การจำแนกรายงานจุดบกพร่องแบบอัตโนมัติ

วิทยานิพนธ์

ของ

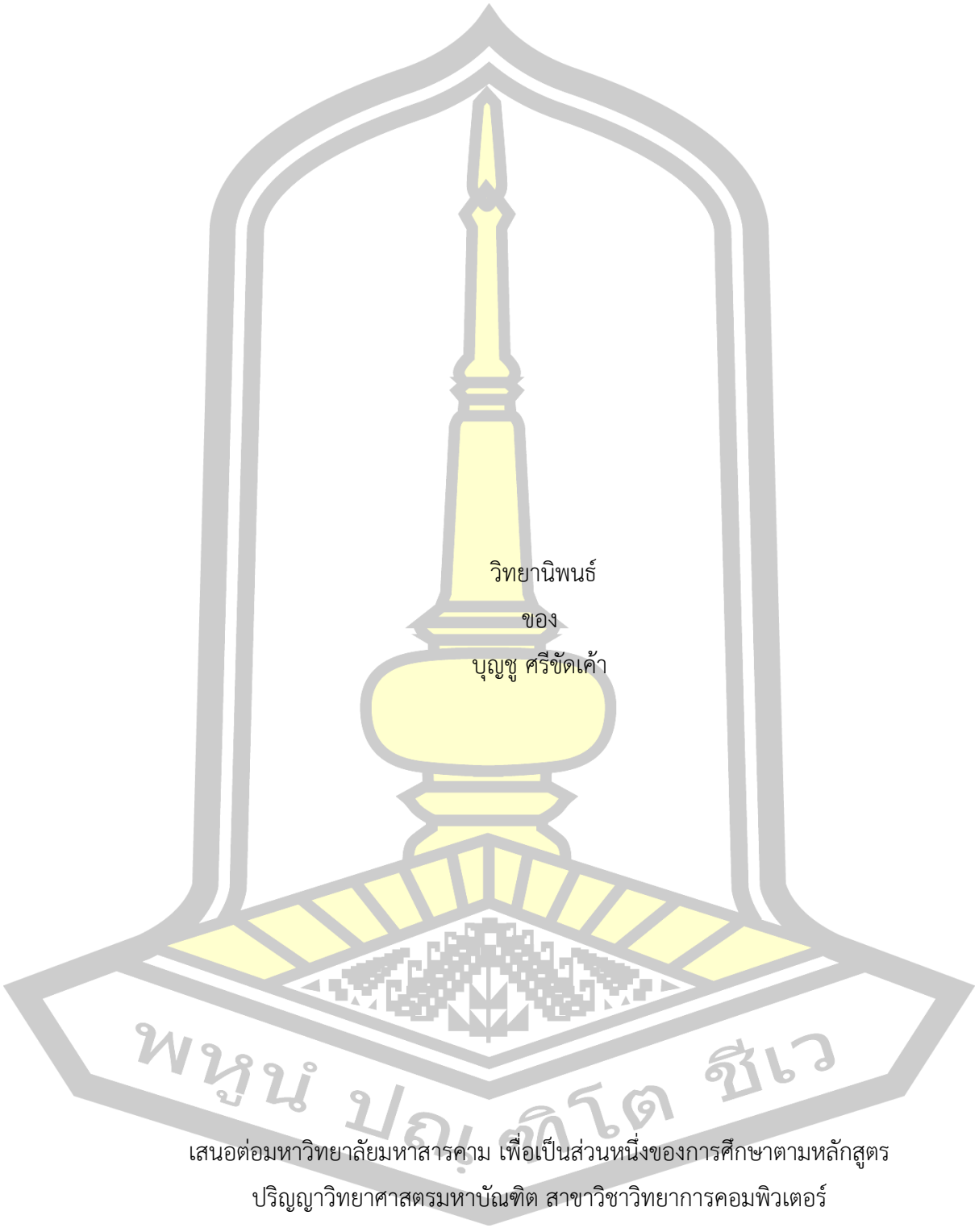
บุญชู ศรีขัดเค้า

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

กรกฎาคม 2562

การจำแนกรายงานจุดบกพร่องแบบอัตโนมัติ



วิทยานิพนธ์

ของ

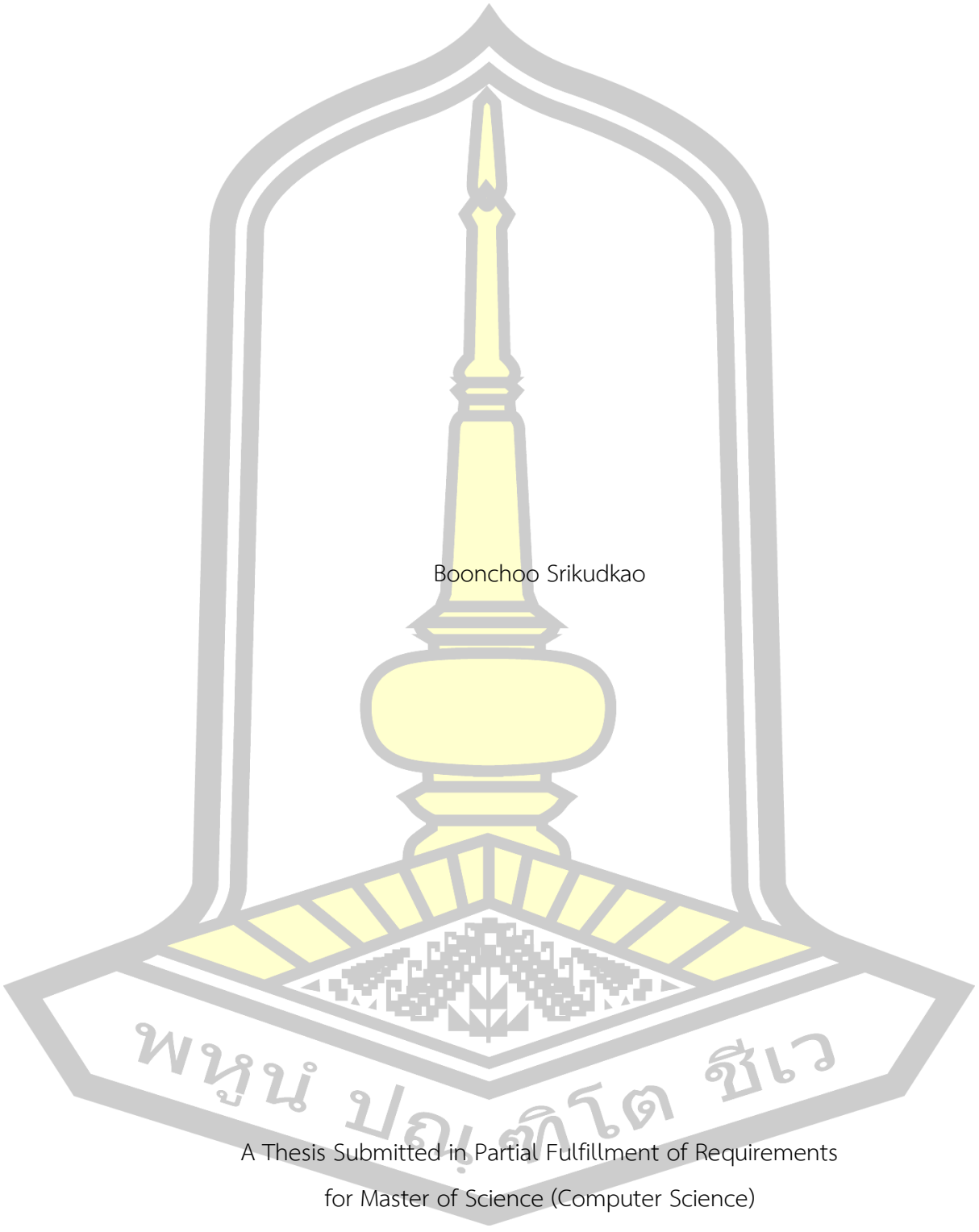
บุญชู ศรีขัดเค้า

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

กรกฎาคม 2562

Automatic Classification of Bug Reports



Boonchoo Srikudkao

A Thesis Submitted in Partial Fulfillment of Requirements

for Master of Science (Computer Science)

July 2019

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาวิทยานิพนธ์ของนายบุญชู ศรีขัดเค้า แล้ว เห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผศ. ดร. ธัชพงศ์ กตัญญูกุล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผศ. ดร. จันทิมา พลพินิจ)

.....กรรมการ
(ผศ. ดร. พัฒนพงษ์ ชมภูวิเศษ)

.....กรรมการ
(ผศ. ดร. ฉัตรเกล้า เจริญผล)

มหาวิทยาลัยอนุมัติให้รับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

.....
(ผศ. ศศิธร แก้วมั่น)
คณบดีคณะวิทยาการสารสนเทศ

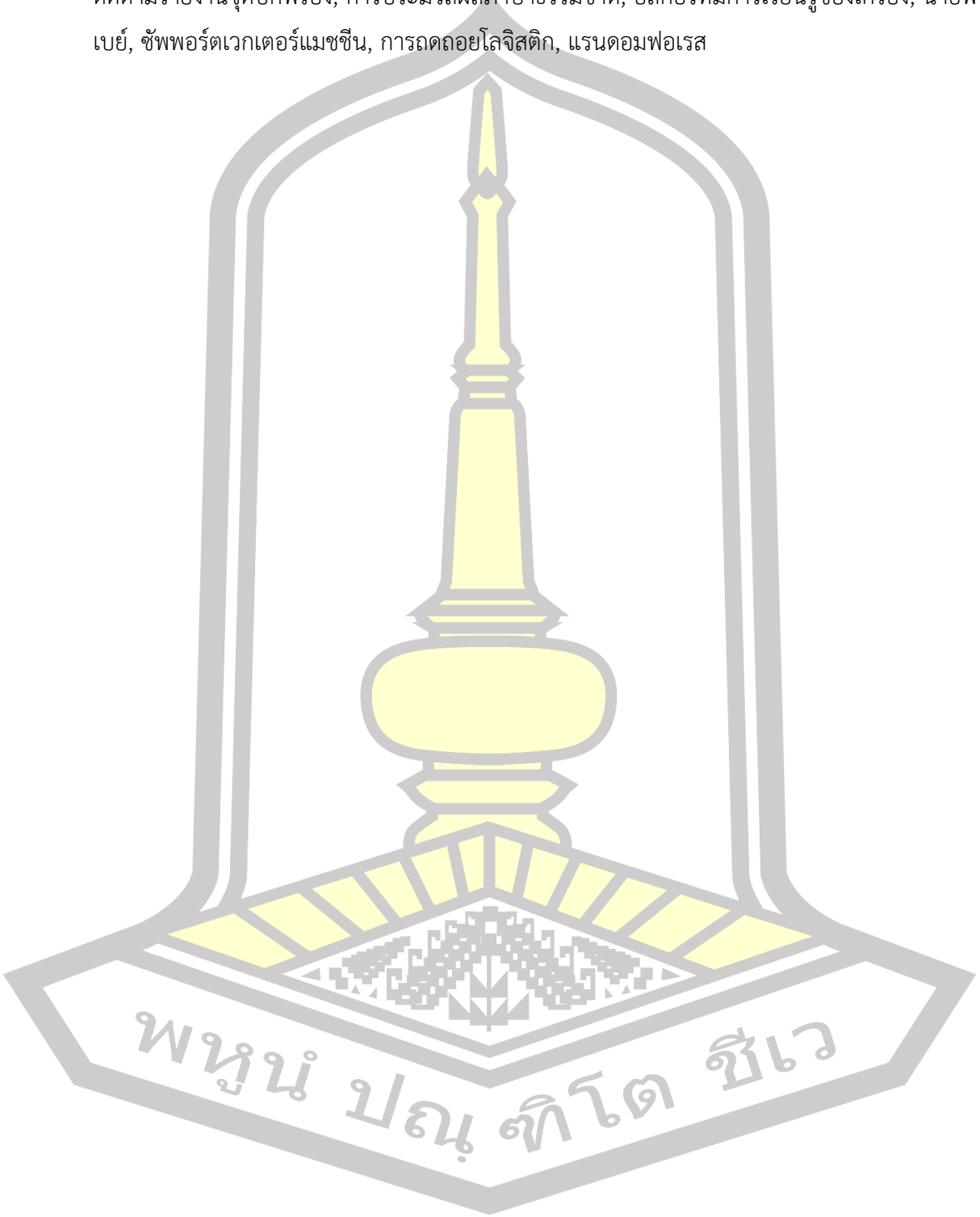
.....
(ผศ. ดร. กริสน์ ชัยมูล)
คณบดีบัณฑิตวิทยาลัย

ชื่อเรื่อง การจำแนกรายงานจุดบกพร่องแบบอัตโนมัติ
ผู้วิจัย บุญชู ศรีขัดเค้า
อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. จันทิมา พลพินิจ
ปริญญา วิทยาศาสตรมหาบัณฑิต **สาขาวิชา** วิทยาการคอมพิวเตอร์
มหาวิทยาลัย มหาวิทยาลัยมหาสารคาม **ปีที่พิมพ์** 2562

บทคัดย่อ

รายงานจุดบกพร่องเป็นข้อมูลที่สำคัญสำหรับการปรับปรุงคุณภาพของซอฟต์แวร์ ในการรวบรวมรายงานจุดบกพร่องจำนวนมากจากผู้ใช้งานทั่วโลก ระบบติดตามจุดบกพร่องจึงได้ถูกพัฒนาและนำเสนอขึ้นมา ซึ่งระบบดังกล่าวจะช่วยให้ผู้ใช้งานสามารถรายงาน อธิบาย ติดตาม จำแนก และแสดงความคิดเห็นเกี่ยวกับจุดบกพร่องที่พบในซอฟต์แวร์ได้ แต่อย่างไรก็กระบวนการในระบบติดตามรายงานจุดบกพร่องยังคงดำเนินการโดยมนุษย์ในการคัดแยกรายงาน ทำให้งานในส่วนนี้กลายเป็นงานที่ใช้เวลานานและแรงงานจำนวนมากในการวิเคราะห์รายงานจุดบกพร่อง ดังนั้นในงานงานวิจัยฉบับนี้จึงมุ่งเน้นในการศึกษาที่เกี่ยวข้องกับรายงานจุดบกพร่อง โดยหนึ่งในปัญหาที่พบมากในการศึกษาที่เกี่ยวกับรายงานจุดบกพร่องคือ การจำแนกรายงานจุดบกพร่องออกเป็นรายงานจุดบกพร่องที่แท้จริงและรายงานที่ไม่ใช่รายงานจุดบกพร่อง ซึ่งการวิจัยนี้ได้นำเสนอแนวทางใหม่ในการจำแนกรายงานจุดบกพร่องแบบอัตโนมัติที่เรียกว่า “การจำแนกรายงานจุดบกพร่องโดยใช้ขั้วคำ (Polarity-based bug report classification)” สำหรับเครื่องมือหลักที่ใช้ในกระบวนการที่นำเสนอนี้คือเทคนิคการประมวลผลทางภาษาธรรมชาติ (Natural Language processing techniques) นอกจากนี้ยังได้ศึกษาเทคนิคการให้คำน้ำหนักคำและอัลกอริทึมการเรียนรู้ของเครื่องที่หลากหลาย เพื่อให้ได้แนวทางที่เหมาะสมที่สุดในการวิจัยนี้ ซึ่งเทคนิคการให้คำน้ำหนักคำที่ศึกษาในงานวิจัยนี้ประกอบด้วย tf, tf-idf, BM25, and MATF (Multi Aspect tf) ในขณะที่อัลกอริทึมการเรียนรู้แบบมีผู้สอนที่ใช้ ได้แก่ นาอีฟเบย์ (Naïve Bayes), ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machines), การถดถอยโลจิสติก (Logistic Regression) และแรนดอมฟอเรส (Random Forest) ภายหลังจากการทดสอบด้วยค่าความระลึก (Recall), ค่าความแม่นยำ (Precision) และค่าเอฟ (F-measure) ปรากฏว่าให้ผลลัพธ์ที่ยอมรับได้เมื่อเปรียบเทียบกับงานวิจัยที่ผ่านมา และในการใช้ชุดข้อมูลของ Herzig พบว่าการให้น้ำหนักคำที่มีความเหมาะสมที่สุดคือ MATF และอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนที่เหมาะสมที่สุดคืออัลกอริทึมการถดถอยโลจิสติก

คำสำคัญ : รายงานจุดบกพร่อง, รายงานที่ไม่ใช่จุดบกพร่อง, ผู้คัดแยกรายงานจุดบกพร่อง, ระบบติดตามรายงานจุดบกพร่อง, การประมวลผลภาษาธรรมชาติ, อัลกอริทึมการเรียนรู้ของเครื่อง, นาอีฟเบย์, ซัพพอร์ตเวกเตอร์แมชชีน, การถดถอยโลจิสติก, แรนดอมฟอเรส



TITLE Automatic Classification of Bug Reports

AUTHOR Boonchoo Srikudkao

ADVISORS Assistant Professor Jantima Polpinij , Ph.D.

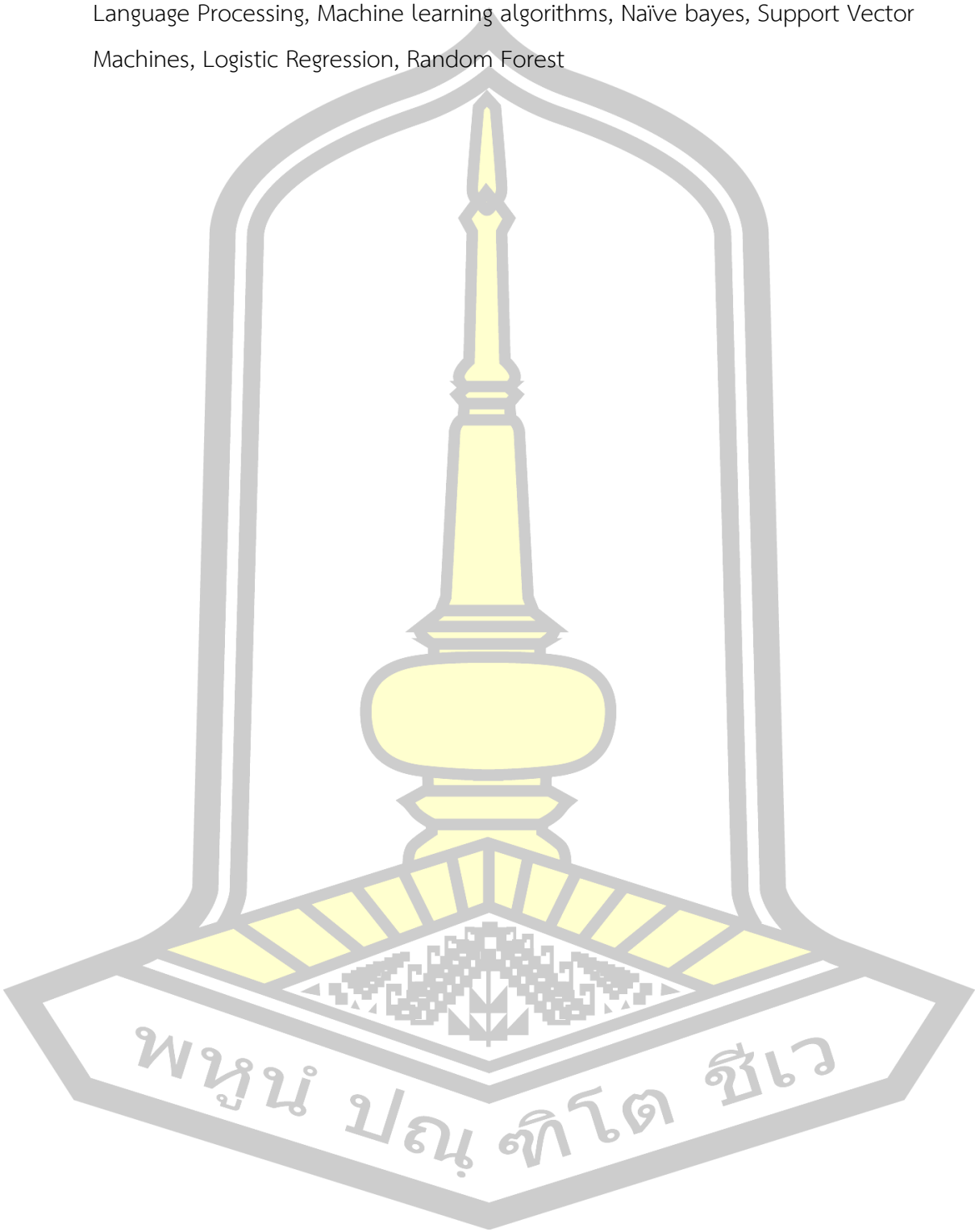
DEGREE Master of Science **MAJOR** Computer Science

UNIVERSITY Mahasarakham University **YEAR** 2019

ABSTRACT

Bug reports become the significant information for improving software quality. To collect the large bug reports from users around the world, many bug tracking systems (BTS) have been developed and proposed. These systems allow users to report, describe, track, classify, and comment on their bug reports. Unfortunately, various tasks on the BTS are still performed by bug triagers manually. It becomes a time consuming and labour intensive for bug report analysis. Therefore, many works focus on studying related to bug reports. One of the most common issues of bug report studies is to classify bug reports into real-bug and non-bug report. This study also presents a novel method of automatic bug report classification, called the polarity-based bug report classification. The proposed method is driven on the Natural Language processing techniques. We also study various weighting schemes and supervised machine learning algorithms to obtain the most appropriate solution for the proposed study. The weighting schemes studied in this work are tf, tf-idf, BM25, and MATF (Multi Aspect tf), while supervised machine learning algorithms are Naïve Bayes, Support Vector Machines, Logistic Regression, and Random Forest. After testing via recall, precision, and F-measure, the results can be accepted after comparing to the previous researches. By using Herzig's dataset, the most appropriate weighting scheme is the MATF, whereas the most appropriate supervised machine learning algorithm is the Logistic Regression.

Keyword : Bug reports, non-bug reports, Bug triagers, Bug Tracking Systems, Natural Language Processing, Machine learning algorithms, Naïve bayes, Support Vector Machines, Logistic Regression, Random Forest



กิตติกรรมประกาศ

วิทยานิพนธ์ เรื่อง “การจำแนกรายงานจุดบกพร่อง” ฉบับนี้สามารถพัฒนาจนสำเร็จลุล่วงไปได้ด้วยดีโดยได้รับการดูแลเป็นอย่างดีจาก ผศ. ดร. จันทิมา พลพินิจ ที่เป็นทั้งที่ปรึกษาโครงการและอาจารย์ที่กรุณาสั่งสอนให้คำแนะนำ คำปรึกษา และช่วยชี้แนะแนวทางตรวจทานแก้ไขข้อบกพร่องในการค้นคว้าหาความรู้ เพื่อนำมาใช้ในการเขียนโครงงานฉบับนี้ ทางคณะผู้จัดทารู้สึกซาบซึ้งในความอนุเคราะห์ จากท่านอาจารย์และขอกราบพระคุณไว้เป็นอย่างสูง

ขอขอบคุณ คุณพ่อ คุณแม่ ญาติพี่น้อง รวมถึงพี่ๆ และเพื่อนๆ โดยเฉพาะในหน่วยวิจัยทางปัญญา (Intellect Laboratory) ที่คอยให้ความช่วยเหลือ ให้กาลังใจ และส่งเสริมในทุกๆ ด้าน นอกจากนี้ข้าพเจ้าขอขอบพระคุณอาจารย์ในภาควิชาวิทยาการคอมพิวเตอร์ทุกท่าน ที่ได้ให้ความรู้ที่สามารถใช้ในการศึกษาเล่าเรียนในระดับที่สูงขึ้น ตลอดจนการใช้ประโยชน์จากความรู้เพื่อการทางานต่อไปโดยคุณค่าและประโยชน์อันพึงมีมาจากโครงงานฉบับนี้ผู้จัดทาขอมอบแด่ผู้มีพระคุณทุกท่าน

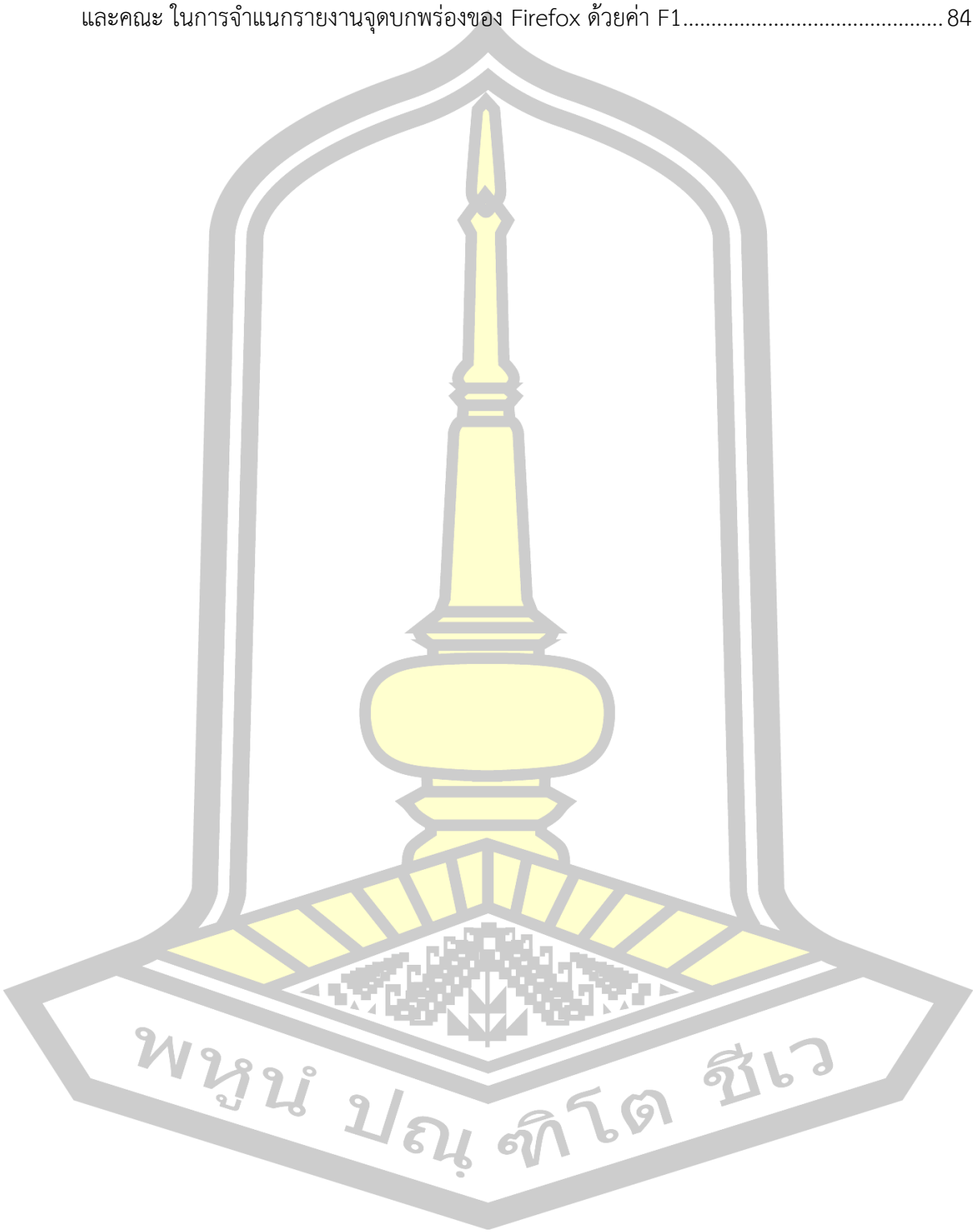
บุญชู ศรีขัดเค้า

สารบัญ

หน้า

สารบัญตาราง

หน้า



สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

จุดบกพร่องของซอฟต์แวร์ (Software Bug) [1, 2] หมายถึง ข้อผิดพลาด ข้อบกพร่อง หรือความผิดปกติของซอฟต์แวร์ในคอมพิวเตอร์ หรือระบบที่ก่อให้เกิดผลลัพธ์ หรือพฤติกรรมที่ไม่คาดคิด หรือการทำงานที่ไม่ราบรื่น ซึ่งเป็นเรื่องปกติที่อาจจะพบ จุดบกพร่องในซอฟต์แวร์ได้ เพื่อที่จะแก้ไขจุดบกพร่องของซอฟต์แวร์ที่พบ จึงต้องมีการรายงานจุดบกพร่องที่พบให้กับผู้พัฒนาซอฟต์แวร์ได้ทำการแก้ไขปรับปรุงซอฟต์แวร์ โดยในองค์กรขนาดเล็กที่มีการพัฒนาซอฟต์แวร์ไว้ในในองค์กร การรายงานจุดบกพร่องที่พบในซอฟต์แวร์สามารถรายงานให้ทางผู้พัฒนาซอฟต์แวร์ได้ทำการแก้ไขปรับปรุงได้โดยตรง แต่ในองค์กรขนาดใหญ่ หรือโอเพ่นซอร์สที่มีผู้ใช้งานอยู่ทั่วโลกพบว่าผู้ใช้งานไม่สามารถรายงานจุดบกพร่องที่พบในซอฟต์แวร์ให้กับทางผู้พัฒนาได้โดยตรง เพื่อแก้ไขปัญหาดังกล่าวจึงมีการพัฒนาระบบติดตามจุดบกพร่องของซอฟต์แวร์ (Bug Tracking System) [1, 3, 4] ซึ่งช่องทางให้ผู้ใช้งานซอฟต์แวร์สามารถรายงานจุดบกพร่องที่พบให้กับทางผู้พัฒนาได้ทำการแก้ไขปรับปรุง

ระบบติดตามจุดบกพร่อง เป็นระบบที่มีหน้าที่จัดการรายงานจุดบกพร่องของซอฟต์แวร์ผ่านทางเว็บไซต์ที่ผู้พัฒนาซอฟต์แวร์ได้จัดเตรียมไว้ หรือระบบติดตามจุดบกพร่องจากผู้ให้บริการอาทิ Bugzilla [5] Jira [6] GNATS [7] Perforce [8] เป็นต้น โดยระบบติดตามจุดบกพร่องได้รับการออกแบบให้ผู้ใช้งานซอฟต์แวร์ หรือผู้พัฒนาซอฟต์แวร์สามารถรายงานจุดบกพร่องที่พบในซอฟต์แวร์เข้าสู่ระบบติดตามจุดบกพร่องได้จากทั่วโลก [3, 9] และระบบติดจามจุดบกพร่องจะทำการรวบรวมรายงานจุดบกพร่องที่ได้รายงานเข้ามาไว้ในคลังสำหรับจัดเก็บจุดบกพร่องซอฟต์แวร์ (Software Bug Repository) [1, 3, 4] ซึ่งเป็นหัวใจสำคัญของระบบติดตามจุดบกพร่อง เนื่องจากเป็นแหล่งที่ความรู้ที่สำคัญที่จะนำไปใช้ในการปรับปรุง และพัฒนาซอฟต์แวร์ [1, 2, 10, 11]

ภายหลังจากที่มีการรายงานจุดบกพร่องที่พบเข้าสู่คลังจัดเก็บรายงานจุดบกพร่องของซอฟต์แวร์แล้ว จะเข้าสู่กระบวนการการแก้ไขปรับปรุงจุดบกของซอฟต์แวร์ จะเป็นการดำเนินการโดยผู้ที่มีส่วนเกี่ยวข้องในแต่ละส่วนของซอฟต์แวร์ เป็นผู้ดำเนินการเอง โดยเริ่มตั้งแต่การคัดกรองซ้ำซ้อนรายงานจุดบกพร่อง การมอบหมายรายงานแก่ผู้พัฒนาซอฟต์แวร์ที่มีความเหมาะสมกับจุดบกพร่องที่พบ และอื่น ๆ ซึ่งล้วนเป็นการทำงานโดยมนุษย์ จากการศึกษาพบว่าเมื่อมีการรายงานจุดบกพร่องซอฟต์แวร์ของบริษัท Mozilla เข้าสู่ระบบจะใช้เวลาเฉลี่ย 26.1 วัน ในการคัดกรองความซ้ำซ้อน และใช้เวลาเฉลี่ย 161.1 วัน ในการมอบหมายรายงานจุดบกพร่องแก่ผู้พัฒนาซอฟต์แวร์ [12]

จากการศึกษาพบว่ามีรายงานจุดบกพร่องที่พบในซอฟต์แวร์เข้าสู่ระบบจำนวนมาก โดยมีการรายงานจุดบกพร่องที่พบในซอฟต์แวร์ของบริษัท Mozilla จำนวนมากกว่า 350 รายงานต่อวัน แต่ในการคัดแยกรายงานจุดบอกพร่องของซอฟต์แวร์สามารถทำได้เพียง 300 รายงานต่อวัน ซึ่งในปัจจุบันพบว่ามีรายงานจุดบกพร่องภายในคลังรายงานจุดบกพร่องบริษัท Mozilla มีจำนวนมากกว่า 1,297,391 รายงาน อีกทั้งในการคัดแยกรายงานจุดบกพร่องของซอฟต์แวร์ต้องใช้ผู้เชี่ยวชาญที่มีประสบการณ์มากกว่า 10 ปี [13, 14] และยังพบอีกว่ามีหนึ่งในสามของรายงานจุดบกพร่องที่มีการรายงานเข้าสู่ระบบไม่ใช่จุดบกพร่องของซอฟต์แวร์จริง [15] ซึ่งส่งผลให้การบำรุงรักษาซอฟต์แวร์ (Software Maintenance) มีใช้เวลานาน และค่าใช่จ่ายที่สูง

จากเหตุผลข้างต้น งานวิจัยฉบับนี้จึงนำเสนอการจำแนกรายงานที่เป็นจุดบกพร่องซอฟต์แวร์ และไม่ใช่จุดบกพร่องของซอฟต์แวร์แบบอัตโนมัติ เพื่อเป็นเครื่องมือที่ช่วยในการวิเคราะห์รายงานจุดบกพร่องในคลังจัดรายงานจุดบกพร่องที่ต้องแก้ไปรับปรุง เพื่อลดเวลา และค่าใช้จ่ายในการบำรุงรักษาซอฟต์แวร์

1.2 วัตถุประสงค์ของการวิจัย

นำเสนอการวิจัยเพื่อพัฒนากระบวนการในการจำแนกรายงานจุดบกพร่องแบบอัตโนมัติแบบ 2 กลุ่มคือ Bug Report และ Non-bug Report บนพื้นฐานของการเรียนรู้แบบมีผู้สอน (Supervised Learning)

1.2 วัตถุประสงค์ของการวิจัย

ได้กระบวนการในการจำแนกรายงานจุดบกพร่องของซอฟต์แวร์ที่มีประสิทธิภาพที่ดีขึ้น

1.4 ขอบเขตของการวิจัย

นำเสนอการระบุรายงานจุดบกพร่องที่แท้จริงแบบอัตโนมัติ ซึ่งเป็นกระบวนการคัดกรองรายงานจุดบกพร่องที่มีจัดเก็บอยู่ในคลังรายงานจุดบกพร่อง (Bug Report Repository)

1. ชุดข้อมูลที่ใช้ในการศึกษา คือ รายงานจุดบกพร่องซอฟต์แวร์ของบริษัท Mozilla ในรูปแบบภาษาอังกฤษ ที่สามารถดาวน์โหลดได้จาก https://bugzilla.mozilla.org และชุดข้อมูลรายงานจุดบกพร่องของ Herzig ซึ่งสามารถเข้าถึงได้ที่ http://www.st.cs.uni-saarland.de/softevo//bugclassify/

2. ชุดข้อมูลที่ใช้ในงานวิจัยนี้สามารถแบ่งตามการใช้งานได้เป็น 2 กลุ่ม คือ

2.1 ชุดข้อมูลเพื่อการสร้างคลังขั้วของคำ (Polarity Corpus) ของคำที่แสดงความเป็นรายงานจุดบกพร่อง และรายงานที่ไม่ใช่จุดบกพร่อง โดยใช้ข้อมูลรายงานจุดบกพร่อง Core ของ

Mozilla จำนวน 50,000 รายงาน โดยเป็นรายงานจุดบกพร่องจำนวน 25,000 รายงาน และรายงานจุดบกพร่องที่ไม่ใช่จุดบกพร่องจำนวน 25,000 รายงาน

2.2 ชุดข้อมูลสำหรับการศึกษาการจำแนกรายงานจุดบกพร่อง โดยทดสอบกับข้อมูลสองชุดข้อมูล ได้แก่ ชุดข้อมูลรายงานจุดบกพร่อง Firefox ของ Mozilla จำนวน 50,000 รายงาน โดยเป็นรายงานจุดบกพร่องจำนวน 25,000 รายงาน และรายงานจุดบกพร่องที่ไม่ใช่จุดบกพร่องจำนวน 25,000 รายงาน และชุดข้อมูลรายงานจุดบกพร่องของ Herzig

3. งานวิจัยนี้นำเสนอกระบวนการในการจำแนกรายงานจุดบกพร่องแบบมีผู้สอน (Supervised Learning) ที่มีพื้นฐานจากเหมืองข้อความ (Text Mining) และการประมวลผลภาษาธรรมชาติ (Natural Language Processing)

4. คุณลักษณะที่ใช้ในงานวิจัยฉบับนี้ จะทำการศึกษาทั้งแบบ Unigram

5. ทดสอบการให้น้ำหนักคุณลักษณะจะทำการศึกษาใน 4 เทคนิค คือ TF, TF-IDF, BM25 และ MATF

1.5 นิยามศัพท์เฉพาะ

1. จุดบกพร่อง (Bug) คือ จุดบกพร่องที่พบในซอฟต์แวร์

2. รายงานจุดบกพร่อง (Bug report) คือ รายงานจุดบกพร่องที่พบในซอฟต์แวร์ที่ผู้ใช้ตรวจพบและมีการรายงานผลให้กับทีมพัฒนาได้ทราบผ่านเครือข่ายอินเตอร์เน็ต

3. Bug Report และ Non-bug Report

3.1 Bug Report เป็นรายงานจุดบกพร่องที่ผ่านการคัดกรองโดยผู้คัดกรองรายงานจุดบกพร่อง ซึ่งมีสถานะตามวงจรชีวิตของจุดบกพร่องเป็น New, Assigned, Resolved (Fixed), Resolved (Wontfix), Resolved (Workforme), Resolved (Duplicate), Verified, Closed และ Reopen

3.2 Non-bug Report เป็นรายงานจุดบกพร่องที่ผ่านการคัดกรองโดยผู้คัดกรองรายงานจุดบกพร่อง ซึ่งมีสถานะตามวงจรชีวิตของจุดบกพร่องเป็น Resolved (Invalid)

4. การสร้างเหมืองจุดบกพร่องของซอฟต์แวร์ (Software Bug Mining) เป็นการประยุกต์เทคนิคด้านเหมืองข้อความและการประมวลผลภาษาธรรมชาติมาใช้ในการวิเคราะห์จุดบกพร่องของซอฟต์แวร์จากรายงานจุดบกพร่อง

5. ระบบติดตามจุดบกพร่อง (Bug Tracking System) คือ ระบบติดจุดบกพร่องของซอฟต์แวร์ จากการรายงานจุดพร่องที่พบโดยผู้ใช้งานซอฟต์แวร์ และผู้พัฒนาซอฟต์แวร์

6. วงจรชีวิตของจุดบกพร่อง (Bug Life Cycle) คือ กระบวนการจัดการรายงานจุดบกพร่องของซอฟต์แวร์ที่มีการรายงานเข้าสู่ระบบติดตามจุดบกพร่อง

7. ผู้ตรวจสอบรายงานจุดบกพร่อง (Bug Triager) คือ ผู้ที่มีหน้าที่คัดกรองรายงานจุดบกพร่องที่มีสถานะเป็น “Unconfirmed” ซึ่งเป็นสถานะแรกของรายงานจุดบกพร่องที่มีการรายงานเข้าสู่ระบบ

8. การคัดแยกจุดบกพร่อง (Bug Triage) คือ การคัดกรองรายงานจุดบกพร่องที่มีสถานะเป็น “Unconfirmed” ซึ่งเป็นสถานะแรกของรายงานจุดบกพร่องที่มีการรายงานเข้าสู่ระบบติดตามจุดบกพร่องตามกระบวนการในวงจรชีวิตของจุดบกพร่อง ซึ่งได้รับโดยผู้คัดแยกที่รับผิดชอบในแต่ละส่วนของซอฟต์แวร์

9. ผู้แก้ไขจุดบกพร่อง คือ ผู้พัฒนาซอฟต์แวร์ (Developer) ที่ได้ทำการแก้ไขปรับปรุงซอฟต์แวร์ตามรายงานจุดบกพร่องที่ได้รายงานเข้าในระบบติดตามจุดบกพร่อง

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 รายงานจุดบกพร่อง (Bug Report)

จุดบกพร่องของซอฟต์แวร์ (Software Bug) [1, 2, 15-17] หมายถึง ข้อผิดพลาด ข้อบกพร่อง หรือความผิดปกติของซอฟต์แวร์ในคอมพิวเตอร์ หรือระบบที่ก่อให้เกิดผลลัพธ์ หรือพฤติกรรมที่ไม่คาดคิด หรือการทำงานที่ไม่ราบรื่น [1] ซึ่งเป็นเรื่องปกติที่ซอฟต์แวร์จะมีจุดบกพร่อง ดังนั้นในการพัฒนาซอฟต์แวร์จึงต้องมีบุคคลที่ต้องทำการตรวจสอบหาความผิดพลาดของซอฟต์แวร์ (Soft Tester) แต่ถึงกระนั้นก็ซอฟต์แวร์อาจจะยังมีจุดบกพร่อง ดังที่คำกล่าวที่ว่า “As programmers can hardly write programs without any bugs” [15] หรือก็ คือ “ในฐานะที่เป็นโปรแกรมเมอร์เป็นไปไม่ได้ที่จะสามารถเขียนโปรแกรมได้โดยที่ไม่มีจุดบกพร่องใดๆ” เนื่องจากจุดบกพร่องอาจเกิดจากความผิดพลาดจากการเขียนโปรแกรมเพียงเล็กน้อย แต่ผลจากความผิดพลาดเพียงเล็กน้อยอาจส่งผลให้ซอฟต์แวร์ไม่สามารถทำงานได้ หรือซอฟต์แวร์สามารถทำงานได้ผลลัทธ์ที่ต้องการ แต่มีการใช้ทรัพยากร หรือเวลามากกว่าที่ควรจะเป็น ซึ่งจุดบกพร่องนั้นมักจะถูกพบโดยผู้ใช้งาน (End-user) ในการใช้งานจริง เพื่อดำเนินการแก้ไขจุดบกพร่องดังกล่าวจึงต้องมีการรายงานการพบจุดบกพร่องนั้นแก่ทางผู้พัฒนาได้ดำเนินการปรับปรุงซอฟต์แวร์นั้น ๆ

รายงานจุดบกพร่องซอฟต์แวร์ (Software Bug Report) [18] คือ เอกสารทางซอฟต์แวร์ที่อธิบายเกี่ยวกับจุดบกพร่องของซอฟต์แวร์นั้น ซึ่งถูกเขียนขึ้นโดยผู้พัฒนาซอฟต์แวร์ ผู้ทดสอบซอฟต์แวร์ หรือผู้ใช้ เพื่อช่วยให้ทางผู้พัฒนาซอฟต์แวร์ได้ทราบข้อมูลเกี่ยวพฤติกรรม หรือการทำงานของซอฟต์แวร์ที่ผู้ทางพัฒนาซอฟต์แวร์นั้น ๆ พัฒนาขึ้น [15] โดยภายในรายงานจุดบกพร่องจะประกอบรายละเอียดเกี่ยวกับจุดบกพร่อง อาทิเช่น หมายเลขของรายงานยจุดบกพร่อง ชื่อเรื่อง วันเวลาที่รายงาน ความรุนแรงหรือความสำคัญ เป็นต้น

2.2 ระบบติดตามจุดบกพร่อง (Bug Tracking System)

เนื่องด้วยซอฟต์แวร์ในปัจจุบันมีขนาดใหญ่ มีความซับซ้อนสูง และยังมุ่งเน้นด้านความปลอดภัยมากขึ้น ส่งผลให้การตรวจจับจุดบกพร่องของซอฟต์แวร์ทำได้ยาก ทั้งใช้เวลานาน [4, 19-21] เพื่อให้สามารถตรวจจับจุดบกพร่องของซอฟต์แวร์ และรวบรวมจุดบกพร่องจึงได้มีการพัฒนาระบบติดตามจุดบกพร่อง เพื่อให้ผู้ใช้งานสามารถรายงานเกี่ยวจุดพร่องที่พบในซอฟต์แวร์ให้ทางผู้พัฒนาทราบ

ระบบติดตามจุดบกพร่อง หรือ ระบบติดตามความผิดปกติ (Defect Tracking System) เป็นระบบที่มีหน้าที่จัดการรายงานจุดบกพร่องของซอฟต์แวร์ผ่านเว็บไซต์ที่ผู้พัฒนาซอฟต์แวร์จัดเตรียมไว้

