรมแบบพลวัต มาเปรียบเทียบกับจำนวนครั้งในการเปรียบเทียบของวิธีการที่วิเคราะห์ไบแกรมบนข้อมูลชุดมาตรฐานต่างๆ

1.5 ส่วนประกอบของวิทยานิพนธ์

ส่วนที่เหลือของวิทยานิพนธ์นี้ ประกอบด้วยส่วนต่าง ๆ ดังนี้

บทที่ 2 กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง โดยแบ่งเนื้อหาออกเป็น 3 ส่วน ดังนี้ ส่วนแรกกล่าวถึงการหาอักขระย่อยและลำดับเหมือนยาวที่สุดในชุดอักขระ ส่วนที่สองกล่าวถึงวิธีการหาอักขระย่อยและลำดับเหมือนยาวที่สุดในชุดอักขระ และส่วนสุดท้ายกล่าวถึงการใช้งานแบบจำลองภาษาไบแกรม

บทที่ 3 กล่าวถึงการหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม โดยแบ่งเนื้อหาออกเป็น 3 ส่วน ดังนี้ ส่วนแรกกล่าวถึงวิธีการใช้ไบแกรมในการแบ่งหน่วยของชุดอักขระ ส่วนที่สองกล่าวถึงขั้นตอนการเปรียบเทียบโดยใช้การพิจารณาจากลำดับ และส่วนสุดท้ายกล่าวถึงการเปรียบเทียบโดยใช้โครงสร้างของซัฟฟิกซ์อะเรย์

บทที่ 4 กล่าวถึง การประเมินผล โดยแบ่งเนื้อหาออกเป็น 4 ส่วน ดังนี้ ส่วนแรกกล่าวถึงข้อมูลที่ใช้ในการทดลอง ส่วนที่สองกล่าวถึงเครื่องมือที่ใช้ในการทดลอง ส่วนที่สามกล่าวถึงการทดลอง และส่วนสุดท้ายกล่าวถึงผลการทดลอง

บทที่ 5 กล่าวถึงการสรุปและข้อเสนอแนะเกี่ยวกับการหาอักขระย่อยเหมือน และการหาลำดับเหมือนยาวที่สุด

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

บทนี้จะกล่าวถึงงานวิจัยที่เกี่ยวข้อง กับนิยาม ทฤษฎีพื้นฐาน รวมทั้งขั้นตอนวิธีที่นิยมใช้ในการวิเคราะห์ในปัจจุบัน โดยเนื้อหาแบ่งออกเป็น 3 ส่วน ดังนี้

ส่วนที่ 1กล่าวถึงนิยาม ขั้นตอนการทำงาน และรูปแบบการวิเคราะห์เพื่อหาอักขระย่อยเหมือนยาวที่สุด และลำดับเหมือนยาวที่สุด

ส่วนที่ 2 กล่าวถึงวิธีการวิเคราะห์ หา อักขระย่อยเหมือนยาวที่สุด และลำดับเหมือนยาวที่สุด ซึ่งประกอบด้วย 3 วิธี คือ

- การใช้การโปรแกรมแบบพลวัต (Dynamic Programming)
- การหาแบบโดยใช้โครงสร้างซัฟฟิกซ์ทรี (Suffix Tree)
- การหาแบบโดยใช้โครงสร้างซัฟฟิกซ์อะเรย์ (Suffix Array)

ส่วนที่ 3 กล่าวถึงไบแกรม และการประยุกต์ของไบแกรม

2.1 การหาอักขระย่อยและลำดับเหมือนยาวที่สุดในชุดอักขระ

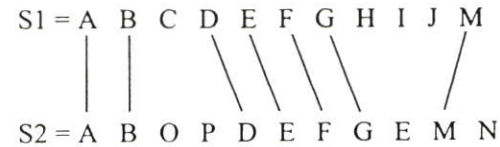
อักขระย่อยเหมือนยาวที่สุด คือ อักขระย่อยที่เหมือน และมีลำดับติดกันที่เกิดขึ้นในบรรดาชุดอักขระที่ต้องการเปรียบเทียบ

ตัวอย่างที่ 2.1 กำหนดให้ชุดอักขระ 2 ชุด ดังนี้

S1 = A B C D E F G H I J M

S2 = A B O P D E F G E M N

อักขระย่อยเหมือนยาวที่สุดเป็นอักขระย่อยที่เหมือนกัน และมีลำดับที่ติดกัน เมื่อพิจารณา S1 และ S2 จะเห็นว่า



นำ S1 และ S2 มาทำการวิเคราะห์หาอักขระย่อยเหมือนผลที่ได้ คือ

CS1 = AB

CS2 = DEFG

CS3 = M

จากอักขระย่อยเหมือนจะสนใจความยาวของผลลัพธ์ที่ได้ ซึ่งเห็นได้ว่า CS2 มีความยาวของตัวอักษรทั้งหมด 4 ตัวอักษร ในขณะที่ CS1 และ CS3 มีความยาวของตัวอักษรเพียง 2 ตัวอักษร และ 1 ตัวอักษร ตามลำดับ ดังนั้นเมื่อต้องการหาอักขระย่อยเหมือนยาวที่สุดผลลัพธ์ที่ได้จึงเป็น CS2

ลำดับเหมือนยาวที่สุด คือ ชุดอักขระย่อยที่มีลำดับที่เหมือนกัน และมีความยาวมากที่สุด

ตัวอย่างที่ 2.2 พิจารณา S1 และ S2 จากตัวอย่างที่ 2.1
จะได้ว่าลำดับย่อยที่ยาวที่สุด คือ ABDEFGM ซึ่งเป็นลำดับที่มีความยาว 7 จะเห็นว่าลำดับเหมือนยาวที่สุดนี้ไม่จำเป็นต้องเป็นลำดับย่อยที่ติดกัน

2.2 วิธีการหาอักขระย่อยและลำดับเหมือนยาวที่สุดในชุดอักขระ

2.2.1 การใช้การโปรแกรมแบบพลวัต

การโปรแกรมแบบพลวัต (Dynamic Programming) คือ การเปรียบเทียบโดยนำอักขระ 2 ชุดที่ต้องการเปรียบเทียบมาแบ่งเป็นอักขระทีละตัว และนำอักขระแต่ละตัวมาเปรียบเทียบผ่านสมการทางคณิตศาสตร์ และทำการจัดเก็บลงตาราง 2 มิติ โดยใช้ประสิทธิภาพเชิงเวลา คือ $O(n^2)$ โดยที่ n คือความยาวของสตริง

2.2.1.1 การหาอักขระย่อยเหมือนยาวที่สุด

$$LCS[I_{1....n}, J_{1....m}] = \begin{cases} LCS[I_{n-1}, J_{m-1}] + 1 & , \quad i[n] = j[m] \\ 0 & , \quad i[n] \neq j[m] \end{cases} \tag{2.1}$$

การโปรแกรมแบบพลวัตในการแก้ปัญหาจากสมการที่ 2.1 เป็นขั้นตอนการเปรียบเทียบเพื่อทำการวิเคราะห์ดูความเหมือน โดยมีขั้นตอนดังนี้

1. นำชุดอักขระที่ต้องการเปรียบเทียบทั้งหมด 2 ชุด โดยกำหนดให้ชุดแรก มีค่า I และชุดที่สองมีค่า J

2.ให้ความยาวของชุดอักขระชุดแรกแทนค่าด้วย x และชุดความยาวของชุดอักขระชุดที่สองแทนค่าด้วย y

3. กำหนดค่า n ให้เป็น 0 ทำการวนลูปชุดอักขระที่ I จนกว่า n จะเท่ากับ x

4. ทำการเปรียบเทียบชุดอักขระ I ตัวที่ n กับชุดอักขระ J ตัวที่ m ว่ามีค่าเหมือนกันหรือไม่ ถ้าเหมือนกันให้นำค่าของชุดอักขระ I ตัวที่ n-1 กับชุดอักขระ J ตัวที่ m-1 มาทำการบวก 1

5. หลังจากหมดชุดอักขระ I แล้วให้นำตัว m มาบวก 1 แล้วกำหนดให้เท่ากับ 0 ใหม่ จากนั้นกลับไปทำในขั้นตอนที่ 3

6. ทำจนกว่า m เท่ากับ y ในชุดอักขระ J

7. นำชุดลำดับที่ได้ทั้งหมดมาเปรียบเทียบว่าชุดลำดับใดมีขนาดยาวที่สุด

ตัวอย่างที่ 2.3 ชุดอักขระที่ต้องการเปรียบเทียบมี 2 ชุดคือ database และ dataset โดยเริ่มจากการกำหนดค่าให้กับชุดอักขระทั้งสองชุดที่ต้องการเปรียบเทียบ

I = database และ J = dataset

หลังจากกำหนดค่าและเปรียบเทียบชุดอักขระตามสมการที่ 2.1 แล้วจะได้ผลลัพธ์ตามตารางที่ 2.1

ตารางที่ 2.1 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.3

	d	a	t	a	b	a	s	e
d	1	0	0	0	0	0	0	0
a	0	2	0	1	0	1	0	0
t	0	0	3	0	0	0	0	0
a	0	1	0	4	0	1	0	0
s	0	0	0	0	0	0	2	0
e	0	0	0	0	0	0	0	3
t	0	0	1	0	0	0	0	0

จากตารางจะสรุปได้ว่ามีลำดับย่อยทั้งหมด 2 ชุดคือ data , ase ซึ่งอักขระย่อยเหมือนยาวที่สุดคืออักขระลำดับที่ 1,2,3 และ 4 นั่นคือ data

ตัวอย่างที่ 2.4 ชุดอักขระที่ต้องการเปรียบเทียบมี 3 ชุดคือ match, mismatch และ matching เริ่มโดยการเปรียบเทียบชุดอักขระ 2 ชุดคือ match และ mismatch หลังจากกำหนดค่าและเปรียบเทียบชุดอักขระด้วยสมการที่ 2.1 แล้วจะได้ผลลัพธ์ตามตารางที่ 2.2

ตารางที่ 2.2 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.4

	m	i	s	m	a	t	c	h
m	1	0	0	1	0	0	0	0
a	0	0	0	0	2	0	0	0
t	0	0	0	0	0	3	0	0
c	0	0	0	0	0	0	4	0
h	0	0	0	0	0	0	0	5

จากตารางสามารถสรุปได้ว่าชุดอักขระ match ซึ่งมีความยาวด้วยกันทั้งหมด 5 อักขระ หลังจากนั้นนำผลลัพธ์ที่ได้ไปเปรียบเทียบกับชุดอักขระชุดที่ 3 คือ matching หลังจากกำหนดค่าและเปรียบเทียบชุดอักขระด้วยสมการที่ 2.1 แล้วจะได้ผลลัพธ์ตามตารางที่ 2.3

ตารางที่ 2.3 ผลลัพธ์ชุดที่สองในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.4

	m	a	t	c	h	i	n	g
m	1	0	0	0	0	0	0	0
a	0	2	0	0	0	0	0	0
t	0	0	3	0	0	0	0	0
c	0	0	0	4	0	0	0	0
h	0	0	0	0	5	0	0	0

จากตารางสามารถสรุปได้ว่าชุดอักขระทั้ง 3 ชุดคือ match, mismatch และ matching มี match เป็นอักขระย่อยเหมือนยาวที่สุด โดยมีความยาว 5 อักขระ

2.2.1.2 การหาลำดับเหมือนยาวที่สุด

$$LCS[I_{1\ldots n}, J_{1\ldots m}] = \begin{cases} 0 & , i = 0 \, or \, j = 0 \\ LCS[I_{n-1}, J_{m-1}] + 1 & , i[n] = j[m] \\ MAX(LCS[I_{n-1}, J_{m-1}], LCS[I_{n-1}, J_{m-1}]) & , i[n] \neq j[m] \end{cases} \quad (2.2)$$

การโปรแกรมแบบพลวัตในการแก้ปัญหาจากสมการที่ 2.2 เป็นขั้นตอนการเปรียบเทียบเพื่อทำการวิเคราะห์หาความเหมือน มีขั้นตอนดังนี้

1. นำชุดอักขระที่ต้องการเปรียบเทียบทั้งหมด 2 ชุด โดยกำหนดให้ชุดแรก มีค่า I และชุดที่สองมีค่า J

2.ให้ความยาวของชุดอักขระชุดแรกแทนค่าด้วย N และชุดความยาวของชุดอักขระชุดที่สองแทนค่าด้วย M

3. กำหนดค่า n ให้เป็น 0 ทำการวนลูปชุดอักขระที่ I จนกว่า n จะเท่ากับ N

4. ทำการเปรียบเทียบชุดอักขระ I ตัวที่ n กับชุดอักขระ J ตัวที่ m ว่ามีค่าเหมือนกันหรือไม่ ถ้าเหมือนกันให้นำค่าของชุดอักขระ I ตัวที่ n-1 กับชุดอักขระ J ตัวที่ m-1 มาทำการบวก 1

5. หลังจากหมดชุดอักขระ I แล้วให้นำตัว m มาบวก 1 แล้วกำหนดให้ เท่ากับ 0 ใหม่ จากนั้นกลับไปทำในขั้นตอนที่ 3

6. ทำจนกว่า m เท่ากับ M ในชุดอักขระ J

7. นำชุดลำดับที่ได้ทั้งหมดมาเปรียบเทียบว่าชุดลำดับใดมีขนาดยาวที่สุด

ตัวอย่างที่ 2.5 ชุดอักขระที่ต้องการเปรียบเทียบมี 2 ชุดคือ database และ dataset โดยเริ่มจากการกำหนดค่าให้กับชุดอักขระทั้งสองชุดที่ต้องการเปรียบเทียบ

I = database และ J = dataset

หลังจากกำหนดค่าและเปรียบเทียบชุดอักขระด้วยสมการที่ 2.2 แล้วจะได้ผลลัพธ์ตามตารางที่ 2.4

