การเปรียบเทียบการใช้การวัดระยะการแก้ไขกับข้อความภาษาไทย

COMPARISON OF EDIT DISTANCE WITH THAI LANGUAGE

โดย

เมธาวี จินดารักษ์

METAWEE JINDARAK

อาจารย์ที่ปรึกษา

ดร. สุภกิจ นุตยะสกุล

รายงานนี้เป็นส่วนหนึ่งของวิชาการศึกษาอิสระ 2

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2556

COMPARISON OF EDIT DISTANCE WITH THAI LANGUAGE

METAWEE JINDARAK

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT OF THE COURSE

INDEPENDENT STUDY 2

MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2013

หัวข้อ	การการเปรียบเทียบการใช้การวัดระยะการแก้ไขกับข้อความภาษาไทย
นักศึกษา	นายเมธาวี จินดารักษ์
รหัสนักศึกษา	55660919
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	เทคโนโลยีสารสนเทศและการจัดการ
ปีการศึกษา	2556
อาจารย์ที่ปรึกษา	ดร. สุภกิจ นุตยะสกุล

บทคัดย่อ

งานวิจัยชิ้นนี้มีจุดประสงค์เพื่อศึกษาวิธีการในการวัดระยะการแก้ไข (Edit Distance) และพิสูจน์หาวิธีการที่เหมาะสมในการนำวิธีดังกล่าวมาประยุกต์ใช้กับภาษาไทย โดยได้ใช้วิธีการวัดระยะการแก้ไขมาประยุกต์ใช้ร่วมกับวิธีการตัดคำสำหรับภาษาไทย โดยงานวิจัยชิ้นนี้ได้นำ Levenshtein Distance ซึ่งเป็นวิธี Edit Distance ที่เป็นที่นิยม ไปทำการทดสอบเปรียบเทียบกับวิธี Needleman-Wunsch Algorithm และ Hamming Distance เพื่อเปรียบเทียบว่าวิธีการดังกล่าวเหมาะสำหรับการประยุกต์ใช้กับภาษาไทยหรือไม่ ซึ่งผลที่ได้จากการทดสอบนั้น ได้แสดงให้เห็นว่า Levenshtein Distance เป็นวิธีมีความเหมาะสมที่จะนำมาประยุกต์ใช้กับภาษาไทยเมื่อเทียบกับวิธีการอื่นๆ ที่นำมาเปรียบเทียบกัน

คำสำคัญ : การวัดระยะการแก้ไข, การเปรียบเทียบข้อความภาษาไทย, ระยะทางเลเวนชเตย์น

Title	Comparison of Edit Distance with Thai language
Student	Mr. Metawee Jindarak
Student ID.	55660919
Degree	Master of Science
Program	Information Technology
Major	Information Technology and Management
Year	2013
Advisor	Dr. Supakit Nootyaskool

ABSTRACT

This research objective is to prove about how to implement edit distance to suit for Thai language. The research experiment is using Thai word segmentation method combine with various edit distance method such as Levenshtein Distance, Needleman-Wunsch Algorithm and Hamming Distance to find out what is the most suitable edit distance method for using with Thai language. The result of the experiment shows that Levenshtein Distance is most suitable for implement with Thai language when comparing with other two edit distance methods.

Keyword : Edit Distance, Thai Sentence Comparison, Levenshtein Distance

กิตติกรรมประกาศ

รายงานการศึกษาอิสระเรื่องการประยุกต์ใช้การวัดระยะการแก้ไขกับข้อความภาษาไทย (Implementation of Edit Distance with Thai language) ฉบับนี้สำเร็จลงได้ ด้วยความอนุเคราะห์เป็นอย่างยิ่งจากบุคคลหลายท่านที่ได้กรุณาช่วยเหลือให้ข้อมูล คำปรึกษาและคำแนะนำ รวมทั้งกำลังใจ

ขอขอบพระคุณคณาจารย์ประจำคณะเทคโนโลยีสารสนเทศที่ได้ถ่ายทอดและให้ความรู้แก่ผู้ศึกษา โดยเฉพาะอย่างยิ่ง ดร.สุภกิจ นุตยะสกุล อาจารย์ที่ปรึกษาของการทำการศึกษาอิสระครั้งนี้ ที่ได้กรุณาให้คำแนะนำ และคำชี้แจงต่างๆ ในการทำงานวิจัยนี้จนสำเร็จลุล่วงเป็นรูปเล่มตามที่ปรากฏ

ขอขอบคุณ คุณเมรินทร์ สอิ้งทอง เพื่อนในรุ่นเดียวกัน ที่ให้คำแนะนำและแนวทางในการจัดรูปเล่มของรายงานการศึกษาอิสระฉบับนี้ และสุดท้ายนี้ขอขอบคุณทุกคนในครอบครัว และเพื่อนๆ ที่ให้กำลังใจมาตลอดการศึกษาในครั้งนี้

ผู้ศึกษาหวังว่ารายงานการวิจัยนี้จะเป็นประโยชน์แก่ผู้ที่เกี่ยวข้องและผู้ที่สนใจบ้างไม่มากก็น้อย หากมีข้อผิดพลาดประการใด ผู้ศึกษาขออภัยมา ณ ที่นี้ด้วย

เมธาวี จินดารักษ์

สารบัญ

หน้า

บทคัดย่อภาษาไทย..........I

บทคัดย่อภาษาอังกฤษ..........II

กิตติกรรมประกาศ..........III

สารบัญ..........IV

สารบัญตาราง..........VI

สารบัญภาพ..........VII

บทที่ 1 บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา..........1

1.2 วัตถุประสงค์ของการศึกษา..........2

1.3 ขอบเขตของการศึกษา..........2

1.4 ขั้นตอนการศึกษา..........2

1.5 ประโยชน์ที่คาดว่าจะได้รับ..........3

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

2.1 TIS-620..........4

2.2 Unicode..........5

2.3 การวัดระยะการแก้ไขข้อความ (Edit Distance)..........5

2.4 ระยะทางเลเวนชเตย์น (Levenshtein Distance)..........5

2.5 ระยะทางแฮมมิง (Hamming Distance)..........9

2.6 วิธีของนีเดอมาน-วานซ์ (Needleman-Wunsch Algorithm)..........10

2.7 การตัดคำภาษาไทย..........13

2.8 งานวิจัยที่เกี่ยวข้อง..........14

บทที่ 3 วิธีดำเนินการวิจัย

3.1 ปัญหาของการใช้ Edit Distance กับภาษาไทย..........16

3.2 การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย..........16

3.3 วิธีการที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความมาษาไทย..........19

สารบัญ(ต่อ)

หน้า

บทที่ 4 ผลการศึกษาวิจัย

4.1 เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ..........23

4.2 รูปแบบของการทดสอบการวัดระยะการแก้ไข..........23

4.3 การทดสอบการวัดระยะการแก้ไขข้อความกับ Character Set รูปแบบต่างๆ..........40

บทที่ 5 สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการวิจัย..........43

5.2 อภิปรายผล..........45

5.3 ข้อเสนอแนะ..........46

บรรณานุกรม..........47

ภาคผนวก..........49

ภาคผนวก ก..........50

ภาคผนวก ข..........57

ประวัติผู้เขียน..........59

สารบัญตาราง

ตารางที่ หน้า

4.1 สเปคของเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบการวัดระยะการแก้ไข......................23

4.2 ชุดข้อมูลตัวอย่างที่ใช้ในการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลี ภาษาไทย.....24

4.3 ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย ชนิดไม่ตัดแบ่งคำก่อนการวัดระยะ.....24

4.4 ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย ชนิดตัดแบ่งคำก่อนการวัดระยะ.....25

4.5 ชุดข้อมูลตัวอย่างที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย ความยาวสั้น.....27

4.6 ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย ความยาวสั้น.....28

4.7 ชุดข้อมูลตัวอย่างที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ ภาษาไทย.....34

4.8 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูล ตัวอย่างชุดที่ 1.....35

4.9 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูล ตัวอย่างชุดที่ 2.....36

4.10 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูล ตัวอย่างชุดที่ 3.....37

4.11 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูล ตัวอย่างชุดที่ 4.....38

4.12 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูล ตัวอย่างชุดที่ 5.....39

4.13 ผลการแสดงผลและแบ่งคำของข้อความภาษาอังกฤษกับ Character Set แบบต่างๆ.......41

4.14 ผลการแสดงผลและแบ่งคำของข้อความภาษาไทยกับ Character Set แบบต่างๆ..............42

สารบัญภาพ

ภาพที่ หน้า

2.1 ภาพตารางแสดงอักขระในรหัสอักขระ TIS-620..4

2.2 ขั้นตอนการแก้ไขข้อความจากคำว่า "ALGORITHM" เป็น "METHOD"......................6

2.3 ตัวอย่างรหัสเทียมของ Levenshtein Distance...8

2.4 ตัวอย่างรหัสเทียมของ Hamming Distance..10

2.5 ตัวอย่างรหัสเทียมของขั้นตอนในการหาขนาดของ Matrix.......................................11

2.6 ตัวอย่างรหัสเทียมของขั้นตอนการนำคะแนนไปเติมลงใน Matrix............................12

3.1 หน้าจอโปรแกรมสำหรับทดสอบ Edit Distance ในส่วนการกรอกค่าที่จะเปรียบเทียบ....17

3.2 กระบวนการในการวัดระยะแก้ไขข้อความภาษาไทย..18

3.3 Source Code ภาษา VB สำหรับคำนวณหาค่า Levenstein Distance..........................19

3.4 Source Code ภาษา VB สำหรับคำนวณหาค่า Hamming Distance............................20

3.5 Source Code ภาษา VB สำหรับคำนวณหาค่า Needleman-Wunsch Algolithm.............21

3.6 หน้าจอโปรแกรมสำหรับทดสอบ Edit Distance ในส่วนการแสดงผลการคำนวณ.........22

4.1 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไข แบบคำหรือวลีภาษาไทย...25

4.2 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไข แบบคำหรือวลีภาษาไทย ชนิดไม่ตัดแบ่งคำก่อนการวัดระยะ.................................26

4.3 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไข แบบคำหรือวลีภาษาไทย ชนิดตัดแบ่งคำก่อนการวัดระยะ.....................................26

4.4 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไข กับข้อความภาษาไทยความยาวสั้น..28

4.5 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไข กับข้อความภาษาไทยความยาวสั้น..29

4.6 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไข กับข้อความภาษาไทยความยาวปานกลาง..30

4.7 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไข กับข้อความภาษาไทยความยาวปานกลาง..31

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

4.7 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง.....31

4.8 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก.....32

4.9 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก.....33

5.1 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขแบบไม่ตัดแบ่งคำ.....44

5.2 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขแบบไม่ตัดแบ่งคำ.....44

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในการเปรียบเทียบความแตกต่างของข้อความ และรูปภาพหรือรูปทรงต่างๆ ในทางคอมพิวเตอร์นั้น มีวิธีการที่นำมาใช้ในการเปรียบเทียบได้หลายวิธี ซึ่งวิธีการหนึ่งที่ถูกนำมาใช้ คือ การวัดระยะการแก้ไข โดยการวัดระยะการแก้ไขนั้น ก็มีผู้คิดค้นอัลกอริทึมที่แตกต่างกันออกไปหลายวิธี โดยแต่ละวิธีจะมีวิธีการในการวัดระยะ และข้อจำกัดในการวัดระยะที่แตกต่างกัน เช่น การวัดระยะด้วยวิธี Hamming Distance จะเป็นการวัดระยะการแก้ไขข้อความที่มีจำนวนตัวอักษรเท่ากัน และเปรียบเทียบเฉพาะตัวอักษรที่ตำแหน่งที่ตรงกันเท่านั้น ซึ่งวิธีนี้ใช้ในการทำ Hamming Code สำหรับแก้ไขข้อมูลที่ผิดพลาด (Error Correction) จากการส่งข้อมูลต้นทางไปยังปลายทางในระบบเครือข่าย[7] เป็นต้น แต่ถ้าจะกล่าวถึงวิธีการวัดระยะการแก้ไขที่เป็นที่นิยม และรู้จักแพร่หลาย คือ Levenshtein Distance ซึ่งเป็นวิธีการที่ใช้ในการวัดจำนวนครั้งของการแก้ไขข้อความจากคำหนึ่งไปเป็นอีกคำหนึ่ง ด้วยการนับจำนวนครั้งของการแทรก การตัดออก และการแทนที่ตัวอักษร ในข้อความนั้นๆ ว่าใช้จำนวนครั้งเท่าใดในการแก้ไขข้อความ

วิธีการวัดระยะการแก้ไขนั้นได้ถูกนำไปใช้ในการเปรียบเทียบหาความแตกต่างทั้งรูปภาพ และข้อความ ซึ่งในต่างประเทศได้มีงานวิจัยหลายชิ้น ได้นำทฤษฎีดังกล่าวมาประยุกต์ใช้ ไม่ว่าจะเป็นการเปรียบเทียบรูปภาพที่เป็นภาพนิ่งหรือรูปทรงต่างๆ การเปรียบเทียบวิดีโอ และที่พบมากที่สุด คือ ใช้ในการเปรียบเทียบข้อความ ซึ่งในการเปรียบเทียบข้อความนี้เอง ก็ได้ถูกนำไปประยุกต์ใช้กับการใช้ในชีวิตประจำวันมากมาย เช่น โปรแกรมสำหรับทำงานด้านเอกสารที่มีการตรวจสอบคำผิด หรือเว็บไซต์ค้นหาข้อมูล (Search Engine) ที่มีการวิเคราะห์คำที่ใช้ในการค้นหา เพื่อหาคำที่ใกล้เคียงกันกับคำที่ค้นหา ด้วยเหตุนี้เองจึงควรที่จะมีการศึกษาเพื่อหาแนวทางในการนำวิธีการดังกล่าวมาประยุกต์ใช้กับภาษาไทย เพื่อจะได้เป็นแนวทางในการนำไปต่อยอด เพื่อให้เกิดประโยชน์ในอนาคตต่อไป

1.2 วัตถุประสงค์ของการจัดทำโครงการ

- เพื่อศึกษาทฤษฎีการวัดระยะการแก้ไข (Edit Distance) และรูปแบบของการนำไปใช้งาน
- เพื่อศึกษาและวิเคราะห์เพื่อหาแนวทางที่เป็นไปได้ในการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทย
- เพื่อทดสอบประสิทธิภาพของการนำทฤษฎีการวัดระยะการแก้ไขหลังจากที่นำมาประยุกต์ใช้กับข้อความภาษาไทย

1.3 ขอบเขตของโครงการ

- ศึกษาทฤษฎีการวัดระยะการแก้ไข (Edit Distance) ตามวิธีการของ Levenshtein Distance
- ศึกษาแนวทางในการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทยด้วยวิธีของ Levenshtein Distance
- ทดสอบประสิทธิภาพในการใช้ Levenshtein Distance ในการวัดระยะการแก้ไขข้อความภาษาไทย เปรียบเทียบกับวิธีการอื่นที่ใกล้เคียงกัน

1.4 ขั้นตอนการดำเนินโครงการ

- ศึกษาและรวบรวมข้อมูลที่เกี่ยวข้องกับทฤษฎีการวัดระยะการแก้ไข (Edit Distance)
- ศึกษาค้นคว้าหาแนวทางที่เป็นไปได้ในการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทย
- ออกแบบวิธีการในการทดสอบการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทย
- ทดสอบประสิทธิภาพของการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทย
- สรุปผลการทดสอบประสิทธิภาพของการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทย

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- ทราบถึงลักษณะและวิธีการของทฤษฎีการวัดระยะการแก้ไข
- ทราบถึงแนวทางที่เป็นไปได้ในการประยุกต์ใช้ทฤษฎีการวัดระยะการแก้ไขกับข้อความภาษาไทย
- ทราบถึงประสิทธิภาพของการนำทฤษฎีการวัดระยะการแก้ไขมาประยุกต์ใช้กับข้อความภาษาไทย เพื่อหาวิธีการที่เหมาะสมในการประยุกต์ใช้กับภาษาไทย

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

สำหรับในบทนี้จะกล่าวถึงทฤษฎีและวิธีการต่างๆ ที่นำมาศึกษาและใช้ในการทดสอบการวัดระยะการแก้ไข เพื่อหาวิธีการที่เหมาะสมในการนำการวัดระยะการแก้ไขมาประยุกต์ใช้กับข้อความภาษาไทย

2.1 TIS-620

มาตรฐานผลิตภัณฑ์อุตสาหกรรม 620-2533 (มอก.620-2533) หรือที่รู้จักกันในชื่อ TIS-620 เป็นชุดอักขระมาตรฐานอุตสาหกรรมของไทย โดยมีชื่อเต็มว่า รหัสสำหรับอักขระไทยที่ใช้กับคอมพิวเตอร์ โดยที่รหัส TIS-620 มีรายละเอียดคล้ายกับรหัส ISO-8859-11 มาก แตกต่างกันแค่เพียงที่ ISO-8859-11 กำหนดให้ A0 เป็น "เว้นวรรคแบบไม่ตัดคำ" (no-break space) แต่ในส่วนของ TIS-620 นั้นจะสงวนตำแหน่ง A0 เอาไว้ แต่ก็ไม่ได้กำหนดค่าใด ๆ ให้ (สำนักงานมาตรฐานผลิตภัณฑ์อุตสาหกรรม, ม.ป.ป.)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	ไม่ได้ใช้															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	ไม่ได้ใช้															
9x																
Ax		ก	ข	ฃ	ค	ฅ	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ญ	ฎ	ฏ
Bx	ฐ	ฑ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ
Cx	ภ	ม	ย	ร	ฤ	ล	ฦ	ว	ศ	ษ	ส	ห	ฬ	อ	ฮ	ฯ
Dx	ะ	◌ั	า	◌ำ	◌ิ	◌ี	◌ึ	◌ื	◌ุ	◌ู	◌ฺ					฿
Ex	เ	แ	โ	ใ	ไ	ๅ	ๆ	◌็	◌่	◌้	◌๊	◌๋	◌์	◌ํ	◌๎	๏
Fx	0	1	2	3	4	5	6	7	8	9	๚	๛				

ภาพที่ 2.1 ภาพตารางแสดงอักขระในรหัสอักขระ TIS-620

2.2 Unicode

มาตรฐานอุตสาหกรรมที่ช่วยให้คอมพิวเตอร์แสดงผลและจัดการข้อความที่ใช้ในระบบการเขียนของภาษาส่วนใหญ่ในโลกให้สามารถสดงผลได้อย่างสอดคล้องกัน โดย Unicode ประกอบด้วยรายการอักขระที่แสดงผลได้มากกว่า 100,000 ตัว ซึ่งพัฒนาต่อยอดมาจากมาตรฐานชุดอักขระสากล (Universal Character Set: UCS) และมีการตีพิมพ์ลงในหนังสือ The Unicode Standard เพื่อจัดทำเป็นแผนผังรหัสสำหรับใช้เป็นรายการอ้างอิง นอกจากนั้นยังมีการอธิบายวิธีการที่ใช้เข้ารหัสและการนำเสนอมาตรฐานของการเข้ารหัสอักขระอีกจำนวนหนึ่ง (The Unicode Consortium, n.p.)

2.3 การวัดระยะการแก้ไข (Edit Distance)

การวัดระยะการแก้ไข (Edit Distance) เป็นวิธีการในการวัดหาค่าความต่างกันระหว่างชุดตัวอักษรสองชุด ระหว่างชุดแรกที่เป็นชุดต้นแบบ และชุดที่สองที่เป็นชุดเปรียบเทียบ โดยค่าความแตกต่างจะวัดจากจำนวนครั้งของการที่จะต้องทำการ แทรก ตัดออก และแทนที่ตัวอักษรในชุดตัวอักษรที่นำมาเปรียบเทียบจนกระทั่งชุดตัวอักษรที่นำมาเปรียบเทียบ (ชุดที่สอง) มีลักษณะเหมือนกับตัวอักษรชุดต้นแบบ (ชุดแรก) ทุกประการ ซึ่งโดยทั่วไปการวัดระยะการแก้ไขจะอ้างอิงถึงระยะทางเลเวนชเตย์น (Levenshtein Distance) ซึ่งเป็นทฤษฎีที่เกี่ยวข้องกับการวัดระยะการแก้ไขที่ถูกตั้งชื่อตาม Vladimir Levenshtein ผู้ที่ปรับปรุงชุดอัลกอริทึมชุดนี้ (Jurafsky, n.p.)