หรือใช้ระบบติดตามจุดบกพร่องจากผู้ให้บริการเฉพาะ อาทิ Bugzilla [5] Jira [6] GNATS [7] Perforce [8] ซึ่งถือว่าเป็นระบบที่มีความสำคัญอย่างมากในการพัฒนาซอฟต์แวร์ โดยได้รับการออกแบบให้ผู้ใช้งาน หรือผู้พัฒนาซอฟต์แวร์สามารถรายงานจุดบกพร่องของซอฟต์แวร์เข้าสู่ระบบได้จากทั่วโลก [9] และจัดเก็บรวบรวมไว้เป็นฐานข้อมูลในคลังสำหรับจัดเก็บจุดบกพร่องซอฟต์แวร์ (Software Bug Repository) ซึ่งเป็นหัวใจสำคัญของระบบติดตามจุดบกพร่อง เพราะถือเป็นแหล่งที่ความรู้ที่สำคัญที่จะนำไปใช้ในการปรับปรุง และพัฒนาซอฟต์แวร์ [1, 10, 11]

ในปัจจุบันมีการรายงานจุดบกพร่องเข้าสู่ระบบจำนวนมาก จากงานวิจัยของ Anvik และคณะ [22] พบมีการรายงานจุดบกพร่องของ Mozilla มีมากกว่า 350 รายงานต่อวัน และในการคัดแยกรายงานจุดบอกพร่องของซอฟต์แวร์สามารถทำได้เพียง 300 รายงานต่อวัน อีกทั้งในการคัดแยกรายงานจุดบกพร่องของซอฟต์แวร์ต้องใช้ผู้เชี่ยวชาญที่มีประสบการ์มากกว่า 10 ปี ซึ่งในปี พ.ศ. 2560 จากงานวิจัย Banerjee และคณะ [23] พบว่ารายงานจุดบกพร่องของ Mozilla มีจำนวนมากกว่า 1.1 ล้านรายงาน และในปัจจุบันรายงานจุดบกพร่องของซอฟต์แวร์ Mozilla มีจำนวนมากกว่า 1,295,031 รายงาน

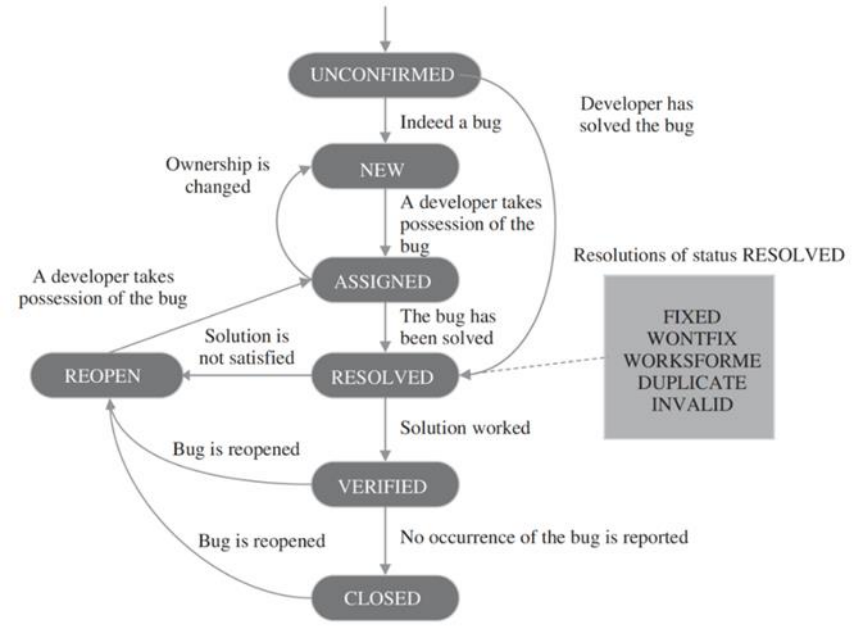
2.2.1 Bugzilla

Bugzilla เป็นโอเพนซอร์ส (Open Source) [1, 5, 11, 24, 25] ที่ถูกพัฒนาใช้เป็นระบบติดตามจุดบกพร่อง โดยมีวัตถุประสงค์เพื่อใช้เป็นเครื่องมือเพื่อให้บุคคล หรือกลุ่มของผู้พัฒนาซอฟต์แวร์ใช้ในการติดตามปัญหาที่พบจากการรายงานจากการรายงานบุคคลที่ได้รับอนุญาต หรือผู้พัฒนา และรวบรวมรายงานจุดบกพร่องที่ได้รับ เพื่อนำไปใช้ในการปรับปรุงแก้ไขซอฟต์แวร์ที่พวกเขาอย่างมีประสิทธิภาพ Bugzilla ได้รับการออกแบบให้มีลักษณะเป็นซอฟต์แวร์เครือข่ายที่ทำงานผ่านเว็บไซต์ ได้รับการพัฒนาขึ้นในราวปี พ.ศ.2541 โดย Terry Weissman โดยแรกเริ่ม Bugzilla ได้รับการพัฒนาด้วยภาษา Tool Command Language (TCL) ซึ่ง ภาษาสคริปต์ที่ต้องใช้ตัวแปรภาษา หรืออินเตอร์พรีเตอร์ในการทำงาน และในปัจจุบันเปลี่ยนมาใช้ภาษา (Practical Extraction and Report Language: Perl)

Bugzilla ได้รับความนิยมเป็นอย่างมาก เนื่องมาจากระบบติดตามจุดบกพร่องของ Bugzilla มีการเปิดให้ดาวน์โหลดไปใช้งานได้ฟรี [4, 11, 25] ซึ่งแต่ต่างจากระบบติดตามจุบบกพร่องอื่นที่มักจะมีค่าใช้จ่าย ถึงแม้ Bugzilla จะเป็นฟรีซอฟต์แวร์ แต่ก็ยังมีคุณสมบัติ และฟังก์ชันหลายอย่างที่เหมือนกับระบบติดตามที่มีค่าใช้จ่าย ดังนั้น Bugzilla จึงเป็นที่นิยมและใช้การอย่างแพร่หลาย แม้แต่ในองค์กรใหญ่ ๆ อย่าง NASA และ IBM [4] ซึ่งในปัจจุบันระบบติดตามจุดบกพร่องของ Bugzilla มีบริษัท และองค์กรกว่า 137 แห่งอยู่ใช้งานระบบในรูปแบบเปิดภายข้อมูลต่อสาธารณะ (Public) อาทิ Mozilla, GNOME, Apache, LibreOffice, Eclipse และมีมากกว่า 1,370 โอเพนซอร์สที่ใช้งานระบบของ Bugzilla ในรูปแบบที่ไม่ได้เปิดเผยข้อมูลแบบสาธารณะ (Private) [5]

2.2.2 วงจรชีวิตของรายงานจุดบกพร่อง (Bug Report Life Cycle)

วงจรชีวิตของรายงานจุดบกพร่องของซอฟต์แวร์ เป็นขั้นตอน หรือกระบวนการการจัดการจุดบกพร่อง ซึ่งในแต่ละซอฟต์แวร์จะมีกระบวนการในการจัดการจุดบกพร่องที่พบในซอฟต์แวร์แตกต่างกันออก โดยจะขึ้นอยู่กับความต้องการขององค์กร หรือความต้องการของผู้พัฒนาซอฟต์แวร์นั้น ๆ ซึ่งส่วนใหญ่จะมีแบ่งขั้นตอนในการจัดการกระบวนการที่สำคัญดังภาพที่ 2-1 [1, 10, 21] ซึ่งจะประกอบด้วย



ภาพที่ 2-1 วงจรชีวิตของรายงานจุดบกพร่อง

ที่มา : [1]

Unconfirmed เป็นสถานะของรายงานจุดบกพร่องแรกเริ่มของรายงานจุดบกพร่อง เมื่อผู้ใช้รายงานจุดบกพร่องเข้าสู่ระบบจะถูกคัดกรองความซ้ำซ้อนเบื้องต้น โดยผู้รับผิดชอบในแต่ละส่วนประกอบของซอฟต์แวร์จะเป็นผู้ตรวจสอบรายงาน เพื่อให้ได้รายงานที่เป็นจุดบกพร่องใหม่เท่านั้น แต่หากเป็นการรายงานโดยผู้พัฒนาเองรายงานจุดบกพร่องจะได้รับการยกเว้นการคัดแยกเบื้องต้น

New เป็นสถานะของรายงานจุดบกพร่องที่ผ่านการคัดแยก ซึ่งได้รับการรับรองว่าเป็นจุดบกพร่องใหม่ และยังเป็นสถานะแรกเริ่มของรายงานจุดบกพร่องที่เป็นการรายงานโดยผู้พัฒนาซอฟต์แวร์ โดยในขั้นตอนต่อไป รายงานจุดบกพร่องจะทำการมอบหมายจุดบกพร่องที่ได้รับการรับรองแล้วให้กับผู้พัฒนาซอฟต์แวร์ที่รับผิดชอบในส่วนนั้น ๆ ของซอฟต์แวร์ได้ทำการแก้ไขปรับปรุง

Assigned เป็นสถานะของรายงานจุดบกพร่องที่ได้มอบหมายความรับผิดชอบให้ผู้พัฒนาที่มีหน้ารับผิดชอบในแต่ละส่วนของซอฟต์แวร์ได้นำไปทำการแก้ไขปรับปรุง

Resolved เป็นสถานะของรายงานจุดบกพร่องที่ผ่านการแก้ไขปรับปรุงโดยผู้พัฒนาเรียบร้อยแล้ว ซึ่งจะถูกส่งต่อให้ผู้ทดสอบซอฟต์แวร์ได้ทำการตรวจสอบความเรียบร้อยของการแก้ไขปรับปรับต่อไป และรายงานที่ได้รับสถานะนี้จะได้รับสถานะย่อย เพื่อแสดงผลจากการแก้ไขปรับปรุง ซึ่งประกอบด้วย Fixed, Wontfix, Workforme, Duplicate และ Invalid

Verified เป็นสถานะของรายงานจุดบกพร่องที่ผ่านการตรวจสอบความเรียบร้อยของการแก้ไขปรับปรุงจุดบกพร่องแล้ว

Closed เป็นสถานะของรายงานจุดบกพร่องที่ได้รับการแก้ไขปรับปรุง และไม่พบปัญหาใด ๆ ซึ่งถือเป็นสถานะสิ้นสุดของการแก้ไขรายงานจุดบกพร่อง

Reopen เป็นสถานะของรายงานจุดบกพร่องที่ได้รับการไขเรียบร้อยแล้ว แต่มีได้รับรายงานเข้ามาอีกครั้งทำให้รายงานจุดบกพร่องจะเข้าสู่กระบวนการขั้นต้นอีกครั้ง โดยจะเข้าสู่การมอบหมายรายงานจุดบกพร่องให้กับผู้พัฒนาในแต่ละส่วนของซอฟต์แวร์

2.2.3 รายละเอียดของรายงานจุดบกพร่อง (Bug Report Detail)

ในหนึ่งรายงานจุดบกพร่องของซอฟต์แวร์จะประกอบไปด้วยข้อมูลรายละเอียดต่าง ๆ ที่สามารถระบุ และอธิบายลักษณะของจุดบกพร่องที่พบในซอฟต์แวร์ ซึ่งข้อมูลที่จะได้จากรายงานจุดบกพร่องจะมีความแตกต่างกันออกไปตามแต่ความต้องการของของทีมผู้พัฒนาซอฟต์แวร์ [3,22] โดยรายงานจุดบกพร่องหนึ่งรายงานสามารถแบ่งออกได้เป็น 3 ส่วน ดังนี้

1) ส่วนสรุปของรายงานจุดบกพร่อง เป็นการอธิบายเกี่ยวกับจุดบกพร่องอย่างคร่าวๆ ดังแสดงในภาพที่ 2-2 ซึ่งส่วนนี้จะประกอบด้วย

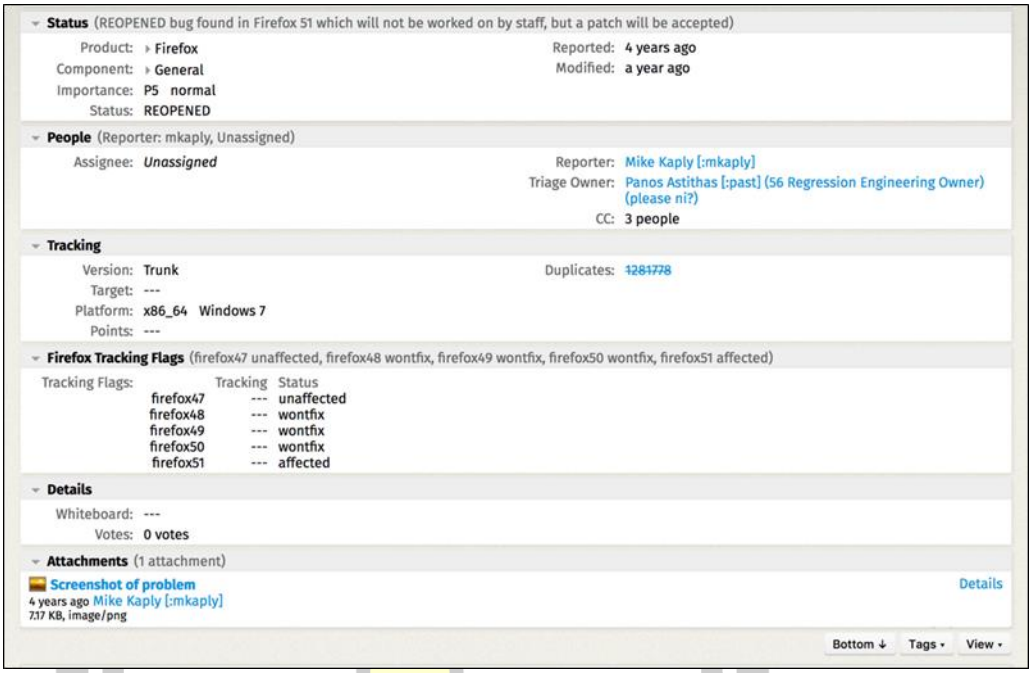
ภาพที่ 2-2 ส่วนสรุปของรายงานจุดบกพร่อง

ที่มา : (https://bugzilla.mozilla.org/show_bug.cgi?id=964821)

o หมายเลขของรายงานจุดบกพร่อง (Bug ID) เป็นการแสดงหมายเลขประจำตัวของรายงานจุดบกพร่อง ในแต่ละรายงาน ซึ่งจะเป็นหมายเลขที่ไม่ซ้ำกัน

o ชื่อเรื่อง (Title หรือ Summary) เป็นการแสดงการอธิบายเกี่ยวกับรายงานจุดบกพร่องที่พบอย่างสั้นๆ ในหนึ่งบรรทัด

2) ส่วนของรายละเอียด เป็นการอธิบายเกี่ยวกับจุดบกพร่องโดยละเอียดดังภาพที่ 2-3



ภาพที่ 2-3 ส่วนสรุปของรายงานจุดบกพร่อง

ที่มา : (https://bugzilla.mozilla.org/show_bug.cgi?id=964821)

โดยในแต่ละองค์กร หรือแต่ละซอฟต์แวร์จะมีการกำหนดรายละเอียดในส่วนนี้ที่แตกต่างกันออกไปตามแต่ละความต้องการของผู้พัฒนาในองค์กรนั้น ซึ่งโดยสำคัญแล้วจะประกอบไปด้วย

o ชื่อส่วนประกอบของซอฟต์แวร์ (Component) เป็นการแสดงคำเรียกของส่วนประกอบของซอฟต์แวร์ หรือฟังชันของซอฟต์แวร์ที่เกี่ยวข้องจุดบกพร่อง

o สถานะของรายงาน (Status) เป็นการแสดงคำ แสดงสถานะของจุดบกพร่องที่รายงานเข้าว่าว่าอยู่ในขั้นตอนใด ในรายงานของ Bugzilla จะประกอบไปด้วยสถานะของรายงานดังตารางที่ 2-1

ตารางที่ 2-1 สถานะของรายงานจุดบกพร่อง

สถานะของราย	คำอธิบาย
unconfirmed	เป็นสถานะแรกของรายงานที่ได้รับเมื่อส่งรายงานเข้าสู่ระบบ
new	เป็นสถานะของรายงานที่ผ่านการการคัดแยกความซ้ำซ้อนเบื้องต้นแล้ว
assigned	เป็นสถานะของรายงานที่ได้การมอบหมายให้ทางผู้พัฒนาในส่วนนั้น ของซอฟต์แวร์แล้ว
resolved	เป็นสถานะของรายงานที่ได้รับแก้ไขปรับปรุงจุดบกพร่องแล้ว
verified	เป็นสถานะของรายงานที่ผ่านการตรวจสอบการแก้ไขจุดบกพร่องแล้ว
reopen	เป็นสถานะของรายงานที่ได้รับการแก้ไขแล้ว แต่มียังพบว่ามีจุดบกพร่องอยู่

ในกรณีที่รายงานได้รับการแก้ไขแล้วจะมีการเพิ่มสถานะการแก้ไขปัญหา (Resolution) ซึ่งสถานะที่บ่งบอกผลที่ได้การแก้ไขจุดบกพร่องดังตารางที่ 2-2

ตารางที่ 2-2 สถานะการแก้ไขปัญหา

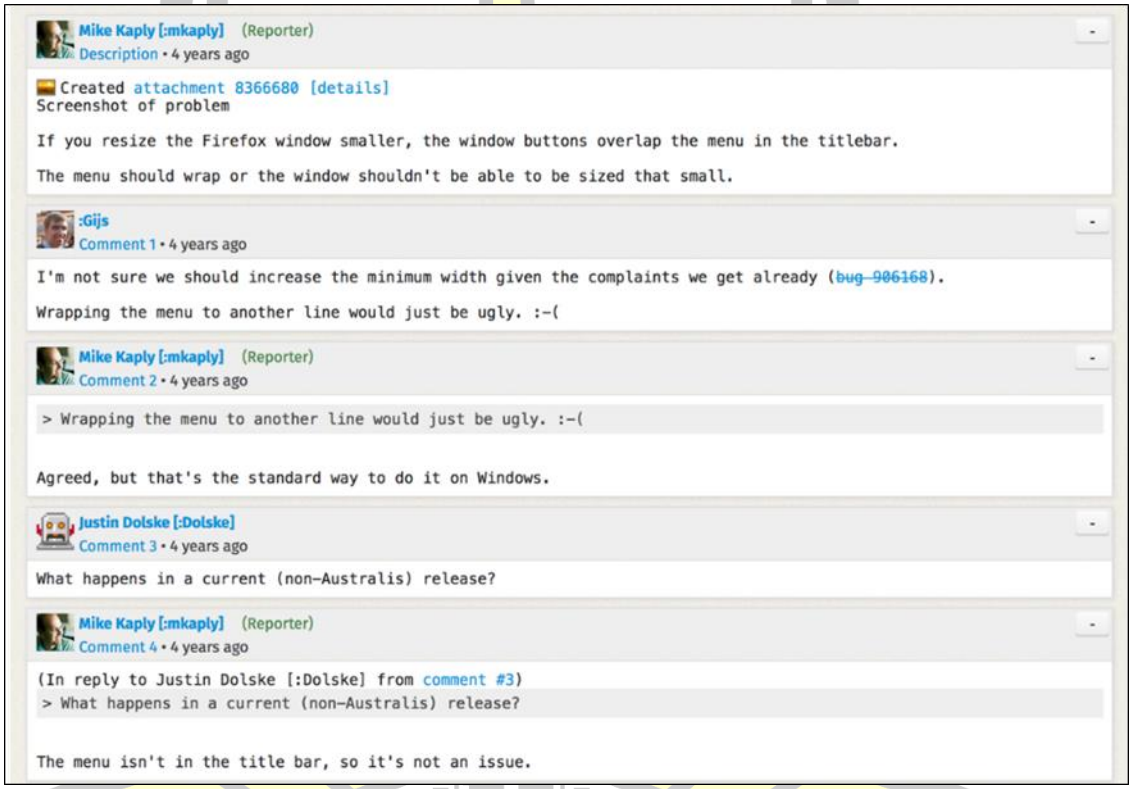
สถานะ	คำอธิบาย
Fixed	เป็นสถานะของรายงานจุดบกพร่องที่ได้รับการแก้ไขเรียบร้อยแล้ว
Wontfix	เป็นสถานะของรายงานจุดบกพร่องที่สามารถแก้ไขได้เนื่องจากเหตุผลบางประการ
Worksforme	เป็นสถานะของรายงานจุดบกพร่องที่ไม่สามารถทำใหม่ได้โดยผู้พัฒนาซอฟต์แวร์
Duplicate	เป็นสถานะของรายงานจุดบกพร่องที่มีการรายงานซ้ำซ้อน
Invalid	เป็นสถานะของรายงานจุดบกพร่อง

- ชื่อผู้รายงาน (Reporter) เป็นการแสดงชื่อ หรือนามสมมุติของผู้ที่รายงานจุดบกพร่องของซอฟต์แวร์
- ผู้พัฒนารับผิดชอบแก้ไข (Assignee) เป็นการแสดงชื่อของผู้พัฒนาที่มีหน้าที่รับผิดชอบในการแก้ไขจุดบกพร่องในส่วนประกอบนั้น ๆ ของซอฟต์แวร์
- รุ่นของซอฟต์แวร์ (Version) เป็นการแสดงรุ่นจองซอฟต์แวร์ที่พบจุดพบพร่องหรือรุ่นที่ได้รับผลกระทบจากจุดบกพร่อง

o ข้อมูลสภาพแวดล้อม (Environment) เป็นการแสดงรายละเอียดสภาพแวดล้อมที่พบจุดบกพร่อง อาทิ ระบบปฏิบัติการ สมรรถนะของเครื่องที่พบจุดบกพร่อง

o คำอธิบายเพิ่มเติม (Description) เป็นการแสดงรายละเอียด หรือคำอธิบายเกี่ยวกับจุดบกพร่องเพิ่มเติม

3) ส่วนของความคิดเห็นเพิ่มเติม เป็นการแสดงความคิดเห็น หรือข้อเสนอของผู้ที่ได้รับอนุญาตมีต่อรายงานจุดบกพร่องของซอฟต์แวร์ ดังภาพที่ 2-4 โดยคิดว่าคิดเห็นนั้นอาจเป็นการให้ข้อมูลเกี่ยวจุดบกพร่องเพิ่มเติม แนวทางในการแก้ไขจุดบกพร่องเบื้องต้น เป็นต้น

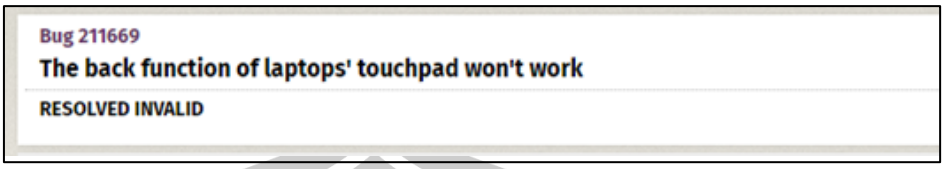


ภาพที่ 2-4 ส่วนของความคิดเห็นเพิ่มเติม

ที่มา : (https://bugzilla.mozilla.org/show_bug.cgi?id=964821)

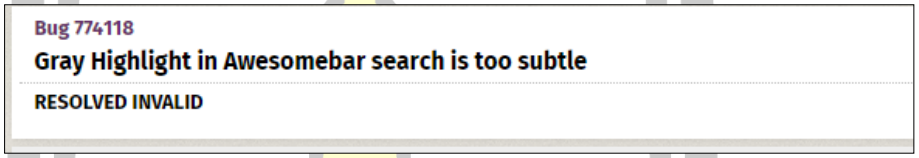
2.2.1 รายงานที่ไม่ใช่จุดบกพร่องของซอฟต์แวร์ (non-Bug Report)

ภายในคลังรายงานจุดพร่องรายงาน นอกจากรายงานจุดบกพร่องของซอฟต์แวร์ ยังพบการรายงานที่ไม่ใช่จุดบกพร่องของซอฟต์แวร์ หรือรายงานที่มีสถานะในวงจรชีวิตของรายงานจุดบกพร่องเป็น Invalid ซึ่งเป็นข้อบกพร่องที่ไม่ได้เกิดจากความผิดพลาดจากโค้ดของซอฟต์แวร์ (Source Code) แต่เป็นความผิดพลาดจากการใช้งานของผู้ใช้ หรือจากฮาร์ดแวร์ ดังภาพที่ 2-5 และอีกกรณีคือรายงานที่เป็นข้อเสนอแนะเกี่ยวกับซอฟต์แวร์ดังภาพที่ 2-5



ภาพที่ 2-5 รายงานจุดบกพร่องที่ได้เกิดความผิดพลาดจากฮาร์ดแวร์
ที่มา : (https://bugzilla.mozilla.org/show_bug.cgi?id=211669)

จากภาพที่ 2-5 เป็นการรายงานเกี่ยวกับการที่ไม่สามารถใช้งานฟังก์ชันของทัชแพด (Touchpad) โดยเหตุเกิดจากการไม่ที่ไม่ได้ติดตั้งไดร์เวอร์ (Driver) ของทัชแพด จึงทำให้ไม่สามารถใช้งานทัชแพดได้อย่างเต็มประสิทธิภาพ



ภาพที่ 2-6 รายงานจุดบกพร่องที่ได้เกิดความผิดพลาดจากฮาร์ดแวร์
ที่มา : (https://bugzilla.mozilla.org/show_bug.cgi?id=77418)

จากภาพที่ 2-6 เป็นรายงานข้อเสนอแนะเกี่ยวกับสีเทาที่ใช้ไฮไลท์ (highlight) คำตอบที่ได้จากการค้นหามีสีที่จางเกินไป

2.3 ทฤษฎีที่เกี่ยวข้อง

2.3.1 การเตรียมข้อมูลก่อนประมวลผล (Data Pre-processing)

การเตรียมข้อมูลก่อนประมวลผล เป็นขั้นตอนในการเตรียมเอกสารที่จะนำมาใช้ในการจัดกลุ่มเอกสาร ซึ่งมีกระบวนการดังนี้

1) การตัดคำ (Tokenization)

การตัดคำ หรือ การแบ่งคำ (Word Segmentation) ในเอกสารออกเป็นคำ ๆ เพื่อให้ได้คำที่จะเข้าสู่กระบวนการในการคัดเลือกคำสำคัญ เพื่อใช้เป็นคุณลักษณะสำคัญ (Feature) ในสำหรับนำไปการประมวลผลภาษา ซึ่งสามารถแบ่งเทคนิคการตัดคำเป็น 3 เทคนิค หลัก ๆ ดังนี้

(1) การตัดคำด้วยช่องว่าง (Space) เป็นเทคนิคการตัดคำ โดยใช้ช่องว่างระหว่างคำเป็นในการแบ่งคำออกเป็นคำ ๆ ซึ่งเป็นเทคนิคที่นิยมใช้ในภาษาอังกฤษ ดังในงานวิจัย [16, 19-22]

(2) การตัดคำด้วยพจนานุกรม (Dictionary-Based) เป็นการตัดคำโดยใช้สายอักขระมาเปรียบเทียบกับคำที่มีอยู่ในพจนานุกรมซึ่งมีวิธีนี้จะต้องทำการจัดเก็บคำไว้ในพจนานุกรม

ซึ่งทำให้ได้ความถูกต้องในการตัดคำสูงกว่าการใช้กฎ แต่ใช้เวลามากกว่าเทคนิคอื่น ๆ โดยเทคนิคที่ใช้ในการตัดคำด้วยพจนานุกรมที่มีนิยมใช้มีดังนี้ [23, 24]

(3) การตัดคำด้วยกฎ (Rule-Based) เป็นการตัดคำโดยการตรวจสอบกฎเกณฑ์ทางอักขระวิธีที่กำหนดลักษณะการประสมอักษร ลักษณะการเว้นวรรค และการขึ้นอยู่หน้า เพื่อใช้เป็นเกณฑ์ในการกำหนดขอบเขตของคำ วิธีการนี้จะมีข้อจำกัดในการทำงาน คือ ความถูกต้องของการตัดคำในระดับพยางค์สูง แต่ความถูกต้องของการตัดคำค่อนข้างต่ำ ซึ่งข้อดีของวิธีนี้คือ มีความรวดเร็วในการทำงาน และใช้ทรัพยากรน้อย

2) การตัดคำคาเมลเคส (Camel case)

คาเมลเคส [17, 25, 26] เป็นคำ หรือ ตัวย่อที่ตรงกลางวลีเริ่มต้นด้วยตัวพิมพ์ใหญ่ โดยไม่มีการเว้นวรรค หรือเครื่องหมายวรรคตอน จึงมีมีส่วนเว้าส่วนโค้งคล้ายอูฐ ตัวอย่างเช่น “iPhone”, “numberPhone”, "johnSmith" เป็นต้น โดยการตัดคำ Camel case เป็นการแยกคำที่อยู่ในรูป Camel case ออกจากกัน ดังตัวอย่าง

numberPhone	จะเปลี่ยนเป็น	number phone
HereComesTheGarden	จะเปลี่ยนเป็น	Here comes the garden

3) การกำจัดคำหยุด (Stop-Word Removal)

การกำจัดคำหยุด [16, 20, 27, 28] เป็นการนำคำที่ไม่มีนัยสำคัญออก โดยที่ไม่ทำให้ความหมายของคำหรือข้อความเปลี่ยนแปลง คำหยุดจะปรากฏอยู่ในทุก ๆ ข้อความ และมีความถี่ในการปรากฏสูง จึงถือได้ว่าคำหยุดเป็นคุณลักษณะที่ไม่เกี่ยวข้อง หรือไม่มีประโยชน์ในการประมวลผลทางภาษา ดังนั้นการกำจัดคำหยุดเป็นกระบวนการที่จะช่วยลดให้ขนาดของดัชนี และยังเวลาในการประมวลผลลง โดยคำหยุดที่พบได้แก่ คำบุพบท (Preposition) คำสรรพนาม (Pronouns) คำเชื่อม (Conjunction) คำนำหน้านาม (Articles)

คลังคำหยุดภาษาอังกฤษที่ของไลบรารี่ NLTK (Natural Language Toolkit) เป็นคลังคำหยุดที่ได้รับความนิยม [23, 28, 29] ซึ่งบรรจุคำหยุดไว้เป็นจำนวน 179 คำ ดังภาพที่ 2-7

a	by	hasn	just	other	their	when
about	can	hasn't	ll	our	theirs	where
above	couldn	have	m	ours	them	which
after	couldn't	haven	ma	ourselves	themselves	while
again	d	haven't	me	out	then	who
against	did	having	mightn	over	there	whom
ain	didn	he	mightn't	own	these	why
all	didn't	her	more	re	they	will
am	do	here	most	s	this	with
an	does	hers	mustn	same	those	won
and	doesn	herself	mustn't	shan	through	won't
any	doesn't	him	my	shan't	to	wouldn
are	doing	himself	myself	she	too	wouldn't
aren	don	his	needn	she's	under	y
aren't	don't	how	needn't	should	until	you
as	down	i	no	should've	up	you'd
at	during	if	nor	shouldn	ve	you'll
be	each	in	not	shouldn't	very	you're
because	few	into	now	so	was	you've
been	for	is	o	some	wasn	your
before	from	isn	of	such	wasn't	yours
being	further	isn't	off	t	we	yourself
below	had	it	on	than	were	yourselves
between	hadn	it's	once	that	weren	
both	hadn't	its	only	that'll	weren't	
but	has	itself	or	the	what	

ภาพที่ 2-7 คลังคำหยุดภาษาอังกฤษจากไลบรารี่ของ NLTK

4) การเปลี่ยนรูปคำ

การเปลี่ยนรูปคำการเปลี่ยนรูปคำ เป็นกระบวนการทำนอร์มัลไลซิ่งข้อความ (Normalizing Text) [23] ซึ่งมีผลต่อขนาดของคคุณลักษณะ (Feature) และมีผลต่อประสิทธิภาพในการจำแนกสรายงานจุดบกพร่อง โดยการเปลี่ยนรูปคำมี 2 เทคนิค ดังนี้

(1) การตัดส่วนขยาย (Stemming)

การหารากศัพท์ [23, 28] เป็นกระบวนการทำนอร์มัลไลซิ่งข้อความ โดยการตัดส่วนขยายของคำทิ้ง เช่น s, es, ing หรือ ed เป็นต้น ดังตัวอย่างเช่น

Hopes จะถูกเปลี่ยนเป็น hop

Hoping จะถูกเปลี่ยนเป็น hope

Hoped จะถูกเปลี่ยนเป็น hop

ซึ่งอัลกอริทึมที่นิยมสำหรับการตัดส่วนขยายของคำคือ Porter’s algorithm [30] ดังในงานวิจัย [17, 19]

(2) การเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม (Lemmatization)

การเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม [28, 31] เป็นกระบวนการทำนอร์มัลไลซิ่งข้อความโดยการเปลี่ยนคำให้อยู่ในรูปแบบคำดังเดิม [23] ซึ่งมีลักษณะคล้ายกับการตัดส่วนขยาย (Stemming) แต่การเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิมจะมีความละเดียวกว่าการตัดส่วนขยาย คือ มีการใช้บริบทของคำมาใช้ในการพิจารณาคำพ้องความหมายของคำที่ต้องการเปลี่ยน [23, 32] ตัวอย่างเช่น

walking	จะถูกเปลี่ยนเป็น	walk
meeting(v)	จะถูกเปลี่ยนเป็น	meet
meeting(n)	จะถูกเปลี่ยนเป็น	meeting
better	จะถูกเปลี่ยนเป็น	good

5) การสร้างตัวแทนเอกสาร (Document Representation)

การสร้างตัวแทนเอกสาร เป็นกระบวนการแปลงเอกสารที่เป็นภาษาธรรมชาติให้อยู่ในรูปแบบที่เหมาะสมและตรงกับรูปแบบที่จะประมวลผลต่อไป [23, 24] ซึ่งโดยทั่วไปจะอยู่ในรูปแบบของปริภูมิเวกเตอร์ หรือ เวกเตอร์สเปซ (Vector Space) หรือถุงคำ (Bag of word: BOW) ซึ่งมีลักษณะ ดังตารางที่ 2-3

ตารางที่ 2-3 รูปแบบของถุงคำ

	W_1	W_2	W_3	W_4	...	W_i
D_1	$W_{1,1}$	$W_{1,1}$	$W_{1,3}$	$W_{1,4}$	...	$W_{1,i}$
D_2	$W_{2,1}$	$W_{2,1}$	$W_{2,3}$	$W_{2,4}$	...	$W_{2,i}$
...	...	...	...	...	...	...
D_j	$W_{j,1}$	$W_{j,2}$	$W_{j,3}$	$W_{j,4}$	...	$W_{j,i}$

จากตารางที่ 2 3 แทนคำ i ที่พบในเอกสาร j ด้วย $W_{j,i}$ ดังนั้นจึงสามารถแทนด้วย $D_j = \{w_{j,1}, w_{j,2}, w_{j,3}, ..., w_{j,i}\}$ ซึ่งโดยทั่วไปคำใดที่ปรากฏในเอกสารจะแทนด้วย 1 (Presence) และหากไม่พบคำนั้นในเอกสารจะแทนด้วย 0 (Absence)

6) การให้น้ำหนักคำ (Team Weighting)

การให้น้ำหนักคำ หรือการกำหนดน้ำหนักคำ เป็นวิธีการให้น้ำหนักสำหรับคำที่มีความสำคัญ หรือใช้เป็นตัวแทนของเอกสารที่ ซึ่งจะเป็นคำปรากฏอยู่เป็นจำนวนมากในเนื้อหาเฉพาะของเอกสารฉบับนั้น ๆ และปรากฏอยู่เป็นจำนวนน้อยในเนื้อหาของเอกสารอื่น ๆ แต่ถ้าคำนั้น ๆ

ปรากฏอยู่เป็นจำนวนมากในทุก ๆ เอกสาร คำนั้นจะถือได้ว่าเป็นคำที่ไม่สามารถเป็นตัวแทนของเอกสารใด ๆ ได้ ตัวอย่างเช่น a, an, and, the ซึ่งคำเหล่านี้มีชื่อเรียกว่า “คำหยุด (Stop Word)”

ซึ่งในงานวิจัยนี้ได้ใช้เทคนิคการให้น้ำหนัก 4 เทคนิค คือ การให้น้ำหนักด้วยเทคนิค TF การให้น้ำหนักด้วยเทคนิค TF-IDF การให้น้ำหนักด้วยเทคนิค BM25 และการให้น้ำหนักด้วยเทคนิค MATF เพื่อเทคนิคการให้น้ำหนักที่มีความเหมาะสมในการจำแนกรายงานจุดบกพร่อง

(1) การให้น้ำหนักด้วยความถี่ของคำที่พบ (Term Frequency: TF) เป็นเทคนิคการให้น้ำหนักด้วยความถี่ของการปรากฏ (Occurrence) ของคำ ซึ่งใช้สัญลักษณ์ $tf_{t,d}$ โดยที่ t หมายถึง คำ และ d หมายถึงเอกสาร โดยเทคนิคนี้มีชื่อเรียกอีกชื่อหนึ่งว่า “ถุงของคำ (Bag of words)” [19, 22, 23, 29]

(2) การให้น้ำหนักด้วยเทคนิค TF-IDF เป็นเทคนิคการให้น้ำหนักคำที่ได้พัฒนามาจากการให้น้ำหนักด้วยเทคนิค tf โดยมีการคำนวณค่าน้ำหนัก ซึ่งการให้น้ำหนักซึ่งเป็นเทคนิคที่ได้รับการนิยม และใช้กันอย่างแพร่หลาย เนื่องจากเป็นเทคนิคที่มีการคำนวณที่ไม่ซับซ้อน และมีประสิทธิภาพสูง ดังในงานวิจัย [1, 16, 17, 35] โดยสามารถคำนวณได้ดังสมการที่ (2.1)

$$TF-IDF_{t,d} = tf \times idf \tag{2.1}$$

ซึ่งความถี่เอกสารแบบผกผัน (Inverse document frequency: IDF) เป็นการให้น้ำหนักด้วยความถี่เอกสารแบบผกผัน เป็นเทคนิคการปรับค่าน้ำหนักของคำให้มีความเหมาะสมมากขึ้น โดยใช้ความถี่ของเอกสาร (Document frequency: df) หรือจำนวนเอกสารทั้งหมดในฐานข้อมูลที่มีคำ t ปรากฏมาพิจารณาประกอบ ด้วยการปรับให้คำที่ปรากฏในเอกสารจำนวนน้อยมีค่าน้ำหนักสูงขึ้น และคำที่มีปรากฏในเอกสารจำนวนมากมีค่าน้ำหนักลดลง [40] ซึ่งคำนวณได้จากสมการที่ (2.2)

$$idf = \log \frac{N}{df_t} \tag{2.2}$$

โดยที่ N คือ จำนวนเอกสารทั้งหมดที่มีในฐานข้อมูล

(3) การให้น้ำหนักด้วยเทคนิค Best Matching หรือ BM25 เป็นเทคนิคให้น้ำหนักคำสำหรับการจัดลำดับเอกสารในการค้นคืนเอกสาร (Information Retrieval) [1, 41] โดยมีพื้นฐานมาจากการเทคนิคการให้น้ำหนักแบบ TF-IDF [41] ซึ่งสามารถคำนวณได้ดังสมการ (2.3)

$$BM25(q_i, d) = \left(\frac{f_{i,j} \times (k_1 + 1)}{f_{i,j} + k_1 \left((1-b) + b \left(\frac{dl}{dl_{avg}} \right) \right)} \right) \times \log \left(\frac{N - n + k}{n + k} \right) \tag{2.3}$$

โดยที่ $f_{i,j}$ คือ ความถี่ของคำ I ที่ปรากฏในเอกสาร j

b คือ ค่าพารามิเตอร์สำหรับนอร์มัลไลเซชัน (Normalization) ของความยาวเอกสาร

k1 คือ ค่าพารามิเตอร์ปรับเรียบ (Smoothing) สำหรับการปรับความอิ่มตัวของค่าความถี่ของคำ

dl คือ ความยาวของเอกสารนั้น ๆ

dl$_{ave}$ คือ ค่าเฉลี่ยความยาวเอกสาร

N คือ จำนวนเอกสารทั้งหมด

n คือ จำนวนเอกสารที่มีคำที่ i อยู่

(4) การให้น้ำหนักด้วยเทคนิค Multi Aspect TF หรือ MATF [42] เป็นการให้น้ำหนักคำในเอกสารที่ได้รับการพัฒนา และนำเสนอโดยนาเสนอโดย Jiaul H. Paik [42] ซึ่ง MATF เป็นเทคนิคที่มุ่งเน้นการแก้ไขปัญหาของการให้น้ำหนักของเอกสารที่มีความยาวของข้อความที่ไม่เท่ากันในการเปรียบเทียบความคล้ายคลึงของเอกสาร ซึ่งสามารถคำนวณได้จากสมการที่ (2.4)

$$MATF(w,D) = TFF \times TDF \tag{2.4}$$

Term Frequency Factor หรือ TFF เป็นขั้นตอนที่รวมให้น้ำหนักที่มีพื้นฐานมาจาก TF ซึ่งสามารถคำนวณได้จากสมการที่ (2.5)

$$TFF(t,D) = w \times BRITF(t,D) + (1-w) \times BLRTF(t,D) \tag{2.5}$$

โดยที่ w, BRITF และ BLRTF สามารถคำนวณได้จากสมการที่ (2.6-2.8) ตามลำดับ

$$BRITF(t,D) = \frac{RITF(t,D)}{1-RITF(t,D)} \tag{2.6}$$

$$BLRTF(t,D) = \frac{LRTF(t,D)}{1-LRTF(t,D)} \tag{2.7}$$

$$w = \frac{2}{1+\log(1+|Q|)} \tag{2.8}$$

Relative Intra-document หรือ RITF เป็นการพิจารณาความสำคัญของคำจากความสัมพันธ์ระหว่างความถี่ของคำในเอกสาร และความถี่ของคำโดยเฉลี่ยของเอกสาร เมื่อแทนคำด้วย “t” และแทนเอกสารด้วย “D” สามารถคำนวณ RITF(t, D) ได้จากสมการ (2.9)

$$RITF(t,D) = \frac{\log_2(1+TF(t,D))}{\log_2(1+Avg.TF(D))} \tag{2.9}$$

โดยที่ TF(t, D) คือ ความถี่ของคำ t ที่พบในเอกสาร D

Avg.TF(t, D) คือ ค่าเฉลี่ยความถี่ของคำทั้งหมดในเอกสาร D

Length Regularized TF หรือ LRTF เป็นการนอร์มัลไลความถี่ของคำโดยพิจารณาจากจำนวนของคำที่มีอยู่ในเอกสาร โดยใช้หลักการที่ว่า “ความยาวที่เหมาะสมของเอกสารควรเป็นค่าเฉลี่ยของคลังเอกสาร และความยาวเฉลี่ยของความถี่คำในเอกสารยังคงเหมือนเดิม” โดย LRTF สามารถคำนวณได้จากสมการที่ (2.10)

$$LRTF(t,D) = TF(t,D) \times \log_2\left(1 + \frac{ADL(C)}{len(D)}\right) \quad (2.10)$$

โดยที่ ADL(C) คือ ค่าความยาวเฉลี่ยของเอกสาร

Len(D) คือ ค่าความยาวของเอกสาร D

Term Discrimination Factor หรือ TDF เป็นขั้นตอนที่รวมให้น้ำหนักที่มีพื้นฐานมาจากความถี่ผกผัน (IDF) ซึ่งสามารถคำนวณได้จากสมการที่ (2.11)

$$TDF(t,D) = IDF(t,D) \times \frac{AEF(t,C)}{1 - AEF(t,C)} \quad (2.11)$$

โดยที่ IDF และ AEF สามารถคำนวณได้จากสมการ (2.12-2.13) ตามลำดับ

$$IDF(t,D) = \log\left(\frac{CS(C)+1}{DF(t,C)}\right) \quad (2.12)$$

โดยที่ CS(S) คือ จำนวนเอกสารทั้งหมดในคลัง

DF(t, C) คือ จำนวนของเอกสารที่พบคำ t

$$AEF(t,C) = \frac{CTF(t,C)}{DF(t,C)} \quad (2.13)$$

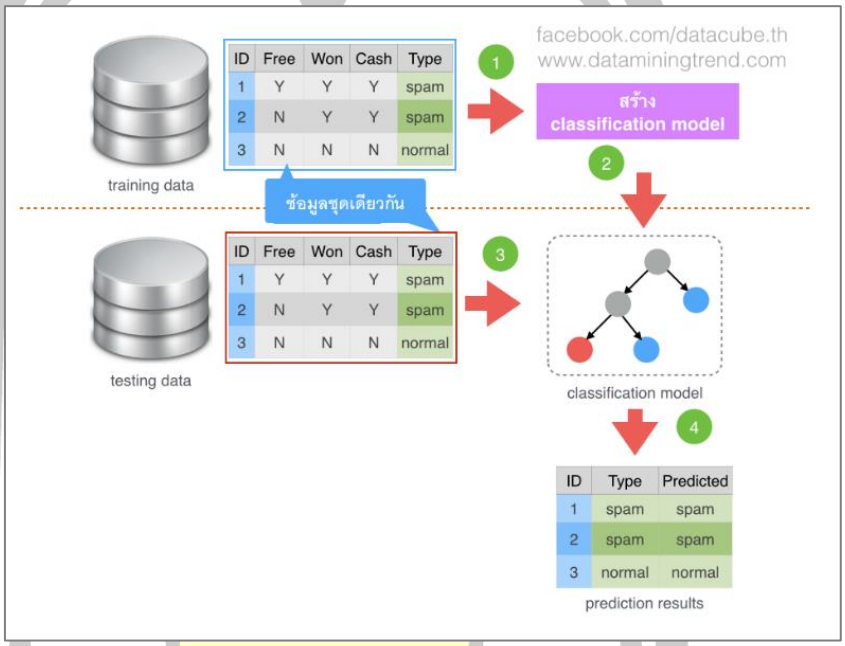
โดยที่ CTF(t, C) คือ จำนวนครั้งที่ปรากฏคำ t ในคลังเอกสาร