ตารางที่ 2.4 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.5

	d	a	t	a	b	a	s	e
d	1	1	1	1	1	1	1	1
a	1	2	2	2	2	2	2	2
t	1	2	3	3	3	3	3	3
a	1	2	3	4	4	4	4	4
s	1	2	3	4	4	4	5	5
e	1	2	3	4	4	4	5	6
t	1	2	3	4	4	4	5	6

จากตารางที่ 2.4 สรุปได้ว่า ลำดับเหมือนยาวที่สุด คือ datase ซึ่งมีความยาวทั้งหมด 6 ตัวอักษร

ตัวอย่างที่ 2.6 ชุดอักขระที่ต้องการเปรียบเทียบมี 3 ชุดคือ match, mismatch และ matching เริ่มโดยการเปรียบเทียบ 2 ชุดคือ match และ mismatch หลังจากกำหนดค่าและเปรียบเทียบชุดอักขระด้วยสมการที่ 2.2 แล้วจะได้ผลลัพธ์ตามตารางที่ 2.5

ตารางที่ 2.5 ผลลัพธ์ในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.6

	m	i	s	m	a	t	c	h
m	1	1	1	1	1	1	1	1
a	1	1	1	1	2	2	2	2
t	1	1	1	1	2	3	3	3
c	1	1	1	1	2	3	4	4
h	1	1	1	1	2	3	4	5

จากตารางสรุปได้ว่าชุดอักขระ match ซึ่งมีความยาว 5 อักขระ เป็นอักขระที่มีลำดับเหมือนยาวที่สุด หลังจากนั้นนำผลลัพธ์ที่ได้ไปเปรียบเทียบกับชุดอักขระชุดที่ 3 คือ matching ซึ่งหลังจากกำหนดค่าและเปรียบเทียบชุดอักขระด้วยสมการที่ 2.2 แล้วจะได้ผลลัพธ์ตามตารางที่ 2.6

ตารางที่ 2.6 ผลลัพธ์ชุดที่สองในการเปรียบเทียบชุดอักขระในตัวอย่างที่ 2.6

	m	a	t	c	h	i	n	g
m	1	1	1	1	1	1	1	1
a	1	2	2	2	2	2	2	2
t	1	2	3	3	3	3	3	3
c	1	2	3	4	4	4	4	4
h	1	2	3	4	5	5	5	5

จากตารางสรุปได้ว่าชุดอักขระ match เป็นอักขระที่มีลำดับเหมือนยาวที่สุด ดังนั้นจึงสรุปได้ว่าชุดอักขระทั้ง 3 ชุดคือ match, mismatch และ matching นั้นมี match เป็นลำดับเหมือนยาวที่สุด และมีความยาวเท่ากับ 5 อักขระ

2.2.2 การใช้ซัฟฟิกซ์ทรี

2.2.2.1 ซัฟฟิกซ์ทรี

ซัฟฟิกส์ทรี คือ โครงสร้างของการจัดการข้อมูลลักษณะหนึ่ง ซึ่งมีการทำงานในแบบของต้นไม้ กล่าวคือจะมีจุดเริ่มต้นที่ โนดราก ไปถึง โนดใบ โดยการทำงานจะนำข้อมูลของซัฟฟิกซ์สตริงที่เชื่อมโยงกัน หรือเหมือนกัน มาต่อเรียงต่อกัน และในทุก ๆ กิ่งก้านของต้นไม้ จะเป็นตัวแสดงของซัฟฟิกซ์สตริง อีกทั้งยังสามารถช่วยให้การสืบค้นทำงานได้อย่างรวดเร็ว

ซัฟฟิกซ์ของสตริง คือ การแบ่งสตริงให้อยู่ในรูปแบบอักขระย่อย จากด้านซ้ายไปด้านขวา และจำได้จำนวนทั้งหมดเท่ากัน n-1 โดยที่ n คือความยาวของสตริงนั้น

ซัฟฟิกซ์ทรีสำหรับสตริง S คือ โครงสร้างต้นไม้ที่ใช้จัดเก็บอักขระย่อยที่เป็นไปได้ทั้งหมดของ S ในรูปซัฟฟิกซ์ โดยการเดินทางของซัฟฟิกซ์สตริ งนั้นจะต้องเริ่มต้นจาก โนดราก ไปถึง โนดใบของโครงสร้างต้นไม้ โดยใช้ประสิทธิภาพเชิงเวลา คือ O(n) โดยที่ n คือความยาวของสตริง S

ในการสร้างซัฟฟิกซ์ทรี สำหรับสตริงใดๆ ทำได้ดังนี้

1. เพิ่ม $ เข้าไปต่อท้ายในชุดอักขระ
2. ทำการแจงซัฟฟิกซ์ของชุดอักขระที่ต้องการ
3. ทำการสร้างซัฟฟิกซ์ทรี

ตัวอย่างที่ 2.7 จากตัวอย่างที่ 2.3

1. ทำการเพิ่ม $ ในชุดอักขระทั้งสองชุดคือ

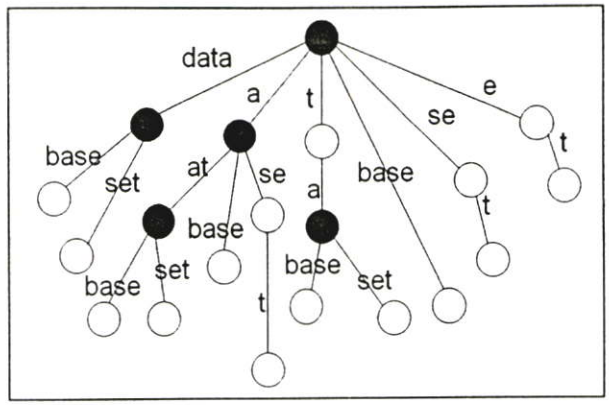
   I = database$ ซึ่งมีอักขระทั้งหมด 6 ตัว ดังนี้ d, a, t, b, s, e

   J = dataset$ ซึ่งมีอักขระทั้งหมด 5 ตัว ดังนี้ d, a, t, s, e
2. ทำการแจงซัฟฟิกซ์ทั้งสองชุดอักขระได้ ดังตารางที่ 2.7

ตารางที่ 2.7 ผลลัพธ์การแจงซัฟฟิกซ์จากตัวอย่างที่ 2.7

ชุดอักขระที่ I = database$	ชุดอักขระที่ J = dataset$
d a t a b a s e $	d a t a s e t $
a t a b a s e $	a t a s e t $
t a b a s e $	t a s e t $
a b a s e $	a s e t $
b a s e $	s e t $
a s e $	e t $
s e $	t $
e $	$
$	

3. หลังจากทำการแจงซัฟฟิกซ์ทั้งสองเป็นที่เรียบร้อยแล้วก็นำค่าที่แจกแจงแล้วมาจัดลงในโครงสร้างต้นไม้



รูปภาพที่ 2.1 ซัฟฟิกซ์ทรี ของชุดอักขระทั้งสองชุด

ผลลัพธ์ที่ได้จากการนำชุดอักขระมาทำการแจงซัฟฟิกซ์และสร้างเป็นทรี สามารถบอกได้ว่า ลำดับที่ใช้ร่วมกันของทั้งสองชุดอักขระ คือ data, tat, ta และ se ซึ่งถ้าเลือกอักขระย่อยเหมือนยาวที่สุด คือ data

ตัวอย่างที่ 2.8 จากตัวอย่างที่ 2.4

1. ทำการเพิ่ม $ ในชุดอักขระทั้งสามชุดคือ

   I = match$ ซึ่งมีอักขระทั้งหมด 5 ตัว ดังนี้ m, a, t, c และ h

   J = mismatch$ ซึ่งมีอักขระทั้งหมด 7 ตัว ดังนี้ m, i, s, a, t, c และ h

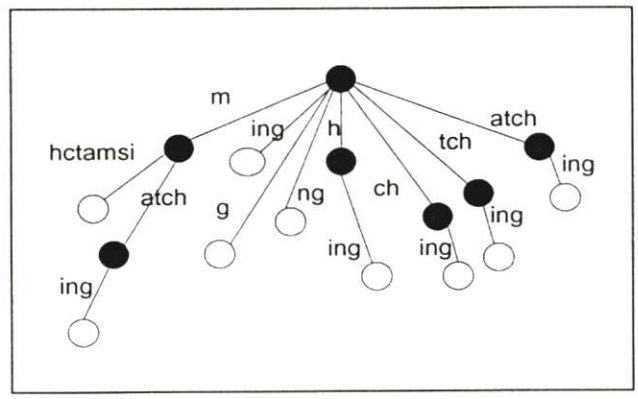
   K= matching$ ซึ่งมีอักขระทั้งหมด 8 ตัว ดังนี้ m, a, t, c, h, i, n และ g

2. ทำการแจงซัฟฟิกซ์ทั้งสามชุดอักขระได้ดังตารางที่ 2.9

ตารางที่ 2.8 ผลลัพธ์การแจงซัฟฟิกซ์จากตัวอย่างที่ 2.8

ชุดอักขระที่ I = match$	ชุดอักขระที่ J = mismatch$	ชุดอักขระที่ K= matching$
match$	mismatch$	matching$
atch$	ismatch$	atching$
tch$	smatch$	tching$
ch$	match$	ching$
h$	atch$	hing$
$	tch$	ing$
	ch$	ng$
	h$	g$
	$	$

3. หลังจากทำการแจงซัฟฟิกซ์ทั้งสองเป็นที่เรียบร้อยแล้วก็นำค่าที่แจกแจงแล้วมาจัดลงในโครงสร้างต้นไม้



รูปภาพที่ 2.2 ซัฟฟิกซ์ทรี ของชุดอักขระทั้งสามชุด

จากโครงสร้างสามารถหาลำดับที่ใช้ร่วมกันได้คือ match ซึ่งเป็นลำดับที่ยาวที่สุดโดยมีความยาว 5 อักขระ

2.2.2.2 การพัฒนาขั้นตอนการสืบค้นอักขระย่อยบนซัฟฟิกซ์ทรี

Wiener [9][10] ได้นำเสนอวิธีการในการแยกแยะหาซัฟฟิกซ์ทรี โดยใช้ First liner-time algorithm ในปี ค.ศ. 1973 เป็นการเริ่มต้นการพัฒนาซัฟฟิกซ์ทรีให้มีโครงสร้างเป็นลักษณะต้นไม้ที่มีการเข้าถึงได้ง่าย รวดเร็ว และไม่ซับซ้อน โดยการลดจำนวนการแจกแจงซัฟฟิกซ์อะเรย์ที่ซ้ำกันให้สามารถอยู่ด้วยกัน และใช้เนื้อที่หน่วยความจำที่ขึ้นอยู่กับความยาวของสตริง โดยใช้ประสิทธิภาพเชิงเวลา คือ O(n) โดยที่ n คือความยาวของสตริง

Edward McCreight [3][10] ได้เสนอ McCreight's suffix tree algorithm โดยทำการพัฒนาต่อยอดมาจากวิธี First liner-time algorithm ซึ่งขั้นตอนวิธีใหม่นี้จะสามารถลด หน่วยความจำลงได้ โดยใช้เทคนิคของพรีฟิกซ์มาช่วยแก้ปัญหาเพื่อทำการแตกกิ่ง โดยทุกจุดที่จะมีการแตกกิ่งนั้นจะทำการแสดงถึงเส้นทางของสัญลักษณ์ที่มีความแตกต่าง โดยวิธีการนี้จะทำให้ลดเนื้อที่ในการทำงานน้อยลงกว่าขั้นตอนวิธีเดิม

Esko Ukkonen [8][10] ได้เสนอแนวความคิดที่มีชื่อว่า ออนไลน์ (on-line) ซึ่งมีความแตกต่างจากวิธีการเดิมไม่มากนัก โดยการทำงานจะเริ่มจากการสร้างต้นไม้ว่าง หลังจากนั้นนำเอาพรีฟิกซ์ที่ได้ทำการแจกแจงมาแล้วลงสู่ซัฟฟิกซ์ทรี โดยจะเริ่มทำการด้านซ้ายไปด้านขวา และเมื่อนำพรีฟิกซ์ลงสู่ต้นไม้หมดแล้วก็จะได้ ซัฟฟิกซ์ทรีที่มีโครงสร้างเหมือนวิธีการเดิม แต่วิธีการนี้ถูกพัฒนาขึ้นมาเพื่อช่วยให้ง่ายต่อการอธิบายและงานต่อการทำงาน

2.2.3 การใช้ซัฟฟิกซ์อะเรย์

2.2.3.1 ซัฟฟิกซ์อะเรย์

ซัฟฟิกซ์อะเรย์ คือ โครงสร้างข้อมูลที่ใช้จัดเก็บข้อมูลสตริงเพื่อนำไปใช้งานในการพัฒนาโปรแกรม

ซัฟฟิกซ์อะเรย์ของสตริง S คือ โครงสร้างข้อมูลที่ใช้จัดเก็บอักขระย่อยที่เป็นไปได้ทั้งหมดของ S ในรูปซัฟฟิกซ์ ซึ่งจัดเก็บอยู่ในอะเรย์ตั้งแต่ตำแหน่งที่ 0 ถึง n-1 โดยใช้ประสิทธิภาพเชิงเวลาคือ O(n log n) โดยที่ n คือความยาวของสตริง S

ในการสร้างซัฟฟิกซ์อะเรย์ สำหรับสตริงใดๆ ทำได้ดังนี้

1. ทำการแจงซัฟฟิกซ์ของชุดอักขระที่ต้องการ
2. ทำการเก็บค่าซัฟฟิกซ์ใส่อะเรย์ตั้งแต่ตำแหน่งที่ 0 ถึงตำแหน่งที่ n-1
3. เรียงลำดับตัวอักษรของค่าซัฟฟิกซ์ในอะเรย์ใหม่ จากอักษรตัวแรกของซัฟฟิกซ์อะเรย์
4. นำซัฟฟิกซ์อะเรย์ที่ได้เรียงลำดับแล้วมาทำการเปรียบเทียบกัน