2.4 ระยะทางเลเวนชเตย์น (Levenshtein Distance)

2.4.1 วิธีการของ Levenshtein Distance

ขั้นตอนวิธีการ Levenshtein Distance จะเป็นการนำชุดตัวอักษร 2 ชุด มาเปรียบเทียบจำนวนความแตกต่างกัน โดยจะพิจารณาจาก 3 รูปแบบ คือ

- การแทรก : เป็นการนำเอาตัวอักษรตัวใดๆ มา เพื่อให้ชุดตัวอักษรชุดนั้นเหมือนกับอีกชุดตัวอักษรหนึ่งในภายหลัง เช่น MAN เป็น MAIN จะแทรกตัว I ให้กับ MAN เพื่อให้ MAN กลายเป็น MAIN เป็นต้น
- การตัดออก : เป็นการตัดตัวอักษรออกครั้งละ 1 ตัว จากชุดตัวอักษรชุดหนึ่ง เพื่อให้ชุดตัวอักษรชุดนั้นเหมือนกับอีกชุดตัวอักษรหนึ่งในภายหลัง เช่น GOD เป็น GO จะตัดตัว D ออก เพื่อให้ GOD กลายเป็น GO เป็นต้น
- การแทนที่ : เป็นการนำตัวอักษรของชุดตัวอักษรหนึ่งไปแทนตัวอักษรของอีกชุดตัวอักษรหนึ่ง เพื่อให้ชุดตัวอักษรชุดนั้นเหมือนกับอีกชุดตัวอักษรหนึ่งในภายหลัง เช่น

CHAT เป็น THAT จะแทนที่ C ด้วย T เพื่อให้ CHAT กลายเป็น THAT หรือ อาจมองในทางกลับกันก็ได้ เป็นต้น (Levenshtein VI, 1966)

2.4.2 ตัวอย่างของ Levenshtein Distance

		M	E	T	H	O	D
	0	1	2	3	4	5	6
A	1	1	2	3	4	5	6
L	2	2	2	3	4	6	6
G	3	3	3	3	4	5	6
O	4	4	4	4	4	4	5
R	5	5	5	5	5	5	5
I	6	6	6	6	6	6	6
T	7	7	7	6	7	7	7
H	8	8	8	7	6	7	8
M	9	8	9	8	7	7	8

ภาพที่ 2.2 ขั้นตอนการแก้ไขข้อความจากคำว่า "ALGORITHM" เป็น "METHOD"

จากตัวอย่างจะเห็นว่า คำว่า algorithm และ คำว่า method มีค่าความต่างกันอยู่คือ 8 โดยได้มาจาก

- การแทรก a, l, g และ o ดำเนินการ 4 ครั้ง
- การตัด o ของ method ออก ดำเนินการ 1 ครั้ง
- การแทนที่ m ด้วย r, e ด้วย i และ d ด้วย m ดำเนินการ 3 ครั้ง

ดังนั้น จะต้องดำเนินการทั้งหมด 8 ครั้ง จึงมีค่าความต่างกันอยู่ 8

2.4.3 การพิสูจน์ความถูกต้อง

เราสามารถแปลงชุดอักขระเริ่มต้น s[1..i] เป็นชุดอักขระเปรียบเทียบ t[1..j] โดยใช้จำนวนครั้งในการกระทำคือ d[i,j] ซึ่งเป็นจำนวนครั้งของการกระทำที่น้อยที่สุด สามารถแสดงการพิสูจน์ความถูกต้องได้ดังนี้ (Gusfield, 1997)

- เราจะต้องเริ่มจากแถว และ สดมภ์ ที่ 0 เพราะว่าชุดอักขระ s[1..i] สามารถแปลงเป็นชุดอักขระว่างเปล่า t[1..0] ได้ โดยการตัดอักขระของชุดอักขระ s[1..i] ออกทั้งหมด i ตัว

และในทำนองเดียวกัน เราก็สามารถแปลงจากชุดอักขระว่างเปล่า s[1..0] เป็นชุดอักขระ t[1..j] ได้ โดยเพิ่มอักขระจำนวน j ตัว

- ถ้า s[i] = t[j] และเราสามารถแปลงชุดอักขระ s[1..i-1] เป็นชุดอักขระ t[1..j-1] ใน k ครั้งแล้ว เราก็สามารถทำในทำนองเดียวกันกับ s[1..i] ได้ใน k ครั้ง โดยสามารถละเลยอักขระตัวสุดท้ายไปได้
- มิฉะนั้นแล้ว การแปลงชุดอักขระโดยมีค่าการกระทำน้อยที่สุดสามารถดำเนินการได้ด้วยสามวิธีดังนี้
    - ถ้าเราสามารถแปลงชุดอักขระ s[1..i] เป็นชุดอักขระ t[1..j-1] ใน k ครั้งแล้ว เราก็สามารถเพิ่มอักขระ t[j] ได้ เพื่อที่จะให้ได้ข้อความ t[1..j] โดยใช้การกระทำ k+1 ครั้ง (การแทรก)
    - ถ้าเราสามารถแปลงชุดอักขระ s[1..i-1] เป็นชุดอักขระt[1..j] ใน k ครั้งแล้ว เราก็สามารถลด s[i] ออก แล้วกระทำการแปลงข้อความตามเดิม จำนวนครั้งจะเท่ากับ k+1 ครั้ง (การตัดออก)
    - ถ้าเราสามารถแปลงชุดอักขระ s[1..i-1] เป็นชุดอักขระ t[1..j-1] ใน k ครั้ง เราก็สามารถทำในทำนองเดียวกันกับชุดอักขระ s[1..i] และเปลี่ยนอักขระต้นแบบ s[i] ไปเป็นอักขระ t[j] ได้ จำนวนครั้งจะเท่ากับ k+1 ครั้ง (การแทนที่)
- จำนวนการกระทำที่ต้องการแปลงจากชุดอักขระ s[1..n] เป็นชุดอักขระ t[1..m] คือจำนวนครั้งที่ใช้ในการแปลงอักขระทั้งหมดใน s ไปเป็นอักขระทั้งหมดใน t ซึ่งจะปรากฏในตำแหน่ง d[n, m]

ในการพิสูจน์ความถูกต้องนั้น สามารถเขียนออกมาเป็นสมการที่ใช้ในการพิสูจน์เพื่อหาค่าความถูกต้องของ Levenshtein Distance ได้ดังนี้

$$D(i,j) = min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 1; \ if \ s_1(i) \neq s_2(j) \\ 0; \ if \ s_1(i) = s_2(j) \end{cases} \end{cases} \tag{2.1}$$

2.4.4 รหัสเทียมของ Levenshtein Distance

```
int LevenshteinDistance(char s[1..m], char t[1..n])
{
  // สำหรับทุกๆค่า i และ j, d[i,j] จะแสดงค่าความแตกต่างระหว่างอักขระ i ตัวแรกของ s และ อักขระ j ตัว
  // แรกของ t   สังเกตว่า แถวลำดับ d จะมีขนาด (m+1)x(n+1)
  declare int d[0..m, 0..n]
  for i from 0 to m
    d[i, 0] := I          // ค่าความแตกต่างระหว่างข้อความแรกใดๆ กับ ข้อความที่สองที่ว่างเปล่า
    for j from 0 to n
      d[0, j] := j        // ค่าความแตกต่างระหว่างข้อความที่สองใดๆ กับ ข้อความแรกที่ว่างเปล่า
    for j from 1 to n
    {
      for i from 1 to m
      {
        if s[i] = t[j] then
          d[i, j] := d[i-1, j-1]        // พิจารณาตัวถัดมา
        else
          d[i, j] := minimum
          (
            d[i-1, j] + 1,              // การตัดออก
            d[i, j-1] + 1,              // การแทรก
            d[i-1, j-1] + 1             // การแทนที่
          )
      }
    }
    return d[m,n]
}
```

ภาพที่ 2.3 ตัวอย่างรหัสเทียมของ Levenshtein Distance

2.5 ระยะทางแฮมมิง (Hamming Distance)

ระยะทางแฮมมิง ตั้งชื่อตาม ริชาร์ด แฮมมิง (Richard Hamming) ผู้นำเสนอให้ใช้ระยะทางแฮมมิง (Hamming Distance) ในการตรวจสอบคำ โดย Hamming Distance ได้นำมาใช้ในการสื่อสาร โดยการนับจำนวนเลขฐานสองที่มีความยาวคงที่ ที่มีหลักต่างไปเพื่อตรวจสอบความผิดพลาดในการสื่อสาร ซึ่งเรียกว่า ระยะทางสัญญาณ (Signal distance) โดยการสื่อสารในระบบดิจิตอลโดยส่วนมากมีโอกาสที่ข้อมูลที่ส่งผ่านระบบสื่อสารผิดพลาดจึงมีการนำ Error Control Coding มาใช้ในการสื่อสาร เพื่อลดความผิดพลาดของการสื่อสาร

Error Control Coding คือการเข้ารหัสสัญญาณ และการถอดรหัสสัญญาณ ซึ่งวิธีการในการลดความผิดพลาดของการสื่อสารจะทำโดยการตรวจจับความผิดพลาด (Error Detection) และแก้ไขข้อมูลความผิดพลาด (Error Correction) นอกจากนี้ Hamming Distance ยังถูกใช้ในการวิเคราะห์ได้หลายอย่างโดยการนำทฤษฎีมาต่อยอด เช่น ด้านการออกแบบฮาร์ดแวร์คอมพิวเตอร์ จะใช้ Hamming Distance ในการลดเวลาในการเปลี่ยนขั้นการทำงานของฮาร์ดแวร์และซอฟต์แวร์ หรือระบบเอทีเอ็ม ซึ่งข้อมูลที่ทำการรับส่งในการสื่อสารจะต้องไม่มีการผิดพลาด จึงมีการนำ Hamming Distance มาใช้ในการตรวจสอบว่าข้อมูลที่ได้รับผิดพลาดหรือไม่ และหากข้อมูลนั้นผิดพลาดสามารถแก้ไขได้หรือไม่ เป็นต้น (Department of Electrical and Computer Engineering, n.p.)

2.5.1 ขั้นตอนและวิธีการของ Hamming Distance

Hamming Distance มีขั้นตอนและวิธีการในการตรวจสอบข้อมูล แบ่งออกได้เป็น 3 ขั้นตอน ดังนี้

- ตรวจสอบข้อความที่จะทำการเปรียบเทียบว่ามีขนาดเท่ากัน
- นำอักษร ณ ตำแหน่งเดียวกันมาเปรียบเทียบว่ามีตำแหน่งไหนที่ไม่เป็นตัวเดียวกันบ้าง
- จำนวนตัวอักษรที่ไม่เป็นตัวเดียวกันที่นับได้นั้น คือ Hamming Distance

2.5.2 รหัสเทียมของ Hamming Distance

```
HammingDistance(A,B) {
  for (ทุกตัวอักษร) {
    if (ตัวอักษรของ A ไม่เท่ากับ ตัวอักษรของ B) {
      นับค่า
    }
  }
  return ตัวนับ
}
```

ภาพที่ 2.4 ตัวอย่างรหัสเทียมของ Hamming Distance

2.6 วิธีของนีเดอมาน-วานซ์ (Needleman-Wunsch Algorithm)

วิธีของนีเดอมาน-วานซ์ คือ การหาลำดับที่ดีที่สุดในการจัดเรียง เพื่อให้ลำดับของชุดข้อมูล A และ B ตรงกันในทุกตำแหน่งให้ได้มากที่สุดเท่าที่จะเป็นไปได้ ซึ่งมีการใช้กันทั่วไปในชีวสารสนเทศศาสตร์ เพื่อเรียงลำดับของ โปรตีน หรือ นิวคลีโอไทด์ โดยที่วิธีนี้ถูกตีพิมพ์ครั้งแรกในปี 1970 โดย Saul B. Needleman และ Christian D. Wunsch ซึ่งวิธีของนีเดอมาน-วานซ์ นั้นเป็นหนึ่งในตัวอย่างของการเขียนโปรแกรมโดยใช้เทคนิคกำหนดการพลวัต และเป็นการใช้กำหนดการพลวัตครั้งแรกในการเปรียบเทียบลำดับชีวภาพ (Needleman, 1969)

วิธีการของนีเดอมาน-วานซ์ มี 2 ขั้นตอน คือ

- ขั้นตอนในการหาขนาดของ Matrix
- ขั้นตอนการนำคะแนนไปเติมลงใน Matrix

2.6.1 ขั้นตอนในการหาขนาดของ Matrix

โดยในการหาการเรียงตัวที่มีคะแนนสูงสุด ทำได้โดยกำหนด F เป็น Matrix โดยมีแถว i และคอลัมน์ j เป็น F_{ij} จะมีหนึ่งคอลัมน์สำหรับแต่ละอักขระใน A และหนึ่งแถวสำหรับแต่ละอักขระใน B ดังนั้นหากเราต้องการทำเรียงลำดับที่มีขนาด n และ m จำเป็นต้องใช้ Matrix ที่มีขนาด O(nm)

2.6.2 รหัสเทียมของขั้นตอนในการหาขนาดของ Matrix

```
for i=0 to length (A)
    F (i,0) ← d*i
  for j=0 to length (B)
    F (0,j) ← d*j
  for i=1 to length (A)
    for j = 1 to length (B)
    {
      Match ← F (i-1,j-1) + S (Ai, Bj)
      Delete ← F (i-1, j) + d
      Insert ← F (i, j-1) + d
      F (i,j) ← max (Match, Insert, Delete)
    }
```

ภาพที่ 2.5 ตัวอย่างรหัสเทียมของขั้นตอนในการหาขนาดของ Matrix

2.6.3 ขั้นตอนการนำคะแนนไปเติมลงใน Matrix

เมื่อเราหาเมตริกซ์ F ได้แล้ว F_{nm} จะเป็นคะแนนสูงที่สุดของการเรียงที่เป็นไปได้ ในการเติมเมตริกซ์ F นี้ ทำได้โดยเริ่มจากการเติมช่องล่างขวา และทำการเปรียบเทียบระหว่าง 3 กรณีที่เป็นไปได้หาว่ากรณีไหนได้คะแนนสูงสุด (กรณีเท่ากัน, แทรก, ลบ) ถ้า “เท่ากัน” นั่นคือ A_i และ B_j นั้นตรงกัน ถ้า “ลบ” หมายความว่า A_i นั้นตรงกับช่องว่าง และถ้า “แทรก” หมายความว่า B_j นั้นตรงกับช่องว่าง (การเติมเมตริกซ์ F โดยทั่วไปนั้น อาจมีช่องที่มีคะแนนเท่ากันได้หลายช่อง นั่นคือมีทางเลือกที่ดีที่สุดได้หลายทาง)

2.6.4 รหัสเทียมของขั้นตอนการนำคะแนนไปเติมลงใน Matrix

```
AlignmentA ← ""
AlignmentB ← ""
i ← length (A)
j ← length (B)
while (i > 0 and j > 0){
  Score ← F (i,j)
  ScoreDiag ← F (i - 1, j - 1)
  ScoreUp ← F (i, j - 1)
  ScoreLeft ← F (i - 1, j)
  if (Score == ScoreDiag + S (Ai, Bj)){
    AlignmentA ← Ai + AlignmentA
    AlignmentB ← Bj + AlignmentB
    i ← i - 1
    j ← j - 1
  }
  else if (Score == ScoreLeft + d){
    AlignmentA ← Ai + AlignmentA
    AlignmentB ← "-" + AlignmentB
    i ← i - 1
  }
  otherwise (Score == ScoreUp + d){
    AlignmentA ← "-" + AlignmentA
    AlignmentB ← Bj + AlignmentB
    j ← j - 1
  }
}
while (i > 0){
  AlignmentA ← Ai + AlignmentA
  AlignmentB ← "-" + AlignmentB
  i ← i - 1
}
while (j > 0){
  AlignmentA ← "-" + AlignmentA
  AlignmentB ← Bj + AlignmentB
  j ← j - 1
}
```

ภาพที่ 2.6 ตัวอย่างรหัสเทียมของขั้นตอนการนำคะแนนไปเติมลงใน Matrix

2.7 การตัดคำภาษาไทย

เนื่องด้วยภาษาไทยนั้นจะมีลักษณะของข้อความที่มีตัวอักษรเขียนติดกันทั้งหมด การจะแบ่งว่าในข้อความใดข้อความหนึ่งประกอบด้วยคำใดบ้างจึงทำได้ยาก ทำให้ต้องมีการคิดวิธีต่างๆ ขึ้นมาเพื่อที่จะใช้ตัดคำสำหรับภาษาไทย ซึ่งมีด้วยกันหลายวิธีดังนี้

การตัดคำแบบยาวที่สุด (Longest Matching) : การตัดคำวิธีนี้ทำโดยเริ่มจากตัวอักษรซ้ายสุดของข้อความไปยังตัวอักษรถัดไป จนกว่าจะพบคำที่มีอยู่ในพจนานุกรม หลังจากนั้นก็ค้นหาคำถัดไปจนกว่าจะจบข้อความ หากพบว่าเป็นคำในพจนานุกรมจากจุดเริ่มต้นเดียวกัน จะเลือกเอาคำที่ยาวที่สุด เช่น การแบ่งคำในประโยค "คนตากลมนอนตากลม" จะเริ่มจากตัวอักษร ค แล้วค้นหาตัวอักษรถัดไปและนำมาเปรียบเทียบคำในพจนานุกรม โดยคำแรกที่แบ่งได้คือ "คน" และตัวอักษรถัดไปคือ ต จากตัวอักษรนี้จะได้คำว่า "ตา" กับคำว่า "ตาก" ซึ่งวิธีนี้ให้เลือกคำที่ยาวที่สุดที่ค้นพบจึงเลือกคำว่า "ตาก" จากนั้นก็จะค้นหาและเปรียบเทียบต่อไป ซึ่งผลลัพธ์ที่ออกมาคือ "คน ตาก ลม นอน ตาก ลม"

การตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching) : เป็นการหาวิธีในการตัดคำที่เป็นไปได้ทั้งหมด เช่น ข้อความว่า "ไปหามเหสี" ซึ่งสามารถตัดคำได้ 2 แบบ คือ “ไป หาม เห สี” และ “ไป หา มเหสี” โดยที่วิธีการนี้จะเลือกเอาข้อความที่แบ่งแล้วได้จำนวนคำน้อยที่สุด คือ “ไป หา มเหสี” ซึ่งมีจำนวนคำที่ตัดได้ 3 คำ ในขณะที่ “ไป หาม เห สี” มี 4 คำ

การตัดคำแบบคำนวณเชิงสถิติเพื่อหาความเป็นไปได้ (Probabilistic Model) : วิธีนี้นำเอาค่าสถิติการเกิดของคำและลำดับของหน้าที่ของคำ (Part of speech) มาช่วยในการคำนวณหาความน่าจะเป็น เพื่อใช้เลือกแบบที่มีโอกาสการเกิดมากที่สุด แต่วิธีนี้มีข้อจำกัด คือ ต้องมีฐานข้อมูลที่มีการตัดคำที่ถูกต้อง และต้องกำหนดหน้าที่ของคำ เพื่อที่จะได้นำไปใช้ในการสร้างสถิติ

การตัดคำแบบใช้คุณลักษณะ (Feature-Based Approach) : วิธีการนี้จะพิจารณาจากบริบทของคำ (Context words) และการเกิดร่วมกันหรือหน้าที่ของคำ (Collocation) เข้ามาช่วยในการตัดคำ เช่น "ตากลม" ถ้าพบคำว่า "โต" ในบริบทก็จะสามารถตัดคำได้ว่า "ตา กลม" ซึ่งวิธีนี้จำเป็นที่จะต้องมีฐานข้อมูลเป็นจำนวนมาก และจะต้องมีการเรียนรู้การสร้างคำในบริบทหรือการเกิดร่วมกันของคำแต่ละคำ เพื่อให้มีข้อมูลที่จะนำมาใช้ในการตัดคำ (โครงการสารานุกรมไทยฯ, ม.ป.ป.)