2.3.2 การแบ่งข้อมูลสำหรับสร้างแบบจำลอง และข้อมูลมูลทดสอบ

การแบ่งข้อมูลสำหรับสร้างแบบจำลอง และข้อมูลสำหรับทดสอบ เป็นการแบ่งข้อมูล เพื่อทำการทดสอบสอบประสิทธิภาพของแบบจำลอง โดยสามารถจำแนกได้ 3 วิธี ดังนี้

1) Self Consistency Test หรือ Use Training Set เป็นวิธีการแบ่งข้อมูลที่ง่ายที่สุด โดยการใช้ในการสร้างแบบจำลอง และข้อมูลที่ใช้ในการทดสอบโมเดลเป็นข้อมูลชุดเดียวกัน ซึ่งเทคนิคนี้เริ่มจากการสร้างแบบจำลองด้วยข้อมูลที่ใช้สำหรับสร้างแบบจำลอง (Training Data) และนำแบบจำลองที่ได้มาใช้ทดสอบกับข้อมูลชุดเดิม ตัวอย่างดังภาพที่ 2 8

การแบ่งข้อมูลด้วยวิธี Self Consistency Test เป็นการแบ่งข้อมูลที่ให้ประสิทธิภาพที่มีค่าสูง แต่อาจเกิดปัญหา Overfitting ได้ เนื่องจากใช้ข้อมูลเดิมทั้งการสร้างแบบจำลอง และการทดสอบ ดังนั้นวิธี Self Consistency Test จึงเหมาะสำหรับใช้ในการทดสอบประสิทธิภาพ เพื่อดูแนวโน้มขอแบบจำลองที่สร้างขึ้นเท่านั้น หากมีค่าประสิทธิภาพที่ต่ำ แสดงว่าแบบจำลองที่สร้างขึ้นไม่เหมาะสมกับข้อมูล และไม่ควรนำไปทดสอบด้วยวิธีการแบ่งข้อมูลวิธีอื่น

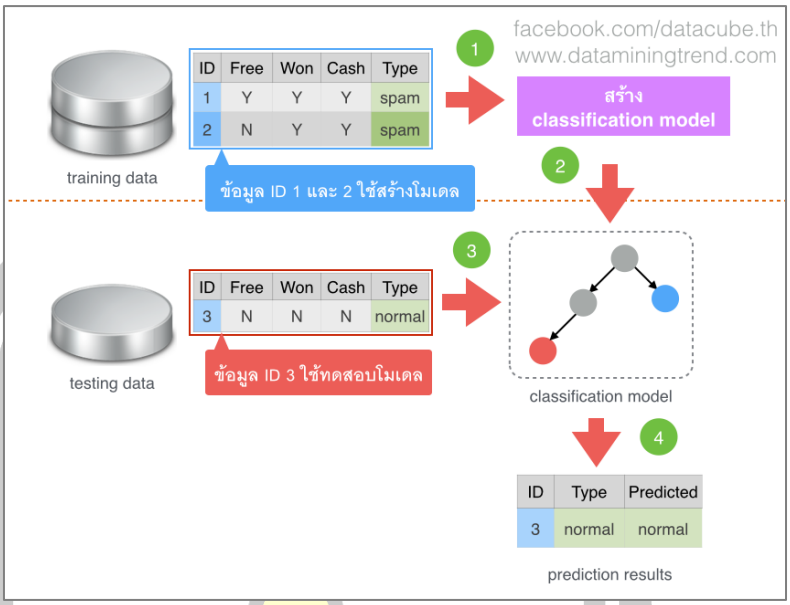


ภาพที่ 2-8 การแบ่งข้อมูลด้วยวิธี Self Consistency Test

ที่มา : (http://dataminingtrend.com/2014/wp-content/uploads/2015/09/train_test.001.png)

2) วิธี Split Test เป็นการแบ่งข้อมูลออกเป็น 2 ส่วน คือ ข้อมูลสำหรับสร้างแบบจำลอง และข้อมูลสำหรับทดสอบด้วยการสุ่ม ตัวอย่างเช่น 70% ต่อ 30% หรือ 80% ต่อ 20% โดยข้อมูลส่วนหนึ่ง (70% หรือ 80%) เป็นข้อมูลที่ใช้ในการสร้างแบบจำลอง และข้อมูลส่วนอีกส่วน (30% หรือ 20%) เป็นข้อมูลสำหรับใช้ในการทดสอบประสิทธิภาพของแบบจำลอง ตัวอย่างดังภาพที่ 2-9

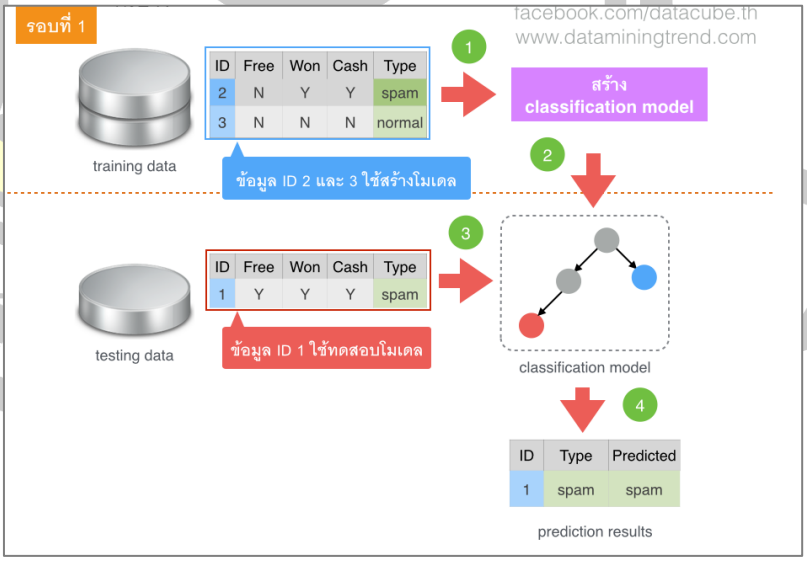
การแบ่งข้อมูลด้วยวิธี Split Test เป็นการแบ่งข้อมูลที่รวดเร็ว และเหมาะสมกับชุดข้อมูลที่มีขนาดใหญ่ เนื่องจากเป็นการสุ่มแบ่งข้อมูลเพียงครั้งเดียว ซึ่งหากการสุ่มแบ่งข้อมูลที่ใช้สำหรับการสร้างแบบจำลอง และข้อมูลสำหรับทดสอบได้เป็นข้อมูลที่มีลักษณะคล้ายกันจะให้ค่าประสิทธิภาพสูง แต่ในทางกลับกันหากการสุ่มแบ่งข้อมูลที่ใช้สำหรับการสร้างแบบจำลอง และข้อมูลสำหรับทดสอบได้เป็นข้อมูลที่มีลักษณะแตกต่างกันจะให้ค่าประสิทธิภาพที่น้อย ดังนั้นจึงควรใช้วิธี Split Test โดยทำการสุ่มหลายครั้ง



ภาพที่ 2-9 การแบ่งข้อมูลด้วยวิธี Split Test

ที่มา : (http://dataminingtrend.com/2014/wp-content/uploads/2015/09/train_test.002.png)

3) Cross-validation Test เป็นวิธีที่นิยมใช้ในการแบ่งข้อมูลในงานวิจัย โดยใช้แบ่งข้อมูล เพื่อของแบบจำลอง [43]โดยวิธีนี้จะแบ่งข้อมูลอออกเป็นจำนวนหลายส่วน ซึ่งนิยมแสดงด้วยค่า k จึงมีอีกชื่อว่า k-Fold Cross Validation ตัวอย่างเช่น k = 5 หรือ 5-fold cross validation คือ การแบ่งข้อมูลออกเป็น 5 ส่วนเท่าๆ กัน จากนั้นใช้ข้อมูลหนึ่งส่วนเป็นข้อมูลทดสอบประสิทธิภาพของแบบจำลอง และส่วนที่เหลือในการสร้างแบบจำลอง โดยจะทำวนเช่นนี้ไปจนครบตามจำนวนที่แบ่งไว้ ดังภาพที่ 2 8



ภาพที่ 2 10 การแบ่งข้อมูลด้วยวิธี Cross-validation Test

ที่มา : (http://dataminingtrend.com/2014/wp-content/uploads/2015/09/train_test.003.png)

จากภาพที่ 2 10 ข้อมูลได้ถูกแบ่งออกเป็น 5 ส่วนที่เท่าๆ กัน แล้วจึงทำการทำการทดสอบประสิทธิภาพของแบบจำลอง 5 ครั้ง ดังนี้ตารางที่ 2-4

ตารางที่ 2-4 ตัวอย่างการแบ่งข้อมูลด้วยวิธี 5 fold Cross validation

รอบที่ 1	ใช้ข้อมูลส่วนที่ 2,3,4 และ 5 สร้างโมเดลและใช้โมเดลทำนายข้อมูลส่วนที่ 1 เพื่อทำการทดสอบ
รอบที่ 2	ใช้ข้อมูลส่วนที่ 1,3,4 และ 5 สร้างโมเดลและใช้โมเดลทำนายข้อมูลส่วนที่ 2 เพื่อทำการทดสอบ
รอบที่ 3	ใช้ข้อมูลส่วนที่ 1,2,4 และ 5 สร้างโมเดลและใช้โมเดลทำนายข้อมูลส่วนที่ 3 เพื่อทำการทดสอบ
รอบที่ 4	ใช้ข้อมูลส่วนที่ 1,2,3 และ 5 สร้างโมเดลและใช้โมเดลทำนายข้อมูลส่วนที่ 4 เพื่อทำการทดสอบ
รอบที่ 5	ใช้ข้อมูลส่วนที่ 1,2,3 และ 4 สร้างโมเดลและใช้โมเดลทำนายข้อมูลส่วนที่ 5 เพื่อทำการทดสอบ

2.3.3 การจำแนกประเภทเอกสารข้อความ (Text Classification)

การจำแนกประเภทเอกสารข้อความ [30, 35, 36, 40] เป็นการจัดแบ่งประเภทของกลุ่มเอกสารข้อความออกเป็นกลุ่มๆ (Class หรือ Category) ซึ่งในงานวิจัยนี้จำแนกรายงานจุดบกพร่องออกเป็น 2 กลุ่ม ได้แก่ กลุ่มของรายงานที่เป็นจุดบกพร่อง และกลุ่มของรายงานที่ไม่ใช่จุดบกพร่อง โดยการใช้ชุดข้อมูลตัวอย่างของเอกสารข้อความที่เรียกว่า ชุดข้อมูลเพื่อการเรียนรู้ (Training Set) สำหรับสร้างแบบจำลองของการจำแนกเอกสาร (Classifier Model) และชุดข้อมูลทดสอบ (Test Set) สำหรับทดสอบประสิทธิภาพของโมเดลการจำแนกที่สร้างขึ้น ซึ่งในงานวิจัยนี้ได้ศึกษาเทคนิคการเรียนรู้จากชุดข้อมูลเรียนรู้ในการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่องแบบอัตโนมัติ 4 เทคนิค ประกอบ นาอีฟเบย์ ซัพพอร์ตเวกเตอร์แมชชีน การวิเคราะห์ถดถอยโลจิสติกส์ และแรดดอมฟอเรส

1) นาอีฟเบย์ (Naïve Bayes)

การเรียนรู้ของเบย์อย่างง่าย [30, 35, 40] เป็นวิธีจำแนกประเภทข้อมูลที่มีประสิทธิภาพวิธีหนึ่ง เหมาะกับกรณีของข้อมูลตัวอย่างจำนวนมาก และมีคุณสมบัติ (Attribute) ของตัวอย่างไม่ขึ้นต่อกัน ดังนั้นเทคนิคการเรียนรู้ของเบย์อย่างง่ายจึงได้รับความนิยมนำไปประยุกต์ใช้งานในด้านการจำแนกประเภทข้อความ (Text Classification) อีกทั้งยังเป็นเทคนิคที่ทำงานไม่ซับซ้อมเมื่อเทียบกับวิธีการอื่น ดังในงานวิจัย [16, 26-28, 34, 44]

กำหนดให้ความน่าจะเป็นของข้อมูลที่จะเป็นกลุ่ม v_j สำหรับข้อมูลที่มีสมบัติ n ตัว หรือใช้สัญลักษณ์ว่า $P(a_1, a_2, ...,an +| v_j)$ ดังสมการที่ (2.14)

$$P(a_1, a_2, a_3, ..., a_n \mid v_j) = \prod_{i=1}^{n} P(a_i \mid v_j) \quad (2.14)$$

โดยที่ $\prod$ หมายถึง ผลคูณของค่า $P(a_i \mid v_j)$ ทั้งหมด

i หมายถึง 1, 2, 3, ..., n

j หมายถึง 1, 2, 3, ..., n

การนำวิธีการเรียนรู้ของเบย์อย่างง่ายไปใช้ มีวิธีการดังต่อไปนี้ คือ

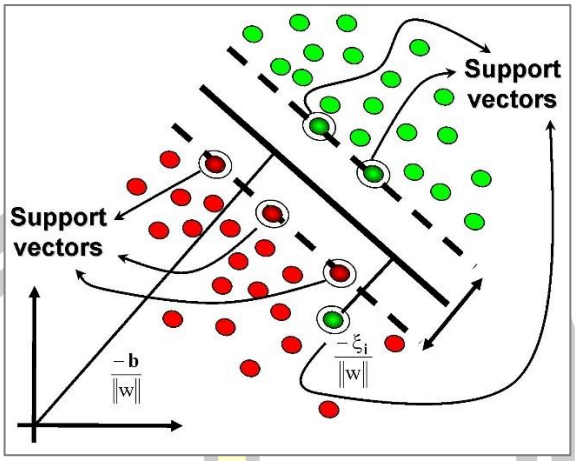
1. หาค่าความน่าจะเป็นของคำที่พบในแต่ละกลุ่มโดยนำค่า $P(a_1, a_2, ...,a_n \mid v_j)$ จากสมการที่ (2.14) มาคูณกับค่าความน่าจะเป็นของกลุ่มนั้น ๆ คือ $P(v_j)$ ได้เท่ากับ v_{NB}

2. นำค่าที่ได้มาเปรียบเทียบกัน กลุ่มที่มีค่าความน่าจะเป็นสูงสุดจะเป็นคำตอบ ดังนั้นเราจะได้วิธีการจำแนกประเภทแบบเบย์อย่างง่าย ดังสมการที่ (2.15)

$$v_{NB} = Avg.MaxP(v_j) \quad (2.15)$$

2) ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine: SVM)

ซัพพอร์ตเวกเตอร์แมชชีน เป็นวิธีการจำแนกกลุ่มข้อมูลที่มีลักษณะเป็นเวกเตอร์ ได้รับการพัฒนาโดย Vapnik ซึ่งซัพพอร์ตเวกเตอร์แมชชีนได้รับความนิยมนำไปปประยุกต์ใช้ในด้านการจำแนกประเภทข้อมูล โดยเฉพาะการจำแนกข้อมูลที่มี 2 คลาส (Two-Class) โดยแนวคิดของซัพพอร์ตเวกเตอร์แมชชีน เป็นการหาเส้นแบ่งที่เหมาะสมที่สุดที่จะใช้ในการแยกข้อมูลสองชุดออกจากกัน (Hyperplane) และหาเส้นตรงที่เหมาะสมในการแบ่งข้อมูลสองกลุ่มออกจากกัน จึงอาศัยการพิจารณาจากเส้นขอบ (Margin) มากที่ระยะห่างสุดที่เรียกว่า Maximum-Margin Hyperplane โดยเส้นขอบจะขยายออกไปจนกว่าจะสัมผัสกับค่าของกลุ่มตัวอย่างที่ใกล้ที่สุด ซึ่งเรียกว่า ซัพพอร์ตเวกเตอร์ (Support Vector) ดังแสดงในภาพที่ 2 11

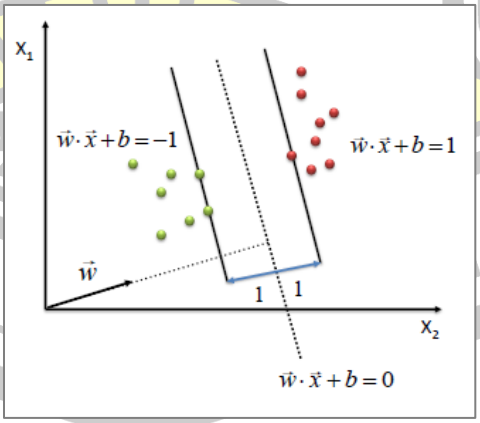


ภาพที่ 2-10 การขยายตัวของเส้นขอบ

ที่มา : (http://4.bp.blogspot.com/-ZRdRvcO0APg/TpFojq12IcI/AAAAAAAAAAU/1ka-fmRXApw/s1600/SVM.JPG)

ซัพพอร์ตเวกเตอร์แมชชีนสามารถแบ่งออกเป็น 2 ประเภท คือ แบบเชิงเส้น (Linear) และแบบไม่เป็นเชิงเส้น (Non-Linear)

(1) ซัพพอร์ตเวกเตอร์แมชชีนแบบเชิงเส้น (Support vector machine with Linear) มีแนวคิด คือ กลุ่มข้อมูลที่นำมาวางลงในคคุณลักษณะสเปซ (Feature Space) จะอยู่ในรูปของเวกเตอร์ ซึ่งกำหนดให้ X =((x_1,y_1), (x_2,y_2), ..., (x_n, y_n)) โดยที่ X คือ ชุดค่าคุณลักษณะ ซึ่งค่าคุณลักษณะที่อยู่ในคคุณลักษณะสเปซนั้นจะถูกแบ่งด้วยเส้นตรง ดังแสดงในภาพที่ 2 11 ในการคำนวณหาสมการเส้นตรงจากเส้นแบ่งข้อมูล จะแทนค่าด้วยสมการเส้นตรง y = mx + b ซึ่งจะมีการกำหนดกลุ่มของข้อมูลทั้งสองฝั่งให้เพียง 2 ค่า โดยแทนด้วย y ดังแสดงในภาพที่ 2-11



ภาพที่ 2-11 การวางตัวของเส้นขอบและเส้นแบ่งเมื่อแทนด้วยสมการเส้นตรง

ที่มา : (https://nssauci.com/image/76612-full_support-vector-machine.png)

จากภาพที่ 2-11เมื่อแทนเส้นขอบ และเส้นแบ่งด้วยสมการเส้นตรง จะเห็นว่าเส้นขอบทั้งสองฝั่งขยายเพิ่มขึ้นสัมผัสกับค่าข้อมูลในคคุณลักษณะสเปซที่ใกล้ที่สุด เส้นขอบที่เพิ่มขึ้นทางด้านขวาแทนด้วยสมการ $w^{T}x + b \geq y \geq 1$ หากค่าข้อมูล y มากกว่า 1 จะถูกกำหนดค่าให้ใหม่โดยให้ y เท่ากับ 1 ส่วน w เป็นค่าความชัน เช่นเดียวกับเส้นขอบทางด้านซ้าย ซึ่ง แทนด้วยสมการ $w^{T}x + b \leq y \leq -1$ หากค่าของข้อมูล y มีค่าน้อยกว่า -1 จะถูกกำหนดค่าให้ใหม่โดยให้ y เท่ากับ -1 ดังนั้นจะได้สมการของเส้นขอบทางด้านขวาดังแสดงในสมการที่ (2.16) และสมการของเส้นขอบทางด้านซ้ายดังดังแสดงในสมการที่ (2.17) และการรวมกันของทั้งสองสมการสามารถแสดงดังสมการที่ (2.18)

$$w^T x+b \geq y \text{ เมื่อกำหนดให้ } y=1 \tag{2.16}$$

$$w^T x+b \leq y \text{ เมื่อกำหนดให้ } y=-1 \tag{2.17}$$

$$y(w^T x+b)-1 \geq 0 \tag{2.18}$$

โดยที่ y คือ ค่ากลุ่มข้อมูล (1,-1) w คือ ค่าความชัน x คือ ค่าลักษณะเด่น และ b คือค่าคงที่ (ค่าตัดแกน y)

เมื่อได้สมการที่ (2.18) แล้ว ต้องคำนวณหาค่าความกว้างของเส้นขอบ ซึ่งต้องคำนวณพจน์ w ให้อยู่ในรูปนอมัลไลซ์ (Normalization) โดยคำนวณจากสมการที่ (2.16) และสมการที่ (2.17) โดยการแทนค่า y

$$w^T x^+ +b=1 \tag{2.19}$$

$$w^T x^- +b=-1 \tag{2.20}$$

$$w^T \left(x^+ + x^-\right)=2 \tag{2.21}$$

$$M=\left(\frac{w}{\|w\|}\right)^T \left(x^+ - x^-\right) \tag{2.22}$$

$$M=\left(\frac{2}{\|w\|}\right) \tag{2.23}$$

โดยที่ M คือความกว้างของเส้นขอบ และภายหลังจากคำนวณหาเส้นแบ่งและค่าความกว้างตามสมการที่ (2.18) และสมการที่ (2.23) ตามลำดับแล้ว จากนั้นทำการแก้สมการต่อด้วยฟังก์ชันลากรานจ์ (Lagrangian) เพื่อหาค่าของ w ในสมการที่ (2.18)

$$L(w,b,a)=\frac{1}{2}(w.w)-\sum_{i-1}^{N}\alpha_i\left[y_i\left(\left(w_i.w_i\right)+b\right)-1\right] \tag{2.24}$$

โดยที่ $\alpha_i \geq 0; i=1,2,\ldots,N$

สมการที่ (2-24) ถูกนำมาคำนวณหาอนุพันธ์ (Differential) เพื่อหาค่าน้ำหนักที่ดีที่สุด (w)

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{N} y_i \alpha_i x_i = 0 \tag{2.25}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = w - \sum_{i=1}^{N} y_i \alpha_i = 0 \tag{2.26}$$

หลังจากได้สมการที่ผ่านการแก้ปัญหาด้วยฟังก์ชันลากรานจ์แล้วค่า w จะถูกลดรูปซึ่งคำนวณได้จากสมการที่ (2-27)

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{2.27}$$

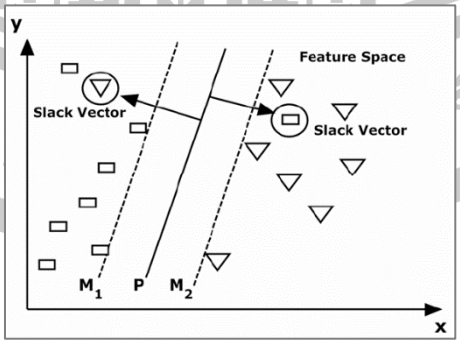
เนื่องจากค่า b ไม่ปรากฏในการแก้ปัญหาด้วยลากรานจ์ ดังนั้นจึงสามารถหาได้จากสมการที่ (2-28)

$$b = -\frac{\max_{y_i=1}(w^T x_i) + \min_{y_i=1}(w^T x_i)}{2} \tag{2.28}$$

จากนั้นนำค่า w ที่แก้ไขปัญหาไว้ไปแทนในสมการที่ (2-18) ซึ่งเป็นสมการในการหาเส้นแบ่งจากสมการที่ (2-29) เป็นการหาค่าสัมประสิทธิ์ α เพื่อนำมาใช้บอกการวางตัวของเส้นแบ่งข้อมูลต่อไป

$$y_i\left(\sum_{i=1}^{N} \alpha_i y_i (x^T x_i) + b\right) - 1 \geq 0 \tag{2.29}$$

การคำนวณซัพพอร์ตเวกเตอร์แมชชีนที่กล่าวมาข้างต้นเป็นการคำนวณในกรณีที่ข้อมูลไม่เกิดข้อผิดพลาด ซึ่งอาจเกิดข้อผิดพลาดในการแบ่งข้อมูล เนื่องจากบางครั้งอาจมีข้อมูลบางตัวที่วางอยู่ผิดกลุ่มออกไป ทำให้เกิดข้อผิดพลาดในการแบ่งข้อมูลได้ การแก้ปัญหานี้คือ การยอมให้มีการเกิดข้อผิดพลาดนี้ได้ในระยะหนึ่งที่ยอมรับได้ ซึ่งค่านั้นเรียกว่า เวกเตอร์อนุโลม (Slack Vector) ดังแสดง



ภาพที่ 2-12 การเกิดเวกเตอร์อนุโลม (Slack Vector)

เมื่อนำค่าเวกเตอร์อนุโลมเข้ามาในสมการที่ (2-18) ซึ่งเป็นขั้นการแก้ปัญหาด้วยลากรานจ์ทำให้เกิดตัวแปร C ขึ้นซึ่งก็เป็นค่าคงที่ค่าหนึ่ง ดังสมการที่ (2-30)

$$L(w,b,\xi,\alpha)=\frac{1}{2}(w.w)+\frac{C}{2}\sum_{i=1}^{N}\xi_i^2-\sum_{i=1}^{N}\alpha_i\left[y_i\left(\left(w_i.w_i\right)+b\right)-1+\xi_i\right] \quad (2.30)$$

จากนั้นสมการที่ (2-30) ถูกนำมาหาอนุพันธ์ (Differential) เช่นเดียวกับขั้นตอนในวิธีการแก้สมการที่ (2-18)

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial w}=w-\sum_{i=1}^{N}y_i\alpha_i x_i=0 \quad (2.31)$$

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial \xi}=C\xi-\alpha=0 \quad (2.32)$$

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial b}=w-\sum_{i=1}^{N}y_i\alpha_i=0 \quad (2.33)$$

โดยที่ ξ คือ เวกเตอร์อนุโลม ซึ่งเมื่อได้สมการที่ผ่านการแก้ปัญหาด้วยลากรานจ์แบบมีค่าเวกเตอร์อนุโลมแล้ว ค่า w ยังคงเป็นเช่นเดิมแต่จะได้ขอบเขตของ α ขึ้นมา ดังสมการที่ (2-34)

$$w=\sum_{i=1}^{N}\alpha_i y_i x_i \quad (2.34)$$

เมื่อ $C \geq \alpha \geq 0$ ดังนั้นค่า C จึงเป็นพารามิเตอร์อีกค่าหนึ่งที่จะต้องกำหนดขึ้นมาเพื่อใช้หาค่าสัมประสิทธิ์ α ซึ่งค่านี้จะช่วยลดความผิดพลาดที่เกิดขึ้น ทำให้เพิ่มประสิทธิภาพในการแบ่งกลุ่มข้อมูล

(2) ซัพพอร์ตเวกเตอร์แมชชีนแบบไม่เป็นเชิงเส้น ในกรณีที่ข้อมูลสองกลุ่มไม่สามารถใช้สมการเชิงเส้นในการแบ่งข้อมูลได้นั้น จะมีการแก้ไขปัญหาดังกล่าว โดยการเปลี่ยนแปลงมิติของข้อมูลให้สูงขึ้น เพื่อให้เกิดการเรียงตัวของข้อมูลใหม่หรือเรียกว่า ปริภูมิลักษณะมิติสูง (High Dimensional Feature Space) โดยจะทำการแปลงข้อมูลเวกเตอร์ไปยังปริภูมิลักษณะมิติสูง โดยใช้ฟังก์ชันเคอร์เนล (Kernel) ซึ่งฟังก์ชันเคอร์เนลที่ได้รับความนิยมมี 3 เทคนิค ได้แก่ ฟังก์ชันพหุนาม (Polynomial) ฟังก์ชันฐานเชิงรัศมี (Radial Basis Function) และ ฟังก์ชันซิกมอยด์ในเคอร์เนล คือการคูณกันของชุดเวกเตอร์ของ x ใด ๆ

$$k(x_i,x_j)=x_i^T,x_j \quad (2.35)$$

ฟังก์ชันพหุนาม (Polynomial) มีสมการดังนี้

$$k(x_i,x_j)=\left(\left\langle x_i,x_j\right\rangle+1\right)^d \quad (2.36)$$

โดยที่ d เป็นจำนวนเต็มใด ๆ

ฟังก์ชันฐานเชิงรัศมี (Radial Basis Function) มีสมการดังนี้

$$k(x_i, x_j) = \exp\left[-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right] \quad (2.37)$$

เมื่อ σ คือค่าพารามิเตอร์ ฟังก์ชันซิกมอยด์ มีสมการดังนี้

$$k(x_i, x_j) = \tanh\left(k\langle x_i, x_j\rangle + \mu\right) \quad (2.38)$$

เมื่อ k และ μ เป็นค่าพารามิเตอร์

จากสมการของเคอร์เนลสามารถที่จะแทนลงไปในตำแหน่งของ $x_i^T x_j$ ในสมการที่ (2-29) โดยเขียนเป็นสมการใหม่ได้ดังสมการที่ (2-39) สมการนี้ใช้ในขั้นตอนที่จะเรียนรู้ว่าจะวางตำแหน่งเส้นแบ่งไว้ที่ตำแหน่งใดโดยทำงานร่วมกับเคอร์เนล เพื่อแปลงให้ข้อมูลที่ยากต่อการแบ่งแบบเชิงเส้นสามารถแบ่งได้เมื่อถูกทำให้เป็นข้อมูลแบบมิติสูง (Higher Dimension) ดังนั้นจึงมีอีกสมการหนึ่งที่ใช้ค่า w และ b เดิมมาจัดตำแหน่งของข้อมูลเพื่อที่ให้ทราบว่าข้อมูลนั้นเป็นกลุ่มใดกำหนดได้ดังสมการที่ (2-40)

$$y_i\left(\sum_{i=1}^{N}\alpha_i y_i K_i(x_i, x_j) + b\right) - 1 \geq 0 \quad (2.39)$$

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{N}\alpha_i y_i K_i(x_i, x_j) + b\right) \quad (2.40)$$

3) การวิเคราะห์การถดถอยโลจิสติก (Logistic Regression Analysis)

การวิเคราะห์การถดถอยโลจิสติก [16, 31, 35, 40, 45] เป็นเทคนิคทางสถิติที่วิเคราะห์ตัวแปรเชิงพหุแบบหนึ่ง ซึ่งใช้ในการทำนายความน่าจะเป็น จากศึกษาความสัมพันธ์ระหว่างตัวแปรอิสระและตัวแปรตาม เพื่อนำสมการที่ได้ไปประมาณ หรือพยากรณ์ตัวแปรตาม เมื่อกำหนดค่าตัวแปรอิสระ ซึ่งเป็นแนวคิดเช่นเดียวกับการวิเคราะห์เชิงเส้น (Linear Regression) [31, 34] โดยการวิเคราะห์การถดถอยโลจิสติกสามารถจำแนกออกตามคุณลักษณะได้ 2 ชนิด ดังนี้

การถดถอยโลจิสติกทวินาม (Binary Logistic Regression) เป็นการวิเคราะห์ที่มีตัวแปรตามเป็นแปรเชิงกลุ่มที่มีค่าได้เพียง 2 ค่า (Dichotomous Variable) ตัวอย่างเช่น เพศชาย/เพศหญิง เป็นโรคมะเร็ง/ไม่เป็นโรคมะเร็ง เป็นรายงานจุดบกพร่อง/ไม่เป็นรานงานจุดบกพร่อง เป็นต้น ส่วนตัวแปรอิสระอาจมีเพียงหนึ่งตัว หรือหลายตัวก็ได้

การถดถอยโลจิสติกพหุกลุ่ม (Multinomial Logistic Regression) เป็นการวิเคราะห์ที่มีตัวแปรตามเป็นตัวแปรเชิงกลุ่มที่มีค่ามากกว่า 2 ค่า ตัวอย่างเช่น การวิเคราะห์โรคมะเร็ง (ไม่เป็นมะเร็ง/เป็นมะเร็งขั้นต้น/..../เป็นมะเร็งขั้นสุดท้าย) ประเภทของภาพยนตร์ (ภาพยนตร์ประเภทชีวิต/

ภาพยนตร์ประเภทตลก/ภาพยนตร์ประเภทโรแมนติก) เป็นต้น ส่วนตัวแปรอิสระอาจมีเพียงหนึ่งตัวหรือหลายตัวก็ได้ ซึ่งสามารถคำนวณได้จากสมการ (2.41)

$$Y = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n}}{1 + e} \quad (2.41)$$

หรือ

$$Y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n)}} \quad (2.42)$$

โดยที่ Y คือ ความน่าจะเป็นตัวแปรตาม

x_i คือ ค่าของตัวแปรอิสระตัวที่ i

β_i คือ ค่าสัมประสิทธิ์การถดถอยของตัวแปรอิสระตัวที่ *i* ซึ่งทั่วไปคำนวณด้วยเทคนิคความน่าจะเป็นสูงสุด (Maximum Likelihood)

e คือ ค่าคงที่ ซึ่งมามาจาก natural log ซึ่งมีค่าประมาณ 2.71828

4) แรนดอมฟอร์เรส (Random Forest)

แรนดอมฟอร์เรส [35] คำนี้มาจาก Random Decision Forests ซึ่งถูกเสนอครั้งแรก โดย Tin Kam Ho จาก Bell Labs ในปี 1995 ต่อมาภายหลังวิธีการนี้ได้ถูกขยายและจัดทำให้เป็นรูปแบบทั่วไปมากขึ้นโดย Leo Breiman ซึ่งวิธีการนี้จะรวมเอาความคิดของวิธีการ bagging ของ Leo Breiman และ การเลือกแบบสุ่มของคุณลักษณะ ของ Tim Kam Ho และ Yali Amit และ Donald Geman เพื่อสร้างกลุ่มของต้นไม้ตัดสินใจที่มีการควบคุมการแปรปรวน การเลือกเชตย่อยแบบสุ่มของคุณลักษณะเป็น ตัวอย่างของวิธีการย่อยแบบสุ่ม

แรนดอมฟอร์เรสจำแนกเอนเซมเบิล ที่ประกอบไปด้วยต้นไม้ตัดสินใจหลายต้น และให้ผลลัพธ์เป็นคลาส ที่เป็นผลลัพธ์ของคลาสจากต้นไม้แต่ละต้น อัลกอลิทึมสำหรับแรนดอมฟอร์เรสถูกพัฒนาโดย Leo Breiman และ Adele Cutler และ “Random Forests” เป็นเครื่องหมายการค้าของพวกเขา

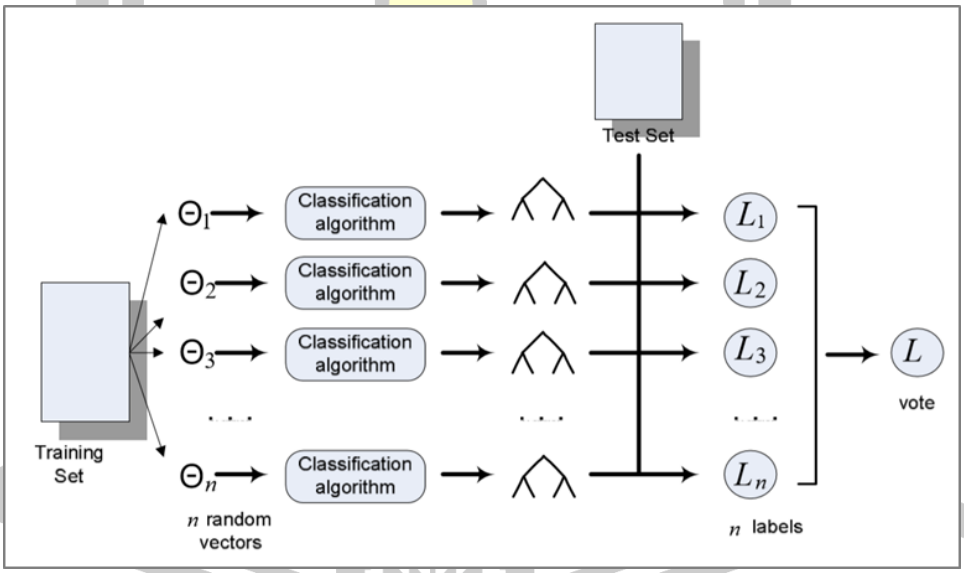
แรนดอมฟอร์เรส เป็นส่วนประกอบของตัวทำนายแบบต้นไม้ซึ่งแต่ละต้นนั้นขึ้นอยู่กับค่าของเวคเตอร์ที่สุ่มขึ้นมาอย่างอิสระต่อกันด้วยการกระจายแบบเดียวกันของต้นไม้ทั้งหมดที่อยู่ในป่าความผิดพลาดโดยทั่วไปของป่าจะมีค่าเข้าสู่ลิมิตเมื่อจำนวนของต้นไม้ทั้งหมดอยู่ในป่ามีจำนวนเยอะขึ้น ความผิดพลาดโดยทั่วไปของป่าในการจำแนกแบบต้นไม้ขึ้นอยู่กับความแข็งแรง(Strength)ของต้นไม้แต่ละต้นในป่า และความสัมพันธ์ (Correlation) ระหว่างกัน

การใช้เลือกสุ่มเพื่อที่จะแบ่งโหนดที่มีอัตราความผิดพลาดที่น่าพอใจกว่าเทคนิค Adaboost แต่ก็มีความเป็นไปได้ที่จะพบข้อมูลรบกวนได้มากกว่าการประมาณการภายในควบคุมความผิดพลาดที่เกิดขึ้น ความแข็งแรงและความสัมพันธ์ สิ่งเหล่านี้ใช้ในการแสดงการตอบสนองในการเพิ่มขึ้นของ

จำนวนคุณลักษณะที่ใช้ในการประมาณการภายในในการแบ่งยังถูกนำมาใช้ในการวัดความสำคัญของตัวแปรอีกด้วย ความคิดเหล่านี้ต่างก็สามารถนำมาใช้ได้กับการถดถอย (regression) ด้วยเหมือนกัน

Leo Breiman แสดงให้เห็นว่า ไม่เพียงแต่ว่าแรนดอมฟอร์เรสที่จะเป็นตัวจำแนกที่มีประสิทธิภาพสูงเท่านั้นแต่ยังได้กล่าวถึงปัญหาจำนวนมากที่มีความซับซ้อนและกระทบกับประสิทธิภาพของวิธีการจำแนกแบบอื่นที่ขัดแย้งกับขอบเขตแอพพลิเคชันแบบต่างๆ โดยเฉพาะไม่ต้องการการทำให้สมมติฐานง่ายขึ้น ในเรื่องแบบจำลองการกระจายตัวของข้อมูล และในเรื่องขบวนการผิดพลาด ยิ่งไปกว่านั้นมัน ก็ง่ายต่อการรองรับประเภทของข้อมูลต่าง ๆ ได้อย่างง่ายดาย และมีความคงทนสูงในการฝึกฝนด้วยขนาดของป่า ในขณะที่จำนวนของต้อนไม้ในแรนดอมฟอร์เรสเพิ่มค่าความผิดพลาดทั่วไป ที่ถูกแสดงให้เห็นในงานวิจัย ได้เบนเข้าหากัน และถูกกำหนดขอบเขต

แรนดอมฟอร์เรสเป็นตัวจำแนกที่ประกอบไปด้วยชุดของตัวจำแนกที่มีโครงสร้างแบบต้นไม้ โดยเป็นเวกเตอร์แบบสุ่มที่มีการกระจายที่เป็นอิสระต่อกันโดยสิ้นเชิง และต้นไม้ทุกต้นจะมีหนึ่งโหวตสำหรับคลาสที่นิยมมากที่สุด ที่นำเข้าเวกเตอร์จำนวน x ครั้ง โดยที่ h คือ x คือเวกเตอร์นำเข้า n คือจำนวนของตัวอย่างในชุดข้อมูลสอน



ภาพที่ 2-13 โครงสร้างโดยทั่วไปของแรนดอมฟอร์เรส

กฎที่ว่าด้วยจำนวนมากหรือการเพิ่มขึ้นของต้นไม้ในป่าแสดงให้เห็นว่าการเกิดความเฉพาะเจาะจงไม่ได้เป็นปัญหาแต่อย่างใด

มีงานวิจัยบางงานได้บอกว่าแรนดอมฟอร์เรสนั้น เกิดความผิดพลาดแบบทั่วไปน้อยกว่าวิธีการอื่น อย่างเช่น วิธีการเลือกแบ่งแบบสุ่มทำได้ดีกว่าวิธีการแบกกิ้งการทำให้ความถูกต้องดีขึ้นจะต้องมีการลดค่าความสัมพันธ์ ในขณะเดียวกันก็ต้องรักษาความแข็งแรงเอาไว้ด้วย โดยที่นำมาศึกษาประกอบไปด้วยการใช้วิธีการเลือกอินพุตแบบสุ่ม หรือใช้ส่วนประกอบจากอินพุตที่ทุกโหนดใน

การสร้างต้นไม้ผลลัพธ์ของป่า ที่ให้ความถูกต้องแม่นยำที่เปรียบเทียบได้กับเอดาบูท (Adaboost) กระบวนการนี้มีคุณลักษณะที่น่าพอใจดังนี้

1. ความถูกต้องดีเทียบเท่ากับเอดาบูท หรือดีกว่าในบางครั้ง
2. ค่อนข้างมีความคงทนต่อข้อมูลที่มีค่าผิดปกติ และ ข้อมูลรบกวน
3. เร็วกว่าแบงกิง หรือ บูสต์ติง
4. ให้ประโยชน์ในการประมาณภายในของความผิดพลาด, ความแข็งแรง, ความสัมพันธ์ และ ความสำคัญของตัวแปร
5. ง่ายและทำให้ขนานได้อย่างง่าย

อัลกอริทึมในการเรียนรู้แรนดอมฟอร์เรส ต้นไม้แต่ละต้นจะสร้างขึ้นโดยใช้ขั้นตอนดังต่อไปนี้

1. ให้ N เป็นจำนวนของกรณีฝึกฝนและ M เป็นจำนวนของตัวแปรในการจำแนก
2. ให้ m เป็นจำนวนของตัวแปรอินพุตที่ใช้กำหนดการตัดสินใจที่โหนดของต้นไม้โดยที่ m ควรจะน้อยกว่า M มาก
3. เลือกชุดของการเรียนรู้สำหรับต้นไม้โดยการเลือก N ครั้งด้วยการแทนที่จาก N กรณีฝึกฝนทั้งหมด (ใช้ตัวอย่าง บูตสแตรป) ใช้ส่วนที่เหลือของกรณีเพื่อประเมินความผิดพลาดของต้นไม้โดยการทำนาย
4. สำหรับแต่ละโหนดของต้นไม้ ให้สุ่มเลือกตัวแปร m บนพื้นฐานการตัดสินใจที่โหนดนั้น คำนวณการแบ่งที่ดีที่สุดบนพื้นฐานตัวแปร m นั้นในชุดข้อมูลสอน
5. ต้นไม้แต่ละต้นจะต้องเติบโตอย่างเต็มที่ และต้องไม่มีการตัดกิ่ง

2.4 การประเมินประสิทธิภาพ (Evaluation)

การประเมินประสิทธิภาพของแบบจำลองในการจำแนกข้อมูลเป็นส่วนสำคัญในการทำเหมืองข้อมูล เนื่องจากการประเมินประสิทธิภาพของแบบจำลอง ซึ่งเป็นตัวชี้วัดความน่าเชื่อถือของแบบจำลองที่สร้างขึ้น โดยวิธีที่นิยมใช้ในการประเมินประสิทธิภาพแบบจำลองวิธีหนึ่งคือ คอนฟิวชันเมตริกซ์ (Confusion Matrix) [31, 35]

คอนฟิวชันเมตริกซ์ เป็นการประเมินประสิทธิภาพแบบจำลองจากการใช้ข้อมูลจริงที่ได้รับการจำแนกข้อมูลอย่างถูกต้องไว้แล้ว และข้อมูลการจำแนกที่ได้จากการทำนายด้วยเทคนิคที่ใช้ในระบบ โดยสามารถประเมินได้หลายค่า ขึ้นอยู่กับข้อมูลที่ต้องการประเมิน ซึ่งเมตริกซ์สามารถใช้วัดประสิทธิภาพ ได้ดังตารางที่ 2-5

ตารางที่ 2-5 ตารางคอนฟิวชันเมตริกซ์

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

จากตารางความคลาดเคลื่อน เพื่อความสะดวกในการคำนวณ ซึ่งจะแทนที่ด้วยจำนวนความถี่ที่อัลกอริทึมทำนาย และความถี่ที่ผลลัพธ์เกิดขึ้นจริง โดยที่

TP (true positive) คือ จำนวนของผลลัพธ์ที่ทำนายว่าจริง และผลลัพธ์นั้นเป็นจริง

TN (true Negative) คือ จำนวนของผลลัพธ์ที่ทำนายว่าไม่จริง แต่ผลลัพธ์นั้นไม่เป็นจริง

FP (false positive) คือ จำนวนของผลลัพธ์ที่ทำนายว่าจริงแต่ผลลัพธ์นั้นไม่เป็นจริง

FN (false Negative) คือ จำนวนของผลลัพธ์ที่ทำนายว่าไม่จริงแต่ผลลัพธ์นั้นเป็นจริง

จากตาราง Confusion Matrix สามารถประเมินประสิทธิภาพของแบบจำลองตามค่ามาตรฐานต่าง ๆ ได้ดังนี้

ค่าความถูกต้อง (Accuracy) คือ ค่าที่แสดงว่าแบบจำลองสามารถทำนายได้ถูกต้องมากน้อยเท่าใด โดยเปรียบเทียบกับอัตราส่วนของข้อมูลทั้งหมด ดังสมการที่ (2.43)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.43}$$