ตัวอย่างที่ 2.9 จากตัวอย่างที่ 2.3

1. ทำการแจงซัฟฟิกซ์ของชุดอักขระที่ต้องการ
2. ทำการแจงซัฟฟิกซ์ทั้งสามชุดอักขระได้ดังนี้

ตารางที่ 2.9 ผลลัพธ์การแจงซัฟฟิกซ์จากตัวอย่างที่ 2.9

ชุดอักขระที่ I = database$	ชุดอักขระที่ J = dataset$
d a t a b a s e $	d a t a s e t $
a t a b a s e $	a t a s e t $
t a b a s e $	t a s e t $
a b a s e $	a s e t $
b a s e $	s e t $
a s e $	e t $
s e $	t $
e $	$
$	

3. ทำการเก็บค่าซัฟฟิกซ์ใส่อะเรย์ตั้งแต่ตำแหน่งที่ 0 ถึงตำแหน่งที่ n-1

ตารางที่ 2.10 การเก็บค่าชุดอักขระ I ลงซัฟฟิกซ์อะเรย์

ดัชนี	ซัฟฟิกซ์							
0	d	a	t	a	b	a	s	e
1	a	t	a	b	a	s	e	
2	t	a	b	a	s	e		
3	a	b	a	s	e			
4	b	a	s	e				
5	a	s	e					
6	s	e						
7	e							

ตารางที่ 2.11 การเก็บค่าชุดอักขระ J ลงซัฟฟิกซ์อะเรย์

ดัชนี	ซัฟฟิกซ์						
0	d	a	t	a	s	e	t
1	a	t	a	s	e	t	
2	t	a	s	e	t		
3	a	s	e	t			
4	s	e	t				
5	e	t					
6	t						

4. ทำการเรียงลำดับตัวอักษรของค่าซัฟฟิกซ์ในอะเรย์ใหม่ จากอักษรตัวแรกของซัฟฟิกซ์อะเรย์นั้น ๆ

ตารางที่ 2.12 การเรียงลำดับตัวอักขระของชุดอักขระ I

ดัชนี	ซัฟฟิกซ์							
3	a	b	a	s	e			
5	a	s	e					
1	a	t	a	b	a	s	e	
4	b	a	s	e				
0	d	a	t	a	b	a	s	e
7	e							
6	s	e						
2	t	a	b	a	s	e		

ตารางที่ 2.13 การเรียงลำดับตัวอักษระของชุดอักขระ J

ดัชนี	ซัฟฟิกซ์						
3	a	s	e	t			
1	a	t	a	s	e	t	
0	d	a	t	a	s	e	t
5	e	t					
4	s	e	t				
6	t						
2	t	a	s	e	t		

ตารางที่ 2.14 การเรียงลำดับตัวอักษรของชุดอักขระ I และ J รวมกัน

ดัชนี	ซัฟฟิกซ์							
3	a	b	a	s	e			
5	a	s	e					
3	a	s	e	t				
1	a	t	a	b	a	s	e	
1	a	t	a	s	e	t		
4	b	a	s	e				
0	d	a	t	a	b	a	s	e
0	d	a	t	a	s	e	t	
7	e							
5	e	t						
6	s	e						
4	s	e	t					
6	t							
2	t	a	b	a	s	e		
2	t	a	s	e	t			

จากตารางที่ 2.11 ได้ทำการเรียงลำดับอักษรของชุดอักขระทั้งสองชุดแล้ว จะพบว่าลักษณะตารางสีทึบจะเป็นชุดอักขระ J และลักษณะตารางสีขาวจะเป็นชุดอักขระ I และเมื่อทำการเรียงลำดับอักษรเรียบร้อยก็จะสามารถหากลุ่มของชุดอักขระที่ยาวที่สุดได้ โดยจะเกิดขึ้นในทั้ง 2 ชุดอักขระนั่นก็คือ d, a, t และ a ซึ่งมีความยาว 4 อักขระ

2.2.3.2 การพัฒนาขั้นตอนการสืบค้นอักขระย่อยบนซัฟฟิกซ์อะเรย์

Udi Manber และ Gene Myers [7][10] ได้นำซัฟฟิกซ์อะเรย์มาใช้งานกับซัฟฟิกซ์สตริง โดยลักษณะการทำงานจะเป็นการนำเอาสตริงไปทำการแจกแจงเป็นซัฟฟิกซ์แล้วนำมาเก็บในรูปของซัฟฟิกซ์อะเรย์ และทำการจัดเรียงลำดับซัฟฟิกซ์อะเรย์ใหม่ ตามลำดับตัวอักษรของซัฟฟิกซ์สตริง ซึ่งจะทำให้เกิดซัฟฟิกซ์อะเรย์ที่ได้จัดเรียงลำดับแล้ว และสามารถนำไปค้นหาสตริงย่อยได้ โดยใช้การค้นหาจากด้านบนลงสู่ด้านล่างของซัฟฟิกซ์อะเรย์

Kenneth Ward Church [4][10] ได้นำเอาซัฟฟิกซ์อะเรย์มาใช้ในงานการจัดกลุ่มเอกสาร ซึ่งมีลักษณะการทำงานคือการนำเอาสตริงหลายชุดมาทำการแจกแจงเป็นซัฟฟิกซ์สตริงและนำเอาซัฟฟิกซ์สตริงทั้งหมดลงสู่ซัฟฟิกซ์อะเรย์เพื่อที่จะหาสตริงย่อยที่เกิดขึ้นบนหลายสตริง ซึ่งจะเป็นการ

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

หาความถี่ที่เกิดบนหนึ่งเอกสาร และการหาสตริงย่อยที่เกิดขึ้นบนเอกสาร และนำมาจัดเก็บเป็นสถิติ โดยสามารถนำมาใช้กับงานทางด้านฐานข้อมูลต่อไปได้

2.3 แบบจำลองภาษา ไบแกรม

2.3.1 แบบจำลองไบแกรม

แบบจำลองเอ็นแกรม คือ แบบจำลองที่ใช้ในการคำนวณหาความน่าจะเป็นของชุดอักขระ และใช้หาค่าความน่าจะเป็นที่เกิดขึ้นร่วมกัน โดยมีแนวความคิดว่า การเกิดขึ้นของตัวอักษรตัวหนึ่งจะขึ้นกับตัวอักษรก่อนหน้าเพียง n-1 ตัว

แกรม คือ หน่วยที่ใช้ในการสร้างแบบจำลอง ซึ่งแบ่งได้ทั้ง เสียง คำ และอักขระ ซึ่งแกรมมีขนาดได้ตั้งแต่ 1 ถึง n เช่น

2-แกรม (bi-gram) จะทำการแบ่งชุดอักขระออกมาเป็นครั้งละ 2 ตัวเช่น computer จะแบ่งได้ดังนี้ co, om, mp, pu, ut, te, er

3-แกรม (tri-gram) จะทำการแบ่งชุดอักขระออกเป็นครั้งละ 3 ตัวเช่น computer จะแบ่งได้ดังนี้ com, omp, mpu, put, ute, ter

โดยงานวิจัยที่นำแบบจำลองเอ็นแกรมไปใช้งาน จะมุ่งเน้นไปทางการวิเคราะห์หาการระบุภาษา

2.3.2 การประยุกต์ของพัฒนาโดยใช้แบบจำลองไบแกรม

Cavnar และ Trenkle [1] ได้นำแบบจำลองเอ็นแกรมมาใช้ในปี ค.ศ. 1994 เพื่อการระบุหาประเภทของข้อความในแต่ละภาษา โดยเลือกใช้การทดสอบตั้งแต่ 1-5 แกรม โดยในแบบจำลองนี้ใช้การพิจาณาที่ละข้อความต่อเนื่องกัน ทั้งในการฝึกและในการทดสอบ ซึ่งหลังจากได้ทำการฝึกแล้วทำการนำข้อมูลทดสอบกับข้อมูลฝึกแต่ละประเภทมาทดสอบ และเปรียบเทียบความน่าจะเป็นว่าตรงกันมากน้อยเพียงใด โดยใช้การคำนวณหาค่าความห่าง (distance measure) เพื่อให้ได้มาซึ่งค่าความห่างที่น้อยที่สุด ซึ่งผลจากการทดลองจะได้ค่าความถูกต้องมากกว่า 90% ผลของการวิเคราะห์ขึ้นอยู่กับขนาดของข้อมูลที่นำเข้ามาทดสอบ และข้อมูลที่ส่งให้ระบบทำการเรียนรู้ ยิ่งมีข้อมูลที่ส่งให้ระบบเรียนรู้มากเท่าใดโอกาสที่ผลจะออกมาถูกต้องก็มากขึ้นตามไปด้วย

Dunning [2] ได้นำแบบจำลองเอ็นแกรมไปใช้ในปี ค.ศ. 1994 เพื่อระบุภาษาของข้อความที่เขียนด้วยภาษาโรมัน โดยใช้เหตุผลในการเลือกใช้แบบจำลองนี้ว่า แบบจำลองเอ็นแกรมเป็นแบบจำลองที่สามารถใช้ข้อมูลในการทดสอบจำนวนน้อยก็สามารถได้ผลลัพธ์ที่น่าพอใจ โดยลักษณะแบบจำลองของเข้า ใช้การพิจารณาดูความน่าจะเป็นของอักขระที่เกิดขึ้น ซึ่งทำการพิจารณาตลอดทั้งข้อความ จากนั้นนำมาเปรียบเทียบความถูกต้อง โดยจากทดลองพบว่ายิ่งทำการใช้ข้อมูลการฝึกและ ข้อมูลที่ใช้ทดสอบมากจะมีโอกาสที่ค่าความแม่นยำจะสูงถึง 99% แต่เมื่อนำไปทดสอบ

77960

กับภาษาอื่นแล้วจะพบข้อผิดพลาดมากเนื่องจากว่า แต่ละภาษามีลักษณะโครงสร้างคล้ายกัน ทำให้อาจจะระบุภาษาผิดได้

Sibun and Reynar [6] ได้นำแบบจำลองเอ็นแกรมไปใช้ในปี ค.ศ. 1996 เพื่อระบุภาษาที่เขียนด้วยอักษรโรมัน โดยในการทดลองใช้การแบ่งหน่วยจำลองเอ็นแกรมเป็น 1-3 แกรม เพื่อใช้ในการฝึก แบบจำลองนี้จะพิจาณาโดยใช้การดูทีละเอกสารข้อความที่ต่อเนื่องกัน ซึ่งจะพิจารณาเฉพาะตัวอักษรเท่านั้นจะตัดในส่วนของสัญลักษณ์ออก ทดสอบโดยทำการวัดความน่าจะเป็นระหว่างข้อมูลชุดที่ใช้ทดสอบกับชุดที่ใช้ฝึกว่าตรงกันในภาษาใด ซึ่งผลจากการทดสอบนั้นสามารถระบุความแม่นยำได้มากกว่า 90%

Peng et al [5] ได้นำแบบจำลองเอ็นแกรมไปใช้ในปี ค.ศ. 1996 เพื่อระบุประเภทของข้อความในรูปแบบของภาษาจีนและภาษาญี่ปุ่น โดยในแบบจำลองนี้จะทำการคาดเดาความน่าจะเป็น ว่าในแต่ละประเภทของข้อความจะมีการรวมกันของอักขระในลักษณะใด ซึ่งทำการดูทีละข้อความต่อเนื่องกัน ซึ่งในการทดลองได้ใช้แบบจำลองเอ็นแกรมเป็น 1-8 แกรม หลังจากได้ทำการสร้างแบบจำลองภาษาและ แบบจำลองที่ใช้ในการแบ่งประเภทแล้ว ทำการทดลองผลการเพื่อทำการระบุประเภทนั้นสามารถทำได้สูงกว่า 80% และการใช้แบบจำลองเอ็นแกรมโดยใช้ 3 แกรม จะได้ผลดีที่สุด

บทที่ 3
การหาลำดับเหมือนยาวที่สุดโดยใช้การวิเคราะห์ไบแกรม

วิทยานิพนธ์นี้นำเสนอการปรับปรุงวิธีการหาอักขระย่อยเหมือนยาวที่สุด และวิธีการหาลำดับเหมือนยาวที่สุดโดยในบทนี้เป็นการนำเสนอแนวคิดใหม่ ที่มีการนำไบแกรมเข้ามาช่วยในการแบ่งหน่วยของชุดอักขระที่ต้องการ และนำเสนอขั้นตอนวิธีใหม่ที่ใช้ในการเปรียบเทียบข้อมูลที่ได้ทำการแบ่งหน่วยไบแกรม โดยใช้วิธีคิดในส่วนของการเปรียบเทียบโดยตรง และวิธีการนำชุดอักขระมาจัดเรียงในแบบของซัฟฟิกซ์อะเรย์ เพื่อช่วยให้การทำงานสะดวกรวดเร็ว และมีประสิทธิภาพยิ่งขึ้น โดยเนื้อหาแบ่งออกเป็น 3 ส่วน ดังนี้

1. วิธีการใช้ไบแกรมในการแบ่งหน่วยของชุดอักขระ
2. การเปรียบเทียบโดยใช้การพิจารณาจากลำดับ
3. การเปรียบเทียบโดยใช้โครงสร้างของซัฟฟิกซ์อะเรย์

3.1 วิธีการใช้ไบแกรมในการแบ่งหน่วยของชุดอักขระ

การแบ่งชุดอักขระเพื่อหาอักขระย่อยเหมือนยาวที่สุด และหาลำดับเหมือนยาวที่สุด นั้น มีข้อแตกต่างอย่างเห็นได้ชัดอยู่ประการหนึ่ง กล่าวคือ การหาอักขระย่อยเหมือนยาวที่สุด จะเป็นการหาลำดับย่อยของชุดอักขระที่อยู่ติดกันโดยไม่สามารถข้ามลำดับได้ ส่วนการหาลำดับเหมือนยาวที่สุดจะเป็นการหาลำดับย่อยของชุดอักขระที่ไม่จำเป็นต้องอยู่ติดกันก็ได้ หรือสามารถข้ามลำดับได้นั่นเอง