2.8 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการใช้ทฤษฎี Edit Distance

- Bing Lui และ Yanhong Zhai นำเสนอวิธีการคัดแยกข้อมูลจากหน้าเว็บเพจ โดยการคัดแยกข้อมูลที่มีลักษณะเป็น Data Record ที่เก็บไว้บนฐานข้อมูลของเว็บไซต์ ออกมาจากข้อมูลส่วนอื่นๆ โดยใช้ Edit Distance Tree และ Visual Cue ซึ่งผลที่ได้จากการคัดแยกข้อมูลจากหน้าเว็บเพจด้วยวิธีดังกล่าวได้ผลลัพธ์ที่มีความถูกต้องสูง (Lui, 2005)

- Davi De Castro Reis และคณะ นำเสนอการค้นหาและคัดแยกข้อมูลข่าวจากหน้าเว็บไซต์ข่าว โดยใช้การวิเคราะห์โครงสร้างข้อมูลของเว็บไซต์แต่ละหน้า แล้วนำข้อมูลที่มีลักษณะโครงสร้างใกล้เคียงกันมาทำการจัดกลุ่ม แล้วใช้ Edit Distance Tree ร่วมกับการใช้อัลกอลิทึม RTDM (Restricted Top-Down Mapping) ในการเปรียบเทียบหน้าเว็บที่มีลักษณะโครงสร้างใกล้เคียงกัน เพื่อใช้ในการตัดส่วนที่ไม่ใช่เนื้อหาข่าวออก และดึงเฉพาะข้อมูลที่เป็นเนื้อหาของข่าวมาแสดงผล ซึ่งได้ผลลัพธ์ที่มีความถูกต้องสูงถึง 87.71% (Reis, 2004)

- Srikanta Tirthapura และคณะ นำเสนอการใช้ Edit Distance ในการจัดทำสารบัญ (Index) ให้แก่รูปภาพที่ถูกเก็บในฐานข้อมูล โดยอาศัยการแปลงรูปภาพที่เป็นรูปทรงต่างๆ ให้อยู่ในรูปของ Shock Graph เมื่อรูปภาพดังกล่าวได้ถูกแปลงให้อยู่ในรูปของ Shock Graph แล้ว จึงนำค่าที่อยู่บน Shock Graph มาทำการเปรียบเทียบกันด้วยวิธี Edit Distance เพื่อตรวจสอบว่ารูปภาพมีความใกล้เคียงหรือเหมือนกัน แล้วจึงนำมาจัดเป็นรูปภาพที่อยู่ในกลุ่มเดียวกัน (Tirthapura, 1998)

- Philip N. Klein และคณะ นำเสนอการประยุกต์ใช้ Edit Distance ในการจับคู่รูปทรงที่มีลักษณะเหมือนกันหรือใกล้เคียงกันโดยอาศัยหลักการ Shock Graph ที่จะทำการแปลงรูปทรงให้อยู่ในรูปของ Shock Graph ที่มีลักษณะเป็นแกนของรูปทรงดังกล่าว ซึ่งจากแกนของ Shock Graph นี้เองที่จะเป็นตัวบอกค่าที่จะใช้ในการนำไปเปรียบเทียบว่าระหว่าง Shock Graph ที่นำมาเปรียบเทียบกันนั้นเป็นรูปทรงที่มีลักษณะเหมือนกันหรือต่างกัน โดยค่าต่างๆ ที่นำมาเปรียบเทียบเหล่านี้ได้มาจากการวัดมุมที่เกิดขึ้นระหว่างเส้นของ Shock Graph เดียวกัน รวมไปถึงความยาวของเส้นด้วย (Klein, 2001)

- Marius Gheorghescu นำเสนอวิธีการในการจำแนกหมวดหมู่ไวรัสคอมพิวเตอร์โดยอัตโนมัติโดยใช้ทฤษฎี Edit Distance ในการจำแนก เนื่องจากไวรัสแต่ละชนิดนั้นจะมีโครงสร้างหลักของชุดคำสั่งที่มีลักษณะเหมือนกัน และแตกต่างกันเพียงแค่บางส่วนของชุดคำสั่ง จึงทำให้สามารถนำโครงสร้างหลักดังกล่าวของไวรัสมาใช้ในการเปรียบเทียบกันด้วยทฤษฎี Edit Distance เพื่อหาว่าไวรัสที่นำมาเปรียบเทียบกันนั้นเป็นไวรัสที่อยู่ในตระกูลเดียวกันหรือไม่ ซึ่งผลที่ได้จากการศึกษานี้ได้ผลเป็นที่น่าพอใจ โดยระบบสำหรับจำแนกหมวดหมู่ไวรัสสามารถทำงานได้แบบ Real Time โดยไม่มีผลกระทบต่อประสิทธิภาพของเครื่องคอมพิวเตอร์ที่ใช้งาน (Gheorghescu, 2006)

- Marco Bertini และคณะ นำเสนอการเปรียบเทียบความเหมือนของวิดีโอคลิปโดยใช้ MPEG-7 Descriptor และวิธี Edit Distance โดยการนำคลิปวิดีโอมาแบ่งออกเป็นส่วนๆ (Segment) และในแต่ละ Segment ก็ทำการแยก Signature ของคลิปวิดีโอออกมา ซึ่ง Signature นี้เองจะเป็นตัวบ่งบอกถึงเอกลักษณ์ของคลิปวิดีโอในแต่ละ Segment เมื่อแยกได้ดังนี้แล้ว จากนั้นจึงใช้วิธี Edit Distance ในการเปรียบเทียบ Signature ของคลิปวิดีโอที่นำมาเปรียบเทียบกัน ซึ่งผลลัพธ์ที่ได้ค่อนข้างถูกต้องและแม่นยำ (Bertini, 2006)
- Daniel P. Lopresti นำเสนอการเปรียบเทียบลายมืออิเล็กทรอนิกส์บนอุปกรณ์คอมพิวเตอร์พกพาเพื่อแปลงไปเป็นตัวอักษร โดยใช้การเปรียบเทียบด้วยการคำนวณตามทฤษฎี Edit Distance โดยการดูจากเส้นที่เขียนลงไปบนอุปกรณ์รับค่า แล้วนำค่าที่ได้จากการเขียนในแต่ละจุดไปเปรียบเทียบกับค่ารูปแบบมาตราฐาน (Pattern Value) ที่เก็บไว้ในฐานข้อมูลโดยใช้สมการ Levenshtien Distance ในการคำนวณ เพื่อหาค่าในฐานข้อมูลที่มีค่าใกล้เคียงกันที่บ่งบอกว่าการเขียนนั้นเป็นตัวอักษรตัวใด แล้วนำมาแสดงผลออกมาทางหน้าจอของอุปกรณ์ ซึ่งผลลัพธ์ที่ได้ค่อนข้างมีความแม่นยำ และงานวิจัยชิ้นนี้ก็ได้รับการจดสิทธิบัตรของประเทศสหรัฐอเมริกาด้วย (Lopresti, 1998)

บทที่ 3
วิธีดำเนินการวิจัย

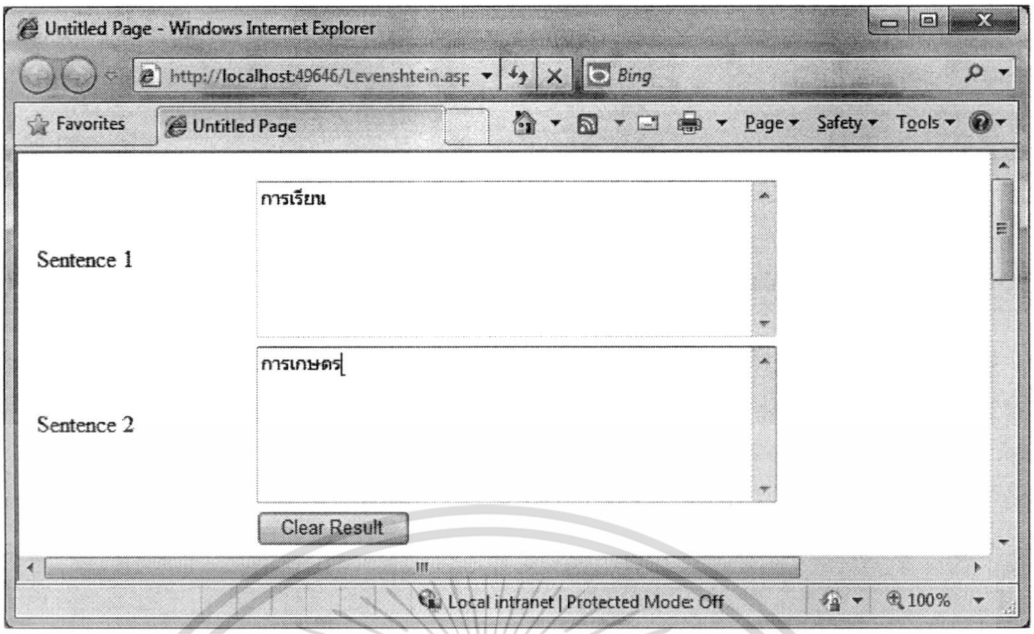
ในบทนี้จะกล่าวถึงปัญหาของการนำการวัดระยะการแก้ไขมาประยุกต์ใช้กับภาษาไทย รวมไปถึงแนวทางในการแก้ปัญหาดังกล่าว เพื่อให้สามารถนำการวัดระยะการแก้ไขมาประยุกต์ใช้กับภาษาไทย และยังกล่าวถึงวิธีการในการทดสอบการวัดระยะการแก้ไขด้วยวิธีการต่างๆ เพื่อใช้ในการหาว่าวิธีการวัดระยะการแก้ไขวิธีการใดเหมาะสมที่จะนำมาประยุกต์ใช้กับภาษาไทย

3.1 ปัญหาของการใช้ Edit Distance กับภาษาไทย

เนื่องด้วยภาษาไทยนั้นจะมีลักษณะของข้อความที่มีตัวอักษรเขียนติดกันทั้งหมด และเมื่อจบประโยคจึงมีการเว้นวรรค ซึ่งไม่มีโครงสร้างของคำและประโยคที่ชัดเจนแบบภาษาอังกฤษที่จะมีการเว้นวรรคเพื่อแบ่งคำ และมีการใส่จุด (.) เมื่อสิ้นสุดประโยค ทำให้ภาษาไทยนั้นไม่สามารถที่จะนำข้อความทั้งสองมาเปรียบเทียบกันได้ เพราะไม่มีการแบ่งขอบเขตของคำที่ชัดเจน เพราะหากไม่มีการแบ่งขอบเขตของคำ ค่าในการเปรียบเทียบจะเกิดความคลาดเคลื่อน เนื่องจากระบบจะไม่สามารถจำแนกได้ว่าส่วนใดเป็นคำที่ผิด หรือส่วนใดถูกต้อง ทำให้ต้องมีการตัดแบ่งคำในข้อความออกมาเป็นชุดของคำก่อนที่จะมาทำการวัดระยะการแก้ไขของข้อความ

3.2 การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย

สำหรับการทดสอบการวัดระยะการแก้ไขข้อความด้วยวิธี Edit Distance กับข้อความภาษาไทยนั้น ได้มีการทดสอบโดยใช้กลุ่มของข้อความภาษาไทยตัวอย่างจำนวนหนึ่ง มาทำการทดสอบบนโปรแกรมที่ถูกพัฒนาขึ้น เพื่อคำนวณหาระยะในการแก้ไขข้อความสำหรับภาษาไทย เพื่อเปรียบเทียบหาวิธีการวัดระยะที่เหมาะสมสำหรับข้อความภาษาไทย ซึ่งได้นำวิธีต่างๆ มาใช้ในการเปรียบเทียบกัน 3 วิธี คือ Levenshtein Distance, Hamming Distance และ Needleman-Wunsch Algorithm โดยที่โปรแกรมจะคำนวณหาค่าระยะการแก้ไขข้อความ จำนวนคำที่แบ่งได้ รูปแบบของคำที่แบ่ง รวมไปถึงระยะเวลาที่ใช้ในการประมาลผลเพื่อคำนวณหาค่าของระยะการแก้ไขที่วัดได้ ซึ่งผลลัพท์ที่ได้นั้นมาจากการนำข้อความตัวอย่างสองข้อความ นำมากรอกลงในช่องสำหรับกรอกข้อความเพื่อนำมาเปรียบเทียบ



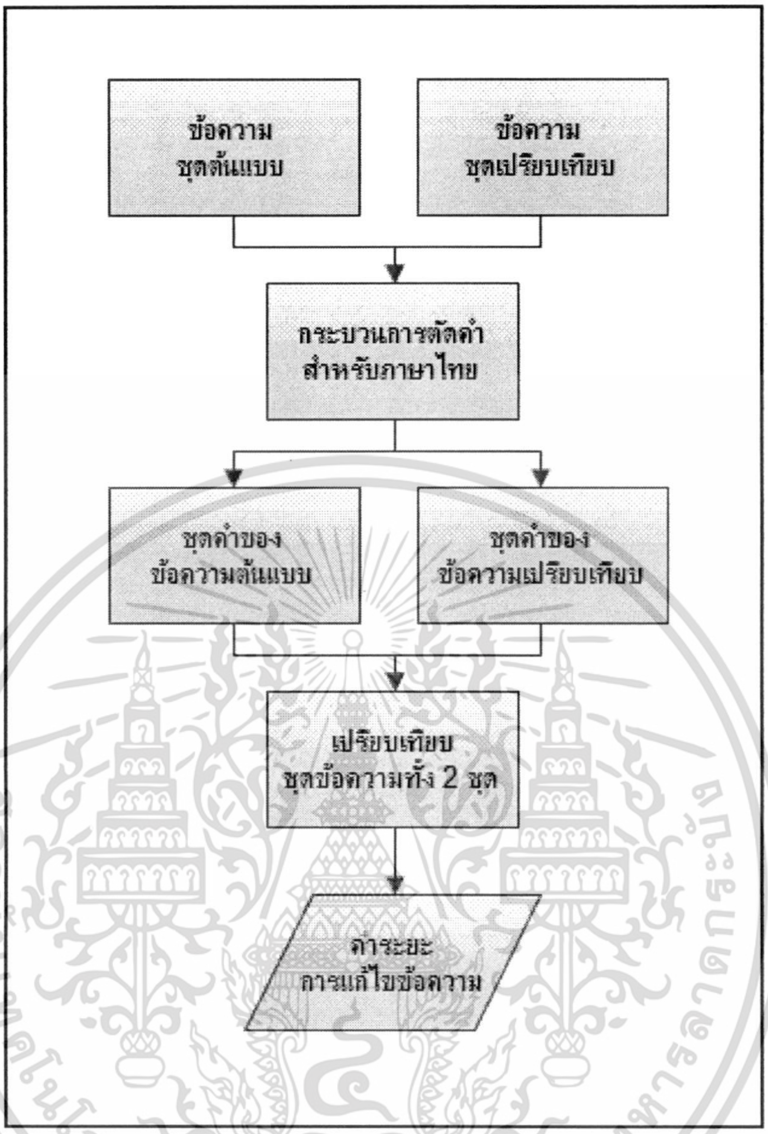
ภาพที่ 3.1 หน้าจอโปรแกรมสำหรับทดสอบ Edit Distance ในส่วนการกรอกค่าที่จะเปรียบเทียบ

3.2.1 ลักษณะการทำงานของโปรแกรม

ลักษณะการทำงานของโปรแกรมนั้นจะแบ่งขั้นตอนหลักๆ ภายในโปรแกรมได้ 2 ส่วน คือ ขั้นตอนในการตัดแบ่งข้อความออกเป็นคำ และขั้นตอนในการนำคำแต่ละคำของทั้งสองข้อความมาทำการเปรียบเทียบกันด้วยวิธี Edit Distance

ขั้นตอนในการตัดแบ่งข้อความออกเป็นคำนั้นได้ใช้โปรแกรม SWATh (Smart Word Analysis for Thai) ซึ่งเป็นโปรแกรมสำหรับตัดคำภาษาไทยที่พัฒนาขึ้นโดย ไพศาล เจริญพร-สวัสดิ์ (Charoenpornsawat, n.p.) โดยที่โปรแกรมดังกล่าวจะใช้ลักษณะการตัดคำภาษาไทยหลายแบบผสมกัน ได้แก่ การตัดคำแบบยาวที่สุด (Longest Matching) และการตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching) ร่วมกับการใช้ค่าสถิติการเกิดของคำและลำดับของหน้าที่ของคำ (Part of speech) เข้ามาช่วยในการตัดคำ

สำหรับขั้นตอนในการนำคำแต่ละคำของทั้งสองข้อความมาทำการเปรียบเทียบกันนั้น จะเป็นการนำคำแต่ละคำของข้อความทั้งสองมาเปรียบเทียบกันแบบคำต่อคำ และเรียงตามลำดับเพื่อหาค่าในการแก้ไขข้อความ โดยการนำคำที่หนึ่งของข้อความที่หนึ่ง มาเปรียบเทียบกับคำที่หนึ่งของข้อความที่สอง และทำเช่นนี้ต่อไปจนกระทั่งถึงคำสุดท้ายของข้อความ



ภาพที่ 3.2 กระบวนการในการวัดระยะแก้ไขข้อความภาษาไทย

3.3 วิธีการที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย

ในการทดสอบการวัดระยะการแก้ไขข้อความภาษาไทยนั้น มีการใช้วิธีการที่ต่างกัน 3 วิธี มาใช้ในการวัดหาระยะการแก้ไข โดยแต่ละวิธีนั้นจะมีลักษณะและข้อจำกัดที่แตกต่างกันดังนี้

- Levenshtein Distance เป็นการวัดหาค่าความแตกต่างระหว่างข้อความสองชุด โดยนับจำนวนครั้งของการแทรก ตัดออก และแทนที่ตัวอักษร ซึ่งในการกระทำดังกล่าวแต่ละครั้ง คะแนนในการวัดค่าจะเท่ากับ +1 ซึ่งผลคะแนนสุดท้ายที่วัดได้ จะเป็นผลรวมของจำนวนครั้งที่ใช้ในการแก้ไขข้อความ

```
Public Function Levenshtein(ByVal s1 As String, ByVal s2 As String)

    Dim i As Integer
    Dim j As Integer
    Dim l1 As Integer
    Dim l2 As Integer
    Dim min1 As Integer
    Dim min2 As Integer

    l1 = Len(s1)
    l2 = Len(s2)
    Dim d(l1, l2)
    For i = 0 To l1
        d(i, 0) = i
    Next
    For j = 0 To l2
        d(0, j) = j
    Next
    For i = 1 To l1
        For j = 1 To l2
            If Mid(s1, i, 1) = Mid(s2, j, 1) Then
                d(i, j) = d(i - 1, j - 1)
            Else
                min1 = d(i - 1, j) + 1
                min2 = d(i, j - 1) + 1
                If min2 < min1 Then
                    min1 = min2
                End If
                min2 = d(i - 1, j - 1) + 1
                If min2 < min1 Then
                    min1 = min2
                End If
                d(i, j) = min1
            End If
        Next
    Next

    Levenshtein = d(l1, l2)
    Return Levenshtein

End Function
```

ภาพที่ 3.3 Source Code ภาษา VB สำหรับคำนวณหาค่า Levenstein Distance

- Hamming Distance เป็นการวัดหาค่าความแตกต่างระหว่างข้อความสองชุด โดยที่ข้อความทั้งสองต้องมีจำนวนตัวอักษรที่เท่ากัน โดยการเปรียบเทียบจะเปรียบเทียบแบบตัวอักษรต่อตัวอักษร จากตัวอักษรตัวแรกจนถึงตัวอักษรตัวสุดท้าย หากพบว่าตัวอักษรต่างกันคะแนนในการวัดค่าจะเท่า +1 ซึ่งผลคะแนนสุดท้ายที่วัดได้ จะเป็นผลรวมของจำนวนตัวอักษรที่มีความแตกต่างกันทั้งหมด