ค่าความระลึก (Recall) คือ ค่าที่แสดงถึงการทำนายข้อมูลที่เป็นจริง เทียบกับอัตราส่วนของข้อมูลที่เป็นจริงทั้งหมด ดังสมการที่ (2.44)

$$Recall = \frac{TP}{TP + FN} \tag{2.44}$$

ค่าความแม่นยำ (Precision) คือ ค่าที่แสดงถึงข้อมูลที่ทำนายว่าจริง ถูกต้องเป็นอัตราส่วนเท่าใด เมื่อเทียบกับข้อมูลที่ทำนายว่าจริง ดังสมการที่ (2.45)

$$Precision = \frac{TP}{TP + FP} \tag{2.45}$$

ค่าเอฟ (F-measure) คือ ค่าที่แสดงถึงข้อมูลที่ทำนายว่าจริง ถูกต้องเป็นอัตราส่วนเท่าใด เมื่อเทียบกับข้อมูลที่ทำนายว่าจริง ดังสมการที่ (2.46)

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{2.46}$$

2.4 งานวิจัยที่เกี่ยวข้อง

Antoniol และคณะ [17] เป็นวิจัยแรกที่ได้ศึกษาเกี่ยวกับรายงานจุดบกพร่องที่ไม่สามารถจัดประเภทได้ ซึ่งนำไปสู่เป็นคัดแยกจุดบกพร่องของซอฟต์แวร์จากสิ่งที่ไม่ใช่จุดบกพร่องของซอฟต์แวร์ โดยผู้วิจัยได้การเก็บรวบรวมรายงานจุดบกพร่องจากโอเพ่นซอร์ส 3 โอเพนซอร์ส ได้แก่ Eclipse จำนวน 10,386 รายงาน Mozilla จำนวน 92,858 รายงาน และ JBoss จำนวน 3,207 รายงาน ซึ่งผู้วิจัยได้ใช้ข้อมูลในส่วนสรุป (Summary) คำอธิบายรายงานจุดบกพร่อง (Description) และข้อความคิดเห็นเกี่ยวกับรายงาน (Discussion) จากรายงานจุดบกพร่องที่สถานะของรายงานมีสถานะของรายงานเป็น “Verified” หรือ “Resolved” โดยได้ทำการคัดเลือกรายงานจากแต่ละโอเพ่นซอร์สจำนวน 600 รายงานด้วยตัวเอง โดยมีกระบวนการในการศึกษาประกอบด้วย การเตรียมข้อมูล เช่น การตัดส่วนขยาย(Stemming) การกำจัดเครื่องหมายวรรคตอน (Punctuation) และแปลงให้อยู่ในรูปแบบเวกเตอร์สเปซโมเดลโดยใช้ความถี่ของคำที่ปรากฏ (Term Frequency: TF) และการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง ซึ่งผู้วิจัยได้เปรียบเทียบเทคนิคที่ใช้ในการจำแนกรายงานจุดบกพร่อง 3 อัลกอริทึม ได้แก่ นาอีฟเบย์ ต้นไม้ตัดสินใจ และการถดถอยโลจิสติก โดยผลจากการทดลองพบว่าเทคนิคการจำแนกด้วยอัลกอริทึมถดถอยโลจิสติกส์ ให้ความแม่นยำในการจำแนกดีที่สุด รองลองมาคือ นาอีฟเบย์ และต้นไม้ตัดสินใจ ตามลำดับ ยกเว้นในการทดลองกับรายงานจุดบกพร่องของซอฟต์แวร์ JBoss :ที่อัลกอริทึมต้นไม้ตัดสินใจ และนาอีฟเบย์มีประสิทธิภาพที่ดีกว่าอัลกอริทึมการถดถอยโลจิสติกส์

Herzig และ คณะ [15] ได้ศึกษาเกี่ยวกับรายงานจุดบกพร่องที่ไม่สามารถจำแนกจุดบกพร่องได้ โดยผู้วิจัยได้ใช้เวลากว่า 90 วัน ในการจำแนกรายงานจุดบกพร่องจากระบบติดตามจุดบกพร่องของ JIRA จำนวน 7,401 รายงานด้วยตนเอง ซึ่งเป็นรายงานจุดกบพร่องจาก 5 โอเพ่นซอร์ส ได้แก่ HTTPClient, Jackrabbit, Lucene-Java, Rhino และ Tomcat5 โดยจากการศึกษาสามารถสรุปได้ดังนี้

1. รายงานจุดบกพร่องที่จัดประเภทรายงานจุดบกพร่องไม่ถูกต้องมีจำนวนมากกว่า 42.6%

2. รายงานจุดบกพร่องที่ไม่เป็นจุดบกพร่องของซอฟต์แวร์ หรือรายงานจุดบกพร่องที่ไม่ได้เกิดจากความผิดพลาดของซอสโค้ด (Source Code) มีจำนวน 33.8% ของรายงานที่ศึกษาทั้งหมด

3. มีรายงานจุดบกพร่องที่ไม่เคยเกิดจุดบกพร่องดังที่ระบุในรายงานจำนวน 39% ของรายงานจุดพร่องที่ศึกษา

Pingclasai และ คณะ [26] ได้ศึกษาต่อจากงานวิจัยของ Herzig [15] ซึ่งเป็นการจำแนกรายงานจุดบกพร่องด้วยตนเอง ดังนั้น ผู้วิจัยจึงได้นำเสนอการจำแนกรายงานจุดบกพร่องออกเป็น

รายงานที่เป็นจุดบกพร่องของซอฟต์แวร์ และรายงานที่ไม่ใช่จุดบกพร่องจุดบกพร่องของซอฟต์แวร์ โดยใช้เทคนิค Topic Modeling โดยมีกระบวนการในการเตรียมข้อมูลได้แก่ การกำจัด HTML Tags การกำจัดเครื่องหมายวรรคตอน การตัดส่วนขยาย และใช้เทคนิค Latent Dirichlet Allocation (LDA) ในการสร้างแบบจำลอง Topic ซึ่งเป็นการแปลงให้อยู่ในรูปของเวกเตอร์ แล้วจึงนำไปสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง โดยผู้วิจัยเปรียบเทียบประสิทธิภาพของการจำแนกรายงานจุดบกพร่องด้วยเทคนิค Topic-Based และเทคนิค Word-Based ด้วย 3 อัลกอริทึม ได้แก่ นาอีฟเบย์ ต้นไม้ตัดสินใจ และการถดถอยโลจิสติก ซึ่งให้ผลดังตารางที่ 2-6

ตารางที่ 2-6 ค่าเอฟของการจำแนกรายงานจุดบกพร่อง

	นาอีฟเบย์			ต้นไม้ตัดสินใจ			การถดถอยโลจิสติก		
	HTTPClient	Jackrabbit	Lucene	HTTPClient	Jackrabbit	Lucene	HTTPClient	Jackrabbit	Lucene
Word-base	0.702	0.690	0.761	0.666	0.619	0.475	0.753	0.711	0.774
Topic-base	0.720	0.715	0.805	0.742	0.738	0.782	0.725	0.762	0.803

จากการตารางพบได้ว่าเทคนิค Topic-Based มีประสิทธิภาพในการจำแนกรายงานจุดบกพร่องที่ดีกว่าเทคนิค Word-Based

Zhou และ คณะ [27] ได้นำเสนอกระบวนการในการจำแนกรายงานจุดบกพร่องระหว่างรายงานจุดบกร่อง และรายงานที่ไม่ใช่รายงานจุดบกพร่อง โดยใช้เทคนิคเหมืองข้อความร่วมกับเทคนิคเหมืองข้อมูล ซึ่งมีกระบวนการ คือ ใช้เทคนิคเหมือนข้อความในการทำนายความเป็นรายงานจุดบกพร่องจากข้อความที่มีในส่วนสรุปของรายงานจุดบกพร่องซึ่งถือเป็นข้อมูลที่ไม่มีโครงสร้าง (Unstructured Data) ด้วยอัลกอริทึมนาอีฟเบย์ทำนายความเป็นจุดบกพร่องออกเป็น 3 ระดับ ได้แก่ ระดับสูง (Hight) ระดับกลาง (Middle) และระดับต่ำ (Low) โดยผลที่ได้อยู่ในรูปแบบข้อมูลที่มีโครงสร้าง (Structured Data) ซึ่งสามารถนำไปประมวลผลด้วยเทคนิคเหมืองข้อมูลร่วมกับข้อมูลอื่นในรายงาน เช่น ความรุนแรง (Severity) ความสำคัญ (Priority) ส่วนประกอบ (Component) เป็นต้น โดยอัลกอริทึมเครือข่ายงานแบบเบย์. (Bayesian Network) โดยในการศึกษาได้ใช้ข้อมูลรายงานจุดบกพร่องจำนวน 3,220 รายงาน จาก 5 โอเพ่นซอร์ส ได้แก่ Mozilla (Core), Eclipse, JBoss, Mozilla (Firefox) และ OpenFOAM ซึ่งมีกระบวนการเตรียมข้อมูลประกอบด้วย การกำจัดคำหยุด การกำจัดเครื่องหมายวรรคตอน และการตัดส่วนขยาย โดยจากการศึกษาพบกว่าการจำแนกรายงานจุดบกพร่องด้วยเทคนิคเหมืองข้อความร่วมกับเทคนิคเหมืองข้อมูลมีค่าความแม่นยำอยู่ระหว่าง 0.80 ถึง 0.94 ค่าความระลึกอยู่ระหว่าง 0.81 ถึง 0.94 และมีค่าเอฟอยู่ระหว่าง 0.80 ถึง 0.94 ซึ่งมีประสิทธิภาพที่ดีกว่าการใช้เทคนิคเหมืองข้อความเพียงอย่างเดียว หรือการใช้เทคนิคเหมืองข้อมูลเพียงอย่างเดียว

Yadav และ Pal [46] ได้นำเสนอการเปรียบเทียบอัลกอริทึมสำหรับการตรวจจับ และคัดแยกรายงานจุดบกพร่อง โดยอัลกอริทึมที่นำเปรียบเทียบได้แก่ J48, ID3 และ Naïve Bayes ในการเตรียมข้อมูลผู้วิจัยได้ใช้ข้อมมูลรายงานจุดบกพร่องทดลองจำนวน 61 รายงาน แบ่งเป็นข้อมูลที่เป็นจุดบกพร่องของซอฟต์แวร์ (Bug) 6 ฉบับ และข้อมูลที่ไม่ใช่จุดบกพร่องของโปรแกรม (non Bug) 55 ฉบับ และใช้โปรแกรม Microsoft Excel 2010 และ Microsoft Word 2010 ในการออกแบบฐานข้อมูล เพื่อให้ได้ข้อมูลที่อยู่ในรูปของไฟล์ CSV (Comma-separated values) แล้วนำไปทำการทดลองบนโปรแกรม Weka ซึ่งผลที่ได้ผลดังตารางที่ 2-7 ซึ่งสามารถสรุปได้ว่า อัลกอริทึม J48 มีสามารถทำงานได้ดีที่สุด โดยสามารถจำแนกได้โดยไม่มีข้อผิดพลาด และมีความแม่นยำ 100 เปอร์เซ็นต์ ในขณะที่อัลกอริทึม Naïve Bayes สามารถจำแนกได้อย่างถูกต้อง 100 เปอร์เซ็นต์ แต่ยังพบว่าข้อผิดพลาดอยู่บ้าง และอัลกอริทึม ID3 มีความแม่นยำในการจำแนกอยู่ที่ 95 เปอร์เซ็นต์ และในส่วนของเวลาในการประมวลผลพบว่าอัลกอริทึม Naïve Bayes ใช้เวลาน้อยกว่าอีกสองอัลกอริทึมที่กล่าวมาข้างต้น

ตารางที่ 2-7 ผลการทดลองการเปรียบประสิทธิภาพอัลกอริทึม J48, ID3 และ นาอีฟเบย์

	J48	ID3	Naïve Bayes
sCorrectly classified Instances	100%	95.08%	100%
Relative absolute error	0	0	0.27%
Root relative square error	0	0	0.84%
Time taken (second)	0.02	0.02	0

Pandey และ คณะ [28] ได้นำเสนอการจำแนกรายงานจุดบกพร่องจากระบบติดตามรายงานจุดบกพร่องแบบอัตโนมัติ โดยจำแนกออกเป็นรายงานจุดบกพร่อง และไม่ใช่รายงานจุดบกพร่อง โดยการศึกษาเปรียบเทียบอัลกอริทึมที่ในการจำแนกรายงานจุดบกพร่อง ได้แก่ อัลกอริทึมนาอีฟเบย์ อัลกอริทึมซัพพอร์ตเวกเตอร์แมชชีน อัลกอริทึมการถดถอยโลจิสติก และอัลกอริทึมการวิเคราะห์การจำแนกเชิงเส้น (Linear Discriminate Analysis) โดยใช้ข้อมูลในส่วนสรุปของรายงานจุดบกพร่องของโอเพ่นซอร์ส HTTPClient และ Lucene จำนวน 500 รายงาน จากงานวิจัย [15] โดยมีกระบวนการในการเตรียมข้อมูลซึ่งประกอบด้วยการกำจัดคำหยุด การตัดส่วนขยาย และการแปลงให้อยู่ในรูปของเวกเตอร์สเปซโมเดลด้วยเทคนิคความถี่ของคำ (Term Frequency: TF) เช่นเดียวกับงานวิจัย [15] ซึ่งจากการศึกษาพบว่าแบบจำลองที่สร้างด้วยอัลกอริทึมนาอีฟเบย์สามารถจำแนกรายงานจุดบกพร่องได้zโดยมีค่าเอฟอยู่ระหว่าง 0.61-0.76 และ

อัลกอริทึมซัพพอร์ตเวกเตอร์แมชชีนมีค่าเอฟอยู่ระหว่าง 0.71-0.72 ซึ่งมีประสิทธิภาพในการจำแนกรายงานจุดบกพร่องสูงกว่าแบบจำลองอื่นที่ศึกษา

ในปีเดียวกัน Pandey และคณะ [16] ได้ขยายงายวิจัยของตนต่อเดิม โดยการใช้รายงานจุดบกพร่องจำนวน 5,587 รายงาน ซึ่งเป็นรายงานจุดบกพร่องจากงานวิจัย [15] โดยเปรียบเทียบอัลกอริทึมในการจำแนกรายงานจุดบกพร่องจำนวน 5 อัลกอริทึม คือ อัลกอริทึมการวิเคราะห์การจำแนกเชิงเส้น (Linear Discriminate Analysis) อัลกอริทึมเพื่อนบ้านใกล้เคียง (K-nearest neighbors) อัลกอริทึมซัพพอร์ตเวกเตอร์แมชชีน อัลกอริทึมต้นไม้ตัดสินใจ และ แรนดอมฟอเรส (Random Forest) ซึ่งพบว่าแบบจำลองที่สร้างด้วยอัลกอริทึมแรนดอมฟอเรส และอัลกอริทึมซัพพอร์ตเวกเตอร์แมชชีนมีประสิทธิภาพในการจำแนกรายงานสูงกว่าอัลกอริทึมอื่น โดยมีค่าความถูกต้องในการจำแนกสูงถึง 0.83

Nizamani และ คณะ [29] ได้เสนอการทำนายประเภทของรายงานจุดบกพร่องว่ารายงานที่มีการส่งเข้ามาในระบบติดตามรายงานจุดบกพร่องเป็นรายงานของจุดบกพร่องที่ควรได้รับการปรับปรุง (Approved) หรือเป็นรายงานที่ควรกำจัดทิ้ง (Reject) ด้วยอัลกอริทึมนาอีฟเบย์ โดยผู้วิจัยได้ใช้รายงานจุดบกพร่องที่มีสถานะ “Resolution” จำนวน 41,635 รายงาน จากระบบติดตามจุดบกพร่อง 35 โอเพ่นซอร์ส ซึ่งมีกระบวนการเตรียมข้อมูลได้แก่ การตัดคำโดยใช้พจนานุกรม การเปลี่ยนคำให้อยู่ในรูปแบบดั้งเดิม การกำจัดเครื่องหมายวรรคตอน การกำจัดตัวเลข และแปลงให้อยู่ในรูปของเวกเตอร์ด้วยเทคนิค TF และประเมินประสิทธิของการจำแนกรายงานจุดบกพร่อง ด้วยการเปรียบเทียบกับ 4 อัลกอริทึม ได้แก่ นาอีฟเบย์ ซัพพอร์ตเวกเตอร์แมชชีน แรนดอมฟอเรส และการถดถอยโลจิสติก ซึ่งให้ผลโดยเฉลี่ยดังตารางที่ 2-8

ตารางที่ 2-8 ผลการทดลองการเปรียบประสิทธิภาพอัลกอริทึม 4 อัลกอริทึม

	ค่าความถูกต้อง	ค่าความแม่นยำ	ค่าความระลึก	ค่าเอฟ
นาอีฟเบย์	89.25%	84.99%	63.26%	72.53%
ซัพพอร์ตเวกเตอร์แมชชีน	76.73%	56.76%	21.97%	31.50%
การถดถอยโลจิสติก	71.36%	41.38%	27.74%	31.94%
แรนดอมฟอเรส	71.84%	46.47%	33.18%	38.71%

จากตารางที่ 2-8 จะเห็นได้ว่านาอีฟเบย์มีประสิทธิภาพในการจำแนกรายงานจุดบกพร่องได้ดีกว่าอัลกอริทึมอื่น แต่มีข้อเสียคือใช้เวลาในการจำแนกรายงานจุดบกพร่องมากที่สุด

Terdchanakul และ คณะ [43] ได้นำเสนอเทคนิคในการจำแนกรายงานจุดบกพร่อง และไม่ใช่รายงานจุดบกพร่อง โดยเทคนิค N-Gram IDF ซึ่งเป็นเทคนิคการสกัดคุณลักษณะ (Feature)

และให้น้ำหนักคุณลักษณะ ซึ่งผู้วิจัยได้ทำการศึกษาโดยทำการเปรียบเทียบประสิทธิภาพเทคนิค N-Gram IDF และเทคนิค Topic Base ด้วยข้อมูลรายงานจุดบกพร่องจุดบกพร่องจาก [15] โดยใช้อัลกอริทึมการถดถอยโลจิสติก และอัลกอริทึม Random Forest ซึ่งจากการศึกษาพบการจำแนกรายงานจุดบกพร่องด้วยเทคนิค N-Gram IDF มีประสิทธิภาพดีกว่าการจำแนกรายงานจุดบกพร่องด้วยเทคนิค Topic Base โดยที่ประสิทธิภาพการจำแนกด้วยอัลกอริทึมการถดถอยโลจิสติมีค่าเอฟอยู่ระหว่าง 0.65 ถึง 0.73 และอัลกอริทึม Random Forest มีค่าเอฟอยู่ระหว่าง 0.63 ถึง 0.69

จากข้างต้นพบว่า การศึกษาเกี่ยวกับการจำแนกรายงานจุดบกพร่องออกเป็น รายงานที่เป็นจุดบกพร่องจริง และรายงานที่ไม่ใช่รายงานจุดบกพร่องนั้น ยังคงมีการดำเนินการอย่างต่อเนื่อง มาจนถึงปัจจุบัน เนื่องจากในระบบ BTS นั้น แม้จะมีการกำหนดมาตรฐานในการรายงานเพื่อให้ได้รายงานจุดบกพร่องที่มีคุณภาพ แต่เทคนิคหรือกระบวนในการคัดแยกรายงานจุดบกพร่องแบบอัตโนมัติยังต้องการการปรับปรุงให้มีประสิทธิภาพดียิ่งขึ้น เพราะหากพิจารณาจากการงานวิจัยที่ใช้ข้อมูลที่ดาวน์โหลดจาก BTS โดยไม่ผ่านการคัดแยกหรือกรองเอกสารด้วยมืออีกครั้ง จากการศึกษาจะพบว่ามีค่าเอฟ (F-measure) จะอยู่ระหว่าง 0.63 ถึง 0.68 เท่านั้น

บทที่ 3
วิธีการดำเนินการวิจัย

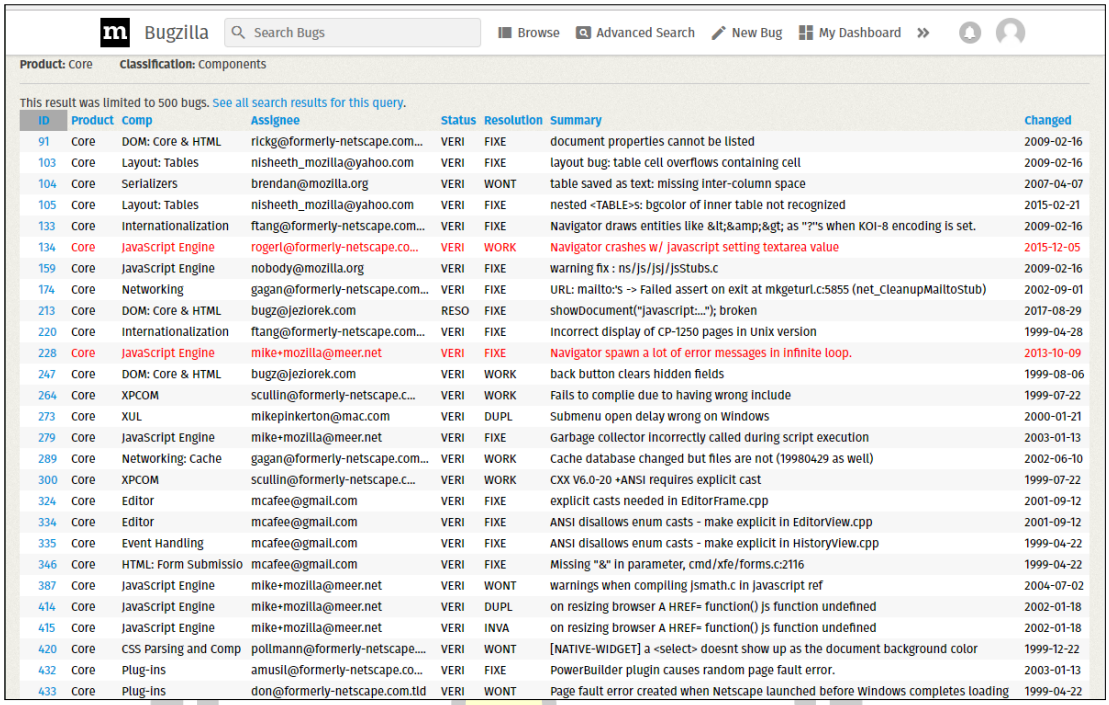
ในบทนี้จะอธิบายถึงชุดข้อมูลรายงานจุดบกพร่องที่ใช้ในการวิจัย และวิธีดำเนินการวิจัยในการจำแนกรายงานจุดบกพร่องออกเป็นรายงานจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่อง ดังนี้

3.1 ชุดข้อมูล (Datasets)

ในงานวิจัยนี้ได้ใช้ชุดข้อมูลรายงานจุดบกพร่องจาก 2 ระบบติดตามจุดบกพร่อง ประกอบด้วย ระบบติดตามจุดบกพร่อง Bugzilla และระบบติดตามจุดบกพร่อง Jira ซึ่งสามารถแบ่งชุดมูลตามการใช้ในงานวิจัยได้ดังนี้

3.1.1 ชุดข้อมูลสำหรับการเตรียมการเบื้องต้น (Preliminary Dataset)

ชุดข้อมูลสำหรับการเตรียมการเบื้องต้น เป็นชุดข้อมูลสำหรับใช้ในการสร้างคลังขั้วคำใช้ชุดข้อมูลรายงานจุดบกพร่อง Core ของ Mozilla ซึ่งเป็นโอเพนซอร์สที่เป็นส่วนประกอบพื้นฐานซอฟต์แวร์ที่ถูกใช้งานในซอฟต์แวร์ต่าง ๆ ของ Mozilla ตัวอย่างเช่น Firefox Thunderbird และอื่น ๆ ซึ่งรวมไปจนถึงการจัดการเนื้อหาเว็บ Gecko, HTML, CSS, รูปแบบ (Layout), สคริปต์ (Scripts), รูปภาพ (Images), เครือข่าย (Networking) และอื่น ๆ [14] โดยรายงานจุดบกพร่องของ Core ถูกแบ่งออกเป็น 138 กลุ่ม และมีรายงานจุดบกพร่องรวมมากกว่า 360,000 รายงาน ดังภาพที่ 3-1 ซึ่งสามารถเข้าถึงรายงานจุดบกพร่องผ่านเว็บไซต์ได้ที่ https://bugzilla.mozilla.org/describecomponents.cgi?product=Core โดยรายงานจุดบกพร่องที่ใช้เป็นรายงานที่เป็นข้อความภาษาอังกฤษ ซึ่งมีสถานะตามวงจรชีวิตของจุดบกพร่อง New, Assigned, Resolved, Verified, Close และ Reopen เท่านั้น



Bugzilla Search Bugs Browse Advanced Search New Bug My Dashboard

Product: Core **Classification:** Components

This result was limited to 500 bugs. See all search results for this query.

ID	Product	Comp	Assignee	Status	Resolution	Summary	Changed
91	Core	DOM: Core & HTML	rickg@formerly-netscape.com...	VERI	FIXE	document properties cannot be listed	2009-02-16
103	Core	Layout: Tables	nisheeth_mozilla@yahoo.com	VERI	FIXE	layout bug: table cell overflows containing cell	2009-02-16
104	Core	Serializers	brendan@mozilla.org	VERI	WONT	table saved as text: missing inter-column space	2007-04-07
105	Core	Layout: Tables	nisheeth_mozilla@yahoo.com	VERI	FIXE	nested <TABLE>s: bgcolor of inner table not recognized	2015-02-21
133	Core	Internationalization	ftang@formerly-netscape.com...	VERI	FIXE	Navigator draws entities like <> as "?"s when KOI-8 encoding is set.	2009-02-16
134	Core	JavaScript Engine	rogerl@formerly-netscape.co...	VERI	WORK	Navigator crashes w/ javascript setting textarea value	2015-12-05
159	Core	JavaScript Engine	nobody@mozilla.org	VERI	FIXE	warning fix : ns/js/jsj/jsStubs.c	2009-02-16
174	Core	Networking	gagan@formerly-netscape.com...	VERI	FIXE	URL: mailto:'s -> Failed assert on exit at mkgeturl.c:5855 (net_CleanupMailtoStub)	2002-09-01
213	Core	DOM: Core & HTML	bugz@jeziorek.com	RESO	FIXE	showDocument("javascript:..."); broken	2017-08-29
220	Core	Internationalization	ftang@formerly-netscape.com...	VERI	FIXE	Incorrect display of CP-1250 pages in Unix version	1999-04-28
228	Core	JavaScript Engine	mike+mozilla@meer.net	VERI	FIXE	Navigator spawn a lot of error messages in infinite loop.	2013-10-09
247	Core	DOM: Core & HTML	bugz@jeziorek.com	VERI	WORK	back button clears hidden fields	1999-08-06
264	Core	XPCOM	scullin@formerly-netscape.c...	VERI	WORK	Fails to compile due to having wrong include	1999-07-22
273	Core	XUL	mikepinkerton@mac.com	VERI	DUPL	Submenu open delay wrong on Windows	2000-01-21
279	Core	JavaScript Engine	mike+mozilla@meer.net	VERI	FIXE	Garbage collector incorrectly called during script execution	2003-01-13
289	Core	Networking: Cache	gagan@formerly-netscape.com...	VERI	WORK	Cache database changed but files are not (19980429 as well)	2002-06-10
300	Core	XPCOM	scullin@formerly-netscape.c...	VERI	WORK	CXX V6.0-20 +ANSI requires explicit cast	1999-07-22
324	Core	Editor	mcafee@gmail.com	VERI	FIXE	explicit casts needed in EditorFrame.cpp	2001-09-12
334	Core	Editor	mcafee@gmail.com	VERI	FIXE	ANSI disallows enum casts - make explicit in EditorView.cpp	2001-09-12
335	Core	Event Handling	mcafee@gmail.com	VERI	FIXE	ANSI disallows enum casts - make explicit in HistoryView.cpp	1999-04-22
346	Core	HTML: Form Submissio	mcafee@gmail.com	VERI	FIXE	Missing "&" in parameter, cmd/xfe/forms.c:2116	1999-04-22
387	Core	JavaScript Engine	mike+mozilla@meer.net	VERI	WONT	warnings when compiling jsmath.c in javascript ref	2004-07-02
414	Core	JavaScript Engine	mike+mozilla@meer.net	VERI	DUPL	on resizing browser A HREF= function() js function undefined	2002-01-18
415	Core	JavaScript Engine	mike+mozilla@meer.net	VERI	INVA	on resizing browser A HREF= function() js function undefined	2002-01-18
420	Core	CSS Parsing and Comp	pollmann@formerly-netscape....	VERI	WONT	[NATIVE-WIDGET] a <select> doesnt show up as the document background color	1999-12-22
432	Core	Plug-ins	amusil@formerly-netscape.co...	VERI	FIXE	PowerBuilder plugin causes random page fault error.	2003-01-13
433	Core	Plug-ins	don@formerly-netscape.com.tld	VERI	WONT	Page fault error created when Netscape launched before Windows completes loading	1999-04-22

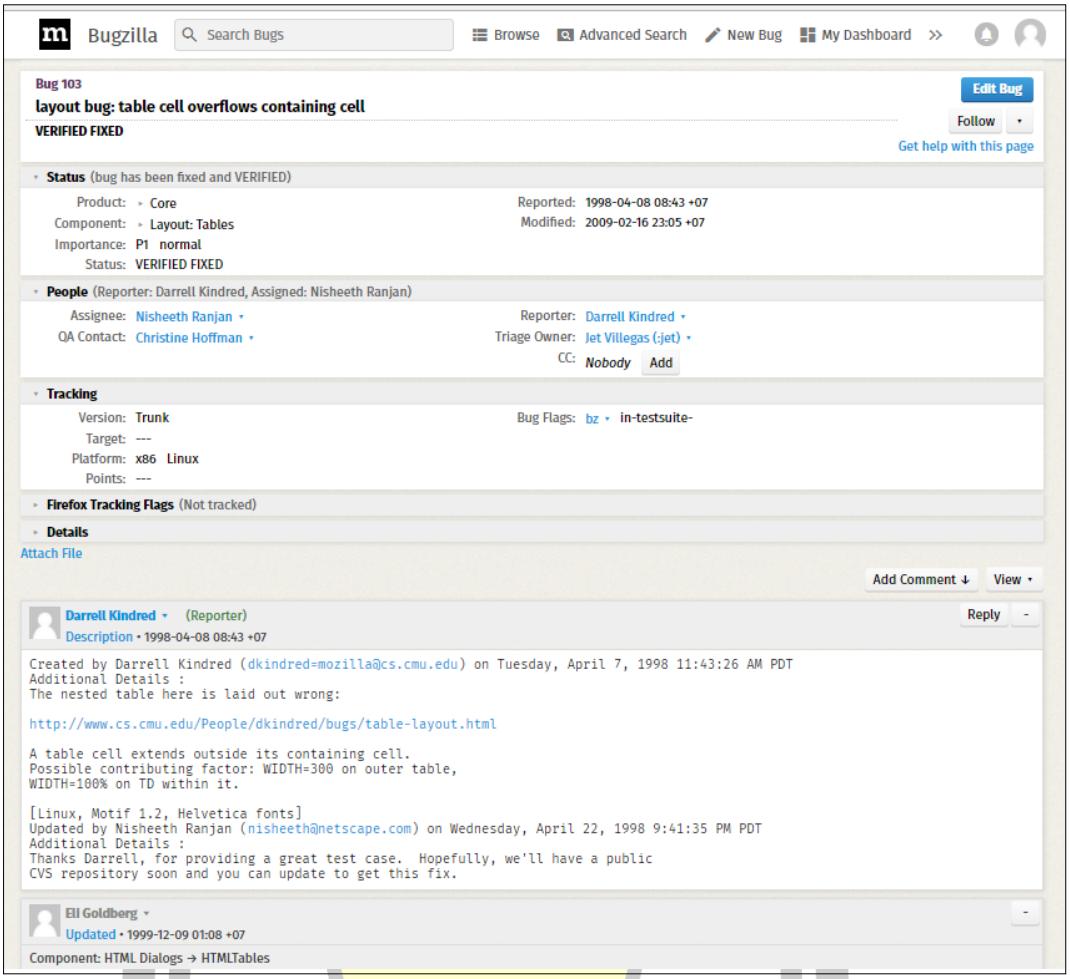
ภาพที่ 3-1 รายการรายงานจุดบกพร่องของ Core

ที่มา : (https://bugzilla.mozilla.org/buglist.cgi?product=Core)

โดยในแต่ละรายงานจุดบกพร่องจะปรากฏข้อมูล ดังภาพที่ 3-2 ซึ่งประกอบด้วยข้อมูลดังตารางที่ 3-1

ตารางที่ 3-1 ข้อมูลรายการของรายงานจุดบกพร่อง Core

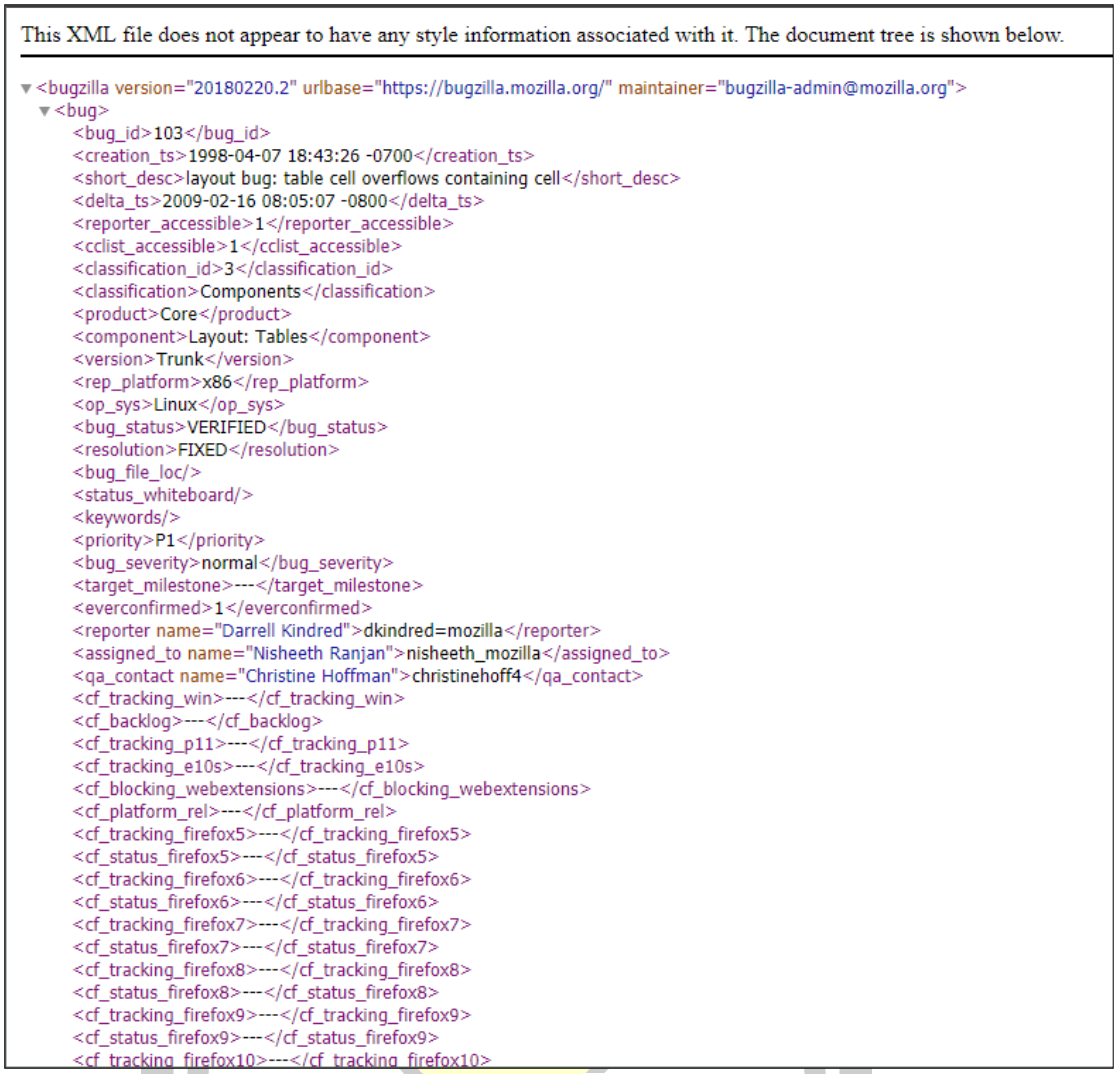
ชื่อคอลัมน์	แสดงข้อมูล	ตัวอย่างเช่น
ID	หมายเลขของรายงานจุดบกพร่อง	103
Product	ชื่อของโอเพนซอร์ซ	Core
Comp	ชื่อของส่วนประกอบของซอฟต์แวร์	Layout: Tables
Assignee	ชื่อของผู้ที่ทำการรายงานจุดบกพร่อง	nisheeth_mozilla@yahoo.com
Status	สถานะปัจจุบันของรายงานจุดบกพร่อง	VERI
Resolution	สถานะย่อยของรายงานจุดบกพร่อง	FIXE
Summary	ส่วนสรุปของรายงานจุดบกร่อง	layout bug: table cell overflows containing cell
Changed	วันที่ทำการแก้ไข ปรับปรุงรายงานล่าสุด	2009-02-16



ภาพที่ 3-2 รายงานจุดบกพร่องของ Core

ที่มา (https://bugzilla.mozilla.org/show_bug.cgi?id=103)

ในงานวิจัยนี้ได้ใช้เฉพาะรายงานจุดบกพร่องได้รับการคัดกรองจากผู้คัดกรองรายงาน (Bug Triager) ซึ่งมีสถานะ (Status) ได้แก่ New, Assigned, Resolved, Verified, Close และ Reopen เท่านั้น โดยในงานวิจัยนี้เก็บรวบรวมข้อมูลรายงานจุดบกพร่องของ Core จากเว็บของ Mozilla ในรูปแบบไฟล์เอกซ์เอ็มแอล (Extensible Markup Language: xml) เนื่องจากเป็นไฟล์ที่มีโครงสร้างที่ประกอบด้วยแท็ก (Tag) เปิด และปิด จึงง่ายต่อการทำความเข้าใจ และสะดวกต่อการเรียกใช้งานข้อมูล โดยสามารถรวบรวมข้อมูลรายงานจุดบกพร่องของ Core ได้จำนวน 278,996 รายงาน ซึ่งในแต่ละรายงานจะมีลักษณะดังภาพที่ 3-3



ภาพที่ 3-3 ข้อมูลรายงานจุดบกพร่องของ Core ในรูปแบบไฟล์ .XML

ซึ่งในการสร้างขั้วของคำในงานวิจัยนี้ใช้ข้อมูลในส่วนสรุป (Summary) ซึ่งสอดคล้องกับงานวิจัยที่ผ่านมาที่นิยมใช้ข้อมูลในส่วนนี้ในการดำเนินงานวิจัย [16, 17, 19, 33, 34]

รายงานจุดบกพร่องที่รวบรวมไว้จะถูกจัดเก็บในรูปของไฟล์ .txt ซึ่งมีขนาดเล็กกว่าไฟล์ต้นฉบับ

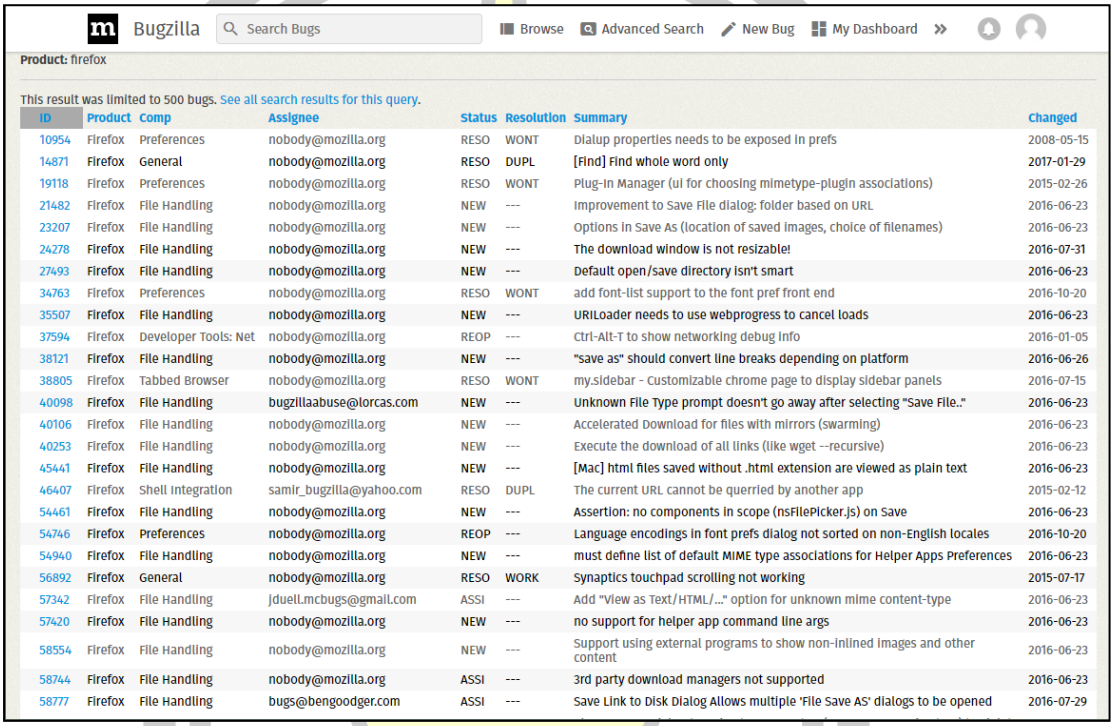
3.1.2 ชุดข้อมูลสำหรับการทดสอบการจำแนกรายงานจุกบกพร่อง

ในงานวิจัยนี้ได้ใช้ทำการทดสอบการจำแนกรายงานจุดบกพร่องกับชุดข้อมูลจำนวน2 ชุดข้อมูล ดังนี้

1) ชุดข้อมูลรายงานจุดบกพร่อง Firefox

ข้อมูลรายงานจุดบกพร่องของ Firefox เป็นข้อมูลรายงานจุดบกพร่องที่พบใน Firefox ของ Mozilla ซึ่ง Firefox เป็นเบราเซอร์ (Browser) หรือก็คือ โปรแกรมค้นดูเว็บ ซึ่งได้รับการพัฒนาโดย

Mozilla ปัจจุบัน Firefox เป็นเบราเซอร์ที่สามารถใช้งานได้ในหลายระบบปฏิบัติการ และยังเป็นเบราว์เซอร์ที่นิยมอันดับ 3 รองจาก Internet Explorer และ Google Chrome [1] โดยรายงานจุดบกพร่องของ Firefox แบ่งออกเป็น 80 กลุ่ม และมีรายงานจุดบกพร่องรวมมากกว่า 180,000 รายงาน ดังภาพที่ 3-4



Bugzilla | Search Bugs | Browse | Advanced Search | New Bug | My Dashboard

Product: firefox

This result was limited to 500 bugs. See all search results for this query.