ซึ่งความแตกต่างดังกล่าว ส่งผลให้วิธีในการแบ่งหน่วยของชุดอักขระมีความแตกต่างกันออกไป โดยในส่วนของการหาอักขระย่อยเหมือนยาวที่สุด นั้นสามารถทำการแบ่งชุดอักขระออกไปตามจำนวนที่ต้องการได้ เช่น ถ้าต้องการแบ่งหน่วยของชุดอักขระ 'ABCDE' ออกเป็นครั้งละ 2 หน่วย ก็สามารถแบ่งออกเป็น AB, BC, CD และ DE ซึ่งจำนวนของหน่วยย่อยที่ได้จะเท่ากับ n-1 โดยที่ n เป็นจำนวนความยาวของชุดอักขระ จะเห็นว่าผลลัพธ์ที่ได้จากการแบ่งครั้งละ 2 จำนวนนั้น เป็นการแบ่งแบบเรียงลำดับ แต่ในส่วนของการหาลำดับเหมือนยาวที่สุดไม่สามารถแบ่งในรูปแบบนี้ได้เพราะโอกาสที่ลำดับย่อยจะเกิดขึ้นได้ไม่จำเป็นต้องมีปัจจัยทางด้านการเรียงชิดติดกันเข้ามาเกี่ยวข้อง หมายความว่าโอกาสที่จะเกิดลำดับย่อยขึ้นในแบบข้ามลำดับนั้นเป็นไปได้สูง ทำให้การเรียงข้อมูลแบบแรกไม่สามารถที่จะเปรียบเทียบหาลำดับเหมือนได้ จากตัวอย่างเดิม คือชุดอักขระ 'ABCDE' การแบ่งหน่วยย่อยออกไปครั้งละ 2 หน่วย เมื่อนำมาเรียงลำดับเพื่อหาลำดับ

เหมือนยาวที่สุดจะได้ผลลัพธ์ดังนี้ AB, AC, AD, AE, BC, BD, BE, CD, CE, DE ซึ่งจำนวนของหน่วยย่อยที่ได้จะเท่ากับ $\sum N$

ตัวอย่างที่ 3.1 จากตัวอย่างที่ 2.3

สามารถนำชุดอักขระในโจทย์มาทำการแบ่งไบแกรมเพื่อใช้สำหรับเปรียบเทียบ ทำได้ดังนี้

ชุดอักขระที่ 1 database

ชุดอักขระที่ 2 dataset

การแบ่งหน่วยย่อยสำหรับใช้ในการเปรียบเทียบหาอักขระย่อยเหมือนยาวที่สุดได้ดังนี้

ตารางที่ 3.1 ตารางผลลัพธ์ของการแบ่งไบแกรมจากตัวอย่างที่ 3.1

database	dataset
da	da
at	at
ta	ta
ab	as
ba	se
as	et
se	

การแบ่งหน่วยย่อยสำหรับใช้การเปรียบเทียบหาลำดับเหมือนยาวที่สุดทำได้ดังนี้

ตารางที่ 3.2 ผลลัพธ์ของการแบ่งหน่วยของชุดอักขระที่ 1

da	dt	da	db	da	ds	de
at	aa	ab	aa	as	ae	
ta	tb	ta	ts	te		
ab	aa	as	ae			
ba	bs	be				
as	ae					
se						

ตารางที่ 3.3 ผลลัพธ์ของการแบ่งหน่วยของชุดอักขระที่ 2

da	dt	da	ds	de	dt
at	aa	as	ae	at	
ta	ts	te	tt		
as	ae	at			
se	st				
se					

ผลลัพธ์ที่ได้จากการแบ่งหน่วยย่อยของตารางที่ 3.1, 3.2 และ 3.3 นั้นเป็นการแบ่งหน่วยย่อยโดยใช้ไบแกรม ซึ่งเป็นการแบ่งตามลักษณะของงานที่ต้องการจะเปรียบเทียบ

3.2 การเปรียบเทียบโดยใช้การพิจาณาจากลำดับ

การเปรียบเทียบโดยการใช้การพิจารณาจากลำดับนั้น มีลักษณะการเปรียบเทียบแบบตรงไปตรงมา ทำให้สามารถเข้าใจได้ง่าย โดยรูปแบบที่ใช้ในการเปรียบเทียบ คือ การนำกลุ่มอักขระที่ได้ทำการแบ่งหน่วยแล้ว มาทำการเปรียบเทียบแบบเรียงลำดับ การเรียงลำดับนี้จะเกิดขึ้นจากลำดับการแบ่งหน่วย เช่น อักขระชุด 'ABCDE' ทำการแบ่งหน่วยในแบบการหาอักขระย่อยเหมือนยาวที่สุดจะได้ AB, BC, CD และ DE ซึ่งเมื่อทำการจัดลำดับให้จะแสดงได้ดังนี้

อักขระไบแกรม	AB	BC	CD	DE
ลำดับ	1	2	3	4

จากผลลัพธ์ที่ได้การนำอักขระชุดที่สองมาเปรียบเทียบจะต้องทำการใส่ลำดับและแบ่งหน่วยย่อย จากนั้นนำอักขระในชุดที่ 2 มาทำการเปรียบเทียบกัน ซึ่งขั้นตอนของการเปรียบเทียบนั้นจะต้องเก็บอักขระที่ได้จากการนำชุดอักขระที่ 1 มาแบ่ง โดยจัดเตรียมตามลำดับ หลังจากนั้นจะนำชุดอักขระที่ 2 ลำดับแรกมาเปรียบเทียบกันว่า ในชุดอักขระที่ 1 นั้นมีหน่วยอักขระใดที่ตรงกับชุดอักขระที่ 2 ลำดับแรก ถ้ามีหน่วยอักขระใดที่ตรงกันจะทำการใส่ลำดับชุดอักขระที่ 2 ลำดับแรกกับแถวที่ตรงกัน แต่ถ้าไม่มีหน่วยอักขระที่ตรงกันก็จะทำการเพิ่มชุดอักขระหลักต่อท้าย ทำเช่นนี้จนกว่าจะหมดชุดอักขระที่ 2 ซึ่งเมื่อหมดชุดอักขระแล้ว ก็จะทำการเปรียบเทียบหาชุดอักขระที่อยู่ใกล้กันว่าชุดอักขระใดมีลำดับยาวที่สุด โดยประสิทธิภาพเชิงเวลา คือ $O(n^k)$ โดยที่ n เป็นจำนวนชุด

ที่แบ่งโดยไบแกรมที่มีความยาวที่สุด และ k คือจำนวนชุดสตริงที่นำมาใช้ในการเปรียบเทียบ โดยลำดับขั้นตอนการทำงานเป็นดังนี้

1. ทำการแบ่งหน่วยโดยใช้ไบแกรม
2. ทำการใส่ลำดับชุดอักขระที่ได้ทำการแบ่งหน่วยแล้ว
3. ทำการเปรียบเทียบชุดอักขระไปที่ละลำดับ
4. นำส่วนที่มีชุดหน่วยอักขระที่เหมือนกันมาพิจารณาต่อ
5. ทำการพิจารณาว่า ชุดหน่วยอักขระใดมีความยาวมากที่สุด

ตัวอย่างที่ 3.2 จากตัวอย่างที่ 2.3 เป็นการหาอักขระย่อยเหมือนยาวที่สุด ทำได้ดังนี้

1. ทำการแบ่งอักขระซึ่งอ้างอิงการแบ่งจากตารางที่ 3.1
2. ทำการใส่ลำดับให้กับชุดอักขระที่แบ่งหน่วยแล้ว

อักขระชุดที่ 1 database

อักขระ	da	at	ta	ab	ba	as	se
ลำดับ	1	2	3	4	5	6	7

อักขระชุดที่ 2 dataset

อักขระ	da	at	ta	as	se	et
ลำดับ	1	2	3	4	5	6

3. ทำการนำลำดับชุดอักขระที่ 2 ลำดับแรกมาเปรียบเทียบจนครบทุกลำดับ

ตารางที่ 3.4 ตารางการเปรียบเทียบชุดอักขระลำดับที่ 1 และ 2

อักขระ	**ลำดับชุดที่ 1**	**ลำดับชุดที่ 2**
da	1	1
at	2	2
ta	3	3
ab	4	
ba	5	
as	6	4
se	7	5
et		6

4. เมื่อทำการเปรียบเทียบชุดอักขระย่อยจนหมดแล้ว จะทำการพิจารณาเฉพาะในส่วนที่มีชุดอักขระเหมือนกัน นั่นคือ ทั้งสองชุด

ตารางที่ 3.5 เลือกชุดอักขระย่อยที่เหมือนกัน

อักขระ	ลำดับชุดที่ 1	ลำดับชุดที่ 2
da	1	1
at	2	2
ta	3	3
ab	4	
ba	5	
as	6	4
se	7	5
et		6

5. เมื่อได้ชุดอักขระย่อยที่เหมือนกันทั้งหมดแล้วก็จะทำการพิจาณาว่าชุดใดเป็นชุดอักขระที่มีความยาวมากที่สุด

อักขระ	ลำดับชุดที่ 1	ลำดับชุดที่ 2
da	1	1
at	2	2
ta	3	3

อักขระ	ลำดับชุดที่ 1	ลำดับชุดที่ 2
as	6	4
se	7	5

ซึ่งจากผลลัพธ์ที่ได้จะเห็นว่ากลุ่มหน่วยอักขระแรกมีความยาวมากกว่า ดังนั้นหน่วยอักขระดังกล่าวจึงเป็นการหาอักขระย่อยเหมือนยาวที่สุดซึ่งประกอบด้วย da , at , ta และเมื่อทำการเปรียบกลับมาเป็นรูปประโยคธรรมดา จะได้คำว่า data

ตัวอย่างที่ 3.3 จากตัวอย่างที่ 2.3 เป็นการหาลำดับเหมือนยาวที่สุด ทำได้ดังนี้

1. ทำการแบ่งอักขระซึ่งอ้างอิงการแบ่งจากตารางที่ 3.2 และตารางที่ 3.3
2. ทำการใส่ลำดับให้กับชุดอักขระที่แบ่งหน่วยแล้ว

ตารางที่ 3.6 อักขระชุดที่ 1

อักขระ	da	dt	da	db	da	ds	de
ลำดับ	1	2	3	4	5	6	7
อักขระ	at	aa	ab	aa	as	ae	ta
ลำดับ	8	9	10	11	12	13	14
อักขระ	tb	ta	ts	te	ab	aa	as
ลำดับ	15	16	17	18	19	20	21
อักขระ	ae	ba	bs	be	as	ae	se
ลำดับ	22	23	24	25	26	27	28

ตารางที่ 3.7 อักขระชุดที่ 2

อักขระ	da	dt	da	ds	de	dt	at
ลำดับ	1	2	3	4	5	6	7
อักขระ	aa	as	ae	at	ta	ts	te
ลำดับ	8	9	10	11	12	13	14
อักขระ	tt	as	ae	at	se	st	se
ลำดับ	15	16	17	18	19	20	21

3. ทำการนำลำดับชุดอักขระที่ 2 ลำดับแรกมาเปรียบเทียบจนครบทุกลำดับ

ตารางที่ 3.8 ตารางการเปรียบเทียบชุดอักขระลำดับที่ 1 และ 2

อักขระ	ลำดับชุดที่ 1	ลำดับชุดที่ 2
da	1	1
dt	2	2
da	3	3
db	4	
da	5	
ds	6	4
de	7	5
at	8	7
aa	9	8
ab	10	
aa	11	
as	12	9
ae	13	10
ta	14	12
tb	15	
ta	16	
ts	17	13
te	18	14
ab	19	
aa	20	
as	21	16
ae	22	17
ba	23	
bs	24	
be	25	
as	26	
ae	27	
se	28	19

ตารางที่ 3.8 (ต่อ)

อักขระ	ลำดับชุดที่ 1	ลำดับชุดที่ 2
dt		6
at		11
tt		15
at		18
st		20
et		21

4. ทำการเปรียบเทียบชุดอักขระย่อยจนหมด จากนั้นทำการพิจารณาเฉพาะในส่วนที่มีชุดอักขระเหมือนกัน
5. เมื่อได้ชุดอักขระย่อยที่เหมือนกันทั้งหมดแล้ว ก็จะทำการพิจาณาว่าชุดใดเป็นชุดอักขระที่มีความยาวมากที่สุด ซึ่งพบลำดับที่เหมือนกันทั้งหมด 5 กลุ่มด้วยกัน คือ
    1. da dt da ds de
    2. at aa as ae
    3. ta ts te
    4. as ae
    5. se

ซึ่งจากผลที่ได้จะเห็นว่ากลุ่มหน่วยอักขระแรกมีความยาวมากกว่า ดังนั้นหน่วยอักขระดังกล่าวจึงเป็นการหาลำดับเหมือนยาวที่สุดซึ่งประกอบด้วย da , dt , da, ds, de และเมื่อทำการจัดเรียงอัขกระกลับมาเป็นรูปประโยคธรรมดา ได้คำว่า datase

จากตัวอย่างที่ 3.2 และตัวอย่างที่ 3.3 พบว่ามีลักษณะการทำงานแบบเดียวกัน แต่วิธีการแบ่งหน่วยย่อยแตกต่างกัน ทำให้ผลลัพธ์ที่ได้มีความแตกต่างกันไปด้วย