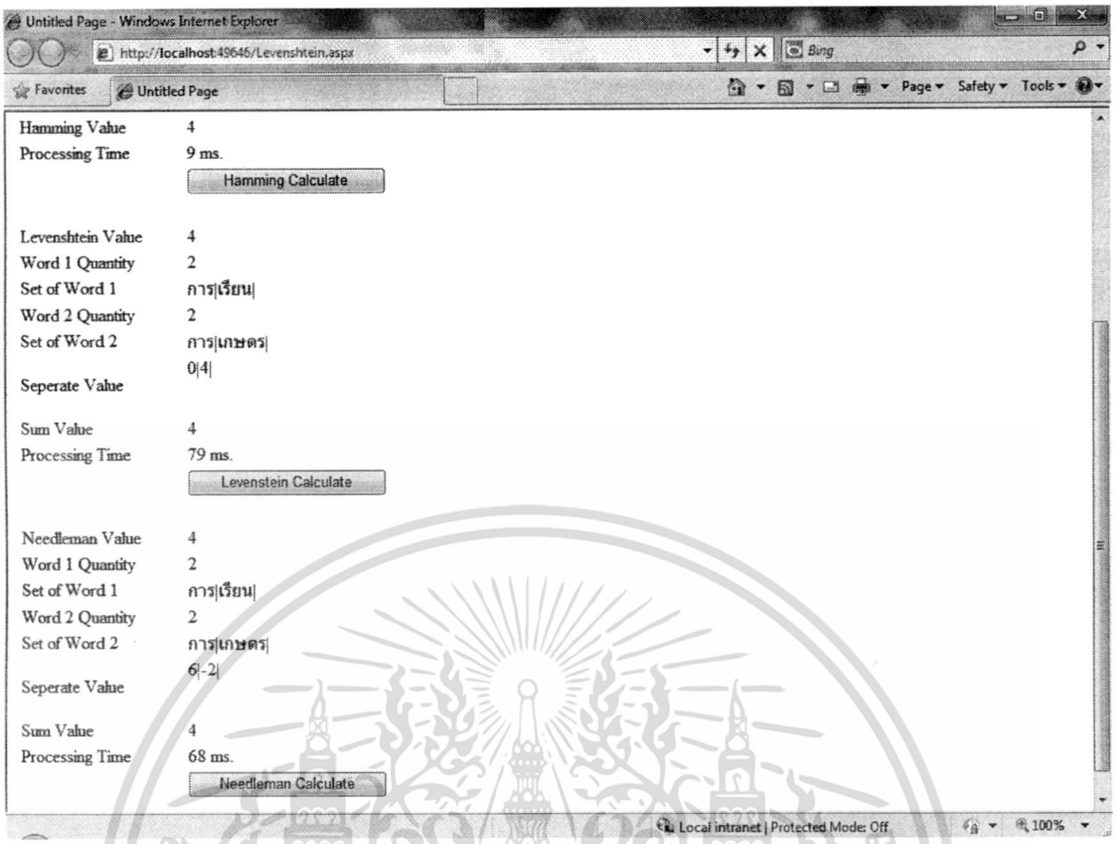
```
Public Function Hamming(ByVal s1 As String, ByVal s2 As String)

    Dim i As Integer
    Dim WordLength As Integer = 0
    Dim HammingCount As Integer = 0

    If s1.Trim.Length <> s2.Trim.Length Then
        MsgBox("The comparison value is not equal !!",_
        MsgBoxStyle.Critical = MsgBoxStyle.OkOnly, "Error")
    Else
        Dim Word1() As Char = s1.ToCharArray()
        Dim Word2() As Char = s2.ToCharArray()
        WordLength = Len(Word1)

        For i = 0 To WordLength - 1
            If Word1(i) = Word2(i) Then
                '-- Next loop --
            Else
                HammingCount = HammingCount + 1
            End If
        Next
    End If

    Hamming = HammingCount
    Return Hamming

End Function
```

ภาพที่ 3.4 Source Code ภาษา VB สำหรับคำนวณหาค่า Hamming Distance

- Needleman-Wunsch Algorithm เป็นการวัดหาค่าความเหมือนระหว่างข้อความสองชุด โดยเป็นการนำคำของข้อความทั้งสองมาหาค่าที่ใช้ในการจัดเรียงตัวอักษร โดยที่เมื่อนำคำมาเปรียบเทียบกันแล้วตำแหน่งที่เปรียบเทียบกันเป็นเป็นตัวอักษรทั้งคู่และเป็นตัวอักษรตัวเดียวกัน ค่าที่ได้จะเป็น +2 แต่ถ้าเป็นตัวอักษรคนละตัวกัน ค่าที่ได้จะเป็น -1 และหากตำแหน่งที่เปรียบเทียบเป็นช่องว่างกับตัวอักษร ค่าที่ได้จะเป็น -2 เมื่อได้ผลรวมมาแล้วให้นำค่าที่ได้มาคูณด้วย -1 เพื่อให้ค่าคะแนนที่ได้ไปในทิศทางเดียวกันกับอีก 2 วิธีข้างต้น เพื่อง่ายต่อการเปรียบเทียบ

```
Public Function Needleman(ByVal s1 As String, ByVal s2 As String)

    Dim refSeq As String = s1
    Dim alignSeq As String = s2

    Dim refSeqCnt As Integer = refSeq.Length + 1
    Dim alineSeqCnt As Integer = alignSeq.Length + 1

    Dim scoringMatrix As Integer(,) = New Integer(alineSeqCnt - 1,_
    refSeqCnt - 1) {}

    For i As Integer = 0 To alineSeqCnt - 1
        scoringMatrix(i, 0) = 0
    Next

    For j As Integer = 0 To refSeqCnt - 1
        scoringMatrix(0, j) = 0
    Next

    Dim FinalValue As Integer

    For i As Integer = 1 To alineSeqCnt - 1
        For j As Integer = 1 To refSeqCnt - 1
            Dim scroeDiag As Integer = 0
            If refSeq.Substring(j - 1, 1) = alignSeq.Substring_
            (i - 1, 1) Then
                scroeDiag = scoringMatrix(i - 1, j - 1) + 2
            Else
                scroeDiag = scoringMatrix(i - 1, j - 1) + (-1)
            End If

            Dim scroeLeft As Integer = scoringMatrix(i, j - 1) + (-2)
            Dim scroeUp As Integer = scoringMatrix(i - 1, j) + (-2)

            Dim maxScore As Integer = Math.Max_
            (Math.Max(scroeDiag, scroeLeft), scroeUp)

            scoringMatrix(i, j) = maxScore
            FinalValue = maxScore
        Next
    Next

    Needleman = FinalValue
    Return Needleman
End Function
```

ภาพที่ 3.5 Source Code ภาษา VB สำหรับคำนวณหาค่า Needleman-Wunsch Algolithm



ภาพที่ 3.6 หน้าจอโปรแกรมสำหรับทดสอบ Edit Distance ในส่วนการแสดงผลการคำนวณ

บทที่ 4
ผลการศึกษาวิจัย

ในบทนี้จะกล่าวถึงการนำชุดข้อความตัวอย่างที่ได้เตรียมไว้มาทำการทดสอบการวัดระยะแก้ไข เพื่อทดสอบหาวิธีการวัดระยะการแก้ไขที่เหมาะสมแก่การนำมาประยุกต์ใช้กับข้อความภาษาไทย โดยการทดสอบนั้นได้ทดสอบกับชุดข้อความตัวอย่างในหลายลักษณะ โดยที่การทดสอบ จะทำการทดสอบบนเครื่องคอมพิวเตอร์เครื่องเดียวกันทั้งหมด

4.1 เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย ได้ใช้เครื่องคอมพิวเตอร์ชนิดโน๊ตบุ๊ค รุ่น HP ProBook 4430s ซึ่งมีสเปคดังนี้

ตารางที่ 4.1 สเปคของเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบการวัดระยะการแก้ไข

Processor (CPU)	Intel® Core™ i5-2450M CPU @ 2.50 GHz
RAM	4 GB
Hard Disk	500 GB
VGA	Intel® HD Graphics Family
Operating System	Windows 7 Professional (32-bit)
Region & Language Setting	Thai (Thailand)

4.2 รูปแบบของการทดสอบการวัดระยะการแก้ไข

ในการทำการทดสอบการวัดระยะการแก้ไขของข้อความ ได้แบ่งรูปแบบของการทดสอบออกเป็น 5 ลักษณะ ดังนี้

- การทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย
- การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น
- การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง
- การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก
- การทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทย

4.2.1 การทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย

ในการทดสอบการวัดระยะการแก้ไขแบบคำเดียว จะเป็นการทดสอบเพื่อหาค่าการวัดระยะการแก้ไขที่ได้ของคำที่นำมาเปรียบเทียบกัน โดยใช้กลุ่มของชุดข้อมูลตัวอย่างจำนวน 400 ชุด มาใช้ในการทดสอบ และนี่คือกลุ่มคำตัวอย่างส่วนหนึ่งที่นำมาใช้ในการทดสอบ

ตารางที่ 4.2 ชุดข้อมูลตัวอย่างที่ใช้ในการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย

ชุดข้อมูลที่	ข้อความที่ 1 (S_1)	ข้อความที่ 2 (S_2)
1	กัด	ฟัน
2	หมู	หมา
3	จาน	ชาม
4	กิ้งก่า	ค้างคาว
5	นรก	สวรรค์

4.2.2 ผลการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย

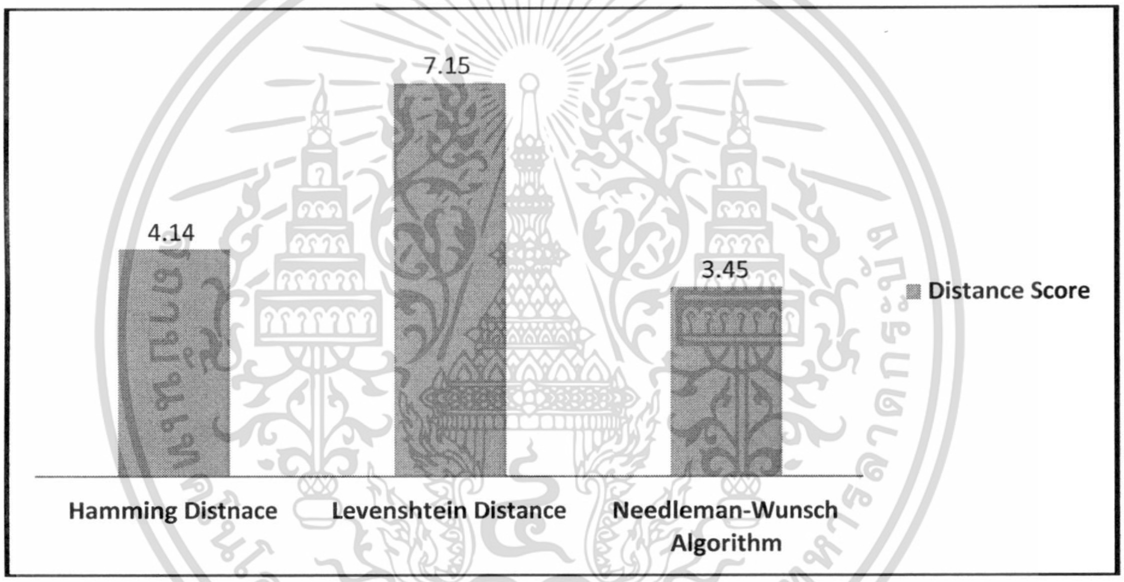
ผลการทดลองที่ได้จากการนำกลุ่มคำตัวอย่างข้างต้นมาทำการทดสอบผ่านโปรแกรมสำหรับคำนวณหาค่าการวัดระยะการแก้ไข โดยที่ Hamming Distance จะวัดระยะแบบไม่ตัดแบ่งคำ แต่ Levenshtein Distance และ Needleman-Wunch Algorithm จะทำการตัดแบ่งคำก่อนการวัดระยะ ซึ่งได้ค่าดังตารางต่อไปนี้

ตารางที่ 4.3 ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย ชนิดไม่ตัดแบ่งคำก่อนการวัดระยะ

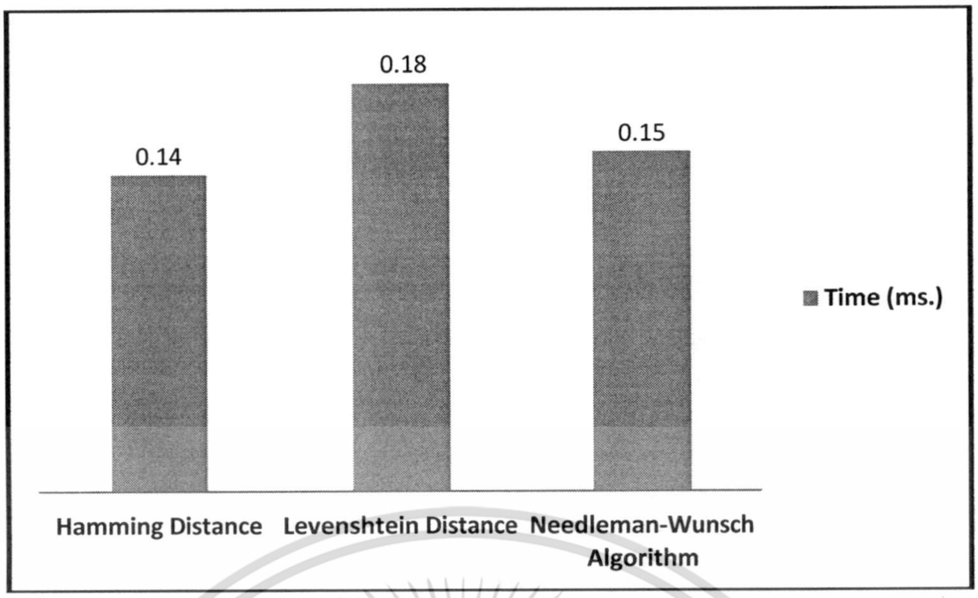
ชุดข้อมูลที่	ข้อความที่ 1 (S_1)	ข้อความที่ 2 (S_2)	Hamming		Levenshtein		Needleman-Wunsch	
			ค่าที่ได้	เวลา	ค่าที่ได้	เวลา	ค่าที่ได้	เวลา
1	กัด	ฟัน	2	1	2	1	0 (0)	0
2	หมู	หมา	1	0	1	1	-3 (3)	0
3	จาน	ชาม	2	0	2	0	0 (0)	1
4	กิ้งก่า	ค้างคาว	6	0	6	0	2 (-2)	0
5	นรก	สวรรค์	N/A	N/A	5	0	2 (-2)	0

ตารางที่ 4.4 ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย ชนิดตัดแบ่งคำก่อนการวัดระยะ

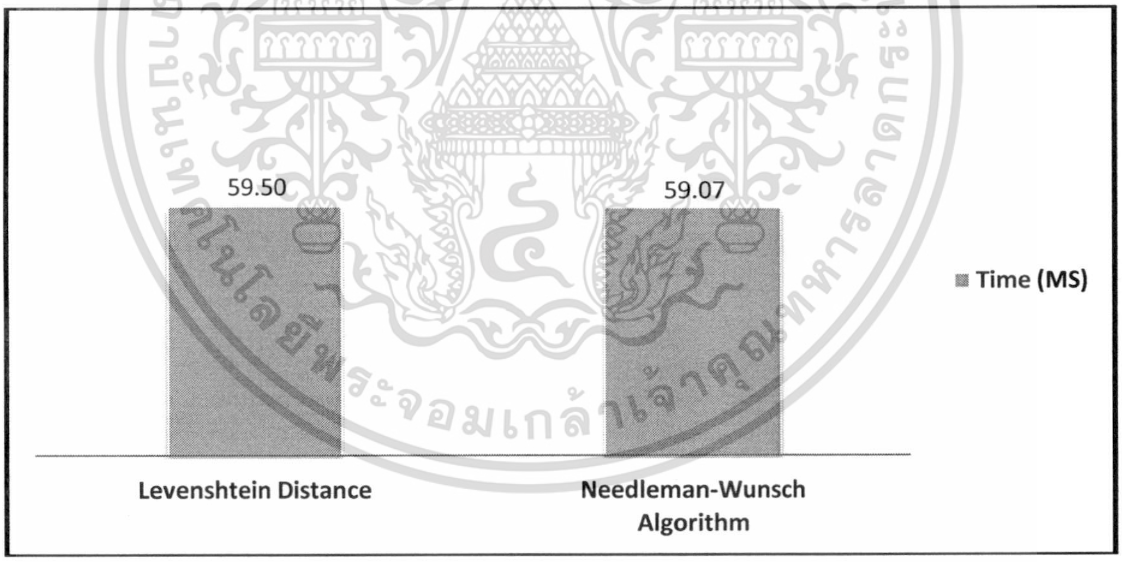
ชุดข้อมูลที่	ข้อความที่ 1 (S_1)	ข้อความที่ 2 (S_2)	Levenshtein Distance		Needleman-Wunsch Algorithm	
			ค่าที่ได้	เวลา	ค่าที่ได้	เวลา
1	กัด	ฟัน	2	49	0 (0)	60
2	หมู	หมา	1	61	-3 (3)	55
3	จาน	ชาม	2	58	0 (0)	58
4	กิ้งก่า	ค้างคาว	6	62	2 (-2)	48
5	นรก	สวรรค์	5	54	2 (-2)	58



ภาพที่ 4.1 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย



ภาพที่ 4.2 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย ชนิดไม่ตัดแบ่งคำก่อนการวัดระยะ



ภาพที่ 4.3 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทย ชนิดตัดแบ่งคำก่อนการวัดระยะ

จากผลการทดลองการวัดระยะการแก้ไขแบบคำหรือวลีภาษาไทยข้างต้นจะเห็นว่า กระบวนการวัดระยะการแก้ไขจากชุดข้อมูลตัวอย่างต่างๆ การวัดระยะการแก้ไขแบบไม่แบ่งคำ ซึ่งใช้กับ Hamming Distance จะใช้เวลาในการประมวลผลน้อยมาก จนโปรแกรมไม่สามารถวัดค่าออกมาได้ในหน่วยของมิลลิวินาที (Millisecond) ซึ่งจะแสดงค่าของเวลาที่วัดได้เท่ากับ 0 ms. ดังที่แสดงในตารางแสดงผลการทดสอบในส่วนของ Hamming Distance

แต่จากตารางข้างต้นจะสังเกตเห็นว่า ในชุดคำที่ 5 คือ "นรก" กับ "สวรรค์" จะไม่สามารถวัดหาค่าการวัดระยะการแก้ไขด้วยวิธี Hamming Distance ได้ เนื่องจากคำที่นำมาเปรียบเทียบกันมีขนาดไม่เท่ากัน ซึ่งเป็นข้อเสียของวิธี Hamming Distance

4.2.3 การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น

ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น จะเป็นการทดสอบเพื่อหาค่าการวัดระยะการแก้ไขที่ได้ของชุดข้อความตัวอย่างที่นำมาเปรียบเทียบกัน โดยใช้กลุ่มของชุดข้อมูลตัวอย่างจำนวน 250 ชุด มาทำการตัดแบ่งคำก่อน แล้วจึงทดสอบเปรียบเทียบด้วยวิธี Levenshtein Distance และ Needleman-Wunsch Algorithm และนี่คือกลุ่มคำตัวอย่างส่วนหนึ่งที่นำมาใช้ในการทดสอบ

ตารางที่ 4.5 ชุดข้อมูลตัวอย่างที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น