ID	Product	Comp	Assignee	Status	Resolution	Summary	Changed
10954	Firefox	Preferences	nobody@mozilla.org	RESO	WONT	Dialup properties needs to be exposed in prefs	2008-05-15
14871	Firefox	General	nobody@mozilla.org	RESO	DUPL	[Find] Find whole word only	2017-01-29
19118	Firefox	Preferences	nobody@mozilla.org	RESO	WONT	Plug-in Manager (ui for choosing mimetype-plugin associations)	2015-02-26
21482	Firefox	File Handling	nobody@mozilla.org	NEW	---	Improvement to Save File dialog: folder based on URL	2016-06-23
23207	Firefox	File Handling	nobody@mozilla.org	NEW	---	Options in Save As (location of saved images, choice of filenames)	2016-06-23
24278	Firefox	File Handling	nobody@mozilla.org	NEW	---	The download window is not resizable!	2016-07-31
27493	Firefox	File Handling	nobody@mozilla.org	NEW	---	Default open/save directory isn't smart	2016-06-23
34763	Firefox	Preferences	nobody@mozilla.org	RESO	WONT	add font-list support to the font pref front end	2016-10-20
35507	Firefox	File Handling	nobody@mozilla.org	NEW	---	URILoader needs to use webprogress to cancel loads	2016-06-23
37594	Firefox	Developer Tools: Net	nobody@mozilla.org	REOP	---	Ctrl-Alt-T to show networking debug info	2016-01-05
38121	Firefox	File Handling	nobody@mozilla.org	NEW	---	"save as" should convert line breaks depending on platform	2016-06-26
38805	Firefox	Tabbed Browser	nobody@mozilla.org	RESO	WONT	my.sidebar - Customizable chrome page to display sidebar panels	2016-07-15
40098	Firefox	File Handling	bugzillaabuse@lorcas.com	NEW	---	Unknown File Type prompt doesn't go away after selecting "Save File.."	2016-06-23
40106	Firefox	File Handling	nobody@mozilla.org	NEW	---	Accelerated Download for files with mirrors (swarming)	2016-06-23
40253	Firefox	File Handling	nobody@mozilla.org	NEW	---	Execute the download of all links (like wget --recursive)	2016-06-23
45441	Firefox	File Handling	nobody@mozilla.org	NEW	---	[Mac] html files saved without .html extension are viewed as plain text	2016-06-23
46407	Firefox	Shell Integration	samir_bugzilla@yahoo.com	RESO	DUPL	The current URL cannot be querried by another app	2015-02-12
54461	Firefox	File Handling	nobody@mozilla.org	NEW	---	Assertion: no components in scope (nsFilePicker.js) on Save	2016-06-23
54746	Firefox	Preferences	nobody@mozilla.org	REOP	---	Language encodings in font prefs dialog not sorted on non-English locales	2016-10-20
54940	Firefox	File Handling	nobody@mozilla.org	NEW	---	must define list of default MIME type associations for Helper Apps Preferences	2016-06-23
56892	Firefox	General	nobody@mozilla.org	RESO	WORK	Synaptics touchpad scrolling not working	2015-07-17
57342	Firefox	File Handling	jduell.mcbugs@gmail.com	ASSI	---	Add "View as Text/HTML/..." option for unknown mime content-type	2016-06-23
57420	Firefox	File Handling	nobody@mozilla.org	NEW	---	no support for helper app command line args	2016-06-23
58554	Firefox	File Handling	nobody@mozilla.org	NEW	---	Support using external programs to show non-inlined images and other content	2016-06-23
58744	Firefox	File Handling	nobody@mozilla.org	ASSI	---	3rd party download managers not supported	2016-06-23
58777	Firefox	File Handling	bugs@bengoodger.com	ASSI	---	Save Link to Disk Dialog Allows multiple 'File Save AS' dialogs to be opened	2016-07-29

ภาพที่ 3-4 รายการรายงานจุดบกพร่องของ Firefox

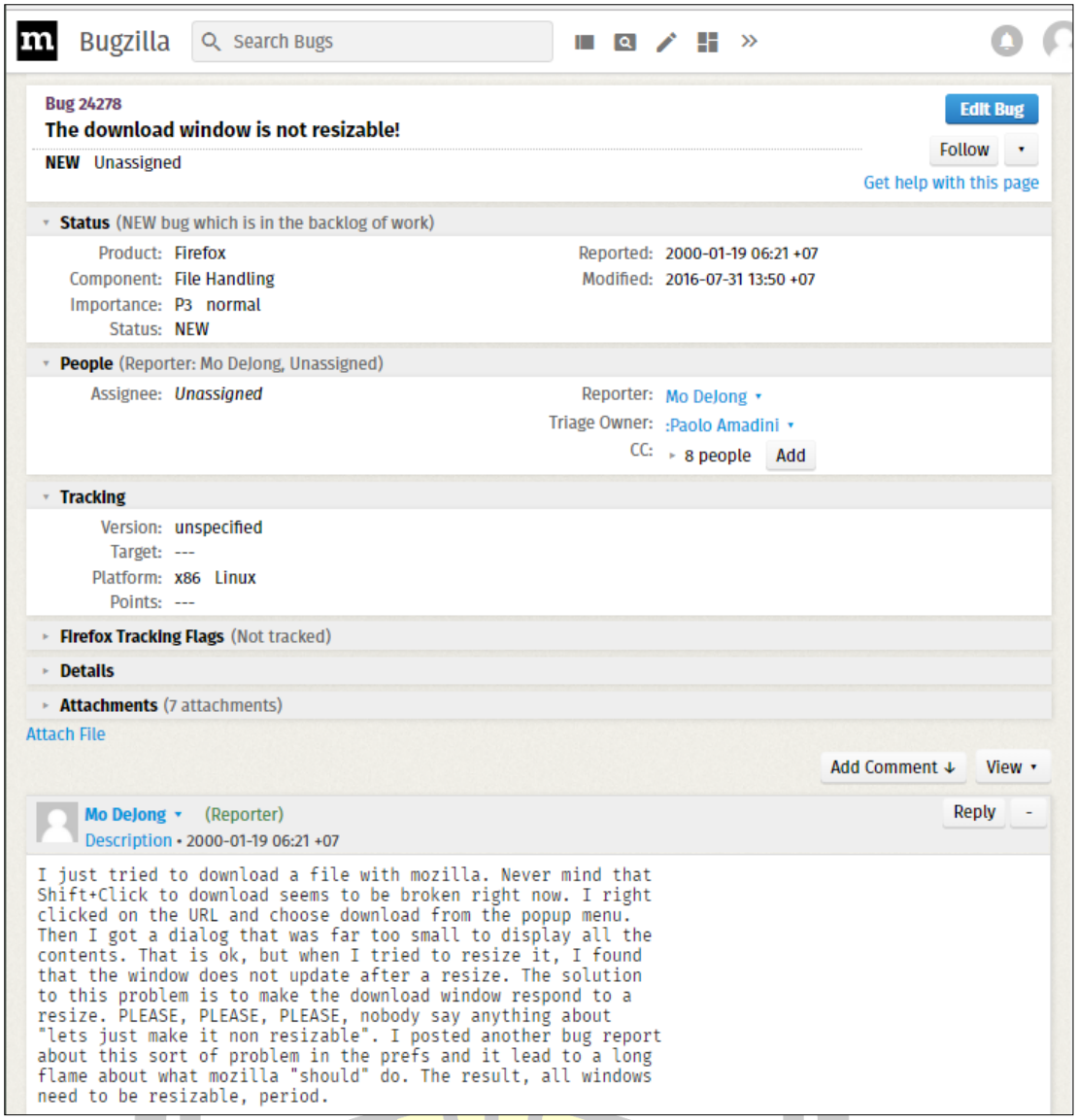
ที่มา : (https://bugzilla.mozilla.org/buglist.cgi?product=Firefox)

โดยในแต่ละรายงานจุดบกพร่องจะปรากฏข้อมูล ดังภาพที่ 3-4 ซึ่งประกอบด้วยข้อมูลดังตารางที่ 3-2

ตารางที่ 3-2 ข้อมูลรายการของรายงานจุดบกพร่อง Firefox

ชื่อคอลัมน์	แสดงข้อมูล	ตัวอย่างเช่น
ID	หมายเลขของรายงานจุดบกพร่อง	24278
Product	ชื่อของโอเพนซอร์ซ	Firefox
Comp	ชื่อของส่วนประกอบของซอฟต์แวร์	File Handling
Assignee	ชื่อของผู้ที่ทำการรายงานจุดบกพร่อง	nobody@mozilla.org
Status	สถานะปัจจุบันของรายงานจุดบกพร่อง	NEW
Resolution	สถานะย่อยของรายงานจุดบกพร่อง	---
Summary	ส่วนสรุปของจุดบกร่อง	The download window is not resizable!
Changed	เวลาที่ทำการแก้ไข ปรับปรุงรายงานล่าสุด	2016-07-31

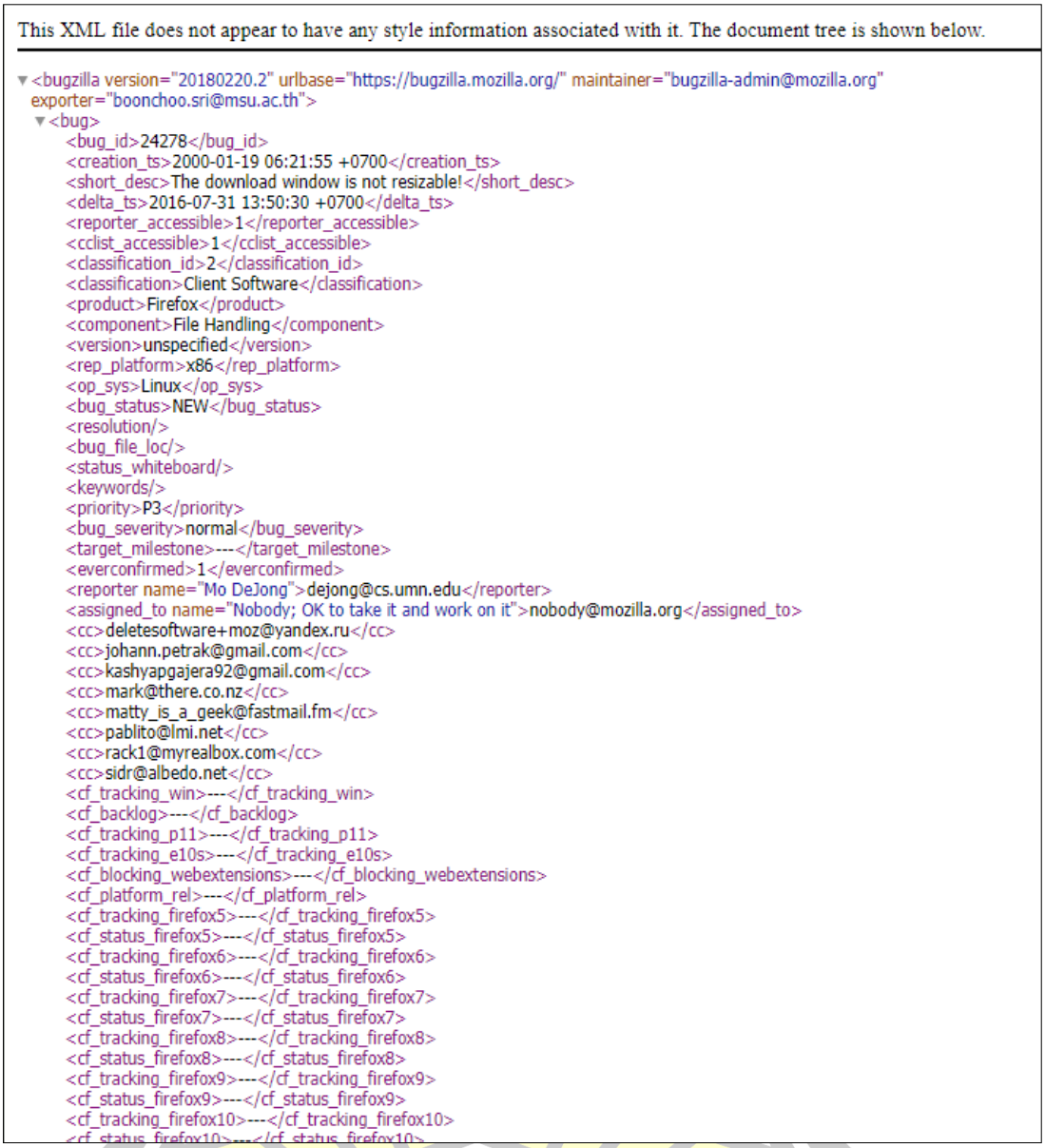
ในงานวิจัยนี้ได้ใช้เฉพาะรายงานจุดบกพร่องได้รับการคัดกรองจากผู้คัดกรองรายงาน (Bug Triager) ซึ่งมีสถานะ (Status) ได้แก่ New, Assigned, Resolved, Verified, Close และ Reopen เท่านั้น โดยในแต่ละรายงานประกอบด้วยข้อมูลที่อธิบายเกี่ยวกับลักษณะของจุดบกพร่องที่พบใน Firefox ดังภาพที่ 3-5



ภาพที่ 3-5 รายงานจุดบกพร่องของ Firefox

ที่มา (https://bugzilla.mozilla.org/show_bug.cgi?id= 24278)

ซึ่งในงานวิจัยนี้เก็บรวบรวมข้อมูลรายงานจุดบกพร่องของ Firefox จากเว็บของ Mozilla ในรูปแบบไฟล์ XML เช่นเดียวกันการเก็บข้อมูลรายงานจุดบกพร่อง Core โดยสามารถรวบรวมข้อมูลรายงานจุดบกพร่องของ Core ได้จำนวน 149,920 รายงาน ซึ่งในแต่ละรายงานจะมีลักษณะดังภาพที่ 3-6



This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<bugzilla version="20180220.2" urlbase="https://bugzilla.mozilla.org/" maintainer="bugzilla-admin@mozilla.org"
  exporter="boonchoo.sri@msu.ac.th">
  <bug>
    <bug_id>24278</bug_id>
    <creation_ts>2000-01-19 06:21:55 +0700</creation_ts>
    <short_desc>The download window is not resizable!</short_desc>
    <delta_ts>2016-07-31 13:50:30 +0700</delta_ts>
    <reporter_accessible>1</reporter_accessible>
    <cclist_accessible>1</cclist_accessible>
    <classification_id>2</classification_id>
    <classification>Client Software</classification>
    <product>Firefox</product>
    <component>File Handling</component>
    <version>unspecified</version>
    <rep_platform>x86</rep_platform>
    <op_sys>Linux</op_sys>
    <bug_status>NEW</bug_status>
    <resolution/>
    <bug_file_loc/>
    <status_whiteboard/>
    <keywords/>
    <priority>P3</priority>
    <bug_severity>normal</bug_severity>
    <target_milestone>---</target_milestone>
    <everconfirmed>1</everconfirmed>
    <reporter name="Mo DeJong">dejong@cs.umn.edu</reporter>
    <assigned_to name="Nobody; OK to take it and work on it">nobody@mozilla.org</assigned_to>
    <cc>deletesoftware+moz@yandex.ru</cc>
    <cc>johann.petrak@gmail.com</cc>
    <cc>kashyapgajera92@gmail.com</cc>
    <cc>mark@there.co.nz</cc>
    <cc>matty_is_a_geek@fastmail.fm</cc>
    <cc>pablito@lmi.net</cc>
    <cc>rack1@myrealbox.com</cc>
    <cc>sidr@albedo.net</cc>
    <cf_tracking_win>---</cf_tracking_win>
    <cf_backlog>---</cf_backlog>
    <cf_tracking_p11>---</cf_tracking_p11>
    <cf_tracking_e10s>---</cf_tracking_e10s>
    <cf_blocking_webextensions>---</cf_blocking_webextensions>
    <cf_platform_rel>---</cf_platform_rel>
    <cf_tracking_firefox5>---</cf_tracking_firefox5>
    <cf_status_firefox5>---</cf_status_firefox5>
    <cf_tracking_firefox6>---</cf_tracking_firefox6>
    <cf_status_firefox6>---</cf_status_firefox6>
    <cf_tracking_firefox7>---</cf_tracking_firefox7>
    <cf_status_firefox7>---</cf_status_firefox7>
    <cf_tracking_firefox8>---</cf_tracking_firefox8>
    <cf_status_firefox8>---</cf_status_firefox8>
    <cf_tracking_firefox9>---</cf_tracking_firefox9>
    <cf_status_firefox9>---</cf_status_firefox9>
    <cf_tracking_firefox10>---</cf_tracking_firefox10>
    <cf_status_firefox10>---</cf_status_firefox10>
```


ภาพที่ 3-6 ข้อมูลรายงานจุดบกพร่องของ Firefox ในรูปแบบไฟล์ XML

ซึ่งในการสร้างขั้วของคำในงานวิจัยนี้ใช้ข้อมูลในส่วนสรุป (Summary) ซึ่งสอดคล้องกับงานวิจัยที่ผ่านมาที่นิยมใช้ข้อมูลในส่วนนี้ในการดำเนินงานวิจัย [16, 17, 19, 33, 34]

รายงานจุดบกพร่องที่รวบรวมไว้จะถูกจัดเก็บในรูปของไฟล์ txt ซึ่งมีขนาดเล็กกว่าไฟล์ต้นฉบับ

2) ชุดข้อมูลรายงานจุดบกพร่องของ Herzig

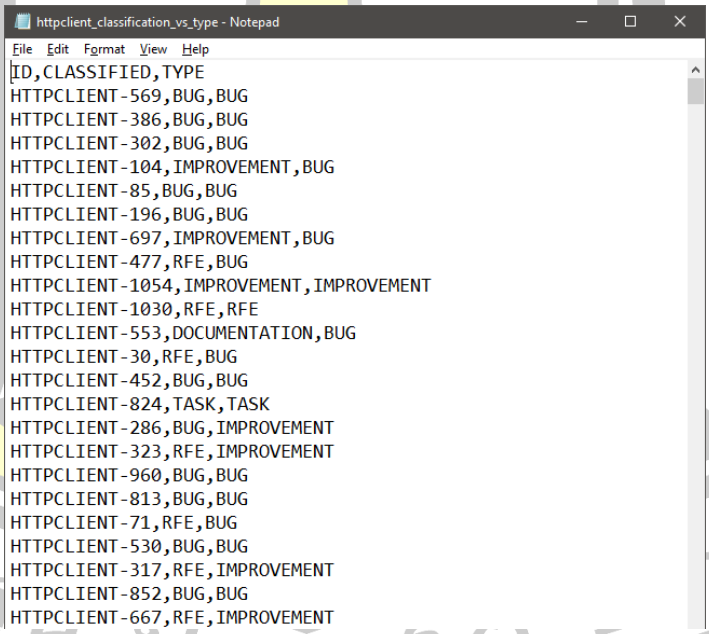
ชุดข้อมูลรายงานจุดพร่องของ Herzig [15] เป็นชุดข้อมูลรายงานจุดบกพร่องที่ Herzig และคณะ ได้ทำการเก็บรวบรวมรายงานจุดบกพร่องที่มีสถานะ RESOLVED, CLOSED, และ VERIFIED จาก 5 โอเพนซอร์ส ได้แก่ HttpClient, Lucene, Jackrabbit, Rhino และ Tomcat5

โดย Herzig และคณะ [15] คัดกรองรายงานจุดบกพร่อง และคัดแยกรายงานจุดบกพร่องด้วยตนเอง จึงได้เป็นรายงานจุดบกพร่องเป็นจำนวน 7,401 รายงาน ดังตารางที่ 3-3

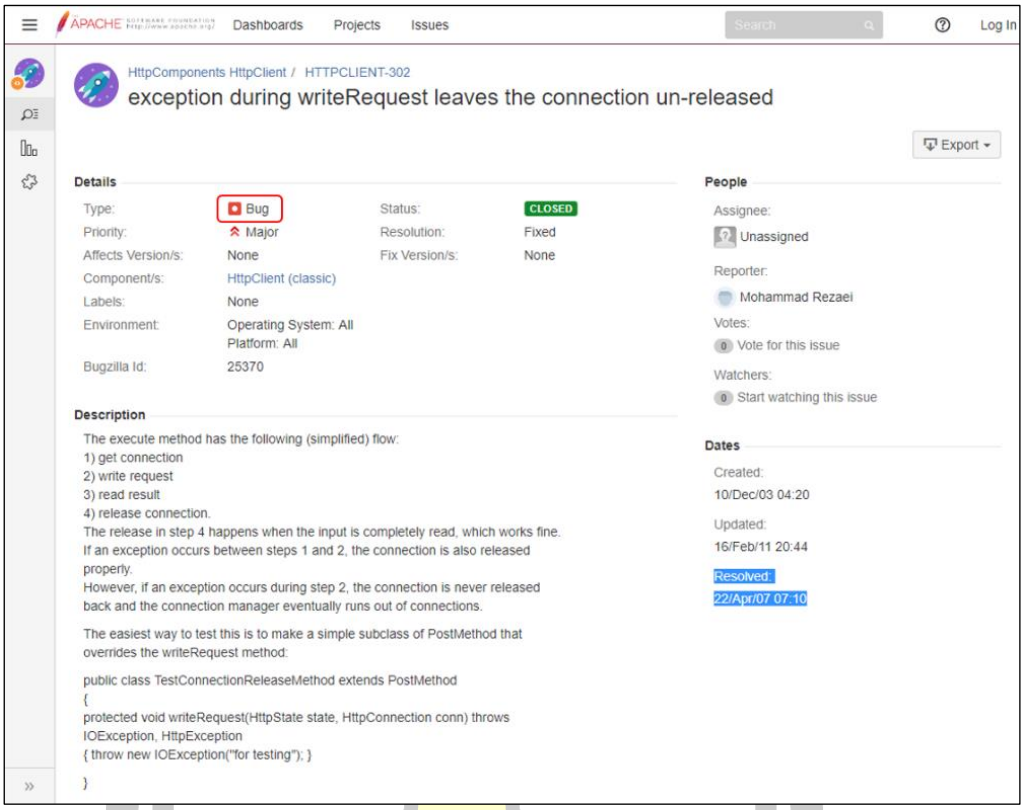
ตารางที่ 3-3 ชุดข้อมูล Herzig

ระบบติดตามจุดบกพร่อง	โอเพนซอร์ส	จำนวนรายงานจุดบกพร่อง
Jira	HttpClient	745
	Lucene	2,441
	Jackrabbit	2,401
Bugzilla	Rhino	1,226
	Tomcat5	584

Herzig และคณะ [15] ได้เผยแพร่หมายเลยของรายงานจุดบกพร่องคัดกรองในรูปแบบไฟล์ CSV ดังภาพที่ 3-7 ซึ่งสามารถเข้าถึงผ่านเว็บไซต์ที่ https://www.st.cs.uni-saarland.de/softevo//bugclassify/ โดยสามารถนำหมายเลยรายงานจุดบกพร่องไปใช้ในการค้นหารายงานจุดบกพร่องจากระบบติดตามจุดบกพร่อง Jira ดังภาพที่ 3-8



ภาพที่ 3-7 ชุดรายงานจุดบกพร่องของ Herzig ในรูปแบบไฟล์ CSV



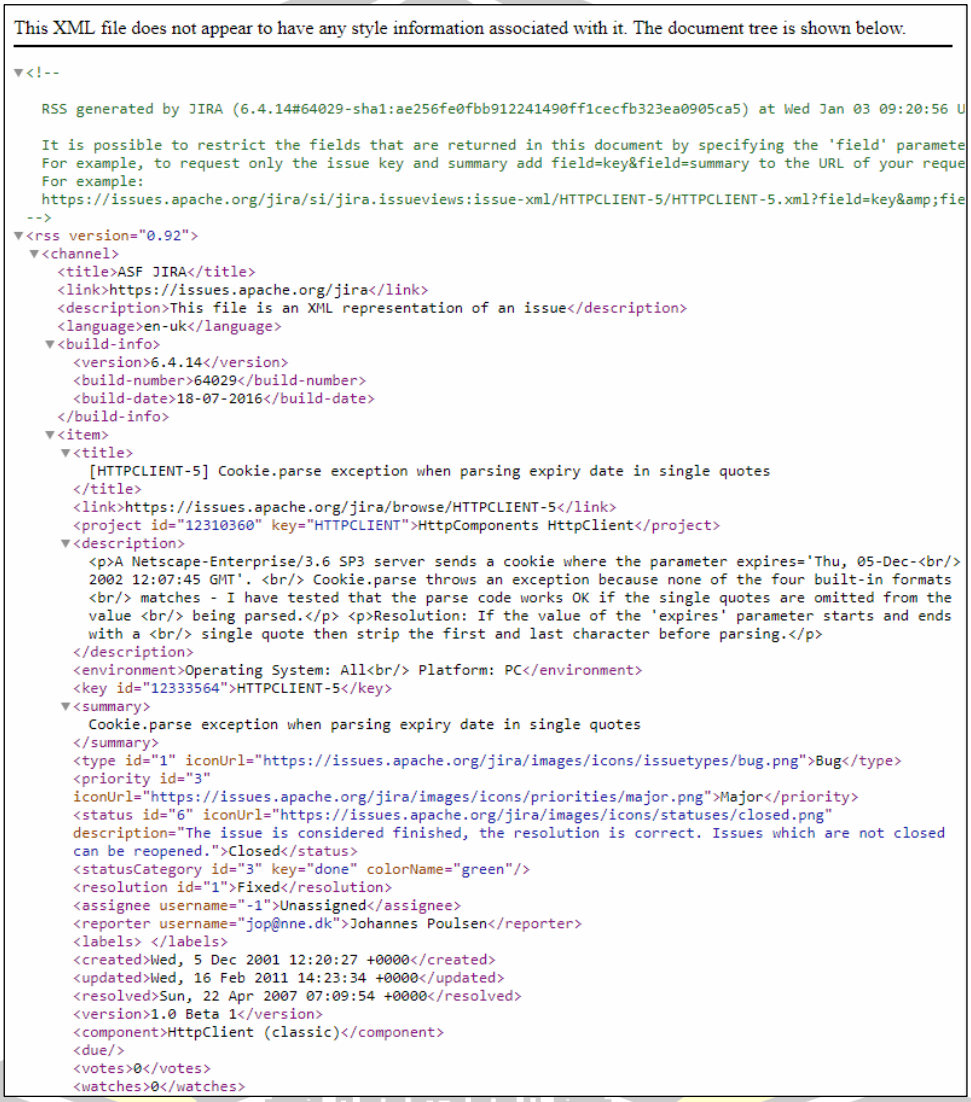
ภาพที่ 3-8 ตัวอย่างรายงานจุดบกพร่องจากระบบติดaตาม Jira
ที่มา : (https://issues.apache.org/jira/browse/HTTPCLIENT-302)

ข้อมูลรายงานจุดบกพร่องของชุดข้อมูล Herzig ที่นิยมใช้รายงานยจุดบกพร่องจาก HttpClient, Lucene และ Jackrabbit ซึ่งเป็นรายงานจุดบกพร่องจากระบบติดตามจุดบกพร่อง Jira ในการศึกษาการจำแนกรายงานจุดบกพร่องออกเป็นรายงานจุดบกพร่อง และรายงานที่ไม่ใช่จุดบกพร่อง โดยมีจำนวนรายงานจุดบกพร่อง ดังตารางที่ 3-4 โดยใช้เฉพาะข้อมูลส่วนสรุปของรายงานจุดบกพร่อง ซึ่งเป็นข้อความที่อธิบายของจุดบกพร่องที่พบในซอฟแวร์ เช่นเดียวกับส่วนคำอธิบายของรายงานจุดบกพร่อง (bug report description)

ตารางที่ 3-4 ชุดข้อมูล Herzig จากระบบติดตามจุดบกพร่อง Jira

โอเพนซอร์ส	จำนวนรายงานจุดบกพร่องทั้งหมด	จำนวนรายงานที่จุดบกพร่อง	จำนวนรายงานที่ไม่ใช่จุดบกพร่อง
HttpClient	745	305	440
Lucene	2,441	697	1,744
Jackrabbit	2,401	937	1,464

ซึ่งในงานวิจัยนี้เก็บรวบรวมรายงานจุดบกพร่องของชุดข้อมูล Herzig จากเว็บของระบบติดตามรายงานจุดบกพร่อง Jira ในรูปแบบไฟล์เอกซ์เอ็มแอล โดยในแต่ละรายงานจะมีลักษณะดังภาพที่ 3-9



ภาพที่ 3-9 ข้อมูลรายงานจุดบกพร่องของ Herzig ในรูปแบบไฟล์ XML

3.13 สรุปเกี่ยวกับข้อมูลรายงานจุดบกพร่องที่ใช้ในงานวิจัย

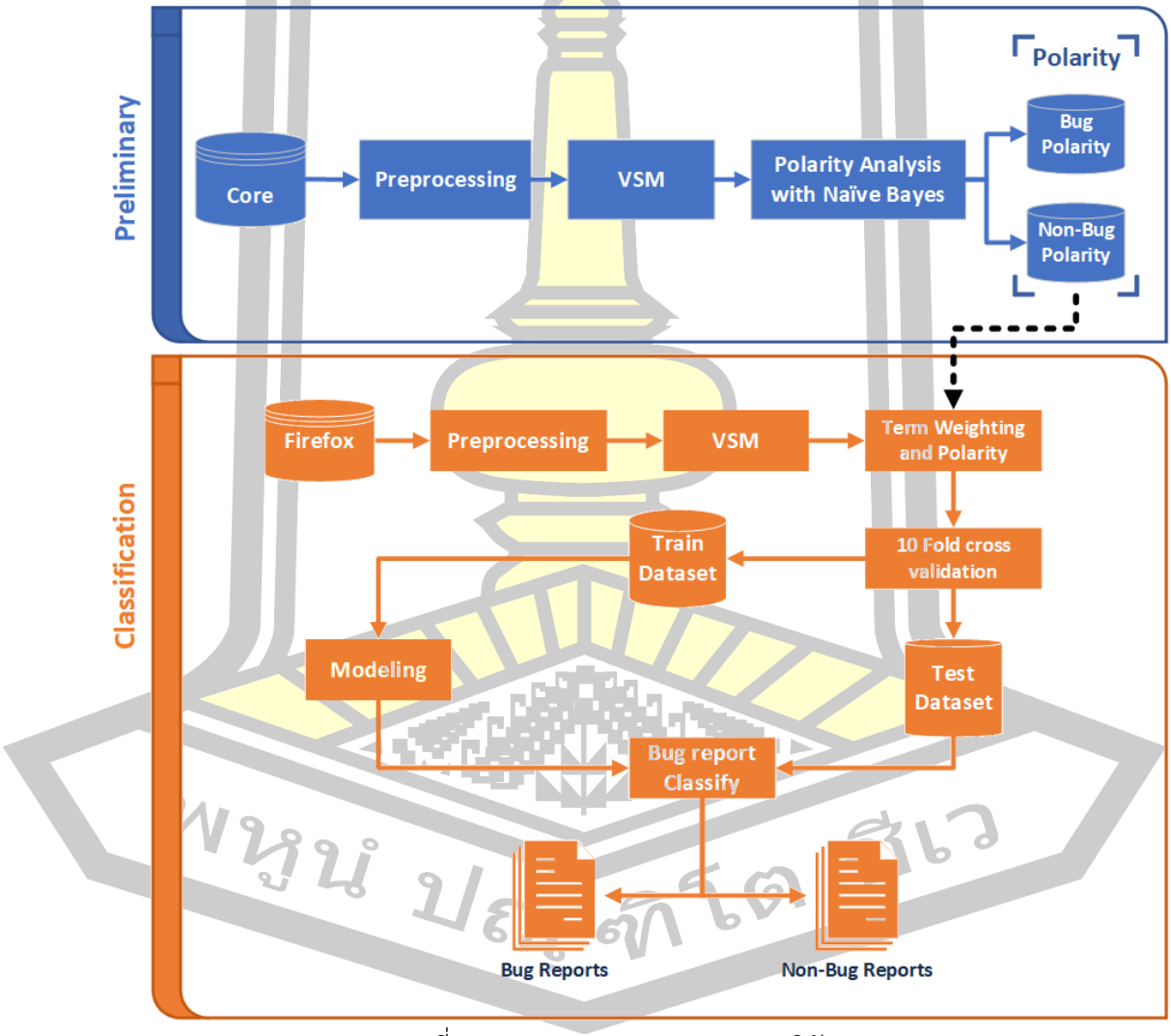
ข้อมูลรายงานจุดบกพร่องที่ใช้วิจัยประกอบด้วยข้อมูลรายงานจุดบกพร่องของ Core และข้อมูลรายงานจุดบกพร่องของ Firefox ซึ่งข้อมูลใช้มีลักษณะโครงสร้างของข้อมูล และมีกระบวนการในการเก็บรวบรวมข้อมูลที่เหมือนกัน แต่มีนำไปใช้ในกระบวนการที่ต่างกันซึ่งสามารถสรุปได้ดังตารางที่ 3-5

ตารางที่ 3-5 เปรียบเทียบข้อมูลที่ใช้ในงานวิจัย

ชุดข้อมูล	จำนวนรายงาน	ใช้ในกระบวนการ
Core	12,459 รายงาน	การสร้างคลังขั้วของคำ
Firefox	50,000 รายงาน	การศึกษาการจำแนกรายงานจุดบกพร่อง
Herzig	5,587 รายงาน	การศึกษาการจำแนกรายงานจุดบกพร่อง

3.2 กรอบการดำเนินงานวิจัย (Research Framework)

การในการศึกษาการจำแนกรายงานจุดบกพร่องออกเป็น รายงานจุดบกพร่อง และรายงานที่ไม่ใช่จุดบกพร่องจะมีขั้นตอนการศึกษาดังภาพที่ 3-10

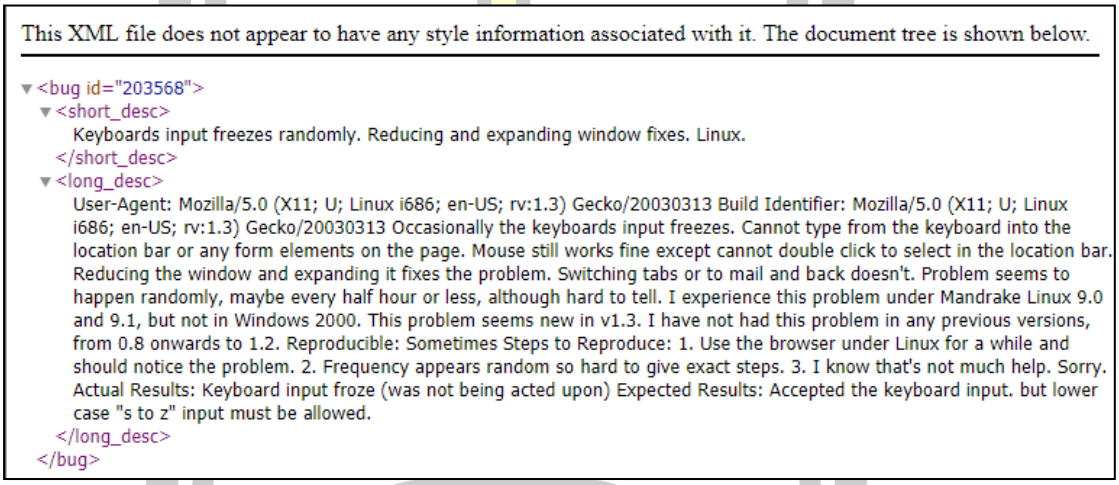


ภาพที่ 3-10 ภาพรวมของกระบวนการวิจัยการ

จากภาพที่ 3-10 ขั้นตอนในการศึกษาการจำแนกรายงานจุดบกพร่องจะแบ่งกระบวนการในการดำเนินงานเป็น 2 ส่วน ได้แก่

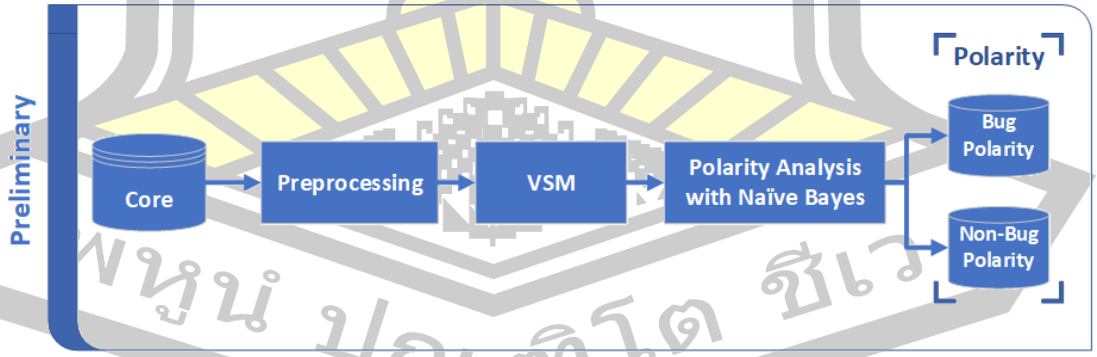
3.2.1 การเตรียมการเบื้องต้น (Preliminary)

การเตรียมการเบื้องต้น เป็นขั้นตอนการเตรียมการก่อนเข้าสู่กระบวนการสร้างแบบจำลองด้วยการสร้างคลังขั้ว (Polarity Corpus) ของคำที่แสดงความเป็นรายงานจุดบกพร่อง และไม่เป็นรายงานจุดบกพร่อง โดยใช้ข้อมูลรายงานจุดบกพร่องของ Core ในการสร้างคลังขั้วของคำ ดังภาพที่ 3-11



ภาพที่ 3-11 ตัวอย่างข้อมูลรายงานจุดบกพร่องของ Core

ซึ่งการเตรียมการเบื้องต้นมีขั้นตอนการดำเนินการสร้างคลังขั้วของคำแสดงความเป็นรายงานจุดบกพร่อง และไม่เป็นรายงานจุดบกพร่อง ดังภาพที่ 3-12



ภาพที่ 3-12 การเตรียมการเบื้องต้น

จากภาพที่ 3-12 ในการเตรียมการเบื้องประกอบด้วยกระบวนการทั้งหมด 3 ขั้นตอน ดังนี้

1) การเตรียมข้อมูล (Preprocessing)

การเตรียมข้อมูล เป็นกระบวนการในการแปลงข้อมูล ซึ่งเป็นภาษาธรรมชาติที่มนุษย์เข้าใจให้คอมพิวเตอร์เข้าใจ และสามารถนำไปประมวลผลได้ โดยในงานวิจัยนี้ได้มีขั้นตอนการเตรียมข้อมูลดังนี้

การตัดคำ (Tokenization)

การตัดคำ เป็นกระบวนการในการแบ่งคำออกเป็นคำ ๆ โดยในศึกษานี้ได้ใช้การตัดคำโดยใช้ช่องว่างระหว่างคำ (Space) [23] ซึ่งเป็นเทคนิคที่ง่ายที่สุด และนิยมใช้ตัดคำในภาษาอังกฤษดังในงานวิจัย [15, 16, 19-21, 27] โดยการใช้ช่องว่างระหว่างคำ (Space) เป็นตัวแบ่งคำออกเป็นคำๆ ดังตารางที่ 3-6

ตารางที่ 3-6 ตัวอย่างการตัดคำ

ข้อความต้นฉบับ	consider lowering priority of svg images to equal raster image preloads priority
ตัดคำด้วยช่องว่าง	consider | lowering | priority | of | svg | images | to | equal | raster | image | preloads | priority

จากตารางที่ 3-6 จะเห็นได้ว่า เมื่อข้อความต้นฉบับผ่านการกำจัดคำหยุดจะมีจำนวนคำลดลง โดยคำที่ถูกกำจัดได้แก่คำว่า *"of"* และ *"to"* ซึ่งเป็นคำที่มีอยู่ในคลังคำหยุดจึงถูกตัดออก

ภายหลังจากการตัดคำด้วยช่องว่างแล้ว ยังพบว่ามีคำที่มีลักษณะเป็นคำ Camel case ซึ่งเป็นคำที่เขียนติดกันโดยไม่เว้นวรรค โดยที่ตัวแรกของคำจะเป็นตัวพิมพ์ม ตัวอย่างเช่น addressBar, arrowScrollbox, IShellFolder, WebSockets เป็นต้น โดยแท้จริงแล้วไม่ได้เป็นคำเดียว จึงต้องแยกออกจากกัน

การกำจัดคำประเภทคำนาม

เพื่อให้ได้คลังขั้วคำที่เป็นไม่เฉพาะเจาะจงจากรายงานจุดบกพร่องของ Core จึงจะต้องตัดคำที่พบเฉพาะในรายงานจุดบกพร่องของ Core ออก ซึ่งส่วนมากจะเป็นคำประเภทคำนาม (Noun) โดยในการตัดคำประเภทคำนามจะเริ่มต้นโดยการระบุชนิดของคำ (Past of speed tagging: POS Tagging) ซึ่งในงานวิจัยนี้ใช้การระบุหน้าที่ของคำ โดยใช้เทคนิค Penn Treebank ในการระบุคำ แล้วจึงตัดคำที่เป็นคำประเภทคำนามทิ้ง ดังตารางที่ 3-7

ตารางที่ 3-7 ตัวอย่างการกำจัดคำนาม

ข้อความต้นฉบับ	consider | lowering | priority | of | svg | images | to | equal | raster | image | preloads | priority
ข้อความที่ได้รับการระบุหน้าที่คำ	<consider_VB > | <lowering_JJ> | <priority_NN | <of_IN> | <svg_JJ> | <images_NNS> | <to_TO> | <equal_VB> | <raster_NN> | <image_NN> | <preloads_VBZ> | <priority_NN>
ข้อความที่การตัดคำนาม	consider | lowering | of | svg | to | equal | preloads

การทำความสะอาดข้อมูลข้อความ (Text Cleaning)

การทำความสะอาดข้อมูลข้อความ เป็นกระบวนการในการตัดตัวอักขระเครื่องหมายวรรคตอน หรือคำที่ไม่ไม่มีความหมาย โดยไม่ทำให้ความหมายของข้อความเปลี่ยน เพื่อเป็นการลดขนาดของคคุณลักษณะที่ใช้ในการประมวลผล ซึ่งส่งผลต่อประสิทธิภาพในการจำแนกรายงานจุดบกพร่องซึ่งในงานวิจัยนี้มีกระบวนการในการทำความสะอาดข้อมูลข้อความรายงานจุดบกพร่อง ดังนี้

(1) การกำจัดเครื่องหมายวรรคตอน (Punctuation Removal) เป็นขั้นตอนในการตัดคคุณลักษณะที่เป็นเครื่องหมายวรรคตอนที่พบในข้ทิ้ง ตัวอย่างเช่น “!”, “?”, “#”, “+”, “;” เป็นต้น

(2) การกำจัดตัวเลข (Number Removal) เป็นขั้นตอนในการตัดคคุณลักษณะที่เป็นตัวทิ้ง

(3) การกำจัดตัวอักขระเดี่ยว (Single Character Removal)) เป็นขั้นตอนในการตัดคคุณลักษณะที่เป็นตัวอักษรเดี่ยว หรืออักขระเดี่ยวทิ้ง

(4) กำจัดคำหยุด (Stop word Removal) เป็นขั้นตอนในการตัดคำที่ไม่มีนัยสำคัญ แต่เป็นคำที่ปรากฏในรายงานจุดบกพร่องอยู่ในทุกรายงาน และปรากฏในปริมาณที่มาก โดยในการงานวิจัยนี้ได้ใช้คลังคำหยุดของ NLTK [35] ซึ่งมีคำหยุดดังภาพที่ NLTK เป็นคลังคำหยุดที่นิยมใช้กันอย่างแพร่หลาย [23, 24, 28, 29] โดยการกำจัดคำหยุดจะมีลักษณะดังตารางที่ 3-8

ตารางที่ 3-8 ตัวอย่างการกำจัดคำหยุด

ข้อความต้นฉบับ	consider | lowering | of | svg | to | equal | preloads
ข้อความที่กำจัดคำหยุด	consider | lowering | svg | equal | preloads

การเปลี่ยนรูปคำ

เป็นกระบวนการในการลดความกำกวมของคำ ซึ่งส่งผลต่อขนาดของคลังคำ และมีผลต่อประสิทธิภาพในการเปรียบเทียบคำ โดยในงานวิจัยนี้ได้ทดสอบการเปลี่ยนรูปคำ 2 เทคนิค ดังนี้

(1) การตัดส่วนขยาย [29, 31] เป็นกระบวนทำนอร์มัลไลซิ่งข้อความ โดยการตัดส่วนขยายของคำทิ้ง เช่น s, es, er, ing หรือ ed เป็นต้น ซึ่งในการวิจัยนี้ได้ใช้อัลกอริทึม Porter ในการตัดส่วนขยายของคำ เนื่องจากเป็นอัลกอริทึมที่ทำงานได้รวดเร็ว และนิยมใช้ในการจำแนกรายงานจุดบกพร่อง [17, 19] ตัวอย่างการตัดส่วนขยายของคำแสดงได้ดังตารางที่ 3-9

ตารางที่ 3-9 ตัวอย่างการตัดส่วนขยาย

ข้อความต้นฉบับ	consider | lowering | svg | equal | preloads
ข้อความที่ตัดส่วนขยายแล้ว	consid | lower | svg | equal | preload

(2) การเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม [24, 28, 36] เป็นกระบวนที่มีลักษณะคล้ายกับการตัดส่วนขยาย (Stemming) แต่จะมีการวิเคราะห์ที่ละเอียดกว่า โดยใช้บริบทหรือชนิดของคำ (Parts of Speech) ประกอบในการพิจารณาเปลี่ยนรูปของคำ ดังนั้นการเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิมจำเป็นต้องใช้พจนานุกรมที่มีได้รับการระบุชนิดคำ ซึ่งในงานวิจัยนี้ได้ใช้พจนานุกรมเวิร์ดเน็ต [24] (WordNet) ในการเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิมสามารถแสดงดังในตารางที่ 3-10

ตารางที่ 3-10 ตัวอย่างการเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม

ข้อความต้นฉบับ	consider | lowering | svg | equal | preloads
ข้อความที่เปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม	consider | lower | svg | equal | preloads

2) การสร้างตัวแทนเอกสาร

เป็นกระบวนการแปลงเอกสารที่เป็นภาษาธรรมชาติให้อยู่ในรูปแบบที่คอมพิวเตอร์สามารถเข้าใจ และนำไปประมวลผลได้ โดยการแปลงข้อความในรายงานจุดบกพร่องให้อยู่ในรูปของเวกเตอร์ที่แสดงความสัมพันธ์คำที่พบในรายงานจุดบกพร่อง และรายงานจุดบกพร่อง ดังตัวอย่างในตารางที่ 3-11

ตารางที่ 3-11 ตัวอย่างการสร้างตัวแทนเอกสาร

	consider	equal	image	lower	preloads	priority	raster	svg
D_1	1	1	2	1	1	2	1	1
D_2	0	3	6	2	0	1	0	0
D_3	1	0	1	3	2	0	1	5
D_4	1	3	0	0	0	1	1	0

2) การวิเคราะห์ขั้วของคำ

เป็นการสร้างคลังขั้วของคำที่แสดงความเป็นรายงานจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่อง โดยการนำแนวคิดจากวิจัยของ Isa และคณะ [37] ที่ได้นำเสนอการเตรียมข้อมูลข้อความในเอกสารด้วยทฤษฎีนาอีฟเบย์มาประยุกต์ในการสร้างคลังขั้วของคำ ซึ่งจะนำไปใช้ประกอบในการวิเคราะห์ขั้วของคำก่อนการสร้างแบบจำลองการจำแนกข้อมูลประเภทเอกสาร ซึ่งสามารถคำนวณค่าขั้วของคำได้ดังสมาการที่ (3.1)

$$P(Category \mid Word) = \frac{P(Word \mid Category) \times P(Category)}{P(Word)} \quad (3.1)$$

ตัวอย่างเช่น สมมุติให้มีรายงานจุดบกพร่องจำนวน 200 รายงาน และที่ไม่ใช่รายงานจุดบกพร่องจำนวน 200 รายงาน โดยในรายงานจุดบกพร่องมีคำทั้งหมด 27,452 คำ และที่ไม่ใช่รายงานจุดบกพร่องมีคำทั้งหมด 22,914 คำ ซึ่งมีคำที่เป็นซ้ำกันทั้งในรายงานจุดบกพร่อง และไม่ใช่รายงานจุดบกพร่องจำนวน 501 คำ และ 457 คำ ตามลำดับ หากต้องการวิเคราะห์ขั้วความเป็นรายงานจุดบกพร่องของคำว่า *“Crash”* จากรายงานทั้งหมด โดยที่พบคำว่า *“Crash”* ในรายงานจุดบกพร่องทั้งหมด 795 ครั้ง และไม่ใช่รายงานจุดบกพร่อง 142 ครั้ง สามารถคำนวณได้ดังนี้

ขั้นตอนที่ 1 คำนวณหาความน่าจะเป็นที่จะพบคำว่า *“crash”* หรือ P(crach) ซึ่งสามารถคำนวณได้ดังนี้

$$P(Crach) = \frac{795+142}{27{,}452+22{,}914} = 0.019 \quad (3.2)$$

ขั้นตอนที่ 2 คำนวณหาความน่าจะเป็นของกลุ่มที่เป็นรายงานจุดบกพร่อง หรือ $P(Category_{Bug})$ ซึ่งสามารถคำนวณได้ดังนี้

$$P(Category_{Bug}) = \frac{501}{501+457} = 0.523 \quad (3.3)$$

ขั้นตอนที่ 3 คำนวณหาความน่าที่ในกลุ่มของรายงานจุดบกพร่องจะพบคำว่า *“crash”* ซึ่งสามารถคำนวณได้ดังนี้

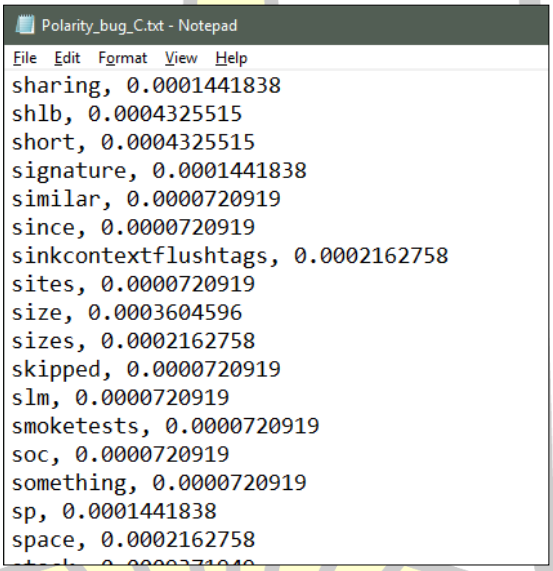
$$P(Crach \mid Category_{Bug}) = \frac{795}{27{,}452+22{,}914} = 0.016 \quad (3.4)$$

ขั้นตอนที่ 4 คำนวณหาค่าน่าจะเป็นของคำว่า “Crash” ในกลุ่มของรายงานจุดบกพร่อง ซึ่งสามารถคำนวณได้ดังนี้

$$P(Category_{Bug} \mid Crach) = \frac{P(Word \mid Category) \times P(Category)}{P(Word)} \quad (3.5)$$

$$P(Category_{Bug} \mid Crach) = \frac{0.016 \times 0.523}{0.019} = 0.44 \quad (3.6)$$

โดยเมื่อคำนวณความน่าจะของรายงานจุดบกพร่อง และไม่ใช่รายงานจุดบกพร่องของคำทั้งหมดจะเรียบร้อยแล้วจะรวบรวมผลที่ได้เก็บไว้ในรูปแบบไฟล์ .txt ดังภาพที่ 3-13

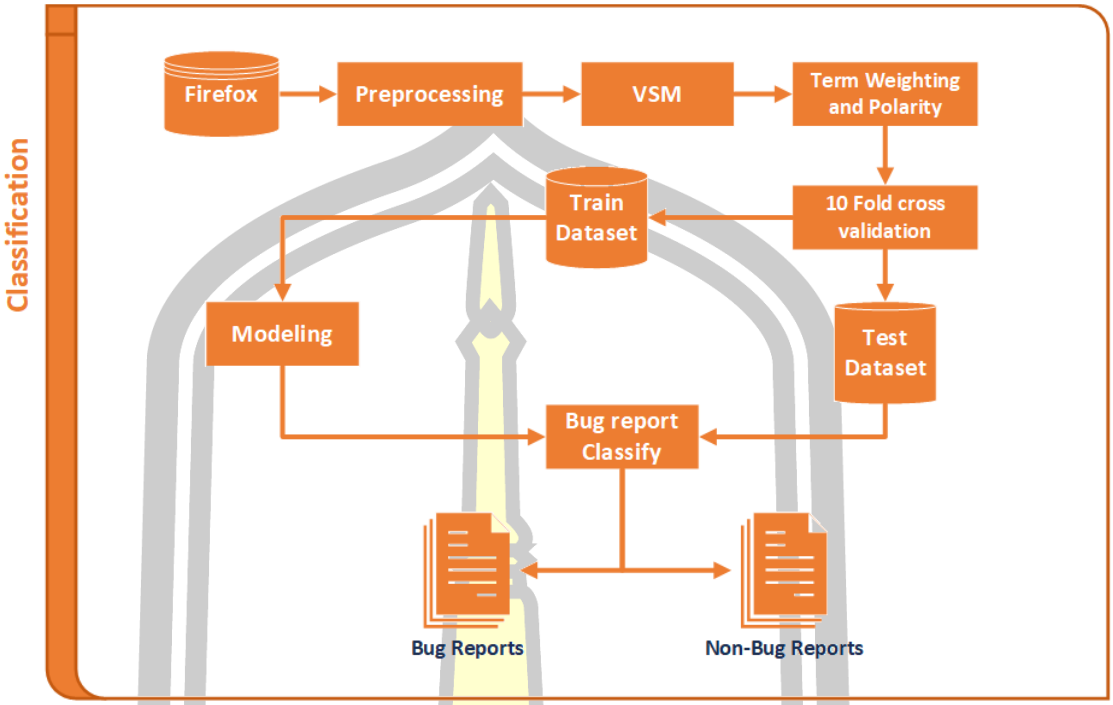


ภาพที่ 3-13 ตัวอย่างค่าขั้วของคำในรูปแบบไฟล์ txt

3.2.2 การสร้างแบบจำลองการจำแนก และการจำแนกรายงานจุดบกพร่อง

การสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง เป็นการสร้างแบบจำลองในการจำแนกประเภทของรายงานจุดบกพร่องออกเป็นรายงานจุดบกพร่อง และไม่เป็นรายงานจุดบกพร่อง โดยในงานวิจัยนี้ได้สร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง และทดสอบการจำแนกรายงานจุดบกพร่อง โดยใช้ชุดข้อมูลรายงานจุดบกพร่อง 2 ชุดข้อมูล คือ ชุดข้อมูลรายงานจุดบกพร่องของ Firefox และ ชุดข้อมูลรายงานจุดบกพร่องของ Herzig

โดยมีขั้นตอนในการดำเนินการสร้างแบบจำลอง และทดสอบประสิทธิภาพของแบบจำลองดังภาพที่ 3-14



ภาพที่ 3-14 กระบวนการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง

จากภาพที่ 3-14 ในการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่องจะประกอบด้วย การตัดคำ การกำจัดคำหยุด การเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม การสร้างตัวแทนรายงาน การให้น้ำหนักคำ การให้ค่าขั้วคำ และการสร้างแบบจำลองการจำแนก ดังที่จะอธิบายต่อจากนี้

1) การเตรียมข้อมูล (Preprocessing)

การเตรียมข้อมูล เป็นกระบวนการในการแปลงข้อมูล ซึ่งเป็นภาษาธรรมชาติที่มนุษย์เข้าใจให้คอมพิวเตอร์เข้าใจ และสามารถนำไปประมวลผลได้ โดยในงานวิจัยนี้ได้มีขั้นตอนการเตรียมข้อมูลดังนี้

การตัดคำ (Tokenization)

การตัดคำ เป็นกระบวนการในการแบ่งคำออกเป็นคำ ๆ โดยในศึกษานี้ได้ใช้การตัดคำโดยใช้ช่องว่างระหว่างคำ (Space) [23] ซึ่งเป็นเทคนิคที่ง่ายที่สุด และนิยมใช้ตัดคำในภาษาอังกฤษดังในงานวิจัย [15, 16, 19-21, 27] โดยการใช้ช่องว่างระหว่างคำ (Space) เป็นตัวแบ่งคำออกเป็นคำ ๆ ดังตารางที่ 3-12

ตารางที่ 3-12 ตัวอย่างการตัดคำ

ข้อความต้นฉบับ	s to z keyboard alphabet input blocked when blocking keyboard functional keys from F5 to F12 with javascript
ตัดคำด้วยช่องว่าง	s | to | z | keyboard | alphabet | input | blocked | when | blocking | keyboard | functional | keys | from | f5 | to | f12 | with | javascript

การทำความสะอาดข้อมูลข้อความ (Text Cleaning)

การทำความสะอาดข้อมูลข้อความ เป็นกระบวนการในการตัดตัวอักขระเครื่องหมายวรรคตอน หรือคำที่ไม่ไม่มีความหมาย โดยไม่ทำให้ความหมายของข้อความเปลี่ยน เพื่อเป็นการลดขนาดของคคุณลักษณะที่ใช้ในการประมวลผล ซึ่งส่งผลต่อประสิทธิภาพในการจำแนกรายงานจุดบกพร่องซึ่งในงานวิจัยนี้มีกระบวนการในการทำความสะอาดข้อมูลข้อความรายงานจุดบกพร่อง ดังนี้

(1) การกำจัดเครื่องหมายวรรคตอน (Punctuation Removal) เป็นขั้นตอนในการตัดคคุณลักษณะที่เป็นเครื่องหมายวรรคตอนที่พบในข้ทิ้ง ตัวอย่างเช่น “!”, “?”, “#”, “+”, “;” เป็นต้น

(2) การกำจัดตัวเลข (Number Removal) เป็นขั้นตอนในการตัดคคุณลักษณะที่เป็นตัวทิ้ง

(3) การกำจัดตัวอักขระเดี่ยว (Single Character Removal)) เป็นขั้นตอนในการตัดคคุณลักษณะที่เป็นตัวอักษรเดี่ยว หรืออักขระเดี่ยวทิ้ง

(4) กำจัดคำหยุด (Stop word Removal) เป็นขั้นตอนในการตัดคำที่ไม่มีนัยสำคัญ แต่เป็นคำที่ปรากฏในรายงานจุดบกพร่องอยู่ในทุกรายงาน และปรากฏในปริมาณที่มาก โดยในการงานวิจัยนี้ได้ใช้คลังคำหยุดของ NLTK [35] ซึ่งมีคำหยุดดังภาพที่ NLTK เป็นคลังคำหยุดที่นิยมใช้กันอย่างแพร่หลาย [23, 24, 28, 29] โดยการกำจัดคำหยุดจะมีลักษณะดังตารางที่ 3-13

ตารางที่ 3-13 ตัวอย่างการตัดคำหยุด

ข้อความต้นฉบับ	s | to | z | keyboard | alphabet | input | blocked | when | blocking | keyboard | functional | keys | from | f5 | to | f12 | with | javascript
ข้อความที่ตัดคำแล้ว	s | z | keyboard | alphabet | input | blocked | blocking | keyboard | functional | keys | f5 | f12 | javascript

การเปลี่ยนรูปคำ

เป็นกระบวนการในการลดความกำกวมของคำ ซึ่งส่งผลต่อขนาดของคลังคำ และมีผลต่อประสิทธิภาพในการเปรียบเทียบคำ โดยในงานวิจัยนี้ได้ทดสอบการเปลี่ยนรูปคำ 2 เทคนิค ดังนี้

(1) การตัดส่วนขยาย [29, 31] เป็นกระบวนทำนอร์มัลไลซิ่งข้อความ โดยการตัดส่วนขยายของคำทิ้ง เช่น s, es, er, ing หรือ ed เป็นต้น ซึ่งในการวิจัยนี้ได้ใช้อัลกอริทึม Porter ในการตัดส่วนขยายของคำ เนื่องจากเป็นอัลกอริทึมที่ทำงานได้รวดเร็ว และนิยมใช้ในการจำแนกรายงานจุดบกพร่อง [17, 19] ตัวอย่างการตัดส่วนขยายของคำแสดงได้ดังตารางที่ 3-14

ตารางที่ 3-14 ตัวอย่างการตัดส่วนขยาย

ข้อความต้นฉบับ	s | z | keyboard | alphabet | input | blocked | blocking | keyboard | functional | keys | f5 | f12 | javascript
ข้อความที่ตัดส่วนขยายแล้ว	s | z | keyboard | alphabet | input | block | block | keyboard | function | key | f5 | f12 | javascript

(2) การเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม [24, 28, 36] เป็นกระบวนที่มีลักษณะคล้ายกับการตัดส่วนขยาย(Stemming) แต่จะมีการวิเคราะห์ที่ละเอียดกว่า โดยใช้บริบทหรือชนิดของคำ (Parts of Speech) ประกอบในการพิจารณาเปลี่ยนรูปของคำ ดังนั้นการเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิมจำเป็นต้องใช้พจนานุกรมที่มีได้รับการระบุชนิดคำ ซึ่งในงานวิจัยนี้ได้ใช้พจนานุกรมเวิร์ดเน็ต (WordNet) [24] ในการเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิมสามารถแสดงได้ดังในตารางที่ 3-15

ตารางที่ 3-15 ตัวอย่างการเปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม

ข้อความต้นฉบับ	s | z | keyboard | alphabet | input | blocked | blocking | keyboard | functional | keys | f5 | f12 | javascript
ข้อความที่เปลี่ยนรูปคำให้อยู่ในรูปแบบดั้งเดิม	s | z | keyboard | alphabet | input | block | block | keyboard | functional | key | f5 | f12 | javascript

2) การสร้างตัวแทนเอกสาร

เป็นกระบวนการแปลงเอกสารที่เป็นภาษาธรรมชาติให้อยู่ในรูปแบบที่คอมพิวเตอร์สามารถเข้าใจ และนำไปประมวลผลได้ โดยการแปลงข้อความในรายงานจุดบกพร่องให้อยู่ในรูปของเวกเตอร์ที่แสดงความสัมพันธ์คำที่พบในรายงานจุดบกพร่อง และรายงานจุดบกพร่อง ดังตัวอย่างในตารางที่

สมมติกำหนดให้รายงานจุดบกพร่อง 4 รายงาน มีข้อความดังนี้

D_1: pressing delete with nothing selected breaks delete in bookmark manager

D_2: cannot file new bookmark while managing a new empty folder

D_3: two bugs for the price of one loading the page causes four instances to be loaded into history and mouseover causes constant loading

D_4: when setup so that there are subfolders in bookmarks folders hovering the mouse over folder does not open folder

จากรายงานจุดบกพร่อง 4 รายงาน โดยที่รายงานจุดบกพร่อง D_1 และ D_2 เป็นรายงานจุดบกพร่อง และ D_3 และ D_4 เป็นรายงานที่ไม่ใช่จุดบกพร่อง และเลือกตัวแทนรายงานจุดบกพร่อง 5 คำ เมื่อแสดงให้อยู่รูปถุงคำที่แสดงความสัมพันธ์ระหว่างรายงานจุดบกพร่อง และคำ จะได้ดังตารางที่ 3-16

ตารางที่ 3-16 ความสัมพันธ์ระหว่างรายงานุดบกพร่อง และคำในรูปถุงคำ

Documents	words					class
	bookmark	folder	manager	mouse	over	
D_1	6	0	6	0	0	Bug
D_2	14	6	2	0	0	Bug
D_3	0	0	0	3	3	Non-bug
D_4	8	13	0	2	4	Non-bug

2) การให้น้ำหนักคำ (Term Weighting)

เป็นการให้น้ำหนัก หรือค่าของคำที่อยู่ในรูปของถุงคำ ซึ่งจะการเปรียบเทคนิคการให้น้ำหนักคำ 4 เทคนิค ได้แก่ TF, TF-IDF, BM25 และ MATF เพื่อหาเทคนิคการให้น้ำหนักที่เหมาะสมกับการจำแนกรายงานจุดบกพร่องที่สุด

(1) การให้น้ำหนักคำด้วยเทคนิค TF เป็นเทคนิคการให้น้ำหนักโดยใช้ความถี่ขอคำที่พบในรายงานจุดบกพร่อง ซึ่งเป็นเทคนิคที่นำมาใช้งานการจำแนกรายงานจุดบกพร่องดังในงานวิจัย [16, 17, 19, 21, 22]

(2) การให้น้ำหนักคำด้วยเทคนิค TF-IDF เป็นเทคนิคการให้น้ำหนักคำที่ได้รับความนิยมและใช้กันอย่างแพร่หลาย ตัวอย่างเช่นงานวิจัย [17, 38-40] เนื่องด้วยเป็นเทคนิคการให้น้ำหนักที่มีประสิทธิภาพ และง่ายในการคำนวณค่าน้ำหนัก โดยการให้น้ำหนักด้วยเทคนิค tf-idf สามารถคำนวณได้จากสมการที่ (2.2) ในบทที่ 2

(3) การให้น้ำหนักคำด้วยเทคนิค BM25 เป็นเทคนิคการให้น้ำหนักคำในเอกสารที่มีประสิทธิภาพในการจัดลำดับการค้นคืนเอกสาร ซึ่งเป็นเทคนิคหนึ่งที่ถูกนำมาใช้งานวิจัยที่เกี่ยวข้องรายงานจุดบกพร่อง ตัวอย่างเช่น การคัดกรองรายงานจุดบกพร่องที่ซ้ำซ้อนดังในงานวิจัย [41] การจัดลำดับความรุนแรงของรายงานจุดบกพร่องดังในงานวิจัย [42] และการแนะนำผู้แก้ไขจุดบกพร่องที่เหมาะสม [43] โดยเทคนิค BM25 มีพื้นฐานมาจากการให้น้ำหนักด้วยเทคนิค TF-IDF ซึ่ง BM25 มีการนำความยาวของเอกสารมาพิจารณาประกอบกับตัวแปรอิสระที่สามารถปรับให้เหมาะสมกับข้อมูลที่ใช้ได้ ได้ดังสมการที่ (2.3) ในบทที่ 2

(4) การให้น้ำหนักคำด้วยเทคนิค MATF [44] เป็นเทคนิคการให้น้ำหนักคำที่ได้รับการนำเสนอในปี 2013 โดย Jiaul H. Paik [44] เทคนิค MATF เป็นการให้น้ำหนักคำที่คำนึงถึงความยาวของข้อความในเอกสารที่ไม่เท่ากัน ซึ่งอาจเหมาะสมกับการให้น้ำหนักคำในรายงานจุดบกพร่องที่มีความยาวของรายงานจุดบกพร่องแต่ละรายงานที่ไม่เท่ากัน โดย MATF สามารถคำนวณค่าน้ำหนักของคำได้ดังสมการ (2.4) ในบทที่ 2

ภายหลังจากการให้น้ำหนักคำด้วย 4 เทคนิคที่ได้กล่าวมาในขั้นต้น ในการวิจัยนี้ได้เพิ่มกระบวนการในการให้น้ำหนักเพิ่มเติม คือ การให้ค่าขั้วของคำ ซึ่งเป็นนำค่าขั้วแสดงความเป็นรายงานจุดบกพร่อง และรายงานที่ไม่ใช่จุดบกพร่องจากคลังขั้วคำที่สร้างไว้มาให้น้ำหนักกับคำในถุงคำที่ได้รับให้น้ำหนักคำจากกระบวนการก่อนหน้าโดยค่าน้ำหนักที่คำนวณได้แทนด้วย X_i ซึ่งสามารถคำนวณได้จากสมการที่ (3.7)

$$X_i = w_i + Polarity_{c,i} \tag{3.7}$$

โดยที่ w_i คือ คำที่ I ที่ผ่านการน้ำหนักในกระบวนการข้าวต้น

$Polarity_{c,i}$ คือ ค่าขั้วของคำที่ I ในกลุ่ม C

3) การแบ่งข้อมูลสำหรับสร้างแบบจำลอง และข้อมูลทดสอบ

ในงานวิจัยนี้ได้ใช้การแบ่งข้อมูลสำหรับสร้างแบบจำลอง และข้อมูลทดสอบของรายงานจุดบอกพร่องด้วยวิธี cross Validation โดยใช้ค่า k = 10 หรือ 10-fold cross validation ซึ่งจะแบ่งข้อมูลรายงานจุดบกพร่องออกเป็น 10 ส่วนเท่าๆ กัน จากนั้นใช้ข้อมูลรายงานจุดบกพร่องหนึ่งส่วนเป็นข้อมูลทดสอบประสิทธิภาพของแบบจำลอง และส่วนที่เหลือเก้าส่วนเป็นข้อมูลรายงานจุดบกพร่องพร่องสำหรับใช้ในการสร้างแบบจำลอง โดยจะทำวนเช่นนี้ไปจนครบตามจำนวนที่แบ่งไว้ ซึ่งวิธีการแบ่งข้อมูลด้วนวิธี 10-fold cross validation เป็นวิธีที่นิยมใช้ในการจำแนกรายงานจุดบกพร่องดังในงานวิจัย [16, 17, 19, 22]

4) การสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง

เป็นการสร้างแบบจำลองจาก นำถุงคำที่ผ่านกระบวนการให้น้ำหนักคำแล้วมาเป็นข้อมูลในฝึกสอน โดยการศึกษานี้ได้นำเสนอการสร้างแบบจำลองจากอัลกอริทึม 4 อัลกอริทึม ได้แก่ นาอีฟเบย์ ซัพพอร์ตเวกเตอร์แมชชีน การถดถอยโลจิสติก และแรนดอมฟอเรส เพื่อหาอัลกอริทึมที่เหมาะสมในการจำแนกรายงานจุดบกพร่อง

การสร้างแบบจำลองด้วยอัลกอริทึมนาอีฟเบย์

เป็นเทคนิคการจำแนกประเภทของข้อมูลที่มีประสิทธิภาพวิธีหนึ่ง และยังเป็นเทคนิคที่นิยมใช้ในการจำแนกรายงานจุดบกพร่อง ดังในงานวิจัย [16, 19-22, 27] ที่ได้ทำการวิจัยการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึมนาอีฟเบย์ ซึ่งในงานวิจัยของ [22] พบว่าแบบจำลองในการจำลองที่สร้างด้วยอัลกอริธึมนาอีฟเบย์มีประสิทธิภาพในการจำแนกรายงานจุดบกพร่องได้ดีกว่าอัลกริทึมอื่นที่ได้ทำการทดสอบ ซึ่งสามารถสร้างแบบจำลองได้ดังภาพที่ 3-15

Algorithm: Train Multinomial Naïve Bayes

Input: R = {Report$_1$, Report$_2$, Report$_3$, …, Report$_n$}
V ← Extract Vocabulary (R)
N ← Count Report (R)
C = {bug, nug}
Output: V, Prior, condProb
1. **foreach** c ∈ C do
2. N_c ← Count Report in Class (R, c)
3. Prior[c] ← N_c / N
4. $text_c$ ← Concatenate Text of Report in Class (R, c)
5. **foreach** t ∈ V do
6. $t_{c,t}$ ← Count Tokens of Term ($text_c$, t)
7. **end**
8. **foreach** t ∈ V do
9. condprob [t][c] ← $\frac{T_{c,t}+1}{\sum(T_{c,t}+1)}$
10. **end**

Algorithm: TestMultinomial Naïve Bayes

Input: V, Prior, condProb, $Report_{new}$
Output: $argMax_c$ Score[c]
1. W ← Extract Tokens from Report(V, $Report_{new}$)
2. **foreach** c ∈ C **do**
3. socre[c] ← log prior[c]
4. **foreach** t ∈ w **do**
5. score[c] += log condProb[t][c]
6. **end**
7. **end**

ภาพที่ 3-15 การสร้างแบบจำลองการจำแนกด้วยอัลกอริธึมนาอีฟเบย์

การสร้างแบบจำลองด้วยอัลกอริทึมพอร์ตเวกเตอร์แมชชีน

เป็นเทคนิคที่จำแนกข้อมูลออกเป็น 2 ส่วนมีที่ประสิทธิภาพ และถูกนำมาใช้ในการจำแนกรายงานจุดบกพร่อง ดังในงานวิจัย [16, 19-22, 27] ซึ่งอัลกอริทึมพอร์ตเวกเตอร์แมชชีนมีขั้นตอนในการจัดกลุ่มรายงานจุดบกพร่องดังต่อไปนี้

1. คำนวณหาค่า y ของรายงานจุดบกพร่อง ซึ่งเป็นค่าของ $y \in \{-1,1\}$ และ ค่า $x \in R^n$ โดยที่

- ถ้าค่าของ $w^\top x+b>0$ จะกำหนดให้ค่า y = +1 ซึ่งจะจัดให้อยู่ใน Class “รายงานจุดบกพร่อง”
- ถ้าค่าของ $w^\top x+b<0$ จะกำหนดให้ค่า y = -1 ซึ่งจะจัดให้อยู่ใน Class “รายงานที่ไม่เป็นจุดบกพร่อง”

2. คำนวณหาเส้นระนาบตัดสินใจที่แบ่งรายงานจุดบกพร่อง หรือที่เรียกว่า Optimal Hyperplane จากสมการ $w^Tx+b=0$

3. นำค่าที่ได้จากขั้นตอนที่ 1 และ 2 มาจัดให้อยู่ในรูปคุณลักษณะสเปซ (Feature Space) เพื่อที่จะหาจุดที่ใกล้เคียงกับเส้นระนาบตัดสินใจที่ โดยจุดที่อยู่เหนือเส้น เส้นระนาบตัดสินใจที่จะเรียกว่า “ขอบบน” และจุดที่อยู่ใต้เส้นจะเรียกว่า “ขอบล่าง”

4. คำนวณหาระยะทางระหว่างเส้นขอบทั้งสอง โดยจะเลือกเอาค่าระยะทางที่ห่างจากเส้นระนาบตัดสินใจที่น้อยที่สุดเป็นตัวแทนในการจัดกลุ่มรายงานจุดบกพร่อง โดยระยะทาง (d) หรือ Maximum Margin จากเส้นขอบ ณ จุด x_i ไปยังเส้นระนาบตัดสินใจ สามารถคำรวณได้จากสมการที่ สมการที่ (3.8)

$$d = \frac{\left|w^T x_i + b\right|}{\left\|w\right\|} \tag{3.8}$$

5. ในกรณีที่ตัวแทนเอกสารไม่สามารจัดกลุ่มได้ด้วยเส้นตรง (Linear) ข้อมูลรายงานนั้นไม่ได้อยู่ในรูปแบบของ Non-Linear Support Vector Machines จึงไม่สามารถแบ่งกลุ่มรายงานจุดบกพร่องได้ เนื่องจากไม่มีจุดที่แน่นอนที่สามารถใช้เป็นหลักในการแบ่งเส้นระบบตัดสินใจได้ ดังนั้นจะต้องทำการปรับจุดโดยใช้ Kernel Function เพื่อให้การทำงานทำได้ง่ายขึ้น และทำให้การแบ่งกลุ่มรายงานจุดบกพร่องได้ถูกต้องยิ่งขึ้น

6. หลังจากการทำการปรับจุดดด้วย Kernel Function แล้วจึงทำการแบ่งกลุ่มให้กับรายงานจุดบกพร่อง

โดยรายงานจุดบกพร่องที่ได้จากการคำนวณ จะพิจารณาว่าถ้า $f(x)>0$ จะจัดอยู่ในกลุ่มที่ 1 หรือกลุ่มของรายงานจุดบกพร่อง และถ้า $f(x)<0$ จะจัดอยู่ในกลุ่มที่ 2 หรือ กลุ่มของรายงานที่ไม่เป็นจุดบกพร่อง

การจำแนกรายงานจุดบกพร่องด้วยการแบบจำลองซัพพอร์ตเวกเตอร์แมชชีน โดยการคำนวณจากสมการ (3.9) และจำแนกว่ารายงานจุดบกพร่องจะอยู่ในกลุ่มใดจะเลือกจากค่า Maximum Margin ที่น้อยที่สุด

$$y = \frac{\left|w^T x_i + b\right|}{\left\|w^T\right\|} \tag{3.9}$$

การสร้างแบบจำลองด้วยอัลกอริทึมการถดถอยโลจิสติก

การสร้างแบบจำลองด้วยอัลกอริทึมการถดถอยโลจิสติก เป็นเทคนิคการทำนายความน่าจะเป็นของข้อมูลที่มีประสิทธิภาพ และนิยมใช้ ดังเช่นในงานวิจัย [19, 21, 33] ที่พบว่าแบบจำลองที่สร้างด้วยอัลกอริทึมการถดถอยโลจิสติกมีประสิทธิภาพที่ดีกว่าอัลกอริทึมอื่นที่ได้ทำการทดสอบ การถดถอยโลจิสติกจะอาศัยตัวอิสระในการทำนายตัวแปรตาม ซึ่งในการศึกษานี้มีตัวแปรอิสระเป็นคำที่เป็นคุณลักษณะ หรือตัวแทนของเอกสาร และตัวแปรตามที่มีค่าเป็นได้เพียง 2 ค่า คือ รายงานจุดบกพร่อง และรายงานที่ไม่ใช่จุดบกพร่อง ซึ่งเรียกว่า การถดถอยแบบ Binary Logistic Regression โดยสามารถคำนวณหาความน่าจะเป็นได้ดังภาพที่ 3-16

Algorithm: Train Logistic Regression

Input: Data Set ($\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where $x_i=(x_1, x_2, x_3, \ldots, x_n)$ is instance vector and y_i is one of {-1, +1})
Iter

Output: β_i