3.3 การเปรียบเทียบโดยใช้โครงสร้างของซัฟฟิกซ์อะเรย์

โครงสร้างของซัฟฟิกซ์อะเรย์ที่กล่าวในหัวข้อ 2.2.3 นั้น มีข้อดี คือ การที่ซัฟฟิกซ์อะเรย์มีดัชนีช่วยในการสืบค้นทำให้การสืบค้นข้อมูลทำได้รวดเร็ว ซึ่งขั้นตอนในการเก็บดัชนีนั้นจะทำการเก็บอักขระตัวแรก โดยในการเปรียบเทียบแบบโครงสร้างซัฟฟิกซ์อะเรย์นั้น จะมีการปรับแต่งข้อมูลก่อน เพื่อให้ได้ข้อมูลที่ไม่มีความซ้ำซ้อนและใช้งานง่าย โดยการปรับแต่งมี 2 กรณี คือ

1. ในไบแกรมที่มีการแบ่งหน่วยแล้วนั้น ถ้าไบแกรมในชุดเดียวกันแต่เป็นลำดับถัดไปซ้ำกับดัชนีที่มีอยู่เดิม ให้ทำการข้ามไปไม่ต้องเพิ่มลงในโครงสร้างซัฟฟิกซ์อะเรย์
2. สำหรับชุดอักขระที่นำมาเปรียบเทียบ เมื่อทำการแบ่งไบแกรมแล้วจะต้องตรวจสอบว่าชุดอักขระนั้น มีอยู่ในดัชนีของชุดเดิมหรือไม่ ถ้าไม่มีก็จะทำการข้ามไปไม่นำมาเปรียบเทียบ

ซึ่งในการทำงานทั้งหมดมีขั้นตอน ดังนี้

1. ทำการแบ่งหน่วยไบแกรมของอักขระที่ต้องการค้นหา
2. จัดชุดอักขระที่ทำการแบ่งแล้วลงซัฟฟิกซ์อะเรย์ โดยนำอักขระตัวแรกมาทำเป็นดัชนี
3. จัดเก็บลำดับการแบ่งข้อมูล โดยถ้ามีข้อมูลอักขระที่เป็นดัชนีซ้ำกันให้ข้ามไปทำอันถัดไป
4. นำชุดอักขระที่ต้องการเปรียบเทียบ และได้แบ่งหน่วยไว้แล้ว มาทำการเปรียบเทียบกับซัฟฟิกซ์อะเรย์ แต่ถ้าไม่มีอยู่ในดัชนีเดิมก็ให้ข้ามไป
5. จัดเก็บชุดข้อมูลที่ตรงกัน เพื่อเก็บไว้ใช้ในการหาความสัมพันธ์ในครั้งต่อไป หรือเพื่อใช้ตอบเป็นผลลัพธ์

ตัวอย่างที่ 3.4 จากตัวอย่างที่ 2.3 สำหรับการหาลำดับเหมือนยาวที่สุด

1. ทำการแบ่งหน่วยไบแกรมของอักขระดังตารางที่ 3.2 และ ตารางที่ 3.3
2. จัดชุดอักขระที่ได้ทำการแบ่งแล้วลงซัฟฟิกซ์อะเรย์

ตารางที่ 3.9 ตัวอย่างการจัดซัฟฟิกซ์อะเรย์ของชุดอักขระที่ 1

ดัชนี	หน่วยอักขระย่อย						
d	da	dt	da	db	Da	ds	de
a	at	aa	ab	aa	As	ae	
t	ta	tb	ta	ts	Te		
a	ab	aa	as	ae			
b	ba	bs	be				
a	as	ae					
s	se						

ตารางที่ 3.10 ตัวอย่างการจัดซัฟฟิกซ์อะเรย์ของชุดอักขระที่ 2

ดัชนี	หน่วยอักขระย่อย					
d	da	dt	da	ds	de	dt
a	at	aa	as	ae	at	
t	ta	ts	te	Tt		
a	as	ae	at			
s	se	st				
e	et					

3. จัดเก็บลำดับการแบ่งข้อมูล โดยถ้ามีข้อมูลอักขระที่เป็นดัชนีซ้ำกันก็จะไม่นำมาพิจารณาเช่น ในชุดอักขระที่ 1 มีดัชนีเป็น a ซ้ำกันก็จะทำการข้าม a ที่พบในครั้งหลังออกไป

ตารางที่ 3.11 การตัดชุดอักขระที่มีดัชนีซ้ำกันของชุดอักขระที่ 1

ดัชนี	หน่วยอักขระย่อย						
d	da	dt	da	db	Da	ds	de
a	at	aa	ab	aa	As	ae	
t	ta	tb	ta	ts	Te		
b	ba	bs	be				
s	se						

ตารางที่ 3.12 การตัดชุดอักขระที่มีดัชนีซ้ำกันของชุดอักขระที่ 2

ดัชนี	หน่วยอักขระย่อย					
d	da	dt	da	ds	de	dt
a	at	aa	as	ae	at	
t	ta	ts	te	Tt		
s	se	st				
e	et					

ในชุดอักขระชุดที่ 2 จะมีการตัดชุดอักขระที่มีดัชนีไม่ตรงกับชุดที่ 1 ออก เช่น ในอักขระชุดที่ 2 มีดัชนีเป็น e แต่อักขระชุดที่ 1 ไม่มีก็จะทำการตัดดัชนี e ออก

4. นำชุดอักขระที่มีดัชนีเหมือนกันมาเปรียบเทียบ เช่น นำดัชนี d มาทำการเปรียบเทียบกัน

อักขระชุดที่ 1	d	da	dt	da	db	da	ds	de
อักขระชุดที่ 2	d	da	dt	da	ds	de	dt	
ผลลัพธ์ที่ได้	d	da	dt	da	ds	de		

ซึ่งจะเก็บผลลัพธ์ไว้ใช้ในการเปรียบเทียบต่อไป จากนั้นทำขั้นตอนที่ 4 เพื่อเปรียบเทียบลำดับถัดไปที่มีดัชนีเหมือนกัน ทำเช่นนี้ไปจนกว่าการเปรียบเทียบดัชนีที่เหมือนกันจนครบทั้งหมด

5. จัดเก็บชุดข้อมูลที่ตรงกัน เพื่อเก็บไว้ใช้ในการหาความสัมพันธ์ในครั้งต่อไปโดยจัดเก็บในรูปแบบซัฟฟิกซ์อะเรย์

เมื่ออักขระที่ต้องการเปรียบเทียบหมด ก็จะทำการนำอักขระชุดใหม่ที่เปรียบเทียบได้มาจัดเรียงได้ผลดังตารางที่ 3.13

ตารางที่ 3.13 ผลลัพธ์จากการเปรียบเทียบซัฟฟิกซ์อะเรย์ทั้งหมด

ดัชนี	หน่วยอักขระย่อย				
d	da	dt	da	ds	de
a	at	aa	as	ae	
t	ta	ts	te		
s	se				

จากผลลัพธ์ที่ได้ สามารถนำมาแสดงเป็นคำตอบของการหาลำดับเหมือนยาวที่สุด คือ ชุดดัชนี d มีอักขระย่อยคือ da, dt, da, ds, de ซึ่งก็คือ datase เป็นผลลัพธ์ หรือถ้าต้องการเปรียบเทียบต่อไปก็สามารถนำตารางที่ 3.13 ไปเป็นตัวตั้งต้นในการเปรียบเทียบต่อไปได้ ซึ่งจะนำไปดำเนินการต่อในข้อที่ 4 ซึ่งสามารถรองรับข้อมูลเข้าได้เป็นจำนวน n ตัว

บทที่ 4
การประเมินผล

ในบทนี้เป็นการนำเสนอการทดสอบวัดประสิทธิภาพของแนวคิดใหม่ทั้ง 2 วิธี โดยทำการพัฒนาโปรแกรมต้นแบบเพื่อนำมาทดสอบวิธีการหาอักขระย่อยเหมือนยาวที่สุด และวิธีการหาลำดับเหมือนยาวที่สุดโดยให้สามารถนับจำนวนครั้งในการเปรียบเทียบอักขระได้

4.1 ข้อมูลที่ใช้ในการทดลอง

ข้อมูลที่ใช้ในการทดลองเป็นชุดข้อมูลสตริงบนเว็บไซต์ต่างๆ โดยข้อมูลที่ใช้ในการทดลองแต่ละชุดจะมีความแตกต่างกัน สามารถใช้สื่อถึงการพิจารณา และการเลือกใช้งานของวิธีการเปรียบเทียบแต่ละวิธีที่มีความแตกต่างกันไป

ชุดอักขระกลุ่มแรกจะเป็นชุดอักขระประเภทกลุ่มคำ ซึ่งนำมาจากกลุ่มคำศัพท์ภาษาอังกฤษทั้งหมด 10 คำโดยทำการเปรียบเทียบทุกคำ ชุดอักขระชุดนี้สามารถแทนการเปรียบเทียบในการค้นหาคำสำคัญในเว็บไซต์ต่างๆ เพื่อทำการค้นหาพจนานุกรม

ชุดอักขระกลุ่มที่ 2 เป็นชุดอักขระในรูปแบบของดีเอ็นเอเป็นชุดอักขระที่มีความยาวและมีการเรียงกันอย่างซับซ้อน ซึ่งเป็นการทำงานเพื่อหาข้อจำกัดในการเปรียบเทียบอักขระว่าสามารถรองรับกับชุดอักขระที่มีจำนวนมาก และมีความซับซ้อนสูงได้มากน้อยเพียงใด

ข้อมูลอักขระ	แหล่งที่มา	จำนวน
คำศัพท์	กลุ่มคำศัพท์ในภาษาอังกฤษ	10 ชุด
ดีเอ็นเอของโปรตีน	http://workbench.sdsc.edu/	3 ชุด

ข้อมูลคำศัพท์ที่ใช้ในการทดลอง ประกอบด้วย

database,	computer,	dataset,	monitors,	printer,
datacenter,	hacker,	adapter,	operator,	character

ข้อมูลดีเอ็นเอที่ใช้ในการทดลอง ประกอบด้วย

ข้อมูลชุดที่ 1

MSALGAVIALLLWGQLFAVDSGNDVTDIADDGCPKPPEIAHGYVEHSVRYQCKNYYKL
RTEGDGVYTLNDKKQWINKAVGDKLPECEADDGCPKPPEIAHGYVEHSVRYQCKNYYK
LRTEGDGVYTLNNEKQWINKAVGDKLPECEAVCGKPKNPANPVQRILGGHLDAKGSFP
WQAKMVSHHNLTTGATLINEQWLLTTAKNLFLNHSENATAKDIAPTLTLYVGKKQLVEI
EKVVLHPNYSQVDIGLIKLKQKVSVNERVMPICLPSKDYAEVGRVGYVSGWGRNANFK
FTDHLKYVMLPVADQDQCIRHYEGSTVPEKKTPKSPVGVQPILNEHTFCAGMSKYQEDT
CYGDAGSAFAVHDLEEDTWYATGILSFDKSCAVAEYGVYVKVTSIQDWVQKTIAEN

ข้อมูลชุดที่ 2

MACRGGAGNGHRASATLSRVSPGSLYTCRTRTHNICMVSDFFYPNMGGVESHIYQLSQC
LIERGHKVIIVTHAYGNRKGIRYLTSGLKVYYLPLKVMYNQSTATTLFHSLPLLRYIFVRE
RVTIIHSHSSFSAMAHDALFHAKTMGLQTVFTDHSLFGFADVSSVLTNKLLTVSLCDTNH
IICVSYTSKENTVLRAALNPEIVSVIPNAVDPTDFTPDPFRRHDSITIVVVSRLVYRKGIDLL
SGIIPELCQKYPDLNFIIGGEGPKRIILEEVRERYQLHDRVRLLGALEHKDVRNVLVQGHIF
LNTSLTEAFCMAIVEAASCGLQVVSTRVGGIPEVLPENLIILCEPSVKSLCEGLEKAIFQLK
SGTLPAPENIHNIVKTFYTWRNVAERTEKVYDRVSVEAVLPMDKRLDRLISHCGPVTGYI
FALLAVFNFLFLIFLRWMTPDSIIDVAIDATGPRGAWTNNYSHSKRGGENNEISETR

ข้อมูลชุดที่ 3

MSKLRMVLLEDSGSADFRRHFVNLSPFTITVVLLLSACFVTSSLGGTDKELRLVDGENKC
SGRVEVKVQEEWGTVCNNGWSMEAVSVICNQLGCPTAIKAPGWANSSAGSGRIWMDH
VSCRGNESALWDCKHDGWGKHSNCTHQQDAGVTCSDGSNLEMRLTRGGNMCSGRIEI
KFQGRWGTVCDDNFNIDHASVICRQLECGSAVSFSGSSNFGEGSGPIWFDDLICNGNESA
LWNCKHQGWGKHNCDHAEDAGVICSKGADLSLRLVDGVTECSGRLEVRFQGEWGTIC
DDGWDSYDAAVACKQLGCPTAVTAIGRVNASKGFGHIWLDSVSCQGHEPAVWQCKHH
EWGKHYCNHNEDAGVTCSDGSDLELRLRGGGSRCAGTVEVEIQRLLGKVCDRGWGLK
EADVVCRQLGCGSALKTSYQVYSKIQATNTWLFLSSCNGNETSLWDCKNWQWGGLTC
DHYEEAKITCSAHREPRLVGGDIPCSGRVEVKHGDTWGSICDSDFSLEAASVLCRELQCG
TVVSILGGAHFGEGNGQIWAEEFQCEGHESHLSLCPVAPRPEGTCSHSRDVGVVCSRYTE
IRLVNGKTPCEGRVELKTLGAWGSLCNSHWDIEDAHVLCQQLKCGVALSTPGGARFGK
GNGQIWRHMFHCTGTEQHMGDCPVTALGASLCPSEQVASVICSGNQSQTLSSCNSSSLG
PTRPTIPEESAVACIESGQLRLVNGGGRCAGRVEIYHEGSWGTICDDSWDLSDAHVVCRQ