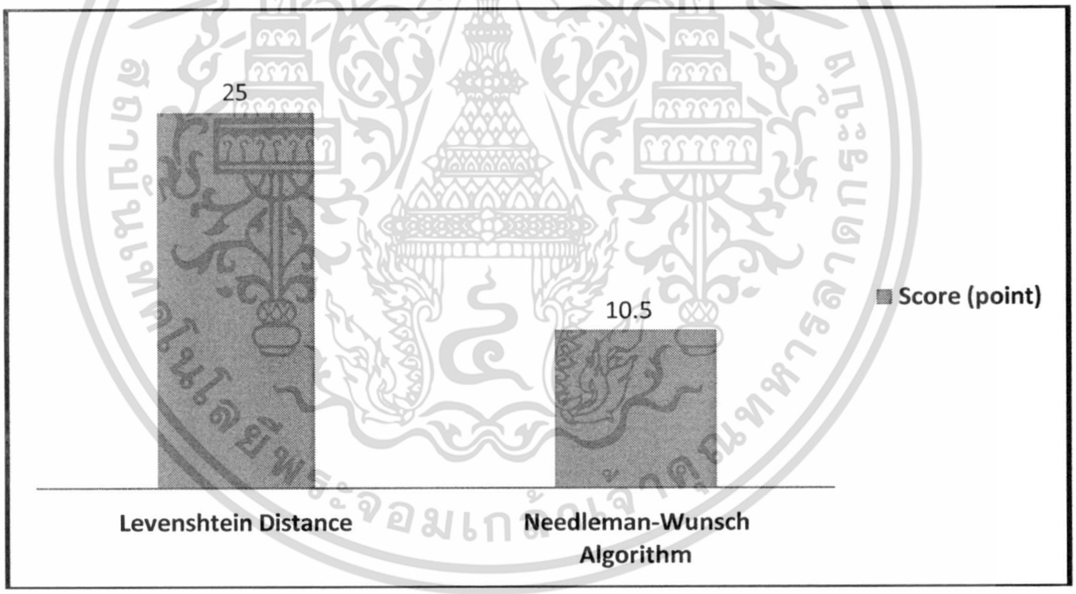
ชุดข้อมูลที่	ข้อความที่ 1 (S_1)	ข้อความที่ 2 (S_2)
1	การเรียน	การเกษตร
2	ตกกระไดพลอยโจน	ตีวัวกระทบคลาด
3	รู้หลบเป็นปีก รู้หลีกเป็นหาง	รักดีหามจั่ว รักชั่วหามเสา
4	ผีแดงแห้วชอว์แบะท่าไปเชลซี	ผีคว้าชัยรับขวัญมาตาเปิดรังทุบคาร์ดิฟฟ์
5	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง	มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

4.2.4 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น

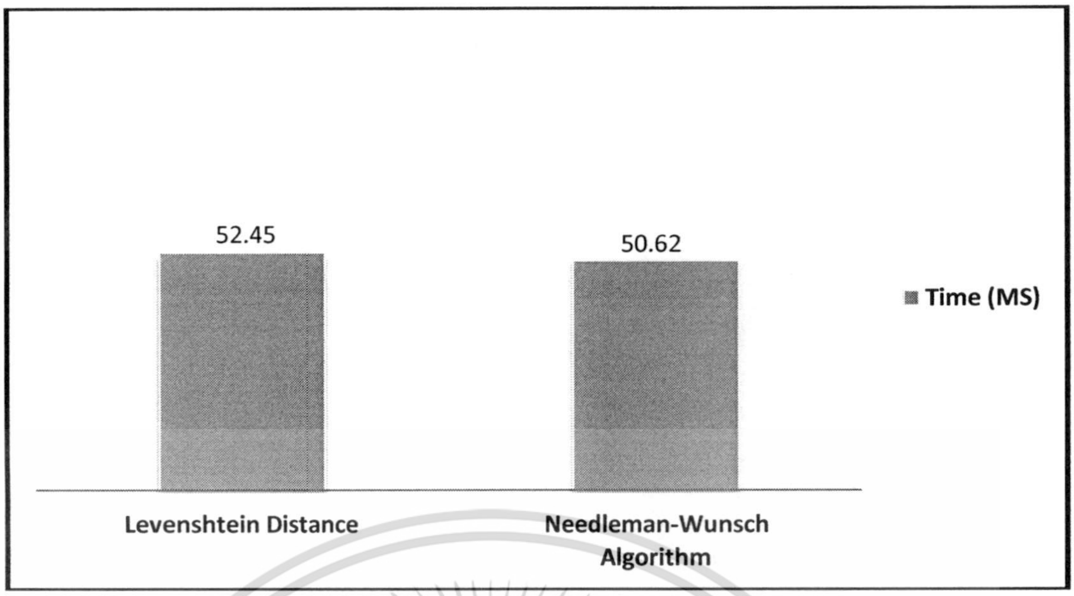
ผลการทดลองที่ได้จากการนำกลุ่มข้อความตัวอย่างข้างต้นมาทำการทดสอบผ่านโปรแกรมสำหรับคำนวณหาค่าการวัดระยะการแก้ไข ได้ค่าดังต่อไปนี้

ตารางที่ 4.6 ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น

ชุดข้อมูลที่	Levenshtein Distance		Needleman-Wunsch Algorithm	
	ค่าที่ได้	เวลา	ค่าที่ได้	เวลา
1	4	52	-4 (4)	50
2	25	54	4 (-4)	50
3	26	53	14 (-14)	51
4	37	53	15 (-15)	53
5	53	59	30 (-30)	52



ภาพที่ 4.4 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น



ภาพที่ 4.5 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น

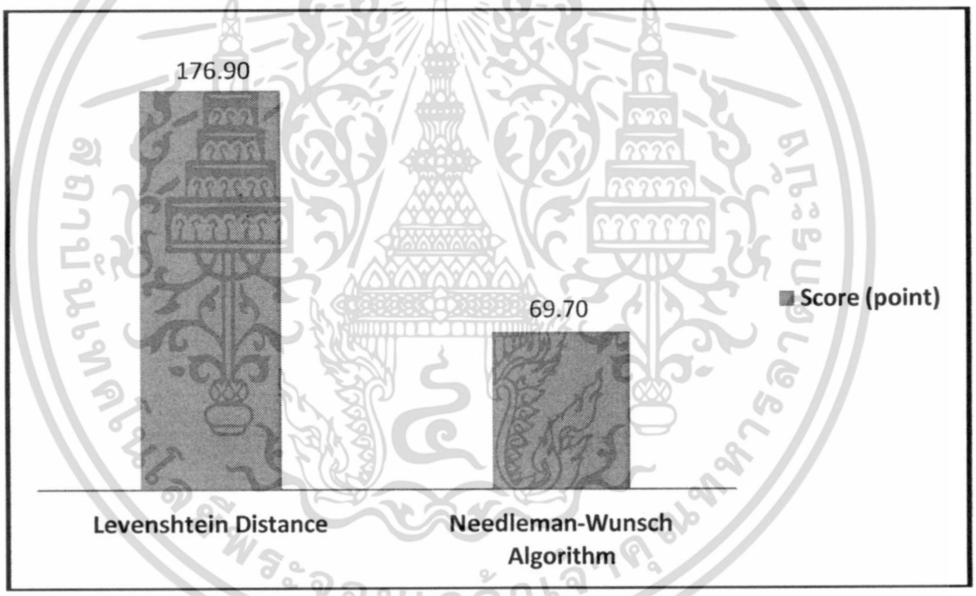
จากตารางผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้นข้างต้น จะเห็นว่าในการวัดระยะการแก้ไขทั้ง 2 วิธี คือ Levenshtein Distance และ Needleman-Wunsch Algorithm ต่างก็ใช้เวลาในการประมวลผลเพื่อหาค่าระยะการแก้ไขแบบมีการตัดแบ่งคำใกล้เคียงกัน โดยสังเกตได้จากแผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น โดยที่ Levenstein Distance ใช้ระยะเวลาประมวลผลเฉลี่ยเท่ากับ 52.45 ms. ในขณะที่ Needleman-Wunsch Algorithm ใช้เวลาประมวลผลเฉลี่ยเท่ากับ 50.62 ms. และเนื่องจาก Hamming Distance นั้นจะวัดหาค่าระยะการแก้ไขได้ในกรณีที่ข้อความที่นำมาเปรียบเทียบกันมีขนาดข้อความเท่ากันเท่านั้น จึงทำให้การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้นไม่นำ Hamming Distance มาใช้ในการทดสอบ เพราะด้วยข้อจำกัดของวิธี Hamming Distance เอง

4.2.5 การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง

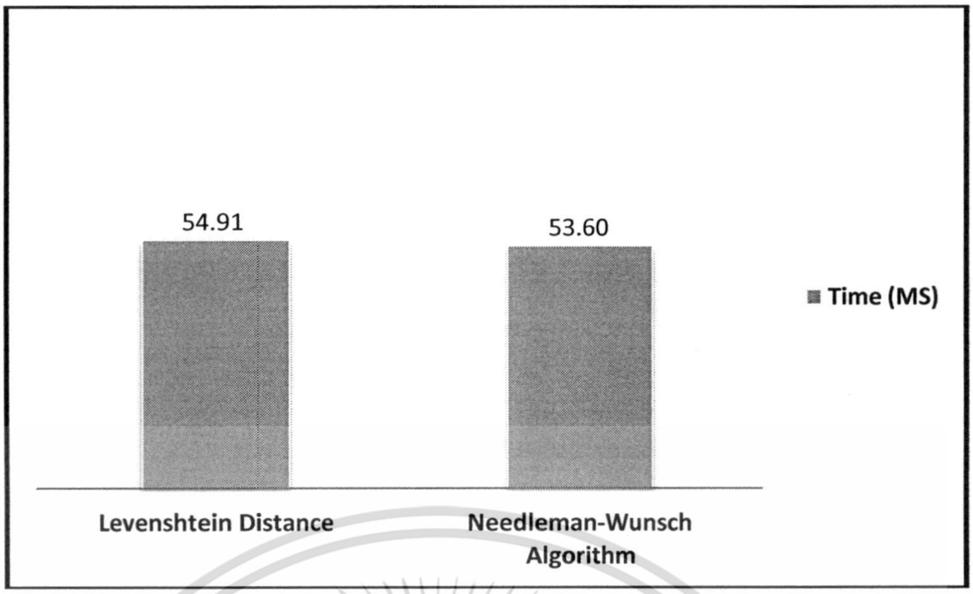
ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง จะเป็นการทดสอบเพื่อหาค่าการวัดระยะการแก้ไขที่ได้ของชุดข้อความตัวอย่างที่นำมาเปรียบเทียบกัน โดยใช้กลุ่มของชุดข้อมูลตัวอย่างจำนวน 200 ชุด มาทำการตัดแบ่งคำก่อน แล้วจึงทดสอบเปรียบเทียบด้วยวิธี Levenshtein Distance และ Needleman-Wunsch Algorithm และนี่คือกลุ่มคำตัวอย่างส่วนหนึ่งที่นำมาใช้ในการทดสอบ

4.2.6 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง

ผลการทดลองที่ได้จากการนำกลุ่มข้อมูลตัวอย่างจำนวน 250 ชุด มาทำการทดสอบผ่านโปรแกรมสำหรับคำนวณหาค่าการวัดระยะการแก้ไข และนำผลที่ได้มาหาค่าเฉลี่ย ซึ่งได้ผลการทดสอบดังต่อไปนี้



ภาพที่ 4.6 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง



ภาพที่ 4.7 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง

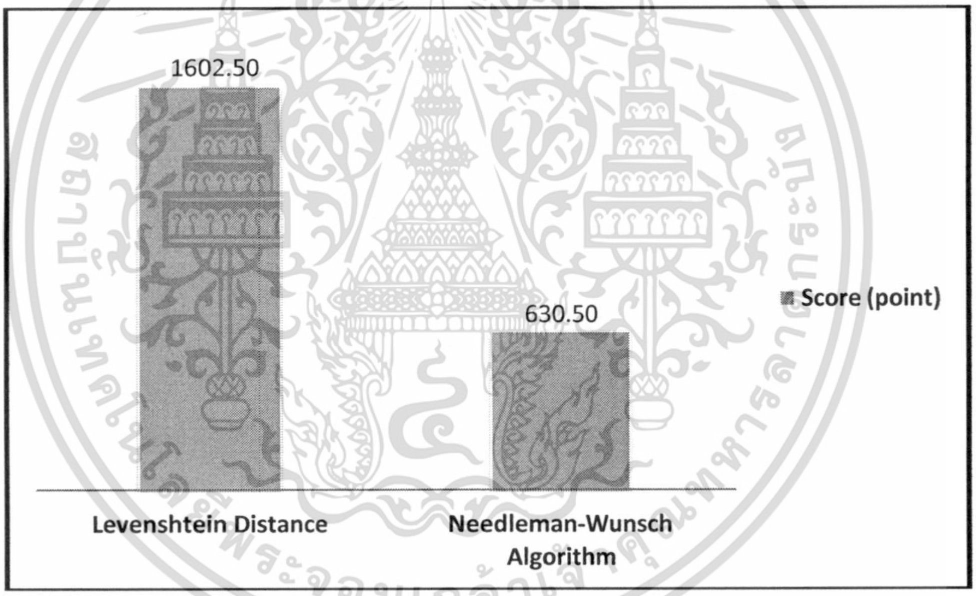
จากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง จะเห็นว่าในการวัดระยะการแก้ไขทั้ง 2 วิธี คือ Levenshtein Distance และ Needleman-Wunsch Algorithm ต่างก็ใช้เวลาในการประมวลผลเพื่อหาค่าระยะการแก้ไขแบบมีการตัดแบ่งคำใกล้เคียงกัน โดยสังเกตได้จากแผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวปานกลาง โดยที่ Levenstein Distance ใช้ระยะเวลาประมวลผลเฉลี่ยเท่ากับ 54.91 ms. ในขณะที่ Needleman-Wunsch Algorithm ใช้เวลาประมวลผลเฉลี่ยเท่ากับ 53.60 ms. และเช่นเดียวกันกับการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น การทดสอบครั้งนี้มีขนาดของข้อความที่ไม่เท่ากัน จึงไม่นำวิธี Hamming Distance มาใช้ในการทดสอบ

4.2.7 การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก

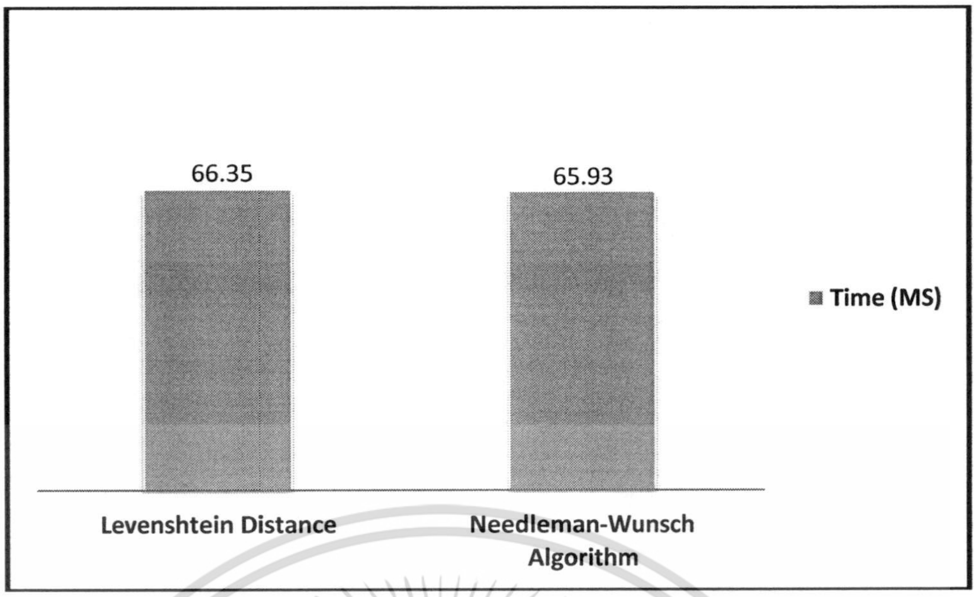
ในการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้น จะเป็นการทดสอบเพื่อหาค่าการวัดระยะการแก้ไขที่ได้ของชุดข้อความตัวอย่างที่นำมาเปรียบเทียบกัน โดยใช้กลุ่มของชุดข้อมูลตัวอย่างจำนวน 100 ชุด มาทำการตัดแบ่งคำก่อน แล้วจึงทดสอบเปรียบเทียบด้วยวิธี Levenshtein Distance และ Needleman-Wunsch Algorithm และนี่คือกลุ่มคำตัวอย่างส่วนหนึ่งที่นำมาใช้ในการทดสอบ

4.2.8 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก

ผลการทดลองที่ได้จากการนำกลุ่มข้อมูลตัวอย่างจำนวน 100 ชุด มาทำการทดสอบผ่านโปรแกรมสำหรับคำนวณหาค่าการวัดระยะการแก้ไข และนำผลที่ได้มาหาค่าเฉลี่ย ซึ่งได้ผลการทดสอบดังต่อไปนี้



ภาพที่ 4.8 แผนภูมิแสดงค่าระยะเฉลี่ยที่วัดได้จากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก



ภาพที่ 4.9 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก

จากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก จะเห็นว่าในการวัดระยะการแก้ไขทั้ง 2 วิธี คือ Levenshtein Distance และ Needleman-Wunsch Algorithm ต่างก็ใช้เวลาในการประมวลผลเพื่อหาค่าระยะการแก้ไขแบบมีการตัดแบ่งคำใกล้เคียงกัน เช่นเดียวกันกับการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวสั้นและความยาวปานกลาง ที่ได้ทดสอบมาก่อนหน้านี้ โดยสังเกตได้จากแผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทยความยาวมาก โดยที่ Levenstein Distance ใช้ระยะเวลาประมวลผลเฉลี่ยเท่ากับ 66.35 ms. ในขณะที่ Needleman-Wunsch Algorithm ใช้เวลาประมวลผลเฉลี่ยเท่ากับ 65.93 ms.