```
Create Feature Space
foreach k ∈ Iter do
    chooseData =D
    foreach i ∈m do
        γ ← Learning Rate
        λ ← Regularization Lambda
        Select a index of chooseData idx randomly
        X ← chooseData[idx]
        del chooseData[idx]
        βi = βi .γ ∂loss(w,xi)/∂w
    end
end
```

Algorithm: TestLogistic Regression

Input: β_i
$Report_{new}$
e =2.71828

Output: C (Class of $Report_{new}$)

```
W ← Extract Tokens from Report(β,Report_new)
foreach w∈W do
    Z += βw*W
end
y = 1/(e^-z)
if y < 0.5
    return C =Bug
else
```

ภาพที่ 3-16 การสร้างแบบจำลองการจำแนกด้วยอัลกอริทึมการวิเคราะห์ถดถอยโลจิสติก

การสร้างแบบจำลองด้วยอัลกอริทึมแรนดอมฟอเรส

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึมแรนดอมฟอเรส เป็นการสร้างแบบจำลองด้วยอัลกอริทึมต้นไม้ตัดสินใจ (Decision Tree Algorithm) ขึ้นมาจำนวนมากจากการสุ่มตัวอย่างจากชุดข้อมูลการเรียนรู้ (Training) และสุ่มคคุณลักษณะ (Feature) โดยในการศึกษานี้มีชุดข้อมูลเป็นรายงานจุดบกพร่อง และคำในรายงานจุดบกพร่องเป็นคคุณลักษณะ จากนั้นนำผลลัพธ์จากแต่ละแบบจำลองมาทำการโหวด เพื่อหาผลลัพธ์ที่แท้จริง ดังภาพที่ 3-17 โดยงานวิจัย [16] พบว่าแบบจำลองสร้างด้วยอัลกอริทึมการแรนดอมฟอเรสมีประสิทธิภาพที่ดีกว่าอัลกอริทึมอื่นที่ได้ทำการทดสอบ

```
Algorithm: Pseudo code for the random forest algorithm
1. To genetate C Classifiers:
2. for i = 1 to c do
3.    Randomly sample the training data D with reolacement to
      produce Di
4.    Create a root node, Ni containing Di
5.    Call BuildTree(Ni)
6. end
7.
8. BuildTree(N)
9. if N contains instances of only one class then
10.   return
11.else
12.   Randomly select x% of the possible splitting feature in N
13.   Select the feature F with the highest information gain to spit
      on
14.   Create f child nodes of N, N1, …,Nf , where F has f possible
      values (F1, …, Ff)
15.   for i = 1 to f do
16.         Set the contents of Ni to Di, where Di is all instances
            in N that math Fi
17.         Call BuildTree(Ni)
18.   end
19. end
```

ภาพที่ 3-17 การสร้างแบบจำลองการจำแนกด้วยอัลกอริทึมแรนดอมฟอเรส

3.2.3 การประเมินประสิทธิภาพแบบจำลอง

ขั้นตอนนี้เป็นขั้นตอนการประเมินประสิทธิภาพของกระบวนการในการจำแนกรายงานจุดบกพร่องที่ศึกษาที่นำเสนอ โดยประเมินด้วยค่าความระลึก (recall: R) ความความแม่นยำ (precision: P) และค่าเอฟ (F-measure: F1) ดังตัวอย่าง

สมมุติ มีรายงานจุดบกพร่องจุดบกพร่องที่เป็นจุดบกพร่องของซอฟต์แวร์จริงจำนวน 100 รายงาน จากรายงานทั้งหมด 200 รายงาน แบบจำลองการจำแนกรายงานจุดบกพร่องทำนายว่ามี

รายงานจุดบกพร่องของซอฟต์แวร์อยู่ 83 รายงาน ซึ่งรายงานที่แบบจำลองทำนายว่าเป็นรายงานจุดบกพร่องของซอฟต์แวร์ทำนายได้ถูกต้องเพียง 59 รายงานเท่านั้น

แทนค่าลงในสมการ (2.24) ในบทที่ 2 เพื่อประเมินประสิทธิภาพค่าความระลึกได้ดังนี้

$$\mathrm{Re}call = \frac{83}{100} = 0.83 \tag{3.10}$$

แทนค่าลงในสมการ (2.25) ในบทที่ 2 เพื่อประเมินประสิทธิภาพค่าความแม่นยำได้ดังนี้

$$\mathrm{Pr}ecisoin = \frac{59}{83} = 0.71 \tag{3.11}$$

แทนค่าลงในสมการ (2.26) ในบทที่ 2 เพื่อประเมินประสิทธิภาพค่าเอฟได้ดังนี้

$$F - measure = \frac{2 \times 0.83 \times 0.71}{0.83 + 0.71} = 0.765 \tag{3.12}$$

ดังนั้นแบบจำลองที่สร้างขึ้นจะมีค่าความแม่นยำในการจำแนกรายงานจุดบกพร่องของซอฟต์แวร์จะมีค่าเท่ากับ 0.71 ค่าความระลึกเท่ากับ 0.83 และมีค่าเอฟเท่ากับ 0.765

บทที่ 4
ผลการวิจัยและการอภิปราย

ในบทนี้จะกล่าวถึงชุดข้อมูลที่ใช้ในการวิจัย และการประเมินประสิทธิภาพของกระบวนการสกัดซีนาริโอจากเอกสารความต้องการของซอฟต์แวร์ โดยแสดงรายละเอียดโดยออกเป็น 2 ส่วน ได้แก่ ชุดข้อมูลที่ใช้ในงานวิจัย และการประเมินประสิทธิภาพและการวิจารณ์ผลที่ได้

4.1 ชุดข้อมูลที่ใช้ในงานวิจัย (Dataset)

ในงานวิจัยนี้ได้ใช้ชุดข้อมูลรายงานจุดบกพร่องจำนวน 2 ชุดข้อมูลสำหรับการทดสอบกระบวนการวิจัยที่นำเสนอ ซึ่งชุดข้อมูลที่ใช้ในการทดสอบมีดังนี้

4.1.1 ชุดข้อมูลทดสอบที่ 1: Herzig's dataset

ชุดข้อมูลรายงานจุดบกพร่องนี้ถูกรวบรวมและทำการคัดแยกเอกสารที่เป็น bug และ non-bug ด้วยมือโดย Herzig [15] ซึ่งมีรายงานจุดบกพร่องทั้งสิ้น 7,401 รายงาน ซึ่งรวบรวมมาจาก HttpClient, Lucene, Jackrabbit, Rhino และ Tomcat5 ซึ่งในงานวิจัยฉบับนี้ได้ใช้ข้อมูลรายงานจุดบกพร่องจากชุดข้อมูลของ Herzig จำนวน 3,880 รายงานที่มาจาก HttpClient, Lucene, และ Jackrabbit โดยแบ่งเป็นข้อมูลในคลาส Bug จำนวน 1,940 รายงาน และคลาส Non-bug จำนวน 1,940 รายงาน ดังนั้นเมื่อทำการแบ่งกลุ่มข้อมูลแบบ 10-fold cross validation จะเป็นข้อมูลที่ใช้ในการเรียนรู้โมเดลเพื่อการจำแนกในแต่ละรอบจำนวน 3,492 รายงาน โดยแยกเป็นคลาส Bug จำนวน 1,746 รายงาน และคลาส Non-bug จำนวน 1,746 รายงาน และเป็นข้อมูลที่ใช้สำหรับการทดสอบ 388 รายงาน โดยแยกเป็นคลาส Bug จำนวน 194 รายงาน และคลาส Non-bug จำนวน 194 รายงาน

สาเหตุที่ไม่สามารถใช้ข้อมูลทั้งหมดของ Herzig ได้ เนื่องจากว่าข้อมูลของ Herzig ขาดความสมดุลในเรื่องของจำนวนข้อมูลในคลาสไม่มีความสมดุลกัน (Class imbalanced data) ดังนั้นในงานวิจัยนี้จึงทำการ resample จำนวนข้อมูลรายงานจุดบกพร่องแบบ undersampling นั่นคือการปรับจำนวนข้อมูลในคลาสที่มีจำนวนมากกว่าให้เท่ากับคลาสที่มีจำนวนข้อมูลน้อยกว่า โดยการสุ่มข้อมูลรายงานจุดบกพร่อง (Random Data) ในกรณีนี้คลาสที่มีจำนวนน้อยกว่าคือ คลาสที่เป็น Non-bug โดยมีจำนวนรายงานจุดบกพร่องอยู่ที่ 1,940 รายงาน ดังนั้นข้อมูลในคลาส Bug จึงถูกทำการสุ่มข้อมูลขึ้นมาจำนวน 1,940 รายงาน ซึ่งเป็นจำนวนที่เท่ากับจำนวนรายงานจุดบกพร่องในคลาสที่เป็น Non-bug

อย่างไรก็ตาม เนื่องจากในหลายๆ งานวิจัยที่นำเอาชุดข้อมูลของ Herzig ไปใช้ จะนิยมใช้ข้อมูลในกลุ่มนี้ [16, 19, 21] เนื่องจากข้อมูลเหล่านี้มาจาก BTS ที่ชื่อว่า Jira ที่มีการเก็บข้อมูลที่ง่าย

ต่อการนำมาใช้ ตัวอย่างข้อมูลรายงานจุดบกพร่องที่รวบรวมโดย Herzig สามารถแสดงได้ดังภาพที่ 4-1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <bug_id>
    HTTPCLIENT-85
    <bug_status>bug</bug_status>
    <Summary>Host request header does not contain port</Summary>
  </bug_id>
```