LGCGEAINATGSAHFGEGTGPIWLDEMKCNGKESRIWQCHSHGWGQQNCRHKEDAGVI
CSEFMSLRLTSEASREACAGRLEVFYNGAWGTVGKSSMSETTVGVVCRQLGCADKGKI
NPASLDKAMSIPMWVDNVQCPKGPDTLWQCPSSPWEKRLASPSEETWITCDNKIRLQEG
PTSCSGRVEIWHGGSWGTVCDDSWDLDDAQVVCQQLGCGPALKAFKEAEFGQGTGPIW
LNEVKCKGNESSLWDCPARRWGHSECGHKEDAAVNCTDISVQKTPQKATTGRSSRQSSF
IAVGILGVVLLAIFVALFFLTKKRRQRQRLAVSSRGENLVHQIQYREMNSCLNADDLDLM
NSSENSHESADFSAAELISVSKFLPISGMEKEAILSHTEKENGNL

4.2 เครื่องมือที่ใช้ในการทดลอง

เครื่องมือที่ใช้ในการทดสอบ มีดังต่อไปนี้

หน่วยประมวลผลกลาง (CPU)	: Intel Pentium M processor 1.5 MHz
หน่วยความจำหลัก(RAM)	: 512 MB
หน่วยความจำสำรอง(Hard Disk)	: 60 GB
ระบบปฏิบัติการ (OS)	: Windows XP Professional Version 2003
โปรแกรมที่ใช้ในการพัฒนา	: Eclipse IDE Version 3.2
ภาษาที่ใช้ในการพัฒนา	: JAVA

4.3 การทดลอง

การทดลองในวิทยานิพนธ์นี้ จะทำการทดลองเพื่อวิเคราะห์การใช้งานไบแกรมในการเปรียบเทียบคู่อันดับ โดยสามารถลดจำนวนครั้งในการเปรียบเทียบตัวอักขระ โดยวิธีการเปรียบเทียบ มีดังนี้

1. เปรียบเทียบข้อมูลคำศัพท์ทั้งหมด 10 ชุดข้อมูล โดยทำการเปรียบเทียบทั้งหมด ดังนี้

ข้อมูลชุดที่ 1	ข้อมูลชุดที่ 2
ข้อมูลชุดที่ 1	ข้อมูลชุดที่ 3
...	...
ข้อมูลชุดที่ 1	ข้อมูลชุดที่ n

ข้อมูลชุดที่ 2	ข้อมูลชุดที่ 1
ข้อมูลชุดที่ 2	ข้อมูลชุดที่ 3
...	...

ข้อมูลชุดที่ n-1	ข้อมูลชุดที่ n

สำหรับการเปรียบเทียบนั้นจะไม่เปรียบเทียบข้อมูลชุดเดียวกัน และการเปรียบเทียบจะทำโดยใช้การหาอักขระย่อยเหมือนยาวที่สุดกับวิธีการเปรียบเทียบ 2 แบบ คือ การหาแบบโปรแกรมพลวัต และการพิจารณาจากลำดับ จากนั้นใช้วิธีการหาลำดับเหมือนยาวที่สุดกับวิธีการเปรียบเทียบ 3 แบบ คือ การหาแบบโปรแกรมพลวัต การหาแบบใช้การพิจารณาจากลำดับ และการหาแบบใช้โครงสร้างซัฟฟิกซ์อะเรย์

2. นำผลลัพธ์ที่ได้จากวิธีการเปรียบเทียบทั้ง 3 วิธีมาทำการเปรียบเทียบตามอัตราส่วนร้อยละเพื่อพิจารณารูปแบบของข้อมูล และผลลัพธ์ที่ได้ว่าแต่ละวิธีนั้นเหมาะสมกับการหาในรูปแบบใด โดยการเปรียบเทียบจะมี 3 ลักษณะดังนี้

1. เปรียบเทียบจำนวนครั้งในการหาผลลัพธ์ ระหว่างการใช้โปรแกรมพลวัต กับ การหาแบบพิจารณาจากลำดับ
2. เปรียบเทียบจำนวนครั้งในการหาผลลัพธ์ ระหว่างการหาแบบโปรแกรมพลวัต กับการหาแบบโครงสร้างซัฟฟิกซ์อะเรย์
3. เปรียบเทียบจำนวนครั้งในการหาผลลัพธ์ ระหว่างการหาแบบพิจารณาจากลำดับ กับการหาแบบโครงสร้างซัฟฟิกซ์อะเรย์

4.4 ผลทดลอง

4.4.1 การเปรียบเทียบหาอักขระย่อยเหมือนยาวที่สุด

จากข้อมูลที่ใช้ในการทดสอบชุดคำศัพท์ได้นำมาทำการเปรียบเทียบระหว่างการหาแบบใช้โปรแกรมพลวัต กับวิธีการหาแบบพิจารณาจากลำดับ เพื่อใช้ในการหาอักขระย่อยเหมือนยาวที่สุด ผลลัพธ์ที่ใช้ในการเปรียบเทียบนั้น จะเป็นจำนวนครั้งในการเปรียบเทียบแต่ละอักขระ ดังปรากฏในตารางที่ 4.1 จากตารางที่ 4.1 พื้นที่สีเท่าเป็นจำนวนครั้งในการเปรียบเทียบโดยใช้โปรแกรมแบบพลวัต และพื้นที่สีขาวเป็นการเปรียบเทียบโดยใช้การพิจารณาจากลำดับ จะเห็นได้ว่าพื้นที่สีขาวใช้จำนวนครั้งในการเปรียบเทียบได้น้อยกว่าพื้นที่สีเทา และสามารถแสดงผลเป็นกราฟเส้นได้ดังปรากฏในรูปที่ 4.1 ซึ่งการแสดงผลของกราฟนั้น ได้ทำการยกตัวอย่างข้อมูลที่ใช้ในการเปรียบเทียบมาทั้งหมด 18 ค่า เพื่อให้สามารถแสดงผลได้อย่างชัดเจน หลังจากนั้นเมื่อนำจำนวนครั้งการ

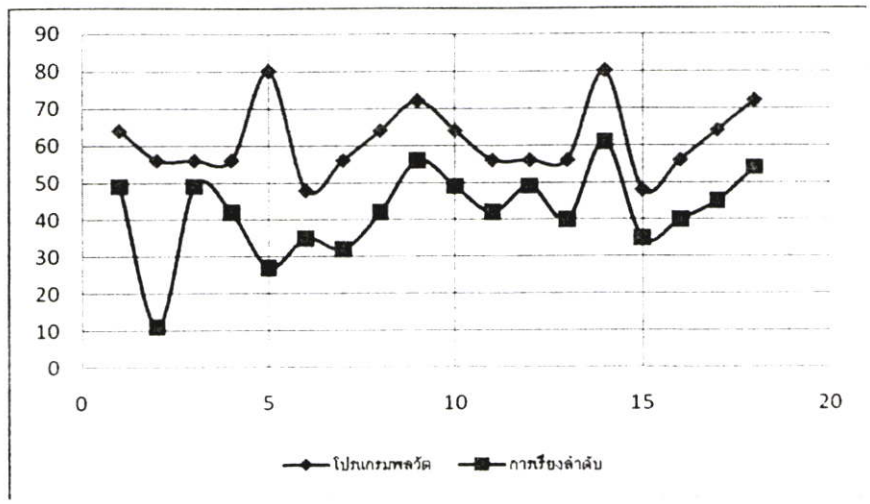
เปรียบเทียบมาคิดหาอัตราการเปลี่ยนแปลงเป็นร้อยละ จะได้ผลลัพธ์ดังตารางที่ 4.2 ซึ่งเห็นได้ว่าไม่มีชุดข้อมูลชุดใดเลยที่ผลลัพธ์มีค่าน้อยกว่า 1 แสดงให้เห็นว่าวิธีการนี้จะทำให้จำนวนครั้งในการเปรียบเทียบลดน้อยลง เมื่อนำเอาผลการวัดจำนวนครั้งของทั้ง 2 วิธีมาเปรียบเทียบจะเห็นได้ว่าอัตราส่วนร้อยละที่ลดลงของวิธีที่ 2 นั้นไม่ได้ลดลงอย่างสม่ำเสมอ บางตำแหน่งการลดจะมีอัตราการลดสูง บางตำแหน่งการลดมีอัตราการลดต่ำ จากการวิเคราะห์วิธีการเปรียบเทียบแบบการพิจารณาลำดับนั้น พบว่ามีองค์ประกอบในการลดจำนวนการเปรียบเทียบอยู่ด้วยกัน 2 ประการ ดังนี้

1. การแบ่งหน่วยไบแกรมจะช่วยให้ความยาวของอักขระนั้นลดลง 1 ค่า
2. ระหว่างการเปรียบเทียบ ถ้าข้อมูลตัวใดมีการเปรียบเทียบผ่านไปแล้วจะไม่ทำการเปรียบเทียบซ้ำ

ซึ่งจากการเปรียบเทียบคำศัพท์ทั้ง 10 ชุด จำนวน 90 ครั้ง พบว่าการใช้การเปรียบเทียบแบบพิจาณาจากลำดับสามารถช่วยลดจำนวนครั้งในการเปรียบเทียบลงได้เป็นค่าเฉลี่ยถึง 1.45 เท่า และสามารถนำข้อมูลชุดที่เปรียบเทียบแล้วไปเปรียบเทียบกับชุดอักขระอื่นที่ต้องการได้อีก

ตารางที่ 4.1 ผลลัพธ์แสดงจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัต และแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาอักขระย่อยเหมือนยาวที่สุด

คำศัพท์	1	2	3	4	5	6	7	8	9	10
1		64	56	56	56	80	48	56	64	72
2	49		56	56	56	80	48	56	64	72
3	11	42		49	49	70	42	49	56	63
4	49	49	42		49	70	42	49	56	63
5	42	40	36	42		70	42	49	56	63
6	27	61	21	63	48		60	70	80	90
7	35	35	30	35	30	37		42	48	54
8	32	40	27	42	34	40	30		56	63
9	42	45	36	45	38	51	31	38		72
10	56	54	48	56	46	62	24	46	46	



รูปที่ 4.1 กราฟแสดงจำนวนครั้งในการเปรียบเทียบการหาอักขระย่อยเหมือนยาวที่สุด

ตารางที่ 4.2 ตารางอัตราส่วนร้อยละจากจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัต และแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาอักขระย่อยเหมือนยาวที่สุด

1										
2	1.31									
3	5.09	1.33								
4	1.14	1.14	1.17							
5	1.33	1.40	1.36	1.17						
6	2.96	1.31	3.33	1.11	1.46					
7	1.37	1.37	1.40	1.20	1.40	1.62				
8	1.75	1.40	1.81	1.17	1.44	1.75	1.40			
9	1.52	1.42	1.56	1.24	1.47	1.57	1.55	1.47		
10	1.29	1.33	1.31	1.13	1.37	1.45	2.25	1.37	1.57	
คำศัพท์	1	2	3	4	5	6	7	8	9	10

4.4.2 การเปรียบเทียบหาลำดับเหมือนยาวที่สุด

จากข้อมูลที่ใช้ในการทดสอบชุดคำศัพท์ได้นำมาทำการเปรียบเทียบระหว่างการหาแบบใช้โปรแกรมพลวัต กับวิธีการหาแบบพิจาณาจากลำดับ เพื่อใช้ในการหาลำดับเหมือนยาวที่สุด ผลลัพธ์ที่ใช้ในการเปรียบเทียบนั้นจะเป็นจำนวนครั้งในเปรียบเทียบแต่ละอักขระดังปรากฏในตารางที่ 4.3 และรูปที่ 4.2 โดยผลลัพธ์ที่ได้นั้น จากตารางที่ 4.3 พื้นที่สีเท่าคือจำนวนครั้งในการเปรียบเทียบโดย

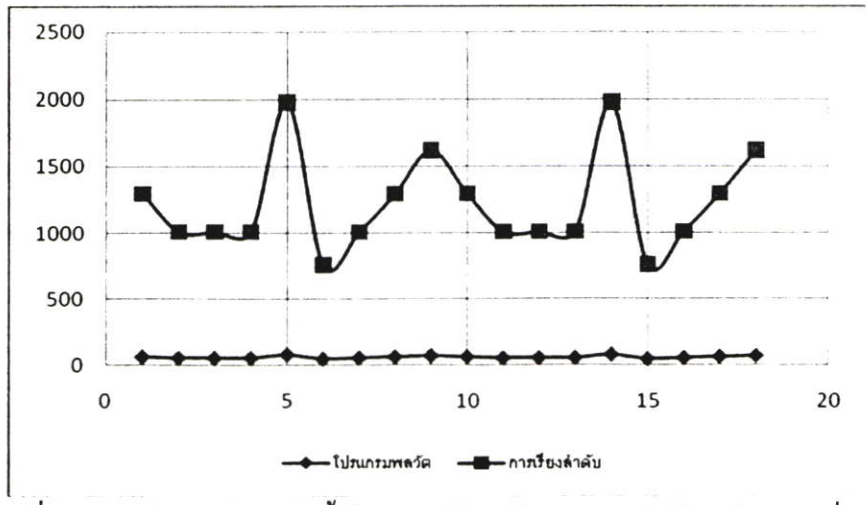
ใช้โปรแกรมแบบพลวัต และพื้นที่สีขาวเป็นการเปรียบเทียบโดยใช้การพิจารณาจากลำดับ จะสังเกตได้ว่า จำนวนการเปรียบเทียบในพื้นที่สีขาวนั้นมีมากกว่าพื้นที่สีเทามาก ซึ่งแสดงการเปรียบเทียบเป็นอัตราร้อยละได้ดังตารางที่ 4.4 ทำให้สามารถสรุปได้ว่า การใช้การพิจารณาแบบลำดับ ไม่เหมาะกับการวิเคราะห์หาลำดับเหมือนยาวที่สุดเนื่องจากไม่สามารถลดขั้นตอนการเปรียบเทียบอักขระได้ และยังทำให้การเปรียบเทียบมีจำนวนเพิ่มมากขึ้น โดยเหตุผลในการเพิ่มขึ้นของจำนวนการเปรียบเทียบ คือ