โดยหลังจากการทดสอบการวัดระยะการแก้ไขจากข้อความที่มีความยาวแตกต่างกัน พบว่าทั้งสองวิธีการ ทั้ง Levenshtein Distance และ Needleman-Wunsch Algorithm ต่างก็ใช้เวลาในการประมวลผลที่ใกล้เคียงกัน แต่ปัจจัยที่มีผลต่อเวลาที่ใช้ในการประมวลผลนั้นจะมากหรือน้อยขึ้นอยู่กับความยาวของข้อความเป็นสำคัญ โดยที่ยิ่งข้อความมีความยาวมาก เวลาที่ใช้ในการประมวลผลก็จะมากตามไปด้วย

4.2.9 การทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทย

ในการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทย จะเป็นการทดสอบเพื่อหาค่าการวัดระยะการแก้ไขที่ได้ของชุดข้อความตัวอย่างที่เป็นข้อความในภาษาต่างๆ ที่ไม่ใช่ภาษาไทยนำมาเปรียบเทียบกัน โดยได้ใช้กลุ่มข้อมูลตัวอย่างบางส่วนดังเช่นตารางต่อไปนี้

ตารางที่ 4.7 ชุดข้อมูลตัวอย่างที่ใช้ในการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทย

ชุดข้อมูลที่	ข้อความที่ 1 (S_1)	ข้อความที่ 2 (S_2)
1	My car is white	Your house is green
2	I go to school every day	My school is close today
3	Guten morgen allerseits	Guten tag alle
4	добрый день	доброе утро
5	おやすみなさい	おはよう

4.2.10 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทย

ผลการทดลองที่ได้จากการนำกลุ่มข้อมูลตัวอย่างบางส่วนจากทั้งหมด ซึ่งได้แสดงในตารางข้างต้นมาทำการทดสอบผ่านโปรแกรมสำหรับคำนวณหาค่าการวัดระยะการแก้ไข ซึ่งได้ค่าที่วัดได้จากโปรแกรมดังต่อไปนี้

ชุดข้อความที่ 1: ข้อความภาษาอังกฤษ

"My car is white" เปรียบเทียบกับ "Your house is green"

ตารางที่ 4.8 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูลตัวอย่างชุดที่ 1

	ข้อความที่ 1	**ข้อความที่ 2**
ข้อความ	My car is white	Your house is green
จำนวนคำที่แบ่งได้	7	7
ลักษณะของคำที่แบ่งได้	My| |car| |is| |white|	Your| |house| |is| |green|
Hamming Distance		
ค่าระยะการแก้ไข	N/A	
เวลาประมวลผล	N/A	
Levenshtein Distance		
ระยะการแก้ไข (ไม่แบ่งคำ)	14	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	4|0|5|0|0|0|5|	
เวลาประมวลผล (แบ่งคำ)	74 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	14	
Needleman-Wunsch Algorithm		
ระยะการแก้ไข (ไม่แบ่งคำ)	3 (-3)	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	-2|2|-3|2|4|2|-3|	
เวลาประมวลผล (แบ่งคำ)	79 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	-2 (2)	

ชุดข้อความที่ 2: ข้อความภาษาอังกฤษ

"I go to school every day" เปรียบเทียบกับ "My school is close today"

ตารางที่ 4.9 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูลตัวอย่าง ชุดที่ 2

	ข้อความที่ 1	**ข้อความที่ 2**
ข้อความ	I go to school every day	My school is close today
จำนวนคำที่แบ่งได้	11	9
ลักษณะของคำที่แบ่งได้	I| |go| |to| |school| |every| |day|	My| |school| |is| |close| |today|
Hamming Distance		
ค่าระยะการแก้ไข	19	
เวลาประมวลผล	0 ms.	
Levenshtein Distance		
ระยะการแก้ไข (ไม่แบ่งคำ)	17	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	2|0|5|0|2|0|4|0|4|2|3|	
เวลาประมวลผล (แบ่งคำ)	74 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	22	
Needleman-Wunsch Algorithm		
ระยะการแก้ไข (ไม่แบ่งคำ)	-7 (7)	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	-1|2|-1|2|-2|2|1|2|-2|-1|0|	
เวลาประมวลผล (แบ่งคำ)	75 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	-2 (2)	

ชุดข้อความที่ 3: ข้อความภาษาเยอรมัน

"Guten morgen allerseits" เปรียบเทียบกับ "Guten tag alle"

ตารางที่ 4.10 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูลตัวอย่างชุดที่ 3

	ข้อความที่ 1	**ข้อความที่ 2**
ข้อความ	Guten morgen allerseits	Guten tag alle
จำนวนคำที่แบ่งได้	5	5
ลักษณะของคำที่แบ่งได้	Guten| |morgen| |allerseits|	Guten| |tag| |alle|
Hamming Distance		
ค่าระยะการแก้ไข	N/A	
เวลาประมวลผล	N/A	
Levenshtein Distance		
ระยะการแก้ไข (ไม่แบ่งคำ)	11	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	0|0|5|0|6|	
เวลาประมวลผล (แบ่งคำ)	82 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	11	
Needleman-Wunsch Algorithm		
ระยะการแก้ไข (ไม่แบ่งคำ)	-4 (4)	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	10|2|-3|2|-4|	
เวลาประมวลผล (แบ่งคำ)	80 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	-7 (7)	

ชุดข้อความที่ 4: ข้อความภาษารัสเซีย

"добрый день" เปรียบเทียบกับ "доброе утро"

ตารางที่ 4.11 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูลตัวอย่างชุดที่ 4

	ข้อความที่ 1	**ข้อความที่ 2**
ข้อความ	добрый день	доброе утро
จำนวนคำที่แบ่งได้	11	11
ลักษณะของคำที่แบ่งได้	?|?|?|?|?|?| |?|?|?|?|	?|?|?|?|?|?| |?|?|?|?|
Hamming Distance		
ค่าระยะการแก้ไข	6	
เวลาประมวลผล	0 ms.	
Levenshtein Distance		
ระยะการแก้ไข (ไม่แบ่งคำ)	6	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	0|0|0|0|0|0|0|0|0|0|0|	
เวลาประมวลผล (แบ่งคำ)	82 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	0	
Needleman-Wunsch Algorithm		
ระยะการแก้ไข (ไม่แบ่งคำ)	-4 (4)	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	2|2|2|2|2|2|2|2|2|2|2|	
เวลาประมวลผล (แบ่งคำ)	80 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	-22 (22)	

ชุดข้อความที่ 5: ข้อความภาษาญี่ปุ่น

"おやすみなさい" เปรียบเทียบกับ "おはよう"

ตารางที่ 4.12 ผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของข้อมูลตัวอย่างชุดที่ 5

	ข้อความที่ 1	**ข้อความที่ 2**
ข้อความ	おやすみなさい	おはよう
จำนวนคำที่แบ่งได้	7	4
ลักษณะของคำที่แบ่งได้	?|?|?|?|?|?|?|	?|?|?|?|
Hamming Distance		
ค่าระยะการแก้ไข	N/A	
เวลาประมวลผล	N/A	
Levenshtein Distance		
ระยะการแก้ไข (ไม่แบ่งคำ)	6	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	0|0|0|0|2|1|1|	
เวลาประมวลผล (แบ่งคำ)	77 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	4	
Needleman-Wunsch Algorithm		
ระยะการแก้ไข (ไม่แบ่งคำ)	4 (-4)	
เวลาประมวลผล (ไม่แบ่งคำ)	0 ms.	
ระยะการแก้ไข (แบ่งคำ)	2|2|2|2|-1|0|0|	
เวลาประมวลผล (แบ่งคำ)	75 ms.	
ผลรวมระยะการแก้ไข (แบ่งคำ)	-7 (7)	

จากตารางผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยข้างต้น จะเห็นว่าชนิดของภาษาไม่มีผลต่อการวัดระยะการแก้ไขแบบไม่แบ่งคำของทั้ง 3 วิธี โดยทั้ง Hamming Distance, Levenshtein Distance และ Needleman-Wunsch Algorithm แต่พบว่า ในการวัดระยะการแก้ไขแบบแบ่งคำจะเกิดปัญหาในกรณีที่ตัวอักษรในข้อความนั้นไม่ใช่ตัวอักษรในภาษาไทยหรือภาษาอังกฤษ เนื่องจากโปรแกรมตัดแบ่งคำไม่รองรับการอ่านตัวอักษรในภาษาอื่นที่ไม่ใช่ภาษาไทยและภาษาอังกฤษ ทำให้ผลลัพธ์ที่ได้กลับมามีความผิดเพี้ยน ดังที่แสดงในตารางผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของชุดข้อความที่ 4 และชุดข้อความที่ 5

4.3 การทดสอบการวัดระยะการแก้ไขข้อความกับ Character Set รูปแบบต่างๆ

ในการทดสอบการวัดระยะการแก้ไขกับข้อความนั้นนอกจากการทำการทดสอบข้างต้นแล้ว ยังได้ทำการทดสอบกับได้ทำการทดสอบข้อความกับ Character Set 3 รูปแบบด้วยกัน คือ

- TIS-620
- Windows-874
- Unicode (UTF-8)

โดยสำหรับการทดสอบการนำข้อความไปแปลงเป็น Character Set แบบต่างๆ ข้างต้น และนำไปทดสอบการวัดระยะการแก้ไขทั้งกับข้อความภาษาอังกฤษและภาษาไทยพบว่า ในการทดสอบกับข้อความภาษาอังกฤษ การแปลง Character Set ไม่มีผลต่อการตัดแบ่งคำและการวัดระยะการแก้ไข ซึ่งสามารถแสดงข้อความได้ถูกต้องดังเช่นข้อความต้นแบบ และสามารถนำไปตัดแบ่งคำและคำนวณหาค่าการวัดระยะการแก้ไขได้อย่างถูกต้องตามหลักการของแต่ละวิธีการ โดยได้ยกตัวอย่างผลลัพธ์จากการทดสอบบางส่วนมาแสดงให้เห็นดังตารางต่อไปนี้

ตารางที่ 4.13 ผลการแสดงผลและแบ่งคำของข้อความภาษาอังกฤษกับ Character Set แบบต่างๆ

ข้อความ	TIS-620	Windows-874	UTF-8
Hello World	Hello| |World|	Hello| |World|	Hello| |World|
My car is white	My| |car| |is| |white|	My| |car| |is| |white|	My| |car| |is| |white|
I am IT student	I| |am| |IT| |student|	I| |am| |IT| |student|	I| |am| |IT| |student|
Information Technology	Information| |Technology|	Information| |Technology|	Information| |Technology|
Implement of Edit Distance with Thai Language	Implement| |of| |Edit| |Distance| |with| |Thai| |Language|	Implement| |of| |Edit| |Distance| |with| |Thai| |Language|	Implement| |of| |Edit| |Distance| |with| |Thai| |Language|

สำหรับการทดสอบการนำชุดข้อความภาษาไทยมาแปลง Character Set เพื่อนำไปใช้ในการวัดระยะการแก้ไขนั้น ได้ผลแตกต่างกันตามแต่ละ Character Set โดยสรุปดังนี้

TIS-620

ในการแปลง Character Set ของข้อความตั้งต้นไปเป็น TIS-620 พบว่าข้อความที่แสดงผลออกมาหลังจากการนำไปตัดแบ่งคำ สามารถแสดงเป็นภาษาไทยได้อย่างถูกต้องดังเช่นข้อความต้นแบบ และสามารถนำไปตัดแบ่งคำ และคำนวณหาค่าการวัดระยะการแก้ไขได้อย่างถูกต้องตามหลักการของแต่ละวิธีการ

Windows-874

สำหรับการแปลง Character Set ของข้อความตั้งต้นไปเป็น Windows-874 นั้น พบว่าข้อความที่แสดงผลออกมาหลังจากการนำไปตัดแบ่งคำ สามารถแสดงเป็นภาษาไทยได้อย่างถูกต้องดังเช่นข้อความต้นแบบเช่นกัน และสามารถนำไปตัดแบ่งคำ และคำนวณหาค่าการวัดระยะการแก้ไขได้อย่างถูกต้องตามหลักการของแต่ละวิธีการ

Unicode (UTF-8)

ในการแปลง Character Set จากข้อความตั้งต้นไปเป็น Unicode พบว่าข้อความที่แสดงผลออกมาหลังจากการนำไปตัดแบ่งคำ แสดงออกมาเป็นตัวอักษรของภาษาอื่น ซึ่งไม่ตรงตามชุดข้อความต้นแบบ ซึ่งส่งผลให้การนำไปตัดแบ่งคำไม่ถูกต้อง และส่งผลให้การคำนวณหาค่าการวัดระยะการแก้ไขผิดเพี้ยนตามไปด้วย

ตารางที่ 4.14 ผลการแสดงผลและแบ่งคำของข้อความภาษาไทยกับ Character Set แบบต่างๆ

<table>
<tr><th>ข้อความ</th><th>TIS-620</th><th>Windows-874</th><th>UTF-8</th></tr>
<tr><td>วัวแก่กินหญ้าอ่อน</td><td>วัว|แก่|กิน|หญ้า|อ่อน|</td><td>วัว|แก่|กิน|หญ้า|อ่อน|</td><td>?|?|?|?|?|?|?|?|?|?|?|?|</td></tr>
<tr><td>หัวล้านได้หวี</td><td>หัวล้าน|ได้|หวี|</td><td>หัวล้าน|ได้|หวี|</td><td>?|?|?|?|?|?|?|?|?|?|?|</td></tr>
<tr><td>อ้อยเข้าปากช้าง</td><td>อ้อยเข้าปากช้าง|</td><td>อ้อยเข้าปากช้าง|</td><td>?|?|?|?|?|?|?|?|?|?|?|</td></tr>
<tr><td>เงียบเป็นเป่าสาก</td><td>เงียบเป็นเป่าสาก|</td><td>เงียบเป็นเป่าสาก|</td><td>?|?|?|?|?|?|?|?|?|?|</td></tr>
<tr><td>ยิงปืนนัดเดียวได้นก
สองตัว</td><td>ยิง|ปืน|นัด|เดียว|ได้|
นก|สอง|ตัว|</td><td>ยิง|ปืน|นัด|เดียว|ได้|
นก|สอง|ตัว|</td><td>?|?|?|?|?|?|?|?|?|?|?|?|?|?|?
|?|?|</td></tr>
</table>

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

บทนี้จะกล่าวถึง ผลสรุปของการวิจัยและข้อเสนอแนะของการวิจัยนี้ สำหรับผู้ที่มีความสนใจสามารถนำไปเป็นแนวทางการศึกษาในอนาคตต่อไป

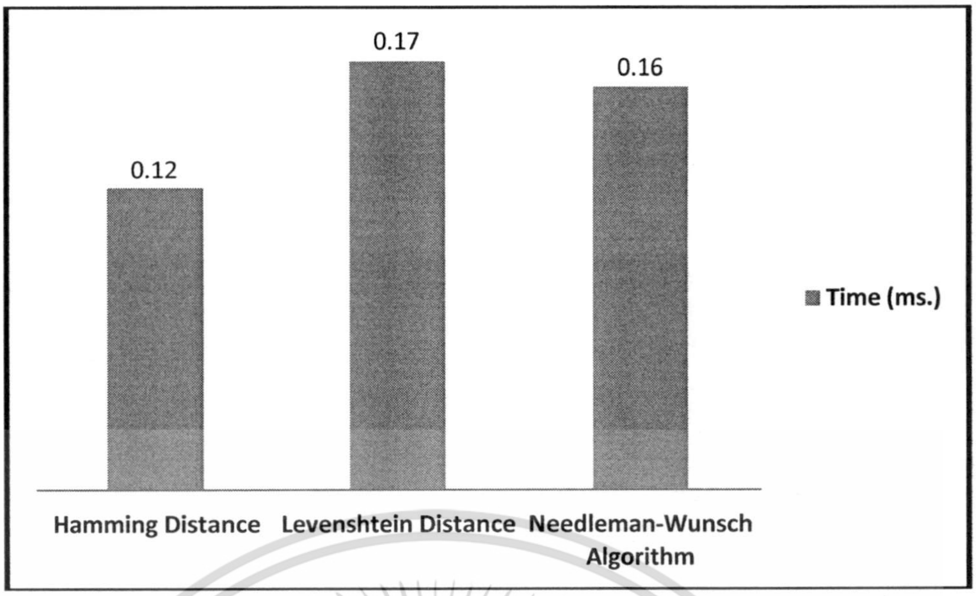
งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาศึกษาทฤษฎีการวัดระยะการแก้ไข (Edit Distance) และรูปแบบของการนำไปประยุกต์ใช้งานกับภาษาไทย ว่ามีความเป็นไปได้มากน้อยเพียงใด และวิธีการใดที่เหมาะสมแก่การนำไปประยุกต์ใช้กับภาษาไทย

5.1 สรุปผลการวิจัย

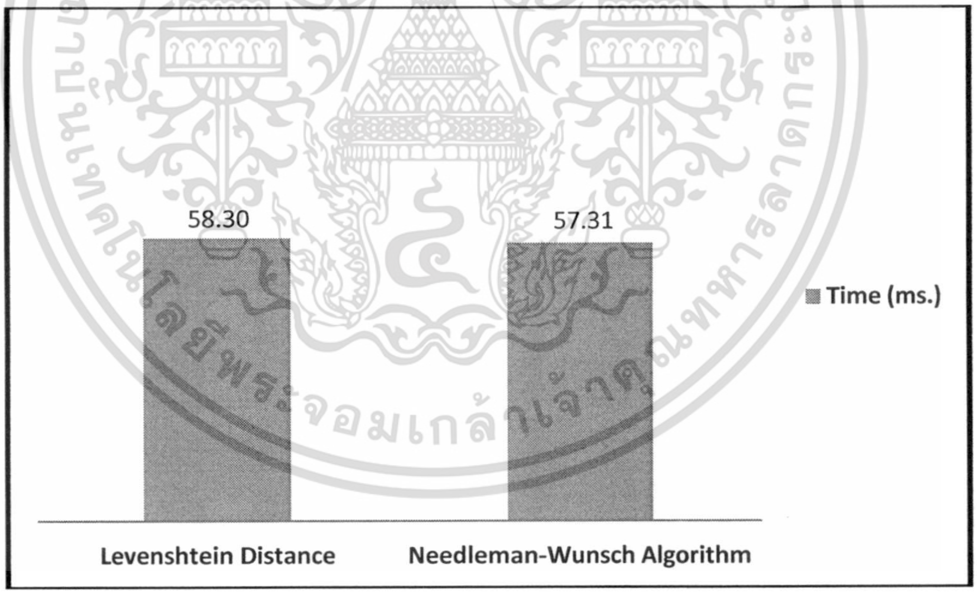
5.1.1 การวิเคราะห์ผลโดยใช้เวลาการประมาลผลเป็นเกณฑ์

จากการวิเคราะห์ผลการทดสอบการวัดระยะการแก้ไขโดยใช้เวลาการประมวลผลเป็นเกณฑ์ จะพบว่าวิธีการวัดระยะการแก้ไขทั้ง 3 วิธี คือ Hamming Distance, Levenshtein Distance และ Needleman-Wunsch Algorithm ต่างใช้เวลาประมวลผลใกล้เคียงกันทั้งสิ้น โดยที่การประมวลผลโดยไม่มีการตัดแบ่งคำ ทั้ง 3 วิธี ใช้เวลาในการประมวลผลน้อยมาก

สำหรับเวลาการประมวลผลแบบแบ่งคำนั้น จะสามารถนำมาใช้ได้เพียงแค่ 2 วิธี คือ Levenshtein Distance และ Needleman-Wunsch Algorithm เท่านั้น เนื่องจาก Hamming Distance นั้นมีข้อจำกัดในด้านการตัดแบ่งคำ ซึ่งจากการทดสอบพบว่า ทั้ง 2 วิธีที่นำมาทดสอบนั้นต่างก็ใช้เวลาในการประมวลผลไม่ต่างกันมาก ดังที่แสดงให้เห็นในแผนภูมิต่อไปนี้



ภาพที่ 5.1 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไข แบบไม่ตัดแบ่งคำ



ภาพที่ 5.2 แผนภูมิแสดงเวลาในการประมวลผลเฉลี่ยจากการทดสอบการวัดระยะการแก้ไข แบบไม่ตัดแบ่งคำ

จากแผนภูมิข้างต้นสามารถสรุปได้ว่า ถ้ามองในด้านปัจจัยของเวลาในการประมวลผล วิธีการทั้ง 3 วิธี มีความเหมาะสมในการมาประยุกต์ใช้กับภาษาไทย เนื่องจากใช้เวลาในการประมวลผลใกล้เคียงกัน

5.1.2 การวิเคราะห์ผลโดยใช้ค่าระยะการแก้ไขเป็นเกณฑ์

จากการวิเคราะห์ผลการทดสอบการวัดระยะการแก้ไขโดยใช้ค่าระยะการแก้ไขเป็นเกณฑ์ จะพบว่าวิธีการวัดระยะการแก้ไขทั้ง 3 วิธี คือ Hamming Distance, Levenshtein Distance และ Needleman-Wunsch Algorithm จะได้ค่าที่แตกต่างกันออกไป โดยที่ Hamming Distance จะเป็นการเปรียบเทียบกันที่ตำแหน่งแบบตัวต่อตัว ทำให้ค่าที่วัดได้จะสูงกว่า Levenstein Distance ในกรณีที่เป็นข้อความที่นำมาเปรียบเทียบมีจำนวนของอักขระเท่ากัน แต่มีอักขระสลับตำแหน่งไปมา เช่นเดียวกันกับ Needleman-Wunsch Algorithm ที่จะได้ค่าเป็นเป็นบวกในกรณีที่ตำแหน่งอักขระที่ตรงกัน และค่าเป็นลบในกรณีที่ตำแหน่งที่เปรียบเทียบกันเป็นอักขระคนละตัวกัน ซึ่งค่าที่ได้จะสามารถบอกได้เพียงว่าชุดข้อความที่นำมาเปรียบเทียบกันนั้นมีความแตกต่างกันมากน้อยแค่ไหน แต่ในขณะที่ Levenshten Distance จะบอกได้ว่าต่างกันมากน้อยแค่ไหน และต้องทำการแก้ไขกี่ครั้ง ข้อความทั้งสองจึงจะเหมือนกัน

สำหรับเวลาการประมวลผลแบบแบ่งคำนั้น จะสามารถนำมาใช้ได้เพียงแค่ 2 วิธี คือ Levenshtein Distance และ Needleman-Wunsch Algorithm เช่นเดียวกัน แต่ในกรณีที่ตัดแบ่งคำออกมา แล้วคำที่นำมาเปรียบเทียบกันมีจำนวนไม่เท่ากัน ค่าที่เปรียบเทียบได้จากการนำคำที่มีอยู่ไปเปรียบเทียบกับค่าว่าง (ไม่มีคำ) จะได้ค่าเป็น 0 ซึ่งเป็นค่าที่ไม่ถูกต้อง ดังดูตัวอย่างได้จากตารางผลการทดสอบการวัดระยะการแก้ไขกับข้อความที่ไม่ใช่ภาษาไทยของชุดข้อความที่ 5 เป็นต้น