ภาพที่ 4-1 ตัวอย่างรายงานจุดบกพร่องของ Herzig ที่ใช้ในการวิจัย

4.1.2 ชุดข้อมูลทดสอบที่ 2: Firefox

ชุดข้อมูลนี้เป็นชุดข้อมูลจากระบบติดตามรายงานจุดบกพร่องที่ชื่อว่า Bugzilla โดยเป็นการรวบรวมเอารายงานจุดบกพร่องของ Firefox ที่พัฒนาโดยบริษัท Mozilla จำนวน 50,000 รายงาน ซึ่งผู้ดำเนินงานวิจัยได้เก็บรวบรวมเมื่อวันที่ 1 ตุลาคม พ.ศ. 2560 ซึ่งในงานวิจัยนี้ใช้รายงานจุดบกพร่องที่มีสถานะเป็น NEWS, ASSIGNED, VERIFIED, และ CLOSED [16, 17, 19, 21, 22] สาเหตุที่ใช้รายงานจุดบกพร่องที่มีสถานะเหล่านี้เนื่องจากเป็นรายงานจุดบกพร่องที่ได้รับการพิจารณาจาก Bug Triager เรียบร้อยแล้ว และในหลายๆ งานวิจัยที่ศึกษาเกี่ยวกับรายงานจุดบกพร่องในรูปแบบที่มีการใช้ข้อมูลจริง ๆ (Real-world data) ก็นิยมใช้ข้อมูลจาก Firefox ที่มีสถานะเหล่านั้น และสาเหตุที่นักวิจัยนิยมใช้ข้อมูลจาก Firefox เพราะเป็นโอเพนซอร์สที่มีการรายงานจุดบกพร่องเข้าไปเป็นจำนวนมากและหลากหลาย [11, 13, 22] ตัวอย่างรายงานจุดบกพร่องจาก Firefox สามารถแสดงได้ดังภาพที่ 4-2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <bug_id>
    408704
    <bug_status>Invalid</bug_status>
    <Summary>This and many more web pages can pop up while the pop-up blocker is on. The pop-up's
      that get through are not on the "allow list".</Summary>
  </bug_id>
```

ภาพที่ 4-2 ตัวอย่างรายงานจุดบกพร่องของ Firefox ที่ใช้ในการวิจัย

สำหรับชุดข้อมูล Firefox ที่รวบรวมมานั้น มีจำนวนข้อมูลทั้งสิ้น 50,000 รายงาน โดยแบ่งเป็นข้อมูลในคลาส Bug จำนวน 25,000 รายงาน และคลาส Non-bug จำนวน 25,000 รายงาน ดังนั้นเมื่อทำการแบ่งกลุ่มข้อมูลแบบ 10-fold cross validation จะเป็นข้อมูลที่ใช้ในการเรียนรู้โมเดลเพื่อการจำแนกในแต่ละรอบจำนวน 45,000 รายงาน ซึ่งแยกเป็นคลาส Bug จำนวน 22,500 รายงาน และคลาส Non-bug จำนวน 22,500 รายงาน และเป็นข้อมูลที่ใช้สำหรับการทดสอบ 5,000 รายงาน ซึ่งแยกเป็นคลาส Bug จำนวน 2,500 รายงาน และคลาส Non-bug จำนวน 2,500 รายงาน

4.2 ประเมินประสิทธิภาพและการวิจารณ์ผลที่ได้

การประเมินประสิทธิภาพในงานวิจัยนี้ ได้ใช้ตาราง Confusion matrix ในการประเมินประสิทธิภาพของแบบจำลองการจำแนกรายงานจุดบกพร่อง โดยทั่วไปในงานวิจัยด้านการจำแนกรายงานจุดบกพร่องมักจะประเมินด้วยค่า F-masure หรือ F1 [16, 17, 26, 28, 34] ซึ่งในงานวิจัยนี้ก็เช่นเดียวกัน ก็จะทำการประเมินด้วยค่า F1 เช่นกัน

ในงานวิจัยนี้ได้เปรียบเทียบคุณลักษณะที่ใช้สำหรับสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง 2 รูปแบบ คือ unigram และ Unigram + Carme Case และยังเปรียบเทียบอัลกอริทึมที่ใช้ในการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่อง 5 อัลกอริทึม ได้แก่ นาอีฟเบย์ (NB), ซัพพอร์ตเวกเตอร์แมชชีนที่ใช้ฟังก์ชันคอร์เนลแบบ Linear (SVM with Linear), ซัพพอร์ตเวกเตอร์-แมชชีนที่ใช้ฟังก์ชันคอร์เนลแบบ RBF (SVM with RBF), การวิเคราะห์การถดถอยโลจิสติก (LR) และแรนดอมฟอเรส (RF) ซึ่งเป็นอัลกอริทึมที่นำมาใช้ในการจำแนกรายงานจุดบกพร่องดังในงานวิจัย [15-17, 19, 21, 22]

ซึ่งงานวิจัยนี้ได้ทำการทดสอบการจำแนกรายงานจุดบกพร่องกับ 2 ชุดข้อมูลคือ ชุดข้อมูลของ Herzig ซึ่งจัดเป็นชุดข้อมูลมาตรฐานของรายงานจุดบกพร่องในปัจจุบัน [15] และชุดข้อมูลรายงานจุดบกพร่องของ Firefox ที่ดาวน์โหลดมาจาก Bugzilla โดยรายละเอียดของการทดสอบสามารถอธิบายได้ดังนี้

4.2.1 ผลการทดสอบด้วยชุดข้อมูลรายงานจุดบกพร่อง Herzig

การทดสอบในส่วนแรก จะเป็นการทดสอบด้วยชุดข้อมูลรายงานจุดบกพร่องของ Herzig ซึ่งจะมีการทดสอบใน 2 รูปแบบ คือ การทดสอบการจำแนกรานงานจุดบกพร่องแบบที่ไม่ใช้ขั้วคำ และการทดสอบการจำแนกรานงานจุดบกพร่องแบบที่ใช้ขั้วคำ ด้วยการแบ่งข้อมูลแบบ 10-fold cross validation จากข้อมูลทั้งหมด 5,590 รายงาน ทำให้มีจำนวนรายงานจุดบกพร่องที่ใช้ในการทดสอบต่อครั้งคือ 559 รายงาน ซึ่งผลการทดสอบสามารถแสดงรายละเอียดได้ดังนี้

1) การจำแนกรายงานจุดบกพร่องโดยไม่เพิ่มการให้น้ำหนักด้วยขั้วคำ

เป็นการทดสอบการจำแนกรายงานจุดบกพร่องในรูปแบบดั้งเดิมคือ การสกัดคุณลักษณะ การให้น้ำหนักคุณลักษณะ การสร้างโมเดลในการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอน และการนำเอาโมเดลที่ได้มาทดสอบประสิทธิภาพของการจำแนกรายงานจุดบกพร่อง ซึ่งกระบวนการดังกล่าวสอดคล้องกับงานวิจัย [16, 17, 19, 21, 22, 27]

อย่างไรก็ตาม ในการวิจัยหลักๆ ด้านการจำแนกรายงานจุดบกพร่องนั้น มักจะใช้คุณลักษณะแบบ Unigram ที่มีการทำ Porter Stemming [16, 17, 19, 21, 22, 27] เพื่อสร้างความ

เชื่อมั่นว่าคุณลักษระดังกล่าวมีความน่าเชื่อถือ ในงานวิจัยนี้จึงได้ทำการทดสอบกับการใช้คุณลักษณะในรูปแบบอื่น ๆ ได้แก่ การใช้เพียง Unigram และการใช้ Unigram ร่วมกับการทำ Lemmatization

นอกจากนี้ โดยทั่วไปแล้วในการวิจัยหลักๆ ด้านการจำแนกรายงานจุดบกพร่องนั้นมักจะใช้การให้น้ำหนักแบบ TF ซึ่งในงานวิจัยนี้ ไม่เพียงการทดสอบในการให้น้ำหนักแบบ TF ยังได้เพิ่มการทดสอบการให้น้ำหนักแบบ TF-IDF, BM25 และ MATF เพื่อประเมินว่าวิธีการให้น้ำหนักแบบใดที่มีความเหมาะสมต่องานด้านการจำแนกรายงานจุดบกพร่อง

สำหรับผลการทดสอบการจำแนกรายงานจุดบกพร่องที่ไม่เพิ่มการให้น้ำหนักด้วยขั้วคำสามารถแสดงได้ดังตารางที่ 4-1

ตารางที่ 4-1 ผลการทดสอบการจำแนกรายงานจุดบกพร่องของชุดข้อมูล Herzig ที่ไม่ใช้ขั้วคำ

Feature	Weighting	กระบวนในงานวิจัยที่นำเสนอ				
		NB	SVM (Linear)	SVM (RBF)	LR	RF
unigram	TF	0.67	0.72	0.75	0.74	0.73
	TF-IDF	0.62	0.65	0.67	0.70	0.67
	BM25	0.64	0.64	0.66	0.66	0.68
	MATF	**0.72**	**0.74**	**0.79**	**0.75**	**0.77**
Unigram+Porter Stemming	TF	0.67	0.72	0.77	0.77	0.73
	TF-IDF	0.65	0.65	0.67	0.70	0.68
	BM25	0.65	0.62	0.67	0.69	0.68
	MATF	**0.74**	**0.82**	**0.87**	**0.79**	**0.86**
Unigram+ Lemmatization	TF	0.65	0.70	0.74	0.75	0.71
	TF-IDF	0.64	0.65	0.68	0.69	0.67
	BM25	0.66	0.64	0.65	0.67	0.68
	MATF	**0.72**	**0.76**	**0.81**	**0.75**	**0.78**

จากตารางที่ 4-1 เมื่อทำการเปรียบเทียบกับงานวิจัยก่อนหน้า 2 งานวิจัยหลักทางด้านการจำแนกรายงานจุดบกพร่องที่มีการใช้ชุดข้อมูลของ Herzig อันได้แก่ งานวิจัยของ Pingclasai และคณะในปี 2013 [19] และงานวิจัย Pandey และคณะในปี 2017 [16] พบว่าให้ผลลัพธ์สอดคล้องกันในเรื่องการใช้คุณลักษณะ นั่นคือ การใช้คุณลักษณะที่เป็น Unigram ร่วมกับการทำ Porter Stemming จะให้ประสิทธิภาพที่น่าพอใจมากกว่าการใช้คุณลักษณะที่เป็น Unigram เพียงอย่างเดียว หรือการใช้คุณลักษณะที่เป็น Unigram ร่วมกับการทำ Lemmatization

นอกจากนี้การให้น้ำหนักคำด้วย TF นั้น จะให้ผลลัพธ์ที่น่าพอใจในการจำแนกข้อมูลด้วยอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนไม่ว่าจะเป็น NB, SVM ที่ใช้เคอร์เนลแบบ, SVM ที่ใช้เคอร์เนลแบบ RBF, LR, และ RF รวมทั้งยังให้ผลลัพธ์ที่ดีกว่าการให้น้ำหนักแบบ TF-IDF และ BM25 เนื่องจากในงานวิจัยที่นำเสนอนี้คำอย่างเช่น “failure”, “crash” หรือ “should” ที่ปรากฏในรายงานจำนวนมาก ซึ่งจัดว่าเป็นคุณลักษณะที่สำคัญในการจำแนกรายงานจุดบกพร่องออกเป็นรายจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่อง แต่น้ำหนักแบบ TF-IDF และ BM25 จะให้ความสำคัญของคุณลักษณะของคำดังกล่าวลดน้อยลง เพราะเป็นคำที่ปรากฏจำนวนมากในรายงานจุดบกพร่อง [17]

อย่างไรก็ตามจะเห็นว่าเมื่อมีการเปลี่ยนแปลงการให้น้ำหนักด้วยเทคนิค MATF ปรากฏว่าให้ผลลัพธ์ที่ดีและสูงกว่าการให้น้ำหนักด้วย TF, TF-IDF และ BM25 เนื่องจากว่า MATF จะพิจารณาความยาวของเอกสารร่วมด้วย ดังนั้นค่าน้ำหนักของคำที่ปรากฏในแต่ละเอกสารถึงแม้จะเป็นคำเดียวกันแต่จะได้ค่าน้ำหนักที่แตกต่างกันไป ซึ่งเป็นการปรับให้เหมาะสมกับความยาวของเอกสาร ทำให้ค่าของน้ำหนักคำที่ได้สอดคล้องกับค่าที่ควรจะเป็นในเอกสารนั้น ๆ [44]

สำหรับอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนที่ใช้ในการศึกษาได้แก่ NB, SVM ที่ใช้เคอร์เนลแบบ, SVM ที่ใช้เคอร์เนลแบบ RBF, LR, และ RF นั้นพบว่า SVM ที่ใช้เคอร์เนลแบบ RBF และ RF ให้ผลลัพธ์ที่น่าพอใจที่สุด ซึ่งก็สอดคล้องกับงานวิจัยของ Pandey และคณะในปี 2017 [16] สาเหตุที่ SVM ที่ใช้เคอร์เนลแบบ RBF เพราะการใช้เคอร์เนล RBF จะช่วยให้ข้อมูลมีมิติที่สูงขึ้น ทำให้ง่ายต่อการจำแนกข้อมูลด้วยเส้น Hyperplane ดังนั้นโอกาสที่จะทำให้เกิดความถูกต้องย่อมสูงขึ้น

ในขณะที่ RF ให้จะทำการคัดแยกข้อมูลด้วยการสร้างต้นไม้มาหลายๆ ต้น ซึ่งในงานวิจัยนี้ทดสอบการสร้างต้นไม้ที่ 30, 50, และ 100 ต้น พบว่าการสร้างต้นไม้ที่จำนวน 100 ต้นจะให้ประสิทธิภาพที่ดีที่สุด สาเหตุที่ทำให้การจำแนกรายงานจุดบกพร่องด้วย RF ให้ประสิทธิภที่น่าพอใจ เพราะ RF สร้างต้นไม้มาหลายต้นเพื่อจำแนกข้อมูล จากนั้นจะพิจารณาผลลัพธ์สุดท้ายจากการโหวตเพื่อให้ได้คำตอบที่ดีที่สุด นอกจากจะช่วยลดลดความผิดพลาดในการทำนายคลาสแล้ว ยังทำให้การจำแนกรายงานจุดบกพร่องด้วย RF มีแนวโน้มที่จะเกิดความถูกต้องด้วยเช่นกัน

2) การจำแนกรายงานจุดบกพร่องโดยเพิ่มการให้น้ำหนักด้วยขั้วคำ

เป็นการทดสอบการจำแนกรายงานจุดบกพร่องที่มีการเพิ่มค่าขั้วของคำเข้าไปในระหว่างการให้น้ำหนักคำด้วย โดยเมื่อเปรียบเทียบกับผลการทดสอบการจำแนกรายงานจุดบกพร่องในแบบที่ไม่ใช้ขั้วคำ ซึ่งผลการทดสอบสามารถแสดงได้ดังตารางที่ 4-2

ตารางที่ 4-2 ผลการทดสอบการจำแนกรายงานจุดบกพร่องของชุดข้อมูล Herzig ที่ใช้ขั้วคำ

Feature	Weighting	กระบวนการในงานวิจัยที่นำเสนอแบบไม่ใช้ขั้วคำ					กระบวนการในงานวิจัยที่นำเสนอแบบใช้ขั้วคำ				
		NB	SVM (Linear)	SVM (RBF)	LR	RF	NB	SVM (Linear)	SVM (RBF)	LR	RF
unigram	TF	0.67	0.72	0.75	0.74	0.73	0.69	0.73	0.77	0.76	0.74
	TF-IDF	0.62	0.65	0.67	0.70	0.67	0.64	0.67	0.68	0.71	0.69
	BM25	0.64	0.64	0.66	0.66	0.68	0.66	0.65	0.66	0.68	0.70
	MATF	**0.72**	**0.74**	**0.79**	**0.75**	**0.76**	**0.75**	**0.77**	**0.81**	**0.79**	**0.78**
Unigram+Porter Stemming	TF	0.67	0.72	0.77	0.77	0.73	0.70	0.75	0.81	0.80	0.83
	TF-IDF	0.65	0.65	0.67	0.70	0.68	0.66	0.67	0.69	0.70	0.70
	BM25	0.65	0.62	0.67	0.69	0.68	0.67	0.65	0.68	0.70	0.71
	MATF	**0.74**	**0.82**	**0.86**	**0.78**	**0.86**	**0.79**	**0.85**	**0.88**	**0.83**	**0.89**
Unigram+ Lemmatization	TF	0.65	0.70	0.74	0.75	0.71	0.69	0.72	0.75	0.78	0.72
	TF-IDF	0.64	0.65	0.68	0.69	0.67	0.65	0.65	0.69	0.68	0.69
	BM25	0.66	0.64	0.65	0.67	0.68	0.67	0.64	0.66	0.68	0.71
	MATF	**0.72**	**0.76**	**0.80**	**0.75**	**0.74**	**0.76**	**0.79**	**0.83**	**0.80**	**0.76**

จากตารางที่ 4-2 จะเห็นว่าผลการทดสอบค่อนข้างจะสอดคล้องกับการทดสอบในแบบที่ไม่ใช้ขั้วคำในประเด็นของคุณลักษณะและอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอน นั่นคือในประเด็นของคุณลักษณะยังพบว่าการใช้ Unigram ร่วมกับการทำ Porter Stemming ให้ผลที่น่าพอใจกว่าการใช้คุณลักษณะที่เป็น Unigram หรือ Unigram ร่วมกับการทำ Lemmatization ขณะที่อัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนที่เป็น SVM ที่ใช้งานร่วมกับเคอร์เนลฟังก์ชัน RBF และ RF ยังเป็นอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนที่ให้ประสิทธิภาพที่ดีในการจำแนกกรายงานจุดบกพร่อง

อย่างไรก็ตาม จากตารางที่ 4-2 พบว่าประสิทธิภาพของการจำแนกข้อมูลเมื่อการการใช้ขั้วคำเข้ามาช่วยนั้นดีกว่าการไม่ใช้ขั้วคำ เนื่องจากการให้ขั้วคำจะไปเพิ่มน้ำหนักความเป็นไปได้ของคำนั้น ๆ ว่าในความเป็น Bug และ Non-bug ว่าควรจะมีทิศทางไปทางใดมากกว่ากัน จึงทำให้การพิจารณาความเป็น Bug และ Non-bug นั้นง่ายขึ้น โดยการเปรียบเทียบประสิทธิในการจำแนกรายงานจุดบกพร่องแบบที่ใช้ขั้วคำเข้ามาช่วย และแบบที่ไม่ใช้ขั้วคำเข้ามาช่วย โดยกระบวนการทางสถิติ สามารถแสดงดังตารางที่ 4-3

ตารางที่ 4-3 ตารางแสดงค่าสถิติพื้นฐานของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Herzig ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Herzig ที่ให้ค่าขั้วคำ

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NB	0.6692	12	0.03753	0.01083
	polarity + NB	0.6942	12	0.04776	0.01379
Pair 2	SVM (Linear)	0.6925	12	0.06092	0.01759
	polarity + SVM (Linear)	0.7117	12	0.06780	0.01957
Pair 3	SVM (RBF)	0.7258	12	0.06868	0.01983
	polarity + SVM (RBF)	0.7425	12	0.07593	0.02192
Pair 4	LR	0.7208	12	0.04033	0.01164
	polarity + LR	0.7425	12	0.05610	0.01620
Pair 5	RF	0.7158	12	0.05485	0.01583
	polarity + RF	0.7433	12	0.06243	0.01802

จากตารางที่ 4-3 สามารถสรุปการจำรายงานจุดบกพร่องของ Herzig โดยมีค่าสถิติพื้นฐานจากค่า F1 ของการจำแนกรายงานจุดบกพร่องที่ไม่ใช้ขั้วคำ และการจำแนกรายงานจุดบกพร่องที่ใช้ขั้วคำ จากการจำแนกของทั้ง 5 อักกอริทึมได้แก่ NB, SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ Linear, SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ RBF, LR และ RF ได้ดังนี้

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม NB ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.669 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.037 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม NB ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.694 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.048

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ Linear ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.692 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.061 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ Linear ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.712 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.068

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ RBF ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.726 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.069 ซึ่งการจำแนกรายงานจุดบกพร่องด้วย SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ RBF ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.742 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.076

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม LR ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.721 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.040 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม LR ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.742 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.056

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม RF ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.716 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.055 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม RF ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.743 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.062

โดยสามารถคำนวณหาค่าสหสัมพันธ์ของค่า F1ในการจำแนกรายงานจุดบกพร่องแบบไม่ใช่ค่าขั้วคำ และการจำแนกรายงานจุดบกพร่องแบบใช้ขั้วคำของแต่อัลกอริทึมได้ดังตารางที่ 4-4

ตารางที่ 4-4 ตารางแสดงค่าสหสัมพันธ์ของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Herzig ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Herzig ที่ให้ค่าขั้วคำ

		N	Correlation	Sig.
Pair 1	NB & polarity + NB	12	0.981	0.000
Pair 2	SVM (Linear) & polarity + SVM (Linear)	12	0.989	0.000
Pair 3	SVM (RBF) & polarity + SVM (RBF)	12	0.994	0.000
Pair 4	LR & polarity + LR	12	0.975	0.000
Pair 5	RF & polarity + RF	12	0.926	0.000

จากตารางที่ 4-4 พบว่าค่าสถิติสหสัมพันธ์ของค่า F1 ในแต่ละอัลกอริทึมมีค่าดังนี้

NB	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.981
SVM(Linear)	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.989
SVM(RBF)	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.994
LR	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.975
RF	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.926

โดยที่ทุกอักกอริทึมมีค่า Sig. เท่ากับ 0.000 ซึ่งน้อยกว่า 0.01 แสดงว่าค่าตัวแปร F1 ของการจำแนกรายงานจุดบกพร่องที่ไม่ให้ค่าขั้วคำ และการจำแนกรายงานจุดบกพร่องที่ให้ค่าขั้วคำมีความสัมพันธ์กันอย่างมีนัยสำคัญทางสถิติที่ 0.01

ซึ่งในการวิเคราะห์ Paired Samples t-test จะทำการคำนวณหาค่าสถิติพื้นฐานของตัวแปรคู่ที่จะทำการทดสอบสมมติฐานในการจำแนกรายงานจุดบกพร่องของ Herzig ที่ไม่ใช่การให้ค่าขั้วคำ และการจำแนกรายงานจุดบกพร่องของ Herzig ที่ให้ค่าขั้วคำของแต่อัลกอริทึมดังแสดงในตารางที่ 4-5

ตารางที่ 4-5 ตารางแสดงค่าสถิติ t-test ของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Herzig ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Herzig ที่ให้ค่าขั้วคำ

Paired Samples Test									
		Paired Differences					t	df	*P*
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	NB - polarity + NB	0.02500	0.01314	0.00379	0.01665	0.03335	6.589	11	0.00002
Pair 2	SVM (Linear) - polarity + SVM (Linear)	0.01917	0.01165	0.00336	0.01177	0.02657	5.702	11	0.00007
Pair 3	SVM (RBF) - polarity + SVM (RBF)	0.01667	0.01073	0.00310	0.00985	0.02348	5.380	11	0.00011
Pair 4	LR - polarity + LR	0.02167	0.01899	0.00548	0.00960	0.03373	3.952	11	0.00113
Pair 5	RF - polarity + RF	0.02750	0.02379	0.00687	0.01239	0.04261	4.005	11	0.00104

จากตารางที่ 4-5 สามารถสรุปผลการทดสอบของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Herzig ในแต่ละอัลกอริทึมได้ดังตารางที่ 4-6

ตารางที่ 4-6 ตารางสรุปค่าสถิติ t-test ของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Herzig ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Herzig ที่ให้ค่าขั้วคำ

อัลกอริทึม	ค่าเฉลี่ยของความแตกต่าง	ส่วนเบี่ยงเบนมาตาฐาน	สถิติทดสอบ t-test	ค่า *P*
NB	0.025	0.013	6.589	0.00002
SVM(Linear)	0.019	0.012	5.702	0.00007
SVM(RBF)	0.017	0.011	5.380	0.00011
LR	0.022	0.019	3.952	0.00113
RF	0.027	0.024	4.005	0.00104

จากตารางที่ 4-6 พบว่าในการเปรียบการจำแนกรายงานจุดบกพร่องจากทุกอัลกอริทึมมีค่า *P*.น้อยกว่า 0.01 แสดงว่าค่าตัวแปรทั้งสองมีความสัมพันธ์กันอย่างมีนัยสำคัญทางสถิติที่ 0.01 นั้นคือการจำแนกรายงานจุดบกพร่องของ Herzig ที่ให้ค่าขั้วคำมีกระสิทธิภาพที่น่าพอใจกว่าการจำแนกรายงานจุดบกพร่องของ Herzig ที่ไม่ให้ค่าขั้วคำ

4.2.2 ผลการทดสอบด้วยชุดข้อมูลรายงานจุดบกพร่อง Firefox

การทดสอบด้วยชุดข้อมูลรายงานจุดบกพร่อง Firefox เป็นการทดสอบที่ใช้รายงานจุดบกพร่อง Firefox ในการสร้างแบบจำลอง และทดสอบการจำแนกรายงานจุดบกพร่อง โดยแบ่งผลการทดสอบออก 2 กลุ่มหลัก คือ การจำแนกรายงานจุดบกพร่องจาก Firefox โดยไม่ใช้ขั้วคำ และการจำแนกรายงานจุดบกพร่องจาก Firefox โดยใช้ขั้วคำ ซึ่งให้ผลการทดสอบดังกล่าวสามารถแสดงในตารางที่ 4-7

ตารางที่ 4-7 ผลการทดสอบการจำแนกรายงานจุดบกพร่องของชุดข้อมูล Firefox

Feature	Weighting	กระบวนการในงานวิจัยที่นำเสนอแบบไม่ใช้ขั้วคำ					กระบวนการในงานวิจัยที่นำเสนอแบบไม่ใช้ขั้วคำ				
		NB	SVM (Linear)	SVM (RBF)	LR	RF	NB	SVM (Linear)	SVM (RBF)	LR	RF
unigram	TF	0.50	0.46	0.49	0.51	0.50	0.54	0.50	0.51	0.56	0.54
	TF-IDF	0.47	0.43	0.48	0.47	0.46	0.51	0.48	0.50	0.54	0.54
	BM25	0.49	0.44	0.46	0.45	0.47	0.50	0.48	0.49	0.55	0.53
	MATF	**0.53**	**0.49**	**0.51**	**0.54**	**0.53**	**0.59**	**0.52**	**0.53**	**0.59**	**0.58**
Unigram + Porter Stemming	TF	0.54	0.50	0.54	0.52	0.54	0.58	0.54	0.58	0.57	0.59
	TF-IDF	0.50	0.45	0.50	0.49	0.52	0.56	0.50	0.53	0.55	0.55
	BM25	0.49	0.43	0.49	0.50	0.51	0.55	0.52	0.55	0.56	0.56
	MATF	**0.57**	**0.54**	**0.58**	**0.56**	**0.59**	**0.60**	**0.58**	**0.63**	**0.60**	**0.64**
Unigram + Lemmatization	TF	0.51	0.46	0.48	0.53	0.50	0.56	0.52	0.53	0.57	0.56
	TF-IDF	0.49	0.44	0.46	0.50	0.47	0.54	0.50	0.51	0.54	0.52
	BM25	0.50	0.43	0.46	0.49	0.47	0.53	0.49	0.52	0.53	0.54
	MATF	**0.54**	**0.50**	**0.52**	**0.55**	**0.53**	**0.59**	**0.54**	**0.56**	**0.59**	**0.58**

ตารางที่ 4-7 ในประเด็นของคุณลักษณะและอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอน จะเห็นว่าผลการทดสอบค่อนข้างจะสอดคล้องกับการทดสอบที่ใช้ข้อมูลของ Herzig ทั้งในแบบที่ไม่ใช้ขั้วคำและการใช้ขั้นคำ เพราะพบว่าการใช้คุณลักษระที่เป็น Unigram ร่วมกับการทำ Porter Stemming ให้ผลลัพธ์ที่น่าพอใจกว่าการใช้คุณลักษณะที่เป็น Unigram หรือ Unigram ร่วมกับการทำ Lemmatization

ในขณะที่อัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนที่เป็น SVM ที่ใช้งานร่วมกับเคอร์เนลฟังก์ชันแบบ RBF และ RF ยังคงเป็นอัลกอริทึมการเรียนรู้ของเครื่องแบบมีผู้สอนที่ให้ประสิทธิภาพที่ดีในการจำแนกกรายงานจุดบกพร่อง

ในการเปรียบเทียบประสิทธิในการจำแนกรายงานจุดบกพร่องแบบที่ใช้ขั้วคำเข้ามาช่วย และแบบที่ไม่ใช้ขั้วคำเข้ามาช่วย โดยกระบวนการทางสถิติ สามารถแสดดังตารางที่ 4-8

ตารางที่ 4-8 ตารางแสดงค่าสถิติพื้นฐานของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Firefox ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Firefox ที่ให้ค่าขั้วคำ

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NB	0.5108	12	0.02843	0.00821
	polarity + NB	0.5542	12	0.03204	0.00925
Pair 2	SVM (Linear)	0.4642	12	0.03554	0.01026
	polarity + SVM (Linear)	0.5142	12	0.02906	0.00839
Pair 3	SVM (RBF)	0.4975	12	0.03596	0.01038
	polarity + SVM (RBF)	0.5367	12	0.03892	0.01124
Pair 4	LR	0.5092	12	0.03260	0.00941
	polarity + LR	0.5625	12	0.02221	0.00641
Pair 5	RF	0.5075	12	0.03769	0.01088
	polarity + RF	0.5608	12	0.03288	0.00949

จากตารางที่ 4-8 สามารถสรุปการจำรายงานจุดบกพร่องของ Firefox โดยมีค่าสถิติพื้นฐานจากค่า F1 ของการจำแนกรายงานจุดบกพร่องที่ไม่ใช้ขั้วคำ และการจำแนกรายงานจุดบกพร่องที่ใช้ขั้วคำ จากการจำแนกของทั้ง 5 อักกอริทึมได้แก่ NB, SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ Linear, SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ RBF, LR และ RF ได้ดังนี้

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม NB ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.511 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.028 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม NB ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.554 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.032

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ Linear ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.464 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.035 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ Linear ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.514 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.029

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ RBF ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.497และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.036 ซึ่งการจำแนกรายงานจุดบกพร่องด้วย SVM ที่ใช้ฟังก์ชันคอร์เนลแบบ RBF ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.561 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.039

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม LR ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.509 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.033 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม LR ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.562 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.022

การจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม RF ที่ไม่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.507 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.038 ซึ่งการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม RF ที่ให้ค่าขั้วคำมีค่า F1 โดยเฉลี่ยที่ 0.464 และมีส่วนเบี่ยงเบนมาตรฐานที่ 0.035

โดยสามารถคำนวณหาค่าสหสัมพันธ์ของค่า F1ในการจำแนกรายงานจุดบกพร่องแบบไม่ใช่ค่าขั้วคำ และการจำแนกรายงานจุดบกพร่องแบบใช้ขั้วคำของแต่อัลกอริทึมได้ดังตารางที่ 4-9

ตารางที่ 4-9 ตารางแสดงค่าสหสัมพันธ์ของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Firefox ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Firefox ที่ให้ค่าขั้วคำ

		N	Correlation	Sig.
Pair 1	NB & polarity + NB	12	0.884	0.000
Pair 2	SVM (Linear) & polarity + SVM (Linear)	12	0.897	0.000
Pair 3	SVM (RBF) & polarity + SVM (RBF)	12	0.922	0.000
Pair 4	LR & polarity + LR	12	0.857	0.000
Pair 5	RF & polarity + RF	12	0.941	0.000

จากตารางที่ 4-9 พบว่าค่าสถิติสหสัมพันธ์ของค่า F1 ในแต่ละอัลกอริทึมมีค่าดังนี้

NB	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.884
SVM(Linear)	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.897
SVM(RBF)	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.922
LR	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.857
RF	มีค่าสถิติสหสัมพันธ์อยู่ที่	0.941

โดยที่ทุกอัลกอริทึมที่ทำการทดสอบมีค่า Sig. เท่ากับ 0.000 ซึ่งน้อยกว่า 0.01 แสดงว่าค่าตัวแปร F1 ของการจำแนกรายงานจุดบกพร่องที่ไม่ให้ค่าขั้วคำ และการจำแนกรายงานจุดบกพร่องที่ให้ค่าขั้วคำมีความสัมพันธ์กันอย่างมีนัยสำคัญทางสถิติที่ 0.01

ซึ่งในการวิเคราะห์ Paired Samples t-test จะทำการคำนวณหาค่าสถิติพื้นฐานของตัวแปรคู่ที่จะทำการทดสอบสมมติฐานในการจำแนกรายงานจุดบกพร่องของ Firefox ที่ไม่ใช่การให้ค่าขั้วคำ และการจำแนกรายงานจุดบกพร่องของ Firefox ที่ให้ค่าขั้วคำของแต่อัลกอริทึมดังแสดงในตารางที่ 4-10

ตารางที่ 4-10 ตารางแสดงค่าสถิติ t-test ของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Firefox ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Firefox ที่ให้ค่าขั้วคำ

Paired Samples Test									
		Paired Differences					t	df	*P*
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	NB - polarity + NB	0.04333	0.01497	0.00432	0.03382	0.05285	10.024	11	0.00001
Pair 2	SVM (Linear) - polarity + SVM (Linear)	0.05000	0.01595	0.00461	0.03986	0.06014	10.856	11	0.00001
Pair 3	SVM (RBF) - polarity + SVM (RBF)	0.03917	0.01505	0.00434	0.02960	0.04873	9.015	11	0.00001
Pair 4	LR - polarity + LR	0.05333	0.01775	0.00512	0.04205	0.06461	10.407	11	0.00001
Pair 5	RF - polarity + RF	0.05333	0.01303	0.00376	0.04506	0.06161	14.182	11	0.00001

จากตารางที่ 4-10 สามารถสรุปผลการทดสอบของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Firefox ในแต่ละอัลกอริทึมได้ดังตารางที่ 4-11

ตารางที่ 4-11 ตารางสรุปค่าสถิติ t-test ของค่า F1 ในการจำแนกรายงานจุดบกพร่องของ Firefox ที่ไม่ให้ค่าขั้วคำ เปรียบเทียบกับการจำแนกรายงานจุดบกพร่องของ Firefox ที่ให้ค่าขั้วคำ

อัลกอริทึม	ค่าเฉลี่ยของความแตกต่าง	ส่วนเบี่ยงเบนมาตาฐาน	สถิติทดสอบ t-test	ค่า *P*
NB	0.043	0.015	10.024	0.00001
SVM(Linear)	0.050	0.016	10.856	0.00001
SVM(RBF)	0.039	0.015	9.015	0.00001
LR	0.053	0.018	10.407	0.00001
RF	0.053	0.013	14.182	0.00001

จากตารางที่ 4-6 พบว่าในการเปรียบการจำแนกรายงานจุดบกพร่องจากทุกอัลกอริทึมมีค่า *P* เท่ากับ 0.00001 ซึ่งมีค่าน้อยกว่า 0.01 แสดงว่าค่าตัวแปรทั้งสองมีความสัมพันธ์กันอย่างมีนัยสำคัญทางสถิติที่ 0.01 นั้นคือ การจำแนกรายงานจุดบกพร่องของ Firefox ที่ให้ค่าขั้วคำมีกระสิทธิภาพที่น่าพอใจกว่าการจำแนกรายงานจุดบกพร่องของ Firefox ที่ไม่ให้ค่าขั้วคำ

สำหรับ ในงานวิจัยที่ใกล้เคียงกับงานวิจัยนี้ที่สุด เพราะใช้ข้อมูลจริงจาก Bugzilla เช่นกัน คือ งานวิจัยของ Nizamani และคณะ [22] ที่นำเสนอในปี 2018 โดยได้ผลลัพธ์ของค่า F1 ในการจำแนกรายงานจุดบกพร่องที่เป็นรายงานจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่อง Non-bug อยู่ที่ 0.72

ภายหลังจากที่ได้การทดสอบการจำแนกรายงานจุดบกพร่องของ Firefox โดยการแบ่งข้อมูลแบบ 10 fold cross validation ซึ่งให้ผลการจำแนกรายงานจุดบกพร่องออกมาเป็นที่น่าพอใจ จึงได้ทำการทดสอบกับการจำแนกรายงานจุดบกพร่อง Firefox โดยแบ่งข้อมูลแบบ holdout method ในอัตราส่วนของชุดข้อมูลรายงานจุดบกพร่องสำหรับสร้างแบบจำลองการจำแนกต่อ ชุดข้อมูลรายงานจุดบกพร่องสำหรับทดสอบเป็น 70:30 โดยสร้างแบบจำลองการจำแนกรายงานจุดบกพร่องด้วยเทคนิค Unigram ส่วนในการให้น้ำหนักด้วยเทคนิค MATF ร่วมกับให้ค่าขั้วคำ และสร้างแบบจำลองการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม ได้แก่ SVM ที่ใช้เคอร์เนลแบบ RBF และ RF และนำไปจำแนกรายงานจุดบกพร่องในชุดรายงานจุดบกพร่องทดสอบที่แบ่งไว้โดยในส่วนของชุดทดสอบของรายงานจุดบกพร่องจะประกอบด้วยการตัดคำด้วยเทคนิค Unigram ให้น้ำหนักคำด้วยเทคนิค MATF ส่วนในการให้ค่าขั้วคำจะพิจารณาจากความน่าจะเป็นของรายงานจุดบกพร่อง

นั้นๆ ด้วยอัลกอริทึมนั้นๆ ก่อนว่า ควรที่จะให้น้ำหนักขั้วคำที่เป็นรายงานจุดบกพร่อง หรือขั้วคำที่เป็นรายงานที่ไม่ใช่จุดบกพร่อง ซึ่งแสดงผลดังตารางที่ 4-12

ตารางที่ 4-12 การทดสอบการจำแนกรายงานจุดบกพร่อง Firefox โดยแบ่งข้อมูลแบบ Holdout

คุณลักษณะ	การให้น้ำหนัก	SVM with RBF	RF
Unigram + Porter stemming	MATF + Polarity	0.62	0.61

อย่างไรก็ตาม งานวิจัยของ Nizamani และคณะ [22] ก็มีความแตกต่างจากงานที่ได้นำเสนอค่อนข้างมากในหลายๆ ประเด็น นั่นคือ Nizamani และคณะ [22] ในงานวิจัยนี้ใช้ข้อมูลจาก 35 โอเพนซอร์สที่มีการจัดเก็บรายงานจุดบกพร่องผ่าน Bugzilla สาเหตุคือเพื่อให้เกิดความหลากหลายของข้อมูลที่นำมาสร้างโมเดลในการจำแนกรายงานจุดบกพร่อง เพื่อที่โมเดลที่พัฒนาขึ้นมาจะสามารถนำไปใช้ในการทำนายกลุ่มรายงานจุดบกพร่องในหลายๆ โอเพนซอร์ส โดยข้อมูลที่ใช้จะใช้ข้อมูลในมีสถานะเป็น RESOLUTION หรือ RESOLVED ซึ่งประกอบด้วยรายงานจุดบกพร่องที่เป็น INVALID คือรายงานที่ไม่ใช่จุดบกพร่อง และ FIXED คือรายงานจุดบกพร่องที่มีการแก้ไขจุดบกพร่องตามรายงานนั้นไปแล้ว ซึ่งก็คือรายงานจุดบกพร่องที่เป็นรายงานจุดบกพร่องจริง ๆ จำนวนทั้งสิ้น 41,635 รายงาน แต่ในการส่วนเนื้อหาของรายงานจุดบกพร่องที่นำมาสร้างโมเดลเพื่อการจำแนกรายงานจุดบกพร่องนั้นจะใช้ในส่วนของ Description โดยในงานวิจัยนี้ใช้คุณลักษณะเป็น Unigram ร่วมกับการทำ Lemmatization และให้น้ำหนักด้วย TF ขณะที่มีการประยุกต์อัลกอริทึมการเรียนรู้ของเครื่อง 4 อัลกอริทึมในการสร้างโมเดลเพื่อการจำแนกรายงานจุดบกพร่องคือ Multinomial NB, SVM, RF, และ LR ซึ่งผลลัพธ์ปรากฏว่าโมเดลในการจำแนกรายงานจุดบกพร่องที่สร้างจาก Multinomial NB กลับให้ผลการทำนายที่ดีที่สุด ซึ่งมีค่า F1 อยู่ที่ 0.72 เนื่องจากการใช้รายงานจุดบกพร่องที่หลากหลาย รวมถึงการใช้เนื้อหาในส่วนที่เป็น Description ทำให้มีคุณลักษณะจำนวนมาก ซึ่งลักษณะดังกล่าวจะเหมาะสมกับ Multinomial NB เพราะพื้นฐานของอัลกอริทึมนี้ต้องการคุณลักษณะที่มีจำนวนมาก โดยเฉพาะในข้อมูลที่เป็นข้อความ

จากข้างต้น งานวิจัยนี้จึงทดสอบการสร้างโมเดลการจำแนกรายงานจุดบกพร่องด้วยการใช้ส่วนของ Description เช่นเดียวกับในงานวิจัยของ Nizamani และคณะ [22] โดยใช้ข้อมูลรายงานจุดบกพร่องของ Firefox ที่ใช้เป็นชุดข้อมูลหลักในงานวิจัยนี้ ซึ่งชุดข้อมูลสำหรับเรียนรู้โมเดลสำหรับจำแนกรายงานจุดบกพร่องในคลาส Bug จำนวน 5,000 รายงาน และคลาส Non-bug จำนวน 5,000 รายงาน ขณะที่ใช้ข้อมูลทดสอบในคลาส Bug จำนวน 1,500 รายงาน และคลาส Non-bug จำนวน 1,500 รายงาน จุดประสงค์ก็เพื่อจะทำการเปรียบเทียบกระบวนการของ Nizamani และ

คณะ [22] และกระบวนการที่ใช้ในงานวิจัยนี้และพบว่าให้ประสิทธิภาพดีที่สุด คือการคุณลักษณะเป็น Unigram ร่วมกับการทำ Porter stemming และให้น้ำหนักด้วย MATF ขณะที่ประยุกต์อัลกอริทึมการเรียนรู้ของเครื่อง 3 อัลกอริทึมในการสร้างโมเดลเพื่อการจำแนกรายงานจุดบกพร่องคือ Multinomial NB, SVM ที่ใช้เคอร์เนลเป็น RBF, และ RF ซึ่งผลการทดสอบสามารถแสดงได้ดังตารางที่ 4-4

ตารางที่ 4-13 ผลการเปรียบเทียบกระบวนการวิจัยที่นำเสนอและกระบวนการวิจัยของ Nizamani และคณะ ในการจำแนกรายงานจุดบกพร่องของ Firefox ด้วยค่า F1

กระบวนการวิจัย	คุณลักษณะ	การให้น้ำหนัก	Multinomial NB	SVM with RBF	RF
Nizamani และคณะ [29]	Unigram + Lematization	TF	0.60	0.61	0.61
กระบวนการที่นำเสนอ	Unigram + Porter stemming	MATF	0.62	0.66	0.64

จากผลการทดสอบที่ปรากฏในตารางที่ 4-4 นั้น จะพบว่ากระบวนการที่นำเสนอในงานวิจัยนี้ แม้จะเปลี่ยนข้อมูลจากการใช้ Summary ไปเป็นการใช้ Description ก็ยังคงให้ผลลัพธ์ที่มีแนวโน้มที่น่าพอใจเมื่อเปรียบเทียบกับงานของ Nizamani และคณะ [29] และแสดงให้เห็นว่า การใช้ข้อมูลเพียง Summary อาจจะไม่เพียงพอต่อการวิเคราะห์การจำแนกรายงานจุดบกพร่อง ซึ่งหากใช้ข้อมูลในส่วนของ Description เพิ่มเข้ามา ก็น่าจะช่วยให้ประสิทธิภาพของการจำแนกรายงานจุดบกพร่องที่เป็น Bug และ Non-bug ได้ดีมากขึ้น

4.3 สรุปผลการทดสอบ

จากผลการทดสอบการจำแนกรายงานจุดบกพร่องในงานวิจัยนี้สามารถสรุป โดยแยกการพิจารณาจากชุดข้อมูลได้ดังนี้คือ

สำหรับชุดข้อมูลมาตรฐานของ Herzig นั้น ผลการทดลองออกมาทิศทางเดียวกันว่า การใช้ข้อมูลในส่วน Summary ของรายงานจุดบกพร่อง โดยใช้คุณลักษณะที่เป็น Unigram ร่วมกับการทำ Porter stemming นั้น น่าจะเป็นคุณลักษณะที่เหมาะสมที่สุด ขณะที่อัลกอริทึมการเรียนรู้ของเครื่องที่เหมาะสมคือ SVM ที่ใช้เคอร์เนลฟังก์ชันแบบ RBF และ RF อย่างไรก็ตาม จากการทดสอบจะเห็นว่าการให้น้ำหนักคำหรือคุณลักษณะด้วย MATF ให้ประสิทธิภาพที่ดีกว่าการให้น้ำหนักแบบ TF, TF-IDF และ BM25

สำหรับชุดข้อมูลของ Firefox นั้น ก็เป็นการใช้ข้อมูลของรายงานจุดบกพร่องที่เป็น Summary เท่านั้น ซึ่งผลการทดลองก็ยังคงออกมาทิศทางเดียวกับการทดสอบด้วยชุดข้อมูลมาตรฐานของ Herzig ซึ่งคุณลักษณะที่เหทมาะสมยังคงเป็น Unigram ร่วมกับการทำ Porter stemming ขณะที่อัลกอริทึมการเรียนรู้ของเครื่องที่เหมาะสมก็ยังคงเป็น SVM ที่ใช้เคอร์เนลฟังก์ชันแบบ RBF และ RF ส่วนการให้น้ำหนักคำหรือคุณลักษณะด้วย MATF ก็ยังคงให้ประสิทธิภาพที่ดีกว่าการให้น้ำหนักแบบ TF, TF-IDF และ BM25

นอกจากนี้ ยังได้ทำการเปรียบเทียบกับกระบวนงานวิจัยที่นำเสนอโดย Nizamani และคณะ [22] ที่มีการใช้ข้อมูลในรายงานจุดบกพร่องเป็น Description โดยใช้คุณลักษณะเป็น Unigram ร่วมกับการทำ Lemmatization และให้น้ำหนักด้วย TF ซึ่งภายหลังจากการเปรียบเทียบกับการใช้คุณลักษณะเป็น Unigram ร่วมกับการทำ Porter stemming และให้น้ำหนักด้วย MATF พบว่าการใช้คุณลักษณะเป็น Unigram ร่วมกับการทำ Porter stemming และให้น้ำหนักด้วย MATF ให้ประสิทธิภาพที่ดีกว่า โดยเฉพาะ เมื่อใช้ร่วมกับอัลกอริทึมการเรียนรู้ของเครื่องไม่ว่าจะเป็นแบบ Multinomial NB, SVM ที่ใช้เคอร์เนลฟังก์ชันแบบ RBF และ RF เพราะให้ผลลัพธ์ที่สูงกว่าผลลัพธ์ที่แสดงในงานวิจัยของ Nizamani และคณะ [22]

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

งานวิจัยยี้เป็นการวิจัยเพื่อนำเสนอการจำแนกรายงานจุดบกพร่องออกเป็น รายงานที่เป็นจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่องแบบอัตโนมัติจากข้อความในรายงานจุดบกพร่องที่เป็นภาษาธรรมชาติ ซึ่งมีรายละเอียดดังต่อไปนี้

5.1 ความสำคัญ และวัตถุประสงค์ของการวิจัย

เพื่อนำเสนอการวิจัยเพื่อพัฒนากระบวนการในการจำแนกรายงานจุดบกพร่องแบบอัตโนมัติแบบ 2 กลุ่มคือ รายงานจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่องบนพื้นฐานของการเรียนรู้แบบมีผู้สอน (Supervised Learning)

5.2 สรุปกระบวนการ

กระบวนการในการดำเนินงานวิจัยนี้ แบ่งออกเป็น 2 ส่วนหลัก ได้แก่ การเตรียมการเบื้องต้น การสร้างแบบจำลอง และจำแนกรายงานจุดบกพร่อง

ส่วนที่ 1: การเตรียมการเบื้องต้น เป็นกระบวนการให้ในสร้างคลังขั้วของคำจากรายงานจุดบกพร่อง Core โดยมีกระบวนการดังนี้

1) การกระบวนการการเตรียมข้อมูล เป็นในการเตรียมข้อมูลที่อยู่ในรูปแบบภาษาธรรมชาติให้เป็นข้อมูลที่คอมพิวเตอร์สามารถนำไปประมวลผลได้ โดยมีกระบวนการได้แก่ การตัดคำ การคัดคำคาเมลเคส การกำจัดคำหยุด การเปลี่ยนรูปคำ และการสร้างตัวแทนเอกสาร

2) การวิเคราะห์ขั้วของคำ เป็นกระบวนการในการวิเคราะห์ค่าความเป็นรายงานจุดบกพร่อง และค่าความไม่เป็นรายงานจุดบกพร่องของคำ โดยการประยุกต์การหาความน่าจะเป็นจากทฤษฎีนาอีฟเบย์ ซึ่งจะได้เป็นคลังขั้วของคำ เพื่อในไปใช้ในการให้ค่าขั้วของคำในกระบวนการต่อไป

ส่วนที่ 2: การสร้างแบบจำลอง และการจำแนกรายงานจุดบกพร่อง โดยใช้รายงานจุดบกพร่อง Firefox โดยมีกระบวนการดังนี้

1) การกระบวนการการเตรียมข้อมูล เป็นในการเตรียมข้อมูลที่อยู่ในรูปแบบภาษาธรรมชาติให้เป็นข้อมูลที่คอมพิวเตอร์สามารถนำไปประมวลผลได้ โดยมีกระบวนการได้แก่ การตัดคำ การคัดคำคาเมลเคส การกำจัดคำหยุด การเปลี่ยนรูปคำ การสร้างตัวแทนเอกสาร การให้น้ำหนักคำด้วยเทคนิค TF, TF-IDF, BM25 และ MATF และการให้น้ำหนักจากคลังขั้วของคำที่ได้จากการเตรียมการเบื้องต้นในส่วนที่ 1

2) การสร้างแบบจำลองการจำแนก และทดสอบการจำแนกรายงานจุดบกพร่อง เป็นกระบวนการในการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่องจากข้อมูลสที่ผ่านการเตรียมข้อมูลในขั้นต้น โดยใช้อัลกอริทึมแบบมีผู้สอน โดยในงานวิจัยนี้ได้ใช้อัลกอริทึมในการสร้างแบบจำลองการจำแนกรายงานจุดบกพร่องจำนวน 5 อัลกอริทึม เพื่อหาอัลกอริทึมที่เหมาะสมสำหรับการจำแนกรายงานจุดบกพร่อง ซึ่งได้แก่ นาอีฟเบย์ ซัพพอร์ตเวกเตอร์แมชชีน (Linear) ซัพพอร์ตเวกเตอร์แมชชีน (RBF) การวิเคราะห์การถดถอยโลจิสติก และแรนดอมฟอเรส

เพื่อให้ทราบว่าแบบจำลองที่สร้างมีความเหมาะสมในการจำแนกรายงานจุดบกพร่อง หรือไม่ จึงต้องทำการทดสอบการจำแนกรายงานจุดบกพร่อง เพื่อเป็นการประเมินประสิทธิภาพในการจำแนกรายงาน ซึ่งในงานวิจัยนี้ได้ใช้เทคนิคคอนฟิวชันเมตริกซ์ในการประเมินประสิทธิ ซึ่งวัดประสิทธิภาพการจำแนกรายงานจุดบกพร่องของแบบจำลองโดยใช้ค่าเอฟ ซึ่งเป็นค่าที่ใช้ในงานวิจัย [16, 17, 26, 28, 29]

5.3 สรุป และอภิปรายผล

จากกระบวนการวิจัยการจำแนกรายงานจุดบกพร่องออกเป็นรายงานจุดบกพร่อง และรายงานที่ไม่เป็นจุดบกพร่องพบว่า จากการทดสอบการการจำแนกรายงานจุดบกพร่องกับสองชุดข้อมูลคือ ข้อมูลรายงานจุดบกพร่องของ Herzig [15] และข้อมูลรายงานจุดบกพร่องของ Firefox ด้วยเทคนิคการประมวลผลทางภาษา และจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึมที่นิยมใช้ในการจำแนกรายงานจุดบกพร่องเป็นพื้นฐาน โดยในการวิจัยได้ทำการทดสอบเปรียบเทียบประสิทธิภาพการจำแนกโดยใช้ค่า F1 ซึ่งพบว่าประสิทธิในการจำแนกรายงานจุดบกพร่องของ Herzig [15] ซึ่งเป็นมาตฐานที่ผ่านการคัดเลือกข้อมูล โดย Herzig และคณะ พบว่าเทคนิค Unigram ร่วมกับเทคนิค Porter Stemming และแบบจำลองการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้เคอร์เนลฟังก์ชันแบบ RBF และ RF ให้ประสิทธิภาพสูงสุด ซึ่งสอดคล้องกับในงานวิจัย [17, 26] และในขณะเดียวกันการจำแนกรายงานจุดบกพร่องของ Firefox ซึ่งเป็นข้อมูลรายงานจุดบกพร่องจริงที่ไม่ได้ผ่านการคัดเลือก พบว่าการจำแนกรายงานจุดบกพร่องด้วยอัลกอริทึม SVM ที่ใช้เคอร์เนลฟังก์ชันแบบ RBF และ RF มีประสิทธิภาพที่ดีที่สุด ส่วนการให้น้ำหนักคำ หรือคุณลักษณะพบว่าการให้นำหนักคำด้วย MATF ก็ยังคงให้ประสิทธิภาพที่ดีกว่าการให้น้ำหนักแบบ TF, TF-IDF และ BM25 และกระบวนการที่นำเสนอในงานวิจัยนี้ที่เพิ่มการให้น้ำหนักด้วยขั้วคำการจำแนกรายงานจุดบกพร่องออกเป็นรายงานจุดบกพร่อง และรายงานที่ไม่ใช่จุดบกพร่องพบว่ามีประสิทธิภาพที่ดีขึ้นจากเดิม

5.4 ปัญหา และอุปสรรค

ปัญหาที่พบคือในการ encode และ decode ข้อความ เพื่อใช้ในการประมวลผล เนื่องจากซอฟต์แวร์มีผู้ใช้งานทั่วโลกการใช้งานทั่วโลก จึงพบภาษาที่ไม่ใช่ภาษาอังกฤษปะปนในรายงานจุดบกพร่อง

5.5 ข้อเสนอแนะ

ควรระบุให้ผู้ที่รายงานจุดบกพร่องเข้าสู่ระบบติดตามจุดบกพร่องให้รายงานจุดบกพร่องเป็นภาษาอังกฤษเท่านั้น เนื่องภาษาอังกฤษยังเป็นภาษากลางที่ทั่วโลก ซึ่งในการจำแนกรายงานจุดบกพร่องที่เป็นเพียงภาษาอังกฤษจะมีประสิทธิภาพที่ดีกว่า อีกทั้งการที่มีรายงานจุดบกพร่องที่เป็นภาษาอื่นถูกรายงานเข้าสู่ระบบอาจทำให้ผู้พัฒนาไม่สามารถทำความเข้าใจกับจุดบกพร่องที่พบได้

บรรณานุกรม

[1] Zhang J, Wang XY, Hao D, Xie B, Zhang L, Mei H. A survey on bug-report analysis. Science China Information Sciences 2015; 581-24.

[2] Zimmermann T, Premraj R, Bettenburg N, Just S, Schroter A, Weiss C. What Makes a Good Bug Report? IEEE Transactions on Software Engineering 2010; 36618-643.

[3] Kaur A, Jindal SG. Bug report collection system (BRCS): IEEE; 2017.

[4] Serrano N, Ciordia I. Bugzilla, ITracker, and other bug trackers. IEEE Software 2005; 2211-13.

[5] Bugzilla.org. Home :: Bugzilla :: bugzilla.org.

[6] Atlassian. JIRA - Issue & Project Tracking Software | Atlassian. Why JIRA? 2015;

[7] GNU.org. PSPP - GNU Project - Free Software Foundation. 2007;

[8] Perforce. Connect teams, protect IP, support rapid releases | Perforce.

[9] Cubranic D, Murphy GC. Automatic bug triage using text categorization. 16th International Conference on Software Engineering & Knowledge Engineering 2004; 92-97.

[10] Kumar Nagwani N, Verma S. CLUBAS: An Algorithm and Java Based Tool for Software Bug Classification Using Bug Attributes Similarities. Journal of Software Engineering and Applications 2012; 05436-447.

[11] Habayeb M, Miranskyy A, Murtaza SS, Buchanan L, Bener A. The Firefox temporal defect dataset: IEEE; 2015.

[12] Jeong G, Kim S, Zimmermann T. Improving bug triage with bug tossing graphs. Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering on European software engineering conference and foundations of software engineering symposium - E 2009; 111.

[13] Anvik J, Murphy GC. Reducing the effort of bug report triage. ACM Transactions on Software Engineering and Methodology 2011; 201-35.

[14] Mozilla.org. Bugzilla@Mozilla.

[15] Herzig K, Just S, Zeller A. It's not a bug, it's a feature: how misclassification impacts bug prediction. Proceedings of the 2013 International Conference on Software Engineering 2013; 392-401.

[16] Pandey N, Sanyal DK, Hudait A, Sen A. Automated classification of software issue reports using machine learning techniques: an empirical study. Innovations in Systems and Software Engineering 2017; 13279-297.

[17] Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc Y-G. Is it a bug or an enhancement? Proceedings of the 2008 conference of the center for advanced studies on collaborative research meeting of minds - CASCON '08 2008; 304.

[18] Zimmermann T, Nagappan N, Guo PJ, Murphy B. Characterizing and predicting which bugs get reopened: IEEE; 2012.

[19] Pingclasai N, Hata H, Matsumoto K-i. Classifying Bug Reports to Bugs and Other Requests Using Topic Modeling: IEEE; 2013.

[20] Zhou Y, Tong Y, Gu R, Gall H. Combining Text Mining and Data Mining for Bug Report Classification: IEEE; 2014.

[21] Pandey N, Hudait A, Sanyal DK, Sen A. Automated Classification of Issue Reports from a Software Issue Tracker. 2018; 423-430.

[22] Nizamani ZA, Liu H, Chen DM, Niu Z. Automatic approval prediction for software enhancement requests. Automated Software Engineering 2017; 1-35.

[23] Elhadad M. Natural Language Processing with Python. Computational Linguistics 2010; 36767-771.

[24] Bird S, Klein E, Loper E. Natural language processing with Python. 2009; 479.

[25] Matter D, Kuhn A, Nierstrasz O. Assigning bug reports using a vocabulary-based expertise model of developers: IEEE; 2009.

[26] Almhana R, Mkaouer W, Kessentini M, Ouni A. Recommending relevant classes for bug reports using multi-objective search. 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE); 3-7 Sept. 2016; 286-295.

[27] Du X, Zheng Z, Xiao G, Yin B. The Automatic Classification of Fault Trigger Based Bug Report: IEEE; 2017.

[28] Thanaki J. Python Natural Language Processing. 2017; 476.

[29] Perkins J, Fattohi F. Python 3 text processing with NLTK 3 cookbook : over 80 practical recipes on natural language processing techniques using Python's NLTK 3.0. 304.

[30] Porter MF. An algorithm for suffix stripping. 1980; 14[3]: 130-137. https://www.emeraldinsight.com/doi/abs/10.1108/eb046814

[31] Korenius T, Laurikkala J, J K, #228, rvelin, Juhola M. Stemming and lemmatization in the clustering of finnish text documents. ACM, 2004.

[32] Huo X, Li M, Zhou Z-H. Learning Unified Features from Natural and Programming Languages for Locating Buggy Source Code: AAAI Press; 2016.

[33] Terdchanakul P, Hata H, Phannachitta P, Matsumoto K. Bug or not? Bug Report classification using N-gram IDF: IEEE; 2017.

[34] Dommati SJ, Agrawal R, G. RMR, Kamath SS. Bug Classification: Feature Extraction and Comparison of Event Model using Na\"ive Bayes Approach. 2013;

[35] Natural Language Toolkit — NLTK 3.2.5 documentation.

[36] Bird C, Bachmann A, Rahman F, Bernstein A. LINKSTER: Enabling Efficient Manual Inspection and Annotation of Mined Data. Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering 2010; 369-370.

[37] Isa D, Lee LH, Kallimani VP, RajKumar R. Text Document Preprocessing with the Bayes Formula for Classification Using the Support Vector Machine. IEEE Transactions on Knowledge and Data Engineering 2008; 201264-1272.

[38] Maalej W, Nabil H. Bug report, feature request, or simply praise? On automatically classifying app reviews: IEEE; 2015.

[39] Zou J, Xu L, Yang M, Yan M, Yang D, Zhang X. Duplication Detection for Software Bug Reports based on Topic Model: IEEE; 2016.

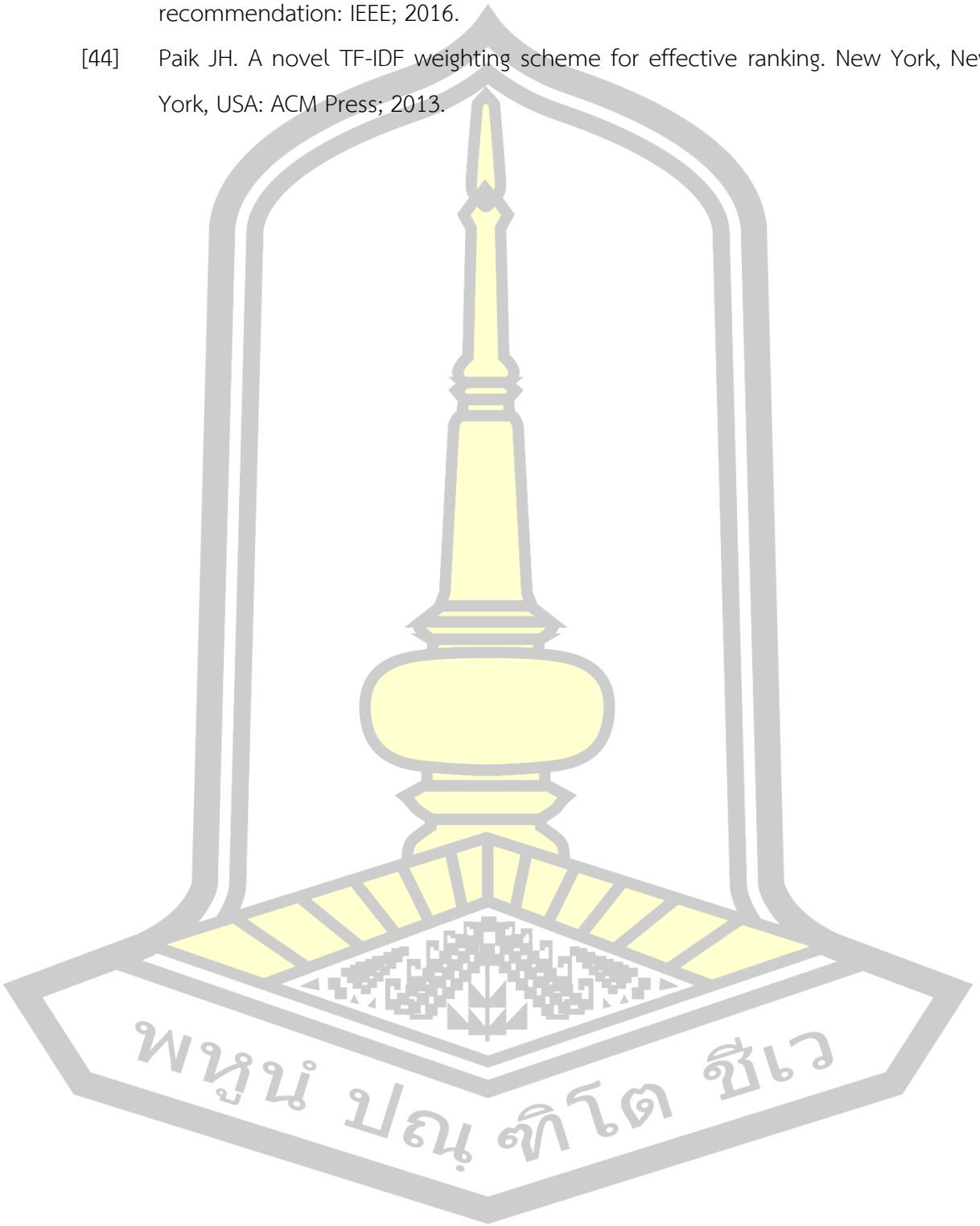
[40] Somasundaram K, Murphy GC. Automatic categorization of bug reports using latent Dirichlet allocation. New York, New York, USA: ACM Press; 2012.

[41] Yang CZ, Du HH, Wu SS, Chen IX. Duplication Detection for Software Bug Reports Based on BM25 Term Weighting; 2012.

[42] Tian Y, Lo D, Sun C. Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction: IEEE; 2012.

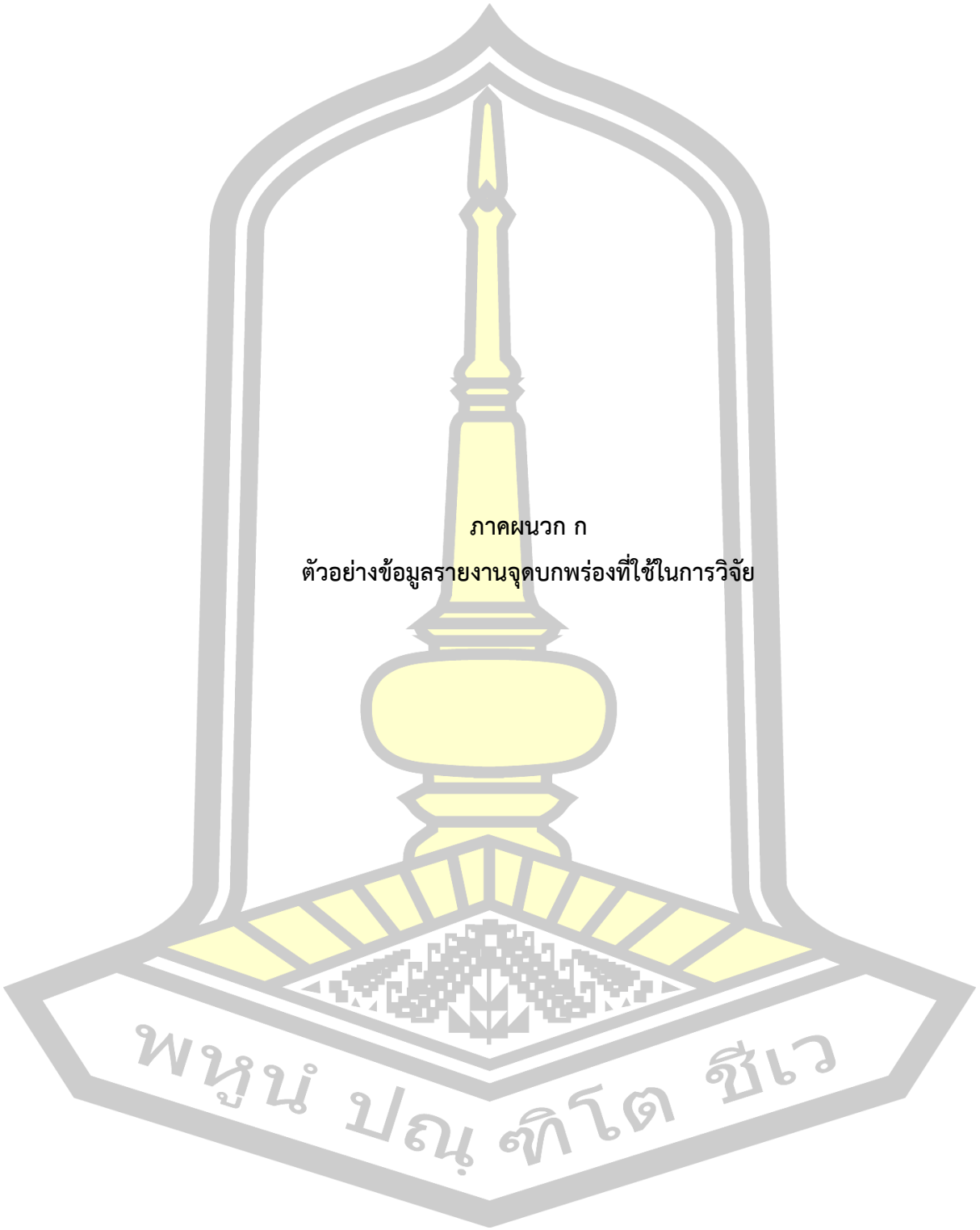
[43] Tian Y, Wijedasa D, Lo D, Le Goues C. Learning to rank for bug report assignee recommendation: IEEE; 2016.

[44] Paik JH. A novel TF-IDF weighting scheme for effective ranking. New York, New York, USA: ACM Press; 2013.



ภาคผนวก

ภาคผนวก ก
ตัวอย่างข้อมูลรายงานจุดบกพร่องที่ใช้ในการวิจัย

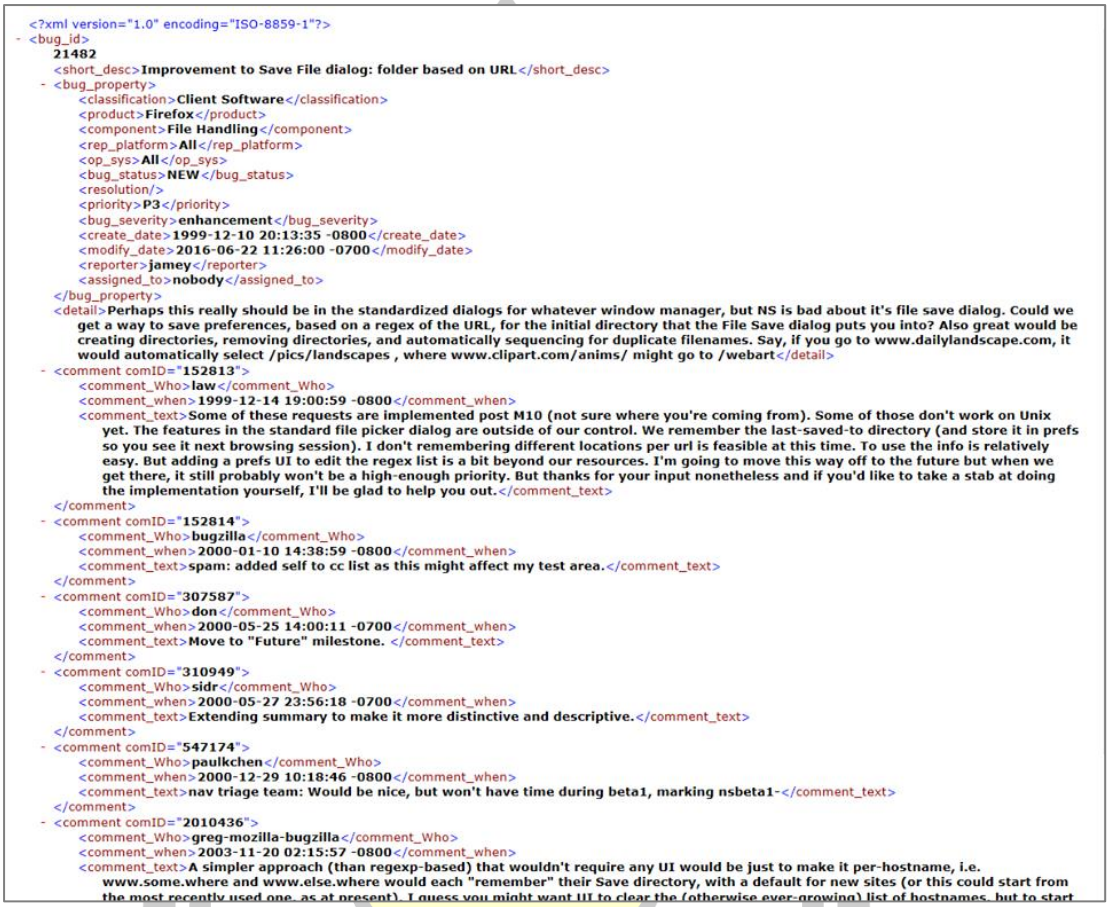


ในงานวิจัยนี้ได้ใช้ข้อมูลรายงานจุดบกพร่องจำนวน 3 ชุดข้อมูล ดังนี้

1. ชุดข้อมูลรายงานจุดบกพร่องของ Core ซึ่งแต่ละรายงานที่เก็บรวบบวมมีลักษณะ ดังนี้