1. ในการแบ่งหน่วยไบแกรมเพื่อใช้ในการเปรียบเทียบหาลำดับเหมือนยาวที่สุด จำนวนหน่วยที่ได้จากการแบ่งมีค่ามากถึง $\sum N$
2. การเปรียบเทียบแต่ละหน่วยไม่สามารถทำข้ามส่วนที่มีการเปรียบเทียบไปแล้วได้ เพราะว่าการหาลำดับเหมือนยาวที่สุดจะสามารถเจออักขระได้ในทุกๆ ตำแหน่ง

จากการเปรียบเทียบคำศัพท์ทั้ง 10 ชุด จำนวน 90 ครั้ง พบว่าการใช้การเปรียบเทียบแบบพิจาณาจากลำดับนั้น มีการเพิ่มจำนวนครั้งในการเปรียบเทียบมากขึ้นเป็นค่าเฉลี่ยได้สูงถึง 19.49 เท่า จากค่าเฉลี่ยที่ได้ จึงสรุปได้ว่าการพิจาณาแบบลำดับไม่เหมาะกับการหาลำดับเหมือนยาวที่สุด

ตารางที่ 4.3 ผลลัพธ์แสดงจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัต และแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด

คำศัพท์	1	2	3	4	5	6	7	8	9	10
1		64	56	56	56	80	48	56	64	72
2	1296		56	56	56	80	48	56	64	72
3	1008	1008		49	49	70	42	49	56	63
4	1008	1008	784		49	70	42	49	56	63
5	1008	1008	784	784		70	42	49	56	63
6	1980	1980	1540	1540	1540		60	70	80	90
7	756	756	588	588	588	1155		42	48	54
8	1008	1008	784	784	784	1540	588		56	63
9	1296	1296	1008	1008	1008	1980	756	1008		72
10	1620	1620	1260	1260	1260	2475	945	1260	1620	



รูปที่ 4.2 กราฟแสดงจำนวนครั้งในการเปรียบเทียบการหาลำดับเหมือนยาวที่สุด

ตารางที่ 4.4 ตารางแสดงอัตราส่วนร้อยละจากจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัต และแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด

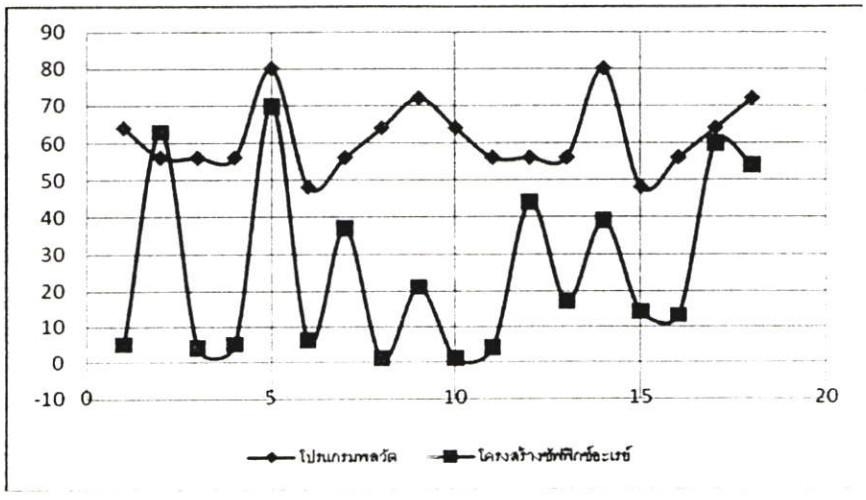
1										
2	20.25									
3	18	18								
4	18	18	16							
5	18	18	16	16						
6	24.75	24.75	22	22	22					
7	15.75	15.75	14	14	14	19.25				
8	18	18	16	16	16	22	14			
9	20.25	20.25	18	18	18	24.75	15.75	18		
10	22.5	22.5	20	20	20	27.5	17.5	20	22.5	
คำศัพท์	1	2	3	4	5	6	7	8	9	10

จากการเปรียบเทียบโดยใช้การพิจารณาแบบลำดับ พบว่าไม่สามารถลดขั้นตอนการเปรียบเทียบอักขระได้ ทั้งนี้ผลลัพธ์ที่ได้จากการแบ่งไบแกรมนั้นจะต้องนำมาเปรียบเทียบกันทุกๆ หน่วย จึงทำให้เกิดความซ้ำซ้อน ซึ่งวิธีการลดการเปรียบเทียบนี้สามารถทำได้โดยการนำโครงสร้างซัฟฟิกซ์อะเรย์เข้ามาช่วย เนื่องจากการใช้โครงสร้างซัฟฟิกซ์อะเรย์นั้นมีการกำหนดดัชนีซึ่งทำให้การเปรียบเทียบอักขระจะทำการเปรียบเทียบเฉพาะส่วนที่มีดัชนีเหมือนกัน ดังปรากฏในตารางที่ 4.5

จากตารางที่ 4.5 พื้นที่สีเทาและขาว คือ การเปรียบเทียบโดยใช้โครงสร้างซัฟฟิกซ์อะเรย์เพียงอย่างเดียว จะสังเกตเห็นว่าจำนวนในการเปรียบเทียบนั้นลดลง เมื่อนำไปเปรียบเทียบกับตารางที่ 4.3 ไม่ว่าจะเป็นพื้นที่สีเทาในตารางที่ 4.3 ในส่วนของการใช้โปรแกรมแบบพลวัต ซึ่งแสดงผลการเปรียบเทียบเป็นรูปที่ 4.3 และนำไปเปรียบเทียบเป็นอัตราร้อยละได้เป็นตารางที่ 4.6 หรือเมื่อนำไปเปรียบเทียบกับพื้นที่สีขาวในตารางที่ 4.3 ในส่วนของการใช้การพิจารณาจากลำดับ จะพบว่าสามารถลดจำนวนการเปรียบเทียบลงได้มาก

ตารางที่ 4.5 ผลลัพธ์แสดงจำนวนครั้งในการเปรียบเทียบแบบโครงสร้างซัฟฟิกซ์อะเรย์กับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด

คำศัพท์	1	2	3	4	5	6	7	8	9	10
1		1	50	3	1	34	21	40	25	30
2	5		3	26	14	23	6	10	23	20
3	63	4		6	4	35	10	42	21	30
4	4	44	2		15	11	0	2	20	2
5	5	17	3	20		20	1	13	24	16
6	70	39	53	16	22		36	82	48	82
7	6	14	4	0	1	25		12	7	33
8	37	13	28	2	14	67	19		35	26
9	1	60	3	28	24	30	7	28		20
10	21	54	15	4	16	94	68	42	36	



รูปที่ 4.3 กราฟแสดงจำนวนครั้งในการเปรียบเทียบการหาลำดับเหมือนยาวที่สุด

ตารางที่ 4.6 ตารางแสดงอัตราส่วนร้อยละจากจำนวนครั้งในการเปรียบเทียบจากแบบการใช้โปรแกรมพลวัตและแบบพิจารณาจากลำดับกับข้อมูลทดสอบชุดคำศัพท์ในการหาลำดับเหมือนยาวที่สุด

คำศัพท์	1	2	3	4	5	6	7	8	9	10
1		64	1.12	18.67	56	2.35	2.29	1.40	2.56	2.40
2	12.80		18.67	2.15	4	3.48	8	5.60	2.78	3.60
3	**0.89**	14		8.17	12.25	2	4.20	1.17	2.67	2.10
4	14	1.27	24.50		3.27	6.36	42	24.50	2.80	31.50
5	11.20	3.29	16.33	2.45		3.50	42	3.77	2.33	3.94
6	1.14	2.05	1.32	4.38	3.18		1.67	**0.85**	1.67	1.10
7	8	3.43	10.50	42	42	2.40		3.50	6.86	1.64
8	1.51	4.31	1.75	24.50	3.50	1.04	2.21		1.60	2.42
9	64	1.07	18.67	2.00	2.33	2.67	6.86	2		3.60
10	3.43	1.33	4.20	15.75	3.94	**0.96**	**0.79**	1.50	2	

จากผลลัพธ์ในรูปที่ 4.3 จะสังเกตได้ว่ามีกราฟบางจุดของการพิจารณาแบบโครงสร้างซัฟฟิกซ์อะเรย์สูงกว่าการพิจาราณาแบบโปรแกรมพลวัต และจากตารางที่ 4.6 เมื่อนำมาเปรียบเทียบหาในอัตราส่วนร้อยละจะพบว่ามีบางตำแหน่งเมื่อคำนวณหาอัตราส่วนแล้วมีค่าน้อยกว่า 1 ซึ่งค่าที่น้อยกว่า 1 นั้นถือเป็นค่าที่ใช้การเปรียบเทียบมากกว่าจำนวนครั้งในการเปรียบเทียบของการใช้โปรแกรมพลวัติ ซึ่งมีด้วยกันทั้งหมด 4 ตำแหน่ง ดังตัวเลขที่ขีดเส้นใต้ในตาราง ซึ่งสาเหตุที่ทำให้จำนวนครั้งในการเปรียบเทียบมากกว่าประกอบด้วย 2 สาเหตุ คือ

1. ลำดับก่อนหลังในการเปรียบเทียบ
2. ความคล้ายกันของชุดอักขระ

ลำดับก่อนหลังที่นำชุดอักขระเข้าไปเปรียบเทียบถือว่ามีผลต่อการเปรียบเทียบ ซึ่งในกรณีนี้เกิดขึ้นเนื่องจากในขั้นตอนการปรับแต่งข้อมูลให้อยู่ในโครงสร้างซัฟฟิกซ์อะเรย์นั้น จะทำการตัดในส่วนของอักขระที่เหมือนกันในอักขระชุดเดียวกันออก และตัดอักขระที่ไม่เหมือนกันกับอักขระชุดแรกออก ซึ่งการตัดอักขระทั้งสองชุดนี้มีผลต่อจำนวนครั้งในการเปรียบเทียบ เช่น การเปรียบเทียบชุดอักขระที่ 6 กับ 10 ถ้าชุดอักขระที่ 6 นำมาเปรียบเทียบก่อน สามารถตัดอักขระที่เหมือนกันในชุดได้ ผลลัพธ์ของชุดอักขระที่เหลือในการเปรียบเทียบจึงเป็น d, a, t, c, e, n และ r จำนวน 7 ชุด และเมื่อนำอักขระชุดที่ 10 มาทำการตัดอักขระที่เหมือนกันในชุดตัวเอง และอักขระที่ไม่เหมือนกันในชุดแรกออกนั้น ผลลัพธ์ที่ได้จะเป็น c, a, r, t และ e จำนวน 5 ชุด ซึ่งเป็นการ

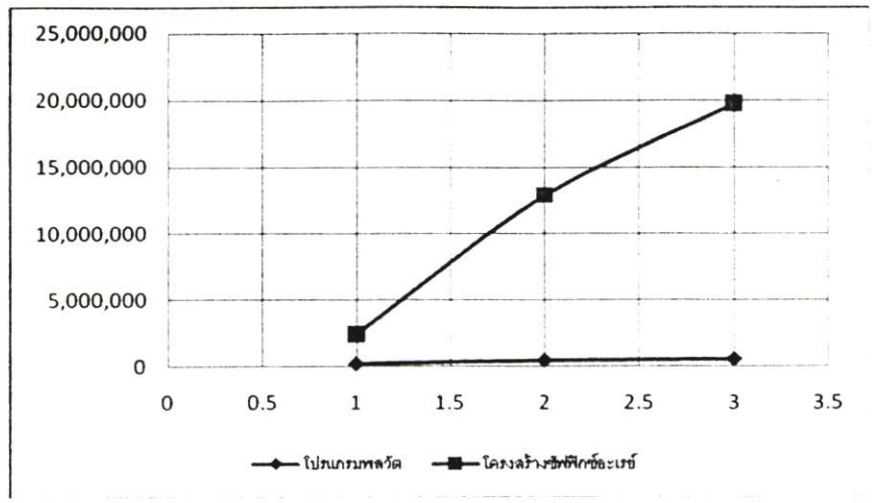
เปรียบเทียบโดยใช้จำนวน 7 ชุด กับ 5 ชุดในการเปรียบเทียบ ในทางกลับกันหากนำชุดอักขระที่ 10 มาเป็นตัวเปรียบเทียบก่อนจะทำให้การตัดอักขระเหลือผลลัพธ์เป็น c, h, a, r, c, t, e และ r มีจำนวน 8 ชุด โดยชุดอักขระที่ 6 สามารถตัดได้เป็น a, t, c, e, และ r จำนวน 5 ชุดทำให้เป็นการเปรียบเทียบโดยใช้จำนวน 8 ชุด กับ 5 ชุด ซึ่งจากตัวเลขของทั้ง 2 กรณี แสดงให้เห็นว่าจำนวนรอบในการเปรียบเทียบแตกต่างกัน ทำให้การเปรียบเทียบได้ผลลัพธ์ที่แตกต่างกัน

ในส่วนของความคล้ายกันของชุดอักขระนั้น ถ้าชุดอักขระที่ใช้ในการเปรียบเทียบทั้ง 2 ชุด มีความคล้ายกันมากจะทำให้โอกาสในการตัดคำในส่วนของการปรับแต่งข้อมูลที่นำเอาชุดอักขระชุดที่สองไปตัดกับชุดอักขระชุดแรกทำได้น้อยลงซึ่งจะทำให้จำนวนครั้งในการเปรียบเทียบยังคงมีสูงอยู่ แต่ถ้าชุดอักขระทั้ง 2 ชุดมีความแตกต่างกันมาก จะทำให้การตัดอักขระที่ไม่เหมือนกันทั้งสองชุดออกไปได้เป็นจำนวนมาก ทำให้รอบในการเปรียบเทียบลดลง และทำให้จำนวนครั้งในการเปรียบเทียบน้อยลงตามไปด้วย เช่น การเปรียบเทียบระหว่างอักขระชุดที่ 7 กับอักขระชุดที่ 10 โดยใช้อักขระชุดที่ 10 เป็นลำดับที่สองในการเปรียบเทียบ ทำให้อักขระชุดที่ 10 เมื่อลดลงแล้วจะเป็น c, h, a, r และ e ทำให้มีจำนวนรอบในการเปรียบเทียบมากกว่า ตัวอย่างการเปรียบเทียบอักขระชุดที่ 3 กับ ชุดที่ 7 โดยให้ชุดที่ 7 เป็นอักขระที่จะเปรียบเทียบเป็นลำดับที่ 2 ทำให้อักขระชุดที่สองที่นำไปเปรียบเทียบเหลือเพียง 2 ตัวคือ a และ e เท่านั้น