5.2 อภิปรายผล

จากการนำวิธีการวัดระยะการแก้ไขวิธีการต่างๆ มาทดสอบ และลองประยุกต์ใช้กับภาษาไทย ไม่ว่าจะเป็นวิธี Hamming Distance, Levenshtein Distance และ Needleman-Wunsch Algorithm พบว่า ในการประยุกต์ใช้การวัดระยะการแก้ไขกับข้อความภาษาไทยนั้น Levenshtein Distance ถือเป็นวิธีที่เหมาะสมกับข้อความภาษาไทย เนื่องจากไม่มีปัญหาเรื่องค่าที่คลาดเคลื่อนจากการตัดแบ่งคำที่ได้จำนวนของคำไม่เท่ากัน และยังได้ค่าที่ตรงตัว โดยไม่ต้องไปแปลผลของค่าที่ได้ ซึ่งจะต่างจาก Needleman-Wunsch Algorithm ที่เป็นวิธีในการวัดหาค่าความเหมือนกันของข้อความ โดยค่าที่ได้จากวิธีนี้จะบอกได้แค่ว่าข้อความทั้งสองที่นำมาเปรียบเทียบกันมีความเหมือนกันมากหรือน้อยเพียงใด ขึ้นอยู่กับค่าที่วัดได้ว่าเป็นบวกหรือลบ

สำหรับ Needleman-Wunsch Algorithm นั้นจะเหมาะกับการวัดระยะการแก้ไขข้อความในลักษณะของคำมากกว่าจะมาวัดแบบทั้งประโยค เพราะหากมีการตัดแบ่งคำแล้ว จำนวนคำของข้อความทั้งสองที่นำมาเปรียบเทียบกันไม่เท่ากัน ค่าที่ได้จาก Needleman-Wunsch Algorithm จะมีความคลาดเคลื่อนไปด้วย ทำให้ผลลัพธ์ที่ออกมาไม่ถูกต้อง

5.3 ข้อเสนอแนะ

ในการศึกษาครั้งนี้ได้มีการวิเคราะห์ถึงแนวทางในการนำทฤษฎีการวัดระยะการแก้ไขมาประยุกต์ใช้ในภาษาไทย ซึ่งยังไม่ค่อยมีผู้ศึกษาทฤษฎีนี้ในประเทศไทยมากนัก ซึ่งหากว่ามีการศึกษาทฤษฎีการวัดระยะการแก้ไขและนำมาประยุกต์ใช้กับภาษาไทยได้อย่างสมบูรณ์ จะทำให้ภาษาไทยสามารถใช้งานในระบบคอมพิวเตอร์อย่างมีประโยชน์มากยิ่งขึ้นเหมือนดังภาษาอื่นๆ ที่มีการนำทฤษฎีนี้ไปใช้ และนำมาพัฒนาใช้ในซอฟต์แวร์ต่างๆ เช่น การทำระบบตรวจสอบคำผิดในโปรแกรมประเภท Word Processing เป็นต้น ซึ่งหากมีการนำไปศึกษาและมีการพัฒนาอย่างจริงจังคาดว่าจะเป็นประโยชน์ต่อคนไทยที่ใช้คอมพิวเตอร์จำนวนไม่น้อย

บรรณานุกรม

โครงการสารานุกรมไทยฯ. ม.ป.ป. **โปรแกรมตัดคำภาษาไทย (Thai Word Segmentation).** [ออนไลน์]. เข้าถึงได้จาก http://kanchanapisek.or.th/kp6/sub/book/book.php?book=25&chap=7&page=t25-7-infodetail05.html

สำนักงานมาตรฐานผลิตภัณฑ์อุตสาหกรรม. ม.ป.ป. **รหัสสำหรับอักขระไทยที่ใช้กับคอมพิวเตอร์.** [ออนไลน์]. เข้าถึงได้จาก http://www.nectec.or.th/it-standards/std620/std620.htm

Bing Liu and Yanhong Zhai. 2005. "NET - A System for Extracting Web Data from Flat and Nested Data Records." 487-495. In **6th International Conference on Web Information Systems Engineering,** New York: Springer Berlin Heidelberg

Dan Gusfield. 1997. **Algorithms on strings, trees, and sequences: computer science and computational biology.** Cambridge, UK: Cambridge University Press.

Dan Jurafsky. n.p. **Minimum Edit Distance.** [Online] Available: www.stanford.edu/class/cs124/lec/med.pdf

Department of Electrical and Computer Engineering - University of New Brunswick. n.p. **Error Correction and the Hamming Code.** [Online] Available: http://www.ee.unb.ca/tervo/ee4253/hamming.shtml

Devi de Castro Reis, Paulo B. Golgher, Altigran S. da Silva and Alberto H. F. Laender. 2004. "Automatic Web News Extraction Using Tree Edit Distance." 502-511. In **WWW '04 Proceedings of the 13th international conference on World Wide Web,** New York: AMC.

Lopresti, **System and Method for Handwriting Matching Using Edit Distance Computation in a Systolic Array Processor.** U.S. patent no. 5757959, May 1998.

Levenshtein VI. 1966. **Binary codes capable of correcting deletions, insertions, and reversals.** n.p.

Marco Bertini, Alberto Del Bimbo and Walter Nunziati. 2006. "Video Clip Matching Using MPEG-7 Descriptors and Edit Distance." 133-142. In **5th International Conference, CIVR 2006**, New York: Springer Berlin Heidelberg

Marius Gheorghescu. 2006. "AN AUTOMATED VIRUS CLASSIFICATION SYSTEM." In **Virus Bulletin Conference October 2005**, Washington: Virus Bulletin.

Paisarn Charoenpornsawat. n.p. **SWATH - Thai Word Segmentation.** [Online] Available: http://www.cs.cmu.edu/~paisarn/software.html

Philip N. Klein, Thomas B. Sebastian and Benjamin B. Kimia. 2001. "Shape matching using edit-distance: an implementation." 781-790. In **SODA '01 Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms**, Philadelphia: SIAM.

S. B. Needleman and C. D. Wunsch. 1969. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." **Department of Biochemistry, Northwestern University.**

Srikanta Tirthapura, Daniel Sharvit, Philip Klein and Benjamin B. Kimia. 1998. "Indexing based on edit-distance matching of shape graphs." In **Multimedia Storage and Archiving Systems III**, Boston: SPIE

The Unicode Consortium. n.p. **Unicode Standard.** [Online] Available: http://www.unicode.org/standard/standard.html

ภาคผนวก

ภาคผนวก ก

บทความตีพิมพ์ในการประชุมทางวิชาการระดับชาติ
ด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 10
เรื่อง การประยุกต์ใช้การวัดระยะการแก้ไขกับข้อความภาษาไทย

การประยุกต์ใช้การวัดระยะการแก้ไขกับข้อความภาษาไทย
Implementation of edit distance with Thai language

เมธาวี จินดารักษ์ (Metawee Jindarak)[1] และสุภกิจ นุตยะสกุล (Supakit Nootyaskool)[2]
สาขาวิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
[1]metawee.j@gmail.com, [2]supakit@it.kmitl.ac.th

บทคัดย่อ

งานวิจัยชิ้นนี้มีจุดประสงค์เพื่อศึกษาวิธีการในการวัดระยะการแก้ไข (Edit Distance) และพิสูจน์หาวิธีการที่เหมาะสมในการนำวิธีดังกล่าวมาประยุกต์ใช้กับภาษาไทย โดยได้ใช้วิธีการวัดระยะการแก้ไขมาประยุกต์ใช้ร่วมกับวิธีการตัดคำสำหรับภาษาไทย โดยงานวิจัยชิ้นนี้ได้นำ Levenshtein Distance ซึ่งเป็นวิธี Edit Distance ที่เป็นที่นิยม ไปทำการทดสอบเปรียบเทียบกับวิธี Needleman-Wunsch Algorithm และ Hamming Distance เพื่อเปรียบเทียบว่าวิธีการดังกล่าวเหมาะสำหรับการประยุกต์ใช้กับภาษาไทยหรือไม่ ซึ่งผลที่ได้จากการทดสอบนั้น ได้แสดงให้เห็นว่า Levenshtein Distance เป็นวิธีมีความเหมาะสมที่จะนำมาประยุกต์ใช้กับภาษาไทยเมื่อเทียบกับวิธีการอื่นๆ ที่นำมาเปรียบเทียบกัน เนื่องจาก Levenshtein Distance จะบอกได้ถึงจำนวนครั้งของการเปลี่ยนแปลงสำหรับการแก้ไขข้อความ ซึ่งต่างจาก Needleman-Wunsch Algorithm ที่บอกได้แค่ความเหมือนหรือต่างกันเพียงใด และ Levenshtein Distance ยังมีข้อจำกัดในการใช้งานน้อยกว่า Hamming Distance ด้วย

คำสำคัญ: การวัดระยะการแก้ไข, การเปรียบเทียบข้อความภาษาไทย, ระยะทางเลเวนชเตย์น

Abstract

This research objective is to prove about how to implement edit distance to suit for Thai language. The research experiment is using Thai word segmentation method combine with various edit distance method such as Levenshtein Distance, Needleman-Wunsch Algorithm and Hamming Distance to find out what is the most suitable edit distance method for using with Thai language. The result of the experiment shows that Levenshtein Distance is most suitable for implement with Thai language when comparing with other two edit distance methods because Levenshtein Distance can show the value of difference between two sentences that Needleman-Wunch Algorithm and Hamming Distance cannot show that value.

Keyword: Edit Distance, Thai Sentence Comparison, Levenshtein Distance

1. บทนำ

ในการเปรียบเทียบความแตกต่างของข้อความ และรูปภาพหรือรูปทรงต่างๆ ในทางคอมพิวเตอร์นั้น มีวิธีการที่นำมาใช้ในการเปรียบเทียบได้หลายวิธี ซึ่งวิธีการหนึ่งที่ถูกนำมาใช้ คือการวัดระยะการแก้ไข โดยการวัดระยะการแก้ไขนั้น ก็มีผู้คิดค้นอัลกอริทึมที่แตกต่างกันออกไปหลายวิธี โดยแต่ละวิธีจะมีวิธีการในการวัดระยะ และข้อจำกัดในการวัดระยะที่แตกต่างกัน เช่น การวัดระยะด้วยวิธี Hamming Distance จะเป็นการวัดระยะการแก้ไขข้อความที่มีจำนวนตัวอักษรเท่ากัน และเปรียบเทียบเฉพาะตัวอักษรที่ตำแหน่งที่ตรงกันเท่านั้น ซึ่งวิธีนี้ใช้ในการทำ Hamming Code สำหรับแก้ไขข้อมูลที่ผิดพลาด (Error Correction) จากการส่งข้อมูลต้นทางไปยังปลายทางในระบบเครือข่าย[1] เป็นต้น แต่ถ้าจะกล่าวถึงวิธีการวัดระยะการแก้ไขที่เป็นที่นิยม และรู้จักแพร่หลาย คือ Levenshtein Distance ซึ่งเป็นวิธีการที่ใช้ในการวัดจำนวนครั้งของการแก้ไขข้อความจากคำหนึ่งไปเป็นอีกคำหนึ่ง ด้วยการนับจำนวนครั้งของการ

แทรก การตัดออก และการแทนที่ตัวอักษร ในข้อความนั้นๆ ว่าใช้จำนวนครั้งเท่าใดในการแก้ไขข้อความ

วิธีการวัดระยะการแก้ไขนั้นมักถูกนำไปใช้ในการเปรียบเทียบหาความแตกต่างระหว่างสิ่งของสองสิ่ง ซึ่งได้มีผู้นำทฤษฎีดังกล่าวไปทำการวิจัยในด้านต่างๆ มากมาย เช่น การนำไปประยุกต์ใช้ในการจับคู่รูปทรงที่มีลักษณะเหมือนกันหรือใกล้เคียงกันโดยอาศัยหลักการ Shock Graph ที่นำเสนอโดย Philip N. Klein และคณะ [2] หรือการจำแนกหมวดหมู่ไวรัสคอมพิวเตอร์โดยอัตโนมัติ ด้วยการนำโครงสร้างหลักของชุดคำสั่งของไวรัสมาจำแนกด้วยวิธีการวัดระยะการแก้ไข ซึ่งคิดค้นขึ้นโดย Marius Gheorghescu จากบริษัทไมโครซอฟท์ [3] หรือการเปรียบเทียบความแตกต่างของวิดีโอคลิปด้วยการใช้ MPEG-7 Descriptor และวิธีการวัดระยะการแก้ไขที่นำเสนอโดย Marco Bertini และคณะ จากมหาวิทยาลัยฟลอเรนซ์ [4] เป็นต้น

2. การวัดระยะการแก้ไขข้อความ (Edit Distance)

การวัดระยะการแก้ไข (Edit Distance) เป็นวิธีการในการวัดหาค่าความต่างกันระหว่างชุดตัวอักษรสองชุด ระหว่างชุดแรกที่เป็นชุดต้นแบบ และชุดที่สองที่เป็นชุดเปรียบเทียบ โดยค่าความแตกต่างจะวัดจากจำนวนครั้งของการที่จะต้องทำการแทรก ตัดออก และแทนที่ตัวอักษรในชุดตัวอักษรที่นำมาเปรียบเทียบจนกระทั่งชุดตัวอักษรที่นำมาเปรียบเทียบ (ชุดที่สอง) มีลักษณะเหมือนกับตัวอักษรชุดต้นแบบ (ชุดแรก) ทุกประการ ซึ่งโดยทั่วไปการวัดระยะการแก้ไขจะอ้างอิงถึงระยะทางเลเวนชเตย์น (Levenshtein Distance) ซึ่งเป็นทฤษฎีที่เกี่ยวข้องกับการวัดระยะการแก้ไขที่ถูกตั้งชื่อตาม Vladimir Levenshtein ผู้ที่ปรับปรุงชุดอัลกอริทึมชุดนี้ [5]

2.1 วิธีการของ Levenshtein Distance

ขั้นตอนวิธีการ Levenshtein Distance จะเป็นการนำชุดตัวอักษร 2 ชุด มาเปรียบเทียบจำนวนความแตกต่างกัน โดยจะพิจารณาจาก 3 รูปแบบ คือ

การแทรก : เป็นการนำเอาตัวอักษรตัวใดๆ มา เพื่อให้ชุดตัวอักษรชุดนั้นเหมือนกับอีกชุดตัวอักษรหนึ่งในภายหลัง เช่น MAN เป็น MAIN จะแทรกตัว I ให้กับ MAN เพื่อให้ MAN กลายเป็น MAIN เป็นต้น

การตัดออก : เป็นการตัดตัวอักษรออกครั้งละ 1 ตัว จากชุดตัวอักษรชุดหนึ่ง เพื่อให้ชุดตัวอักษรชุดนั้นเหมือนกับอีกชุดตัวอักษรหนึ่งในภายหลัง เช่น GOD เป็น GO จะตัดตัว D ออก เพื่อให้ GOD กลายเป็น GO เป็นต้น

การแทนที่ : เป็นการนำตัวอักษรของชุดตัวอักษรหนึ่งไปแทนตัวอักษรของอีกชุดตัวอักษรหนึ่ง เพื่อให้ชุดตัวอักษรชุดนั้นเหมือนกับอีกชุดตัวอักษรหนึ่งในภายหลัง เช่น CHAT เป็น THAT จะแทนที่ C ด้วย T เพื่อให้ CHAT กลายเป็น THAT หรือ อาจมองในทางกลับกันก็ได้ เป็นต้น [6]

2.2 ตัวอย่างของ Levenshtein Distance

ตารางที่ 1: ตารางการคำนวณหาค่าระยะการแก้ไขระหว่างคำว่า ENGLISH และ THAI

	#	T	H	A	I
#	0	1	2	3	4
E	1	1	2	3	4
N	2	2	2	3	4
G	3	3	3	3	4
L	4	4	4	4	4
I	5	5	5	5	4
S	6	6	6	6	5
H	7	7	6	7	6

จากตัวอย่างจะเห็นว่า คำว่า ENGLISH และคำว่า THAI มีค่าความต่างกันอยู่คือ 6 โดยได้มาจาก

- การตัดอักษร G, S, H ของคำว่า ENGLISH ออก โดยดำเนินการ 3 ครั้ง
- การแทนที่อักษร E ด้วยอักษร T, อักษร N ด้วยอักษร H และอักษร L ด้วยอักษร A โดยดำเนินการ 3 ครั้ง

ดังนั้นจะเห็นว่า จากการแก้ไขคำว่า ENGLISH ให้เป็นคำว่า THAI จะต้องต้องดำเนินการตัดอักษร 3 ครั้ง และแทนที่ตัวอักษรด้วยตัวอักษรอื่นอีก 3 ครั้ง รวมเป็นการดำเนินการทั้งหมด 6 ครั้ง จึงหมายความว่า ทั้งสองคำมีค่าความแตกต่างกันอยู่เท่ากับ 6

2.3 รหัสเทียมของ Levenshtein Distance

```
int LevenshteinDistance(char s[1..m], char t[1..n])
declare int d[0..m, 0..n]

for i from 0 to m
   d[i, 0] := i
for j from 0 to n
   d[0, j] := j

for j from 1 to n
  for i from 1 to m
     if s[i] = t[j] then
        d[i,j] := d[i-1,j-1]
        else
           d[i,j] := minimum
           (
              d[i-1, j] + 1        // การตัดออก
              d[i, j-1] + 1,       // การแทรก
              d[i-1, j-1] + 1      // การแทนที่
           )
return d[m,n]
```

ภาพที่ 1: รหัสเทียมของ Levenshtien Distance

3. การประยุกต์ใช้การวัดระยะการแก้ไขกับข้อความภาษาไทย

ภาษาไทยเป็นภาษามาตรฐานภาษาหนึ่งที่ใช้งานบนคอมพิวเตอร์ โดยภาษาไทยนั้นมีตัวอักขระที่ใช้งานบนคอมพิวเตอร์ทั้งสิ้น 87 ตัว และมีรหัสสำหรับอักขระไทยที่ใช้บนคอมพิวเตอร์ คือ TIS-620 ซึ่งเป็นรหัสสำหรับอักขระไทยในตารางรหัส EBCDIC (Extended Binary Coded Decimal Interchange Code) ที่มีขนาด 8 บิต ซึ่งภายหลังยังได้มีการพัฒนาไปใช้ในรหัส Unicode ด้วย [7]

ตารางที่ 2: ตารางแสดงรหัสภาษาไทยสำหรับคอมพิวเตอร์ TIS-620

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	ไม่ได้ใช้															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	ไม่ได้ใช้															
9x																
Ax		ก	ข	ฃ	ค	ฅ	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ญ	ฎ	ฏ
Bx	ฐ	ฑ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ
Cx	ภ	ม	ย	ร	ฤ	ล	ฦ	ว	ศ	ษ	ส	ห	ฬ	อ	ฮ	ฯ
Dx	ะ	ั	า	ำ	ิ	ี	ึ	ื	ุ	ู	ฺ					฿
Ex	เ	แ	โ	ใ	ไ	ๅ	ๆ	็	่	้	๊	๋	์	ํ	๎	๏
Fx	0	1	2	3	4	5	6	7	8	9	๚	๛				

3.1 ปัญหาของการใช้ Edit Distance กับภาษาไทย

เนื่องด้วยภาษาไทยนั้นจะมีลักษณะของข้อความที่มีตัวอักษรเขียนติดกันทั้งหมด และเมื่อจบประโยคจึงมีการเว้นวรรค ซึ่งไม่มีโครงสร้างของคำและประโยคที่ชัดเจนแบบภาษาอังกฤษที่จะมีการเว้นวรรคเพื่อแบ่งคำ และมีการใส่จุด (.) เมื่อสิ้นสุดประโยค ทำให้ภาษาไทยนั้นไม่สามารถที่จะนำข้อความทั้งสองมาเปรียบเทียบกันได้ เพราะไม่มีการแบ่งขอบเขตของคำที่ชัดเจน เพราะหากไม่มีการแบ่งขอบเขตของคำค่าในการเปรียบเทียบจะเกิดความคลาดเคลื่อน เนื่องจากระบบจะไม่สามารถจำแนกได้ว่าส่วนใดเป็นคำที่ผิด หรือส่วนใดถูกต้อง ทำให้ต้องมีการตัดแบ่งคำในข้อความออกมาเป็นชุดของคำก่อนที่จะมาทำการวัดระยะการแก้ไขของข้อความ ซึ่งวิธีในการตัดคำภาษาไทยนั้นก็สามารถทำได้หลายวิธี

3.2 รูปแบบของการตัดคำภาษาไทย

การตัดคำแบบยาวที่สุด (Longest Matching) : การตัดคำวิธีนี้ทำโดยเริ่มจากตัวอักษรซ้ายสุดของข้อความไปยังตัวอักษรถัดไป จนกว่าจะพบคำที่มีอยู่ในพจนานุกรม หลังจากนั้นก็ค้นหาคำถัดไปจนกว่าจะจบข้อความ หากพบว่าเป็นคำในพจนานุกรมจากจุดเริ่มต้นเดียวกัน จะเลือกเอาคำที่ยาวที่สุด เช่น การแบ่งคำในประโยค "คนตากลมนอนตากลม" จะเริ่มจากตัวอักษร ค แล้วค้นหาตัวอักษรถัดไปและนำมาเปรียบเทียบคำในพจนานุกรม โดยคำแรกที่แบ่งได้คือ "คน" และตัวอักษรถัดไปคือ ต จากตัวอักษรนี้จะได้คำว่า "ตา" กับคำว่า "ตาก" ซึ่งวิธีนี้ให้เลือกคำที่ยาวที่สุดที่ค้นพบจึงเลือกคำว่า "ตาก" จากนั้นก็จะค้นหาและเปรียบเทียบต่อไป ซึ่งผลลัพธ์ที่ออกมาคือ "คน ตาก ลม นอน ตาก ลม"

การตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching) : เป็นการหาวิธีในการตัดคำที่เป็นไปได้ทั้งหมด เช่น ข้อความว่า "ไปหามเหสี" ซึ่งสามารถตัดคำได้ 2 แบบ คือ “ไป หาม เห สี” และ “ไป หา มเหสี” โดยที่วิธีการนี้จะเลือกเอาข้อความที่แบ่งแล้วได้จำนวนคำน้อยที่สุด คือ “ไป หา มเหสี” ซึ่งมีจำนวนคำที่ตัดได้ 3 คำ ในขณะที่ “ไป หาม เห สี” มี 4 คำ

การตัดคำแบบคำนวณเชิงสถิติเพื่อหาความเป็นไปได้ (Probabilistic Model) : วิธีนี้นำเอาค่าสถิติการเกิดของคำและลำดับของหน้าที่ของคำ (Part of speech) มาช่วยในการคำนวณหาความน่าจะเป็น เพื่อใช้เลือกแบบที่มีโอกาสการเกิดมากที่สุด แต่วิธีนี้มีข้อจำกัด คือ ต้องมีฐานข้อมูลที่มีการตัดคำที่ถูกต้อง และต้องกำหนดหน้าที่ของคำ เพื่อที่จะได้นำไปใช้ในการสร้างสถิติ

การตัดคำแบบใช้คุณลักษณะ (Feature-Based Approach) : วิธีการนี้จะพิจารณาจากบริบทของคำ (Context words) และการเกิดร่วมกันหรือหน้าที่ของคำ (Collocation) เข้ามาช่วยในการตัดคำ เช่น "ตากลม" ถ้าพบคำว่า "โต" ในบริบทก็จะสามารถตัดคำได้ว่า "ตา กลม" ซึ่งวิธีนี้จำเป็นที่จะต้องมีฐานข้อมูลเป็นจำนวนมาก และจะต้องมีการเรียนรู้การสร้างคำในบริบทหรือการเกิดร่วมกันของคำแต่ละคำ เพื่อให้มีข้อมูลที่จะนำมาใช้ในการตัดคำ [8]

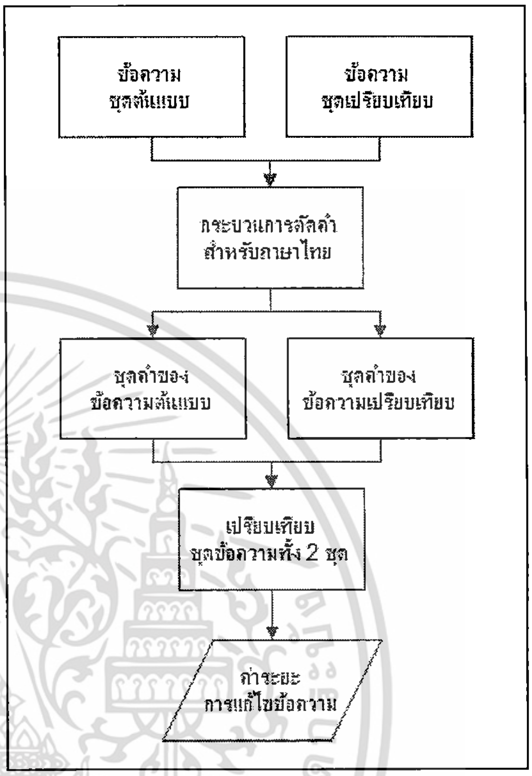
4. การทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย

สำหรับการทดสอบการวัดระยะการแก้ไขข้อความด้วยวิธี Edit Distance กับข้อความภาษาไทยนั้น ได้มีการทดสอบโดยใช้กลุ่มของข้อความภาษาไทยตัวอย่างจำนวนหนึ่ง มาทำการทดสอบบนโปรแกรมที่ถูกพัฒนาขึ้น เพื่อคำนวณหาระยะในการแก้ไขข้อความสำหรับภาษาไทย เพื่อเปรียบเทียบหาวิธีการวัดระยะที่เหมาะสมสำหรับข้อความภาษาไทย ซึ่งได้นำวิธีต่างๆ มาใช้ในการเปรียบเทียบกัน 3 วิธี คือ Levenshtein Distance, Hamming Distance และ Needleman-Wunsch Algorithm [9] โดยที่โปรแกรมจะคำนวณหาค่าระยะการแก้ไขข้อความ จำนวนคำที่แบ่งได้ รูปแบบของคำที่แบ่ง รวมไปถึงระยะเวลาที่ใช้ในการคำนวณ

การทำงานของโปรแกรมนั้นจะแบ่งขั้นตอนหลักๆภายในโปรแกรมได้ 2 ส่วน คือ ขั้นตอนในการตัดแบ่งข้อความออกเป็นคำ และขั้นตอนในการนำคำแต่ละคำของทั้งสองข้อความมาทำการเปรียบเทียบกันด้วยวิธี Edit Distance

ขั้นตอนในการตัดแบ่งข้อความออกเป็นคำนั้นได้ใช้โปรแกรม SWATh (Smart Word Analysis for Thai) ซึ่งเป็นโปรแกรมสำหรับตัดคำภาษาไทยที่พัฒนาขึ้นโดย ไพศาล เจริญพรสวัสดิ์ [10] โดยที่โปรแกรมดังกล่าวจะใช้ลักษณะการตัดคำภาษาไทยหลายแบบผสมกัน ได้แก่ การตัดคำแบบยาวที่สุด (Longest Matching) และการตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching) ร่วมกับการใช้ค่าสถิติการเกิดของคำและลำดับของหน้าที่ของคำ (Part of speech) เข้ามาช่วยในการตัดคำ

สำหรับขั้นตอนในการนำคำแต่ละคำของทั้งสองข้อความมาทำการเปรียบเทียบกันนั้น จะเป็นการนำคำแต่ละคำของข้อความทั้งสองมาเปรียบเทียบกันแบบคำต่อคำ และเรียงตามลำดับ เพื่อหาค่าในการแก้ไขข้อความ โดยการนำคำที่หนึ่งของข้อความที่หนึ่ง มาเปรียบเทียบกับคำที่หนึ่งของข้อความที่สอง และทำเช่นนี้ต่อไปจนกระทั่งถึงคำสุดท้ายของข้อความ



ภาพที่ 2: กระบวนการในการวัดระยะแก้ไขข้อความภาษาไทย

4.1 วิธีการทดสอบการวัดระยะการแก้ไขข้อความภาษาไทย

ในการทดสอบการวัดระยะการแก้ไขข้อความภาษาไทยนั้น จะใช้วิธีการต่างกัน 3 วิธี มาใช้ในการวัดหาระยะการแก้ไข โดยแต่ละวิธีนั้นจะมีลักษณะและข้อจำกัดที่แตกต่างกันดังนี้

- Levenshtein Distance เป็นการวัดหาค่าความแตกต่างระหว่างข้อความสองชุด โดยนับจำนวนครั้งของการแทรก ตัดออก และแทนที่ตัวอักษร ซึ่งในการกระทำดังกล่าวแต่ละครั้งคะแนนในการวัดค่าจะเท่ากับ +1 ซึ่งผลคะแนนสุดท้ายที่วัดได้จะเป็นผลรวมของจำนวนครั้งที่ใช้ในการแก้ไขข้อความ

- Hamming Distance เป็นการวัดหาค่าความแตกต่างระหว่างข้อความสองชุด โดยที่ข้อความทั้งสองต้องมีจำนวนตัวอักษรที่เท่ากัน โดยการเปรียบเทียบจะเปรียบเทียบแบบตัวอักษรต่อตัวอักษร จากตัวอักษรตัวแรกจนถึงตัวอักษรตัวสุดท้าย หากพบว่าตัวอักษรต่างกันคะแนนในการวัดค่าจะเท่า

+1 ซึ่งผลคะแนนสุดท้ายที่วัดได้ จะเป็นผลรวมของจำนวนตัวอักษรที่มีความแตกต่างกันทั้งหมด

- Needleman-Wunsch Algorithm เป็นการวัดหาค่าความเหมือนระหว่างข้อความสองชุด โดยเป็นการนำคำของข้อความทั้งสองมาหาค่าที่ใช้ในการจัดเรียงตัวอักษร โดยที่เมื่อนำคำมาเปรียบเทียบกันแล้วตำแหน่งที่เปรียบเทียบกันเป็นเป็นตัวอักษรทั้งคู่และเป็นตัวอักษรตัวเดียวกัน ค่าที่ได้จะเป็น +2 แต่ถ้าเป็นตัวอักษรคนละตัวกัน ค่าที่ได้จะเป็น -1 และหากตำแหน่งที่เปรียบเทียบเป็นช่องว่างกับตัวอักษร ค่าที่ได้จะเป็น -2

4.2 ผลการทดสอบการวัดระยะการแก้ไขข้อความภาษาไทย

ในการทดสอบการวัดระยะแก้ไขข้อความภาษาไทยได้ใช้ชุดข้อความภาษาไทยตัวอย่าง นำมาเปรียบเทียบกันเป็นคู่ จำนวน 5 ชุดข้อความ ดังนี้

ชุดที่ 1: "การเรียน" เปรียบเทียบกับ "การเกษตร"

ชุดที่ 2: "ตกกระไดพลอยโจน" เปรียบเทียบกับ "ตีวัวกระทบคลาด"

ชุดที่ 3: "รู้หลบเป็นปีก รู้หลีกเป็นหาง" เปรียบเทียบกับ "รักดีหามจั่ว รักชั่วหามเสา"

ชุดที่ 4: "ผีแดงแห้วชอว์แบะท่าไปเชลซี" เปรียบเทียบกับ "ผีคว้าชัยรับขวัญมาตาเปิดรังทุบคาร์ดิฟฟ์"

ชุดที่ 5: "สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง" เปรียบเทียบกับ "มหาวิทยาลัยเทคโนโลยีพระจอมเกล้า พระนครเหนือ"

โดยผลการทดสอบจากการเปรียบเทียบชุดข้อความตัวอย่างข้างต้น จะได้ค่าจากการเปรียบเทียบตามตารางดังต่อไปนี้

ตารางที่ 3: ตารางแสดงผลการทดสอบการวัดระยะการแก้ไขข้อความภาษาไทยด้วยวิธีการต่างๆ

ชุดข้อความ	Hamming		Levenshtein			Needleman-Wunsch		
	Score A	Time	Score A	Score B	Time	Score A	Score B	Time
1	4	1	4	4	83	4	4	76
2	13	1	12	25	85	-5	-4	76
3	N/A	N/A	24	26	97	-18	-14	88
4	N/A	N/A	34	37	98	-22	-15	81
5	N/A	N/A	29	53	82	13	-30	84

Score A : ค่าที่ได้จากการวัดระยะการแก้ไข โดยไม่ตัดแบ่งคำ
Score B : ค่าที่ได้จากการวัดระยะการแก้ไข โดยมีการตัดแบ่งคำ
Time : ระยะเวลาที่ใช้ในการคำนวณหาค่าในการวัดระยการแก้ไข (Millisecond)

จากตารางแสดงผลการทดสอบการวัดระยะการแก้ไขข้อความภาษาไทยข้างต้น จะเห็นว่าในการวัดระยะการแก้ไขข้อความภาษาไทยนั้น วิธีการที่ใช้เวลาในการวัดระยะได้รวดเร็วที่สุดจะเป็น Hamming Distance, Needleman-Wunsch Algorithm และ Levenshtein Distance ตามลำดับ แต่ Hamming Distance นั้นจะไม่สามารถตัดแบ่งคำก่อนการวัดระยะได้ เพราะจะทำให้จำนวนตัวอักษรในการนำมาเปรียบเทียบไม่เท่ากัน จึงทำให้ Hamming Distance ต้องเปรียบเทียบทั้งข้อความ โดยที่ทั้งสองข้อความต้องมีจำนวนตัวอักษรเท่ากันด้วย

สำหรับ Levenshtein Distance และ Needleman-Wunch Algorithm นั้นสามารถตัดแบ่งคำก่อนการวัดค่าได้ แต่หากจำนวนของคำที่นำมาเปรียบเทียบกันมีไม่เท่ากัน ค่าที่ได้จาก Needleman-Wunsch Algorithm จะมีความคลาดเคลื่อน เนื่องจากค่าที่วัดได้จากตัวอักษรกับคำที่ไม่มีข้อความ ("") จะได้ค่าเป็น 0 ซึ่งคลาดเคลื่อนจากความเป็นจริง แต่ Levenshtein Distance นั้นจะวัดค่าได้ถูกต้องและแม่นยำกว่า

5. สรุปผลการทดสอบการวัดระยะการแก้ไขกับข้อความภาษาไทย

ในการประยุกต์ใช้การวัดระยะการแก้ไขกับข้อความภาษาไทยนั้น Levenshtein Distance ถือเป็นวิธีที่เหมาะสมกับข้อความภาษาไทย เนื่องจากไม่มีปัญหาเรื่องค่าที่คลาดเคลื่อนจากการตัดแบ่งคำที่ได้จำนวนของคำไม่เท่ากัน และยังได้ค่าที่ตรงตัว โดยไม่ต้องไปแปลผลของค่าที่ได้ ซึ่งจะต่างจาก Needleman-Wunsch Algorithm ที่เป็นวิธีในการวัดหาค่าความเหมือนกันของข้อความ โดยค่าที่ได้จากวิธีนี้จะบอกได้แค่ว่าข้อความทั้งสองที่นำมาเปรียบเทียบกันมีความเหมือนกันมากหรือน้อยเพียงใด ขึ้นอยู่กับค่าที่วัดได้ว่าเป็นบวกหรือลบ

สำหรับ Needleman-Wunsch Algorithm นี้จะเหมาะกับการวัดระยะการแก้ไขข้อความในลักษณะของคำมากกว่าจะมาวัดแบบทั้งประโยค เพราะหากมีการตัดแบ่งคำแล้ว ค่าที่ได้จาก Needleman-Wunsch Algorithm จะมีความคลาดเคลื่อน ทำให้ผลลัพธ์ที่ออกมาไม่ถูกต้อง

ซึ่งจากผลการทดสอบดังกล่าว สามารถนำแนวทางของการทดสอบนี้ไปต่อยอดเพื่อประยุกต์ใช้กับภาษาไทยได้ เช่น การทำระบบการแก้ไขคำผิดสำหรับภาษาไทยโดยอัตโนมัติ (Auto Correction) เป็นต้น จะทำให้เกิดประโยชน์มากยิ่งขึ้นต่อการใช้ภาษาไทยบนระบบคอมพิวเตอร์

เอกสารอ้างอิง

[1] Department of Electrical and Computer Engineering - University of New Brunswick. n.p. *Error Correction and the Hamming Code.* [Online] Available: http://www.ee.unb.ca/tervo/ee4253/hamming.shtml

[2] Philip N. Klein, Thomas B. Sebastian and Benjamin B. Kimia. 2001. "Shape matching using edit-distance: an implementation." 781-790. In *SODA '01 Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia: SIAM.

[3] Marius Gheorghescu. 2006. "An Automated Virus Classification System." In *Virus Bulletin Conference October 2005*, Washington: Virus Bulletin.

[4] Marco Bertini, Alberto Del Bimbo and Walter Nunziati. 2006. "Video Clip Matching Using MPEG-7 Descriptors and Edit Distance." 133-142. In *5th International Conference, CIVR 2006*, New York: Springer Berlin Heidelberg

[5] Levenshtein VI (1966). "Binary codes capable of correcting deletions, insertions, and reversals". *Soviet Physics Doklady 10*: 707–10.

[6] Dan Jurafsky. n.p. *Minimum Edit Distance.* [Online] Available: www.stanford.edu/class/cs124/lec/med.pdf

[7] สำนักงานมาตรฐานผลิตภัณฑ์อุตสาหกรรม. ม.ป.ป. *รหัสสำหรับอักขระไทยที่ใช้กับคอมพิวเตอร์.* [Online] Available: http://www.nectec.or.th/it-standards/std620/std620.htm

[8] "โปรแกรมตัดคำภาษาไทย (Thai Word Segmentation)." *สารานุกรมไทยสำหรับเยาวชน โดยพระราชประสงค์ในพระบาทสมเด็จพระเจ้าอยู่หัว.* เล่มที่ 25

[9] S. B. Needleman and C. D. Wunsch. 1969. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *Department of Biochemistry, Northwestern University.*

[10] Paisarn Charoenpornsawat. n.p. *SWATH - Thai Word Segmentation.* [Online] Available: http://www.cs.cmu.edu/~paisarn/software.html

ภาคผนวก ข

หนังสือตอบรับการตีพิมพ์บทความในการประชุมทางวิชาการระดับชาติ ด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 10

Dear Metawee Jindarak

As your paper has been ACCEPTED for ORAL presentation at the NCCIT2014 and you have registered and paid the registration fees of your paper, I would like to officially invite you to present your paper below.

NCCIT2014 - 31
Title: Implementation of edit distance with Thai language
Author(s): Metawee Jindarak and Supakit Nootyaskool

All accepted papers will be published in the Proceedings of the 10th National Conference on Computing and Information Technology (NCCIT2014).

Information of NCCIT2014 regarding accommodation and transportation is available on the conference web site http://www.nccit.net and more will come very soon. The conference information will be updated shortly to include the full technical program.

For your oral presentation, you must prepare PowerPoint slides to present your work to the audiences about 15 minutes (10-12 min talk and 3-5 min for questions).

We look forward to welcoming you on May 8th-9th, 2014 at NCCIT2014, Angsana Laguna Phuket, Thailand.

Sincerely,

P. Meesad

Associate Professor Dr. Phayung Meesad
NCCIT2014 General Chair
Dean, Faculty of Information Technology, KMUTNB
pym@kmutnb.ac.th, admin@nccit.net
http://www.nccit.net

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายเมธาวี จินดารักษ์
ที่อยู่	84/142 หมู่บ้านฟลอร่าวิลล์ ถ.รังสิต-ปทุมธานี ต.บางพูน อ.เมืองปทุมธานี จ.ปทุมธานี 12000
ประวัติการศึกษา	ปี พ.ศ. 2552 วิทยาการคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์