```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <bug_id>
    174
    <short_desc>URL: mailto:'s -> Failed assert on exit at mkgeturl.c:5855 (net_CleanupMailtoStub)</short_desc>
  - <bug_property>
      <classification>Components</classification>
      <product>Core</product>
      <component>Networking</component>
      <rep_platform>x86</rep_platform>
      <op_sys>Linux</op_sys>
      <bug_status>VERIFIED</bug_status>
      <resolution>FIXED</resolution>
      <priority>P3</priority>
      <bug_severity>minor</bug_severity>
      <create_date>1998-04-13 10:25:18 -0700</create_date>
      <modify_date>2002-09-01 07:21:08 -0700</modify_date>
      <reporter>mcguirk</reporter>
      <assigned_to>gagan</assigned_to>
    </bug_property>
    <detail>Created by Dan McGuirk (mcguirk@indirect.com) on Monday, April 13, 1998 3:25:18 AM PDT Additional Details : Exiting the browser triggers
      an XP_ASSERT(0) on the unimplemented function net_CleanupMailtoStub.</detail>
  - <comment comID="332">
      <comment_Who>gagan</comment_Who>
      <comment_when>1998-09-08 13:57:59 -0700</comment_when>
      <comment_text>As far as I understand there isn't a resolution on mailto: urls till there is a mail solution for mozilla. In the meantime there are
        asserts that make sure we are not calling them unnecessarily. I will look into this.</comment_text>
    </comment>
  - <comment comID="333">
      <comment_Who>paulmac</comment_Who>
      <comment_when>1999-02-22 18:07:59 -0800</comment_when>
      <comment_text>setting paulmac as QA contact for all gagan's bugs (sorry for the spam)</comment_text>
    </comment>
  - <comment comID="334">
      <comment_Who>dp</comment_Who>
      <comment_when>1999-04-08 03:32:59 -0700</comment_when>
      <comment_text>Old bug. Should be fixed. I dont see this.</comment_text>
    </comment>
  - <comment comID="335">
      <comment_Who>paulmac</comment_Who>
      <comment_when>1999-04-14 20:56:59 -0700</comment_when>
      <comment_text>roger, marking verified</comment_text>
    </comment>
  - <comment comID="336">
      <comment_Who>leger</comment_Who>
      <comment_when>1999-07-20 12:13:59 -0700</comment_when>
      <comment_text>Changing all Networking Library/Browser bugs to Networking-Core component for Browser. Occasionally, Bugzilla will burp and
        cause Verified bugs to reopen when I do this in a bulk change. If this happens, I will fix. ;-)</comment_text>
    </comment>
  - <comment comID="337">
      <comment_Who>leger</comment_Who>
      <comment_when>1999-12-13 16:31:59 -0800</comment_when>
      <comment_text>Bulk move of all Networking-Core (to be deleted component) bugs to new Networking component.</comment_text>
    </comment>
</bug_id>
```

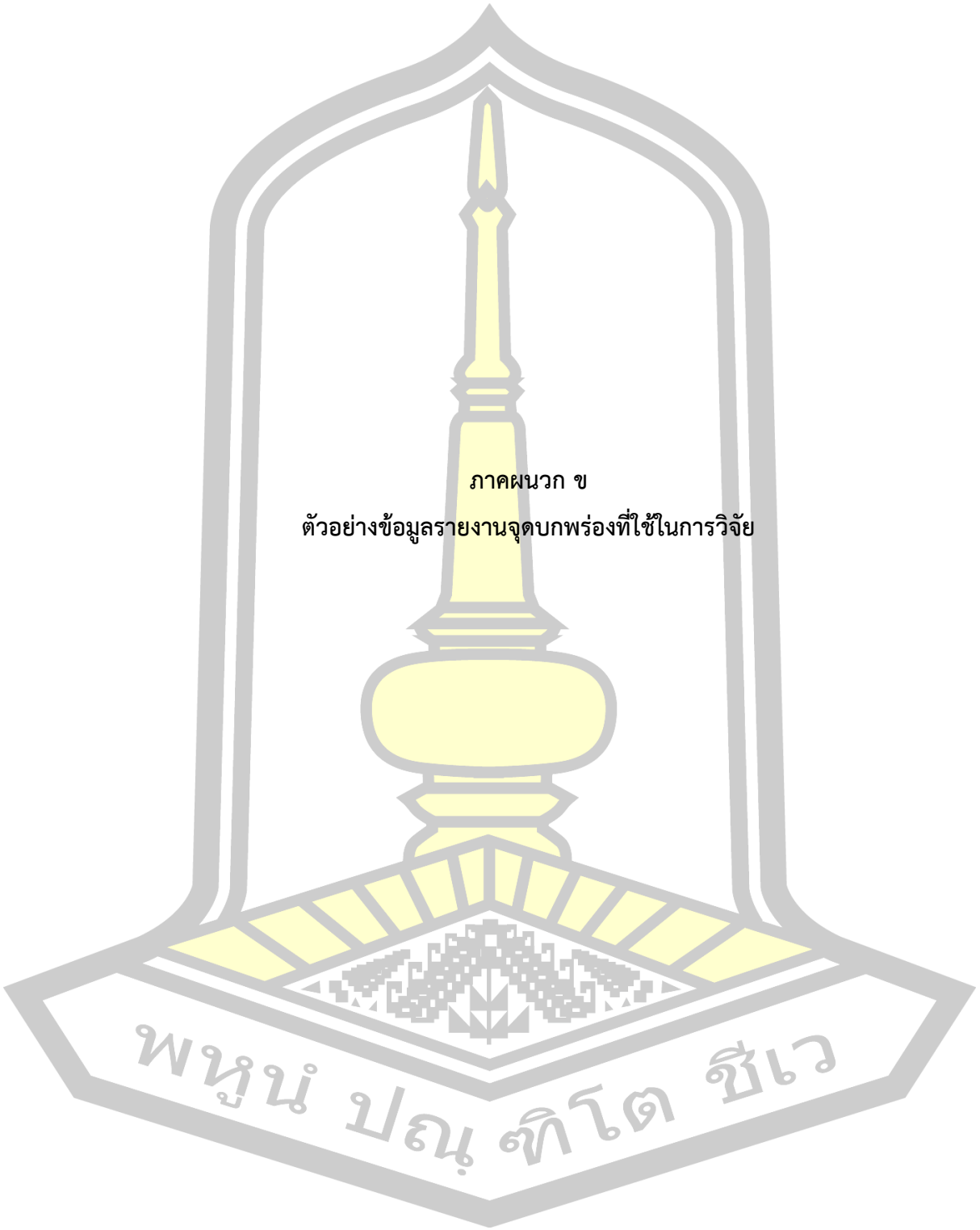
2. ชุดข้อมูลรายงานจุดบกพร่องของ Firefox ซึ่งแต่ละรายงานที่เก็บรวบบวมมีลักษณะ ดังนี้



```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <bug_id>
    21482
    <short_desc>Improvement to Save File dialog: folder based on URL</short_desc>
  - <bug_property>
      <classification>Client Software</classification>
      <product>Firefox</product>
      <component>File Handling</component>
      <rep_platform>All</rep_platform>
      <op_sys>All</op_sys>
      <bug_status>NEW</bug_status>
      <resolution/>
      <priority>P3</priority>
      <bug_severity>enhancement</bug_severity>
      <create_date>1999-12-10 20:13:35 -0800</create_date>
      <modify_date>2016-06-22 11:26:00 -0700</modify_date>
      <reporter>jamey</reporter>
      <assigned_to>nobody</assigned_to>
    </bug_property>
    <detail>Perhaps this really should be in the standardized dialogs for whatever window manager, but NS is bad about it's file save dialog. Could we
      get a way to save preferences, based on a regex of the URL, for the initial directory that the File Save dialog puts you into? Also great would be
      creating directories, removing directories, and automatically sequencing for duplicate filenames. Say, if you go to www.dailylandscape.com, it
      would automatically select /pics/landscapes , where www.clipart.com/anims/ might go to /webart</detail>
  - <comment comID="152813">
      <comment_Who>law</comment_Who>
      <comment_when>1999-12-14 19:00:59 -0800</comment_when>
      <comment_text>Some of these requests are implemented post M10 (not sure where you're coming from). Some of those don't work on Unix
        yet. The features in the standard file picker dialog are outside of our control. We remember the last-saved-to directory (and store it in prefs
        so you see it next browsing session). I don't remembering different locations per url is feasible at this time. To use the info is relatively
        easy. But adding a prefs UI to edit the regex list is a bit beyond our resources. I'm going to move this way off to the future but when we
        get there, it still probably won't be a high-enough priority. But thanks for your input nonetheless and if you'd like to take a stab at doing
        the implementation yourself, I'll be glad to help you out.</comment_text>
    </comment>
  - <comment comID="152814">
      <comment_Who>bugzilla</comment_Who>
      <comment_when>2000-01-10 14:38:59 -0800</comment_when>
      <comment_text>spam: added self to cc list as this might affect my test area.</comment_text>
    </comment>
  - <comment comID="307587">
      <comment_Who>don</comment_Who>
      <comment_when>2000-05-25 14:00:11 -0700</comment_when>
      <comment_text>Move to "Future" milestone. </comment_text>
    </comment>
  - <comment comID="310949">
      <comment_Who>sidr</comment_Who>
      <comment_when>2000-05-27 23:56:18 -0700</comment_when>
      <comment_text>Extending summary to make it more distinctive and descriptive.</comment_text>
    </comment>
  - <comment comID="547174">
      <comment_Who>paulkchen</comment_Who>
      <comment_when>2000-12-29 10:18:46 -0800</comment_when>
      <comment_text>nav triage team: Would be nice, but won't have time during beta1, marking nsbeta1-</comment_text>
    </comment>
  - <comment comID="2010436">
      <comment_Who>greg-mozilla-bugzilla</comment_Who>
      <comment_when>2003-11-20 02:15:57 -0800</comment_when>
      <comment_text>A simpler approach (than regexp-based) that wouldn't require any UI would be just to make it per-hostname, i.e.
        www.some.where and www.else.where would each "remember" their Save directory, with a default for new sites (or this could start from
        the most recently used one, as at present). I guess you might want UI to clear the (otherwise ever-growing) list of hostnames, but to start
```

3. ชุดข้อมูลรายงานจุดบกพร่องของ Herzig ซึ่งแต่ละรายงานที่เก็บรวบบวมมีลักษณะ ดังนี้


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- RSS generated by JIRA (7.6.3#76005-sha1:8a4e38d34af948780dbf52044e7aafb13a7cae58) at Sat Jul 14 09:55:28 UTC 2018 It is possible to restrict the fields
that are returned in this document by specifying the 'field' parameter in your request. For example, to request only the issue key and summary append
'field=key&field=summary' to the URL of your request. -->
- <rss version="0.92">
  - <channel>
      <title>ASF JIRA</title>
      <link>https://issues.apache.org/jira</link>
      <description>This file is an XML representation of an issue</description>
      <language>en-uk</language>
    - <build-info>
        <version>7.6.3</version>
        <build-number>76005</build-number>
        <build-date>09-01-2018</build-date>
      </build-info>
    - <item>
        <title>[HTTPCLIENT-11] null domains break Cookie.java</title>
        <link>https://issues.apache.org/jira/browse/HTTPCLIENT-11</link>
        <project id="12310360" key="HTTPCLIENT">HttpComponents HttpClient</project>
        <description><p>the domain is assumed to be non-null in a few places in Cookie.java, see matches<br/> method, for
          example</p></description>
        <environment>Operating System: other<br/> Platform: Other</environment>
        <key id="12333570">HTTPCLIENT-11</key>
        <summary>null domains break Cookie.java</summary>
        <type id="1" iconUrl="https://issues.apache.org/jira/secure/viewavatar?
          size=xsmall&avatarId=21133&avatarType=issuetype">Bug</type>
        <priority id="3" iconUrl="https://issues.apache.org/jira/images/icons/priorities/major.svg">Major</priority>
        <status id="6" iconUrl="https://issues.apache.org/jira/images/icons/statuses/closed.png" description="The issue is considered finished, the
          resolution is correct. Issues which are not closed can be reopened.">Closed</status>
        <statusCategory id="3" key="done" colorName="green"/>
        <resolution id="1">Fixed</resolution>
        <assignee username="olegk">Oleg Kalnichevski</assignee>
        <reporter username="diongillard">dion gillard</reporter>
        <labels> </labels>
        <created>Mon, 18 Feb 2002 07:28:34 +0000</created>
        <updated>Mon, 16 Feb 2009 12:29:15 +0000</updated>
        <resolved>Mon, 16 Feb 2009 12:29:15 +0000</resolved>
        <version>2.0 Alpha 1</version>
        <fixVersion>2.0 Milestone 2</fixVersion>
        <component>HttpClient (classic)</component>
        <due/>
        <votes>0</votes>
        <watches>0</watches>
      - <comments>
          <comment id="12379198" created="Mon, 18 Feb 2002 11:56:07 +0000" author="diongillard"><p>It breaks it in several places. Including
            the parse method. i'm not sure if<br/> parse should accept null domains/paths</p></comment>
          <comment id="12379199" created="Tue, 19 Feb 2002 08:19:19 +0000" author="diongillard"><p>Partial fixes have been implemented for
            <a href="https://issues.apache.org/jira/browse/HTTPCLIENT-9" title="Null paths break the compare method" class="issue-link"
            data-issue-key="HTTPCLIENT-9"><del>HTTPCLIENT-9</del></a>.</p> <p>Still to resolve whether parse or createCookieHeader
            should be allowed to pass<br/> null parameters for domain or path</p></comment>
          <comment id="12379200" created="Mon, 25 Feb 2002 23:30:27 +0000" author="rwaldhof@us.britannica.com"><p>Not a
            bug.</p></comment>
          <comment id="12379201" created="Tue, 26 Feb 2002 06:48:35 +0000" author="diongillard"><p>so which methods like parse, matches
            etc should handle nulls?</p></comment>
          <comment id="12379202" created="Tue, 26 Feb 2002 11:54:55 +0000" author="diongillard"><p>We should throw NPE's where
            appropriate.</p></comment>
          <comment id="12379203" created="Wed, 1 May 2002 14:21:27 +0000" author="rwaldhof@us.britannica.com"><p>dIon, is this bug still
            valid? Can you describe the NPEs you think we need, or <br/> provide a unit test that fails?</p></comment>
```



ภาคผนวก ข

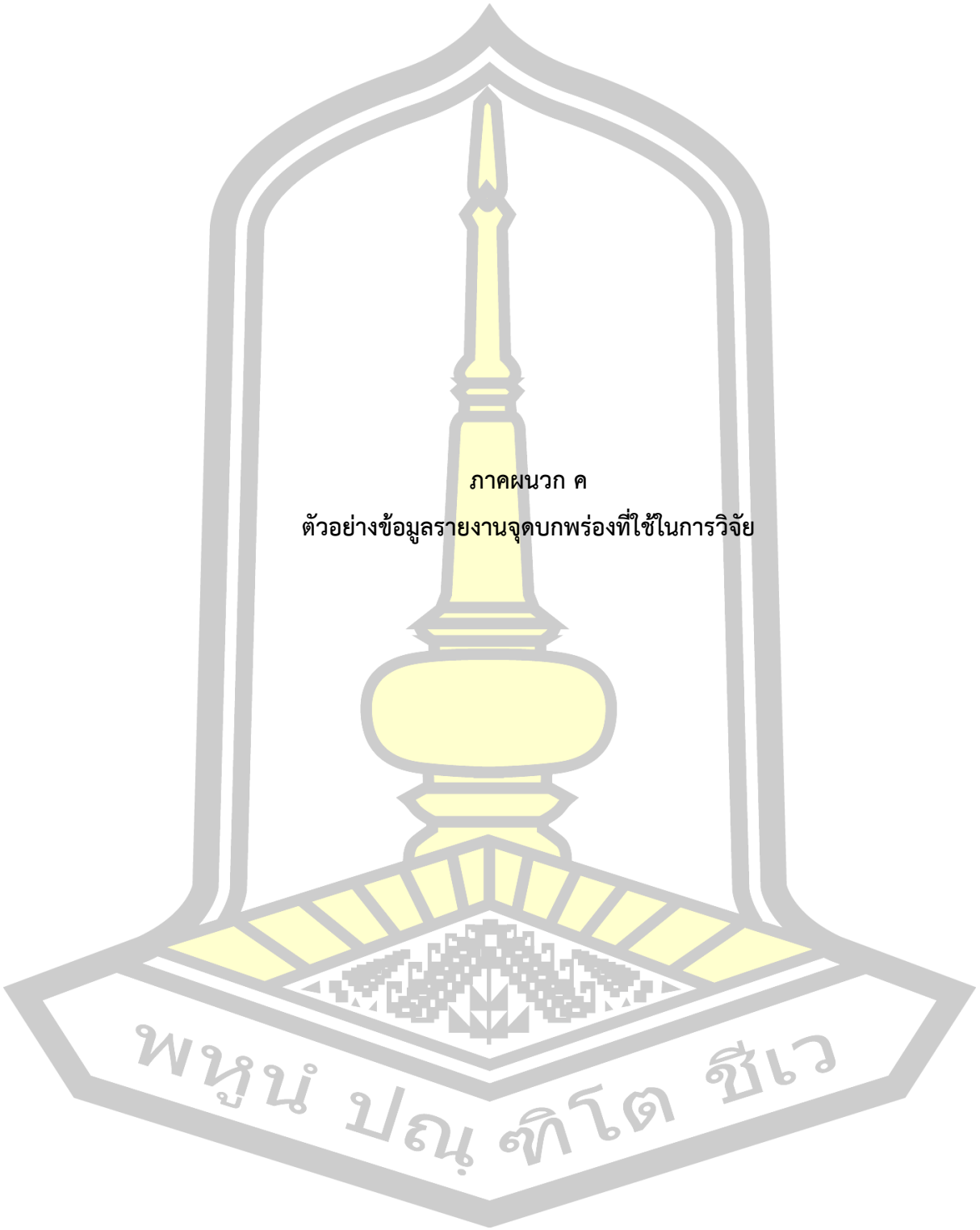
ตัวอย่างข้อมูลรายงานจุดบกพร่องที่ใช้ในการวิจัย

ตัวอย่างคลังขั้วคำของรายงานจุดบกพร่อง

```
theoretical,0.0000053006
theoretically,0.0000098440
theory,0.0000007572
thepihut,0.0000015145
ther,0.0000045434
thereafter,0.0000098440
thereby,0.0000053006
therefor,0.0000030289
therefore,0.0002241392
theres,0.0000053006
therichins,0.0000007572
theserverside,0.0000007572
thew,0.0000007572
thewebsocketprotocol,0.0000030289
thi,0.0000007572
thick,0.0000174162
thid,0.0000007572
thier,0.0000022717
thin,0.0002506422
thing,0.0000295319
things,0.0000022717
thingy,0.0000015145
think,0.0017575240
thinkable,0.0000007572
thinking,0.0000696649
thinks,0.0000590637
thinvnc,0.0000015145
third,0.0002953185
thirdly,0.0000015145
thirdparty,0.0000015145
thirty,0.0000022717
thiscall,0.0000159018
thismailbox,0.0000007572
thisp,0.0000007572
thisptr,0.0000007572
thiss,0.0000007572
thisshouldwork,0.0000007572
thisv,0.0000045434
thjqg,0.0000022717
tho,0.0000090867
thock,0.0000060578
thoese,0.0000007572
thorough,0.0000060578
thoroughly,0.0000075723
thorugh,0.0000007572
thoug,0.0000007572
though,0.0009768229
thoughs,0.0000007572
thought,0.0002309542
thouh,0.0000007572
thousand,0.0000083295
thrashing,0.0000007572
thread,0.0017575240
threaded,0.0000265029
threadep14ns,0.0000007572
threadex,0.0000355897
threading,0.0000166590
threadm,0.0000007572
threadnames,0.0000007572
threads,0.0005754926
threads0,0.0000045434
threadsafe,0.0000583065
threadsafety,0.0000007572
threadstartex,0.0000068150
threatening,0.0000015145
threatx,0.0000007572
thred,0.0000015145
three,0.0004656946
threefold,0.0000007572
threejs,0.0000022717
```

ตัวอย่างคลังขั้วคำของรายงานที่ไม่ใช่หจุดบกพร่อง

```
thins,0.0000008337
third,0.0000014387
thirdly,0.0000016674
this0,0.0000008337
thisaintnews,0.0000008337
thiscall,0.0000087538
thisistotallya,0.0000008337
thisstring,0.0000008337
thisv,0.0000025011
thjjlv,0.0000008337
thn,0.0000008337
thnx,0.0000008337
tho,0.0000010004
thoali689,0.0000008337
thock,0.0000013339
thomann,0.0000008337
thompson,0.0000008337
thorough,0.0000011116
thoroughly,0.0000041685
thos,0.0000008337
thoses,0.0000008337
thou,0.0000008337
though,0.0000014612
thought,0.0000013313
thoughts,0.0000008337
thougt,0.0000008337
thourough,0.0000008337
thouroughly,0.0000008337
thousand,0.0000013101
thows,0.0000008337
thr,0.0000008337
thrash,0.0000008337
thread,0.0000026913
thread2,0.0000008337
threaded,0.0000026527
threadex,0.0000097959
threading,0.0000036683
threads,0.0000038401
threadsafe,0.0000030569
threadstartex,0.0000015007
threatening,0.0000016674
threatens,0.0000008337
thredbo,0.0000008337
thredco,0.0000008337
three,0.0000014047
threebrain,0.0000008337
threedee,0.0000008337
threejs,0.0000012505
threfore,0.0000008337
threshold,0.0000015203
threw,0.0000057717
thrive,0.0000008337
throbber,0.0000015007
throbbing,0.0000016674
throgh,0.0000008337
throttle,0.0000016674
throttleable,0.0000016674
throttled,0.0000008337
throttling,0.0000041685
through,0.0000015525
throughout,0.0000015245
throughoutly,0.0000008337
throughput,0.0000008337
throught,0.0000008337
throw,0.0000015696
throwed,0.0000008337
throwing,0.0000020287
thrown,0.0000015562
throws,0.0000015474
thru,0.0000011910
```



ภาคผนวก ค

ตัวอย่างข้อมูลรายงานจุดบกพร่องที่ใช้ในการวิจัย

ตัวอย่างตัวแทนเอกสารที่ผ่านการให้น้ำหนัก และค่าขั้วคำที่เป็นรานงานจุดบกพร่อง

abnormal	about	above	abp	accelerate	access	accessabil	account	acrobat	adblock	add	added	adding	addon	address	adds	a
0	1.331436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0

ประวัติผู้เขียน

ชื่อ	นายบุญชู ศรีขัดเค้า
วันเกิด	วันที่ 6 มิถุนายน พ.ศ. 2534
สถานที่เกิด	อำเภอเมืองร้อยเอ็ด จังหวัดร้อยเอ็ด
สถานที่อยู่ปัจจุบัน	บ้านเลขที่ 5 ถนนรณชัยชาญยุทธ ซอย 17 ตำบลในเมือง อำเภอเมืองร้อยเอ็ด จังหวัดร้อยเอ็ด รหัสไปรษณีย์ 45000
ประวัติการศึกษา	พ.ศ. 2550 มัธยมศึกษาตอนต้น โรงเรียนพระกุมารร้อยเอ็ด อำเภอเมืองร้อยเอ็ด จังหวัดร้อยเอ็ด พ.ศ. 2553 มัธยมศึกษาตอนปลาย โรงเรียนร้อยเอ็ดวิทยาลัย อำเภอเมืองร้อยเอ็ด จังหวัดร้อยเอ็ด พ.ศ. 2558 วิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น พ.ศ. 2562 วิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม
ผลงานวิจัย	Srikudkao B, Luaphol B, Srikanjanapert N,Bheganan P,Polpinij J. Automatic Bug Report Analysis: Actual-bug and Non-bug. Proceeding of The Twenty-Fourth International Symposium on Artificial Life and Robotics 2019 (AROB 24th 2019)