ในส่วนของชุดอักขระแบบที่ 2 คือการใช้ดีเอ็นเอเพื่อเปรียบเทียบนั้น ผลของการเปรียบเทียบแสดงอยู่ในตารางที่ 4.7 และรูปที่ 4.4 โดยเป็นผลของการเปรียบเทียบการหาด้วยโปรแกรมพลวัต และการเปรียบเทียบโดยใช้โครงสร้างซัฟฟิกซ์อะเรย์ ซึ่งเมื่อหาอัตราการเปลี่ยนแปลงเป็นร้อยละแล้วจะเห็นถึงความแตกต่างของทั้ง 2 วิธี ซึ่งสามารถสรุปได้ว่าการที่อักขระมีความเหมือนกันมากจะทำให้การใช้การเปรียบเทียบแบบซัฟฟิกซ์อะเรย์นั้น มีค่าในการเปรียบเทียบที่สูงขึ้นมาก

ตารางที่ 4.7 ตารางแสดงจำนวนครั้งในการเปรียบเทียบของชุดอักขระดีเอ็นเอของทั้ง 2 วิธี

ชุดของข้อมูล	ชุดของข้อมูล	โปรแกรมพลวัต	โครงสร้างซัฟฟิกซ์อะเรย์	อัตราการเปลี่ยนแปลง (ร้อยละ)
ชุดที่ 1	ชุดที่ 2	196,504	2,444,513	12.44
ชุดที่ 1	ชุดที่ 3	469,336	12,856,389	27.39
ชุดที่ 2	ชุดที่ 3	559,504	19,786,183	35.36



รูปที่ 4.4 กราฟแสดงจำนวนครั้งในการเปรียบเทียบของวิธีคิดทั้ง 2 วิธี

จากการเปรียบเทียบคำศัพท์ทั้ง 10 ชุด จำนวน 90 ครั้งพบว่าการใช้การเปรียบเทียบแบบโครงสร้างซัฟฟิกซ์อะเรย์นั้น ทำให้จำนวนครั้งของการเปรียบเทียบลดลงโดยเฉลี่ยถึง 3 เท่า แต่ถ้านำไปเปรียบเทียบโดยใช้ชุดข้อมูลดีเอ็นเอทำให้จำนวนการเปรียบเทียบเพิ่มมากขึ้น ซึ่งจากการทดลองนี้สามารถสรุปได้ว่า การเปรียบเทียบแบบโครงสร้างซัฟฟิกซ์อะเรย์เหมาะสมกับการหาลำดับเหมือนยาวที่สุดกับรูปแบบอักขระที่มีความแตกต่างกันมาก แต่ไม่เหมาะสมกับการหาชุดอักขระที่คล้ายกันมาก เพราะทำให้จำนวนการเปรียบเทียบอักขระมีจำนวนมากขึ้น

จากการทดลองที่ผ่านมา ทั้งการหาอักขระย่อยเหมือนยาวที่สุด และการหาลำดับเหมือนยาวที่สุดโดยใช้วิธีการเปรียบเทียบ 3 แบบ คือ การหาแบบโปรแกรมแบบพลวัต การหาแบบใช้การพิจารณาจากลำดับ และการหาแบบใช้โครงสร้างซัฟฟิกซ์อะเรย์ สามารถสรุปผลได้ดังตารางที่ 4.8

ตารางที่ 4.8 สรุปข้อดีข้อเสียของการวิเคราะห์ทั้ง 3 วิธี

วิธีการ	ข้อดี	ข้อเสีย
โปรแกรมแบบพลวัต	- สามารถที่จะใช้งานกับโครงสร้างอักขระที่มีความคล้ายกันได้ดี	- ยากต่อการปรับให้ใช้งานในแบบการเปรียบเทียบชุดอักขระมากกว่า 2 ชุดขึ้นไป
การพิจารณาจากการเรียง-ลำดับ	- เป็นการทำงานที่ไม่จำเป็นต้องทำการปรับแต่งข้อมูลก่อน และได้ผลลัพธ์ที่ตรงกับความต้องการ - สะดวกต่อการเปรียบเทียบชุดอักขระมากกว่า 2 ชุดขึ้นไป - สามารถใช้งานได้ดีกับการหาอักขระย่อยเหมือนยาวที่สุด	- ใช้เวลามากเกินไปในการเปรียบเทียบชุดอักขระ ที่ใช้ในการหาลำดับเหมือนยาวที่สุด
การใช้ซัฟฟิกซ์อะเรย์	- สามารถเปรียบเทียบชุดอักขระที่มีความแตกต่างกันได้รวดเร็ว - สะดวกต่อการเปรียบเทียบชุดอักขระมากกว่า 2 ชุดขึ้นไป - สามารถใช้งานได้ดีกับการเปรียบ-เทียบหาลำดับเหมือนยาวที่สุด ที่อักขระมีความหลากหลายมาก	- จำเป็นต้องทำการปรับแต่งข้อมูลก่อนในการสร้างซัฟฟิกซ์อะเรย์ ซึ่งเป็นการสิ้นเปลืองเวลา

บทที่ 5
สรุป และข้อเสนอแนะ

5.1 สรุป

วิทยานิพนธ์ฉบับนี้ได้ทำการศึกษาทดลองวิธีการหาอักขระย่อยเหมือนยาวที่สุด และวิธีการหาลำดับเหมือนยาวที่สุดโดยทำการทดลองเพื่อเปรียบเทียบจำนวนครั้งของการทำงานจากการเปรียบเทียบ 3 วิธี คือ การหาแบบโปรแกรมพลวัต การหาแบบใช้การพิจารณาจากลำดับ และการหาแบบใช้โครงสร้างซัฟฟิกซ์อะเรย์ ซึ่งวิธีการหาแบบโปรแกรมพลวัต เป็นวิธีการเปรียบเทียบแบบเดิม ส่วนการหาแบบใช้การพิจารณาจากลำดับ และการหาแบบใช้โครงสร้างซัฟฟิกซ์อะเรย์เป็นวิธีการเปรียบเทียบแบบใหม่ สำหรับการพัฒนาวิธีการเปรียบเทียบทั้ง 3 วิธีดังกล่าว ได้ทำการพัฒนาด้วยภาษา JAVA และในส่วนของการพัฒนาจะทำการนับจำนวนครั้งในการเปรียบเทียบเพื่อให้สะดวกและง่ายต่อการนำมาวิเคราะห์ ผลจากการวิเคราะห์นั้น สามารถแสดงให้เห็นถึงความสามารถ และความเหมาะสม สำหรับการนำไปใช้งานของแต่ละวิธี โดยจากการทดลอง และการวิเคราะห์พบว่าวิธีการพิจารณาจากลำดับช่วยลดขั้นตอนการเปรียบเทียบการหาอักขระย่อยเหมือนยาวที่สุดได้เป็นอย่างดี แต่ไม่สามารถใช้กับการหาลำดับเหมือนยาวที่สุดได้ ส่วนการใช้โครงสร้างซัฟฟิกซ์อะเรย์สามารถใช้หาลำดับเหมือนยาวที่สุดได้ โดยสามารถใช้ได้ดีกับชุดอักขระที่มีรูปแบบที่มีความแตกต่างกันมาก แต่ไม่เหมาะกับอักขระที่มีรูปแบบคล้ายกัน เช่น ดีเอ็นเอเป็นต้น

วิธีการหาแบบพิจารณาจากลำดับ และการหาแบบโครงสร้างซัฟฟิกซ์อะเรย์นั้น สามารถรองรับกับชุดข้อมูลได้ไม่จำกัด โดยสะดวกต่อการนำไปเปรียบเทียบกับชุดข้อมูลที่ต้องการเปรียบเทียบจำนวนมาก โดยแตกต่างจากการเปรียบเทียบแบบโปรแกรมพลวัต ที่ต้องการนำไปเปรียบเทียบกับข้อมูลที่มีจำนวนมากกว่า 2 ชุดจำเป็นต้องมีการพัฒนาเพิ่มเติมเข้าไป ทำให้ไม่สะดวกต่อการใช้งาน ดังนั้นการเลือกใช้วิธีการวิเคราะห์หาอักขระย่อยเหมือนยาวที่สุดและวิธีการหาลำดับเหมือนยาวที่สุดจึงขึ้นอยู่กับรูปแบบของข้อมูลที่ต้องการวิเคราะห์ และความต้องการในการวิเคราะห์ด้วย

5.2 ข้อเสนอแนะ

1. วิทยานิพนธ์นี้ได้ทำการทดลองแล้วพบว่ามีระหว่างการวิเคราะห์นั้นได้มีการแบ่งหน่วยไบแกรมออกมาเป็นจำนวนมากและจำเป็นจะต้องจัดเก็บเพื่อนำไปเปรียบเทียบทุกหน่วย ทำให้ใช้พื้นที่หน่วยความจำมาก จึงควรหาวิธีปรับลดพื้นที่ในการจัดเก็บข้อมูล เพื่อลดพื้นที่ใช้งานให้น้อยลง สำหรับการเปรียบเทียบแบบการเรียงลำดับ และโครงสร้างซัฟฟิกซ์อะเรย์

2. หลังจากการวิเคราะห์ทำการเปรียบเทียบแล้วนั้นผลลัพธ์ที่ได้เป็นจะอยู่ในรูปแบบของกลุ่มของอักขระที่คาดว่ายาวที่สุด แต่ยังไม่ใช่ผลลัพธ์ที่ถูกต้องที่สุด ซึ่งถ้ามีวิธีที่สามารถทำให้ได้ผลลัพธ์ที่ถูกต้องหลังจากทำการเปรียบเทียบแล้วจะทำให้การทำงานมีประสิทธิภาพมากยิ่งขึ้น

เอกสารอ้างอิง

[1] Cavnar, W. and Trenkle, J., **N-gram-based text categorization.** *In Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR).* (11-13 April 1994): pp.161-175. Las Vegas, USA.

[2] Dunning, T., **Statistical identification of language.** *In Technical report CRL MCCS-94-273,* Computing Research Lab, March. New Mexcio State University.

[3] McCreight, E. M. 1976. **A space–economical suffix tree construction algorithm.** *Journal of the ACM* 23:262–272.

[4] Mikio Yamamoto and Kenneth W. Church. 2001. **Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus.** *Computational Linguistics*, 27(1):1-30.

[5] Peng, F., Schuurmans D., Wang S. and Huang X., **Text classification in Asianlanguages without word segmentation.** *In Proceedings of The Sixth InternationalWorkshop on Information Retrieval with Asian Languages (IRAL 2003).*Association for Computational Linguistics, July 7. Sapporo, Japan

[6] Sibun Penelope and Reynar Jeffrey C., **Language identification: examining the issues.** *In 5th symposium on document analysis and information retrieval*, pp.125-135. Las Vegas, USA.

[7] Udi Manber and Gene Myers. **Suffix arrays: a new method for on-line string searches.** In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 1990.

[8] Ukkonen, E. 1995. **On-line construction of suffix trees.** *Algorithmica* 14:249–260.

[9] Weiner, P. 1973. **Linear pattern matching algorithms.** In *Conference Record,IEEE* 14th *Annual Symposium on Switching and Automata Theory*, 1–11.

[10] ทศนัย ชุ่มวัฒนะ. **"การสืบค้นรูปแบบสตริงย่อยแบบความยาวสูงสุดที่เกิดบ่อย."** วิทยานิพนธ์วิทยาศาสตรมหาบัณฑิต, บัณฑิตวิทยาลัย, ภาควิชาวิทยาการคอมพิวเตอร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.2549.

ประวัติผู้เขียน

ชื่อ – สกุล	นายบดินทร์ แสงอรุณ
วัน เดือน ปีเกิด	17 ตุลาคม 2521
ที่อยู่	1179 ซอยจรัญสนิทวงศ์ 13 ถนนจรัญสนิทวงศ์
	แขวงวัดท่าพระ เขตบางกอกใหญ่ กรุงเทพฯ 10600
ประวัติการศึกษา	2536 ประถมศึกษาปีที่5 โรงเรียนวัดบ้านส้อง
	2537 ประถมศึกษาปีที่6 โรงเรียนวัดวิจิตรการนิมิตร
	2539 มัธยมศึกษาตอนต้น สาขาวิชาศิลปะ
	โรงเรียนนวลนรดิศวิทยาคมรัชมังคลาภิเษก
	2541 ประกาศนียบัตรวิชาชีพ สาขาวิชาบัญชี
	โรงเรียนตั้งตรงจิตรพาณิชย์การ
	2542 ประกาศนียบัตรวิชาชีพชั้นสูง สาขาวิชาคอมพิวเตอร์ธุรกิจ
	วิทยาลัยอาชีวศึกษาธนบุรี
	2543 บริหารธุรกิจบัณฑิต สาขาวิชาคอมพิวเตอร์ธุรกิจ
	มหาวิทยาลัยสยาม
ประวัติการทำงาน	
พ.ศ. 2544 –ปัจจุบัน	ตำแหน่ง ผู้ช่วยนักวิจัย 1
	สังกัด หน่วยปฏิบัติการวิจัยคลังอนุพันธ์ความรู้ (KEA)
	ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC)